BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

2.1.1. Evolusi dan Keterbatasan Sistem Penggajian Konvensional

Perkembangan teknologi pemrosesan penggajian telah bertransformasi secara signifikan dari pencatatan manual menuju sistem otomatis berbasis komputasi awan. Meenugu pada tahun 2025 menjelaskan bahwa evolusi ini berhasil meningkatkan efisiensi dan mengurangi intervensi manual dalam manajemen sumber daya manusia. Namun, meskipun telah terotomatisasi, infrastruktur tradisional ini masih bertumpu pada arsitektur terpusat (Web2) yang memiliki latensi laten dalam pemrosesan transaksi [1].

Persoalan pada arsitektur terpusat tidak hanya terbatas pada kekakuan siklus waktu (bulanan atau bi-mingguan), tetapi juga pada aspek keamanan data. Faruk dan Khan menyoroti bahwa sistem *Human Resource Information System* (HRIS) dan *payroll* terpusat sangat rentan terhadap manipulasi internal (*insider threats*) dan penipuan data (*payroll fraud*), seperti penciptaan pekerja fiktif (*ghost workers*). Dalam studinya, mereka menemukan bahwa integrasi sistem cerdas memerlukan lapisan keamanan tambahan yang tidak dapat diubah (*immutable*) untuk melindungi data sensitif tenaga kerja [3]. Keterbatasan struktural inilah yang mendorong kebutuhan akan transisi