BAB 10
KESIMPULAN DAN SARAN

10.1 Kesimpulan

Berdasarkan hasil perancangan sistem pengenalan material di Gudang *Raw Material*, diperoleh kesimpulan sebagai berikut.

a. Perancangan sistem pengenalan material di Gudang *Raw Material* telah selesai dikembangkan sebagai solusi untuk mengurangi persentase frekuensi kesalahan operasional per bulan. *Colly* atau kemasan material standar yang diusulkan telah disetujui dan telah diterapkan melalui fase uji coba (*trial*) secara bertahap dan terbatas pada beberapa *supplier*. Adapun sistem RFID yang meliputi sistem kerja dan sistem informasi usulan berhasil untuk diimplementasikan pada proses operasional *receiving, stock opname*, dan *outgoing*.
b. Hasil pengujian menunjukkan bahwa tujuan perancangan berhasil mengurangi kesalahan operasional di Gudang *Raw Material* sesuai penentuan *critical success factor*, yaitu berkurang 50% dari semula. Hasil pengukuran menunjukkan total persentase kesalahan operasional setelah implementasi mencapai 0,160% frekuensi transaksi material, sedangkan sebelumnya mencapai 0,444% dari frekuensi transaksi material per bulan.
c. Hasil rancangan membutuhkan investasi senilai Rp509.840.000 untuk pengadaan infrastruktur pendukung berjalannya sistem pengenalan material di Gudang *Raw Material*. Berdasarkan perhitungan, terdapat penghematan label kertas senilai Rp540.000.000, sehingga *net benefit* atau manfaat bersih mencapai Rp30.160.000 dalam periode 5 tahun pertama, kemudian dapat meningkat pada periode berikutnya hingga mencapai Rp135.000.000 tanpa pengadaan *handheld*.

10.2 Saran

Berdasarkan implementasi sistem pengenalan material di Gudang *Raw Material*, telah dilakukan evaluasi yang memunculkan saran terkait pengembangan dan perbaikan implementasi sebagai berikut.

a. Mengusulkan integrasi sistem RFID dengan sistem ERP yang digunakan, sehingga aliran data digital material dapat disatukan menjadi satu sumber basis data.
b. Mengusulkan integrasi perangkat *scanner* pada *material handling* yang digunakan
c. Menggunakan portal atau gerbang RFID pada titik masuk area *receiving*, *inventory*, dan *outgoing*, sehingga pergerakan material (keluar-masuk) akan terekam secara otomatis tanpa aktivitas pindai *handheld* manual seiring aliran material melewati portal tersebut.

DAFTAR PUSTAKA

Afiifah, K., Azzahra, F. Z., & Anggoro, A. D. (2022). Analisis teknik entity-relationship diagram dalam perancangan database sebuah literature review. *JURNAL INTECH*, *3*(1), 8–11.

Alexander, M., & Kusleika, D. (2025). *Microsoft Excel 365 bible* (2nd ed.). Wiley Professional Development (P&T).

Andía, G., Duroc, Y., & Tedjini, S. (2018). Non-linearities in passive RFID: Third harmonic concept and applications. In *Non-Linearities in Passive RFID: Third Harmonic Concept and Applications*. https://doi.org/10.1002/9781119451426

Arasyandi, M., & Bakhtiar, A. (2016). Analisa beban kerja mental dengan metode NASA TLX pada operator kargo di PT. Dharma Bandar Mandala (PT. DBM). *Industrial Engineering Online Journal*, *5*(4).

Billard, L., & Diday, E. (2019). *Clustering methodology for symbolic data*. Wiley Professional Development (P&T).

Dennis, A., Wixom, B., & Tegarden, D. (2020a). *Systems analysis and design: an object-oriented approach with UML* (6th ed.). Wiley Global Education US.

Dennis, A., Wixom, B., & Tegarden, D. (2020b). *Systems analysis and design: an object-oriented approach with UML* (6th ed.). Wiley Global Education US.

Groover, M. P. (2016). *Automation, Production Systems, and Computer-Integrated Manufacturing 4th Edition*.

Hidayat, R., Novitasari, R., & Verawati, E. (2024). Sistem Informasi Monitoring Warehouse Management Sistem Pada PT. Lisaboy. *IKRA-ITH Informatika : Jurnal Komputer Dan Informatika*, *8*. https://doi.org/10.37817/ikraith-informatika.v8i2

Hoffer, J. A. ., Ramesh, V. ., & Topi, Heikki. (2019). *Modern database management*. Pearson Education.

Kantardzic, M. (2019). *Data mining* (3rd ed.). Wiley Professional Development (P&T).

Kendall, K. E., & Kendall, J. E. (2020). *Systems Analysis and Design*.

Kolesnikov, O., Golovko, G., Yastreba, V., & Piatyntsev, Ye. (2024). Leveraging cloud technologies and serverless architecture for efficient web development: a case study from real-world application. *Системи Управління, Навігації Та Зв'язку. Збірник Наукових Праць*, *1*(75), 98–103. https://doi.org/10.26906/sunz.2024.1.098

Li, Y., Huang, Y., Li, X., Ma, J., Zhang, J., & Li, J. (2021). The influence of brightness combinations and background colour on legibility and subjective preference under negative polarity. *Ergonomics*, *65*(8).

Manzini, R., & Accorsi, R. (2024). Warehousing and Material Handling Systems for the Digital Industry: The New Challenges for the Digital Circular Economy. In *Warehousing and Material Handling Systems for the Digital Industry: The New Challenges for the Digital Circular Economy*. https://doi.org/10.1007/978-3-031-50273-6

Migliaccio, A., & Iannone, G. (2023). *Systems engineering neural networks*. Wiley Global Research (STMS).

Morana, J. (2018). Innovation in Logistics. In *Logistics* (pp. 35–45). Wiley. https://doi.org/10.1002/9781119508731.ch2

Nasri, J., Hiswara, I., & Kosasih, R. (2021). Perancangan sistem informasi persediaan barang berbasis web dengan analisa PIECES. *JRIS : Jurnal Rekayasa Informasi Swadharma*, *2*(1). https://doi.org/10.56486/jris.vol2no1.125

Nugraha, D. W., Rustam, A. H., & Thamrin Basri, M. (2026). *Peran data flow diagram dalam mengidentifikasi kebutuhan fungsional sistem informasi* (Vol. 1, Number 1).

Pathan, Z. S., Argade, P. M., Sawant, P. S., Dhere, B. B., & Pingale, V. (2025). KOA: An AI-integrated productivity intelligence platform for coding skill development and real-time performance analytics. *International Journal of Advanced Research in Science Communication and Technology*, 188. https://doi.org/10.48175/ijarsct-30631

Rahayu, N., Ma'ruf, Y., & Sunarsa, A. (2022). Creative Education of Research in Information Technology and Artificial informatics Sistem Informasi Warehouse Management System (WMS) Pada PT. Citra Banjar Abadi. *Creative Education of Research in Information Technology and Artificial Informatics*, *8*(1).

Rao, R. N. (2017). *Core python programming*. Wiley Global Education Asia.

Ringstrom, D. H. (2025). *Microsoft 365 Excel for dummies*. Wiley Professional Development (P&T).

Santalova, M. S., Pogudaeva, M. Y., Gladilina, I. P., Soklakova, I. V., & Sergeeva, S. A. (2024). OPTIMIZATION OF WAREHOUSE LOGISTICS IN THE COMPANY. In *Advanced Series in Management* (Vol. 33A). https://doi.org/10.1108/S1877-63612024000033A020

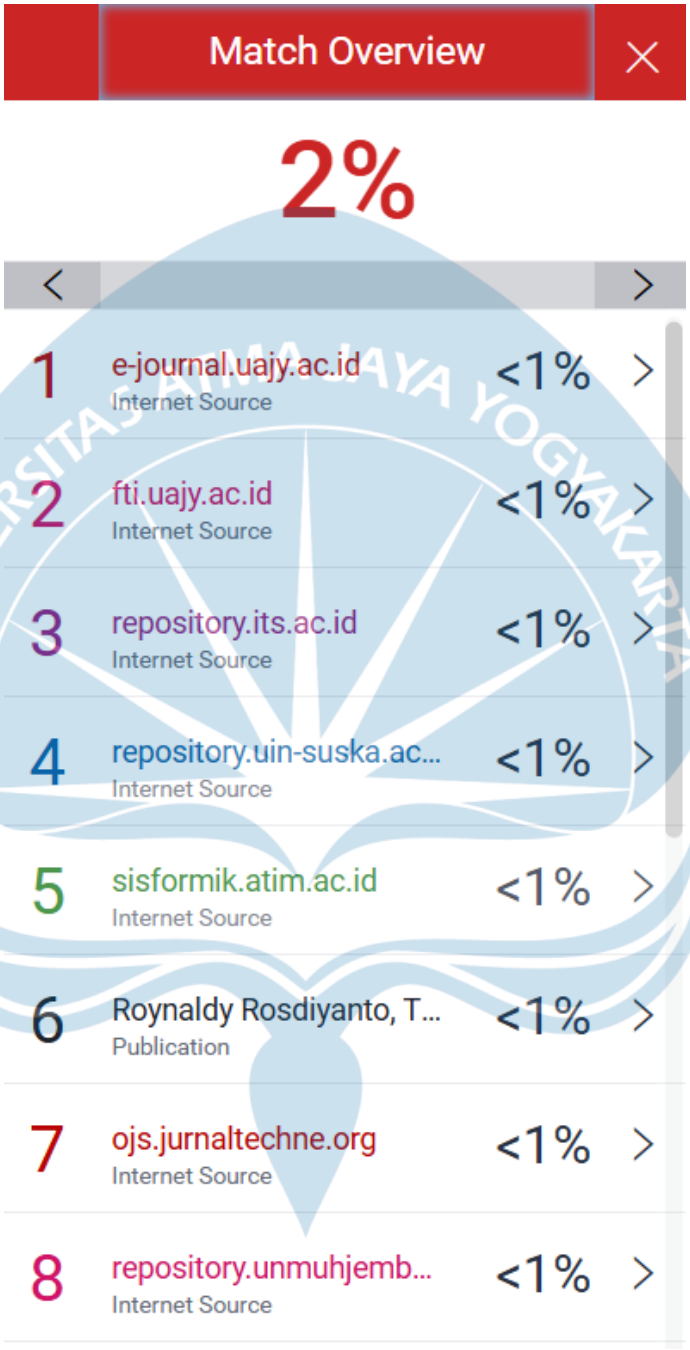
Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2019). *Data mining for business analytics concept, techniques and application in python*. Wiley Global Research (STMS).

Tampubolon, D., & Simanjuntak, P. (2023). Implementasi Aplikasi Penginputan Inventory Menggunakan Metode Agile Berbasis Web. *Comasie*, *9*(5), 606.

Turban, E., Pollard, C., & Wood, G. (2025). *Information technology for management: navigating the digital divide to enhance local and global performance, growth, and sustainability* (13th ed.). Wiley Global Education US.

LAMPIRAN

Lampiran 1 Hasil Pemeriksaan Kesamaan (*Similarity*) dari Turnitin



Lampiran 2 Tangkapan Layar Catatan dari Calon Dosen Pembimbing 1



UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 3 Surat Pernyataan Semua Anggota Kelompok

SURAT PERNYATAAN PELAKSANAAN TUGAS AKHIR
SEBAGAI *CAPSTONE DESIGN PROJECT* SECARA BERKELOMPOK

Kami yang bertanda tangan di bawah ini:

No	Nama Mahasiswa	NPM
1	Silvester Pierre Marendra Putra	220611778
2	Ni Nyoman Bintang Anjani Putri Suparta	220611952
3	Sephia Josephine Feoh	220612000

Dengan ini menyatakan bahwa kami akan melaksanakan **Tugas Akhir dalam bentuk *Capstone Design Project*** secara **berkelompok**, dengan judul/topik yang diusulkan:

Perancangan Sistem Identifikasi dan Pelacakan Material untuk Mengurangi Kesalahan Operasional di Gudang *Raw Material*

Adapun pernyataan komitmen kami adalah sebagai berikut:

1. Setiap anggota kelompok menjalankan Tugas Akhir tersebut secara kolaboratif, profesional, dan bertanggung jawab sesuai kaidah akademik yang berlaku di Departemen Teknik Industri FTI UAJY.
2. Setiap anggota kelompok bertanggung jawab untuk memberikan kontribusi yang proporsional dan melaksanakan tugas sesuai pembagian peran yang disepakati.
3. Setiap anggota kelompok menjalankan seluruh proses sesuai ketentuan yang berlaku di Departemen Teknik Industri FTI UAJY dan menjunjung tinggi integritas akademik.
4. Apabila di kemudian hari terdapat pelanggaran terhadap ketentuan akademik atau ketidaksesuaian dalam pelaksanaan tugas, kami bersedia menerima sangsi sesuai peraturan yang ditetapkan oleh Departemen Teknik Industri FTI UAJY.

Demikian surat pernyataan ini dibuat dengan sebenar-benarnya untuk dipergunakan sebagaimana mestinya.

Yogyakarta, 19 Desember 2025

Tanda Tangan

Silvester Pierre Marendra Putra
NPM: 220611778



Ni Nyoman Bintang Anjani Putri Suparta
NPM: 220611952



Sephia Josephine Feoh
NPM: 220612000



Lampiran 4 Instrumen Pengambilan Data

Pengambilan Data Waktu Kerja

DATA WAKTU KERJA

Nama Pekerja :
1
2
3
4
5
6
7

Tanggal :
Jam :
Bagian : Receiving

Tugas : Bongkar
Diamati Oleh :

No	Aktivitas	Jam Mulai	Jam Selesai	Durasi (Menit)	Jam Truk Sudah Kosong	Durasi Bongkar Truk (Menit)	Kategori (Bekerja/Menganggur)	Output					Pembagian Kerja	Keterangan
								Colly	Lainnya			Pallet		
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														

Pengambilan Data Waktu *Idle*

DATA WAKTU

Nama Pekerja :
1 [redacted]
2
3
4
5
6
7

Tanggal : Kamis 18 September 2025
Jam : 08.00-16.00
Bagian : Receiving

Tugas : Bongkar
Diamati Oleh :

No	Aktivitas	Jam Mulai	Jam Selesai	Durasi (Menit)	Jam Truk Sudah Kosong	Durasi Bongkar Truk (Menit)	Kategori (Bekerja/Menganggur)	Output					Pembagian Kerja	Keterangan
								Colly	Lainnya			Pallet		
1	Menunggu Surat Jalan	8:05	8:21	16			Menganggur							
2	Bongkar	8:21	8:58	37	8:35	14	Bekerja	8	120 Sak	75 Can		9	4 operator bongkar, 2 *pallet mover*	Truk PT XYZ
3	*Idle*	8:58	9:31	33			Menganggur							
4	Menunggu Surat Jalan	9:31	10:03	32			Menganggur							
5	Bongkar	10:03	10:22	19	10:14	11	Bekerja	117				7	4 operator bongkar, 1 *pallet mover*	Pickup
6	*Idle*	10:22	11:00	38			Menganggur							
7	Menunggu Surat Jalan	11:00	11:04	4			Menganggur							
8	Bongkar	11:04	11:42	38	11:24	20	Bekerja	468				14	5 operator bongkar, 1 *pallet mover*	Truk Kontainer
9	*Idle*	11:42	12:00	18			Menganggur							
10	Istirahat	12:00	13:05	65			-							
11	*Idle*	13:05	13:22	17			Menganggur							
12	Menunggu Surat Jalan	13:22	13:40	18			Menganggur							
13	Bongkar	13:40	14:13	33	14:00	20	Bekerja	154				26	3 operator bongkar, 2 *pallet mover*	Truk Kontainer
14	Bongkar	14:13	15:15	62	14:50	37	Bekerja	821	1 Peti	27 Pack	15 Ball	30	6 operator bongkar, 2 *pallet mover*	Truk PT XYZ
15	*Idle*	15:15	16:00	45			Menganggur							

Pengambilan Data Dimensi Kubikasi Kemasan Material

Material	Material Description	Sloc	Sbin	Tipe Packing Terkecil	QTY Packing Terkecil	Unit of Measure ([redacted]i SAP	1 Packing Terbesar Bisa diisi Berapa Packing Terkecil	Panjang Packing Terbesar (cm)	Lebar Packing Terbesar (cm)	Tinggi Packing Terbesar (cm)	Panjang Packing Terkecil (cm)	Lebar Packing Terkecil (cm)	Tinggi Packing Terkecil (cm)	Berat Packing Terkecil (Kg)	1 Palet Bisa diisi Berapa Packing Terbesar	Maximal Tumpukan Packing Besar dalam 1 Palet	Note

Pengecekan Ukuran Fisik Kubikasi Kemasan Material

Material	Material Description	Nama Vendor	Tipe Packing Terkecil	QTY Packing Terkecil	Unit of Measu[redacted] (UM) di SAP	1 Packing Terbesar Bisa diisi Berapa Packing Terkecil	Panjang Packing Terbesar (cm)	Lebar Packing Terbesar (cm)	Tinggi Packing Terbesar (cm)	Volume Packing Terbesar	Panjang Packing Terkecil (cm)	Lebar Packing Terkecil (cm)	Tinggi Packing Terkecil (cm)	Volume Packing Terkecil	TES Dimensi Pack Besar dan Pack Kecil

Lampiran 5 Hasil *Cluster* Simplifikasi *Colly* Material

Cluster	Count	Max Panjang Pack (cm)	Max Lebar Pack (cm)	Max Tinggi Pack (cm)	*Silhouette Score Cluster*
0	51	53	34	34	0.524264023
1	9	78	59	46.5	0.160260589
2	14	30.5	26	18	0.289925887
3	22	60	45	40	0.579258765
4	6	65	56	87	0.951477963
5	4	116	20	23	0.908775749
6	12	61.5	46	21	0.450969835
7	7	117	71	92	0.787466345
8	23	50	35	24	0.228849832
9	11	55	39	56	0.739451882
10	1	92	44	54	0
11	15	32	33	30	0.597529492
12	1	94	78	3	0
13	2	11	9.5	15	0.414734273
14	5	98	33	28	0.43911169
15	5	73	57	61	0.450036119
16	9	36	27	21	0.353546289
17	4	58	47	29	0.676060726
18	6	55	23	21	0.586710337
19	6	55	35	46	0.383202825
20	21	42	33	42	0.542553723
21	15	53	39	33	0.482317922
22	14	55	52	45	0.39903588
23	2	129	30	20	1
24	59	55	31	28	0.606382107
25	3	118	35.5	36	0.916174384
26	8	93	57	92	0.915470166
27	15	63	47	58	0.746892862
28	9	42	42	28	0.470109477
29	22	59	41	38	0.246390129
30	7	75	40.5	55	0.31164433
31	1	84	54	21	0
32	20	58	55	30	0.58306912
33	39	43	30	24	0.458871919
34	6	45	38	56	0.59732215
35	27	55	35	28	0.593377832
36	12	31	27	32	0.500339182
37	2	66	29	40	0.558791323
38	2	57	57	7	0.578700776
39	3	90	70	48	0.363455014
40	2	92	85	92	0.380902432
41	1	5	50	40	0

Lampiran 5 Hasil *Cluster* Simplifikasi *Colly* Material

Cluster	Count	Max Panjang Pack (cm)	Max Lebar Pack (cm)	Max Tinggi Pack (cm)	*Silhouette Score Cluster*
42	14	45	27	37	0.568764602
43	14	67	49	36	0.346124296
44	2	38	32	82	0.63660147
45	6	55.5	43	46	0.312434089
46	11	67	65	64	0.637193039
47	1	111	45.5	116	0
48	1	117	46	38	0
49	6	93	39	63	0.505829749
50	1	76	33	21.5	0
51	7	109	76	87	1
52	18	36	22	18	0.304557528
53	9	48	33	18	0.244232249
54	3	31	31	48	1
55	2	103	22	40	1
56	7	70	50.5	47	0.220104942
57	3	33	46	15	0.999999954
58	2	82	29	7	1
59	2	114	85	25	0.391697498
60	3	23	14.5	10	0.418819497
61	4	80	50	65	0.559107463
62	2	87	42	40	0.680403098
63	10	54	27	17	0.507406715
64	2	117	47	15	1
65	18	54	39	19.5	0.410882513
66	19	53	28	35	0.559014284
67	6	68	50	13.5	0.603509675
68	7	50	17	20	0.42433125
69	2	98	64	8	0.329373071
70	7	41.5	31	11	0.182507766
71	4	102	26	21.5	0.537771962
72	1	5	30	21	0
73	22	58	47	37	0.519031798
74	8	83	36	58	0.556485401
75	2	75	68	25	1
76	8	53	50	54	0.484091704
77	10	49	41	27	0.417606751
78	3	22	22	30	0.056817103
79	21	54	40	24	0.301929922
80	6	85	27	27	0.932208942
81	48	45	35	25	0.491606644
82	1	82	40	87	0
83	7	60.5	52	39.5	0.360663724

Lampiran 5 Hasil *Cluster* Simplifikasi *Colly* Material

Cluster	Count	Max Panjang Pack (cm)	Max Lebar Pack (cm)	Max Tinggi Pack (cm)	*Silhouette Score Cluster*
84	15	27	26	22	0.695705833
85	2	77	38	31	0.267063233
86	19	47	37	20	0.391652071
87	9	39	21	25	0.508130196
88	33	38	28	16	0.559825493
89	7	48	48	42	0.287265743
90	3	78	69	63	0.25845071
91	2	22	21	52	1
92	3	16	3	4	0.63216627
93	3	99	41	73.5	0.887194444
94	18	61	42	28	0.304169982
95	4	73	24	21	0.453127861
96	9	70	43.5	41.5	0.472084775
97	2	60	50	99	0.828727217
98	5	23	17	18	0.311924209
99	1	26	44	77	0
100	9	65	36	51.5	0.516362792
101	5	63	36	19	0.256614459
102	19	37	28	35	0.515458808
103	8	32	31.5	25	0.617486285
104	9	45	41	40	0.397364505
105	7	84	55	50	0.77459993
106	3	29	21	39	0.585872806
107	12	62	31	26	0.337193038
108	13	60	37	34	0.405698624
109	2	102	56	62	0.377204946
110	3	41	40	15	0.549790568
111	7	60	57	52	0.459629551
112	1	77	63	5	0
113	1	63	46	77	0
114	3	102	37	37	0.544432805
115	3	89	59	36	0.420332413
116	28	51	32	28	0.261961529
117	9	67	44.5	25	0.432293847
118	23	55	40	39	0.275792491
119	7	60	35	40	0.77205128
120	9	53	44	35	0.265859458
121	5	61	54	18	0.322914466
122	5	92	37	37	0.005859187
123	2	89	41	31	0.173368099
124	5	63	36	63	0.450181826
125	1	110	75	66	0

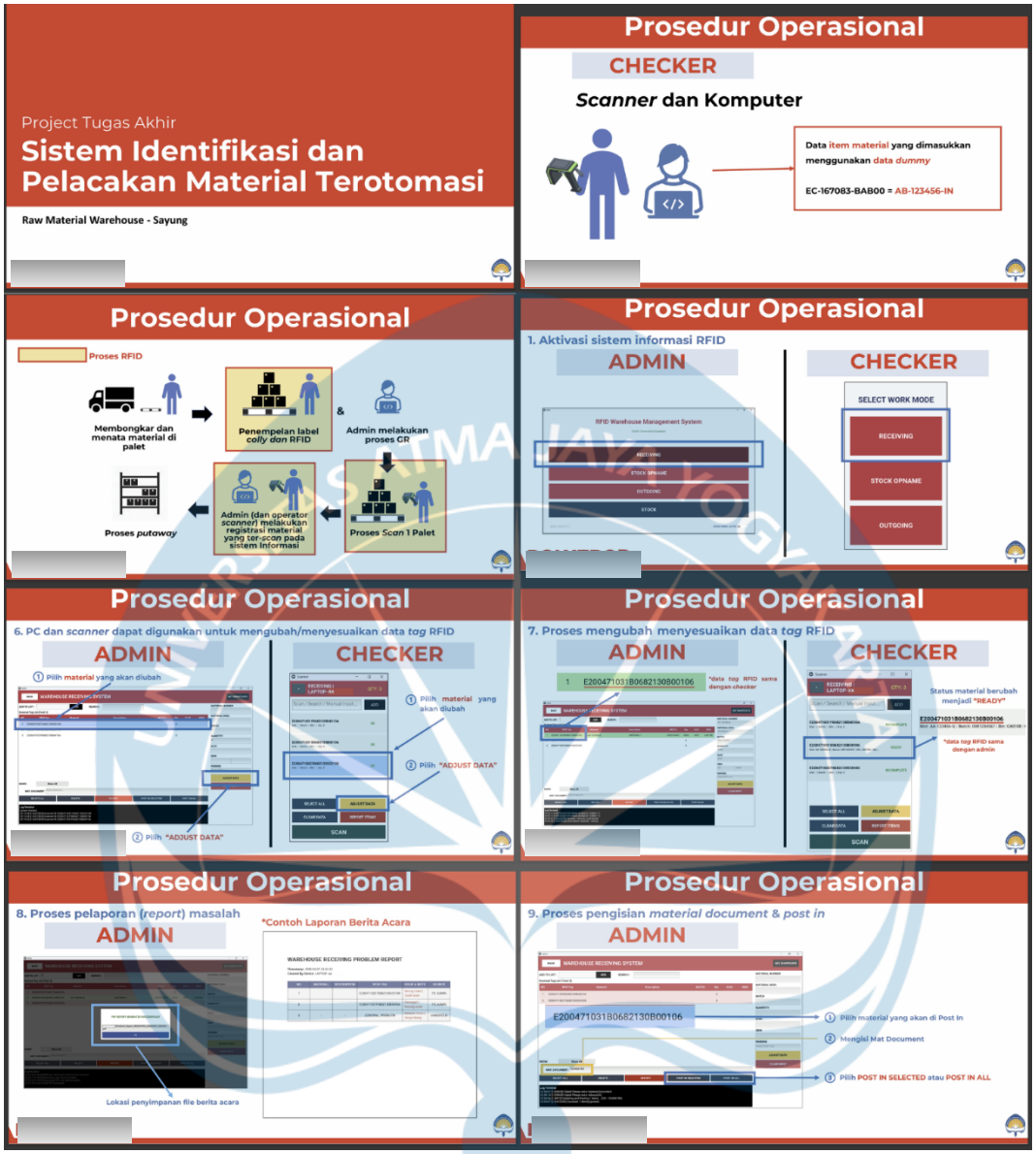
Lampiran 5 Hasil *Cluster* Simplifikasi *Colly* Material

Cluster	Count	Max Panjang Pack (cm)	Max Lebar Pack (cm)	Max Tinggi Pack (cm)	*Silhouette Score Cluster*
126	2	54	47	10	0.526227552
127	3	69	31	30	0.505093914
128	5	18	13	8	0.21106895
129	20	48	37	30	0.363397361
130	1	83	76.5	67	0
131	2	60	11	14	0.047445544
132	3	50	22	42	0.515585581
133	1	4	40	20	0
134	2	67	61	40	0.872185543
135	7	60	40	44	0.391097102
136	8	35	35	36	0.331464728
137	10	38	28	27	0.342589389
138	1	94	78	19	0
139	5	43	51	24	0.240341487
140	1	102	51	43	0
141	2	92	62	37	1
142	20	58	49	41	0.469472443
143	8	28	23	14	0.200126521
144	1	44	45	84	0
145	4	57	25.5	33	0.709736624
146	6	69	37	20	0.903480677
147	2	70	34	5	0.26718059
148	8	54	53	36	0.593329258
149	12	50	48	49	0.326772017
150	3	28	11	14	-0.033715316
151	1	6.45	16	26.5	0
143	8	28	23	14	0.200126521
144	1	44	45	84	0
145	4	57	25.5	33	0.709736624
146	6	69	37	20	0.903480677
147	2	70	34	5	0.26718059
148	8	54	53	36	0.593329258
149	12	50	48	49	0.326772017
150	3	28	11	14	-0.033715316
151	1	6.45	16	26.5	0
152	8	54	35	40	0.102850051
153	10	66	56	40	0.271231674
154	2	128	38	38	0.909687178
155	4	58	56	90	1
156	1	53	71	53	0
157	3	82	49	28.5	0.272176259
158	9	46.7	38	39	0.251832116

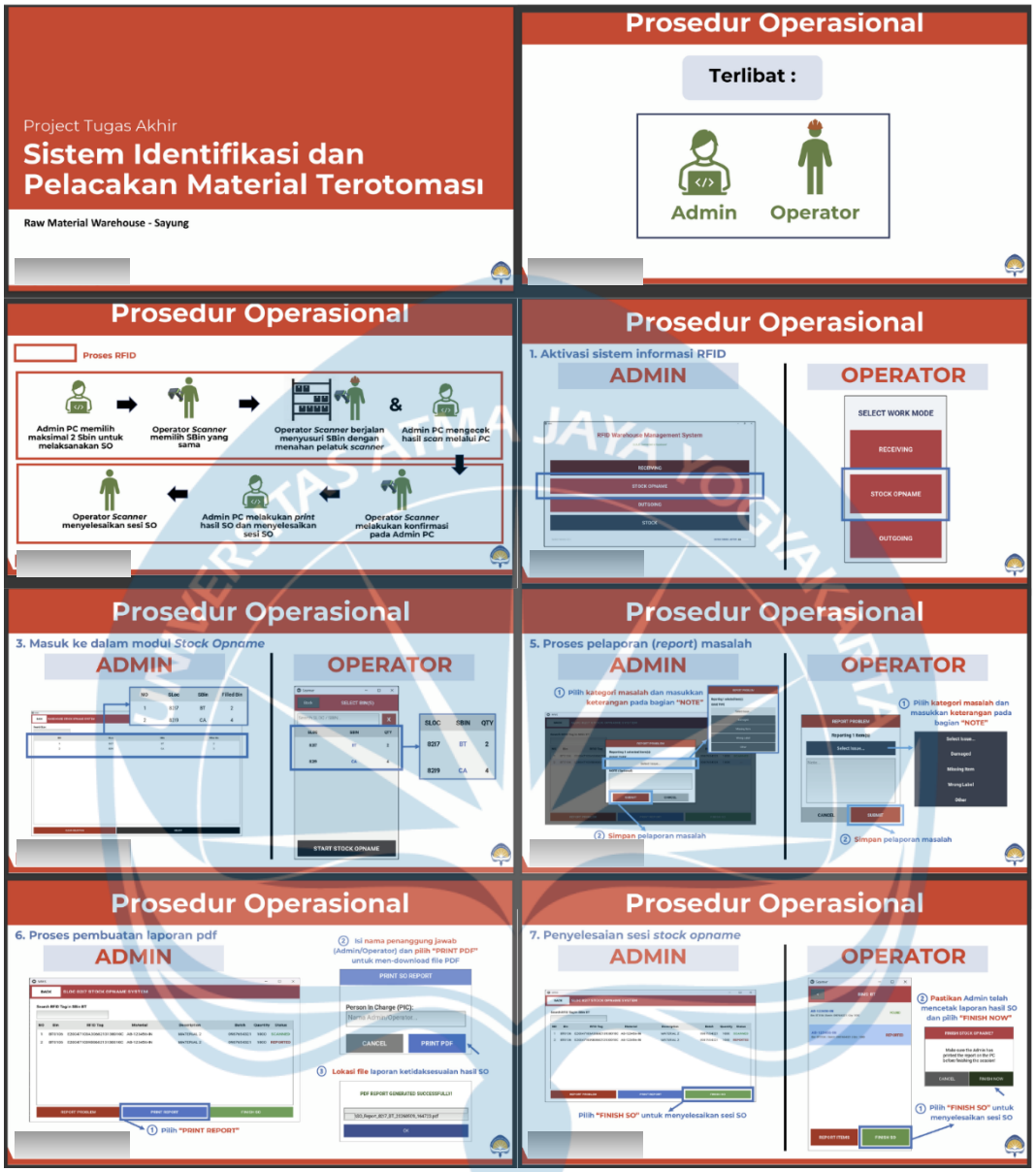
Lampiran 5 Hasil *Cluster* Simplifikasi *Colly* Material

Cluster	Count	Max Panjang Pack (cm)	Max Lebar Pack (cm)	Max Tinggi Pack (cm)	*Silhouette Score Cluster*
159	6	64	58	34	0.30697869
160	9	56	49	32	0.65812672
161	1	45	47	62	0
162	4	96	31	16	0.377147263
163	6	59	48	49	0.280734077
164	1	76	61	70	0
165	17	56	41	42	0.455617477
166	8	46	46	33	0.427535125
167	1	93	16	36	0
168	2	72	40	63	0.75065662
169	20	43	32.5	23	0.222772776
170	2	56	57	23	0.438197561
171	1	95	46	34	0
172	1	122	50	47	0
173	5	38	38	25	0.351702772
174	4	120	25	26	0.413095811
175	12	35	31	36	0.231536779
176	2	116	23	16	0.759662636
177	1	85	62	16	0
178	5	44.5	24	33	0.454393073
179	7	68	30	36	0.631884816
180	2	76	50	53	0.584269491
181	6	43	25	15	0.213915577
182	10	60	36	26	0.510600376
183	17	57	31	23	0.141281214
184	8	45	34.5	31	0.180071972
185	5	41	29	42	0.565128123
186	14	45	30	30	0.279632802
187	1	86	56	63	0
188	5	56	30	31	0.588312592

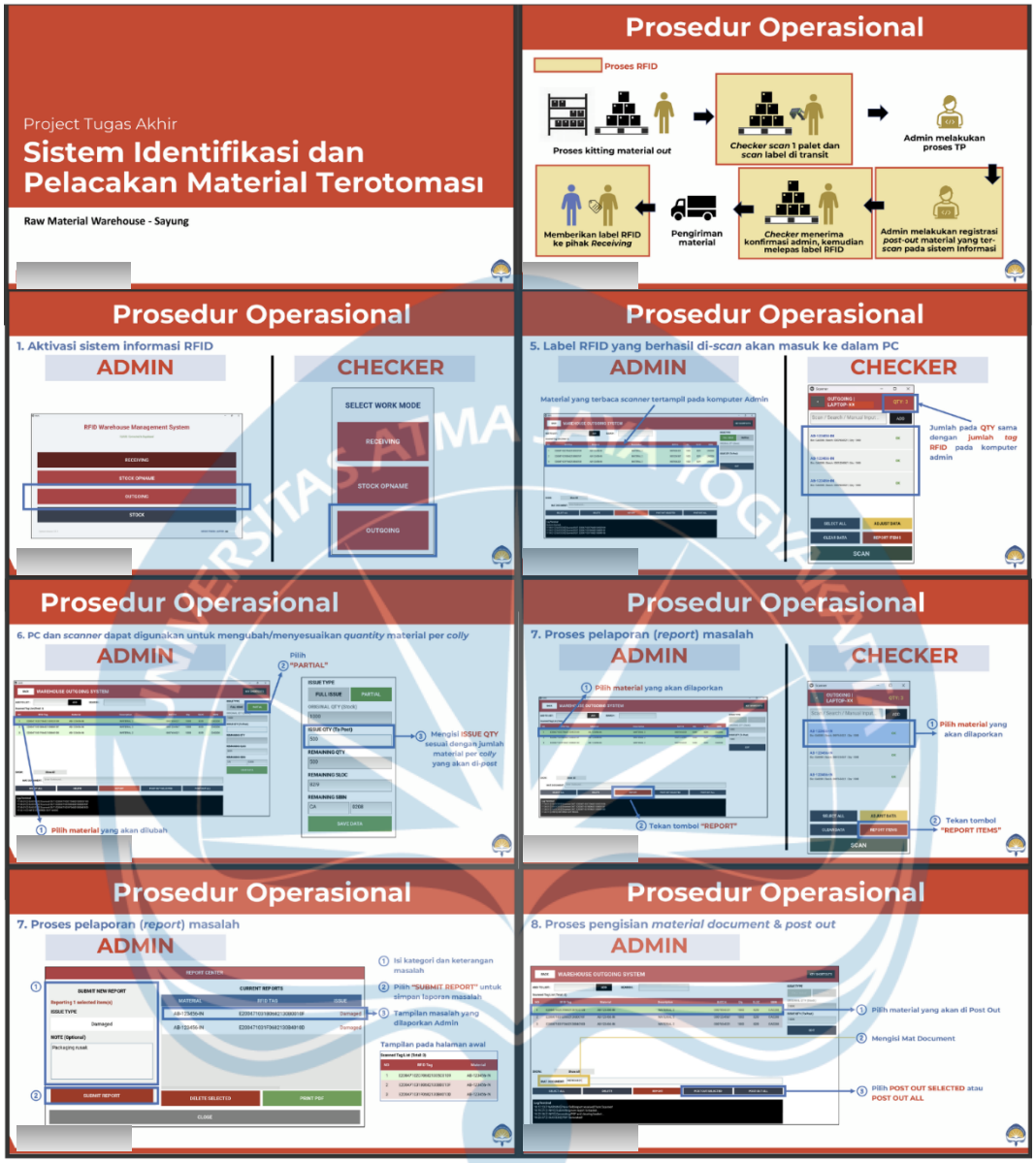
Lampiran 6 Prosedur Penggunaan Sistem Informasi Bagian *Receiving*



Lampiran 7 Prosedur Penggunaan Sistem Informasi Bagian *Inventory*

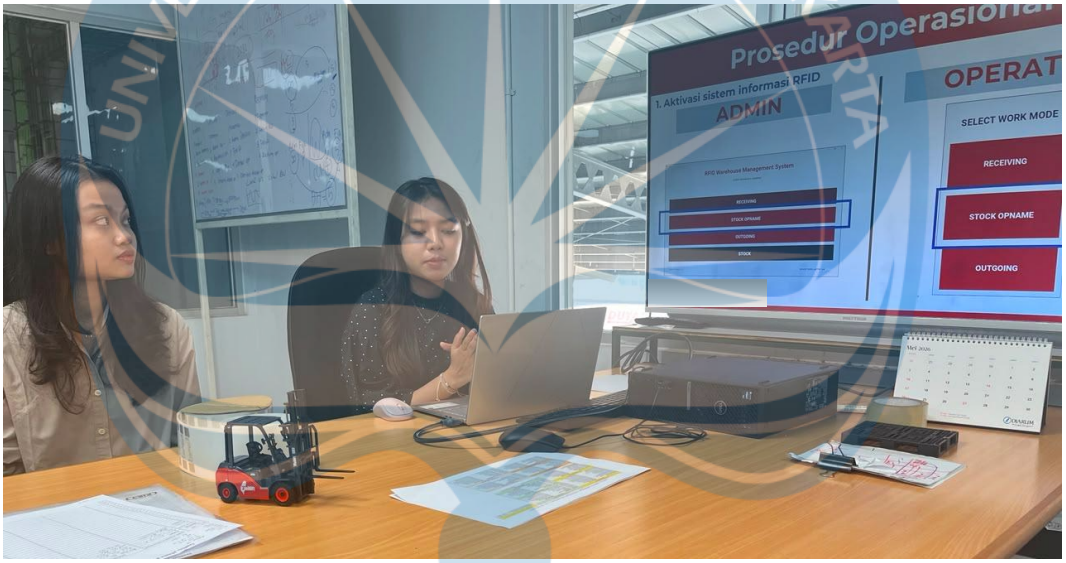


Lampiran 8 Prosedur Penggunaan Sistem Informasi Bagian *Outgoing*



Lampiran 9 Dokumentasi Proses Implementasi







Lampiran 10 Program Koding *K-Means Clustering*

```
import os
import pandas as pd
import numpy as np
from openpyxl import load_workbook
from openpyxl.styles import Border, Side
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples
from math import floor

# Konfigurasi
INPUT_FILE   = r"D:\Documents\DATA NEW 19 DES 2025.xlsx"
OUTPUT_FILE  = r"D:\Documents\Result.xlsx"
K_RANGE      = range(100, 201, 1)
RANDOM_STATE = 42
N_INIT       = 50
PALLET_L     = 130
PALLET_W     = 120
LIMIT_H      = 250

# Fungsi utilitas
def find_first_of_alternatives(columns, alternatives):
    for alt in alternatives:
        match = [col for col in columns if alt.lower() in str(col).lower()]
        if match:
            return match[0]
    raise ValueError(f"Tidak ada kolom yang cocok dengan {alternatives}")

def to_float(x):
    try:
        return float(str(x).replace(",", "."))
    except:
        return np.nan

# Load Excel
print("Accessing Excel:",INPUT_FILE)
wb = load_workbook(INPUT_FILE, data_only=True)
ws = wb.active

raw_rows = []
for row in ws.iter_rows(values_only=True):
    raw_rows.append(list(row))

raw_header = raw_rows[0]
raw_data = raw_rows[1:]
df_raw = pd.DataFrame(raw_data, columns=raw_header)
data_rows = []
for row in ws.iter_rows(min_row=2, values_only=False):
    if ws.row_dimensions[row[0].row].hidden:
        continue
    data_rows.append([cell.value for cell in row])

columns = [cell.value for cell in ws[1]]
df = pd.DataFrame(data_rows, columns=columns)
df = df.dropna(how="all").reset_index(drop=True)

# Identifikasi kolom
col_length      = find_first_of_alternatives(df.columns, ["Panjang Packing Terkecil"])
col_width       = find_first_of_alternatives(df.columns, ["Lebar Packing Terkecil"])
col_height      = find_first_of_alternatives(df.columns, ["Tinggi Packing Terkecil"])
col_weight      = find_first_of_alternatives(df.columns, ["Bruto", "Berat", "Berat
(gram)"])
col_qty_small   = find_first_of_alternatives(df.columns, ["Std Packing Per Packaging
Terkecil (UM SAP)", "Std Packing"])
col_qty_big     = find_first_of_alternatives(df.columns, ["Total Packing Kecil Dalam 1
Packing Besar", "Total Packing Kecil"])
col_length_big  = find_first_of_alternatives(df.columns, ["Panjang Packing Terbesar"])
col_width_big   = find_first_of_alternatives(df.columns, ["Lebar Packing Terbesar"])
```

Lampiran 10 Program Koding *K-Means Clustering*

```
col_height_big  = find_first_of_alternatives(df.columns, ["Tinggi Packing Terbesar"])
col_material    = find_first_of_alternatives(df.columns, ["Material Number", "Material"])
col_type_small  = find_first_of_alternatives(df.columns, ["Tipe Packaging Terkecil"])
col_description = find_first_of_alternatives(df.columns, ["Deskripsi", "Material
Description", "Description"])
col_vendor      = find_first_of_alternatives(df.columns, ["Kode Vendor", "Vendor",
"Supplier", "Kode Supplier", "Vendor Code"])
col_QTYSTock    = find_first_of_alternatives(df.columns, ["Qty Stock"])

# Preprocessing numeric
df["Quantity Stock"]           = df[col_QTYSTock].apply(to_float)
df["Panjang Pack. Kecil (cm)"] = df[col_length].apply(to_float)
df["Lebar Pack. Kecil (cm)"]   = df[col_width].apply(to_float)
df["Tinggi Pack. Kecil (cm)"]  = df[col_height].apply(to_float)
df["Berat (gram)"]             = df[col_weight].apply(to_float)
df["Qty Pack. Kecil"]          = df[col_qty_small].apply(to_float)

df["Volume Pack. Kecil (cm3)"] = (
    df["Panjang Pack. Kecil (cm)"] *
    df["Lebar Pack. Kecil (cm)"] *
    df["Tinggi Pack. Kecil (cm)"]
)

df["Panjang Pack. Besar (cm)"] = df[col_length_big].apply(to_float)
df["Lebar Pack. Besar (cm)"]   = df[col_width_big].apply(to_float)
df["Tinggi Pack. Besar (cm)"]  = df[col_height_big].apply(to_float)
df["Volume Pack. Besar (cm3)"] = (
    df["Panjang Pack. Besar (cm)"] *
    df["Lebar Pack. Besar (cm)"] *
    df["Tinggi Pack. Besar (cm)"]
)

df_before_group = df.copy()

# Filter: pilih hanya material dengan volume besar maksimum per nomor material
df_selected = df_before_group.copy()

# Eliminasi material tidak masuk kriteria pallet dan volum kecil > volum besar
mask_invalid = (
    (df_selected["Volume Pack. Kecil (cm3)"] > df_selected["Volume Pack. Besar (cm3)"]) |
    (df_selected["Panjang Pack. Besar (cm)"] > PALLET_L) |
    (df_selected["Lebar Pack. Besar (cm)"]   > PALLET_L) |
    (df_selected["Tinggi Pack. Besar (cm)"]  > LIMIT_H)
)

df_excluded = df_selected[mask_invalid].copy().reset_index(drop=True)
df_valid = df_selected[~mask_invalid].copy().reset_index(drop=True)

# Clustering
features = df_valid[["Panjang Pack. Besar (cm)", "Lebar Pack. Besar (cm)", "Tinggi Pack.
Besar (cm)"]].fillna(0)
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

best_score = -1
best_inertia = None
best_labels = None
best_k = None
```

Lampiran 10 Program Koding *K-Means Clustering*

```
for k in K_RANGE:
    kmeans = KMeans(n_clusters=k, random_state=RANDOM_STATE, n_init=N_INIT)
    labels = kmeans.fit_predict(features_scaled)

    if len(set(labels)) > 1:
        try:
            score = silhouette_score(features_scaled, labels)
        except Exception as e:
            continue
        if score > best_score:
            best_score = score
            best_inertia = kmeans.inertia_
            best_labels = labels
            best_k = k
# Case Error
if best_labels is None or len(best_labels) != len(df_valid):
    raise RuntimeError("Terjadi masalah: best_labels tidak terisi atau panjangnya tidak
sama dengan jumlah data valid.")

df_valid = df_valid.copy()
df_valid["Cluster"] = best_labels.astype(int)

# Silhouette per sample & per-cluster score summary
sil_samples = silhouette_samples(features_scaled, best_labels)
df_valid["Silhouette_Sample"] = sil_samples

silhouette_summary = (
    df_valid.groupby("Cluster")["Silhouette_Sample"]
    .mean()
    .reset_index()
    .rename(columns={"Silhouette_Sample": "Silhouette_Score_Cluster"})
)

# Summary per cluster
summary = df_valid.groupby("Cluster").agg(
    Count = ("Cluster", "size"),
    Max_Panjang_Pack = ("Panjang Pack. Besar (cm)", "max"),
    Max_Lebar_Pack   = ("Lebar Pack. Besar (cm)", "max"),
    Max_Tinggi_Pack  = ("Tinggi Pack. Besar (cm)", "max"),
    Max_Berat        = ("Berat (gram)", "max"),
    Avg_Berat        = ("Berat (gram)", "mean")
).reset_index()

summary["Max_Volume_Pack"] = (summary["Max_Panjang_Pack"] * summary["Max_Lebar_Pack"] *
summary["Max_Tinggi_Pack"])
summary = summary.merge(silhouette_summary, on="Cluster", how="left")

cluster_dims = summary.set_index("Cluster")[["Max_Panjang_Pack", "Max_Lebar_Pack",
"Max_Tinggi_Pack"]].to_dict(orient="index")

df_valid["Panjang Pack. Besar NEW (cm)"] =
df_valid["Cluster"].map(summary.set_index("Cluster")["Max_Panjang_Pack"])
df_valid["Lebar Pack. Besar NEW (cm)"]   =
df_valid["Cluster"].map(summary.set_index("Cluster")["Max_Lebar_Pack"])
df_valid["Tinggi Pack. Besar NEW (cm)"]  =
df_valid["Cluster"].map(summary.set_index("Cluster")["Max_Tinggi_Pack"])
df_valid["Volume Pack. Besar NEW (cm3)"] = (
    df_valid["Panjang Pack. Besar NEW (cm)"] *
    df_valid["Lebar Pack. Besar NEW (cm)"] *
    df_valid["Tinggi Pack. Besar NEW (cm)"]
)

df_valid["Fit Volume Pack. Besar (cm3)"] = np.floor(df_valid["Volume Pack. Besar NEW
(cm3)"] / df_valid["Volume Pack. Kecil (cm3)"]).fillna(1).round(2)
df_valid["Selisih Volume (cm3)"] = (df_valid["Volume Pack. Besar NEW (cm3)"] -
df_valid["Volume Pack. Besar (cm3)"])
```

Lampiran 10 Program Koding *K-Means Clustering*

```
output_cols = [
    col_material, col_description, col_type_small, col_qty_small,
    col_vendor,
    "Cluster",
    "Panjang Pack. Kecil (cm)", "Lebar Pack. Kecil (cm)", "Tinggi Pack. Kecil (cm)",
    "Panjang Pack. Besar (cm)", "Lebar Pack. Besar (cm)", "Tinggi Pack. Besar (cm)",
    "Panjang Pack. Besar NEW (cm)", "Lebar Pack. Besar NEW (cm)", "Tinggi Pack. Besar NEW
(cm)",
    "Volume Pack. Besar NEW (cm3)", "Volume Pack. Besar (cm3)",
    "Selisih Volume (cm3)", col_weight
]

for c in output_cols:
    if c not in df_valid.columns:
        df_valid[c] = np.nan

if os.path.exists(OUTPUT_FILE):
    try:
        os.remove(OUTPUT_FILE)
    except Exception as e:
        print("Gagal menghapus file lama:", e)

# Excel
with pd.ExcelWriter(OUTPUT_FILE, engine="openpyxl", mode="w") as writer:
    df_raw.to_excel(writer, sheet_name="Data Raw", index=False)

    df_valid.to_excel(writer, sheet_name="Processed Material", index=False)

    df_excluded.to_excel(writer, sheet_name="Excluded Material", index=False)

    summary.to_excel(writer, sheet_name="Summary", index=False)

    active_cluster_ids = sorted(df_valid["Cluster"].dropna().astype(int).unique())
    for cluster_id in active_cluster_ids:
        cluster_df = df_valid[df_valid["Cluster"].astype(int) == int(cluster_id)].copy()
        if cluster_df.shape[0] == 0:
            continue
        sheet_name = f"Cluster {int(cluster_id)}"
        if len(sheet_name) > 31:
            sheet_name = sheet_name[:31]
        cluster_df[output_cols].to_excel(writer, sheet_name=sheet_name, index=False)

    wb = writer.book
    thin = Side(border_style="thin", color="000000")
    border = Border(top=thin, left=thin, right=thin, bottom=thin)

    for sheet_name in wb.sheetnames:
        ws = wb[sheet_name]

        for col in ws.columns:
            max_length = 0
            col_letter = col[0].column_letter
            for cell in col:
                try:
                    val = str(cell.value)
                    if len(val) > max_length:
                        max_length = len(val)
                except:
                    pass
            ws.column_dimensions[col_letter].width = max_length + 2

        for row in ws.iter_rows():
            for cell in row:
                cell.border = border

        ws.auto_filter.ref = ws.dimensions
```

Lampiran 10 Program Koding *K-Means Clustering*

```
# Statistik
all_clusters = set(range(best_k)) if best_k is not None else
set(df_valid["Cluster"].dropna().astype(int).unique())
active_clusters = set(df_valid["Cluster"].dropna().astype(int).unique())
empty_clusters = sorted(list(all_clusters - active_clusters))

print("=" * 40)
print(f"Jumlah Cluster                    : {len(all_clusters)}")
print(f"Jumlah Cluster Kosong             : {len(empty_clusters)}")
print(f"Cluster yang kosong               : {empty_clusters if empty_clusters else '-
'}")
print(f"Total Cluster Aktif               : {len(active_clusters)}")
print(f"Jumlah Excluded                   : {len(df_excluded)}")
print(f"Inertia/SSE                       : {best_inertia}")
print(f"Silhouette Score Terbaik          : {best_score:.4f}")
print("-" * 40)
print(f"Hasil disimpan ke: {OUTPUT_FILE}")
print("=" * 40)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
import os
import string
import threading
import socket
from datetime import datetime
from supabase import create_client, Client, ClientOptions

from reportlab.lib import colors
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet

from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.lang import Builder
from kivy.clock import Clock
from kivy.core.window import Window
from kivy.uix.behaviors import ButtonBehavior
from kivy.uix.gridlayout import GridLayout
from kivy.uix.label import Label
from kivy.graphics import Color, Rectangle
from kivy.uix.behaviors import TouchRippleBehavior
from kivy.uix.dropdown import DropDown
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.config import Config
from kivy.factory import Factory
from kivy.uix.modalview import ModalView
from kivy.core.clipboard import Clipboard

import tkinter as tk
from tkinter import filedialog

SUPABASE_URL = "https://XYZ.supabase.co"
SUPABASE_KEY = "XYZ"

KV_DESIGN = '''
ScreenManager:
    MainMenuScreen:
    TransactionScreen:
    StockOpnameScreen:
    StockOpnameDetailScreen:
    StockScreen:

<MainMenuScreen>:
    name: 'main_menu'
    canvas.before:
        Color:
            rgba: 0.95, 0.96, 0.98, 1
        Rectangle:
            pos: self.pos
            size: self.size

    BoxLayout:
        orientation: 'vertical'
        padding: '40dp'
        spacing: '30dp'

        BoxLayout:
            orientation: 'vertical'
            size_hint_y: None
            height: '120dp'
            spacing: '10dp'
            Label:
                text: 'RFID Warehouse Management System'
                font_size: '42sp'
                bold: True
                color: 220/255, 20/255, 60/255, 1
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        Label:
            id: status_label
            text: 'CLOUD: Connected to Supabase'
            font_size: '16sp'
            color: 0.2, 0.6, 0.2, 1

    Widget:
        size_hint_y: None
        height: '20dp'

    BoxLayout:
        orientation: 'vertical'
        spacing: '20dp'

        Button:
            id: btn_RECEIVING
            text: 'RECEIVING'
            font_size: '32sp'
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255, 1
            on_release: app.open_transaction('IN')

        Button:
            id: btn_so
            text: 'STOCK OPNAME'
            font_size: '32sp'
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255,
            on_release: app.open_stock_opname()

        Button:
            id: btn_OUTGOING
            text: 'OUTGOING'
            font_size: '32sp'
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255,
            on_release: app.open_transaction('OUT')

        Button:
            id: btn_stock
            text: 'STOCK'
            font_size: '32sp'
            bold: True
            background_normal: ''
            background_color: 44/255, 62/255, 80/255, 1
            on_release: app.open_password_popup()

    BoxLayout:
        size_hint_y: None
        height: '40dp'
        padding: ['10dp', '0dp']
        Label:
            text: 'RFID Tracking & Identification System'
            font_size: '14sp'
            color: 0.6, 0.6, 0.6, 1
            halign: 'left'
            text_size: self.size
        Label:
            text: 'DEVICE TOKEN: ' + app.pc_token
            font_size: '14sp'
            bold: True
            color: 0.1, 0.2, 0.4, 1
            halign: 'right'
            text_size: self.size
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
<TransactionScreen>:
    name: 'transaction_page'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        Rectangle:
            pos: self.pos
            size: self.size

    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            id: header_box
            size_hint_y: None
            height: '80dp'
            padding: ['20dp', '0dp']
            spacing: '15dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

            Button:
                text: 'BACK'
                size_hint_x: None
                width: '100dp'
                size_hint_y: None
                height: '40dp'
                pos_hint: {'center_y': 0.5}
                background_normal: ''
                background_color: 1, 1, 1, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.screen_manager.current = 'main_menu'

            Label:
                id: nav_title
                text: 'WAREHOUSE RECEIVING SYSTEM'
                font_size: '28sp'
                bold: True
                halign: 'left'
                text_size: self.size
                valign: 'middle'

            Widget:
                size_hint_x: 1

            Button:
                text: 'KEY SHORTCUTS'
                size_hint_x: None
                width: '140dp'
                size_hint_y: None
                height: '40dp'
                pos_hint: {'center_y': 0.5}
                background_normal: ''
                background_color: 0.2, 0.2, 0.2, 1
                color: 1, 1, 1, 1
                bold: True
                on_release: app.open_shortcut_guide()

        BoxLayout:
            orientation: 'horizontal'
            padding: 10
            spacing: 15

            BoxLayout:
                orientation: 'vertical'
                spacing: 10
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
BoxLayout:
    size_hint_y: None
    height: '35dp'
    spacing: 10

    Label:
        text: 'ADD TO LIST :'
        color: 0, 0, 0, 1
        bold: True
        size_hint_x: None
        width: '100dp'
        halign: 'left'
        valign: 'middle'
        text_size: self.size
    TextInput:
        id: input_add
        size_hint_x: None
        width: '200dp'
        multiline: False
        on_text_validate: app.process_scan_in()
    Button:
        text: 'ADD'
        size_hint_x: None
        width: '80dp'
        background_normal: ''
        background_color: 0, 0, 0, 1
        bold: True
        on_release: app.process_scan_in()

    Widget:
        size_hint_x: None
        width: '30dp'

    Label:
        text: 'SEARCH :'
        color: 0, 0, 0, 1
        bold: True
        size_hint_x: None
        width: '80dp'
        halign: 'left'
        valign: 'middle'
        text_size: self.size
    TextInput:
        id: input_search
        size_hint_x: None
        width: '250dp'
        multiline: False
        on_text: app.filter_table(self.text)

    Widget:
        size_hint_x: 1

Label:
    id: lbl_trans_count
    text: 'Scanned Tag List (Total: 0)'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    halign: 'left'
    text_size: self.size

ScrollView:
    do_scroll_x: True
    do_scroll_y: False
    bar_width: '10dp'

    BoxLayout:
        orientation: 'vertical'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
GridLayout:
    cols: 8
    size_hint_y: None
    height: '40dp'
    canvas.before:
        Color:
            rgba: 220/255, 20/255, 60/255, 1
        Rectangle:
            pos: self.pos
            size: self.size
    Label:
        text: 'NO'
        bold: True
        size_hint_x: 0.4
    Label:
        text: 'RFID Tag'
        bold: True
        size_hint_x: 2.2
    Label:
        text: 'Material'
        bold: True
        size_hint_x: 1.4
    Label:
        text: 'Description'
        bold: True
        size_hint_x: 4.0
    Label:
        text: 'BATCH'
        bold: True
        size_hint_x: 1.1
    Label:
        text: 'Qty'
        bold: True
        size_hint_x: 0.6
    Label:
        text: 'SLOC'
        bold: True
        size_hint_x: 0.8
    Label:
        text: 'SBIN'
        bold: True
        size_hint_x: 0.8


ScrollView:
    id: table_scroll
    do_scroll_x: False

    BoxLayout:
        orientation: 'vertical'
        size_hint_y: None
        height: max(self.minimum_height, table_scroll.height)

        BoxLayout:
            id: table_content
            orientation: 'vertical'
            size_hint_y: None
            height: self.minimum_height
            padding: 2

        Widget:
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
BoxLayout:
    size_hint_y: None
    height: '35dp'
    spacing: 10
    padding: [0, 5, 0, 5]

    Label:
        text: 'SHOW:'
        color: 0, 0, 0, 1
        bold: True
        size_hint_x: None
        width: '100dp'
        halign: 'left'
        valign: 'middle'
        text_size: self.size

    Spinner:
        id: filter_spinner
        text: 'Show All'
        values: ['Show All', 'Incomplete', 'Material Ready', 'Reported
                Material']
        size_hint_x: None
        width: '200dp'
        background_normal: ''
        background_color: 0.9, 0.9, 0.9, 1
        color: 0, 0, 0, 1
        bold: True
        on_text: app.handle_spinner_filter(self.text)

    Widget:
        size_hint_x: 1

BoxLayout:
    size_hint_y: None
    height: '40dp'
    spacing: 10
    padding: [0, 10, 0, 0]
    Label:
        id: global_doc_label
        text: 'INBOUND ID:'
        color: 0, 0, 0, 1
        bold: True
        size_hint_x: None
        width: '160dp'
        halign: 'right'
        valign: 'middle'
        text_size: self.size
    TextInput:
        id: global_doc_id
        hint_text: 'Enter ID...'
        multiline: False
        size_hint_y: None
        height: '35dp'
        on_text: app.filter_doc_id_history(self.text)
        on_focus: if self.focus: app.show_doc_id_dropdown(self)

BoxLayout:
    size_hint_y: None
    height: '40dp'
    spacing: 10
    padding: [0, 10, 0, 0]

    Button:
        text: 'SELECT ALL'
        background_normal: ''
        background_color: 44/255, 62/255, 80/255, 1
        bold: True
        on_release: app.select_all_rows()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Button:
                text: 'DELETE'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                on_release: app.delete_selected_rows()

            Button:
                id: btn_edit_selected
                text: 'EDIT DATA'
                background_normal: ''
                background_color: 255/255, 193/255, 7/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.handle_bottom_edit_click()

            Button:
                id: btn_report_center
                text: 'REPORT'
                background_normal: ''
                background_color: 255/255, 0, 0, 1
                bold: True
                on_release: app.open_trans_report_popup()
            Button:
                id: btn_post_selected
                text: 'POST IN SELECTED'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                on_release: app.post_in_selected()
            Button:
                id: btn_post_all
                text: 'POST IN ALL'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                on_release: app.post_in_all()
        Widget:
            size_hint_y: None
            height: '10dp'

        BoxLayout:
            orientation: 'vertical'
            size_hint_y: None
            height: '120dp'
            canvas.before:
                Color:
                    rgba: 0, 0, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Label:
                text: 'Log Terminal'
                color: 1, 1, 1, 1
                bold: True
                size_hint_y: None
                height: '25dp'
                text_size: self.size
                halign: 'left'
                padding_x: 10
            ScrollView:
                id: log_scroll
                Label:
                    id: log_text
                    text: 'System Started...\\n'
                    color: 1, 1, 1, 1
                    text_size: self.width, None
                    size_hint_y: None
                    height: self.texture_size[1]
                    padding: [10, 5]
                    halign: 'left'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        ScrollView:
            id: right_panel_scroll
            size_hint_x: None
            width: '0dp'
            opacity: 0
            bar_width: '5dp'

            BoxLayout:
                orientation: 'vertical'
                size_hint_y: None
                height: self.minimum_height
                padding: ['10dp', '0dp', '5dp', '0dp']
                spacing: 7

                BoxLayout:
                    id: panel_inbound
                    orientation: 'vertical'
                    size_hint_y: None
                    height: self.minimum_height
                    spacing: 7

                    Label:
                        text: 'MATERIAL'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_y: None
                        height: '20dp'
                        text_size: self.size
                        halign: 'left'
                    TextInput:
                        id: adj_mat
                        size_hint_y: None
                        height: '35dp'
                        multiline: False
                        on_text_validate: app.fetch_catalog_data(self.text)
                        on_focus: if not self.focus: app.fetch_catalog_data(self.text)

                    Label:
                        text: 'DESCRIPTION'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_y: None
                        height: '20dp'
                        text_size: self.size
                        halign: 'left'
                    TextInput:
                        id: adj_desc
                        size_hint_y: None
                        height: '35dp'
                        multiline: False

                    Label:
                        text: 'BATCH'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_y: None
                        height: '20dp'
                        text_size: self.size
                        halign: 'left'
                    TextInput:
                        id: adj_batch
                        text: ''
                        size_hint_y: None
                        height: '35dp'
                        multiline: False
```

UNIVERSITAS AMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
Label:
    text: 'QUANTITY'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    text_size: self.size
    halign: 'left'
TextInput:
    id: adj_qty
    size_hint_y: None
    height: '35dp'
    multiline: False

Label:
    text: 'SLOC'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    text_size: self.size
    halign: 'left'
TextInput:
    id: adj_sloc
    text: ''
    size_hint_y: None
    height: '35dp'
    multiline: False
    on_text: app.filter_sloc_dropdown(self.text)

Label:
    text: 'SBIN'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    text_size: self.size
    halign: 'left'
BoxLayout:
    orientation: 'horizontal'
    size_hint_y: None
    height: '35dp'
    spacing: 5
    TextInput:
        id: adj_sbin_h
        text: ''
        multiline: False
        on_text: app.filter_sbin_h_dropdown(self.text)
    TextInput:
        id: adj_sbin_n
        text: ''
        multiline: False
        on_text: app.filter_sbin_n_dropdown(self.text)

Label:
    text: 'PARSING'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    text_size: self.size
    halign: 'left'
TextInput:
    id: sap_input
    hint_text: 'Paste SAP row...'
    multiline: False
    size_hint_y: None
    height: '35dp'
    on_text: app.on_sap_paste(self.text)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Button:
                text: 'SAVE DATA'
                size_hint_y: None
                height: '45dp'
                background_normal: ''
                background_color: 40/255, 167/255, 69/255, 1
                bold: True
                on_release: app.handle_adjust_click()

            Button:
                text: 'CANCEL EDIT'
                size_hint_y: None
                height: '45dp'
                background_normal: ''
                background_color: 220/255, 53/255, 69/255, 1
                bold: True
                on_release: app.set_edit_mode(False)

        BoxLayout:
            id: panel_outbound
            orientation: 'vertical'
            size_hint_y: None
            height: self.minimum_height
            spacing: 7

            Label:
                text: 'ISSUE TYPE'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size

            BoxLayout:
                size_hint_y: None
                height: '40dp'
                spacing: 5
                Button:
                    id: btn_type_full
                    text: 'FULL ISSUE'
                    bold: True
                    background_normal: ''
                    background_color: 0.1, 0.6, 0.1, 1
                    on_release: app.toggle_issue_type('FULL')
                Button:
                    id: btn_type_partial
                    text: 'PARTIAL'
                    bold: True
                    background_normal: ''
                    background_color: 0.8, 0.8, 0.8, 1
                    color: 0, 0, 0, 1
                    on_release: app.toggle_issue_type('PARTIAL')

            Label:
                text: 'ORIGINAL QTY (Stock)'
                color: 0.5, 0.5, 0.5, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: out_orig_qty
                readonly: True
                background_color: 0.9, 0.9, 0.9, 1
                size_hint_y: None
                height: '35dp'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
Label:
    text: 'ISSUE QTY (To Post)'
    color: 0, 0, 0, 1
    bold: True
    size_hint_y: None
    height: '20dp'
    halign: 'left'
    text_size: self.size

TextInput:
    id: out_issue_qty
    disabled: False
    readonly: True
    background_color: 0.9, 0.9, 0.9, 0.5
    size_hint_y: None
    height: '35dp'
    on_text: app.calculate_remaining_qty()

BoxLayout:
    id: ecer_wrapper
    orientation: 'vertical'
    size_hint_y: None
    height: self.minimum_height

    BoxLayout:
        id: panel_ecer_location
        orientation: 'vertical'
        size_hint_y: None
        height: self.minimum_height
        spacing: 7

        Label:
            text: 'REMAINING QTY'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
        TextInput:
            id: out_rem_qty
            readonly: True
            background_color: 0.9, 0.9, 0.9, 1
            size_hint_y: None
            height: '35dp'
            foreground_color: 0, 0, 0, 1

        Label:
            text: 'REMAINING SLOC'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
        TextInput:
            id: out_rem_sloc
            size_hint_y: None
            height: '35dp'

        Label:
            text: 'REMAINING SBIN'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                            BoxLayout:
                                size_hint_y: None
                                height: '35dp'
                                spacing: 5
                                TextInput:
                                    id: out_rem_sbin_h
                                TextInput:
                                    id: out_rem_sbin_n

                        Button:
                            text: 'SAVE DATA'
                            size_hint_y: None
                            height: '45dp'
                            background_normal: ''
                            background_color: 40/255, 167/255, 69/255, 1
                            color: 1, 1, 1, 1
                            bold: True
                            on_release: app.handle_outbound_click()

                        Button:
                            text: 'CANCEL EDIT'
                            size_hint_y: None
                            height: '45dp'
                            background_normal: ''
                            background_color: 220/255, 53/255, 69/255, 1
                            bold: True
                            on_release: app.set_edit_mode(False)

<StockOpnameScreen>:
    name: 'stock_opname_page'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: 10
            spacing: 10
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: 'BACK'
                size_hint_x: None
                width: '100dp'
                background_normal: ''
                background_color: 1, 1, 1, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.root.current = 'main_menu'
            Label:
                text: 'WAREHOUSE STOCK OPNAME SYSTEM'
                bold: True
                font_size: '18sp'
                text_size: self.size
                halign: 'left'
                valign: 'middle'

        BoxLayout:
            orientation: 'vertical'
            padding: 30
            spacing: 10
            Label:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            text: 'Search SLoc'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            text_size: self.size
            halign: 'left'
        TextInput:
            id: so_search
            size_hint_x: None
            width: '250dp'
            size_hint_y: None
            height: '35dp'
            multiline: False
            on_text: app.filter_so_selection(self.text)

        BoxLayout:
            orientation: 'vertical'
            size_hint_max_x: '800dp'
            size_hint_y: 1
            canvas.before:
                Color:
                    rgba: 0, 0, 0, 1
                Line:
                    width: 1
                    rectangle: self.x, self.y, self.width, self.height
            GridLayout:
                cols: 4
                size_hint_y: None
                height: '40dp'
                canvas.before:
                    Color:
                        rgba: 0.9, 0.9, 0.9, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                Label:
                    text: 'NO'
                    color: 0, 0, 0, 1
                    bold: True
                Label:
                    text: 'SLoc'
                    color: 0, 0, 0, 1
                    bold: True
                Label:
                    text: 'SBin'
                    color: 0, 0, 0, 1
                    bold: True
                Label:
                    text: 'Filled Bin'
                    color: 0, 0, 0, 1
                    bold: True
            ScrollView:
                id: so_select_scroll
                BoxLayout:
                    id: so_select_content
                    orientation: 'vertical'
                    size_hint_y: None
                    height: self.minimum_height
                    padding: 2

        Widget:
            size_hint_y: None
            height: '10dp'
        BoxLayout:
            size_hint_y: None
            height: '45dp'
            spacing: 10
            Button:
                text: 'CLEAR SELECTION'
                size_hint_x: 0.4
                background_normal: ''
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.clear_so_selection()
                Button:
                    text: 'SELECT'
                    size_hint_x: 0.6
                    background_normal: ''
                    background_color: 0, 0, 0, 1
                    bold: True
                    on_release: app.enter_so_detail()

<StockOpnameDetailScreen>:
    name: 'so_detail_page'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: 10
            spacing: 10
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: 'BACK'
                size_hint_x: None
                width: '100dp'
                background_normal: ''
                background_color: 1, 1, 1, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.confirm_exit_so_detail()
            Label:
                id: so_detail_title
                text: 'SLOC XX STOCK OPNAME SYSTEM'
                bold: True
                font_size: '18sp'
                text_size: self.size
                halign: 'left'
                valign: 'middle'

        BoxLayout:
            orientation: 'vertical'
            padding: 30
            spacing: 10
            Label:
                id: lbl_so_detail_search
                text: 'Search RFID Tag in SBin XX'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                text_size: self.size
                halign: 'left'
            TextInput:
                id: so_detail_search
                size_hint_x: None
                width: '250dp'
                size_hint_y: None
                height: '35dp'
                multiline: False
                on_text: app.filter_so_detail(self.text)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            BoxLayout:
                orientation: 'vertical'
                size_hint_y: 1
                canvas.before:
                    Color:
                        rgba: 0, 0, 0, 1
                    Line:
                        width: 1
                        rectangle: self.x, self.y, self.width, self.height
                GridLayout:
                    cols: 9
                    size_hint_y: None
                    height: '40dp'
                    canvas.before:
                        Color:
                            rgba: 0.9, 0.9, 0.9, 1
                        Rectangle:
                            pos: self.pos
                            size: self.size
                    Label:
                        text: 'NO'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 0.4
                    Label:
                        text: 'SLoc'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 0.6
                    Label:
                        text: 'SBin'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 0.8
                    Label:
                        text: 'RFID Tag'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 2.5
                    Label:
                        text: 'Material'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 1.2
                    Label:
                        text: 'Description'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 3.0
                    Label:
                        text: 'Batch'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 1.2
                    Label:
                        text: 'Quantity'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 0.6
                    Label:
                        text: 'Status'
                        color: 0, 0, 0, 1
                        bold: True
                        size_hint_x: 1.0
                ScrollView:
                    id: so_detail_scroll
                    BoxLayout:
                        id: so_detail_content
                        orientation: 'vertical'
                        size_hint_y: None
                        height: self.minimum_height
                        padding: 2
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            BoxLayout:
                size_hint_y: None
                height: '45dp'
                spacing: 10

                Button:
                    text: 'REPORT PROBLEM'
                    background_normal: ''
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.open_report_popup()

                Button:
                    text: 'PRINT REPORT'
                    background_normal: ''
                    background_color: 0, 112/255, 210/255, 1
                    bold: True
                    on_release: app.open_pic_popup()

                Button:
                    text: 'FINISH SO'
                    background_normal: ''
                    background_color: 40/255, 167/255, 69/255, 1
                    bold: True
                    on_release: app.finish_so_pc()

<SOPICPopup@ModalView>:
    size_hint: None, None
    size: '300dp', '200dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                size: self.size
                pos: self.pos
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'PRINT SO REPORT'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    size: self.size
                    pos: self.pos
        BoxLayout:
            padding: 20
            orientation: 'vertical'
            Label:
                text: 'Person In Charge (PIC):'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: pic_input
                hint_text: 'Nama Admin/Operator...'
                multiline: False
                size_hint_y: None
                height: '35dp'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: [10, 0, 10, 10]
            spacing: 10
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 0.5, 0.5, 0.5, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'EXCEL'
                background_normal: ''
                background_color: 40/255, 167/255, 69/255, 1
                bold: True
                on_release: app.submit_pic_and_print(pic_input.text, 'excel')
            Button:
                text: 'PDF'
                background_normal: ''
                background_color: 204/255, 0, 0, 1
                bold: True
                on_release: app.submit_pic_and_print(pic_input.text, 'pdf')

<TransReportPopup@ModalView>:
    size_hint: None, None
    size: '950dp', '480dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height

        Label:
            text: 'REPORT CENTER'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

        BoxLayout:
            orientation: 'horizontal'
            padding: 15
            spacing: 20

            BoxLayout:
                orientation: 'vertical'
                size_hint_x: 0.35
                spacing: 10
                canvas.before:
                    Color:
                        rgba: 0.95, 0.95, 0.95, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                padding: 10
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        Label:
            text: 'SUBMIT NEW REPORT'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '30dp'

        Label:
            id: report_lbl_items
            text: 'Reporting X items'
            color: 0.8, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            text_size: self.size
            halign: 'left'

        Label:
            text: 'ISSUE TYPE'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            text_size: self.size
            halign: 'left'
        Spinner:
            id: report_spinner
            text: 'Select Issue...'
            values: ['Damaged', 'Missing Item', 'Wrong Label', 'Other']
            size_hint_y: None
            height: '40dp'
            background_normal: ''
            background_color: 1, 1, 1, 1
            color: 0, 0, 0, 1

        Label:
            text: 'NOTE (Optional)'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            text_size: self.size
            halign: 'left'
        TextInput:
            id: report_note
            multiline: True

        Widget:
            size_hint_y: None
            height: '10dp'

        Button:
            text: 'SUBMIT REPORT'
            size_hint_y: None
            height: '45dp'
            background_normal: ''
            background_color: 204/255, 0, 0, 1
            bold: True
            on_release: app.submit_trans_report()

    BoxLayout:
        orientation: 'vertical'
        size_hint_x: 0.65
        spacing: 5

        Label:
            text: 'CURRENT REPORTS'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '30dp'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            ScrollView:
                do_scroll_x: True
                do_scroll_y: True
                bar_width: '10dp'
                canvas.before:
                    Color:
                        rgba: 0.9, 0.9, 0.9, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size

                BoxLayout:
                    orientation: 'vertical'
                    size_hint_x: None
                    width: '1000dp'
                    size_hint_y: None
                    height: self.minimum_height

                    GridLayout:
                        cols: 5
                        size_hint_y: None
                        height: '35dp'
                        canvas.before:
                            Color:
                                rgba: 0.2, 0.4, 0.6, 1
                            Rectangle:
                                pos: self.pos
                                size: self.size
                        Label:
                            text: 'MATERIAL'
                            bold: True
                            size_hint_x: 1.2
                        Label:
                            text: 'RFID TAG'
                            bold: True
                            size_hint_x: 2.0
                        Label:
                            text: 'ISSUE'
                            bold: True
                            size_hint_x: 1.0
                        Label:
                            text: 'NOTE'
                            bold: True
                            size_hint_x: 2.0
                        Label:
                            text: 'SOURCE'
                            bold: True
                            size_hint_x: 0.8

                    BoxLayout:
                        id: review_list_content
                        orientation: 'vertical'
                        size_hint_y: None
                        height: self.minimum_height
                        spacing: 2
                        padding: 2
            BoxLayout:
                size_hint_y: None
                height: '45dp'
                spacing: 10
                Button:
                    text: 'DELETE SELECTED'
                    background_normal: ''
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.delete_selected_basket_items()
                Button:
                    text: 'PRINT PDF'
                    background_normal: ''
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.print_and_clear_trans_report()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: [20, 0, 20, 10]
            Button:
                text: 'CLOSE'
                background_normal: ''
                background_color: 0.5, 0.5, 0.5, 1
                bold: True
                on_release: root.dismiss()

<ReportPopup@ModalView>:
    size_hint: None, None
    size: '400dp', '350dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'REPORT PROBLEM'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 204/255, 0, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        BoxLayout:
            orientation: 'vertical'
            padding: 20
            spacing: 10
            Label:
                id: report_lbl_items
                text: 'Reporting X items'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                text_size: self.size
                halign: 'left'

            Label:
                text: 'ISSUE TYPE'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                text_size: self.size
                halign: 'left'
            Spinner:
                id: report_spinner
                text: 'Select Issue...'
                values: ['Damaged', 'Missing Item', 'Wrong Label', 'Other']
                size_hint_y: None
                height: '40dp'
                background_normal: ''
                background_color: 0.9, 0.9, 0.9, 1
                color: 0, 0, 0, 1
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Label:
                text: 'NOTE (Optional)'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                text_size: self.size
                halign: 'left'
            TextInput:
                id: report_note
                multiline: True

        BoxLayout:
            orientation: 'horizontal'
            size_hint_y: None
            height: '60dp'
            spacing: 20
            padding: [40, 10, 40, 10]
            Button:
                text: 'SUBMIT'
                background_normal: ''
                background_color: 204/255, 0, 0, 1
                bold: True
                on_release: app.confirm_report()
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()

<ExitMessageBox>:
    size_hint: None, None
    size: '350dp', '150dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0.05, 0.15, 0.3, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'CONFIRM EXIT'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            text: 'Are you sure you want to exit the App?'
            color: 0, 0, 0, 1
            bold: True
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: 10
            spacing: 20
            Button:
                id: btn_yes
                text: 'YES'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                background_normal: ''
                bold: True
                on_release: root.selected = 'YES'; root.confirm()
            Button:
                id: btn_no
                text: 'NO'
                background_normal: ''
                color: 0, 0, 0, 1
                bold: True
                on_release: root.selected = 'NO'; root.confirm()

<PrintSuccessPopup@ModalView>:
    size_hint: None, None
    size: '450dp', '200dp'
    auto_dismiss: True
    BoxLayout:
        orientation: 'vertical'
        padding: 20
        spacing: 15
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'PDF REPORT GENERATED SUCCESSFULLY!'
            color: 0.1, 0.4, 0.1, 1
            bold: True
            font_size: '16sp'
        TextInput:
            id: file_path_text
            readonly: True
            background_color: 0.95, 0.95, 0.95, 1
            foreground_color: 0, 0, 0, 1
            size_hint_y: None
            height: '40dp'
        Button:
            text: 'OK'
            size_hint_y: None
            height: '40dp'
            background_normal: ''
            background_color: 0.1, 0.3, 0.6, 1
            on_release: root.dismiss()

<ConfirmExitSOPopup>:
    size_hint: None, None
    size: '350dp', '150dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0.8, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'EXIT STOCK OPNAME?'
            size_hint_y: None
            height: '40dp'
            bold: True
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            canvas.before:
                Color:
                    rgba: 0.8, 0, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            text: 'Progress will be ended. Are you sure?'
            color: 0, 0, 0, 1
            bold: True
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: 10
            spacing: 20
            Button:
                id: btn_yes
                text: 'YES'
                background_normal: ''
                bold: True
                on_release: root.selected = 'YES'; root.confirm()
            Button:
                id: btn_no
                text: 'CANCEL'
                background_normal: ''
                color: 0, 0, 0, 1
                bold: True
                on_release: root.selected = 'NO'; root.confirm()

<ConfirmFinishNoPrintPopup@ModalView>:
    size_hint: None, None
    size: '400dp', '200dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0.8, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'WARNING: REPORT NOT PRINTED!'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            text: 'Are you sure you want to finish without printing?'
            color: 0, 0, 0, 1
            bold: True
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: 10
            spacing: 20
            Button:
                text: 'Cancel'
                background_normal: ''
                background_color: 0.8, 0.8, 0.8, 1
                color: 0, 0, 0, 1
                bold: True
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                on_release: root.dismiss()
            Button:
                text: 'Finish'
                background_normal: ''
                background_color: 220/255, 20/255, 60/255, 1
                bold: True
                on_release: app.execute_finish_so_clear(); root.dismiss()

<ShortcutGuidePopup@ModalView>:
    size_hint: None, None
    size: '500dp', '450dp'
    auto_dismiss: True
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height

        Label:
            text: 'KEYBOARD SHORTCUTS GUIDE'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

        ScrollView:
            BoxLayout:
                orientation: 'vertical'
                size_hint_y: None
                height: self.minimum_height
                padding: 20
                spacing: 15

                Label:
                    text: 'NAVIGATION & SELECTION'
                    color: 0, 0, 0, 1
                    bold: True
                    size_hint_y: None
                    height: '25dp'
                    text_size: self.size
                    halign: 'left'

                GridLayout:
                    cols: 2
                    size_hint_y: None
                    height: '140dp'
                    spacing: 10

                    Label:
                        text: '[ UP / DOWN ARROW ]'
                        color: 0, 0, 0, 1
                        bold: True
                        text_size: self.size
                        halign: 'left'
                    Label:
                        text: 'Navigate table rows or dropdowns'
                        color: 0.3, 0.3, 0.3, 1
                        text_size: self.size
                        halign: 'left'
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Label:
                text: '[ SHIFT ] + [ UP / DOWN ]'
                color: 0, 0, 0, 1
                bold: True
                text_size: self.size
                halign: 'left'
            Label:
                text: 'Multi-select consecutive rows'
                color: 0.3, 0.3, 0.3, 1
                text_size: self.size
                halign: 'left'

            Label:
                text: '[ CTRL ] + [ A ]'
                color: 0, 0, 0, 1
                bold: True
                text_size: self.size
                halign: 'left'
            Label:
                text: 'Select all scanned items in the table'
                color: 0.3, 0.3, 0.3, 1
                text_size: self.size
                halign: 'left'

            Label:
                text: '[ ESC ]'
                color: 0, 0, 0, 1
                bold: True
                text_size: self.size
                halign: 'left'
            Label:
                text: 'Unselect all / Close dropdown / Back'
                color: 0.3, 0.3, 0.3, 1
                text_size: self.size
                halign: 'left'

        Widget:
            size_hint_y: None
            height: '5dp'
            canvas.before:
                Color:
                    rgba: 0.8, 0.8, 0.8, 1
                Line:
                    points: self.x, self.y, self.right, self.y
                    width: 1

        Label:
            text: 'ACTION & EDITING'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '25dp'
            text_size: self.size
            halign: 'left'

        GridLayout:
            cols: 2
            size_hint_y: None
            height: '140dp'
            spacing: 10

            Label:
                text: '[ ENTER ]'
                color: 0, 0, 0, 1
                bold: True
                text_size: self.size
                halign: 'left'
            Label:
                text: 'Focus Scanner Input / Next field'
                color: 0.3, 0.3, 0.3, 1
                text_size: self.size
                halign: 'left'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    Label:
                        text: '[ DELETE ]'
                        color: 0, 0, 0, 1
                        bold: True
                        text_size: self.size
                        halign: 'left'
                    Label:
                        text: 'Remove selected items from list'
                        color: 0.3, 0.3, 0.3, 1
                        text_size: self.size
                        halign: 'left'

                    Label:
                        text: '[ CTRL ] + [ ENTER ]'
                        color: 0, 0, 0, 1
                        bold: True
                        text_size: self.size
                        halign: 'left'
                    Label:
                        text: 'Enter Edit Mode / Save Data'
                        color: 0.3, 0.3, 0.3, 1
                        text_size: self.size
                        halign: 'left'

                    Label:
                        text: '[ CTRL ] + [ SPACE ]'
                        color: 0, 0, 0, 1
                        bold: True
                        text_size: self.size
                        halign: 'left'
                    Label:
                        text: 'Toggle Full / Partial (Outgoing Only)'
                        color: 0.3, 0.3, 0.3, 1
                        text_size: self.size
                        halign: 'left'

            BoxLayout:
                size_hint_y: None
                height: '50dp'
                padding: [20, 0, 20, 10]
                Button:
                    text: 'CLOSE GUIDE'
                    background_normal: ''
                    background_color: 220/255, 20/255, 60/255, 1
                    bold: True
                    on_release: root.dismiss()

<PasswordPopup@ModalView>:
    size_hint: None, None
    size: '300dp', '200dp'
    auto_dismiss: True
    on_open: pwd_input.focus = True
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                size: self.size
                pos: self.pos
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'SUPERVISORY LOGIN'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                Color:
                    rgba: 44/255, 62/255, 80/255, 1
                Rectangle:
                    size: self.size
                    pos: self.pos
        BoxLayout:
            padding: 20
            orientation: 'vertical'
            TextInput:
                id: pwd_input
                password: True
                hint_text: 'Enter Password...'
                multiline: False
                size_hint_y: None
                height: '40dp'
                on_text_validate: app.verify_password(self.text)
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: [20, 0, 20, 10]
            spacing: 15
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 0.5, 0.5, 0.5, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'LOGIN'
                background_normal: ''
                background_color: 0.1, 0.6, 0.1, 1
                bold: True
                on_release: app.verify_password(pwd_input.text)

<StockAdjustPopup@ModalView>:
    size_hint: None, None
    size: '320dp', '420dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'ADJUST STOCK DATA'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 255/255, 193/255, 7/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        ScrollView:
            BoxLayout:
                orientation: 'vertical'
                padding: '15dp'
                spacing: '5dp'
                size_hint_y: None
                height: self.minimum_height

                Label:
                    id: pop_stock_adj_info
                    text: 'Adjusting X Item(s)'
                    color: 0, 0, 0, 1
                    bold: True
                    size_hint_y: None
                    height: '20dp'
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Label:
                text: 'MATERIAL'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: pop_mat
                multiline: False
                size_hint_y: None
                height: '35dp'

            Label:
                text: 'DESCRIPTION'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: pop_desc
                multiline: False
                size_hint_y: None
                height: '35dp'

            Label:
                text: 'QUANTITY'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: pop_qty
                input_filter: 'int'
                multiline: False
                size_hint_y: None
                height: '35dp'

            Label:
                text: 'BATCH'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            TextInput:
                id: pop_batch
                multiline: False
                size_hint_y: None
                height: '35dp'

            Label:
                text: 'SLOC'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'
                halign: 'left'
                text_size: self.size
            Spinner:
                id: pop_sloc
                text: 'Select SLoc'
                values: ['82I1', '82I3', '82I5', '82I7', '82I8', '82I9', 'TR']
                size_hint_y: None
                height: '35dp'
                background_normal: ''
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    background_color: 0.9, 0.9, 0.9, 1
                    color: 0, 0, 0, 1

                Label:
                    text: 'SBIN'
                    color: 0, 0, 0, 1
                    bold: True
                    size_hint_y: None
                    height: '20dp'
                    halign: 'left'
                    text_size: self.size
                TextInput:
                    id: pop_sbin
                    multiline: False
                    size_hint_y: None
                    height: '35dp'
                    on_text: app.auto_fill_sloc(self.text, 'stock')

        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'SAVE ADJUSTMENT'
                background_normal: ''
                background_color: 0.2, 0.6, 0.2, 1
                bold: True
                on_release: app.submit_stock_adjust(); root.dismiss()

<StockScreen>:
    name: 'stock_page'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: 10
            spacing: 10
            canvas.before:
                Color:
                    rgba: 44/255, 62/255, 80/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: 'BACK'
                size_hint_x: None
                width: '100dp'
                background_normal: ''
                background_color: 1, 1, 1, 1
                color: 0,0,0,1
                bold: True
                on_release: app.screen_manager.current = 'main_menu'
            Label:
                text: 'SUPERVISORY STOCK DATA'
                bold: True
                font_size: '18sp'
                halign: 'left'
                valign: 'middle'
                text_size: self.size
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
BoxLayout:
    orientation: 'vertical'
    padding: [10, 10, 10, 0]

    GridLayout:
        cols: 8
        size_hint_y: None
        height: '35dp'
        canvas.before:
            Color:
                rgba: 0.8, 0.8, 0.8, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'NO'
            color: 0,0,0,1
            bold: True
            size_hint_x: 0.5
        Label:
            text: 'RFID TAG'
            color: 0,0,0,1
            bold: True
            size_hint_x: 2.0
        Label:
            text: 'MATERIAL'
            color: 0,0,0,1
            bold: True
            size_hint_x: 1.5
        Label:
            text: 'DESCRIPTION'
            color: 0,0,0,1
            bold: True
            size_hint_x: 3.0
        Label:
            text: 'BATCH'
            color: 0,0,0,1
            bold: True
            size_hint_x: 1.0
        Label:
            text: 'QTY'
            color: 0,0,0,1
            bold: True
            size_hint_x: 0.8
        Label:
            text: 'SLOC'
            color: 0,0,0,1
            bold: True
            size_hint_x: 0.8
        Label:
            text: 'SBIN'
            color: 0,0,0,1
            bold: True
            size_hint_x: 0.8

    GridLayout:
        cols: 8
        size_hint_y: None
        height: '35dp'
        padding: [0, 2, 0, 5]

        Label:
            text: ''
            size_hint_x: 0.5

        BoxLayout:
            size_hint_x: 2.0
            spacing: 2
            padding: [0, 0, 10, 0]
            TextInput:
                id: flt_rfid
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            hint_text: 'Search...'
            multiline: False
            on_text: app.filter_stock_data()
        Button:
            text: 'X'
            size_hint_x: None
            width: '25dp'
            background_normal: ''
            background_color: 0.8, 0.2, 0.2, 1
            bold: True
            on_release: flt_rfid.text = ''

    BoxLayout:
        size_hint_x: 1.5
        spacing: 2
        padding: [0, 0, 10, 0]
        TextInput:
            id: flt_mat
            hint_text: 'Search...'
            multiline: False
            on_text: app.filter_stock_data()
        Button:
            text: 'X'
            size_hint_x: None
            width: '25dp'
            background_normal: ''
            background_color: 0.8, 0.2, 0.2, 1
            bold: True
            on_release: flt_mat.text = ''

    BoxLayout:
        size_hint_x: 3.0
        spacing: 2
        padding: [0, 0, 10, 0]
        TextInput:
            id: flt_desc
            hint_text: 'Search...'
            multiline: False
            on_text: app.filter_stock_data()
        Button:
            text: 'X'
            size_hint_x: None
            width: '25dp'
            background_normal: ''
            background_color: 0.8, 0.2, 0.2, 1
            bold: True
            on_release: flt_desc.text = ''

    BoxLayout:
        size_hint_x: 1.0
        spacing: 2
        padding: [0, 0, 10, 0]
        TextInput:
            id: flt_batch
            hint_text: 'Search...'
            multiline: False
            on_text: app.filter_stock_data()
        Button:
            text: 'X'
            size_hint_x: None
            width: '25dp'
            background_normal: ''
            background_color: 0.8, 0.2, 0.2, 1
            bold: True
            on_release: flt_batch.text = ''

    Label:
        text: ''
        size_hint_x: 0.8

    BoxLayout:
        size_hint_x: 0.8
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                spacing: 2
                padding: [0, 0, 10, 0]
                TextInput:
                    id: flt_sloc
                    hint_text: 'Search...'
                    multiline: False
                    on_text: app.filter_stock_data()
                Button:
                    text: 'X'
                    size_hint_x: None
                    width: '25dp'
                    background_normal: ''
                    background_color: 0.8, 0.2, 0.2, 1
                    bold: True
                    on_release: flt_sloc.text = ''

            BoxLayout:
                size_hint_x: 0.8
                spacing: 2
                padding: [0, 0, 10, 0]
                TextInput:
                    id: flt_sbin
                    hint_text: 'Search...'
                    multiline: False
                    on_text: app.filter_stock_data()
                Button:
                    text: 'X'
                    size_hint_x: None
                    width: '25dp'
                    background_normal: ''
                    background_color: 0.8, 0.2, 0.2, 1
                    bold: True
                    on_release: flt_sbin.text = ''

        ScrollView:
            id: stock_list_scroll
            do_scroll_x: False
            do_scroll_y: True
            scroll_type: ['bars', 'content']
            bar_width: '12dp'
            bar_color: 0.4, 0.4, 0.4, 0.9
            bar_inactive_color: 0.6, 0.6, 0.6, 0.7

            BoxLayout:
                id: stock_list_content
                orientation: 'vertical'
                size_hint_y: None
                height: self.minimum_height
                spacing: 2

    BoxLayout:
        size_hint_y: None
        height: '60dp'
        padding: [10, 10, 10, 10]
        spacing: 15

        Button:
            text: 'CLEAR FILTERS'
            background_normal: ''
            background_color: 44/255, 62/255, 80/255, 1
            bold: True
            on_release: app.clear_stock_filters()

        Button:
            text: 'SELECT ALL'
            background_normal: ''
            background_color: 44/255, 62/255, 80/255, 1
            bold: True
            on_release: app.select_all_stock_rows()

        Button:
            text: 'ADJUST DATA'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                background_normal: ''
                background_color: 255/255, 193/255, 7/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.open_stock_adjust_popup()

            Button:
                text: 'DELETE STOCK'
                background_normal: ''
                background_color: 204/255, 0, 0, 1
                bold: True
                on_release: app.open_delete_stock_popup()

            Button:
                text: 'REFRESH STOCK'
                background_normal: ''
                background_color: 0.1, 0.6, 0.1, 1
                bold: True
                on_release: app.fetch_stock_data()

<ConfirmDeleteStockPopup@ModalView>:
    size_hint: None, None
    size: '350dp', '200dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0.8, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'CONFIRM DELETE STOCK'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            id: lbl_del_msg
            text: 'Are you sure you want to delete X items?'
            color: 0, 0, 0, 1
            bold: True
            halign: 'center'
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: 10
            spacing: 20
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 0.8, 0.8, 0.8, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'YES, DELETE'
                background_normal: ''
                background_color: 220/255, 20/255, 60/255, 1
                bold: True
                on_release: app.execute_delete_stock(); root.dismiss()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
'''

class MainMenuScreen(Screen): pass
class TransactionScreen(Screen): pass
class StockOpnameScreen(Screen): pass
class StockOpnameDetailScreen(Screen): pass
class StockScreen(Screen): pass

class WMSApp(App):
    def build(self):
        Window.maximize()
        Window.bind(on_key_down=self._on_key_down)
        Window.bind(on_request_close=self.handle_close_request)

        self.pc_token = socket.gethostname().upper()

        opts = ClientOptions(
            headers={"Connection": "close"}
        )

        self.supabase: Client = create_client(
            SUPABASE_URL,
            SUPABASE_KEY,
            options=opts
        )

        self.main_menu_index = 0

        base_alphabet = list(string.ascii_uppercase)
        self.alphabet = [c1 + c2 for c1 in base_alphabet for c2 in base_alphabet]

        self.SLOC_RULES = {
            "82I1": ("AA", "AP"), "82I3": ("AQ", "BF"), "82I5": ("BG", "BL"),
            "82I7": ("BM", "BV"), "82I8": ("BW", "BX"), "82I9": ("BY", "CT"),
            "TR": ("TR", "TR")
        }
        self.LIST_SLOC = sorted(list(self.SLOC_RULES.keys()))
        self.LIST_BIN_NUM = [f"{r:02d}{c:02d}" for r in range(1, 82) for c in range(1, 9)]
        self.LIST_TR_NUM = [f"{i:04d}" for i in range(1, 5001)]

        self.ALL_VALID_PREFIXES = []
        for pair in self.alphabet:
            for sloc, (start, end) in self.SLOC_RULES.items():
                if start <= pair <= end:
                    self.ALL_VALID_PREFIXES.append(pair)
                    break

        self.all_items_cache = []
        self._load_doc_history()
        self.outbound_instructions = {}
        self.selected_rows = []
        self.last_selected_row = None
        self.is_auto_filling = False
        self.edit_mode = False

        self.ignored_uids = set()
        self.log_history = {"IN": "System Started...\n", "OUT": "System Started...\n"}

        self.so_selection_cache = []
        self.so_detail_cache = []
        self.so_scanned_list = []
        self.so_custom_status = {}
        self.so_adjusted_history = []
        self.active_so_sloc = ""
        self.active_so_sbin = ""
        self.so_refresh_event = None
        self.so_popup = None
        self.report_popup = None
        self.exit_popup = None
        self.transaction_refresh_event = None
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        Config.set('input', 'mouse', 'mouse,multitouch_on_demand')
        self.screen_manager = Builder.load_string(KV_DESIGN)
        Clock.schedule_once(lambda dt: self._nav_main_menu(None), 1.0)
        return self.screen_manager

    def on_start(self):
        threading.Thread(target=self.connect_to_cloud, daemon=True).start()
        if hasattr(self, 'bg_monitor_event') and self.bg_monitor_event:
            self.bg_monitor_event.cancel()
        self.bg_monitor_event = Clock.schedule_interval(self._bg_data_integrity_monitor, 3)

    def connect_to_cloud(self):
        try:
            self.supabase.table("MASTER_DATA").select("count", count="exact").limit(1).execute()
            Clock.schedule_once(lambda dt: self.update_status("CLOUD: Connected to Supabase!",
[46/255, 125/255, 50/255, 1]))
        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.update_status(f"CLOUD ERROR: {err_msg}",
[198/255, 40/255, 40/255, 1]))

    def _bg_data_integrity_monitor(self, dt):
        threading.Thread(target=self._thread_integrity_validation, daemon=True).start()

    def _thread_integrity_validation(self):
        if getattr(self, 'is_posting', False): return
        try:
            for mode in ["IN", "OUT"]:
                table_name = f"SCAN_{mode}"
                report_table = f"REPORT_{mode}"
                res = self.supabase.table(table_name).select("RFID_UID").eq("PC_TOKEN",
self.pc_token).limit(99999).execute()

                if not res.data: continue
                uids_in_db = [r['RFID_UID'] for r in res.data]

                stock_res = self.supabase.table("STOCK").select("RFID_UID").in_("RFID_UID",
uids_in_db).execute()
                uids_in_stock = [r['RFID_UID'] for r in stock_res.data] if stock_res.data else
[]

                report_res =
self.supabase.table(report_table).select("RFID_UID").in_("RFID_UID", uids_in_db).execute()
                uids_reported_db = [r['RFID_UID'] for r in report_res.data] if report_res.data
else []

                invalid_uids = []
                reported_uids = []

                for uid in uids_in_db:
                    if mode == "IN" and uid in uids_in_stock: invalid_uids.append(uid)
                    elif mode == "OUT" and uid not in uids_in_stock: invalid_uids.append(uid)

                    if uid in uids_reported_db:
                        reported_uids.append(uid)

                if invalid_uids:
                    self.supabase.table(table_name).delete().in_("RFID_UID",
invalid_uids).eq("PC_TOKEN", self.pc_token).execute()

                Clock.schedule_once(lambda dt, m=mode, i=invalid_uids, r=reported_uids:
self._sync_ui_state(m, i, r))
        except: pass

    def _sync_ui_state(self, mode, invalid_uids, reported_uids):
        if invalid_uids:
            for uid in invalid_uids:
                reason = "Already In Stock" if mode == "IN" else "Not In Stock Anymore"
                self.write_to_log(f"{reason}: {uid} removed (Invalid for {mode})", "WARNING")

        if self.screen_manager.current == 'transaction_page' and self.mode == mode:
            table = self.screen_manager.get_screen('transaction_page').ids.table_content
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            set_invalid = set(str(x) for x in invalid_uids)
            set_reported = set(str(x) for x in reported_uids)

            for row in list(table.children):
                uid = str(getattr(row, 'rfid_tag', ""))
                if not uid: continue

                if uid in set_invalid:
                    table.remove_widget(row)
                    self.all_items_cache = [item for item in self.all_items_cache if
str(item[1]) != uid]

                elif uid in set_reported:
                    if getattr(row, 'base_color', None) != [1, 0.85, 0.85, 1]:
                        row.current_status = "REPORTED"
                        row.base_color = [1, 0.85, 0.85, 1]
                        row.select_color = [1, 0.6, 0.6, 1]

                        if not getattr(row, 'is_selected', False):
                            row.bg_color.rgba = row.base_color

                        if not getattr(row, 'is_selected', False):
                            row.bg_color.rgba = row.base_color

                        for i, item in enumerate(self.all_items_cache):
                            if str(item[1]) == uid:
                                temp = list(item)
                                while len(temp) < 10: temp.append("")
                                temp[9] = "REPORTED"
                                self.all_items_cache[i] = tuple(temp)

    def update_status(self, message, color):
        self.screen_manager.get_screen('main_menu').ids.status_label.text = message
        self.screen_manager.get_screen('main_menu').ids.status_label.color = color

    def open_transaction(self, mode):
        self.mode = mode
        trans_screen = self.screen_manager.get_screen('transaction_page')

        self.all_items_cache.clear()
        self.last_report_count = 0
        trans_screen.ids.table_content.clear_widgets()
        trans_screen.ids.global_doc_id.text = ""
        trans_screen.ids.log_text.text = self.log_history[mode]

        self.ignored_uids.clear()

        if mode == 'IN':
            trans_screen.ids.nav_title.text = 'WAREHOUSE RECEIVING SYSTEM'
            trans_screen.ids.global_doc_label.text = 'MAT. DOCUMENT:'
            trans_screen.ids.global_doc_id.hint_text = 'Enter Inbound...'
            trans_screen.ids.header_box.canvas.before.children[0].rgba = [220/255, 20/255,
60/255, 1]
            trans_screen.ids.btn_post_selected.background_color = [44/255, 62/255, 80/255, 1]
            trans_screen.ids.btn_post_all.background_color = [44/255, 62/255, 80/255, 1]

            trans_screen.ids.btn_post_selected.text = 'POST IN SELECTED'
            trans_screen.ids.btn_post_all.text = 'POST IN ALL'

        else:
            trans_screen.ids.nav_title.text = 'WAREHOUSE OUTGOING SYSTEM'
            trans_screen.ids.global_doc_label.text = 'MAT. DOCUMENT:'
            trans_screen.ids.global_doc_id.hint_text = 'Enter Outbound...'
            trans_screen.ids.header_box.canvas.before.children[0].rgba = [220/255, 20/255,
60/255, 1]
            trans_screen.ids.btn_post_selected.background_color = [44/255, 62/255, 80/255, 1]
            trans_screen.ids.btn_post_all.background_color = [44/255, 62/255, 80/255, 1]

            trans_screen.ids.btn_post_selected.text = 'POST OUT SELECTED'
            trans_screen.ids.btn_post_all.text = 'POST OUT ALL'
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        self.current_prefix_list = []
        self.selected_rows.clear()
        self.last_selected_row = None
        self.set_edit_mode(False)
        self.screen_manager.current = 'transaction_page'
        self._update_trans_count()

        if self.transaction_refresh_event:
            self.transaction_refresh_event.cancel()
        self.transaction_refresh_event =
Clock.schedule_interval(self._refresh_transaction_scans, 3)

        self.outbound_instructions = {}
        ids = trans_screen.ids

        if not hasattr(self, 'saved_panel_in'):
            self.saved_panel_in = ids.panel_inbound.__self__ if hasattr(ids.panel_inbound,
'__self__') else ids.panel_inbound
            self.saved_panel_out = ids.panel_outbound.__self__ if hasattr(ids.panel_outbound,
'__self__') else ids.panel_outbound
            self.panel_container = self.saved_panel_in.parent

        self.panel_container.clear_widgets()

        if mode == 'IN':
            self.panel_container.add_widget(self.saved_panel_in)
        else:
            self.panel_container.add_widget(self.saved_panel_out)
            self.outbound_issue_mode = 'FULL'
            Clock.schedule_once(lambda dt: self.toggle_issue_type('FULL', force=True), 0.1)

    def open_shortcut_guide(self):
        if not hasattr(self, 'shortcut_popup') or not self.shortcut_popup:
            self.shortcut_popup = Factory.ShortcutGuidePopup()
        self.shortcut_popup.open()

    def _on_key_down(self, window, keycode, scancode, codepoint, modifiers):
        if self.exit_popup and self.exit_popup.parent:
            if keycode == 27:
                self.exit_popup.dismiss()
                return True
            if keycode in [275, 276, 9]:
                self.exit_popup.toggle_selection()
                return True
            if keycode == 13:
                self.exit_popup.confirm()
                return True
            return True

        if hasattr(self, 'shortcut_popup') and self.shortcut_popup and
self.shortcut_popup.parent:
            if keycode == 27:
                self.shortcut_popup.dismiss()
                return True
            return True

        if hasattr(self, 'confirm_exit_so_popup') and self.confirm_exit_so_popup and
self.confirm_exit_so_popup.parent:
            if keycode == 27:
                self.confirm_exit_so_popup.dismiss()
                return True
            if keycode in [275, 276, 9]:
                self.confirm_exit_so_popup.toggle_selection()
                return True
            if keycode == 13:
                self.confirm_exit_so_popup.confirm()
                return True
            return True

        if hasattr(self, 'trans_report_popup') and self.trans_report_popup and
self.trans_report_popup.parent:
            if keycode == 27:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                self.trans_report_popup.dismiss(); return True

            if keycode == 127:
                self.delete_selected_basket_items(); return True

            if keycode == 97 and 'ctrl' in modifiers:
                content = self.trans_report_popup.ids.review_list_content
                for row in content.children:
                    if isinstance(row, ReviewReportRow):
                        row.is_selected = True
                        row.bg_color.rgba = [0.8, 0.9, 1, 1]
                return True

            if keycode in [273, 274]:
                content = self.trans_report_popup.ids.review_list_content
                rows = [c for c in reversed(content.children) if isinstance(c, ReviewReportRow)]
                if not rows: return True

                last_idx = -1
                for i, r in enumerate(rows):
                    if r.is_selected: last_idx = i

                target_idx = min(len(rows)-1, max(0, last_idx + (1 if keycode == 274 else -1)))
                if 'ctrl' not in modifiers and 'shift' not in modifiers:
                    for r in rows:
                        r.is_selected = False
                        r.bg_color.rgba = [1, 1, 1, 1]

                rows[target_idx].is_selected = True
                rows[target_idx].bg_color.rgba = [0.8, 0.9, 1, 1]
                return True

            return False

        if hasattr(self, 'pwd_popup') and self.pwd_popup and self.pwd_popup.parent:
            if keycode == 27:
                self.pwd_popup.dismiss()
                return True
            if keycode in [13, 271]:
                self.verify_password(self.pwd_popup.ids.pwd_input.text)
                return True
            return False

        screen = self.screen_manager.current

        if screen == 'main_menu':
            if keycode == 27:
                self.handle_close_request()
                return True
            return self._nav_main_menu(keycode)

        if screen == 'transaction_page':
            return self._nav_transaction(keycode, codepoint, modifiers)

        if screen == 'stock_opname_page':
            return self._nav_so_main(keycode, codepoint, modifiers)

        if screen == 'so_detail_page':
            return self._nav_so_detail(keycode, codepoint, modifiers)

        if screen == 'stock_page':
            return self._nav_stock(keycode, codepoint, modifiers)

        return False

    def _nav_transaction(self, keycode, codepoint, modifiers):
        ids = self.screen_manager.get_screen('transaction_page').ids
        if self.mode == 'IN':
            inputs = [ids.adj_mat, ids.adj_desc, ids.adj_batch, ids.adj_qty, ids.adj_sloc,
ids.adj_sbin_h, ids.adj_sbin_n]
        else:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            inputs = [ids.out_issue_qty, ids.out_rem_sloc, ids.out_rem_sbin_h,
ids.out_rem_sbin_n]

        focus_widgets = [ids.input_add, ids.input_search, ids.global_doc_id, ids.sap_input] +
inputs
        is_typing = any(w.focus for w in focus_widgets)
        active_widget = next((w for w in focus_widgets if w.focus), None)

        dropdown_open = False
        if active_widget and hasattr(active_widget, 'dropdown') and
getattr(active_widget.dropdown, 'parent', None):
            dropdown_open = True

        spinner_open = False
        spinner_dd = getattr(ids.filter_spinner, '_dropdown', None) if hasattr(ids,
'filter_spinner') else None
        if spinner_dd and spinner_dd.parent:
            spinner_open = True

        if keycode == 27:
            if spinner_open: spinner_dd.dismiss(); return True
            if dropdown_open: active_widget.dropdown.dismiss(); return True
            if self.edit_mode:
                self.clear_adjustment_inputs()
                self.set_edit_mode(False)
            elif is_typing:
                for w in focus_widgets: w.focus = False
            elif self.selected_rows:
                self.unselect_all()
            else:
                self.screen_manager.current = 'main_menu'
            return True

        if keycode == 32 and 'ctrl' in modifiers and self.mode == 'OUT' and getattr(self,
'edit_mode', False):
            new_mode = 'PARTIAL' if getattr(self, 'outbound_issue_mode', 'FULL') == 'FULL' else
'FULL'
            self.toggle_issue_type(new_mode)
            return True

        if keycode == 13:
            if spinner_open:
                buttons = [c for c in reversed(spinner_dd.container.children) if isinstance(c,
Button)]
                idx = getattr(ids.filter_spinner, 'dropdown_index', -1)
                if 0 <= idx < len(buttons): buttons[idx].dispatch('on_release')
                return True

            if dropdown_open:
                if hasattr(active_widget, 'dropdown_index') and active_widget.dropdown_index !=
-1:
                    buttons = [c for c in reversed(active_widget.dropdown.container.children) if
isinstance(c, Button)]
                    if 0 <= active_widget.dropdown_index < len(buttons):
                        buttons[active_widget.dropdown_index].dispatch('on_release')
                        return True

            if 'ctrl' in modifiers:
                if self.edit_mode:
                    if self.mode == 'IN': self.save_adjustment()
                    else: self.save_outbound_settings()
                else:
                    if self.selected_rows: self.set_edit_mode(True)
                return True

            if self.edit_mode:
                active_idx = next((i for i, w in enumerate(inputs) if w.focus), -1)

                if self.mode == 'IN' and active_idx in [4, 5, 6] and not dropdown_open:
                    active_widget = inputs[active_idx]
                    if active_idx == 4: self.filter_sloc_dropdown(active_widget.text)
                    elif active_idx == 5: self.filter_sbin_h_dropdown(active_widget.text)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    elif active_idx == 6: self.filter_sbin_n_dropdown(active_widget.text)
                    return True

                if active_idx != -1 and active_idx < len(inputs) - 1:
                    inputs[active_idx + 1].focus = True
                return True

            is_typing = False
            for widget in Window.children[0].walk():
                if isinstance(widget, TextInput) and widget.focus:
                    is_typing = True
                    break

            if not is_typing:
                ids.input_add.focus = True
                return True

        if 'ctrl' in modifiers and keycode == 97 and not is_typing:
            self.select_all_rows(); return True

        if keycode == 127 and not self.edit_mode and not is_typing:
            self.delete_selected_rows(); return True

        if keycode in [273, 274]:
            if 'ctrl' in modifiers and hasattr(ids, 'filter_spinner'):
                vals = ids.filter_spinner.values
                if vals:
                    try: c_idx = vals.index(ids.filter_spinner.text)
                    except: c_idx = 0
                    n_idx = max(0, min(len(vals)-1, c_idx + (1 if keycode == 274 else -1)))
                    ids.filter_spinner.text = vals[n_idx]
                return True

            if spinner_open:
                buttons = [c for c in reversed(spinner_dd.container.children) if isinstance(c,
Button)]
                if not buttons: return True

                if not hasattr(ids.filter_spinner, 'dropdown_index'):
                    ids.filter_spinner.dropdown_index = -1

                idx = ids.filter_spinner.dropdown_index
                if 0 <= idx < len(buttons):
                    buttons[idx].background_color = [1, 1, 1, 1]

                idx = max(0, min(len(buttons)-1, idx + (1 if keycode == 274 else -1)))
                ids.filter_spinner.dropdown_index = idx

                buttons[idx].background_color = [0.6, 0.8, 1, 1]
                spinner_dd.scroll_to(buttons[idx])
                return True

            if dropdown_open:
                buttons = [c for c in reversed(active_widget.dropdown.container.children) if
isinstance(c, Button)]
                if not buttons: return True
                if not hasattr(active_widget, 'dropdown_index'): active_widget.dropdown_index =
-1
                if 0 <= active_widget.dropdown_index < len(buttons):
buttons[active_widget.dropdown_index].background_color = [0.9, 0.9, 0.9, 1]

                active_widget.dropdown_index += (1 if keycode == 274 else -1)
                active_widget.dropdown_index = max(0, min(len(buttons)-1,
active_widget.dropdown_index))

                buttons[active_widget.dropdown_index].background_color = [0.6, 0.8, 1, 1]
                active_widget.dropdown.scroll_to(buttons[active_widget.dropdown_index])
                return True
            elif self.edit_mode:
                active_idx = next((i for i, w in enumerate(inputs) if w.focus), -1)
                if active_idx != -1:
                    target = active_idx + (1 if keycode == 274 else -1)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    if 0 <= target < len(inputs): inputs[target].focus = True
                return True
            else:
                return self._scroll_table_selection(ids.table_content, ids.table_scroll,
keycode)

        forbidden_mods = ['ctrl', 'alt', 'meta']
        has_forbidden_mod = any(m in modifiers for m in forbidden_mods)

        if codepoint and codepoint.isprintable() and not has_forbidden_mod and not
self.edit_mode:
            if ids.input_search.focus or ids.global_doc_id.focus or ids.sap_input.focus:
                return False

            if not ids.input_add.focus:
                ids.input_add.focus = True
                ids.input_add.text += codepoint
                Clock.schedule_once(lambda dt: setattr(ids.input_add, 'cursor',
(len(ids.input_add.text), 0)), 0)
                return True

            return False

        return False


    def _nav_so_main(self, keycode, codepoint, modifiers):
        ids = self.screen_manager.get_screen('stock_opname_page').ids
        if keycode == 27:
            if ids.so_search.focus: ids.so_search.focus = False
            elif self.selected_rows: self.unselect_all()
            else: self.screen_manager.current = 'main_menu'
            return True

        if keycode == 13 and not ids.so_search.focus:
            self.enter_so_detail()
            return True

        if keycode in [273, 274]:
            return self._scroll_table_selection(ids.so_select_content, ids.so_select_scroll,
keycode)
        return False

    def _nav_so_detail(self, keycode, codepoint, modifiers):
        ids = self.screen_manager.get_screen('so_detail_page').ids

        if hasattr(self, 'pic_popup') and self.pic_popup and self.pic_popup.parent:
            if keycode == 27: self.pic_popup.dismiss(); return True
            return False

        if self.so_popup and self.so_popup.parent:
            if keycode == 27: self.so_popup.dismiss(); return True
            if 'ctrl' in modifiers and keycode == 13: self.confirm_so_adjust(); return True
            return False

        if hasattr(self, 'report_popup') and self.report_popup and self.report_popup.parent:
            if keycode == 27: self.report_popup.dismiss(); return True
            return False

        if keycode == 27:
            if ids.so_detail_search.focus: ids.so_detail_search.focus = False
            elif self.selected_rows: self.unselect_all()
            else: self.confirm_exit_so_detail()
            return True

        if 'ctrl' in modifiers and keycode == 13:
            self.open_so_adjust_popup()
            return True

        if 'ctrl' in modifiers and keycode == 97 and not ids.so_detail_search.focus:
            self.select_all_rows()
            return True
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        if keycode in [273, 274]:
            return self._scroll_table_selection(ids.so_detail_content, ids.so_detail_scroll,
keycode)
        return False

    def _nav_stock(self, keycode, codepoint, modifiers):
        ids = self.screen_manager.get_screen('stock_page').ids

        focus_widgets = []
        for key in ['stock_search', 'flt_rfid', 'flt_mat', 'flt_desc', 'flt_batch', 'flt_sloc',
'flt_sbin']:
            if key in ids:
                focus_widgets.append(ids[key])

        is_typing = any(w.focus for w in focus_widgets)

        if keycode == 27:
            if is_typing:
                for w in focus_widgets: w.focus = False
            elif getattr(self, 'stock_selected_rows', []):
                self.unselect_all_stock()
            else:
                self.screen_manager.current = 'main_menu'
            return True

        if keycode in [273, 274] and not is_typing:
            table = ids.stock_list_content
            scroll_view = ids.stock_list_scroll
            all_rows = [c for c in reversed(table.children) if hasattr(c, 'bg_color') and
c.height > 0]

            if not all_rows: return True

            is_down = (keycode == 274)
            is_ctrl = 'ctrl' in modifiers
            is_shift = 'shift' in modifiers

            anchor_row = getattr(self, 'stock_last_selected_row', None)
            if anchor_row not in all_rows:
                anchor_row = all_rows[0]
            anchor_idx = all_rows.index(anchor_row)

            if is_ctrl:
                target_idx = len(all_rows) - 1 if is_down else 0
            else:
                if is_shift and getattr(self, 'stock_selected_rows', []):
                    ujung_sekarang = self.stock_selected_rows[-1]
                    idx_ujung = all_rows.index(ujung_sekarang)
                    target_idx = idx_ujung + (1 if is_down else -1)
                else:
                    target_idx = anchor_idx + (1 if is_down else -1)

            target_idx = max(0, min(len(all_rows) - 1, target_idx))
            target_row = all_rows[target_idx]

            if is_shift:
                self.unselect_all_stock()
                self.stock_last_selected_row = anchor_row

                step = 1 if target_idx >= anchor_idx else -1
                for r_idx in range(anchor_idx, target_idx + step, step):
                    r = all_rows[r_idx]
                    self.stock_selected_rows.append(r)
                    r.is_selected = True
                    r.bg_color.rgba = getattr(r, 'select_color', [0.8, 0.9, 1, 1])
            else:
                self.unselect_all_stock()
                self.stock_selected_rows = [target_row]
                target_row.is_selected = True
                target_row.bg_color.rgba = getattr(target_row, 'select_color', [0.8, 0.9, 1, 1])
                self.stock_last_selected_row = target_row
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            if table.height > scroll_view.height:
                scroll_view.scroll_to(target_row)
            else:
                scroll_view.scroll_y = 1

            return True

        if 'ctrl' in modifiers and keycode == 97 and not is_typing:
            self.select_all_stock_rows()
            return True

        return False

    def unselect_all_stock(self):
        if hasattr(self, 'stock_selected_rows'):
            for r in self.stock_selected_rows:
                r.bg_color.rgba = getattr(r, 'base_color', [1, 1, 1, 1])
                r.is_selected = False
            self.stock_selected_rows.clear()
            self.stock_last_selected_row = None

    def select_all_stock_rows(self):
        content = self.screen_manager.get_screen('stock_page').ids.stock_list_content
        self.unselect_all_stock()

        self.stock_selected_rows = []
        for row in reversed(content.children):
            if isinstance(row, StockRow) and row.height > 0:
                self.stock_selected_rows.append(row)
                row.is_selected = True
                row.bg_color.rgba = row.select_color

        if self.stock_selected_rows:
            self.stock_last_selected_row = self.stock_selected_rows[-1]

    def _scroll_table_selection(self, table, scroll_view, keycode):
        all_rows = [c for c in reversed(table.children) if hasattr(c, 'bg_color') and c.height >
0]
        if not all_rows: return False

        idx = all_rows.index(self.last_selected_row) if self.last_selected_row in all_rows else
-1
        target_idx = min(len(all_rows)-1, max(0, idx + (1 if keycode == 274 else -1)))
        target_row = all_rows[target_idx]

        self.on_row_select(target_row)

        if table.height > scroll_view.height:
            scroll_view.scroll_to(target_row)
        else:
            scroll_view.scroll_y = 1

        return True

    def unselect_all(self):
        for r in self.selected_rows:
            r.bg_color.rgba = getattr(r, 'base_color', [1, 1, 1, 1])
        self.selected_rows.clear()
        self.last_selected_row = None
        if self.screen_manager.current == 'transaction_page':
            self._clear_adjustment_panel()

    def clear_stock_filters(self):
        ids = self.screen_manager.get_screen('stock_page').ids
        for key in ['flt_rfid', 'flt_mat', 'flt_desc', 'flt_batch', 'flt_sloc', 'flt_sbin']:
            if key in ids:
                ids[key].text = ""
        self.unselect_all_stock()

    def handle_bottom_edit_click(self):
        if not getattr(self, 'selected_rows', []):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            self.write_to_log("Select rows first before editing!", "WARNING")
            return

        if not getattr(self, 'edit_mode', False):
            self.set_edit_mode(True)
        else:
            self.set_edit_mode(False)

    def set_edit_mode(self, active):
        self.edit_mode = active
        if self.screen_manager.current != 'transaction_page': return
        ids = self.screen_manager.get_screen('transaction_page').ids
        from kivy.metrics import dp

        if active:
            ids.right_panel_scroll.width = dp(260)
            ids.right_panel_scroll.opacity = 1
            if hasattr(ids, 'btn_edit_selected'):
                ids.btn_edit_selected.background_color = [0.6, 0.6, 0.6, 1]
                ids.btn_edit_selected.color = [1, 1, 1, 1]
        else:
            ids.right_panel_scroll.width = 0
            ids.right_panel_scroll.opacity = 0
            if hasattr(ids, 'btn_edit_selected'):
                ids.btn_edit_selected.background_color = [255/255, 193/255, 7/255, 1]
                ids.btn_edit_selected.color = [0, 0, 0, 1]

        if self.mode == 'IN':
            inputs = [ids.adj_mat, ids.adj_desc, ids.adj_batch, ids.adj_qty, ids.adj_sloc,
ids.adj_sbin_h, ids.adj_sbin_n]
            for inp in inputs:
                inp.readonly = not active
                if not active:
                    inp.background_color = [0.9, 0.9, 0.9, 0.5]; inp.foreground_color = [0.4,
0.4, 0.4, 1]; inp.cursor_color = [0, 0, 0, 0]
                else:
                    inp.background_color = [1, 1, 1, 1]; inp.foreground_color = [0, 0, 0, 1];
inp.cursor_color = [1, 0, 0, 1]

            if active:
                ids.adj_mat.focus = True
                self.write_to_log("ENTERED EDIT MODE.", "INFO")
            else:
                for inp in inputs: inp.focus = False

        else:
            inputs = [ids.out_rem_sloc, ids.out_rem_sbin_h, ids.out_rem_sbin_n]
            ids.btn_type_full.disabled = not active
            ids.btn_type_partial.disabled = not active

            for inp in inputs:
                inp.readonly = not active
                if not active:
                    inp.background_color = [0.9, 0.9, 0.9, 0.5]; inp.foreground_color = [0.4,
0.4, 0.4, 1]; float
                else:
                    inp.background_color = [1, 1, 1, 1]; inp.foreground_color = [0, 0, 0, 1];
inp.cursor_color = [1, 0, 0, 1]

            if active:
                if getattr(self, 'outbound_issue_mode', 'FULL') == 'PARTIAL':
                    self.toggle_issue_type('PARTIAL', force=True)
                self.write_to_log("ENTERED EDIT MODE.", "INFO")
            else:
                ids.out_issue_qty.readonly = True
                ids.out_issue_qty.background_color = [0.9, 0.9, 0.9, 0.5]
                ids.out_issue_qty.cursor_color = [0, 0, 0, 0]
                ids.out_issue_qty.focus = False
                for inp in inputs: inp.focus = False

    def on_sap_paste(self, text):
        if not text: return
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        cols = text.strip().split()

        if len(cols) < 4:
            return

        if not self.selected_rows:
            self.write_to_log("Select rows first before pasting from SAP!", "WARNING")
            self.screen_manager.get_screen('transaction_page').ids.sap_input.text = ""
            return

        try:
            if cols[0].upper().startswith('R') and cols[0][1:].isdigit():
                cols = cols[1:]

            if len(cols) < 4:
                self.write_to_log("Invalid SAP format! Expected: SBIN, MAT, BATCH, QTY",
"ERROR")
                self.screen_manager.get_screen('transaction_page').ids.sap_input.text = ""
                return

            sap_sbin = cols[0].upper()
            sap_mat = cols[1].upper()
            sap_batch = cols[2].upper()
            sap_qty_str = cols[3]

            ids = self.screen_manager.get_screen('transaction_page').ids

            if not self.edit_mode:
                self.set_edit_mode(True)

            self.is_auto_filling = True

            num_selected = len(self.selected_rows)
            try:
                sap_qty_total = int(sap_qty_str)
                qty_per_colly = sap_qty_total // num_selected if num_selected > 0 else
sap_qty_total

                if num_selected > 0 and sap_qty_total % num_selected != 0:
                    self.write_to_log(f"⚠ SAP QTY ({sap_qty_total}) is not evenly divisible by
{num_selected} rows! Please adjust QTY manually.", "WARNING")
                else:
                    self.write_to_log(f"SAP QTY ({sap_qty_total}) divided by {num_selected}
({qty_per_colly} / {num_selected}).", "INFO")

            except ValueError:
                qty_per_colly = sap_qty_str

            ids.adj_mat.text = sap_mat
            ids.adj_batch.text = sap_batch
            ids.adj_qty.text = str(qty_per_colly)

            sbin_h = sap_sbin[:2] if len(sap_sbin) >= 2 else sap_sbin
            sbin_n = sap_sbin[2:] if len(sap_sbin) > 2 else ""

            ids.adj_sbin_h.text = sbin_h
            ids.adj_sbin_n.text = sbin_n

            matched_sloc = self.get_sloc_from_sbin(sbin_h)
            if matched_sloc:
                ids.adj_sloc.text = matched_sloc
                self.on_sloc_change(None, matched_sloc)
            else:
                ids.adj_sloc.text = ""

            self.is_auto_filling = False

            self.fetch_catalog_data(sap_mat)

        except Exception as e:
            self.write_to_log(f"Paste Error: {str(e)}", "ERROR")
            self.is_auto_filling = False
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        finally:
            self.screen_manager.get_screen('transaction_page').ids.sap_input.text = ""

    def fetch_catalog_data(self, material_num):
        mat = material_num.strip().upper()
        ids = self.screen_manager.get_screen('transaction_page').ids

        if not mat:
            ids.adj_desc.text = ""
            return

        current_desc = ids.adj_desc.text
        if getattr(self, 'last_fetched_mat', '') == mat and current_desc not in ["",
"Loading..."]:
            return

        if not getattr(self, 'edit_mode', False) or getattr(self, 'is_auto_filling', False):
            return

        self.last_fetched_mat = mat

        self.is_auto_filling = True
        ids.adj_desc.text = "Loading..."
        self.is_auto_filling = False

        threading.Thread(target=self._thread_fetch_catalog, args=(mat,), daemon=True).start()

    def _thread_fetch_catalog(self, material_num):
        import time
        max_retries = 3
        last_err = ""

        for attempt in range(max_retries):
            try:
                res =
self.supabase.table("MATERIAL_CATALOG").select("DESCRIPTION").eq("MATERIAL",
material_num).execute()

                Clock.schedule_once(lambda dt: self._apply_catalog_data(res.data))
                return

            except Exception as e:
                last_err = str(e)
                time.sleep(1)

        Clock.schedule_once(lambda dt: self._apply_catalog_data(None))
        Clock.schedule_once(lambda dt, err=last_err: self.write_to_log(f"Catalog Error
(Network): {err[:50]}...", "ERROR"))

    def _apply_catalog_data(self, data):
        ids = self.screen_manager.get_screen('transaction_page').ids
        self.is_auto_filling = True

        if not data:
            self.write_to_log("Material NOT FOUND in Catalog. Please type manually.", "WARNING")
            ids.adj_desc.text = ""
        else:
            ids.adj_desc.text = data[0].get("DESCRIPTION", "")

        self.is_auto_filling = False

    def _thread_update_catalog_learning(self, payload):
        try:
            for p in payload:
                mat = str(p.get("MATERIAL", "")).strip().upper()
                desc = str(p.get("DESCRIPTION", "")).strip().upper()

                if not mat or not desc: continue
                if mat == "UNKNOWN": continue

                res = self.supabase.table("MATERIAL_CATALOG").select("MATERIAL").eq("MATERIAL",
mat).execute()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                if not res.data:
                    try:
                        self.supabase.table("MATERIAL_CATALOG").delete().eq("MATERIAL",
mat).execute()
                    except:
                        pass

                    self.supabase.table("MATERIAL_CATALOG").insert({
                        "MATERIAL": mat,
                        "DESCRIPTION": desc
                    }).execute()

                    msg = f"Learned new catalog entry: {mat}"
                    Clock.schedule_once(lambda dt, m=msg: self.write_to_log(m, "SUCCESS"))

        except Exception as e:
            pass

    def filter_table(self, query):
        query = query.lower()
        trans_screen = self.screen_manager.get_screen('transaction_page')
        table = trans_screen.ids.table_content
        table.clear_widgets()
        self.selected_rows.clear()
        self.last_selected_row = None
        for data in self.all_items_cache:
            string_data = " ".join([str(val).lower() for val in data])
            if query in string_data:
                new_row = TableRow(no=data[0], rfid_tag=data[1], row_data=data[2:],
app_instance=self)
                table.add_widget(new_row)

    def _nav_main_menu(self, keycode):
        if not self.screen_manager.has_screen('main_menu'):
            return False

        ids = self.screen_manager.get_screen('main_menu').ids
        if not all(k in ids for k in ['btn_RECEIVING', 'btn_so', 'btn_OUTGOING', 'btn_stock']):
            return False

        buttons = [ids.btn_RECEIVING, ids.btn_so, ids.btn_OUTGOING, ids.btn_stock]
        colors = [
            {'base': [220/255, 20/255, 60/255, 1], 'light': [120/255, 0/255, 20/255, 1]},
            {'base': [220/255, 20/255, 60/255, 1], 'light': [120/255, 0/255, 20/255, 1]},
            {'base': [220/255, 20/255, 60/255, 1], 'light': [120/255, 0/255, 20/255, 1]},
            {'base': [44/255, 62/255, 80/255, 1], 'light': [24/255, 42/255, 60/255, 1]}
        ]
        if keycode is None:
            for i, btn in enumerate(buttons):
                btn.background_color = colors[i]['light'] if i == self.main_menu_index else
colors[i]['base']
            return True

        if keycode in [13, 271]:
            buttons[self.main_menu_index].dispatch('on_release')
            return True

        if keycode in [273, 274]:
            buttons[self.main_menu_index].background_color =
colors[self.main_menu_index]['base']

            if keycode == 274:
                self.main_menu_index = (self.main_menu_index + 1) % len(buttons)
            else:
                self.main_menu_index = (self.main_menu_index - 1) % len(buttons)

            buttons[self.main_menu_index].background_color =
colors[self.main_menu_index]['light']
            return True

        return False
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
    def _load_doc_history(self):
        import os, json
        self.doc_id_history = []
        try:
            if os.path.exists("doc_history.json"):
                with open("doc_history.json", "r") as f:
                    self.doc_id_history = json.load(f)
        except: pass

    def _save_doc_history(self):
        import json
        try:
            with open("doc_history.json", "w") as f:
                json.dump(self.doc_id_history, f)
        except: pass

    def add_to_doc_history(self, doc_id):
        if not doc_id: return

        from datetime import datetime
        timestamp_now = datetime.now().strftime("%H:%M:%S")

        clean_history = []
        for h in self.doc_id_history:
            if isinstance(h, dict) and h.get("doc_id") == doc_id:
                continue
            elif isinstance(h, str) and h == doc_id:
                continue
            clean_history.append(h)

        self.doc_id_history = clean_history

        self.doc_id_history.insert(0, {"doc_id": doc_id, "time": timestamp_now})

        self.doc_id_history = self.doc_id_history[:3]
        self._save_doc_history()

    def show_doc_id_dropdown(self, widget):
        if self.doc_id_history:
            self.filter_doc_id_history(widget.text)

    def filter_doc_id_history(self, text):
        ids = self.screen_manager.get_screen('transaction_page').ids
        widget = ids.global_doc_id

        if not hasattr(widget, 'dropdown') or widget.dropdown is None:
            widget.dropdown = DropDown(auto_dismiss=True, max_height=200)

        widget.dropdown.clear_widgets()
        widget.dropdown_index = -1

        query = text.strip().upper()

        valid_history = []
        for h in self.doc_id_history:
            if isinstance(h, dict):
                valid_history.append(h)
            elif isinstance(h, str):
                valid_history.append({"doc_id": str(h), "time": "--:--:--"})

        filtered = [h for h in valid_history if query in h["doc_id"].upper()] if query else
valid_history

        if not filtered:
            widget.dropdown.dismiss()
            return

        for h in filtered:
            display_text = f"{h['doc_id']}   [color=666666](Last: {h['time']})[/color]"
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            btn = Button(text=display_text, size_hint_y=None, height='40dp', markup=True,
                         background_normal='', background_color=[0.95,0.95,0.95,1],
color=[0,0,0,1])

            btn.bind(on_release=lambda btn_obj, target_id=h['doc_id']:
self._select_doc_id(widget, target_id))
            widget.dropdown.add_widget(btn)

        if not widget.dropdown.parent and widget.focus:
            widget.dropdown.open(widget)

    def _select_doc_id(self, widget, target_id):
        widget.text = target_id
        widget.dropdown.dismiss()

    def write_to_log(self, message, tipe="INFO"):
        if not hasattr(self, 'mode') or self.mode not in getattr(self, 'log_history', {}):
            return

        msg_lower = message.lower()
        if any(keyword in msg_lower for keyword in ["timeout", "connection", "network", "host",
"ssl", "api"]):
            tipe = "WARNING"
            message = "Network latency detected. Synchronizing request in background..."

            history_lines = self.log_history[self.mode].split('\n')
            last_log = history_lines[-2] if len(history_lines) > 1 else ""
            if "Network latency" in last_log:
                return

        symbol = {"ERROR": "[❌ ERROR]", "SUCCESS": "[✔ SUCCESS]", "WARNING": "[!
WARNING]"}.get(tipe, "[🛈 INFO]")
        new_log = f"{datetime.now().strftime('%H:%M:%S')} {symbol} {message}\n"

        self.log_history[self.mode] += new_log

        if len(self.log_history[self.mode]) > 3000:
            self.log_history[self.mode] = self.log_history[self.mode][-3000:]

        if self.screen_manager.current == 'transaction_page':
            trans_screen = self.screen_manager.get_screen('transaction_page')
            trans_screen.ids.log_text.text = self.log_history[self.mode]
            Clock.schedule_once(lambda dt: setattr(trans_screen.ids.log_scroll, 'scroll_y', 0),
0.1)

    def process_scan_in(self):
        trans_screen = self.screen_manager.get_screen('transaction_page')
        uid = trans_screen.ids.input_add.text.strip().upper()
        if not uid or self.check_duplicate(uid): return
        trans_screen.ids.input_add.text = ""
        self.write_to_log(f"Processing RFID: {uid}...", "INFO")

        dummy_row = {'RFID_UID': uid}
        threading.Thread(target=self._thread_scan_in, args=(dummy_row,), daemon=True).start()

    def _thread_scan_in(self, row_db):
        uid = row_db.get('RFID_UID') if isinstance(row_db, dict) else row_db
        if isinstance(row_db, str): row_db = {'RFID_UID': uid}

        db_mat = row_db.get('MATERIAL') or ""
        db_desc = row_db.get('DESCRIPTION') or ""
        db_qty = str(row_db.get('QTY') or "0")
        db_sloc = row_db.get('SLOC') or ""
        db_sbin = row_db.get('SBIN') or ""
        db_batch = row_db.get('BATCH') or ""

        try:
            stock_check = self.supabase.table("STOCK").select("*").eq("RFID_UID", uid).execute()

            if self.mode == "IN":
                if stock_check.data:
                    if uid not in self.ignored_uids:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                        Clock.schedule_once(lambda dt, u=uid: self.write_to_log(f"RFID {u} is
already in STOCK! Auto-cleared.", "WARNING"))
                        self.ignored_uids.add(uid)
                        self.supabase.table("SCAN_IN").delete().eq("PC_TOKEN",
self.pc_token).eq("RFID_UID", uid).execute()
                    return

                if db_mat and db_mat != "UNKNOWN":
                    row_data = (db_mat, db_desc, db_qty, db_sloc, db_sbin, db_batch)
                    Clock.schedule_once(lambda dt, u=uid: self.write_to_log(f"Scanned IN: {u}
(Data from Handheld)", "SUCCESS"))
                else:
                    master_check = self.supabase.table("MASTER_DATA").select("MATERIAL,
DESCRIPTION, QTY").eq("RFID_UID", uid).execute()
                    if not master_check.data:
                        row_data = ("", "", "", "", "", "")
                    else:
                        row = master_check.data[0]
                        row_data = (row.get('MATERIAL', ''), row.get('DESCRIPTION', ''),
str(row.get('QTY', '')), "", "", "")
                    Clock.schedule_once(lambda dt, u=uid: self.write_to_log(f"Scanned IN: {u}",
"SUCCESS"))

            else:
                if not stock_check.data:
                    if uid not in self.ignored_uids:
                        Clock.schedule_once(lambda dt, u=uid: self.write_to_log(f"RFID {u} NOT
IN STOCK! Auto-cleared.", "ERROR"))
                        self.ignored_uids.add(uid)
                        self.supabase.table("SCAN_OUT").delete().eq("PC_TOKEN",
self.pc_token).eq("RFID_UID", uid).execute()
                    return

                row = stock_check.data[0]
                final_mat = db_mat if db_mat else row.get('MATERIAL', '')
                final_desc = db_desc if db_desc else row.get('DESCRIPTION', '')
                final_qty = db_qty if db_qty != "0" else str(row.get('QTY', '0'))
                final_sloc = db_sloc if db_sloc else row.get('SLOC', '')
                final_sbin = db_sbin if db_sbin else row.get('SBIN', '')
                final_batch = db_batch if db_batch else row.get('BATCH', '')

                row_data = (final_mat, final_desc, final_qty, final_sloc, final_sbin,
final_batch)

                qty_stock = int(row.get('QTY', 0))
                db_status = row_db.get('STATUS', 'OK')
                is_full = (db_status != 'PARTIAL')

                issue_qty = int(final_qty) if final_qty.isdigit() else qty_stock
                rem_qty = qty_stock - issue_qty if not is_full else 0

                self.outbound_instructions[uid] = {
                    'type': 'FULL' if is_full else 'PARTIAL',
                    'orig_qty': qty_stock, 'issue_qty': issue_qty,
                    'rem_qty': rem_qty, 'rem_sloc': final_sloc, 'rem_sbin': final_sbin
                }
                Clock.schedule_once(lambda dt, u=uid: self.write_to_log(f"Scanned OUT: {u}",
"SUCCESS"))

            Clock.schedule_once(lambda dt, u=uid, rd=row_data: self.add_to_tree(u, rd))

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"DB ERROR: {err_msg}", "ERROR"))

    def _refresh_transaction_scans(self, dt):
        if self.screen_manager.current != 'transaction_page':
            if self.transaction_refresh_event:
                self.transaction_refresh_event.cancel()
            return
        threading.Thread(target=self._thread_poll_hardware_scanners, daemon=True).start()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
    def _thread_poll_hardware_scanners(self):
        try:
            table_name = "SCAN_IN" if self.mode == "IN" else "SCAN_OUT"
            res = self.supabase.table(table_name).select("*").eq("PC_TOKEN",
self.pc_token).limit(99999).execute()
            if not res.data: return

            existing_uids = [str(item[1]) for item in self.all_items_cache]
            to_process = []
            to_delete_instantly = []
            updates_for_existing = []

            for row in res.data:
                uid = row['RFID_UID']
                status = row.get('STATUS', 'OK')

                if (self.mode == "IN" and status == "IN STOCK") or \
                   (self.mode == "OUT" and status == "NOT IN STOCK"):
                    if uid not in self.ignored_uids:
                        to_delete_instantly.append(uid)
                elif uid not in existing_uids and uid not in self.ignored_uids:
                    to_process.append(row)
                elif uid in existing_uids:
                    updates_for_existing.append(row)

            if to_delete_instantly:
                self.supabase.table(table_name).delete().in_("RFID_UID",
to_delete_instantly).eq("PC_TOKEN", self.pc_token).execute()
                for uid in to_delete_instantly:
                    msg = f"RFID {uid} REJECTED ({status})"
                    Clock.schedule_once(lambda dt, m=msg: self.write_to_log(m, "WARNING"))
                    self.ignored_uids.add(uid)

            for row_db in to_process:
                self._thread_scan_in(row_db)

            if updates_for_existing:
                Clock.schedule_once(lambda dt: self._sync_existing_items(updates_for_existing))

            target_report_table = "REPORT_IN" if self.mode == "IN" else "REPORT_OUT"
            report_res =
self.supabase.table(target_report_table).select("REPORT_CODE").like("REPORT_CODE",
f"%{self.pc_token[-3:]}%").execute()

            if report_res.data is not None:
                current_report_count = len(report_res.data)
                if current_report_count > getattr(self, 'last_report_count', 0):
                    handheld_reports = [r for r in report_res.data if "-PC" not in
r.get("REPORT_CODE", "")]
                    hh_count = len(handheld_reports)

                    if hh_count > 0:
                        def _trigger_alert(dt, count=hh_count):
                            ids = self.screen_manager.get_screen('transaction_page').ids
                            if hasattr(ids, 'btn_report_center'):
                                ids.btn_report_center.text = f'REPORT ({count} NEW!)'
                                ids.btn_report_center.background_color = [255/255, 140/255, 0,
1]
                            self.write_to_log("New field report received from Scanner!",
"WARNING")

                        Clock.schedule_once(_trigger_alert)

                self.last_report_count = current_report_count

        except Exception as e:
            pass

    def _sync_existing_items(self, updates):
        table = self.screen_manager.get_screen('transaction_page').ids.table_content
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        for row_db in updates:
            uid = row_db['RFID_UID']
            db_status = row_db.get('STATUS', 'OK')

            db_mat = row_db.get('MATERIAL') or ""
            db_desc = row_db.get('DESCRIPTION') or ""
            db_qty = str(row_db.get('QTY') or "0")
            db_sloc = row_db.get('SLOC') or ""
            db_sbin = row_db.get('SBIN') or ""
            db_batch = row_db.get('BATCH') or ""

            for i, cache_item in enumerate(self.all_items_cache):
                if str(cache_item[1]) == uid:
                    old_mat = str(cache_item[2]).strip().upper()
                    old_qty = str(cache_item[4]).strip()
                    old_sloc = str(cache_item[5]).strip().upper()
                    old_sbin = str(cache_item[6]).strip().upper()
                    old_status = str(cache_item[8] if len(cache_item) > 8 else "")

                    if old_status == "REPORTED" and db_status != "REPORTED":
                        db_status = "REPORTED"

                    is_changed = False
                    if db_mat and db_mat.strip().upper() != "UNKNOWN" and db_mat.strip().upper()
!= old_mat: is_changed = True
                    if db_qty != "0" and db_qty != old_qty: is_changed = True
                    if db_sloc and db_sloc.strip().upper() != old_sloc: is_changed = True
                    if db_sbin and db_sbin.strip().upper() != old_sbin: is_changed = True
                    if db_status != old_status: is_changed = True

                    if is_changed:
                        new_mat = db_mat if db_mat else cache_item[2]
                        new_desc = db_desc if db_desc else cache_item[3]
                        new_qty = db_qty if db_qty != "0" else old_qty
                        new_sloc = db_sloc if db_sloc else cache_item[5]
                        new_sbin = db_sbin if db_sbin else cache_item[6]
                        new_batch = db_batch if db_batch else cache_item[7]
                        new_item = list(cache_item)
                        new_item[2] = new_mat
                        new_item[3] = new_desc
                        new_item[4] = new_qty
                        new_item[5] = new_sloc
                        new_item[6] = new_sbin
                        new_item[7] = new_batch
                        while len(new_item) < 9: new_item.append("")
                        new_item[8] = db_status
                        self.all_items_cache[i] = tuple(new_item)

                        if self.mode == "OUT":
                            qty_stock = self.outbound_instructions[uid]['orig_qty'] if uid in
self.outbound_instructions else int(new_qty)
                            issue_qty = int(new_qty) if str(new_qty).isdigit() else qty_stock
                            is_full = (db_status != 'PARTIAL')
                            rem_qty = qty_stock - issue_qty if not is_full else 0

                            self.outbound_instructions[uid] = {
                                'type': 'FULL' if is_full else 'PARTIAL',
                                'orig_qty': qty_stock, 'issue_qty': issue_qty,
                                'rem_qty': rem_qty, 'rem_sloc': new_sloc, 'rem_sbin': new_sbin
                            }

                        for row_ui in table.children:
                            if getattr(row_ui, 'rfid_tag', None) == uid:
                                row_ui.row_data = (new_mat, new_desc, new_qty, new_sloc,
new_sbin, new_batch)
                                row_ui.lbl_mat.text = new_mat
                                row_ui.lbl_desc.text = new_desc
                                row_ui.lbl_qty.text = new_qty
                                row_ui.lbl_sloc.text = new_sloc
                                row_ui.lbl_sbin.text = new_sbin
                                row_ui.lbl_batch.text = new_batch
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                                is_complete = all([new_mat, new_desc, new_qty, new_sloc,
new_sbin, new_batch]) and str(new_qty) != "0" and new_mat != "UNKNOWN"

                                if db_status == "REPORTED":
                                    row_ui.current_status = "REPORTED"
                                    row_ui.base_color = [1, 0.85, 0.85, 1]; row_ui.select_color
= [1, 0.6, 0.6, 1]
                                elif new_sloc == "TR" or new_sbin.startswith("TR"):
                                    row_ui.current_status = "TRANSIT"
                                    row_ui.base_color = [1, 0.96, 0.7, 1]; row_ui.select_color =
[1, 0.8, 0.2, 1]
                                elif is_complete:
                                    row_ui.current_status = "READY"
                                    row_ui.base_color = [0.85, 1, 0.85, 1]; row_ui.select_color
= [0.6, 0.9, 0.6, 1]
                                else:
                                    row_ui.current_status = "INCOMPLETE"
                                    row_ui.base_color = [1, 1, 1, 1]; row_ui.select_color =
[0.6, 0.8, 1, 1]

                                if not getattr(row_ui, 'is_selected', False):
                                    row_ui.bg_color.rgba = row_ui.base_color

                                if getattr(row_ui, 'is_selected', False) and self.selected_rows
and self.selected_rows[0] == row_ui:
                                    Clock.schedule_once(lambda dt, r=row_ui:
self.fill_adjustment_panel(r))
                                break
                    break

    def check_duplicate(self, uid):
        if any(str(data[1]) == str(uid) for data in self.all_items_cache):
            self.write_to_log(f"Rejected: {uid} ALREADY IN LIST", "WARNING")
            return True
        return False

    def add_to_tree(self, uid, row_data):
        if any(str(data[1]) == str(uid) for data in self.all_items_cache):
            return

        no = len(self.all_items_cache) + 1
        if len(row_data) < 6:
            row_data = list(row_data)
            row_data.append("READY")
            row_data = tuple(row_data)

        full_data = (no, uid, *row_data)
        self.all_items_cache.append(full_data)

        search_query =
self.screen_manager.get_screen('transaction_page').ids.input_search.text.lower()
        if not search_query or search_query in " ".join([str(val).lower() for val in
full_data]):
            new_row = TableRow(no=no, rfid_tag=uid, row_data=row_data, app_instance=self)
            self.screen_manager.get_screen('transaction_page').ids.table_content.add_widget(new_
row)

        self._update_trans_count()

    def _recalculate_row_numbers(self):
        if self.screen_manager.current != 'transaction_page': return

        for i in range(len(self.all_items_cache)):
            item = list(self.all_items_cache[i])
            item[0] = i + 1
            self.all_items_cache[i] = tuple(item)

        table = self.screen_manager.get_screen('transaction_page').ids.table_content

        no_urut = 1
        for row in reversed(table.children):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            if isinstance(row, TableRow) and hasattr(row, 'lbl_no'):
                row.lbl_no.text = str(no_urut)
                no_urut += 1

        self._update_trans_count()

    def _update_trans_count(self):
        if self.screen_manager.has_screen('transaction_page'):
            count = len(self.all_items_cache)
            self.screen_manager.get_screen('transaction_page').ids.lbl_trans_count.text =
f'Scanned Tag List (Total: {count})'

    def on_row_select(self, row_widget):
        table = row_widget.parent
        if not table: return

        all_rows = [c for c in reversed(table.children) if hasattr(c, 'bg_color')]
        modifiers = Window.modifiers

        if isinstance(row_widget, SORowSelection):
            if row_widget in self.selected_rows:
                self.selected_rows.remove(row_widget)
                row_widget.bg_color.rgba = [1, 1, 1, 1]
            else:
                if len(self.selected_rows) >= 2:
                    self.write_to_log("Max 2 Bins selected!", "WARNING")
                    return
                self.selected_rows.append(row_widget)
                row_widget.bg_color.rgba = [0.8, 0.9, 1, 1]
            self.last_selected_row = row_widget
            return

        if 'ctrl' in modifiers:
            if row_widget in self.selected_rows:
                self.selected_rows.remove(row_widget)
                row_widget.bg_color.rgba = getattr(row_widget, 'base_color', [1, 1, 1, 1])
                row_widget.is_selected = False
            else:
                self.selected_rows.append(row_widget)
                row_widget.bg_color.rgba = getattr(row_widget, 'select_color', [0.8, 0.9, 1, 1])
                row_widget.is_selected = True
            self.last_selected_row = row_widget
        elif 'shift' in modifiers and self.last_selected_row in all_rows:
            idx1, idx2 = all_rows.index(self.last_selected_row), all_rows.index(row_widget)
            for r in all_rows[min(idx1, idx2):max(idx1, idx2)+1]:
                if r not in self.selected_rows:
                    self.selected_rows.append(r)
                    r.bg_color.rgba = getattr(r, 'select_color', [0.8, 0.9, 1, 1])
                    r.is_selected = True
            self.last_selected_row = row_widget
        else:
            for r in self.selected_rows:
                r.bg_color.rgba = getattr(r, 'base_color', [1, 1, 1, 1])
                r.is_selected = False
            self.selected_rows = [row_widget]
            row_widget.bg_color.rgba = getattr(row_widget, 'select_color', [0.8, 0.9, 1, 1])
            row_widget.is_selected = True
            self.last_selected_row = row_widget

        if self.screen_manager.current == 'transaction_page':
            if self.edit_mode:
                self.set_edit_mode(False)
            if self.selected_rows:
                self.fill_adjustment_panel(row_widget)
            else:
                self._clear_adjustment_panel()

    def fill_adjustment_panel(self, row_widget):
        self.is_auto_filling = True
        ids = self.screen_manager.get_screen('transaction_page').ids
        d = row_widget.row_data
        uid = row_widget.rfid_tag
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        if self.mode == 'IN':
            ids.adj_mat.text = str(d[0]); ids.adj_desc.text = str(d[1])
            ids.adj_qty.text = str(d[2]); ids.adj_sloc.text = str(d[3])
            self.on_sloc_change(None, str(d[3]))
            sbin = str(d[4])
            ids.adj_sbin_h.text = sbin[:2] if len(sbin) >= 2 else ""
            ids.adj_sbin_n.text = sbin[2:] if len(sbin) >= 2 else ""
            ids.adj_batch.text = str(d[5]) if len(d) > 5 else ""
        else:
            instr = self.outbound_instructions.get(uid, {})
            if not instr:
                self.is_auto_filling = False; return
            ids.out_orig_qty.text = str(instr.get('orig_qty', 0))
            if instr.get('type') == 'FULL':
                self.toggle_issue_type('FULL', force=True)
                ids.out_issue_qty.text = str(instr.get('orig_qty', 0))
            else:
                self.toggle_issue_type('PARTIAL', force=True)
                ids.out_issue_qty.text = str(instr.get('issue_qty', 0))
            ids.out_rem_sloc.text = instr.get('rem_sloc', '')
            sbin = instr.get('rem_sbin', '')
            ids.out_rem_sbin_h.text = sbin[:2] if len(sbin) >= 2 else ""
            ids.out_rem_sbin_n.text = sbin[2:] if len(sbin) >= 2 else ""
            self.calculate_remaining_qty()
        self.is_auto_filling = False

    def _clear_adjustment_panel(self):
        self.is_auto_filling = True
        ids = self.screen_manager.get_screen('transaction_page').ids
        ids.adj_mat.text = ""; ids.adj_desc.text = ""; ids.adj_qty.text = ""
        ids.adj_sloc.text = ""; ids.adj_sbin_h.text = ""; ids.adj_sbin_n.text = ""
        ids.adj_batch.text = ""
        self.is_auto_filling = False
        self.set_edit_mode(False)

    def clear_adjustment_inputs(self):
        if not getattr(self, 'edit_mode', False): return

        self.is_auto_filling = True
        ids = self.screen_manager.get_screen('transaction_page').ids
        ids.adj_mat.text = ""; ids.adj_desc.text = ""
        ids.adj_qty.text = ""; ids.adj_sloc.text = ""
        ids.adj_sbin_h.text = ""; ids.adj_sbin_n.text = ""
        ids.adj_batch.text = ""; ids.sap_input.text = ""
        self.is_auto_filling = False

        Clock.schedule_once(lambda dt: setattr(ids.adj_mat, 'focus', True), 0.1)

    def toggle_issue_type(self, type_str, force=False):
        if not force and not getattr(self, 'edit_mode', False):
            self.write_to_log("Press CTRL+ENTER first to Edit!", "WARNING")
            return

        ids = self.screen_manager.get_screen('transaction_page').ids
        self.outbound_issue_mode = type_str

        if not hasattr(self, 'saved_ecer_panel'):
            self.saved_ecer_panel = ids.panel_ecer_location.__self__ if
hasattr(ids.panel_ecer_location, '__self__') else ids.panel_ecer_location

        if type_str == 'FULL':
            ids.btn_type_full.background_color = [0.1, 0.6, 0.1, 1]; ids.btn_type_full.color =
[1, 1, 1, 1]
            ids.btn_type_partial.background_color = [0.8, 0.8, 0.8, 1];
ids.btn_type_partial.color = [0, 0, 0, 1]

            ids.out_issue_qty.readonly = True
            ids.out_issue_qty.background_color = [0.9, 0.9, 0.9, 0.5]
            ids.out_issue_qty.cursor_color = [0, 0, 0, 0]
            ids.out_issue_qty.text = ids.out_orig_qty.text
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            ids.ecer_wrapper.clear_widgets()
        else:
            ids.btn_type_full.background_color = [0.8, 0.8, 0.8, 1]; ids.btn_type_full.color =
[0, 0, 0, 1]
            ids.btn_type_partial.background_color = [0.1, 0.6, 0.1, 1];
ids.btn_type_partial.color = [1, 1, 1, 1]

            ids.out_issue_qty.readonly = False
            ids.out_issue_qty.background_color = [1, 1, 1, 1]
            ids.out_issue_qty.foreground_color = [0, 0, 0, 1]
            ids.out_issue_qty.cursor_color = [1, 0, 0, 1]
            ids.out_issue_qty.text = ""
            ids.out_rem_qty.text = ""

            ids.ecer_wrapper.clear_widgets()
            ids.ecer_wrapper.add_widget(self.saved_ecer_panel)

            if getattr(self, 'edit_mode', False):
                Clock.schedule_once(lambda dt: setattr(ids.out_issue_qty, 'focus', True), 0.1)

    def calculate_remaining_qty(self):
        ids = self.screen_manager.get_screen('transaction_page').ids
        try:
            orig = int(ids.out_orig_qty.text) if ids.out_orig_qty.text else 0
            iss = int(ids.out_issue_qty.text) if ids.out_issue_qty.text else 0
            rem = orig - iss
            ids.out_rem_qty.text = str(rem)

            ids.out_rem_qty.foreground_color = [1, 0, 0, 1] if rem < 0 else [0, 0, 0, 1]
        except ValueError:
            ids.out_rem_qty.text = "ERR"

    def save_outbound_settings(self):
        if not self.selected_rows: return
        ids = self.screen_manager.get_screen('transaction_page').ids
        uid = self.selected_rows[0].rfid_tag
        is_full = (getattr(self, 'outbound_issue_mode', 'FULL') == 'FULL')

        try:
            orig = int(ids.out_orig_qty.text)
            iss = int(ids.out_issue_qty.text)
            rem = orig - iss
        except ValueError:
            self.write_to_log("Invalid QTY format!", "ERROR"); return

        if not is_full:
            if iss <= 0:
                self.write_to_log("Issue QTY must be > 0 for Partial Issue!", "ERROR"); return
            if rem <= 0:
                self.write_to_log("Auto-converted to FULL ISSUE (Input QTY >= Stock QTY)",
"INFO")
                is_full = True
                self.toggle_issue_type('FULL', force=True)
            else:
                sbin = ids.out_rem_sbin_h.text.upper() + ids.out_rem_sbin_n.text.upper()
                if not self.is_valid_sbin(sbin):
                    self.write_to_log("Invalid Remaining SBin format!", "ERROR"); return

        self.outbound_instructions[uid] = {
            'type': 'FULL' if is_full else 'PARTIAL',
            'orig_qty': orig, 'issue_qty': orig if is_full else iss, 'rem_qty': 0 if is_full
else rem,
            'rem_sloc': ids.out_rem_sloc.text.upper() if not is_full else '',
            'rem_sbin': (ids.out_rem_sbin_h.text.upper() + ids.out_rem_sbin_n.text.upper()) if
not is_full else ''
        }

        final_issue_qty = orig if is_full else iss
        self.selected_rows[0].lbl_qty.text = str(final_issue_qty)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        temp_data[2] = str(final_issue_qty)
        self.selected_rows[0].row_data = tuple(temp_data)

        for i, cache_item in enumerate(self.all_items_cache):
            if str(cache_item[1]) == uid:
                temp = list(cache_item)
                temp[4] = str(final_issue_qty)
                if not is_full:
                    while len(temp) < 9: temp.append("")
                    temp[8] = "PARTIAL"
                self.all_items_cache[i] = tuple(temp)
                break

        def _bg_update_scan_out():
            try:
                payload = {"QTY": str(final_issue_qty)}
                if not is_full: payload["STATUS"] = "PARTIAL"
                self.supabase.table("SCAN_OUT").update(payload).eq("PC_TOKEN",
self.pc_token).eq("RFID_UID", uid).execute()
            except: pass
        threading.Thread(target=_bg_update_scan_out, daemon=True).start()

        self.write_to_log(f"Outbound settings saved for {uid}", "SUCCESS")

        self.selected_rows[0].current_status = "READY"
        self.selected_rows[0].select_color = [0.6, 0.9, 0.6, 1]
        self.selected_rows[0].bg_color.rgba = self.selected_rows[0].select_color

        if getattr(self, 'edit_mode', False):
            self.set_edit_mode(False)

    def select_all_rows(self):
        screen_name = self.screen_manager.current
        self.selected_rows.clear()

        if screen_name == 'transaction_page':
            table = self.screen_manager.get_screen('transaction_page').ids.table_content
            row_class = TableRow
        elif screen_name == 'so_detail_page':
            table = self.screen_manager.get_screen('so_detail_page').ids.so_detail_content
            row_class = SORowDetail
        else:
            return

        for child in table.children:
            if isinstance(child, row_class) and child.height > 0:
                self.selected_rows.append(child)
                child.is_selected = True

                target_color = getattr(child, 'select_color', [0.8, 0.9, 1, 1])
                child.bg_color.rgba = target_color

        if self.selected_rows:
            self.last_selected_row = self.selected_rows[0]
            if screen_name == 'transaction_page':
                self.fill_adjustment_panel(self.selected_rows[0])

    def delete_selected_rows(self):
        if not self.selected_rows: return

        if self.screen_manager.current != 'transaction_page':
            self.unselect_all()
            return

        table = self.screen_manager.get_screen('transaction_page').ids.table_content
        rfids_to_delete = []

        for row in self.selected_rows:
            rfids_to_delete.append(row.rfid_tag)
            if row in table.children: table.remove_widget(row)
            self.all_items_cache = [i for i in self.all_items_cache if i[1] != row.rfid_tag]
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        count = len(self.selected_rows)
        self.selected_rows.clear()
        self.last_selected_row = None
        self._clear_adjustment_panel()
        self._recalculate_row_numbers()
        self.write_to_log(f"Deleted {count} items.", "INFO")

        if rfids_to_delete:
            threading.Thread(target=self._thread_delete_from_db, args=(rfids_to_delete,),
daemon=True).start()

    def _thread_delete_from_db(self, rfids):
        if not rfids: return

        try:
            table_name = "SCAN_IN" if self.mode == "IN" else "SCAN_OUT"

            chunk_size = 40
            for i in range(0, len(rfids), chunk_size):
                chunk = rfids[i:i+chunk_size]
                self.supabase.table(table_name).delete().eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", chunk).execute()

            for uid in rfids:
                self.ignored_uids.discard(uid)

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"DB Delete Error: {err_msg}",
"ERROR"))

    def handle_spinner_filter(self, text):
        mapping = {
            'Show All': 'ALL',
            'Incomplete': 'INCOMPLETE',
            'Material Ready': 'READY',
            'Reported Material': 'REPORTED'
        }
        target_status = mapping.get(text, 'ALL')
        self.filter_by_status(target_status)

    def filter_by_status(self, target_status):
        table = self.screen_manager.get_screen('transaction_page').ids.table_content
        table.clear_widgets()

        def _check_complete(d):
            for i in range(6):
                val = str(d[i]).strip() if i < len(d) else ""
                if not val or val == "UNKNOWN" or val == "0": return False
            return True

        no_urut = 1
        for item in self.all_items_cache:
            rfid = item[1]
            row_data = item[2:]

            db_status = row_data[-1] if len(row_data) > 6 else ""
            is_complete = _check_complete(row_data)

            if db_status == "REPORTED": current_status = "REPORTED"
            elif is_complete: current_status = "READY"
            else: current_status = "INCOMPLETE"

            if target_status == "ALL" or current_status == target_status:
                row = TableRow(no=no_urut, rfid_tag=rfid, row_data=row_data, app_instance=self)
                table.add_widget(row)
                no_urut += 1
        self.unselect_all()

    def get_sloc_from_sbin(self, sbin_prefix):
        val = sbin_prefix.upper()
        for sloc, (start, end) in self.SLOC_RULES.items():
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            if start <= val <= end: return sloc
        return None

    def is_valid_sbin(self, sbin):
        sbin = sbin.upper()
        if len(sbin) != 6: return False

        prefix = sbin[:2]
        num_part = sbin[2:]

        if not prefix.isalpha() or not num_part.isdigit(): return False
        if not self.get_sloc_from_sbin(prefix): return False
        if prefix == "TR":
            return True

        try:
            r = int(num_part[:2])
            c = int(num_part[2:])
            if not (1 <= r <= 81 and 1 <= c <= 8): return False
        except:
            return False

        return True

    def auto_fill_sloc(self, text, popup_type):
        val = text.strip().upper()
        if len(val) >= 2:
            prefix = val[:2]
            sloc = None

            if prefix == "TR":
                sloc = "TR"
            else:
                sloc = self.get_sloc_from_sbin(prefix)

            if sloc:
                if popup_type == 'stock' and hasattr(self, 'stock_adj_popup') and
self.stock_adj_popup:
                    self.stock_adj_popup.ids.pop_sloc.text = sloc

    def filter_sloc_dropdown(self, text):
        if not self.edit_mode: return
        ids = self.screen_manager.get_screen('transaction_page').ids
        if ids.adj_sloc.focus and not self.is_auto_filling:
            self._create_filter_dropdown(ids.adj_sloc, self.LIST_SLOC)

    def filter_sbin_h_dropdown(self, text):
        if not self.edit_mode: return
        ids = self.screen_manager.get_screen('transaction_page').ids
        if ids.adj_sbin_h.focus and not self.is_auto_filling:
            sloc_text = ids.adj_sloc.text.upper()
            target = self.current_prefix_list if sloc_text in self.SLOC_RULES and getattr(self,
'current_prefix_list', []) else getattr(self, 'ALL_VALID_PREFIXES', [])
            self._create_filter_dropdown(ids.adj_sbin_h, target)

            val = text.strip().upper()
            if len(val) == 2:
                matched_sloc = self.get_sloc_from_sbin(val)
                if matched_sloc and sloc_text != matched_sloc:
                    self.is_auto_filling = True
                    ids.adj_sloc.text = matched_sloc
                    self.on_sloc_change(None, matched_sloc)
                    self.is_auto_filling = False

    def filter_sbin_n_dropdown(self, text):
        if not self.edit_mode: return
        ids = self.screen_manager.get_screen('transaction_page').ids

        if ids.adj_sbin_n.focus and not self.is_auto_filling:
            huruf_depan = ids.adj_sbin_h.text.upper()

            if huruf_depan == "TR":
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                self._create_filter_dropdown(ids.adj_sbin_n, getattr(self, 'LIST_TR_NUM', []))
            else:
                self._create_filter_dropdown(ids.adj_sbin_n, self.LIST_BIN_NUM)

    def _create_filter_dropdown(self, widget, full_list):
        if not hasattr(widget, 'dropdown') or widget.dropdown is None:
            widget.dropdown = DropDown(auto_dismiss=True, max_height=250)

        widget.dropdown.clear_widgets()
        widget.dropdown_index = -1

        f_list = [i for i in full_list if widget.text.upper() in i]
        if not f_list:
            widget.dropdown.dismiss()
            return

        for item in f_list:
            btn = Button(text=item, size_hint_y=None, height='35dp', background_normal='',
background_color=[0.9,0.9,0.9,1], color=[0,0,0,1])
            btn.bind(on_release=lambda b: self._select_dropdown_item(widget, b.text))
            widget.dropdown.add_widget(btn)

        if not widget.dropdown.parent:
            widget.dropdown.open(widget)

    def _select_dropdown_item(self, input_widget, value):
        self.is_auto_filling = True
        input_widget.text = value
        input_widget.dropdown.dismiss()
        ids = self.screen_manager.get_screen('transaction_page').ids

        if input_widget == ids.adj_sloc:
            self.on_sloc_change(None, value)
            Clock.schedule_once(lambda dt: setattr(ids.adj_sbin_h, 'focus', True), 0.1)

        elif input_widget == ids.adj_sbin_h:
            matched_sloc = self.get_sloc_from_sbin(value)
            if matched_sloc and ids.adj_sloc.text.upper() != matched_sloc:
                ids.adj_sloc.text = matched_sloc
                self.on_sloc_change(None, matched_sloc)
            Clock.schedule_once(lambda dt: setattr(ids.adj_sbin_n, 'focus', True), 0.1)

        self.is_auto_filling = False

    def on_sloc_change(self, instance, text):
        sloc_val = text.upper()
        self.current_prefix_list = []
        if sloc_val in self.SLOC_RULES:
            start_prefix, end_prefix = self.SLOC_RULES[sloc_val]
            self.current_prefix_list = [p for p in self.alphabet if start_prefix <= p <=
end_prefix]

        ids = self.screen_manager.get_screen('transaction_page').ids
        if self.current_prefix_list and ids.adj_sbin_h.text.upper() not in
self.current_prefix_list:
            ids.adj_sbin_h.text = ""

    def save_adjustment(self):
        ids = self.screen_manager.get_screen('transaction_page').ids
        m_num = ids.adj_mat.text.strip().upper()
        m_desc = ids.adj_desc.text.strip().upper()
        m_qty = ids.adj_qty.text.strip()
        if not m_qty:
            m_qty = "0"

        m_sloc = ids.adj_sloc.text.upper()
        m_sbin = ids.adj_sbin_h.text.upper() + ids.adj_sbin_n.text.upper()
        m_batch = ids.adj_batch.text.strip().upper()

        if not m_qty.replace('.', '', 1).isdigit():
            self.write_to_log("Quantity must be a number!", "ERROR")
            return
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        payload_master = []
        payload_scan = []
        timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        for row in self.selected_rows:
            target_rfid = row.rfid_tag

            row.row_data = (m_num, m_desc, m_qty, m_sloc, m_sbin, m_batch)
            row.lbl_mat.text, row.lbl_desc.text = m_num, m_desc
            row.lbl_qty.text, row.lbl_sloc.text, row.lbl_sbin.text = m_qty, m_sloc, m_sbin
            row.lbl_batch.text = m_batch

            if getattr(row, 'current_status', '') != "REPORTED":
                is_complete = all([m_num, m_desc, m_qty, m_sloc, m_sbin, m_batch]) and
str(m_qty) != "0" and m_num != "UNKNOWN"

                if m_sloc == "TR" or m_sbin.startswith("TR"):
                    row.current_status = "TRANSIT"
                    row.base_color = [1, 0.96, 0.7, 1]
                    row.select_color = [1, 0.8, 0.2, 1]
                elif is_complete:
                    row.current_status = "READY"
                    row.base_color = [0.85, 1, 0.85, 1]
                    row.select_color = [0.6, 0.9, 0.6, 1]
                else:
                    row.current_status = "INCOMPLETE"
                    row.base_color = [1, 1, 1, 1]
                    row.select_color = [0.6, 0.8, 1, 1]

                row.bg_color.rgba = row.select_color

                for i, item in enumerate(self.all_items_cache):
                    if str(item[1]) == str(target_rfid):
                        self.all_items_cache[i] = (item[0], target_rfid, m_num, m_desc, m_qty,
m_sloc, m_sbin, m_batch, row.current_status)
                        break

                payload_master.append({
                    "RFID_UID": target_rfid,
                    "MATERIAL": m_num,
                    "DESCRIPTION": m_desc,
                    "QTY": m_qty,
                    "TIMESTAMP": timestamp_now
                })

                payload_scan.append({
                    "RFID_UID": target_rfid,
                    "PC_TOKEN": self.pc_token,
                    "MATERIAL": m_num,
                    "DESCRIPTION": m_desc,
                    "QTY": m_qty,
                    "SLOC": m_sloc,
                    "SBIN": m_sbin,
                    "BATCH": m_batch,
                    "STATUS": row.current_status,
                    "TIMESTAMP": timestamp_now
                })

        self.write_to_log(f"Adjusted {len(self.selected_rows)} items in Local Cache!",
"SUCCESS")
        if payload_master:
            threading.Thread(target=self._thread_save_master_cloud, args=(payload_master,
payload_scan), daemon=True).start()
            threading.Thread(target=self._thread_update_catalog_learning,
args=(payload_master,), daemon=True).start()
            update_data = [{"rfid": p["RFID_UID"], "mat": p["MATERIAL"], "desc":
p["DESCRIPTION"]} for p in payload_master]
            threading.Thread(target=self._thread_update_report_names, args=(update_data,),
daemon=True).start()

    def _thread_save_master_cloud(self, payload_master, payload_scan=None):
        import time
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        max_retries = 3

        for attempt in range(max_retries):
            try:
                self.supabase.table("MASTER_DATA").upsert(payload_master).execute()

                if payload_scan:
                    table_name = "SCAN_IN" if getattr(self, 'mode', 'IN') == "IN" else
"SCAN_OUT"
                    for item in payload_scan:
                        self.supabase.table(table_name).update(item).eq("RFID_UID",
item["RFID_UID"]).eq("PC_TOKEN", self.pc_token).execute()

                Clock.schedule_once(lambda dt: self.write_to_log(f"Cloud Master Updated:
{len(payload_master)} item(s)", "SUCCESS"))
                return
            except Exception as e:
                if attempt < max_retries - 1:
                    time.sleep(1)
                else:
                    err_msg = str(e)
                    Clock.schedule_once(lambda dt: self.write_to_log(f"Cloud Update FAILED:
{err_msg[:50]}...", "ERROR"))

    def _thread_update_report_names(self, updates):
        try:
            target_table = "REPORT_IN" if self.mode == "IN" else "REPORT_OUT"
            for u in updates:

                self.supabase.table(target_table).update({
                    "MATERIAL": u["mat"],
                    "DESCRIPTION": u["desc"]
                }).eq("RFID_UID", u["rfid"]).execute()

            Clock.schedule_once(lambda dt: self._sync_report_names_ui(updates))
        except:
            pass

    def _sync_report_names_ui(self, updates):
        if not hasattr(self, 'cached_review_reports') or not self.cached_review_reports:
            return

        is_changed = False
        for r in self.cached_review_reports:
            for u in updates:
                if r.get("RFID_UID") == u["rfid"]:
                    r["MATERIAL"] = u["mat"]
                    r["DESCRIPTION"] = u["desc"]
                    is_changed = True

        if is_changed and hasattr(self, 'trans_report_popup') and
self.trans_report_popup.parent:
            self._populate_review_popup()

    def post_in_selected(self):
        if not self.selected_rows:
            self.write_to_log("No rows selected to post!", "WARNING")
            return

        global_doc_id =
self.screen_manager.get_screen('transaction_page').ids.global_doc_id.text.strip().upper()
        if not global_doc_id:
            self.write_to_log("Failed! Please enter Material Document.", "ERROR")
            return

        self.add_to_doc_history(global_doc_id)
        target_rfids = [r.rfid_tag for r in self.selected_rows]
        items_to_post = [item for item in self.all_items_cache if item[1] in target_rfids]
        rows_to_remove = list(self.selected_rows)

        self.write_to_log(f"Validating and Posting {len(items_to_post)} items... (DO:
{global_doc_id})", "INFO")
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        threading.Thread(target=self._thread_post_data, args=(items_to_post, global_doc_id,
rows_to_remove), daemon=True).start()

    def post_in_all(self):
        if not self.all_items_cache:
            self.write_to_log("No data to post!", "WARNING")
            return

        global_doc_id =
self.screen_manager.get_screen('transaction_page').ids.global_doc_id.text.strip().upper()
        if not global_doc_id:
            self.write_to_log("Failed! Please enter Material Document.", "ERROR")
            return

        self.add_to_doc_history(global_doc_id)
        items_to_post = list(self.all_items_cache)
        all_rows = [c for c in
self.screen_manager.get_screen('transaction_page').ids.table_content.children if isinstance(c,
TableRow)]

        self.write_to_log(f"Validating and Posting ALL items... (DO: {global_doc_id})", "INFO")
        threading.Thread(target=self._thread_post_data, args=(items_to_post, global_doc_id,
all_rows), daemon=True).start()

    def _thread_post_data(self, items, global_doc_id, rows_to_remove):
        import time
        try:
            payload_insert = []
            valid_items_to_delete = []
            outbound_full_uids = []
            outbound_partial_updates = []
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            for item in items:
                uid = str(item[1])
                m_num = str(item[2]).strip()
                m_desc = str(item[3]).strip()

                if self.mode == "IN":
                    issue_qty = str(item[4]).strip()
                else:
                    instr = self.outbound_instructions.get(uid, {})
                    issue_qty = str(instr.get('issue_qty', str(item[4]).strip()))

                if not m_num or m_num == "UNKNOWN" or not issue_qty or issue_qty == "0":
                    msg = f"Item {uid} skipped: Incomplete data (Check Mat/Qty)!"
                    Clock.schedule_once(lambda dt, m=msg: self.write_to_log(m, "WARNING"))
                    continue

                base_data = {
                    "RFID_UID": uid, "MATERIAL": m_num, "DESCRIPTION": m_desc,
                    "QTY": issue_qty, "SLOC": str(item[5]).strip(),
                    "SBIN": str(item[6]).strip(), "BATCH": str(item[7]).strip(),
                    "TIMESTAMP": timestamp_now
                }
                if self.mode == "IN":
                    base_data["INBOUND"] = global_doc_id
                else:
                    base_data["OUTBOUND"] = global_doc_id
                    if instr.get('type') == 'FULL':
                        outbound_full_uids.append(uid)
                    else:
                        outbound_partial_updates.append({
                            "RFID_UID": uid,
                            "MATERIAL": m_num,
                            "DESCRIPTION": m_desc,
                            "BATCH": str(item[7]).strip(),
                            "QTY": instr['rem_qty'],
                            "SLOC": instr['rem_sloc'],
                            "SBIN": instr['rem_sbin'],
                            "TIMESTAMP": timestamp_now
                        })
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                payload_insert.append(base_data)
                valid_items_to_delete.append(uid)

            if not payload_insert:
                Clock.schedule_once(lambda dt: self.write_to_log("No valid items to post! Please
adjust data first.", "ERROR"))
                return

            valid_items = [it for it in items if str(it[1]) in valid_items_to_delete]

            max_retries = 3
            success = False
            last_err = ""

            insert_chunk = 500
            delete_chunk = 50

            for attempt in range(max_retries):
                try:
                    if self.mode == "IN":
                        payload_master = [{"RFID_UID": p["RFID_UID"], "MATERIAL": p["MATERIAL"],
"DESCRIPTION": p["DESCRIPTION"], "QTY": p["QTY"], "TIMESTAMP": p["TIMESTAMP"]} for p in
payload_insert]

                        for i in range(0, len(payload_master), insert_chunk):
                            self.supabase.table("MASTER_DATA").upsert(payload_master[i:i+insert_
chunk]).execute()

                        for i in range(0, len(payload_insert), insert_chunk):
                            self.supabase.table("MATERIAL_IN").upsert(payload_insert[i:i+insert_
chunk]).execute()

                        payload_stock = [{k: v for k, v in p.items() if k not in ["INBOUND",
"OUTBOUND"]} for p in payload_insert]
                        for i in range(0, len(payload_stock), insert_chunk):
                            self.supabase.table("STOCK").upsert(payload_stock[i:i+insert_chunk])
.execute()

                        for i in range(0, len(valid_items_to_delete), delete_chunk):
                            chunk = valid_items_to_delete[i:i+delete_chunk]
                            self.supabase.table("SCAN_IN").delete().eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", chunk).execute()
                    else:
                        for i in range(0, len(payload_insert), insert_chunk):
                            self.supabase.table("MATERIAL_OUT").upsert(payload_insert[i:i+insert
_chunk]).execute()

                        if outbound_full_uids:
                            for i in range(0, len(outbound_full_uids), delete_chunk):
                                chunk = outbound_full_uids[i:i+delete_chunk]
                                self.supabase.table("STOCK").delete().in_("RFID_UID",
chunk).execute()

                                try:
                                    self.supabase.table("MASTER_DATA").delete().in_("RFID_UID",
chunk).execute()

                                except:
                                    amnesia_payload = [{"RFID_UID": u, "MATERIAL": "",
"DESCRIPTION": "", "QTY": ""} for u in chunk]
                                    self.supabase.table("MASTER_DATA").upsert(amnesia_payload).e
xecute()

                        if outbound_partial_updates:
                            for i in range(0, len(outbound_partial_updates), insert_chunk):
                                self.supabase.table("STOCK").upsert(outbound_partial_updates[i:i
+insert_chunk]).execute()

                        for i in range(0, len(valid_items_to_delete), delete_chunk):
                            chunk = valid_items_to_delete[i:i+delete_chunk]
                            self.supabase.table("SCAN_OUT").delete().eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", chunk).execute()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    success = True
                    break
                except Exception as e:
                    last_err = str(e)
                    time.sleep(1.5)

            if success:
                Clock.schedule_once(lambda dt: self._post_success_cleanup(len(valid_items),
valid_items_to_delete))
            else:
                Clock.schedule_once(lambda dt, err=last_err: self.write_to_log(f"POST ERROR:
{err}", "ERROR"))

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"SYSTEM ERROR: {err_msg}",
"ERROR"))

    def _post_success_cleanup(self, count, rfids_posted):
        table = self.screen_manager.get_screen('transaction_page').ids.table_content

        rows_to_delete = [r for r in table.children if getattr(r, 'rfid_tag', None) in
rfids_posted]
        for r in rows_to_delete:
            table.remove_widget(r)
            if r in self.selected_rows: self.selected_rows.remove(r)

        self.all_items_cache = [item for item in self.all_items_cache if str(item[1]) not in
rfids_posted]

        self._recalculate_row_numbers()

        if not self.selected_rows: self._clear_adjustment_panel()
        else: self.fill_adjustment_panel(self.selected_rows[0])

        self.write_to_log(f"Success! {count} item(s) posted.", "SUCCESS")

    def open_trans_report_popup(self):
        ids = self.screen_manager.get_screen('transaction_page').ids
        if hasattr(ids, 'btn_report_center'):
            ids.btn_report_center.text = 'REPORT'
            ids.btn_report_center.background_color = [204/255, 0, 0, 1]

        if not hasattr(self, 'trans_report_popup') or not self.trans_report_popup:
            self.trans_report_popup = Factory.TransReportPopup()

        num_sel = len(self.selected_rows)
        self.trans_report_popup.ids.report_lbl_items.text = f"Reporting {num_sel} selected
item(s)" if num_sel > 0 else "NO ITEMS SELECTED"
        self.trans_report_popup.ids.report_spinner.text = 'Select Issue...'
        self.trans_report_popup.ids.report_note.text = ''
        self.trans_report_popup.ids.review_list_content.clear_widgets()

        self.trans_report_popup.open()

        threading.Thread(target=self._thread_fetch_review_reports, daemon=True).start()

    def submit_trans_report(self):
        if not self.selected_rows:
            self.write_to_log("Select rows first to submit a new report!", "WARNING")
            return

        issue_type = self.trans_report_popup.ids.report_spinner.text
        note = self.trans_report_popup.ids.report_note.text

        if issue_type == 'Select Issue...':
            self.write_to_log("Please select an issue type!", "WARNING")
            return

        items_data = []
        for row in self.selected_rows:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            if hasattr(row, 'row_data'):
                mat, desc = row.row_data[0], row.row_data[1]
            else:
                mat, desc = row.lbl_mat.text, row.lbl_desc.text
            items_data.append({'rfid': row.rfid_tag, 'mat': mat, 'desc': desc})

        self.write_to_log("Submitting new report to basket...", "INFO")
        threading.Thread(target=self._thread_submit_trans_report, args=(items_data, issue_type,
note), daemon=True).start()

    def _thread_submit_trans_report(self, items_data, issue_type, note):
        try:
            target_table = "REPORT_IN" if self.mode == "IN" else "REPORT_OUT"
            prefix = "IN" if self.mode == "IN" else "OUT"
            date_str = datetime.now().strftime("%y%m%d")

            res_exist = self.supabase.table(target_table).select("REPORT_CODE,
RFID_UID").like("REPORT_CODE", f"%{self.pc_token[-3:]}%").execute()
            existing_rfids = [r['RFID_UID'] for r in res_exist.data] if res_exist.data else []
            existing_codes = set([r['REPORT_CODE'] for r in res_exist.data]) if res_exist.data
else set()

            valid_items = []
            for data in items_data:
                if data['rfid'] in existing_rfids:
                    Clock.schedule_once(lambda dt, r=data['rfid']: self.write_to_log(f"RFID {r}
already in basket! Skipped.", "WARNING"))
                else:
                    valid_items.append(data)

            if not valid_items:
                Clock.schedule_once(lambda dt: self.write_to_log("No new items to report (All
duplicates).", "WARNING"))
                return

            report_code = f"{prefix}-{date_str}{(len(existing_codes) + 1):02d}-PC"

            safe_issue = issue_type.replace('\n', ' ').replace('\r', '')
            safe_note = note.replace('\n', ' ').replace('\r', '')

            payload = []
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            for data in valid_items:
                payload.append({
                    "REPORT_CODE": report_code, "RFID_UID": data['rfid'],
                    "MATERIAL": data['mat'], "DESCRIPTION": data['desc'],
                    "REPORT": safe_issue, "NOTE": safe_note, "TIMESTAMP": timestamp_now
                })

            self.supabase.table(target_table).insert(payload).execute()

            try:
                scan_table = "SCAN_IN" if self.mode == "IN" else "SCAN_OUT"
                uids_to_update = [data['rfid'] for data in valid_items]
                for uid in uids_to_update:
                    self.supabase.table(scan_table).update({"STATUS":
"REPORTED"}).eq("PC_TOKEN", self.pc_token).eq("RFID_UID", uid).execute()
            except Exception as e:
                Clock.schedule_once(lambda dt, err=str(e): self.write_to_log(f"Failed to update
SCAN status: {err}", "WARNING"))

            Clock.schedule_once(lambda dt: self._trans_report_success_refresh(report_code,
payload))

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"REPORT ERROR: {err_msg}",
"ERROR"))

    def _trans_report_success_refresh(self, report_code, payload):
        self.trans_report_popup.ids.report_spinner.text = 'Select Issue...'
        self.trans_report_popup.ids.report_note.text = ''
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        uids_reported = [p['RFID_UID'] for p in payload]
        table = self.screen_manager.get_screen('transaction_page').ids.table_content
        for row in table.children:
            if getattr(row, 'rfid_tag', None) in uids_reported:
                row.current_status = "REPORTED"
                row.base_color = [1, 0.85, 0.85, 1]
                row.select_color = [1, 0.6, 0.6, 1]

                if not getattr(row, 'is_selected', False):
                    row.bg_color.rgba = row.base_color

                for i, cache_item in enumerate(self.all_items_cache):
                    if str(cache_item[1]) == str(row.rfid_tag):
                        new_item = list(cache_item)
                        while len(new_item) < 10: new_item.append("")
                        new_item[9] = "REPORTED"
                        self.all_items_cache[i] = tuple(new_item)

        self.unselect_all()
        if not hasattr(self, 'cached_review_reports') or self.cached_review_reports is None:
            self.cached_review_reports = []
        self.cached_review_reports.extend(payload)
        self._populate_review_popup()

    def _thread_fetch_review_reports(self):
        try:
            target_table = "REPORT_IN" if self.mode == "IN" else "REPORT_OUT"
            res = self.supabase.table(target_table).select("*").execute()

            self.cached_review_reports = [r for r in res.data if self.pc_token[-3:] in
r.get("REPORT_CODE", "") or "-PC" in r.get("REPORT_CODE", "")] if res.data else []
            Clock.schedule_once(lambda dt: self._populate_review_popup())
        except Exception as e:
            pass

    def _populate_review_popup(self):
        content = self.trans_report_popup.ids.review_list_content
        content.clear_widgets()
        if not getattr(self, 'cached_review_reports', []):
            empty_lbl = Label(text="-- EMPTY --", color=[0.5,0.5,0.5,1], bold=True,
size_hint_y=None, height='50dp')
            content.add_widget(empty_lbl)
            return

        for r in reversed(self.cached_review_reports):
            source = "HANDHELD" if "PC" not in r.get("REPORT_CODE", "") else "PC ADMIN"
            row = ReviewReportRow(
                mat=r.get("MATERIAL", "-"), rfid=r.get("RFID_UID", "-"),
                issue=r.get("REPORT", "-"), note=r.get("NOTE", "-"), source=source,
                report_code=r.get("REPORT_CODE", ""), app_instance=self
            )
            content.add_widget(row)

    def delete_selected_basket_items(self):
        content = self.trans_report_popup.ids.review_list_content
        rows_to_delete = [c for c in content.children if isinstance(c, ReviewReportRow) and
c.is_selected]

        if not rows_to_delete:
            self.write_to_log("Please select reports to clear!", "WARNING")
            return

        rfids_to_revert = [str(r.rfid_tag) for r in rows_to_delete]
        codes_to_delete = [r.report_code for r in rows_to_delete]

        payload_scan_revert = []

        table = self.screen_manager.get_screen('transaction_page').ids.table_content
        for row in table.children:
            uid = str(getattr(row, 'rfid_tag', ""))
            if uid in rfids_to_revert:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                m_num = row.lbl_mat.text
                m_desc = row.lbl_desc.text
                m_qty = row.lbl_qty.text
                m_sloc = row.lbl_sloc.text
                m_sbin = row.lbl_sbin.text
                m_batch = getattr(row.lbl_batch, 'text', "")

                is_complete = all([m_num, m_desc, m_qty, m_sloc, m_sbin, m_batch]) and
str(m_qty) != "0" and m_num != "UNKNOWN"

                if m_sloc == "TR" or m_sbin.startswith("TR"):
                    row.current_status = "TRANSIT"
                    row.base_color = [1, 0.96, 0.7, 1]
                    row.select_color = [1, 0.8, 0.2, 1]
                elif is_complete:
                    row.current_status = "READY"
                    row.base_color = [0.85, 1, 0.85, 1]
                    row.select_color = [0.6, 0.9, 0.6, 1]
                else:
                    row.current_status = "INCOMPLETE"
                    row.base_color = [1, 1, 1, 1]
                    row.select_color = [0.6, 0.8, 1, 1]

                if not getattr(row, 'is_selected', False):
                    row.bg_color.rgba = row.base_color
                else:
                    row.bg_color.rgba = row.select_color

                for i, item in enumerate(self.all_items_cache):
                    if str(item[1]) == uid:
                        temp = list(item)
                        while len(temp) < 9: temp.append("")
                        temp[8] = row.current_status
                        self.all_items_cache[i] = tuple(temp)

                payload_scan_revert.append({"RFID_UID": uid, "STATUS": row.current_status})

        threading.Thread(target=self._thread_delete_report_only, args=(codes_to_delete,
payload_scan_revert), daemon=True).start()

        for r in rows_to_delete:
            content.remove_widget(r)
        self.cached_review_reports = [r for r in self.cached_review_reports if
r.get("REPORT_CODE") not in codes_to_delete]
        self.write_to_log(f"Report(s) cleared. Material status re-evaluated.", "SUCCESS")

    def _thread_delete_report_only(self, codes, payload_scan_revert=None):
        try:
            t_report = f"REPORT_{self.mode}"
            self.supabase.table(t_report).delete().in_("REPORT_CODE", codes).execute()

            if payload_scan_revert:
                t_scan = f"SCAN_{self.mode}"
                for item in payload_scan_revert:
                    self.supabase.table(t_scan).update({"STATUS":
item["STATUS"]}).eq("PC_TOKEN", self.pc_token).eq("RFID_UID", item["RFID_UID"]).execute()

        except Exception as e:
            print(f"Error deleting report record: {e}")

    def print_and_clear_trans_report(self):
        if not hasattr(self, 'cached_review_reports') or not self.cached_review_reports:
            self.write_to_log("List is empty, nothing to print!", "WARNING")
            return

        doc_type = "RECEIVING" if self.mode == "IN" else "OUTGOING"
        timestamp_str = datetime.now().strftime("%Y%m%d_%H%M%S")
        default_name = f"Problem_Report_{doc_type}_{timestamp_str}.pdf"

        root = tk.Tk()
        root.withdraw()
        root.attributes('-topmost', True)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        filepath = filedialog.asksaveasfilename(
            initialdir=getattr(self, 'last_save_dir', os.path.expanduser('~')),
            initialfile=default_name,
            defaultextension=".pdf",
            filetypes=[("PDF Document", "*.pdf"), ("All files", "*.*")],
            title=f"Save {doc_type} Report As..."
        )
        root.destroy()

        if not filepath:
            self.write_to_log("Report generation cancelled by user.", "WARNING")
            return

        self.last_save_dir = os.path.dirname(filepath)

        self.write_to_log("Generating PDF and clearing basket...", "INFO")
        threading.Thread(target=self._thread_generate_trans_pdf_and_clear, args=(filepath,
doc_type), daemon=True).start()


    def _thread_generate_trans_pdf_and_clear(self, filepath, doc_type):
        try:
            sorted_reports = sorted(self.cached_review_reports, key=lambda x:
str(x.get("MATERIAL", "")))
            table_data = [['NO', 'MATERIAL', 'DESCRIPTION', 'RFID TAG', 'ISSUE & NOTE',
'SOURCE']]
            styles = getSampleStyleSheet()
            style_cell = styles['Normal']; style_cell.fontSize = 8

            for i, r in enumerate(sorted_reports):
                mat, desc, uid = r.get("MATERIAL", "-"), r.get("DESCRIPTION", "-"),
r.get("RFID_UID", "-")
                issue, note = r.get("REPORT", ""), r.get("NOTE", "")
                source = "HANDHELD" if "PC" not in r.get("REPORT_CODE", "") else "PC ADMIN"

                short_desc = desc[:20] + ".." if len(desc) > 20 else desc
                combined = f"{issue}"
                if note: combined += f" | {note}"

                issue_paragraph = Paragraph(f"<font color='red'>{combined}</font>", style_cell)
                table_data.append([str(i+1), mat, short_desc, uid, issue_paragraph, source])

            doc = SimpleDocTemplate(filepath, pagesize=A4)
            elements = []

            title_text = (
                f"<b><font size=14>WAREHOUSE {doc_type} PROBLEM REPORT</font></b><br/><br/>"
                f"<b>Timestamp:</b> {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}<br/>"
                f"<b>Cleared By Device:</b> {self.pc_token}<br/>"
            )
            elements.append(Paragraph(title_text, styles['Normal']))
            elements.append(Spacer(1, 20))

            header_color = "#3b5998" if self.mode == "IN" else "#8b4513"
            t = Table(table_data)
            t.setStyle(TableStyle([
                ('BACKGROUND', (0,0), (-1,0), colors.HexColor(header_color)),
                ('TEXTCOLOR', (0,0), (-1,0), colors.whitesmoke),
                ('ALIGN', (0,0), (-1,-1), 'CENTER'),
                ('VALIGN', (0,0), (-1,-1), 'MIDDLE'),
                ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
                ('FONTSIZE', (0,0), (-1,-1), 8),
                ('BACKGROUND', (0,1), (-1,-1), colors.HexColor("#f4f4f4")),
                ('GRID', (0,0), (-1,-1), 0.5, colors.black)
            ]))
            elements.append(t)
            doc.build(elements)

            target_table = "REPORT_IN" if self.mode == "IN" else "REPORT_OUT"
            codes_to_delete = list(set([r['REPORT_CODE'] for r in self.cached_review_reports]))
            if codes_to_delete:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                self.supabase.table(target_table).delete().in_("REPORT_CODE",
codes_to_delete).execute()

            def _success(dt):
                self.trans_report_popup.dismiss()
                filename_only = os.path.basename(filepath)
                self.write_to_log(f"PDF Generated! Saved as {filename_only}", "SUCCESS")
                if not hasattr(self, 'print_success_popup') or not self.print_success_popup:
                    self.print_success_popup = Factory.PrintSuccessPopup()
                self.print_success_popup.ids.file_path_text.text = filepath
                self.print_success_popup.open()

            Clock.schedule_once(_success, 0.1)

        except Exception as e:
            Clock.schedule_once(lambda dt: self.write_to_log(f"PDF/Clear Error: {e}", "ERROR"))

    def open_stock_opname(self):
        self.screen_manager.current = 'stock_opname_page'
        self.selected_rows.clear()
        self.last_selected_row = None
        self.screen_manager.get_screen('stock_opname_page').ids.so_select_content.clear_widgets(
)
        threading.Thread(target=self._thread_load_so_selection, daemon=True).start()

    def _thread_load_so_selection(self):
        try:
            print("[DEBUG SO] Memulai tarikan data SO Selection...")
            res = self.supabase.table("STOCK").select("SLOC, SBIN").execute()
            stock_data = res.data
            print(f"[DEBUG SO] Berhasil narik {len(stock_data)} baris dari tabel STOCK.")

            raw_list = []
            for sloc in self.LIST_SLOC:
                start_p, end_p = self.SLOC_RULES[sloc]
                valid_prefixes = [p for p in self.alphabet if start_p <= p <= end_p]
                for pfx in valid_prefixes:
                    raw_list.append({"sloc": str(sloc), "sbin": pfx, "filled": 0})

            for item in stock_data:
                item_sloc = str(item.get("SLOC", "")).upper()
                item_sbin = str(item.get("SBIN", "")).upper()

                if len(item_sbin) >= 2:
                    pfx = item_sbin[:2]
                    for r in raw_list:
                        if r["sloc"] == item_sloc and r["sbin"] == pfx:
                            r["filled"] += 1
                            break

            terisi = [r for r in raw_list if r["filled"] > 0]
            print(f"[DEBUG SO] SBin yang berhasil match & terisi: {len(terisi)} SBin.")

            self.so_selection_cache = terisi
            Clock.schedule_once(lambda dt: self.populate_so_selection())

        except Exception as e:
            print(f"[ERROR FATAL SO] Gagal load SO Selection: {e}")

    def populate_so_selection(self):
        table = self.screen_manager.get_screen('stock_opname_page').ids.so_select_content
        table.clear_widgets()
        for i, item in enumerate(self.so_selection_cache):
            row = SORowSelection(no=i+1, sloc=item['sloc'], sbin=item['sbin'],
filled=item['filled'], app_instance=self)
            table.add_widget(row)

    def clear_so_selection(self):
        self.unselect_all()
        self.write_to_log("Selection cleared.", "INFO")

    def filter_so_selection(self, text):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        query = text.lower()
        table = self.screen_manager.get_screen('stock_opname_page').ids.so_select_content
        table.clear_widgets()
        for i, item in enumerate(self.so_selection_cache):
            if query in item['sloc'].lower() or query in item['sbin'].lower():
                table.add_widget(SORowSelection(no=i+1, sloc=item['sloc'], sbin=item['sbin'],
filled=item['filled'], app_instance=self))

    def _mark_so_custom_status(self, uid, new_status):
        current = self.so_custom_status.get(uid, "")

        if new_status == "REPORTED":
            self.so_custom_status[uid] = "REPORTED"
        elif new_status == "ADJUSTED" and current != "REPORTED":
            self.so_custom_status[uid] = "ADJUSTED"
        elif not current or current == "--":
            self.so_custom_status[uid] = new_status

    def enter_so_detail(self):
        if not self.selected_rows or not isinstance(self.selected_rows[0], SORowSelection):
return
        if len(self.selected_rows) > 2:
            self.write_to_log("Maksimal pilih 2 Bin untuk Stock Opname!", "WARNING"); return

        self.sudah_print_so = False

        self.active_so_selections = [{"sloc": r.sloc, "sbin": r.sbin} for r in
self.selected_rows]
        slocs = list(dict.fromkeys([r['sloc'] for r in self.active_so_selections]))
        self.active_so_sloc = " & ".join(slocs)
        sbins = [r['sbin'] for r in self.active_so_selections]
        self.active_so_sbin = " & ".join(sbins)

        detail_screen = self.screen_manager.get_screen('so_detail_page')
        detail_screen.ids.so_detail_title.text = f'SLOC {self.active_so_sloc} STOCK OPNAME
SYSTEM'
        detail_screen.ids.lbl_so_detail_search.text = f'Search RFID Tag in SBin
{self.active_so_sbin}'

        self.selected_rows.clear()
        self.last_selected_row = None
        self.screen_manager.current = 'so_detail_page'
        detail_screen.ids.so_detail_content.clear_widgets()

        threading.Thread(target=self._thread_init_so_detail, daemon=True).start()

        if self.so_refresh_event: self.so_refresh_event.cancel()
        self.so_refresh_event = Clock.schedule_interval(self._refresh_so_status, 3)

    def _thread_init_so_detail(self):
        try:
            self.supabase.table("SCAN_SO").delete().eq("PC_TOKEN", self.pc_token).execute()
        except: pass
        self._thread_load_so_detail()

    def confirm_exit_so_detail(self):
        if not hasattr(self, 'confirm_exit_so_popup') or not self.confirm_exit_so_popup:
            self.confirm_exit_so_popup = ConfirmExitSOPopup(app_instance=self)
        self.confirm_exit_so_popup.open()

    def execute_exit_so_detail(self):
        if self.so_refresh_event: self.so_refresh_event.cancel()
        self.open_stock_opname()

    def _thread_load_so_detail(self):
        try:
            self.so_detail_cache = []

            for sel in getattr(self, 'active_so_selections', []):
                res_stock = self.supabase.table("STOCK").select("*").eq("SLOC",
sel['sloc']).ilike("SBIN", f"{sel['sbin']}%").execute()
                if res_stock.data:
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    self.so_detail_cache.extend(res_stock.data)

            res_scan = self.supabase.table("SCAN_SO").select("RFID_UID").eq("PC_TOKEN",
self.pc_token).execute()
            self.so_scanned_list = [row['RFID_UID'] for row in res_scan.data] if res_scan.data
else []
            self.so_custom_status = {}

            Clock.schedule_once(lambda dt: self.populate_so_detail())
        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"Error load SO Detail: {err_msg}",
"ERROR"))

    def _refresh_so_status(self, dt):
        threading.Thread(target=self._thread_refresh_scan, daemon=True).start()

    def _thread_refresh_scan(self):
        try:
            res_scan = self.supabase.table("SCAN_SO").select("RFID_UID").eq("PC_TOKEN",
self.pc_token).execute()
            if res_scan.data:
                new_scans = [row['RFID_UID'] for row in res_scan.data]
                for uid in new_scans:
                    if uid not in self.so_scanned_list:
                        self.so_scanned_list.append(uid)

            rfids_in_page = [item.get("RFID_UID") for item in getattr(self, 'so_detail_cache',
[])]
            if rfids_in_page:
                from datetime import datetime, timedelta
                waktu_batas = (datetime.now() - timedelta(hours=24)).strftime("%Y-%m-%d
00:00:00")
                res_report = self.supabase.table("REPORT_SO").select("RFID_UID,
REPORT").in_("RFID_UID", rfids_in_page).gte("TIMESTAMP", waktu_batas).execute()

                if res_report.data:
                    for r in res_report.data:
                        uid = r['RFID_UID']
                        issue = r.get('REPORT', '')

                        current = self.so_custom_status.get(uid, "")
                        new_stat = "ADJUSTED" if issue == "Data Adjusted" else "REPORTED"

                        if new_stat == "REPORTED":
                            self.so_custom_status[uid] = "REPORTED"
                        elif new_stat == "ADJUSTED" and current != "REPORTED":
                            self.so_custom_status[uid] = "ADJUSTED"

            Clock.schedule_once(lambda dt: self._update_so_ui_status())
        except:
            pass

    def finish_so_pc(self):
        if getattr(self, 'sudah_print_so', False):
            self.execute_finish_so_clear()
        else:
            if not hasattr(self, 'confirm_finish_no_print_popup') or not
self.confirm_finish_no_print_popup:
                self.confirm_finish_no_print_popup = Factory.ConfirmFinishNoPrintPopup()
            self.confirm_finish_no_print_popup.open()

    def execute_finish_so_clear(self):
        print("Finishing SO session & Executing Database State Machine...")
        if self.so_refresh_event: self.so_refresh_event.cancel()

        threading.Thread(target=self._thread_finalize_so_database, daemon=True).start()

    def _thread_finalize_so_database(self):
        try:
            rfids_in_page = [item.get("RFID_UID") for item in self.so_detail_cache]
            today_start = datetime.now().strftime("%Y-%m-%d 00:00:00")
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            res_reports = self.supabase.table("REPORT_SO").select("REPORT_CODE, RFID_UID,
REPORT").in_("RFID_UID", rfids_in_page).gte("TIMESTAMP", today_start).execute()
            reported_uids = [r['RFID_UID'] for r in res_reports.data] if res_reports.data else
[]

            auto_missing_payload = []
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            date_str = datetime.now().strftime("%y%m%d")
            base_code = f"SO-{date_str}"

            res_code =
self.supabase.table("REPORT_SO").select("REPORT_CODE").like("REPORT_CODE",
f"{base_code}%").execute()
            existing_codes = set([r['REPORT_CODE'] for r in res_code.data]) if res_code.data
else set()
            new_report_code = f"{base_code}{(len(existing_codes) + 1):02d}"

            for item in self.so_detail_cache:
                uid = item.get("RFID_UID", "")
                if uid not in self.so_scanned_list and uid not in reported_uids:
                    auto_missing_payload.append({
                        "REPORT_CODE": new_report_code, "RFID_UID": uid,
                        "MATERIAL": item.get("MATERIAL", ""), "DESCRIPTION":
item.get("DESCRIPTION", ""),
                        "REPORT": "Missing Item", "NOTE": "Auto-generated by System",
                        "TIMESTAMP": timestamp_now
                    })

            if auto_missing_payload:
                self.supabase.table("REPORT_SO").insert(auto_missing_payload).execute()

            all_codes = list(set([r['REPORT_CODE'] for r in res_reports.data if 'REPORT_CODE' in
r])) if res_reports.data else []
            if auto_missing_payload: all_codes.append(new_report_code)

            if all_codes:
                self.supabase.table("REPORT_SO").delete().in_("REPORT_CODE",
all_codes).neq("REPORT", "Missing Item").neq("REPORT", "Data Adjusted").execute()
                self.supabase.table("REPORT_SO").update({"REPORT": "PENDING
RADAR"}).in_("REPORT_CODE", all_codes).eq("REPORT", "Missing Item").execute()

            self.supabase.table("SCAN_SO").delete().eq("PC_TOKEN", self.pc_token).execute()

            Clock.schedule_once(lambda dt: self._finish_so_ui_reset())

        except Exception as e:
            Clock.schedule_once(lambda dt, err=str(e): self.write_to_log(f"Finalize DB Error:
{err}", "ERROR"))

    def _finish_so_ui_reset(self):
        self.so_scanned_list.clear()
        self.so_detail_cache.clear()
        self.so_custom_status.clear()
        self.unselect_all()
        self.sudah_print_so = False
        self.write_to_log("SO Session Finished and Locked for Radar.", "SUCCESS")
        self.open_stock_opname()

    def populate_so_detail(self):
        table = self.screen_manager.get_screen('so_detail_page').ids.so_detail_content
        table.clear_widgets()

        scanned_count = 0
        expected_count = len(self.so_detail_cache)

        for i, item in enumerate(self.so_detail_cache):
            uid = item.get("RFID_UID", "")

            stat_text = "--"
            if uid in self.so_custom_status:
                stat_text = self.so_custom_status[uid]
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            elif uid in self.so_scanned_list:
                stat_text = "SCANNED"

            if stat_text in ["SCANNED", "REPORTED", "ADJUSTED"]:
                scanned_count += 1

            row = SORowDetail(
                no=i+1, sloc=item.get("SLOC",""), sbin=item.get("SBIN",""), rfid_tag=uid,
mat=item.get("MATERIAL",""),
                desc=item.get("DESCRIPTION",""), batch=item.get("BATCH","-"),
qty=str(item.get("QTY","0")), status=stat_text, app_instance=self
            )

            if stat_text == "SCANNED":
                row.lbl_status.color = [40/255, 167/255, 69/255, 1]
            elif stat_text == "REPORTED":
                row.lbl_status.color = [204/255, 0, 0, 1]
            elif stat_text == "ADJUSTED":
                row.lbl_status.color = [255/255, 140/255, 0, 1]
            elif stat_text == "REP & ADJ":
                row.lbl_status.color = [1, 0.75, 0, 1]
            else:
                row.lbl_status.color = [204/255, 0, 0, 1]

            table.add_widget(row)

        detail_screen = self.screen_manager.get_screen('so_detail_page')
        detail_screen.ids.so_detail_title.text = f'SLOC {self.active_so_sloc} STOCK OPNAME
SYSTEM  |  PROGRESS: {scanned_count}/{expected_count}'

    def _update_so_ui_status(self):
        table = self.screen_manager.get_screen('so_detail_page').ids.so_detail_content
        scanned_count = 0

        for row in table.children:
            if isinstance(row, SORowDetail):
                stat_text = "--"
                if row.rfid_tag in self.so_custom_status:
                    stat_text = self.so_custom_status[row.rfid_tag]
                elif row.rfid_tag in self.so_scanned_list:
                    stat_text = "SCANNED"

                row.lbl_status.text = stat_text

                if stat_text == "SCANNED":
                    row.lbl_status.color = [40/255, 167/255, 69/255, 1]
                    scanned_count += 1
                elif stat_text == "REPORTED":
                    row.lbl_status.color = [204/255, 0, 0, 1]
                    scanned_count += 1
                elif stat_text == "ADJUSTED":
                    row.lbl_status.color = [255/255, 140/255, 0, 1]
                    scanned_count += 1
                else:
                    row.lbl_status.color = [204/255, 0, 0, 1]

        expected_count = len(self.so_detail_cache)
        detail_screen = self.screen_manager.get_screen('so_detail_page')
        detail_screen.ids.so_detail_title.text = f'SLOC {self.active_so_sloc} STOCK OPNAME
SYSTEM  |  PROGRESS: {scanned_count}/{expected_count}'

    def filter_so_detail(self, text):
        query = text.lower()
        table = self.screen_manager.get_screen('so_detail_page').ids.so_detail_content
        for row in table.children:
            if isinstance(row, SORowDetail):
                string_data = f"{row.rfid_tag} {row.lbl_mat.text} {row.lbl_desc.text}
{row.lbl_sup.text}".lower()
                row.height = '35dp' if query in string_data else 0
                row.opacity = 1 if query in string_data else 0

    def on_pop_sbin_change(self, instance, value):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        val = value.strip().upper()
        if len(val) >= 2:
            prefix = val[:2]
            matched_sloc = self.get_sloc_from_sbin(prefix)
            if matched_sloc:
                self.so_popup.ids.pop_sloc.text = matched_sloc

    def _thread_generate_report(self, format_type, filepath):
        try:
            rfids_in_page = [item.get("RFID_UID") for item in self.so_detail_cache]
            today_start = datetime.now().strftime("%Y-%m-%d 00:00:00")

            res_reports = self.supabase.table("REPORT_SO").select("REPORT_CODE, RFID_UID,
REPORT, NOTE").in_("RFID_UID", rfids_in_page).gte("TIMESTAMP", today_start).execute()

            reported_dict = {}
            if res_reports.data:
                for r in res_reports.data:
                    uid = r['RFID_UID']
                    issue = r.get('REPORT', '-')
                    note = r.get('NOTE', '').strip()
                    if uid not in reported_dict: reported_dict[uid] = {"notes": set(),
"adjusts": set(), "issues": set()}
                    if issue == "Data Adjusted":
                        if note: reported_dict[uid]["adjusts"].add(note)
                    else:
                        if issue != "-": reported_dict[uid]["issues"].add(issue)
                        if note and note != "-": reported_dict[uid]["notes"].add(note)

            pdf_table_data = [['NO', 'SLOC', 'SBIN', 'RFID', 'MATERIAL', 'DESC', 'BATCH', 'QTY',
'FOUND?', 'ISSUE & NOTE']]
            excel_table_data = [['NO', 'SLOC', 'SBIN', 'RFID', 'MATERIAL', 'DESCRIPTION',
'BATCH', 'QTY', 'FOUND?', 'ISSUE & NOTE']]

            total_items = len(self.so_detail_cache)
            scanned_count = 0

            styles = getSampleStyleSheet()
            style_cell = styles['Normal']; style_cell.fontSize = 8

            auto_missing_payload = []
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            date_str = datetime.now().strftime("%y%m%d")
            base_code = f"SO-{date_str}"

            res_code =
self.supabase.table("REPORT_SO").select("REPORT_CODE").like("REPORT_CODE",
f"{base_code}%").execute()
            existing_codes = set([r['REPORT_CODE'] for r in res_code.data]) if res_code.data
else set()
            new_report_code = f"{base_code}{(len(existing_codes) + 1):02d}"

            for i, item in enumerate(self.so_detail_cache):
                uid = item.get("RFID_UID", "")
                sloc = item.get("SLOC", "")
                sbin = item.get("SBIN", "")
                mat = item.get("MATERIAL", "")
                desc = item.get("DESCRIPTION", "")
                batch = item.get("BATCH", "")
                qty = str(item.get("QTY", "0"))
                if uid in self.so_scanned_list:
                    is_scanned = "YES"
                    scanned_count += 1
                else:
                    is_scanned = "NO"
                    if uid not in reported_dict:
                        auto_missing_payload.append({
                            "REPORT_CODE": new_report_code, "RFID_UID": uid,
                            "MATERIAL": mat, "DESCRIPTION": desc,
                            "REPORT": "Missing Item", "NOTE": "Auto-generated by System",
                            "TIMESTAMP": timestamp_now
                        })
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                short_desc = desc[:12] + ".." if len(desc) > 12 else desc

                if uid in reported_dict:
                    rd = reported_dict[uid]
                    combined = []
                    if rd["notes"]: combined.append(" | ".join(list(rd["notes"])))
                    elif rd["issues"]: combined.append(" | ".join(list(rd["issues"])))
                    if rd["adjusts"]:
                        adj_joined = " ".join(list(rd["adjusts"]))
                        combined.append(f"<i>{adj_joined}</i>")

                    final_note_str = "; ".join(combined)
                    report_issue_pdf = Paragraph(f"<font color='red'>{final_note_str}</font>",
style_cell)
                    report_issue_excel = final_note_str.replace("<i>", "").replace("</i>", "")
                elif is_scanned == "NO":
                    report_issue_pdf = Paragraph(f"<font color='red'>Missing Item</font>",
style_cell)
                    report_issue_excel = "Missing Item"
                else:
                    report_issue_pdf = "-"
                    report_issue_excel = "-"

                pdf_table_data.append([str(i+1), sloc, sbin, uid, mat, short_desc, batch, qty,
is_scanned, report_issue_pdf])
                excel_table_data.append([str(i+1), sloc, sbin, uid, mat, desc, batch, qty,
is_scanned, report_issue_excel])

            accuracy = (scanned_count / total_items * 100) if total_items > 0 else 0.0

            if format_type == 'excel':
                try:
                    from openpyxl import Workbook
                    from openpyxl.styles import Font, Border, Side
                except ImportError:
                    Clock.schedule_once(lambda dt: self.write_to_log("ERROR: pip install
openpyxl dulu!", "ERROR"))
                    return

                wb = Workbook()
                ws = wb.active
                ws.title = "SO Report"

                ws.append(["WAREHOUSE STOCK OPNAME REPORT"])
                ws.append([])
                ws.append(["Location:", f"SLOC {self.active_so_sloc} | SBin:
{self.active_so_sbin}"])
                ws.append(["PIC / Operator:", getattr(self, 'current_so_pic', 'ADMIN')])
                ws.append(["Timestamp:", datetime.now().strftime('%Y-%m-%d %H:%M:%S')])
                ws.append([])
                ws.append(["SUMMARY"])
                ws.append(["Total Expected Items:", total_items])
                ws.append(["Total Scanned Items:", scanned_count])
                ws.append(["Stock Accuracy:", f"{accuracy:.2f}%"])
                ws.append([])

                for row_data in excel_table_data:
                    ws.append(row_data)

                ws['A1'].font = Font(bold=True, size=14)


                thin_border = Border(
                    left=Side(style='thin'), right=Side(style='thin'),
                    top=Side(style='thin'), bottom=Side(style='thin')
                )

                for row in ws.iter_rows(min_row=12, max_row=ws.max_row, min_col=1,
max_col=ws.max_column):
                    for cell in row:
                        cell.border = thin_border
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                    if cell.row == 12:
                        cell.font = Font(bold=True)

                for col in ws.columns:
                    max_length = 0
                    column_letter = col[0].column_letter
                    for cell in col:
                        try:
                            if len(str(cell.value)) > max_length:
                                max_length = len(str(cell.value))
                        except:
                            pass
                    ws.column_dimensions[column_letter].width = (max_length + 2.5)

                wb.save(filepath)

            else:
                doc = SimpleDocTemplate(filepath, pagesize=A4)
                elements = []
                color_acc = "green" if accuracy == 100 else "red"
                title_text = (
                    f"<b><font size=14>WAREHOUSE STOCK OPNAME REPORT</font></b><br/><br/>"
                    f"<b>Location:</b> SLOC {self.active_so_sloc} | SBin:
{self.active_so_sbin}<br/>"
                    f"<b>PIC / Operator:</b> {getattr(self, 'current_so_pic', 'ADMIN')}<br/>"
                    f"<b>Timestamp:</b> {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}<br/><br/>"
                    f"<b>    SUMMARY    </b><br/>"
                    f"Total Expected Items : {total_items}<br/>"
                    f"Total Scanned Items  : {scanned_count}<br/>"
                    f"<b>Stock Accuracy       : <font
color='{color_acc}'>{accuracy:.2f}%</font></b>"
                )
                elements.append(Paragraph(title_text, styles['Normal']))
                elements.append(Spacer(1, 20))

                t = Table(pdf_table_data)
                t.setStyle(TableStyle([
                    ('BACKGROUND', (0,0), (-1,0), colors.HexColor("#3b5998")),
                    ('TEXTCOLOR', (0,0), (-1,0), colors.whitesmoke),
                    ('ALIGN', (0,0), (-1,-1), 'CENTER'),
                    ('VALIGN', (0,0), (-1,-1), 'MIDDLE'),
                    ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
                    ('FONTSIZE', (0,0), (-1,-1), 8),
                    ('BOTTOMPADDING', (0,0), (-1,0), 8),
                    ('BACKGROUND', (0,1), (-1,-1), colors.HexColor("#f4f4f4")),
                    ('GRID', (0,0), (-1,-1), 0.5, colors.black)
                ]))
                elements.append(t)
                doc.build(elements)

            if auto_missing_payload:
                self.supabase.table("REPORT_SO").insert(auto_missing_payload).execute()

            if res_reports.data or auto_missing_payload:
                all_codes = list(set([r['REPORT_CODE'] for r in res_reports.data if
'REPORT_CODE' in r]))
                if auto_missing_payload: all_codes.append(new_report_code)

                if all_codes:
                    self.supabase.table("REPORT_SO").delete().in_("REPORT_CODE",
all_codes).neq("REPORT", "Missing Item").neq("REPORT", "Data Adjusted").execute()

                    self.supabase.table("REPORT_SO").update({"REPORT": "PENDING
RADAR"}).in_("REPORT_CODE", all_codes).eq("REPORT", "Missing Item").execute()

            def show_popup(dt):
                filename_only = os.path.basename(filepath)
                self.write_to_log(f"Report Generated! Saved as {filename_only}", "SUCCESS")
                if not hasattr(self, 'print_success_popup') or not self.print_success_popup:
                    self.print_success_popup = Factory.PrintSuccessPopup()
                self.print_success_popup.ids.file_path_text.text = filepath
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                self.print_success_popup.open()
            Clock.schedule_once(show_popup, 0.1)

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"REPORT GEN ERROR: {err_msg}",
"ERROR"))

    def open_report_popup(self):
        if self.screen_manager.current != 'so_detail_page' and not self.selected_rows:
            self.write_to_log("Select rows first to report!", "WARNING")
            return

        if not hasattr(self, 'report_popup') or not self.report_popup:
            self.report_popup = Factory.ReportPopup()

        if self.selected_rows:
            self.report_popup.ids.report_lbl_items.text = f"Reporting {len(self.selected_rows)}
selected item(s)"
        else:
            self.report_popup.ids.report_lbl_items.text = "GENERAL PROBLEM (No items selected)"

        self.report_popup.ids.report_spinner.text = 'Select Issue...'
        self.report_popup.ids.report_note.text = ''
        self.report_popup.open()

    def confirm_report(self):
        if not self.report_popup: return
        issue_type = self.report_popup.ids.report_spinner.text
        note = self.report_popup.ids.report_note.text

        if issue_type == 'Select Issue...':
            self.write_to_log("Please select an issue type!", "WARNING"); return

        items_data = []
        if self.selected_rows:
            for row in self.selected_rows:
                if hasattr(row, 'row_data'):
                    mat, desc = row.row_data[0], row.row_data[1]
                else:
                    mat, desc = row.lbl_mat.text, row.lbl_desc.text
                items_data.append({'rfid': row.rfid_tag, 'mat': mat, 'desc': desc})
        else:
            items_data.append({'rfid': "GENERAL_PROBLEM", 'mat': "-", 'desc': "-"})

        screen_context = self.screen_manager.current
        self.report_popup.dismiss()
        self.write_to_log("Generating Report Code and Submitting...", "INFO")
        threading.Thread(target=self._thread_submit_report, args=(items_data, issue_type, note,
screen_context), daemon=True).start()

    def open_pic_popup(self):
        if not self.so_detail_cache:
            self.write_to_log("No data to print!", "WARNING"); return
        if not hasattr(self, 'pic_popup') or not self.pic_popup:
            self.pic_popup = Factory.SOPICPopup()
        self.pic_popup.ids.pic_input.text = ""
        self.pic_popup.open()

    def submit_pic_and_print(self, pic_name, format_type='pdf'):
        self.current_so_pic = pic_name.strip().upper() if pic_name.strip() else "ADMIN"

        if hasattr(self, 'pic_popup') and self.pic_popup:
            self.pic_popup.dismiss()

        Clock.schedule_once(lambda dt: self._open_windows_explorer(format_type), 0.2)

    def _open_windows_explorer(self, format_type):
        import os
        import tkinter as tk
        from tkinter import filedialog
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        timestamp_str = datetime.now().strftime("%Y%m%d_%H%M%S")
        safe_sloc = self.active_so_sloc.replace(' & ', '_')
        safe_sbin = self.active_so_sbin.replace(' & ', '_')

        root = tk.Tk()
        root.withdraw()
        root.attributes('-topmost', True)

        if format_type == 'excel':
            default_name = f"SO_Report_{safe_sloc}_{safe_sbin}_{timestamp_str}.xlsx"
            file_types = [("Excel Workbook", "*.xlsx"), ("All files", "*.*")]
            ext = ".xlsx"
        else:
            default_name = f"SO_Report_{safe_sloc}_{safe_sbin}_{timestamp_str}.pdf"
            file_types = [("PDF Document", "*.pdf"), ("All files", "*.*")]
            ext = ".pdf"

        filepath = filedialog.asksaveasfilename(
            initialdir=getattr(self, 'last_save_dir', os.path.expanduser('~')),
            initialfile=default_name,
            defaultextension=ext,
            filetypes=file_types,
            title=f"Save {format_type.upper()} Report As..."
        )
        root.destroy()

        if not filepath:
            self.write_to_log("Report generation cancelled by user.", "WARNING")
            return

        self.last_save_dir = os.path.dirname(filepath)

        self.sudah_print_so = True
        self.write_to_log(f"Generating {format_type.upper()} Report for PIC:
{self.current_so_pic}...", "INFO")

        threading.Thread(target=self._thread_generate_report, args=(format_type, filepath),
daemon=True).start()
    def _thread_submit_report(self, items_data, issue_type, note, screen_context):
        try:
            if screen_context == 'so_detail_page':
                prefix = "SO"
                target_table = "REPORT_SO"
            else:
                prefix = "IN" if getattr(self, 'mode', 'IN') == "IN" else "OUT"
                target_table = f"REPORT_{prefix}"

            date_str = datetime.now().strftime("%y%m%d")
            base_code = f"{prefix}-{date_str}"

            res = self.supabase.table(target_table).select("REPORT_CODE").like("REPORT_CODE",
f"{base_code}%").execute()
            existing_codes = set([r['REPORT_CODE'] for r in res.data]) if res.data else set()
            report_code = f"{base_code}{(len(existing_codes) + 1):02d}"

            payload = []
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            for data in items_data:
                payload.append({
                    "REPORT_CODE": report_code, "RFID_UID": data['rfid'],
                    "MATERIAL": data['mat'], "DESCRIPTION": data['desc'],
                    "REPORT": issue_type, "NOTE": note, "TIMESTAMP": timestamp_now
                })

            self.supabase.table(target_table).insert(payload).execute()
            Clock.schedule_once(lambda dt: self._report_success_cleanup(report_code,
screen_context))

        except Exception as e:
            err_msg = str(e)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
            Clock.schedule_once(lambda dt: self.write_to_log(f"REPORT ERROR: {err_msg}",
"ERROR"))

    def _report_success_cleanup(self, report_code, screen_context):
        self.write_to_log(f"Problem Reported successfully! (Code: {report_code})", "SUCCESS")

        if screen_context == 'so_detail_page':
            for row in self.selected_rows:
                self._mark_so_custom_status(row.rfid_tag, "REPORTED")
            self._update_so_ui_status()
            self.unselect_all()
        else:
            self.delete_selected_rows()

    def handle_close_request(self, *args):
        if not self.exit_popup:
            self.exit_popup = ExitMessageBox()
        if not self.exit_popup.parent:
            self.exit_popup.open()
        return True

    def open_password_popup(self):
        if not hasattr(self, 'pwd_popup') or not self.pwd_popup:
            self.pwd_popup = Factory.PasswordPopup()
        self.pwd_popup.ids.pwd_input.text = ""
        self.pwd_popup.open()

    def verify_password(self, pwd):
        if pwd == "ADMIN123":
            self.pwd_popup.dismiss()
            self.screen_manager.current = 'stock_page'

            self.stock_selected_rows = []
            self.stock_last_selected_row = None

            Clock.schedule_once(lambda dt: self.fetch_stock_data(), 0.2)
        else:
            self.pwd_popup.ids.pwd_input.text = ""
            self.pwd_popup.ids.pwd_input.hint_text = "WRONG PASSWORD!"

    def fetch_stock_data(self):
        content = self.screen_manager.get_screen('stock_page').ids.stock_list_content
        content.clear_widgets()
        content.add_widget(Label(text="Loading Stock Data...", color=[0,0,0,1],
size_hint_y=None, height='40dp'))

        self.stock_selected_rows = []
        self.stock_last_selected_row = None

        threading.Thread(target=self._thread_fetch_stock, daemon=True).start()

    def _thread_fetch_stock(self):
        try:
            res = self.supabase.table("STOCK").select("*").execute()
            self.global_stock_cache = res.data if res.data else []
            Clock.schedule_once(lambda dt: self.filter_stock_data())
        except Exception as e:
            Clock.schedule_once(lambda dt: self._show_stock_error(str(e)))

    def populate_stock_table(self, data_list):
        content = self.screen_manager.get_screen('stock_page').ids.stock_list_content
        content.clear_widgets()
        self.stock_selected_rows = []

        for i, item in enumerate(data_list):
            row = StockRow(
                no=i+1,
                rfid=item.get("RFID_UID", "-"),
                mat=item.get("MATERIAL", "-"),
                desc=item.get("DESCRIPTION", "-"),
                batch=item.get("BATCH", "-"),
                qty=item.get("QTY", "0"),
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
                sloc=item.get("SLOC", "-"),
                sbin=item.get("SBIN", "-"),
                app_instance=self
            )
            content.add_widget(row)

    def _show_stock_error(self, err):
        content = self.screen_manager.get_screen('stock_page').ids.stock_list_content
        content.clear_widgets()
        content.add_widget(Label(text=f"Error Loading: {err}", color=[0.8,0,0,1],
size_hint_y=None, height='40dp'))

    def filter_stock_data(self, *args):
        if not hasattr(self, 'global_stock_cache'): return
        ids = self.screen_manager.get_screen('stock_page').ids

        f_rfid = ids.flt_rfid.text.strip().upper()
        f_mat = ids.flt_mat.text.strip().upper()
        f_desc = ids.flt_desc.text.strip().upper()
        f_batch = ids.flt_batch.text.strip().upper()
        f_sloc = ids.flt_sloc.text.strip().upper()
        f_sbin = ids.flt_sbin.text.strip().upper()

        filtered = []
        for item in self.global_stock_cache:
            if f_rfid and f_rfid not in str(item.get("RFID_UID", "")).upper(): continue
            if f_mat and f_mat not in str(item.get("MATERIAL", "")).upper(): continue
            if f_desc and f_desc not in str(item.get("DESCRIPTION", "")).upper(): continue
            if f_batch and f_batch not in str(item.get("BATCH", "")).upper(): continue
            if f_sloc and f_sloc not in str(item.get("SLOC", "")).upper(): continue
            if f_sbin and f_sbin not in str(item.get("SBIN", "")).upper(): continue

            filtered.append(item)

        self.populate_stock_table(filtered)

    def on_stock_row_select(self, row_widget):
        table = row_widget.parent
        if not table: return

        all_rows = [c for c in reversed(table.children) if hasattr(c, 'bg_color')]
        modifiers = Window.modifiers

        if 'ctrl' in modifiers:
            if row_widget in self.stock_selected_rows:
                self.stock_selected_rows.remove(row_widget)
                row_widget.bg_color.rgba = getattr(row_widget, 'base_color', [1, 1, 1, 1])
                row_widget.is_selected = False
            else:
                self.stock_selected_rows.append(row_widget)
                row_widget.bg_color.rgba = getattr(row_widget, 'select_color', [0.8, 0.9, 1, 1])
                row_widget.is_selected = True
            self.stock_last_selected_row = row_widget
        elif 'shift' in modifiers and self.stock_last_selected_row in all_rows:
            idx1, idx2 = all_rows.index(self.stock_last_selected_row),
all_rows.index(row_widget)
            for r in all_rows[min(idx1, idx2):max(idx1, idx2)+1]:
                if r not in self.stock_selected_rows:
                    self.stock_selected_rows.append(r)
                    r.bg_color.rgba = getattr(r, 'select_color', [0.8, 0.9, 1, 1])
                    r.is_selected = True
            self.stock_last_selected_row = row_widget
        else:
            for r in self.stock_selected_rows:
                r.bg_color.rgba = getattr(r, 'base_color', [1, 1, 1, 1])
                r.is_selected = False
            self.stock_selected_rows = [row_widget]
            row_widget.bg_color.rgba = getattr(row_widget, 'select_color', [0.8, 0.9, 1, 1])
            row_widget.is_selected = True
            self.stock_last_selected_row = row_widget

    def open_stock_adjust_popup(self):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        if not hasattr(self, 'stock_selected_rows') or not self.stock_selected_rows:
            return

        if not hasattr(self, 'stock_adj_popup') or not self.stock_adj_popup:
            self.stock_adj_popup = Factory.StockAdjustPopup()

        first_row = self.stock_selected_rows[0]
        data = first_row.row_data

        self.stock_adj_popup.ids.pop_stock_adj_info.text = f"Adjusting
{len(self.stock_selected_rows)} Item(s) in Stock"
        self.stock_adj_popup.ids.pop_mat.text = str(data["mat"])
        self.stock_adj_popup.ids.pop_desc.text = str(data["desc"])
        self.stock_adj_popup.ids.pop_qty.text = str(data["qty"])
        self.stock_adj_popup.ids.pop_batch.text = str(data["batch"])
        self.stock_adj_popup.ids.pop_sloc.text = str(data["sloc"])
        self.stock_adj_popup.ids.pop_sbin.text = str(data["sbin"])

        self.stock_adj_popup.open()

    def submit_stock_adjust(self):
        uids = [r.rfid_tag for r in self.stock_selected_rows]
        if not uids: return

        p_mat = self.stock_adj_popup.ids.pop_mat.text.strip().upper()
        p_desc = self.stock_adj_popup.ids.pop_desc.text.strip().upper()
        try: p_qty = int(self.stock_adj_popup.ids.pop_qty.text.strip())
        except: p_qty = 0
        p_batch = self.stock_adj_popup.ids.pop_batch.text.strip().upper()
        p_sloc = self.stock_adj_popup.ids.pop_sloc.text.strip().upper()
        p_sbin = self.stock_adj_popup.ids.pop_sbin.text.strip().upper()

        if not all([p_mat, p_desc, p_sloc, p_sbin]) or p_qty == 0 or p_sloc == "SELECT SLOC":
            return

        self.pending_adjust_data = {
            'type': 'stock_supervisory', 'uids': uids, 'mat': p_mat, 'desc': p_desc,
            'qty': p_qty, 'batch': p_batch, 'sloc': p_sloc, 'sbin': p_sbin
        }

        if not hasattr(self, 'confirm_loc_popup') or not self.confirm_loc_popup:
            self.confirm_loc_popup = Factory.ConfirmLocPopup()

        self.confirm_loc_popup.ids.lbl_conf_count.text = f"Adjusting {len(uids)} Stock Item(s)"
        self.confirm_loc_popup.ids.lbl_conf_mat.text = p_mat
        self.confirm_loc_popup.ids.lbl_conf_loc.text = f"{p_sloc} - {p_sbin}"
        self.confirm_loc_popup.open()

    def open_delete_stock_popup(self):
        if not getattr(self, 'stock_selected_rows', []):
            self.write_to_log("Select rows first to delete from Stock!", "WARNING")
            return

        if not hasattr(self, 'delete_stock_popup') or not self.delete_stock_popup:
            self.delete_stock_popup = Factory.ConfirmDeleteStockPopup()

        count = len(self.stock_selected_rows)
        self.delete_stock_popup.ids.lbl_del_msg.text = f"Are you sure you want to
PERMANENTLY\ndelete {count} item(s) from STOCK?"
        self.delete_stock_popup.open()

    def execute_delete_stock(self):
        uids_to_delete = [r.rfid_tag for r in self.stock_selected_rows]
        if not uids_to_delete: return

        self.write_to_log(f"Deleting {len(uids_to_delete)} items from Stock Database...",
"INFO")

        threading.Thread(target=self._thread_delete_stock, args=(uids_to_delete,),
daemon=True).start()

    def _thread_delete_stock(self, uids):
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        try:
            delete_chunk = 50
            for i in range(0, len(uids), delete_chunk):
                chunk = uids[i:i+delete_chunk]
                self.supabase.table("STOCK").delete().in_("RFID_UID", chunk).execute()

            if hasattr(self, 'global_stock_cache'):
                self.global_stock_cache = [item for item in self.global_stock_cache if
str(item.get("RFID_UID", "")) not in uids]

            Clock.schedule_once(lambda dt: self._post_delete_stock_ui())

        except Exception as e:
            err_msg = str(e)
            Clock.schedule_once(lambda dt: self.write_to_log(f"DELETE STOCK ERROR: {err_msg}",
"ERROR"))

    def _post_delete_stock_ui(self):
        self.filter_stock_data()
        self.unselect_all_stock()
        self.write_to_log("Stock item(s) permanently deleted successfully.", "SUCCESS")

class ReviewReportRow(ButtonBehavior, GridLayout):
    def __init__(self, mat, rfid, issue, note, source, report_code, app_instance, **kwargs):
        super().__init__(**kwargs)
        self.cols = 5
        self.size_hint_y = None
        self.height = '40dp'
        self.report_code = report_code
        self.rfid_tag = str(rfid)
        self.is_selected = False
        self.app_instance = app_instance

        with self.canvas.before:
            self.bg_color = Color(1, 1, 1, 1)
            self.rect = Rectangle(size=self.size, pos=self.pos)
        self.bind(pos=lambda i,v: setattr(self.rect, 'pos', v), size=lambda i,v:
setattr(self.rect, 'size', v))

        def mk(t, x, is_red=False):
            c = [0.8, 0, 0, 1] if is_red else [0, 0, 0, 1]
            lbl = Label(text=str(t), color=c, shorten=True, size_hint_x=x, halign='center',
valign='middle')
            lbl.bind(size=lambda l, s: setattr(l, 'text_size', (l.width, None)))
            return lbl

        self.add_widget(mk(mat, 1.2))
        self.add_widget(mk(rfid, 2.5))
        self.add_widget(mk(issue, 1.0, is_red=True))
        self.add_widget(mk(note, 2.0))
        self.add_widget(mk(source, 0.8))

    def on_release(self):
        self.is_selected = not self.is_selected
        self.bg_color.rgba = [0.8, 0.9, 1, 1] if self.is_selected else [1, 1, 1, 1]

class TableRow(TouchRippleBehavior, ButtonBehavior, GridLayout):
    def __init__(self, no, rfid_tag, row_data, app_instance, **kwargs):
        super().__init__(**kwargs)
        self.cols = 8
        self.size_hint_y = None; self.height = '35dp'
        self.app_instance = app_instance; self.rfid_tag = str(rfid_tag)
        self.row_data = row_data
        self.is_selected = False

        def _check_complete(d):
            for i in range(6):
                val = str(d[i]).strip() if i < len(d) else ""
                if not val or val == "UNKNOWN" or val == "0": return False
            return True

        self.is_complete = _check_complete(row_data)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        status_text = row_data[6] if len(row_data) > 6 else ""

        sloc_text = str(row_data[3]).strip().upper() if len(row_data) > 3 else ""
        sbin_text = str(row_data[4]).strip().upper() if len(row_data) > 4 else ""

        if status_text == "REPORTED":
            self.current_status = "REPORTED"
            self.base_color = [1, 0.85, 0.85, 1]; self.select_color = [1, 0.6, 0.6, 1]
        elif sloc_text == "TR" or sbin_text.startswith("TR"):
            self.current_status = "TRANSIT"
            self.base_color = [1, 0.96, 0.7, 1]; self.select_color = [1, 0.8, 0.2, 1]
        elif self.is_complete:
            self.current_status = "READY"
            self.base_color = [0.85, 1, 0.85, 1]; self.select_color = [0.6, 0.9, 0.6, 1]
        else:
            self.current_status = "INCOMPLETE"
            self.base_color = [1, 1, 1, 1]; self.select_color = [0.6, 0.8, 1, 1]

        with self.canvas.before:
            self.bg_color = Color(*self.base_color)
            self.rect = Rectangle(size=self.size, pos=self.pos)
        self.bind(pos=self.update_rect, size=self.update_rect)

        def make_cell(text_val, hint_x=1.0):
            return Label(text=str(text_val), color=[0,0,0,1], shorten=True, halign='center',
size_hint_x=hint_x)

        self.lbl_no = make_cell(no, 0.4); self.add_widget(self.lbl_no)
        self.add_widget(make_cell(rfid_tag, 2.2))
        self.lbl_mat = make_cell(row_data[0], 1.5); self.add_widget(self.lbl_mat)
        self.lbl_desc = make_cell(row_data[1], 4.0); self.add_widget(self.lbl_desc)
        self.lbl_batch = make_cell(row_data[5] if len(row_data) > 5 else "", 1.1);
self.add_widget(self.lbl_batch)
        self.lbl_qty = make_cell(row_data[2], 0.6); self.add_widget(self.lbl_qty)
        self.lbl_sloc = make_cell(row_data[3], 0.8); self.add_widget(self.lbl_sloc)
        self.lbl_sbin = make_cell(row_data[4], 0.8); self.add_widget(self.lbl_sbin)

    def update_rect(self, instance, value):
        self.rect.pos = instance.pos; self.rect.size = instance.size

    def on_touch_down(self, touch):
        if self.collide_point(*touch.pos): return super().on_touch_down(touch)
        return False

    def on_release(self):
        self.app_instance.on_row_select(self)

class SORowSelection(TouchRippleBehavior, ButtonBehavior, GridLayout):
    def __init__(self, no, sloc, sbin, filled, app_instance, **kwargs):
        super().__init__(**kwargs)
        self.cols = 4; self.size_hint_y = None; self.height = '35dp'
        self.app_instance = app_instance; self.sloc = sloc; self.sbin = sbin
        with self.canvas.before:
            self.bg_color = Color(1, 1, 1, 1); self.rect = Rectangle(size=self.size,
pos=self.pos)
        self.bind(pos=lambda i,v: setattr(self.rect, 'pos', v), size=lambda i,v:
setattr(self.rect, 'size', v))
        def mk(t): return Label(text=str(t), color=[0,0,0,1], shorten=True, halign='center')
        self.add_widget(mk(no)); self.add_widget(mk(sloc)); self.add_widget(mk(sbin));
self.add_widget(mk(filled))
    def on_release(self): self.app_instance.on_row_select(self)

class SORowDetail(TouchRippleBehavior, ButtonBehavior, GridLayout):
    def __init__(self, no, sloc, sbin, rfid_tag, mat, desc, batch, qty, status, app_instance,
**kwargs):
        super().__init__(**kwargs)
        self.cols = 9; self.size_hint_y = None; self.height = '35dp'
        self.app_instance = app_instance; self.rfid_tag = rfid_tag; self.sbin = sbin
        with self.canvas.before:
            self.bg_color = Color(1, 1, 1, 1); self.rect = Rectangle(size=self.size,
pos=self.pos)
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
        self.bind(pos=lambda i,v: setattr(self.rect, 'pos', v), size=lambda i,v:
setattr(self.rect, 'size', v))
        def mk(t, x): return Label(text=str(t), color=[0,0,0,1], shorten=True, size_hint_x=x,
halign='center')

        self.add_widget(mk(no, 0.4))
        self.lbl_sloc = mk(sloc, 0.6); self.add_widget(self.lbl_sloc)
        self.lbl_sbin = mk(sbin, 0.8); self.add_widget(self.lbl_sbin)
        self.add_widget(mk(rfid_tag, 2.5))
        self.lbl_mat = mk(mat, 1.2); self.add_widget(self.lbl_mat)
        self.lbl_desc = mk(desc, 3.0); self.add_widget(self.lbl_desc)
        self.lbl_batch = mk(batch, 1.2); self.add_widget(self.lbl_batch)
        self.lbl_qty = mk(qty, 0.6); self.add_widget(self.lbl_qty)

        self.lbl_status = mk(status, 1.0)
        self.lbl_status.color = [40/255, 167/255, 69/255, 1] if status == "SCANNED" else
[204/255, 0, 0, 1]
        self.lbl_status.bold = True
        self.add_widget(self.lbl_status)

    def on_release(self): self.app_instance.on_row_select(self)

class ConfirmExitSOPopup(ModalView):
    def __init__(self, app_instance, **kwargs):
        super().__init__(**kwargs)
        self.app_instance = app_instance
        self.selected = 'NO'

    def on_open(self):
        self.selected = 'NO'
        self.update_colors()

    def toggle_selection(self):
        self.selected = 'YES' if self.selected == 'NO' else 'NO'
        self.update_colors()

    def update_colors(self):
        crimson_focus = [220/255, 20/255, 60/255, 1]
        grey_idle = [0.8, 0.8, 0.8, 1]

        if self.selected == 'YES':
            self.ids.btn_yes.background_color = crimson_focus
            self.ids.btn_yes.color = [1, 1, 1, 1]

            self.ids.btn_no.background_color = grey_idle
            self.ids.btn_no.color = [0, 0, 0, 1]
        else:
            self.ids.btn_yes.background_color = grey_idle
            self.ids.btn_yes.color = [0, 0, 0, 1]

            self.ids.btn_no.background_color = crimson_focus
            self.ids.btn_no.color = [1, 1, 1, 1]

    def confirm(self):
        if self.selected == 'YES':
            self.app_instance.execute_exit_so_detail()
            self.dismiss()
        else:
            self.dismiss()

class ExitMessageBox(ModalView):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.selected = 'NO'

    def on_open(self):
        self.selected = 'NO'
        self.update_colors()

    def toggle_selection(self):
        self.selected = 'YES' if self.selected == 'NO' else 'NO'
        self.update_colors()
```

Lampiran 11 Program Koding Sistem Informasi RFID Komputer

```
    def update_colors(self):
        crimson_focus = [220/255, 20/255, 60/255, 1]
        grey_idle = [0.8, 0.8, 0.8, 1]

        if self.selected == 'YES':
            self.ids.btn_yes.background_color = crimson_focus
            self.ids.btn_yes.color = [1, 1, 1, 1]
            self.ids.btn_no.background_color = grey_idle
            self.ids.btn_no.color = [0, 0, 0, 1]
        else:
            self.ids.btn_yes.background_color = grey_idle
            self.ids.btn_yes.color = [0, 0, 0, 1]
            self.ids.btn_no.background_color = crimson_focus
            self.ids.btn_no.color = [1, 1, 1, 1]

    def confirm(self):
        if self.selected == 'YES':
            App.get_running_app().stop()
        else:
            self.dismiss()

class StockRow(TouchRippleBehavior, ButtonBehavior, GridLayout):
    def __init__(self, no, rfid, mat, desc, batch, qty, sloc, sbin, app_instance, **kwargs):
        super().__init__(**kwargs)
        self.cols = 8
        self.size_hint_y = None
        self.height = '35dp'
        self.app_instance = app_instance
        self.rfid_tag = str(rfid)
        self.is_selected = False
        self.base_color = [1, 1, 1, 1]
        self.select_color = [0.8, 0.9, 1, 1]

        self.row_data = {
            "mat": mat, "desc": desc, "batch": batch, "qty": qty, "sloc": sloc, "sbin": sbin
        }

        with self.canvas.before:
            self.bg_color = Color(*self.base_color)
            self.rect = Rectangle(size=self.size, pos=self.pos)
        self.bind(pos=self.update_rect, size=self.update_rect)

        def mk(t, x):
            return Label(text=str(t), color=[0,0,0,1], shorten=True, size_hint_x=x,
halign='center')

        self.add_widget(mk(no, 0.5))
        self.add_widget(mk(rfid, 2.0))
        self.add_widget(mk(mat, 1.5))
        self.add_widget(mk(desc, 3.0))
        self.add_widget(mk(batch, 1.0))
        self.add_widget(mk(qty, 0.8))
        self.add_widget(mk(sloc, 0.8))
        self.add_widget(mk(sbin, 0.8))

    def update_rect(self, instance, value):
        self.rect.pos = instance.pos; self.rect.size = instance.size

    def on_touch_down(self, touch):
        if self.collide_point(*touch.pos): return super().on_touch_down(touch)
        return False

    def on_release(self):
        self.app_instance.on_stock_row_select(self)

if __name__ == '__main__':
    WMSApp().run()
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
import os
import certifi
import json

os.environ['SSL_CERT_FILE'] = certifi.where()
os.environ['REQUESTS_CA_BUNDLE'] = certifi.where()

import threading
from kivy.uix.boxlayout import BoxLayout
from kivy.properties import StringProperty
from datetime import datetime
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen, NoTransition
from kivy.clock import Clock
from kivy.factory import Factory
from kivy.core.window import Window
from supabase import create_client

SUPABASE_URL = "https://jwixigpboxcfokijxwwo.supabase.co"
SUPABASE_KEY =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Imp3aXhpZ3Bib3hjZm9r
aWp4d3dvIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NzE4MjY3NTcsImV4cCI6MjA4NzQwMjc1N30.BPOiuwzKoy8H64
_FZfGP7PLwJr2Nx6gnkaHramsvdVY"
supabase = create_client(SUPABASE_URL, SUPABASE_KEY)

KV = '''
ScreenManager:
    MenuScreen:
    TokenScreen:
    ScannerScreen:
    SOSelectionScreen:
    SODetailScreen:
    RadarSelectionScreen:
    SORadarScreen:

<MenuScreen>:
    name: 'menu_screen'
    canvas.before:
        Color:
            rgba: 0.95, 0.96, 0.98, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        padding: '20dp'
        spacing: '20dp'

        Label:
            text: 'SELECT WORK MODE'
            color: 0.1, 0.2, 0.4, 1
            font_size: '28sp'
            bold: True
            size_hint_y: None
            height: '80dp'

        Button:
            text: 'RECEIVING'
            font_size: '24sp'
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255, 1
            on_release: app.select_mode('IN')

        Button:
            text: 'STOCK OPNAME'
            font_size: '24sp'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255, 1
            on_release: app.open_so_mode_popup()

        Button:
            text: 'OUTGOING'
            font_size: '24sp'
            bold: True
            background_normal: ''
            background_color: 220/255, 20/255, 60/255, 1
            on_release: app.select_mode('OUT')

<TokenScreen>:
    name: 'token_screen'
    canvas.before:
        Color:
            rgba: 0.95, 0.95, 0.95, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        padding: '30dp'
        spacing: '20dp'

        BoxLayout:
            size_hint_y: None
            height: '40dp'
            Button:
                text: 'Back'
                size_hint_x: None
                width: '150dp'
                background_normal: ''
                background_color: 122/255, 122/255, 122/255, 1
                bold: True
                on_release: app.root.current = 'menu_screen'
            Widget:

        Label:
            id: lbl_token_mode
            text: 'SELECT TOKEN'
            font_size: '28sp'
            bold: True
            color: 44/255, 62/255, 80/255, 1
            size_hint_y: None
            height: '80dp'

        BoxLayout:
            size_hint_y: None
            height: '60dp'
            spacing: '10dp'
            Spinner:
                id: token_spinner
                text: 'Select Device...'
                values: []
                font_size: '20sp'
                color: 1, 1, 1, 1
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                on_text: app.on_token_selected(self.text)
            Button:
                text: 'DELETE'
                size_hint_x: None
                width: '110dp'
                font_size: '18sp'
                bold: True
                background_normal: ''
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                background_color: 220/255, 20/255, 60/255, 1
                on_release: app.delete_selected_device(token_spinner.text)

        Widget:

        Button:
            text: 'CONNECT & START'
            font_size: '22sp'
            bold: True
            size_hint_y: None
            height: '70dp'
            background_normal: ''
            background_color: 44/255, 62/255, 80/255, 1
            on_release: app.connect_token(token_spinner.text)

<AddDevicePopup@ModalView>:
    size_hint: None, None
    size: '300dp', '200dp'
    auto_dismiss: True
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'ADD NEW DEVICE (PC TOKEN)'
            size_hint_y: None
            height: '40dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        BoxLayout:
            padding: '20dp'
            TextInput:
                id: pop_new_dev
                hint_text: 'e.g. WH-PC-01'
                multiline: False
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
                on_release: app.cancel_add_device(); root.dismiss()
            Button:
                text: 'SAVE'
                background_color: 44/255, 62/255, 80/255, 1
                on_release: app.save_new_device(pop_new_dev.text); root.dismiss()

<ScanTableRow@ButtonBehavior+BoxLayout>:
    size_hint_y: None
    height: '75dp'
    spacing: 5
    padding: [10, 5, 5, 5]
    uid_color: 0, 0.3, 0.7, 1
    status_color: 0.1, 0.6, 0.1, 1
    bg_color: 0.95, 0.95, 0.95, 1
    canvas.before:
        Color:
            rgba: self.bg_color
        Rectangle:
            pos: self.pos
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        size_hint_x: 0.7
        spacing: 2
        Label:
            id: lbl_uid
            text: ''
            color: root.uid_color
            bold: True
            font_size: '13sp'
            text_size: self.size
            halign: 'left'
            valign: 'bottom'
            shorten: True
            shorten_from: 'right'
        Label:
            id: lbl_details
            text: 'Mat: - | - -'
            color: 0.3, 0.3, 0.3, 1
            font_size: '9sp'
            bold: True
            text_size: self.size
            halign: 'left'
            valign: 'top'
            shorten: True
            shorten_from: 'right'
    Label:
        id: lbl_status
        text: ''
        size_hint_x: 0.3
        color: root.status_color
        font_size: '12sp'
        bold: True
        text_size: self.size
        halign: 'center'
        valign: 'middle'

<ScannerScreen>:
    name: 'scanner_screen'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: '10dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: '<'
                size_hint_x: None
                width: '50dp'
                on_release: app.return_to_menu()
            Label:
                id: lbl_scan_mode
                text: 'MODE: --'
                bold: True
                font_size: '18sp'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            halign: 'left'
            valign: 'middle'
            text_size: self.size
            padding_x: '10dp'
        Label:
            id: lbl_scan_count
            text: 'QTY: 0'
            bold: True
            font_size: '18sp'
            color: 1, 0.8, 0, 1
            size_hint_x: None
            width: '80dp'

    BoxLayout:
        size_hint_y: None
        height: '60dp'
        padding: '10dp'
        spacing: '10dp'
        TextInput:
            id: manual_input
            hint_text: 'Scan / Search / Manual Input...'
            multiline: False
            font_size: '18sp'
            on_text: app.filter_apk_scans(self.text)
        Button:
            text: 'ADD'
            size_hint_x: None
            width: '80dp'
            background_color: 44/255, 62/255, 80/255, 1
            on_release: app._execute_combined_scan()

    ScrollView:
        BoxLayout:
            id: scan_list_content
            orientation: 'vertical'
            size_hint_y: None
            height: self.minimum_height
            padding: '2dp'
            spacing: '4dp'

    BoxLayout:
        size_hint_y: None
        height: '170dp'
        orientation: 'vertical'
        padding: '10dp'
        spacing: '10dp'

        GridLayout:
            cols: 2
            spacing: '10dp'
            size_hint_y: None
            height: '90dp'

            Button:
                text: 'SELECT ALL'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                on_release: app.select_all_scans()
            Button:
                text: 'ADJUST DATA'
                background_normal: ''
                background_color: 255/255, 193/255, 7/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: app.route_adjust_popup()
            Button:
                text: 'CLEAR DATA'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    background_normal: ''
                    background_color: 44/255, 62/255, 80/255, 1
                    bold: True
                    on_release: app.request_clear_scans()
                Button:
                    text: 'REPORT ITEMS'
                    background_normal: ''
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.open_bulk_report_popup()

            Button:
                text: 'SCAN'
                size_hint_y: None
                height: '50dp'
                font_size: '22sp'
                bold: True
                background_normal: ''
                background_color: 0.11, 0.30, 0.24, 1
                on_release: app.trigger_hardware_scan()

<SOBinRow@ButtonBehavior+BoxLayout>:
    size_hint_y: None
    height: '55dp'
    bg_color: 0.95, 0.95, 0.95, 1
    text_color_normal: 0, 0, 0, 1
    text_color_highlight: 0, 0.3, 0.7, 1
    canvas.before:
        Color:
            rgba: self.bg_color
        Rectangle:
            pos: self.pos
            size: self.size
    Label:
        id: lbl_sloc
        text: ''
        size_hint_x: 0.35
        color: root.text_color_normal
        bold: True
    Label:
        id: lbl_sbin
        text: ''
        size_hint_x: 0.45
        color: root.text_color_highlight
        bold: True
    Label:
        id: lbl_qty
        text: ''
        size_hint_x: 0.20
        color: root.text_color_normal
        bold: True

<SOSelectionScreen>:
    name: 'so_sel_screen'
    canvas.before:
        Color:
            rgba: 0.95, 0.95, 0.95, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: '10dp'
            canvas.before:
                Color:
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                rgba: 220/255, 20/255, 60/255, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Button:
            text: 'Back'
            size_hint_x: None
            width: '80dp'
            on_release: app.return_to_menu()
        Label:
            text: 'SELECT BIN(S)'
            bold: True
            font_size: '18sp'

    BoxLayout:
        size_hint_y: None
        height: '60dp'
        padding: '10dp'
        spacing: '10dp'
        TextInput:
            id: search_bin
            hint_text: 'Search SLOC / SBIN...'
            multiline: False
            font_size: '16sp'
            on_text: app.filter_so_bins(self.text)
        Button:
            text: 'X'
            size_hint_x: None
            width: '50dp'
            font_size: '20sp'
            bold: True
            background_normal: ''
            background_color: 0.8, 0.2, 0.2, 1
            on_release: search_bin.text = ''

    BoxLayout:
        size_hint_y: None
        height: '35dp'
        canvas.before:
            Color:
                rgba: 0.8, 0.8, 0.8, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'SLOC'
            size_hint_x: 0.35
            color: 0,0,0,1
            bold: True
        Label:
            text: 'SBIN'
            size_hint_x: 0.45
            color: 0,0,0,1
            bold: True
        Label:
            text: 'QTY'
            size_hint_x: 0.20
            color: 0,0,0,1
            bold: True

    ScrollView:
        BoxLayout:
            id: bin_list_content
            orientation: 'vertical'
            size_hint_y: None
            height: self.minimum_height
            padding: '2dp'
            spacing: '4dp'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        BoxLayout:
            size_hint_y: None
            height: '80dp'
            padding: '10dp'
            Button:
                text: 'START STOCK OPNAME'
                font_size: '22sp'
                bold: True
                background_color: 44/255, 62/255, 80/255, 1
                on_release: app.start_so_opname()

<SOModePopup@ModalView>:
    size_hint: None, None
    size: '350dp', '250dp'
    auto_dismiss: True
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height

        Label:
            text: 'SELECT SO MODE'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

        BoxLayout:
            orientation: 'vertical'
            padding: 20
            spacing: 15

            Button:
                text: 'SO SCAN LIST'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                font_size: '16sp'
                on_release: app.open_so_scan_list(); root.dismiss()

            Button:
                text: 'SO RADAR'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                bold: True
                font_size: '16sp'
                on_release: app.open_so_radar(); root.dismiss()

<SODetailScreen>:
    name: 'so_detail_screen'
    canvas.before:
        Color:
            rgba: 1, 1, 1, 1
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: '10dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: '<'
                size_hint_x: None
                width: '50dp'
                on_release: app.root.current = 'so_sel_screen'
            Label:
                id: lbl_so_bins
                text: 'BINS: -- | PROGRESS: 0/0 '
                bold: True
                font_size: '16sp'

        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: ['10dp', '5dp', '10dp', '5dp']
            spacing: '10dp'

            TextInput:
                id: so_detail_search
                hint_text: 'Search Material / Tag...'
                multiline: False
                size_hint_x: 0.6
                font_size: '15sp'
                on_text: app.apply_so_filter()

            Button:
                id: btn_so_filter
                text: 'FILTER (3)'
                size_hint_x: 0.4
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                color: 1, 1, 1, 1
                bold: True
                on_release: app.open_so_filter_popup()

        ScrollView:
            BoxLayout:
                id: so_list_content
                orientation: 'vertical'
                size_hint_y: None
                height: self.minimum_height
                padding: '2dp'
                spacing: '4dp'

        BoxLayout:
            size_hint_y: None
            height: '90dp'
            orientation: 'vertical'
            padding: '10dp'
            spacing: '10dp'

            GridLayout:
                cols: 2
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                spacing: '15dp'

                Button:
                    text: 'REPORT ITEMS'
                    font_size: '16sp'
                    background_normal: ''
                    background_color: 204/255, 0, 0, 1
                    bold: True
                    on_release: app.open_bulk_report_popup()

                Button:
                    text: 'FINISH SO'
                    font_size: '16sp'
                    background_normal: ''
                    background_color: 0.2, 0.6, 0.2, 1
                    bold: True
                    on_release: app.open_finish_so_confirm()

            TextInput:
                id: hidden_so_input
                size_hint_y: None
                height: '0dp'
                opacity: 0
                multiline: False
                on_text_validate: app._execute_combined_scan()

<RadarTableRow@ButtonBehavior+BoxLayout>:
    size_hint_y: None
    height: '55dp'
    padding: ['10dp', '2dp', '5dp', '2dp']
    spacing: '5dp'
    bg_color: 0.95, 0.95, 0.95, 1
    text_color: 0, 0, 0, 1
    canvas.before:
        Color:
            rgba: self.bg_color
        Rectangle:
            pos: self.pos
            size: self.size

    Label:
        id: lbl_loc
        text: '-'
        size_hint_x: 0.15
        color: root.text_color
        font_size: '12sp'
        bold: True
        text_size: self.size
        halign: 'left'
        valign: 'middle'
    Label:
        id: lbl_mat
        text: '-'
        size_hint_x: 0.22
        color: root.text_color
        font_size: '12sp'
        bold: True
        shorten: True
        shorten_from: 'right'
        text_size: self.size
        halign: 'left'
        valign: 'middle'
    Label:
        id: lbl_desc
        text: '-'
        size_hint_x: 0.23
        color: 0.3, 0.3, 0.3, 1
        font_size: '11sp'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        shorten: True
        shorten_from: 'right'
        text_size: self.size
        halign: 'left'
        valign: 'middle'
    Label:
        id: lbl_batch
        text: '-'
        size_hint_x: 0.16
        color: root.text_color
        font_size: '12sp'
        shorten: True
    Label:
        id: lbl_qty
        text: '0'
        size_hint_x: 0.08
        color: root.text_color
        font_size: '12sp'
    Label:
        id: lbl_status
        text: '-'
        size_hint_x: 0.16
        color: root.text_color
        bold: True
        font_size: '12sp'

<RadarSelectionScreen>:
    name: 'radar_sel_screen'
    canvas.before:
        Color:
            rgba: 0.95, 0.95, 0.95, 1
        Rectangle:
            pos: self.pos
            size: self.size
    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: '10dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: '<'
                size_hint_x: None
                width: '50dp'
                on_release: app.root.current = 'menu_screen'
            Label:
                text: 'RADAR TARGET SELECTION'
                bold: True
                font_size: '18sp'

        BoxLayout:
            size_hint_y: None
            height: '50dp'
            padding: '10dp'
            spacing: '10dp'
            TextInput:
                id: radar_search
                hint_text: 'Search Material / Tag / Desc...'
                multiline: False
                on_text: app.filter_radar_list(self.text)
            BoxLayout:
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            size_hint_x: None
            width: '140dp'
            CheckBox:
                id: radar_filter_missing
                active: False
                size_hint_x: 0.3
                color: 0,0,0,1
                on_active: app.apply_radar_filter()
            Label:
                text: 'Missing Only'
                size_hint_x: 0.7
                color: 0,0,0,1
                bold: True

    ScrollView:
        do_scroll_x: True
        do_scroll_y: True
        bar_width: '12dp'
        bar_color: 0.3, 0.3, 0.3, 0.9
        bar_inactive_color: 0.3, 0.3, 0.3, 0.4
        scroll_type: ['bars', 'content']

        BoxLayout:
            orientation: 'vertical'
            size_hint_x: None
            width: '800dp'
            size_hint_y: None
            height: self.minimum_height

            BoxLayout:
                size_hint_y: None
                height: '35dp'
                padding: ['10dp', '0dp', '5dp', '0dp']
                spacing: '5dp'
                canvas.before:
                    Color:
                        rgba: 0.8, 0.8, 0.8, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                Label:
                    text: 'LOC'
                    size_hint_x: 0.15
                    color: 0,0,0,1
                    bold: True
                    text_size: self.size
                    halign: 'left'
                    valign: 'middle'
                Label:
                    text: 'MATERIAL'
                    size_hint_x: 0.22
                    color: 0,0,0,1
                    bold: True
                    text_size: self.size
                    halign: 'left'
                    valign: 'middle'
                Label:
                    text: 'DESCRIPTION'
                    size_hint_x: 0.23
                    color: 0,0,0,1
                    bold: True
                    text_size: self.size
                    halign: 'left'
                    valign: 'middle'
                Label:
                    text: 'BATCH'
                    size_hint_x: 0.16
                    color: 0,0,0,1
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    bold: True
                Label:
                    text: 'QTY'
                    size_hint_x: 0.08
                    color: 0,0,0,1
                    bold: True
                Label:
                    text: 'STATUS'
                    size_hint_x: 0.16
                    color: 0,0,0,1
                    bold: True

            BoxLayout:
                id: radar_select_content
                orientation: 'vertical'
                size_hint_y: None
                height: self.minimum_height
                padding: '2dp'
                spacing: '2dp'

        BoxLayout:
            size_hint_y: None
            height: '80dp'
            padding: '10dp'
            Button:
                text: 'START'
                font_size: '22sp'
                bold: True
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                on_release: app.execute_start_radar()

<SORadarScreen>:
    name: 'so_radar_screen'
    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            size_hint_y: None
            height: '60dp'
            padding: '10dp'
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: '<'
                size_hint_x: None
                width: '50dp'
                on_release: app.root.current = 'radar_sel_screen'
            Label:
                text: 'RADAR TRACKING'
                bold: True
                font_size: '18sp'

        BoxLayout:
            orientation: 'vertical'
            size_hint_y: None
            height: '100dp'
            padding: '15dp'
            canvas.before:
                Color:
                    rgba: 0.9, 0.9, 0.9, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            Label:
                id: radar_lbl_mat
                text: 'MAT: -'
                color: 0, 0, 0, 1
                font_size: '16sp'
                bold: True
                text_size: self.size
                halign: 'left'
            Label:
                id: radar_lbl_desc
                text: 'DESC: -'
                color: 0.3, 0.3, 0.3, 1
                font_size: '14sp'
                text_size: self.size
                halign: 'left'
            Label:
                id: radar_lbl_loc
                text: 'EXPECTED LOC: -'
                color: 0.8, 0, 0, 1
                font_size: '16sp'
                bold: True
                text_size: self.size
                halign: 'left'

        BoxLayout:
            id: radar_signal_box
            orientation: 'vertical'
            padding: '20dp'
            canvas.before:
                Color:
                    rgba: 0.2, 0.2, 0.2, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

            Label:
                text: '(( • ))'
                font_size: '80sp'
                bold: True
                color: 1, 1, 1, 0.8

            Label:
                id: radar_status_text
                text: 'SEARCHING...'
                font_size: '30sp'
                bold: True
                color: 1, 1, 1, 1
                halign: 'center'
                valign: 'middle'
                text_size: self.size
                markup: True

        Button:
            id: btn_mark_found
            text: 'MARK AS FOUND'
            size_hint_y: None
            height: '70dp'
            background_normal: ''
            background_color: 0, 0.7, 0, 1
            disabled: True
            font_size: '20sp'
            bold: True
            on_release: app.mark_as_found()

<ConfirmClearPopup@ModalView>:
    size_hint: None, None
    size: '300dp', '200dp'
    auto_dismiss: False
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'CONFIRM DELETE'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 0.8, 0, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            id: pop_clear_msg
            text: 'Are you sure?'
            color: 0, 0, 0, 1
            halign: 'center'
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
                on_release: root.dismiss()
            Button:
                text: 'YES, DELETE'
                background_color: 0.8, 0, 0, 1
                on_release: app.execute_clear_scans(); root.dismiss()

<FinishSOPopup@ModalView>:
    size_hint: None, None
    size: '300dp', '200dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'FINISH STOCK OPNAME?'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 220/255, 20/255, 60/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        Label:
            text: 'Make sure the Admin has\\nprinted the report on the PC\\nbefore
finishing the session!'
            color: 0,0,0,1
            halign: 'center'
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                on_release: root.dismiss()
            Button:
                text: 'FINISH NOW'
                background_color: 0.2, 0.6, 0.2, 1
                on_release: app.execute_finish_so(); root.dismiss()

<WhiteSpinnerOption@SpinnerOption>:
    background_normal: ''
    background_color: 44/255, 62/255, 80/255, 1
    color: 1, 1, 1, 1
    bold: True

<ReportBulkPopup@ModalView>:
    size_hint: None, None
    size: '320dp', '350dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'REPORT PROBLEM'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 0.8, 0, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        BoxLayout:
            orientation: 'vertical'
            padding: '15dp'
            spacing: '10dp'
            Label:
                id: pop_report_info
                text: 'Reporting X Item(s)'
                color: 0, 0, 0, 1
                bold: True
                size_hint_y: None
                height: '20dp'

            Spinner:
                id: pop_issue
                text: 'Select Issue...'
                values: ['Damaged', 'Missing Item', 'Wrong Label', 'Other']
                size_hint_y: None
                height: '45dp'
                background_normal: ''
                background_color: 44/255, 62/255, 80/255, 1
                color: 1, 1, 1, 1
                bold: True
                option_cls: 'WhiteSpinnerOption'

            TextInput:
                id: pop_note
                hint_text: 'Note...'
                multiline: True
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'SUBMIT'
                background_normal: ''
                background_color: 0.8, 0, 0, 1
                bold: True
                on_release: app.submit_bulk_report(pop_issue.text, pop_note.text);
root.dismiss()

<ConfirmLocPopup@ModalView>:
    size_hint: None, None
    size: '320dp', '280dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'CONFIRMATION BOX'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 44/255, 62/255, 80/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        BoxLayout:
            orientation: 'vertical'
            padding: '20dp'

            Label:
                id: lbl_conf_count
                text: 'Adjusting - Item(s)'
                color: 0, 0.4, 0.8, 1
                bold: True
                font_size: '16sp'
                size_hint_y: 0.3

            Label:
                text: 'MATERIAL:'
                color: 0.3, 0.3, 0.3, 1
                bold: True
                font_size: '14sp'
                size_hint_y: 0.3
            Label:
                id: lbl_conf_mat
                text: '-'
                color: 0, 0, 0, 1
                bold: True
                font_size: '20sp'
                size_hint_y: 0.7
            Widget:
                size_hint_y: None
                height: '10dp'
            Label:
                text: 'STORAGE LOCATION:'
                color: 0.3, 0.3, 0.3, 1
                bold: True
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                font_size: '14sp'
                size_hint_y: 0.3
            Label:
                id: lbl_conf_loc
                text: '-'
                color: 0.8, 0, 0, 1
                bold: True
                font_size: '20sp'
                size_hint_y: 0.7
                text_size: self.width, None
                halign: 'center'
                valign: 'middle'
        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'Cancel'
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'Confirm & save'
                background_normal: ''
                background_color: 0.2, 0.6, 0.2, 1
                bold: True
                on_release: app.execute_adjust_final(); root.dismiss()

<TransAdjustPopup@ModalView>:
    size_hint: None, None
    size: '320dp', '380dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'ADJUST DATA'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 255/255, 193/255, 7/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

        ScrollView:
            id: popup_scroll_view
            BoxLayout:
                orientation: 'vertical'
                padding: '15dp'
                spacing: '5dp'
                size_hint_y: None
                height: self.minimum_height

                Label:
                    id: pop_trans_adj_info
                    text: 'Adjusting X Item(s)'
                    color: 0, 0, 0, 1
                    bold: True
                    size_hint_y: None
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        height: '20dp'

    Label:
        text: 'MATERIAL'
        color: 0, 0, 0, 1
        bold: True
        size_hint_y: None
        height: '20dp'
        halign: 'left'
        text_size: self.size
    TextInput:
        id: pop_mat
        multiline: False
        size_hint_y: None
        height: '35dp'

    Label:
        text: 'DESCRIPTION'
        color: 0, 0, 0, 1
        bold: True
        size_hint_y: None
        height: '20dp'
        halign: 'left'
        text_size: self.size
    TextInput:
        id: pop_desc
        multiline: False
        size_hint_y: None
        height: '35dp'

    Label:
        text: 'QUANTITY'
        color: 0, 0, 0, 1
        bold: True
        size_hint_y: None
        height: '20dp'
        halign: 'left'
        text_size: self.size
    TextInput:
        id: pop_qty
        input_filter: 'int'
        multiline: False
        size_hint_y: None
        height: '35dp'

    Label:
        text: 'BATCH'
        color: 0, 0, 0, 1
        bold: True
        size_hint_y: None
        height: '20dp'
        halign: 'left'
        text_size: self.size
    TextInput:
        id: pop_batch
        multiline: False
        size_hint_y: None
        height: '35dp'

    Label:
        text: 'SLOC'
        color: 0, 0, 0, 1
        bold: True
        size_hint_y: None
        height: '20dp'
        halign: 'left'
        text_size: self.size
    Spinner:
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    id: pop_sloc
                    text: 'Select SLoc'
                    values: ['82I1', '82I3', '82I5', '82I7', '82I8', '82I9', 'TR']
                    size_hint_y: None
                    height: '35dp'
                    background_normal: ''
                    background_color: 0.9, 0.9, 0.9, 1
                    color: 0, 0, 0, 1

                Label:
                    text: 'SBIN'
                    color: 0, 0, 0, 1
                    bold: True
                    size_hint_y: None
                    height: '20dp'
                    halign: 'left'
                    text_size: self.size
                TextInput:
                    id: pop_sbin
                    multiline: False
                    on_text_validate: self.focus = False
                    size_hint_y: None
                    height: '35dp'
                    on_text: app.auto_fill_sloc(self.text, 'trans')

        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'SAVE ADJUSTMENT'
                background_normal: ''
                background_color: 0.2, 0.6, 0.2, 1
                bold: True
                on_release: app.submit_trans_adjust(); root.dismiss()

<OutAdjustPopup@ModalView>:
    size_hint: None, None
    size: '320dp', '400dp'
    auto_dismiss: False
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: 'OUTGOING PICK'
            size_hint_y: None
            height: '40dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 255/255, 140/255, 0, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

        ScrollView:
            BoxLayout:
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        orientation: 'vertical'
        padding: '15dp'
        spacing: '5dp'
        size_hint_y: None
        height: self.minimum_height

        Label:
            id: pop_out_info
            text: 'Mat: -'
            color: 0, 0, 0, 1
            bold: True
            size_hint_y: None
            height: '25dp'
        Label:
            id: pop_out_stock
            text: 'Stock Gudang: 0'
            color: 0.1, 0.6, 0.1, 1
            bold: True
            size_hint_y: None
            height: '20dp'

        Label:
            text: 'QTY TO PICK (OUT):'
            color: 0.3, 0.3, 0.3, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
        TextInput:
            id: pop_out_qty
            input_filter: 'int'
            multiline: False
            size_hint_y: None
            height: '35dp'
            font_size: '18sp'

        Label:
            text: 'SLOC'
            color: 0.3, 0.3, 0.3, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
        Spinner:
            id: pop_out_sloc
            text: 'Select SLoc'
            values: ['82I1', '82I3', '82I5', '82I7', '82I8', '82I9', 'TR']
            size_hint_y: None
            height: '35dp'
            background_normal: ''
            background_color: 0.9, 0.9, 0.9, 1
            color: 0, 0, 0, 1

        Label:
            text: 'SBIN'
            color: 0.3, 0.3, 0.3, 1
            bold: True
            size_hint_y: None
            height: '20dp'
            halign: 'left'
            text_size: self.size
        TextInput:
            id: pop_out_sbin
            multiline: False
            on_text_validate: self.focus = False
            size_hint_y: None
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    height: '35dp'
                    on_text: app.auto_fill_sloc(self.text, 'out')

        BoxLayout:
            size_hint_y: None
            height: '50dp'
            Button:
                text: 'CANCEL'
                background_normal: ''
                background_color: 180/255, 180/255, 180/255, 1
                color: 0, 0, 0, 1
                bold: True
                on_release: root.dismiss()
            Button:
                text: 'SAVE PICK'
                background_normal: ''
                background_color: 0.2, 0.6, 0.2, 1
                bold: True
                on_release: app.submit_out_adjust(); root.dismiss()

<SOFilterPopup@ModalView>:
    size_hint: None, None
    size: '300dp', '240dp'
    auto_dismiss: True
    BoxLayout:
        orientation: 'vertical'
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
            Color:
                rgba: 0, 0, 0, 1
            Line:
                width: 1.5
                rectangle: self.x, self.y, self.width, self.height
        Label:
            text: 'CHECKLIST STATUS'
            size_hint_y: None
            height: '35dp'
            bold: True
            canvas.before:
                Color:
                    rgba: 44/255, 62/255, 80/255, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
        BoxLayout:
            orientation: 'vertical'
            padding: ['10dp', '5dp']
            spacing: '2dp'

            BoxLayout:
                size_hint_y: None
                height: '35dp'
                CheckBox:
                    id: chk_found
                    active: True
                    size_hint_x: None
                    width: '40dp'
                    color: 0,0,0,1
                    on_active: app.apply_so_filter()
                Label:
                    text: 'FOUND'
                    color: 0,0,0,1
                    bold: True
                    halign: 'left'
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    text_size: self.size
                    valign: 'middle'

            BoxLayout:
                size_hint_y: None
                height: '35dp'
                CheckBox:
                    id: chk_waiting
                    active: True
                    size_hint_x: None
                    width: '40dp'
                    color: 0,0,0,1
                    on_active: app.apply_so_filter()
                Label:
                    text: 'WAITING'
                    color: 0,0,0,1
                    bold: True
                    halign: 'left'
                    text_size: self.size
                    valign: 'middle'

            BoxLayout:
                size_hint_y: None
                height: '35dp'
                CheckBox:
                    id: chk_reported
                    active: True
                    size_hint_x: None
                    width: '40dp'
                    color: 0,0,0,1
                    on_active: app.apply_so_filter()
                Label:
                    text: 'REPORTED'
                    color: 0,0,0,1
                    bold: True
                    halign: 'left'
                    text_size: self.size
                    valign: 'middle'

            BoxLayout:
                size_hint_y: None
                height: '35dp'
                CheckBox:
                    id: chk_extra
                    active: False
                    size_hint_x: None
                    width: '40dp'
                    color: 0,0,0,1
                    on_active: app.apply_so_filter()
                Label:
                    text: 'EXTRA ITEMS'
                    color: 0,0,0,1
                    bold: True
                    halign: 'left'
                    text_size: self.size
                    valign: 'middle'

        Button:
            text: 'CLOSE'
            size_hint_y: None
            height: '40dp'
            background_normal: ''
            background_color: 220/255, 20/255, 60/255, 1
            bold: True
            on_release: root.dismiss()
'''

class TokenScreen(Screen): pass
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
class MenuScreen(Screen): pass
class ScannerScreen(Screen): pass
class SOSelectionScreen(Screen): pass
class SODetailScreen(Screen): pass
class RadarSelectionScreen(Screen): pass
class SORadarScreen(Screen): pass

class ScannerApp(App):
    mode = StringProperty("")
    pending_mode = StringProperty("")

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.full_bin_list = []
        self.custom_devices = {"IN": [], "SO": [], "OUT": []}
        self._load_custom_devices()

        self.SLOC_RULES = {
            "82I1": ("AA", "AP"), "82I3": ("AQ", "BF"), "82I5": ("BG", "BL"),
            "82I7": ("BM", "BV"), "82I8": ("BW", "BX"), "82I9": ("BY", "CT")
        }

    def _load_custom_devices(self):
        try:
            device_file = os.path.join(self.user_data_dir, 'devices.json')
            if os.path.exists(device_file):
                with open(device_file, 'r') as f:
                    data = json.load(f)
                    if isinstance(data, dict):
                        for k in ["IN", "SO", "OUT"]:
                            if k in data:
                                self.custom_devices[k] = data[k]
        except: pass

    def _save_custom_devices(self):
        try:
            device_file = os.path.join(self.user_data_dir, 'devices.json')
            with open(device_file, 'w') as f:
                json.dump(self.custom_devices, f)
        except: pass

    def get_sloc_from_sbin(self, sbin_prefix):
        val = sbin_prefix.upper()
        for sloc, (start, end) in self.SLOC_RULES.items():
            if start <= val <= end: return sloc
        return None

    def is_valid_sbin(self, sbin):
        sbin = sbin.upper()
        if len(sbin) != 6: return False
        prefix = sbin[:2]
        num_part = sbin[2:]
        if not prefix.isalpha() or not num_part.isdigit(): return False
        if not self.get_sloc_from_sbin(prefix): return False
        try:
            r = int(num_part[:2])
            c = int(num_part[2:])
            if not (1 <= r <= 81 and 1 <= c <= 8): return False
        except: return False
        return True

    def auto_fill_sloc(self, text, popup_type):
        val = text.strip().upper()
        if len(val) >= 2:
            prefix = val[:2]
            sloc = None
            if prefix == "TR":
                sloc = "TR"
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            else:
                sloc = self.get_sloc_from_sbin(prefix)

            if sloc:
                if popup_type == 'trans' and hasattr(self, 'trans_adj_popup') and
self.trans_adj_popup:
                    self.trans_adj_popup.ids.pop_sloc.text = sloc
                elif popup_type == 'out' and hasattr(self, 'out_adj_popup') and
self.out_adj_popup:
                    self.out_adj_popup.ids.pop_out_sloc.text = sloc

    def build(self):
        self.pc_token = ""
        self.session_scans = []
        self.scan_buffer = ""
        self.scan_timer = None
        self.recent_scans = set()

        self.epc_length_history = []
        self.selected_scan_uids = []

        Window.bind(on_key_down=self.on_key_down)
        root_widget = Builder.load_string(KV)
        root_widget.transition = NoTransition()
        return root_widget

    def toggle_scan_row(self, uid, row_widget):
        if uid in self.selected_scan_uids:
            self.selected_scan_uids.remove(uid)
            row_widget.bg_color = (0.95, 0.95, 0.95, 1)
        else:
            self.selected_scan_uids.append(uid)
            row_widget.bg_color = (0.6, 0.8, 1, 1)

    def on_key_down(self, window, key, scancode, codepoint, modifiers):
        screen = self.root.current
        is_search_bar_focused = False
        if screen == 'radar_sel_screen':
            if self.root.get_screen('radar_sel_screen').ids.radar_search.focus:
                is_search_bar_focused = True

        if 'ctrl' in modifiers and key == 118 and not is_search_bar_focused:
            from kivy.core.clipboard import Clipboard
            teks_paste = Clipboard.paste()
            self.scan_buffer += teks_paste
            if screen == 'so_radar_screen':
                self.process_radar_scan()
            return True

        if key == 27:
            popups = ['report_popup', 'adjust_popup', 'finish_so_popup', 'so_mode_popup']
            for p in popups:
                if hasattr(self, p) and getattr(self, p) and getattr(self, p).parent:
                    getattr(self, p).dismiss()
                    return True

        screen = self.root.current

        if key == 27:
            if screen == 'menu_screen': return False
            elif screen == 'so_detail_screen':
                self.root.current = 'so_sel_screen'
                return True
            elif screen == 'so_radar_screen':
                self.root.current = 'radar_sel_screen'
                return True
            else:
                self.return_to_menu()
```

UNIVERSITAS ATMA JAYA YOGYAKARTA

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                return True

        if screen in ['scanner_screen', 'so_detail_screen', 'so_radar_screen']:
            if is_search_bar_focused:
                return False

            if not codepoint and key not in [13, 271, 27, 118]:
                if screen == 'so_radar_screen':
                    radar_screen = self.root.get_screen('so_radar_screen')
                    if radar_screen.ids.radar_status_text.text == "STANDBY":
                        radar_screen.ids.radar_status_text.text = "SCANNING..."
                        radar_screen.ids.radar_signal_box.canvas.before.children[0].rgba =
(0.2, 0.2, 0.2, 1)

                    if hasattr(self, 'idle_timer') and self.idle_timer:
                        self.idle_timer.cancel()
                    self.idle_timer = Clock.schedule_once(self.set_standby, 4.0)

            if key in [13, 271]:
                self.scan_buffer += " "

                if screen == 'so_radar_screen':
                    radar_screen = self.root.get_screen('so_radar_screen')
                    if radar_screen.ids.radar_status_text.text == "STANDBY":
                        radar_screen.ids.radar_status_text.text = "SCANNING..."
                        radar_screen.ids.radar_signal_box.canvas.before.children[0].rgba =
(0.2, 0.2, 0.2, 1)

                    self.process_radar_scan()
                else:
                    if self.scan_timer: self.scan_timer.cancel()
                    self.scan_timer = Clock.schedule_once(lambda dt:
self._execute_combined_scan(), 0.2)
                return True

            if codepoint and codepoint.isprintable():
                self.scan_buffer += codepoint

                if len(self.scan_buffer) > 1000:
                    self.scan_buffer = self.scan_buffer[-1000:]

                if screen == 'so_radar_screen':
                    if hasattr(self, 'radar_target_uid'):
                        import re
                        clean_target = re.sub(r'[^A-Z0-9]', '',
self.radar_target_uid.upper())
                        clean_buffer = re.sub(r'[^A-Z0-9]', '', self.scan_buffer.upper())

                        if clean_target in clean_buffer:
                            self.process_radar_scan()
                else:
                    if self.scan_timer: self.scan_timer.cancel()
                    self.scan_timer = Clock.schedule_once(lambda dt:
self._execute_combined_scan(), 0.2)

                return True

        return False

    def _execute_combined_scan(self):
        screen = self.root.current
        import re

        clean_buffer = re.sub(r'[^A-Za-z0-9\s]', '', self.scan_buffer)

        if screen == 'scanner_screen':
            box = self.root.get_screen('scanner_screen').ids.manual_input
            gabungan = clean_buffer + " " + box.text
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            if gabungan.strip():
                self.process_scan(gabungan)

            self.scan_buffer = ""
            box.text = ""
            box.focus = False

        elif screen == 'so_detail_screen':
            box = self.root.get_screen('so_detail_screen').ids.hidden_so_input
            gabungan = clean_buffer + " " + box.text

            if gabungan.strip():
                self.process_so_scan(gabungan)

            self.scan_buffer = ""
            box.text = ""
            box.focus = False

    def lock_radar_target(self, uid):
        self.radar_target_uid = uid
        screen = self.root.get_screen('so_radar_screen')

        target_data = getattr(self, 'radar_full_data', {}).get(uid, {})
        mat = target_data.get('MATERIAL', 'UNKNOWN MAT')
        desc = target_data.get('DESCRIPTION', '-')
        sloc = target_data.get('SLOC', '-')
        sbin = target_data.get('SBIN', '-')
        loc = f"{sloc} - {sbin}"

        screen.ids.radar_lbl_mat.text = f"MAT: {mat}  ({uid[-6:]})"
        screen.ids.radar_lbl_desc.text = f"DESC: {desc}"
        screen.ids.radar_lbl_loc.text = f"EXPECTED LOC: {loc}"

        screen.ids.radar_status_text.text = "STANDBY"
        screen.ids.radar_signal_box.canvas.before.children[0].rgba = (0.2, 0.2, 0.2, 1)
        screen.ids.btn_mark_found.disabled = True

    def process_radar_scan(self):
        import re
        raw_input = self.scan_buffer.strip().upper()
        self.scan_buffer = ""

        screen = self.root.get_screen('so_radar_screen')

        if hasattr(self, 'idle_timer') and self.idle_timer:
            self.idle_timer.cancel()
        self.idle_timer = Clock.schedule_once(self.set_standby, 4.0)

        if not raw_input:
            return

        if self.mode == "SO_RADAR" and hasattr(self, 'radar_target_uid'):
            clean_input = re.sub(r'[^A-Z0-9]', '', raw_input)
            clean_target = re.sub(r'[^A-Z0-9]', '', self.radar_target_uid.upper())

            if clean_target in clean_input:
                screen.ids.radar_signal_box.canvas.before.children[0].rgba = (0, 0.8, 0,
1)
                screen.ids.btn_mark_found.disabled = False

                screen.ids.radar_status_text.text =
f"FOUND:\n[size=18sp]{self.radar_target_uid}[/size]"

                if hasattr(self, 'radar_timer') and self.radar_timer:
                    self.radar_timer.cancel()
                self.radar_timer = Clock.schedule_once(self.target_lost, 4.0)
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
    def target_lost(self, dt):
        screen = self.root.get_screen('so_radar_screen')
        screen.ids.radar_signal_box.canvas.before.children[0].rgba = (0.8, 0, 0, 1)
        screen.ids.radar_status_text.text = "SIGNAL LOST..."

    def set_standby(self, dt):
        screen = self.root.get_screen('so_radar_screen')
        screen.ids.radar_signal_box.canvas.before.children[0].rgba = (0.2, 0.2, 0.2, 1)
        screen.ids.radar_status_text.text = "STANDBY"

    def _update_scan_count(self):
        try:
            screen = self.root.get_screen('scanner_screen')
            screen.ids.lbl_scan_count.text = f"QTY: {len(self.session_scans)}"
        except: pass

    def open_so_radar(self):
        self.selected_radar_uid = None
        screen = self.root.get_screen('radar_sel_screen')
        screen.ids.radar_search.text = ""
        screen.ids.radar_filter_missing.active = False
        screen.ids.radar_select_content.clear_widgets()
        self.root.current = 'radar_sel_screen'

        threading.Thread(target=self._thread_load_radar_data, daemon=True).start()

    def _thread_load_radar_data(self):
        try:
            res_stock = supabase.table("STOCK").select("*").execute()
            stock_data = res_stock.data if res_stock.data else []

            res_report = supabase.table("REPORT_SO").select("RFID_UID").eq("REPORT",
"Missing Item").execute()
            missing_uids = set(r['RFID_UID'] for r in res_report.data) if res_report.data
else set()

            radar_full_list = []
            for item in stock_data:
                uid = item['RFID_UID']
                item['is_missing'] = uid in missing_uids
                radar_full_list.append(item)

            radar_full_list.sort(key=lambda x: (str(x.get('SLOC', '')), str(x.get('SBIN',
'''')))

            self.radar_full_data = {item['RFID_UID']: item for item in radar_full_list}

            Clock.schedule_once(lambda dt: self._render_radar_list(radar_full_list))
        except Exception as e:
            print(f"Error compiling Radar inventory matrix: {e}")

    def _render_radar_list(self, data_list):
        content = self.root.get_screen('radar_sel_screen').ids.radar_select_content
        content.clear_widgets()

        self.radar_row_widgets = []

        for info in data_list:
            uid = info['RFID_UID']
            is_missing = info.get('is_missing', False)
            status_text = "Missing" if is_missing else "-"

            mat = str(info.get('MATERIAL') or "-")
            desc = str(info.get('DESCRIPTION') or "-")
            sloc = str(info.get('SLOC') or "-")
            sbin = str(info.get('SBIN') or "-")
            loc = f"{sloc} {sbin}"
            batch = str(info.get('BATCH') or "-")
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            qty = str(info.get('QTY') or "0")

            row = Factory.RadarTableRow()
            row.rfid_tag = uid
            row.is_missing = is_missing
            row.search_string = f"{mat} {desc} {loc} {batch} {uid}".lower()

            row.ids.lbl_mat.text = mat
            row.ids.lbl_desc.text = desc
            row.ids.lbl_loc.text = loc
            row.ids.lbl_batch.text = batch
            row.ids.lbl_qty.text = qty
            row.ids.lbl_status.text = status_text

            if is_missing:
                row.ids.lbl_status.color = (0.8, 0, 0, 1)
            else:
                row.ids.lbl_status.color = (0.3, 0.3, 0.3, 1)

            if hasattr(self, 'selected_radar_uid') and self.selected_radar_uid == uid:
                row.bg_color = (0.6, 0.8, 1, 1)

            row.bind(on_release=lambda instance, u=uid: self.toggle_radar_selection(u,
instance))

            self.radar_row_widgets.append(row)

        self.apply_radar_filter()

    def apply_radar_filter(self, *args):
        screen = self.root.get_screen('radar_sel_screen')
        is_missing_only = screen.ids.radar_filter_missing.active
        query = screen.ids.radar_search.text.lower()
        content = screen.ids.radar_select_content

        if not hasattr(self, 'radar_row_widgets'):
            return

        content.clear_widgets()

        for row in self.radar_row_widgets:
            match_query = (not query) or (query in row.search_string)
            match_status = (not is_missing_only) or row.is_missing

            if match_query and match_status:
                content.add_widget(row)

    def filter_radar_list(self, text):
        self.apply_radar_filter()

    def start_single_radar_hunt(self, uid):
        self.mode = "SO_RADAR"
        self.pc_token = "RADAR-DEVICE"
        self.root.current = 'so_radar_screen'
        self.lock_radar_target(uid)

    def mark_as_found(self):
        if not hasattr(self, 'radar_target_uid'): return

        self.root.current = 'radar_sel_screen'

        uid_to_delete = self.radar_target_uid
        threading.Thread(target=self._thread_execute_mark_found, args=(uid_to_delete,),
daemon=True).start()

    def _thread_execute_mark_found(self, target_uid):
        try:
            supabase.table("REPORT_SO").delete().eq("RFID_UID", target_uid).execute()
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            print(f"Target {target_uid} marked as found!")

            self._thread_load_radar_data()
        except Exception as e:
            print(f"Failed Mark as Found in DB: {e}")

    def toggle_radar_selection(self, uid, row_widget):
        content = self.root.get_screen('radar_sel_screen').ids.radar_select_content

        if hasattr(self, 'selected_radar_uid') and self.selected_radar_uid == uid:
            self.selected_radar_uid = None
            row_widget.bg_color = (0.95, 0.95, 0.95, 1)
        else:
            self.selected_radar_uid = uid
            for row in content.children:
                row.bg_color = (0.95, 0.95, 0.95, 1)
            row_widget.bg_color = (0.6, 0.8, 1, 1)

    def execute_start_radar(self):
        if hasattr(self, 'selected_radar_uid') and self.selected_radar_uid:
            self.start_single_radar_hunt(self.selected_radar_uid)
        else:
            print("Pilih material dulu dari tabel!")

    def select_mode(self, mode):
        self.pending_mode = mode
        mode_text = "RECEIVING" if mode == "IN" else "OUTGOING" if mode == "OUT" else
"STOCK OPNAME"
        self.root.get_screen('token_screen').ids.lbl_token_mode.text = f"{mode_text}
LOGIN"

        spinner = self.root.get_screen('token_screen').ids.token_spinner
        base_values = ['SYG-DSK-0605']
        mode_custom_devices = self.custom_devices.get(mode, [])
        all_values = base_values + mode_custom_devices + ['+ ADD NEW DEVICE']
        spinner.values = all_values

        if spinner.text not in all_values and spinner.text != 'Select Device...':
            spinner.text = all_values[0] if all_values else 'Select Device...'

        self.root.current = 'token_screen'

    def on_token_selected(self, text):
        if text == '+ ADD NEW DEVICE':
            self.root.get_screen('token_screen').ids.token_spinner.text = 'Select
Device...'
            if not hasattr(self, 'add_dev_popup') or not self.add_dev_popup:
                self.add_dev_popup = Factory.AddDevicePopup()
            self.add_dev_popup.ids.pop_new_dev.text = ''
            self.add_dev_popup.open()

    def cancel_add_device(self):
        self.select_mode(self.pending_mode)

    def save_new_device(self, dev_name):
        dev_name = dev_name.strip().upper()
        mode = self.pending_mode
        if dev_name:
            if mode not in self.custom_devices:
                self.custom_devices[mode] = []
            if dev_name in self.custom_devices[mode]:
                self.custom_devices[mode].remove(dev_name)
            self.custom_devices[mode].append(dev_name)
            if len(self.custom_devices[mode]) > 4:
                self.custom_devices[mode] = self.custom_devices[mode][-4:]
            self._save_custom_devices()
        self.select_mode(self.pending_mode)
        self.root.get_screen('token_screen').ids.token_spinner.text = dev_name
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
    def delete_selected_device(self, dev_name):
        if dev_name in ['Select Device...', '+ ADD NEW DEVICE', 'SYG-DSK-0605']:
            return

        mode = self.pending_mode
        if mode in self.custom_devices and dev_name in self.custom_devices[mode]:
            self.custom_devices[mode].remove(dev_name)
            self._save_custom_devices()

        self.select_mode(mode)
        self.root.get_screen('token_screen').ids.token_spinner.text = 'Select Device...'

    def connect_token(self, token):
        if token == 'Select Device...' or token == '+ ADD NEW DEVICE': return
        self.pc_token = token
        self.open_scanner(self.pending_mode)

    def open_so_mode_popup(self):
        if not hasattr(self, 'so_mode_popup') or not self.so_mode_popup:
            self.so_mode_popup = Factory.SOModePopup()
        self.so_mode_popup.open()

    def open_so_scan_list(self):
        self.select_mode("SO")

    def return_to_menu(self):
        if hasattr(self, 'sync_event') and self.sync_event:
            self.sync_event.cancel()
        self.selected_scan_uids.clear()
        self.root.current = 'menu_screen'

    def open_scanner(self, mode):
        self.mode = mode
        self.epc_length_history.clear()
        self.selected_scan_uids.clear()
        if hasattr(self, 'recent_scans'): self.recent_scans.clear()

        if mode == "SO":
            self.root.current = 'so_sel_screen'
            self.root.get_screen('so_sel_screen').ids.bin_list_content.clear_widgets()
            self.so_active_bins = set()
            if hasattr(self, 'sync_event') and self.sync_event: self.sync_event.cancel()
            threading.Thread(target=self._thread_load_bins, daemon=True).start()
        else:
            self.session_scans.clear()
            self._update_scan_count()
            self.root.get_screen('scanner_screen').ids.scan_list_content.clear_widgets()
            mode_text = "RECEIVING" if mode == "IN" else "OUTGOING"
            self.root.get_screen('scanner_screen').ids.lbl_scan_mode.text = f"{mode_text}
| {self.pc_token}"
            self.root.current = 'scanner_screen'
            if hasattr(self, 'sync_event') and self.sync_event: self.sync_event.cancel()
            self.sync_event = Clock.schedule_interval(lambda dt: self._trigger_sync(), 5)

    def trigger_hardware_scan(self):
        if self.mode == "SO":
            self.root.get_screen('so_detail_screen').ids.hidden_so_input.focus = True
        else:
            self.root.get_screen('scanner_screen').ids.manual_input.focus = True

    def is_valid_epc(self, epc):
        import string
        if not all(c in string.hexdigits for c in epc): return False
        if len(epc) < 16 or len(epc) % 2 != 0: return False

        if not epc.startswith("E200"):
            return False
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        if len(self.epc_length_history) < 5:
            self.epc_length_history.append(len(epc))
            return True
        else:
            from collections import Counter
            mayoritas = Counter(self.epc_length_history).most_common(1)[0][0]
            if len(epc) == mayoritas: return True
            else: return False

    def filter_apk_scans(self, query):
        query = query.upper()
        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        for row in content.children:
            uid_val = getattr(row, 'rfid_tag', '')
            text_to_search = f"{uid_val} {row.ids.lbl_uid.text}
{row.ids.lbl_details.text}".upper()
            if query in text_to_search:
                row.height = '75dp'
                row.opacity = 1
            else:
                row.height = 0
                row.opacity = 0

    def process_scan(self, raw_input):
        uids = [u.strip().upper() for u in raw_input.split() if u.strip()]
        for uid in uids:
            if not self.is_valid_epc(uid): continue
            if uid in self.session_scans: continue
            self.session_scans.insert(0, uid)
            self.recent_scans.add(uid)
            self._add_ui_widget(uid, "SYNCING...", {})
            threading.Thread(target=self._thread_process_and_push, args=(uid,),
daemon=True).start()
        self._update_scan_count()

    def _thread_process_and_push(self, uid):
        status = "OK"
        detail_data = {}

        if self.mode == "OUT":
            try:
                res_stock = supabase.table("STOCK").select("*").eq("RFID_UID",
uid).execute()
                if res_stock.data:
                    detail_data = res_stock.data[0]
                    status = "OK"
                else:
                    status = "NOT IN STOCK"
            except Exception as e:
                print(f"Error checking stock: {e}")

        elif self.mode == "IN":
            try:
                cek_stock = supabase.table("STOCK").select("RFID_UID").eq("RFID_UID",
uid).execute()
                if cek_stock.data:
                    status = "IN STOCK"
                else:
                    status = "OK"
            except: pass

        try:
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            payload = {
                "RFID_UID": uid,
                "PC_TOKEN": self.pc_token,
                "TIMESTAMP": timestamp_now,
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                "STATUS": status
            }

            if self.mode == "OUT":
                payload.update({
                    "MATERIAL": detail_data.get("MATERIAL", ""),
                    "DESCRIPTION": detail_data.get("DESCRIPTION", ""),
                    "QTY": detail_data.get("QTY", 0),
                    "SLOC": detail_data.get("SLOC", ""),
                    "SBIN": detail_data.get("SBIN", ""),
                    "BATCH": detail_data.get("BATCH", "")
                })

            supabase.table(f"SCAN_{self.mode}").upsert(payload).execute()
            Clock.schedule_once(lambda dt: self._update_row_status(uid, status))
        except Exception as e:
            Clock.schedule_once(lambda dt: self._update_row_status(uid, "ERR / NO NET"))

    def _update_row_status(self, uid, new_status):
        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        for row in content.children:
            raw_row_uid = getattr(row, 'rfid_tag',
row.ids.lbl_uid.text.split('\n')[0].strip())
            if raw_row_uid == uid:
                row.ids.lbl_status.text = new_status
                if "ERR" in new_status:
                    row.status_color = (0.8, 0, 0, 1)
                elif new_status == "OK":
                    row.status_color = (0.1, 0.6, 0.1, 1)
                    row.uid_color = (0, 0.3, 0.7, 1) if self.mode == "OUT" else (0, 0, 0,
1)
                elif new_status == "NOT IN STOCK":
                    row.status_color = (0.8, 0, 0, 1)
                    row.uid_color = (0.8, 0, 0, 1)
                elif new_status == "IN STOCK":
                    row.status_color = (0.8, 0, 0, 1)
                elif new_status == "REPORTED":
                    row.status_color = (0.8, 0, 0, 1)
                elif new_status == "ADJUSTED":
                    row.status_color = (1, 0.6, 0, 1)

    def _trigger_sync(self):
        if not self.session_scans: return
        threading.Thread(target=self._thread_fetch_and_compare, daemon=True).start()

    def _thread_fetch_and_compare(self):
        try:
            res = supabase.table(f"SCAN_{self.mode}").select("*").eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", self.session_scans).execute()
            if res.data is None: return
            db_data_dict = {row['RFID_UID']: row for row in res.data}

            self.session_scans = [uid for uid in self.session_scans if uid in db_data_dict
or uid in self.recent_scans]
            self.recent_scans.clear()
            Clock.schedule_once(lambda dt: self._rebuild_ui_list(db_data_dict))
        except: pass

    def _rebuild_ui_list(self, db_data_dict=None):
        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        db_data_dict = db_data_dict or {}

        existing_widgets = {}
        for row in content.children:
            u_id = getattr(row, 'rfid_tag', '')
            existing_widgets[u_id] = row

        for uid in reversed(self.session_scans):
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            db_row = db_data_dict.get(uid, {})

            if db_row:
                if not hasattr(self, 'local_item_cache'): self.local_item_cache = {}
                self.local_item_cache[uid] = db_row

            if uid in existing_widgets:
                row_widget = existing_widgets[uid]

                if db_row:
                    mat = db_row.get("MATERIAL") or "-"
                    sloc = db_row.get("SLOC") or "-"
                    sbin = db_row.get("SBIN") or "-"
                    batch = db_row.get("BATCH") or "-"
                    qty = db_row.get("QTY") or "0"

                    if self.mode == "OUT":
                        row_widget.ids.lbl_uid.text = str(mat) if mat and mat != "-" else
"UNKNOWN MAT"
                        row_widget.ids.lbl_details.text = f"{sloc} {sbin} | Batch: {batch}
| Qty: {qty}"
                    else:
                        row_widget.ids.lbl_details.text = f"Mat: {mat} | Batch: {batch} |
{sloc} {sbin} | Qty: {qty}"

                new_stat = db_row.get("STATUS", "OK") if db_row else
row_widget.ids.lbl_status.text
                row_widget.ids.lbl_status.text = new_stat

                if new_stat == "REPORTED":
                    row_widget.status_color = (0.8, 0, 0, 1)
                    row_widget.uid_color = (0, 0, 0, 1)
                elif new_stat == "ADJUSTED":
                    row_widget.status_color = (1, 0.6, 0, 1)
                    row_widget.uid_color = (0, 0, 0, 1)
                elif new_stat == "NOT IN STOCK":
                    row_widget.status_color = (0.8, 0, 0, 1)
                    row_widget.uid_color = (0.8, 0, 0, 1)
                elif "ERR" in new_stat:
                    row_widget.status_color = (0.8, 0, 0, 1)
                else:
                    row_widget.status_color = (0.1, 0.6, 0.1, 1)
                    row_widget.uid_color = (0, 0.3, 0.7, 1) if self.mode == "OUT" else (0,
0, 0, 1)
            else:
                status = db_row.get("STATUS", "SYNCING...") if db_row else "SYNCING..."
                self._add_ui_widget(uid, status, db_row)

        for uid, widget in existing_widgets.items():
            if uid not in self.session_scans:
                content.remove_widget(widget)

        self._update_scan_count()

    def _add_ui_widget(self, uid, status, extra_data=None):
        if extra_data is None: extra_data = {}
        if status is None: status = "CLEAR" if self.mode == "IN" else "OK"

        row = Factory.ScanTableRow()
        row.rfid_tag = str(uid)

        mat = extra_data.get("MATERIAL") or "-"
        sloc = extra_data.get("SLOC") or "-"
        sbin = extra_data.get("SBIN") or "-"
        batch = extra_data.get("BATCH") or "-"
        qty = extra_data.get("QTY") or "0"

        if self.mode == "OUT":
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            row.ids.lbl_uid.text = str(mat) if mat and mat != "-" else "UNKNOWN MAT"
            row.ids.lbl_details.text = f"{sloc} {sbin} | Batch: {batch} | Qty: {qty}"
        else:
            row.ids.lbl_uid.text = str(uid)
            row.ids.lbl_details.text = f"Mat: {mat} | Batch: {batch} | {sloc} {sbin} |
Qty: {qty}"

        if uid in self.selected_scan_uids:
            row.bg_color = (0.6, 0.8, 1, 1)
        row.bind(on_release=lambda instance, u=uid: self.toggle_scan_row(u, instance))

        if status in ["REPORTED", "ADJUSTED"]:
            row.ids.lbl_status.text = status
            if status == "REPORTED": row.status_color = (0.8, 0, 0, 1)
            elif status == "ADJUSTED": row.status_color = (1, 0.6, 0, 1)
            else: row.status_color = (0.5, 0, 0.5, 1)
            row.uid_color = (0, 0, 0, 1)
        elif self.mode == "IN":
            row.uid_color = (0, 0, 0, 1)
            if status == "IN STOCK":
                row.ids.lbl_status.text = "IN STOCK"
                row.status_color = (0.8, 0, 0, 1)
            elif status in ["CLEAR", "Sent", "OK", "SCANNED"]:
                row.ids.lbl_status.text = "OK"
                row.status_color = (0.1, 0.6, 0.1, 1)
            elif status == "SYNCING...":
                row.ids.lbl_status.text = "Syncing..."
                row.status_color = (1, 0.6, 0, 1)
            elif status == "READY":
                row.ids.lbl_status.text = "READY"
                row.status_color = (0.1, 0.6, 0.1, 1)
            elif status == "TRANSIT":
                row.ids.lbl_status.text = "TRANSIT"
                row.status_color = (0.8, 0.6, 0, 1)
            elif status == "INCOMPLETE":
                row.ids.lbl_status.text = "INCOMPLETE"
                row.status_color = (0.5, 0.5, 0.5, 1)
            else:
                row.ids.lbl_status.text = status
                row.status_color = (0.5, 0.5, 0.5, 1)
        else:
            if status == "NOT IN STOCK":
                row.uid_color = (0.8, 0, 0, 1); row.status_color = (0.8, 0, 0, 1)
                row.ids.lbl_status.text = "NOT IN STOCK!"
            elif status == "SYNCING...":
                row.uid_color = (1, 0.6, 0, 1); row.status_color = (1, 0.6, 0, 1)
                row.ids.lbl_status.text = "Syncing..."
            else:
                row.uid_color = (0, 0.3, 0.7, 1); row.status_color = (0.1, 0.6, 0.1, 1)
                row.ids.lbl_status.text = "OK"

        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        content.add_widget(row, index=len(content.children))

    def request_clear_scans(self):
        self.uids_to_clear = list(self.selected_scan_uids)

        if not hasattr(self, 'clear_popup') or not self.clear_popup:
            self.clear_popup = Factory.ConfirmClearPopup()

        if self.uids_to_clear:
            self.clear_popup.ids.pop_clear_msg.text = f"Delete {len(self.uids_to_clear)}
selected item(s)?"
        else:
            if not self.session_scans: return
            self.uids_to_clear = list(self.session_scans)
            self.clear_popup.ids.pop_clear_msg.text = "Delete ALL items in this session?"
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        self.clear_popup.open()

    def execute_clear_scans(self):
        if not self.uids_to_clear: return

        self.session_scans = [uid for uid in self.session_scans if uid not in
self.uids_to_clear]
        self.selected_scan_uids = [u for u in self.selected_scan_uids if u not in
self.uids_to_clear]

        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        rows_to_remove = []
        for row in content.children:
            raw_uid = getattr(row, 'rfid_tag', '')
            if raw_uid in self.uids_to_clear:
                rows_to_remove.append(row)

        for row in rows_to_remove:
            content.remove_widget(row)

        self._update_scan_count()

        uids_copy = list(self.uids_to_clear)
        threading.Thread(target=self._thread_delete_supabase, args=(uids_copy,),
daemon=True).start()

        self.uids_to_clear = []

    def select_all_scans(self):
        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        self.selected_scan_uids.clear()
        for row in content.children:
            raw_uid = getattr(row, 'rfid_tag', '')
            self.selected_scan_uids.append(raw_uid)
            row.bg_color = (0.6, 0.8, 1, 1)

    def route_adjust_popup(self):
        if self.mode == "IN": self.open_trans_adjust_popup()
        elif self.mode == "OUT": self.open_out_adjust_popup()

    def open_trans_adjust_popup(self):
        if not getattr(self, 'selected_scan_uids', []): return
        if not hasattr(self, 'trans_adj_popup') or not self.trans_adj_popup:
            self.trans_adj_popup = Factory.TransAdjustPopup()

        self.trans_adj_popup.ids.popup_scroll_view.scroll_y = 1

        uids = list(self.selected_scan_uids)
        self.trans_adj_popup.ids.pop_trans_adj_info.text = f"Adjusting {len(uids)}
Item(s)"

        first_uid = uids[0]
        cache = getattr(self, 'local_item_cache', {})
        data = cache.get(first_uid, {})

        self.trans_adj_popup.ids.pop_mat.text = str(data.get("MATERIAL") or "")
        self.trans_adj_popup.ids.pop_desc.text = str(data.get("DESCRIPTION") or "")
        qty_val = data.get("QTY")
        self.trans_adj_popup.ids.pop_qty.text = str(qty_val) if qty_val and str(qty_val)
!= "0" else ""
        self.trans_adj_popup.ids.pop_batch.text = str(data.get("BATCH") or "")

        sloc = data.get("SLOC")
        self.trans_adj_popup.ids.pop_sloc.text = str(sloc) if sloc else "Select SLoc"
        self.trans_adj_popup.ids.pop_sbin.text = str(data.get("SBIN") or "")

        self.trans_adj_popup.open()
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
    def submit_trans_adjust(self):
        uids = list(self.selected_scan_uids)
        if not uids: return

        p_mat = self.trans_adj_popup.ids.pop_mat.text.strip().upper()
        p_desc = self.trans_adj_popup.ids.pop_desc.text.strip().upper()
        p_qty = self.trans_adj_popup.ids.pop_qty.text.strip()
        p_batch = getattr(self.trans_adj_popup.ids, 'pop_batch', None)
        p_batch_val = p_batch.text.strip().upper() if p_batch else ""
        p_sloc = self.trans_adj_popup.ids.pop_sloc.text.strip().upper()

        p_sbin_raw = self.trans_adj_popup.ids.pop_sbin.text.strip().upper()
        p_sbin = p_sbin_raw

        if p_sloc == "TR":
            if p_sbin_raw.isdigit():
                p_sbin = f"TR{int(p_sbin_raw):04d}"
            elif p_sbin_raw.startswith("TR") and p_sbin_raw[2:].isdigit():
                p_sbin = f"TR{int(p_sbin_raw[2:]):04d}"

        status_baru = "READY"
        if p_sloc == "TR" or p_sbin.startswith("TR"): status_baru = "TRANSIT"
        elif not all([p_mat, p_desc, p_qty, p_sloc, p_sbin]) or p_qty == "0" or p_sloc ==
"SELECT SLOC": status_baru = "INCOMPLETE"

        self.pending_adjust_data = {
            'type': 'trans', 'uids': uids, 'mat': p_mat, 'desc': p_desc,
            'qty': p_qty, 'batch': p_batch_val, 'sloc': p_sloc, 'sbin': p_sbin, 'status':
status_baru
        }

        if not hasattr(self, 'confirm_loc_popup') or not self.confirm_loc_popup:
            self.confirm_loc_popup = Factory.ConfirmLocPopup()

        self.confirm_loc_popup.ids.lbl_conf_count.text = f"Adjusting {len(uids)} Item(s)"
        self.confirm_loc_popup.ids.lbl_conf_mat.text = p_mat if p_mat else "UNKNOWN MAT"
        self.confirm_loc_popup.ids.lbl_conf_loc.text = f"{p_sloc} - {p_sbin}"
        self.trans_adj_popup.dismiss()
        self.confirm_loc_popup.open()

    def open_out_adjust_popup(self):
        if not getattr(self, 'selected_scan_uids', []): return
        uid = self.selected_scan_uids[0]

        cache = getattr(self, 'local_item_cache', {}).get(uid, {})
        mat_name = cache.get("MATERIAL", "UNKNOWN MATERIAL")
        current_out_qty = cache.get("QTY", 0)

        if not hasattr(self, 'out_adj_popup') or not self.out_adj_popup:
            self.out_adj_popup = Factory.OutAdjustPopup()

        def _fetch_real_stock():
            try:
                res = supabase.table("STOCK").select("QTY").eq("RFID_UID", uid).execute()
                stock_qty = int(res.data[0]['QTY']) if res.data else 0
            except:
                stock_qty = int(current_out_qty) if current_out_qty else 0

            Clock.schedule_once(lambda dt: self._show_out_popup(mat_name, stock_qty,
current_out_qty, cache))

        threading.Thread(target=_fetch_real_stock, daemon=True).start()

    def submit_out_adjust(self):
        uids = list(self.selected_scan_uids)
        if not uids: return
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        try: p_qty = int(self.out_adj_popup.ids.pop_out_qty.text.strip())
        except ValueError: p_qty = 0

        if p_qty <= 0: return

        p_sloc = self.out_adj_popup.ids.pop_out_sloc.text.strip().upper()
        p_sbin_raw = self.out_adj_popup.ids.pop_out_sbin.text.strip().upper()
        p_sbin = p_sbin_raw

        if p_sloc == "TR":
            if p_sbin_raw.isdigit():
                p_sbin = f"TR{int(p_sbin_raw):04d}"
            elif p_sbin_raw.startswith("TR") and p_sbin_raw[2:].isdigit():
                p_sbin = f"TR{int(p_sbin_raw[2:]):04d}"

        first_uid = uids[0]
        original_stock = getattr(self, 'current_out_stock_qty', 0)
        cache = getattr(self, 'local_item_cache', {}).get(first_uid, {})

        p_mat = cache.get("MATERIAL", "UNKNOWN")

        if p_qty < original_stock:
            status_baru = "PARTIAL"
            loc_text = f"MOTONG STOK DI {p_sloc}-{p_sbin}"
        else:
            status_baru = "FULL"
            loc_text = f"FULL PICK DARI {p_sloc}-{p_sbin}"

        self.pending_adjust_data = {
            'type': 'out', 'uids': uids, 'mat': p_mat,
            'qty': str(p_qty), 'status': status_baru,
            'sloc': p_sloc, 'sbin': p_sbin
        }

        if not hasattr(self, 'confirm_loc_popup') or not self.confirm_loc_popup:
            self.confirm_loc_popup = Factory.ConfirmLocPopup()

        self.confirm_loc_popup.ids.lbl_conf_count.text = f"Picking {len(uids)} Item(s)"
        self.confirm_loc_popup.ids.lbl_conf_mat.text = p_mat
        self.confirm_loc_popup.ids.lbl_conf_loc.text = loc_text
        self.confirm_loc_popup.open()

    def _show_out_popup(self, mat, stock_qty, current_qty, cache):
        self.current_out_stock_qty = stock_qty
        self.out_adj_popup.ids.pop_out_info.text = f"Mat: {mat}"
        self.out_adj_popup.ids.pop_out_stock.text = f"Colly Stock: {stock_qty}"
        self.out_adj_popup.ids.pop_out_qty.text = str(current_qty) if str(current_qty) !=
"0" else str(stock_qty)

        sloc = cache.get("SLOC")
        self.out_adj_popup.ids.pop_out_sloc.text = sloc if sloc else "Select SLoc"
        self.out_adj_popup.ids.pop_out_sbin.text = cache.get("SBIN") or ""

        self.out_adj_popup.open()

    def _thread_delete_supabase(self, uids):
        supabase.table(f"SCAN_{self.mode}").delete().eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", uids).execute()

    def _thread_load_bins(self):
        try:
            Clock.schedule_once(lambda dt: self._clear_bin_search())
            res = supabase.table("STOCK").select("SLOC, SBIN").execute()
            if not res.data: return
            bin_counts = {}
            for item in res.data:
                sloc = str(item.get("SLOC", "")).strip().upper()
                sbin = str(item.get("SBIN", "")).strip().upper()
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                if not sloc or not sbin or len(sbin) < 2: continue

                prefix = sbin[:2]

                key = f"{sloc}|{prefix}"
                if key not in bin_counts: bin_counts[key] = {"sloc": sloc, "sbin": prefix,
"qty": 0}
                bin_counts[key]["qty"] += 1

            self.full_bin_list = sorted(list(bin_counts.values()), key=lambda x:
(x['sloc'], x['sbin']))
            Clock.schedule_once(lambda dt: self._populate_bin_list(self.full_bin_list))
        except Exception as e: print(f"Failed to load bins: {e}")

    def _clear_bin_search(self):
        self.root.get_screen('so_sel_screen').ids.search_bin.text = ""

    def _populate_bin_list(self, data_list):
        content = self.root.get_screen('so_sel_screen').ids.bin_list_content
        content.clear_widgets()

        active_bins = getattr(self, 'so_active_bins', set())
        pinned_items = [item for item in self.full_bin_list if item['sbin'] in
active_bins]
        pinned_sbins = [item['sbin'] for item in pinned_items]

        other_items = [item for item in data_list if item['sbin'] not in pinned_sbins]

        final_list = pinned_items + other_items

        for item in final_list:
            row = Factory.SOBinRow()
            row.ids.lbl_sloc.text = item['sloc']
            row.ids.lbl_sbin.text = item['sbin']
            row.ids.lbl_qty.text = str(item['qty'])
            if item['sbin'] in active_bins:
                self._set_row_selected_ui(row, True)
            else:
                self._set_row_selected_ui(row, False)
            row.bind(on_release=lambda instance, s=item['sbin']:
self.toggle_bin_selection(s, instance))
            content.add_widget(row)

    def toggle_bin_selection(self, sbin, row_widget):
        if sbin in self.so_active_bins:
            self.so_active_bins.remove(sbin)
        else:
            self.so_active_bins.add(sbin)

        search_text = self.root.get_screen('so_sel_screen').ids.search_bin.text
        self.filter_so_bins(search_text)

    def _set_row_selected_ui(self, row, is_selected):
        if is_selected:
            row.bg_color = (0.2, 0.4, 0.8, 1)
            row.text_color_normal = (1, 1, 1, 1)
            row.text_color_highlight = (1, 1, 1, 1)
        else:
            row.bg_color = (0.95, 0.95, 0.95, 1)
            row.text_color_normal = (0, 0, 0, 1)
            row.text_color_highlight = (0, 0.3, 0.7, 1)

    def filter_so_bins(self, query):
        query = query.upper()
        filtered_data = [item for item in self.full_bin_list if query in item['sloc'] or
query in item['sbin']]
        self._populate_bin_list(filtered_data)
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
    def start_so_opname(self):
        if not getattr(self, 'so_active_bins', set()): return
        self.root.current = 'so_detail_screen'
        active_list = list(self.so_active_bins)
        bins_text = ', '.join(active_list)
        display_text = bins_text if len(bins_text) <= 20 else bins_text[:20] + "..."
        self.root.get_screen('so_detail_screen').ids.lbl_so_bins.text = f"BINS:
{display_text}"
        self.so_item_custom_status = {}
        threading.Thread(target=self._thread_load_so_expected, daemon=True).start()

    def _thread_load_so_expected(self):
        try:
            self.so_expected_items = {}
            for pfx in list(self.so_active_bins):
                res = supabase.table("STOCK").select("*").ilike("SBIN",
f"{pfx}%").execute()
                if res.data:
                    for row in res.data:
                        self.so_expected_items[row['RFID_UID']] = row

            self.so_found_items = []
            self.so_extra_items = []
            self.so_row_widgets = {}

            Clock.schedule_once(lambda dt: self._init_so_widgets())
        except Exception as e: print(f"Failed to load expected SO: {e}")

    def _init_so_widgets(self):
        for uid, data in self.so_expected_items.items():
            row = Factory.ScanTableRow()
            row.rfid_tag = str(uid)
            row.search_string = f"{uid} {data.get('MATERIAL','')}
{data.get('DESCRIPTION','')} {data.get('SLOC','')} {data.get('SBIN','')}".lower()

            mat = data.get('MATERIAL', '-')
            sloc = data.get('SLOC', '-')
            sbin = data.get('SBIN', '-')
            batch = data.get('BATCH', '-')
            qty = str(data.get('QTY', '0'))

            row.ids.lbl_uid.text = str(mat) if mat and mat != "-" else "UNKNOWN"
            row.ids.lbl_details.text = f"{sloc} {sbin} | Batch: {batch} | Qty: {qty}"
            row.bind(on_release=lambda instance, u=uid: self.toggle_scan_row(u, instance))

            self.so_row_widgets[uid] = row

        self._update_so_ui_states()

    def process_so_scan(self, raw_input):
        uids = [u.strip().upper() for u in raw_input.split() if u.strip()]
        new_extra_added = False
        for uid in uids:
            if not self.is_valid_epc(uid) and uid not in self.so_expected_items:
                continue

            if uid in self.so_found_items or uid in self.so_extra_items: continue

            if uid in self.so_expected_items:
                self.so_found_items.append(uid)
            else:
                self.so_extra_items.append(uid)
                new_extra_added = True

            threading.Thread(target=self._thread_push_so_scan, args=(uid,),
daemon=True).start()

        if new_extra_added:
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
            self._create_missing_extra_widgets()

        self._update_so_ui_states()

    def _create_missing_extra_widgets(self):
        for uid in self.so_extra_items:
            if uid not in self.so_row_widgets:
                row = Factory.ScanTableRow()
                row.rfid_tag = str(uid)
                row.search_string = uid.lower()
                row.ids.lbl_uid.text = "UNKNOWN"
                row.ids.lbl_details.text = "- - | Batch: - | Qty: -"
                row.bind(on_release=lambda instance, u=uid: self.toggle_scan_row(u,
instance))
                self.so_row_widgets[uid] = row

    def _update_so_ui_states(self):
        expected_count = len(self.so_expected_items)
        found_count = len(self.so_found_items)
        bins_text = ', '.join(list(self.so_active_bins))
        disp_bins = bins_text if len(bins_text) <= 12 else bins_text[:12] + ".."
        self.root.get_screen('so_detail_screen').ids.lbl_so_bins.text = f"PROG:
{found_count}/{expected_count} | {disp_bins}"

        for uid, row in self.so_row_widgets.items():
            custom_stat = getattr(self, 'so_item_custom_status', {}).get(uid, None)

            if uid in self.selected_scan_uids: row.bg_color = (0.6, 0.8, 1, 1)
            else: row.bg_color = (0.95, 0.95, 0.95, 1)

            if custom_stat:
                row.ids.lbl_status.text = custom_stat
                row.status_color = (0.8, 0, 0, 1) if custom_stat == "REPORTED" else (0.5,
0, 0.5, 1)
            elif uid in self.so_extra_items:
                row.uid_color = (0.8, 0.4, 0, 1); row.status_color = (0.8, 0.4, 0, 1)
                row.ids.lbl_status.text = "EXTRA"
            elif uid in self.so_found_items:
                row.ids.lbl_status.text = "FOUND"
                row.status_color = (0.1, 0.6, 0.1, 1)
            else:
                row.ids.lbl_status.text = "WAITING"
                row.status_color = (0.8, 0, 0, 1)

        self.apply_so_filter()

    def open_so_filter_popup(self):
        if not hasattr(self, 'so_filter_popup') or not self.so_filter_popup:
            self.so_filter_popup = Factory.SOFilterPopup()
        self.so_filter_popup.open()

    def apply_so_filter(self, *args):
        screen = self.root.get_screen('so_detail_screen')
        query = screen.ids.so_detail_search.text.lower()
        content = screen.ids.so_list_content

        if not hasattr(self, 'so_filter_popup') or not self.so_filter_popup:
            s_fnd = True; s_wait = True; s_rep = True; s_ext = False
        else:
            s_fnd = self.so_filter_popup.ids.chk_found.active
            s_wait = self.so_filter_popup.ids.chk_waiting.active
            s_rep = self.so_filter_popup.ids.chk_reported.active
            s_ext = self.so_filter_popup.ids.chk_extra.active

        active_count = sum([s_fnd, s_wait, s_rep, s_ext])
        screen.ids.btn_so_filter.text = f"FILTER ({active_count})"

        content.clear_widgets()
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
        if not hasattr(self, 'so_row_widgets'): return

        for uid, row in self.so_row_widgets.items():
            stat = row.ids.lbl_status.text

            if stat == "FOUND" and not s_fnd: continue
            if stat == "WAITING" and not s_wait: continue
            if stat == "EXTRA" and not s_ext: continue
            if stat in ["REPORTED", "ADJUSTED"] and not s_rep: continue

            if query and query not in row.search_string: continue

            content.add_widget(row)

    def _thread_push_so_scan(self, uid):
        try:
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            data = {"RFID_UID": uid, "PC_TOKEN": self.pc_token, "TIMESTAMP":
timestamp_now}
            supabase.table("SCAN_SO").insert(data).execute()
        except: pass

    def open_bulk_report_popup(self):
        self.target_report_uids = list(self.selected_scan_uids)

        if not hasattr(self, 'report_popup') or not self.report_popup:
            self.report_popup = Factory.ReportBulkPopup()

        if self.target_report_uids:
            self.report_popup.ids.pop_report_info.text = f"Reporting
{len(self.target_report_uids)} Item(s)"
            self.report_popup.ids.pop_issue.values = ['Damaged', 'Missing Item', 'Wrong
Label', 'Other']
        else:
            self.report_popup.ids.pop_report_info.text = "GENERAL PROBLEM"
            self.report_popup.ids.pop_issue.values = ['Scanner Error', 'Network Issue',
'Facility Damage', 'Other']

        self.report_popup.ids.pop_issue.text = 'Select Issue...'
        self.report_popup.ids.pop_note.text = ''
        self.report_popup.open()

    def submit_bulk_report(self, issue_type, note):
        if issue_type == 'Select Issue...':
            return

        uids_to_report = list(self.target_report_uids)
        if not uids_to_report:
            uids_to_report = ["GENERAL_PROBLEM"]

        threading.Thread(
            target=self._thread_submit_bulk_report,
            args=(uids_to_report, issue_type, note),
            daemon=True
        ).start()

    def _thread_submit_bulk_report(self, uids, issue_type, note):
        try:
            target_table = f"REPORT_{self.mode}"
            date_str = datetime.now().strftime("%y%m%d%H%M")
            report_code = f"{self.mode}-{self.pc_token[-3:]}-{date_str}"
            timestamp_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            payload = []

            if self.mode == "SO":
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                res_stock = supabase.table("STOCK").select("RFID_UID").in_("RFID_UID",
uids).execute()
                valid_rfids = [r['RFID_UID'] for r in res_stock.data] if res_stock.data
else []

                for uid in uids:
                    if uid == "GENERAL_PROBLEM":
                        item_data = {
                            "REPORT_CODE": report_code, "RFID_UID": uid,
                            "REPORT": issue_type, "NOTE": note, "TIMESTAMP":
timestamp_now,
                            "MATERIAL": "-", "DESCRIPTION": "-"
                        }
                        payload.append(item_data)
                    elif uid in valid_rfids:
                        item_data = {
                            "REPORT_CODE": report_code, "RFID_UID": uid,
                            "REPORT": issue_type, "NOTE": note, "TIMESTAMP":
timestamp_now,
                            "MATERIAL": self.so_expected_items.get(uid,
{}).get("MATERIAL", ""),
                            "DESCRIPTION": self.so_expected_items.get(uid,
{}).get("DESCRIPTION", "")
                        }
                        payload.append(item_data)

            else:
                sumber_tabel = "STOCK" if self.mode == "OUT" else "MASTER_DATA"
                res_master = supabase.table(sumber_tabel).select("RFID_UID, MATERIAL,
DESCRIPTION").in_("RFID_UID", uids).execute()

                master_dict = {r['RFID_UID']: r for r in res_master.data} if
res_master.data else {}

                for uid in uids:
                    if uid == "GENERAL_PROBLEM":
                        mat_val = "-"
                        desc_val = "-"
                    else:
                        mat_val = master_dict.get(uid, {}).get("MATERIAL", "UNKNOWN")
                        desc_val = master_dict.get(uid, {}).get("DESCRIPTION", "UNKNOWN")

                    item_data = {
                        "REPORT_CODE": report_code, "RFID_UID": uid,
                        "REPORT": issue_type, "NOTE": note, "TIMESTAMP": timestamp_now,
                        "MATERIAL": mat_val,
                        "DESCRIPTION": desc_val
                    }
                    payload.append(item_data)

            if payload:
                supabase.table(target_table).insert(payload).execute()

                if self.mode != "SO" and uids != ["GENERAL_PROBLEM"]:
                    try:
                        supabase.table(f"SCAN_{self.mode}").update({"STATUS":
"REPORTED"}).eq("PC_TOKEN", self.pc_token).in_("RFID_UID", uids).execute()
                    except Exception as e:
                        print(f"Gagal update status SCAN: {e}")

                if self.mode == "SO":
                    if not hasattr(self, 'so_item_custom_status'):
self.so_item_custom_status = {}
                    for uid in uids:
                        if uid != "GENERAL_PROBLEM":
                            self.so_item_custom_status[uid] = "REPORTED"
                    Clock.schedule_once(lambda dt: self._rebuild_so_ui())
                else:
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    Clock.schedule_once(lambda dt: self._mark_ui_custom_status(uids,
"REPORTED"))
            else:
                print("Laporan dibatalkan karena tidak ada data payload yang valid.")

        except Exception as e:
            print(f"FATAL REPORT ERROR: {e}")

    def execute_adjust_final(self):
        data = getattr(self, 'pending_adjust_data', None)
        if not data: return

        uids = data['uids']

        if data['type'] == 'trans':
            def _push_db():
                try:
                    try: qty_int = int(data['qty'])
                    except ValueError: qty_int = 0

                    update_data = {
                        "MATERIAL": data['mat'], "DESCRIPTION": data['desc'],
                        "QTY": qty_int, "BATCH": data['batch'], "SLOC": data['sloc'],
                        "SBIN": data['sbin'], "STATUS": data['status']
                    }

                    try:
                        supabase.table(f"SCAN_{self.mode}").update(update_data).eq("PC_TOK
EN", self.pc_token).in_("RFID_UID", uids).execute()
                    except Exception as e:
                        update_data = {
                            "MATERIAL": data['mat'], "QTY": qty_int,
                            "SLOC": data['sloc'], "SBIN": data['sbin'], "STATUS":
data['status']
                        }
                        supabase.table(f"SCAN_{self.mode}").update(update_data).eq("PC_TOK
EN", self.pc_token).in_("RFID_UID", uids).execute()

                    if not hasattr(self, 'local_item_cache'): self.local_item_cache = {}
                    for u in uids: self.local_item_cache[u] = update_data
                    Clock.schedule_once(lambda dt: self._mark_ui_custom_status(uids,
data['status'], update_data))

                except Exception as e:
                    Clock.schedule_once(lambda dt: self._mark_ui_custom_status(uids, "ERR
DB"))

            threading.Thread(target=_push_db, daemon=True).start()

        elif data['type'] == 'out':
            def _push_db():
                try:
                    qty_int = int(data['qty'])
                    update_data = {
                        "QTY": qty_int, "STATUS": data['status'],
                        "SLOC": data['sloc'], "SBIN": data['sbin']
                    }

                    supabase.table(f"SCAN_{self.mode}").update(update_data).eq("PC_TOKEN",
self.pc_token).in_("RFID_UID", uids).execute()

                    if not hasattr(self, 'local_item_cache'): self.local_item_cache = {}
                    for u in uids:
                        if u in self.local_item_cache:
self.local_item_cache[u].update(update_data)
                        else: self.local_item_cache[u] = update_data
```

Lampiran 12 Program Koding Sistem Informasi *Scanner*

```
                    Clock.schedule_once(lambda dt: self._mark_ui_custom_status(uids,
data['status'], update_data))
                except Exception as e:
                    Clock.schedule_once(lambda dt: self._mark_ui_custom_status(uids, "ERR
DB"))
            threading.Thread(target=_push_db, daemon=True).start()

    def _mark_ui_custom_status(self, uids, new_status, update_data=None):
        content = self.root.get_screen('scanner_screen').ids.scan_list_content
        for row in content.children:
            raw_uid = getattr(row, 'rfid_tag',
row.ids.lbl_uid.text.split('\n')[0].strip())

            if raw_uid in uids:
                row.ids.lbl_status.text = new_status
                row.uid_color = (0, 0, 0, 1)

                if new_status == "READY": row.status_color = (0.1, 0.6, 0.1, 1)
                elif new_status == "REPORTED": row.status_color = (0.8, 0, 0, 1)
                elif new_status == "ADJUSTED": row.status_color = (1, 0.6, 0, 1)
                elif new_status == "TRANSIT": row.status_color = (0.8, 0.6, 0, 1)
                elif new_status == "INCOMPLETE": row.status_color = (0.5, 0.5, 0.5, 1)
                elif new_status == "PARTIAL": row.status_color = (1, 0.5, 0, 1)
                elif new_status == "FULL": row.status_color = (0.1, 0.6, 0.1, 1)
                elif "ERR" in new_status: row.status_color = (0.8, 0, 0, 1)
                else: row.status_color = (0, 0, 0, 1)

                if update_data:
                    mat = update_data.get("MATERIAL") or "-"
                    sloc = update_data.get("SLOC") or "-"
                    sbin = update_data.get("SBIN") or "-"
                    batch = update_data.get("BATCH") or "-"
                    qty = update_data.get("QTY") or "0"

                    if self.mode == "OUT":
                        row.ids.lbl_uid.text = str(mat) if mat and mat != "-" else
"UNKNOWN MAT"
                        row.ids.lbl_details.text = f"{sloc} {sbin} | Batch: {batch} | Qty:
{qty}"
                    else:
                        row.ids.lbl_details.text = f"Mat: {mat} | Batch: {batch} | {sloc}
{sbin} | Qty: {qty}"

        self.selected_scan_uids.clear()
        for r in content.children: r.bg_color = (0.95, 0.95, 0.95, 1)

    def open_finish_so_confirm(self):
        if not hasattr(self, 'finish_so_popup') or not self.finish_so_popup:
            self.finish_so_popup = Factory.FinishSOPopup()
        self.finish_so_popup.open()

    def execute_finish_so(self):
        threading.Thread(target=self._thread_finish_so, daemon=True).start()
        self.return_to_menu()

    def _thread_finish_so(self):
        try:
            supabase.table("SCAN_SO").delete().eq("PC_TOKEN", self.pc_token).execute()
        except: pass

if __name__ == '__main__':
    ScannerApp().run()
```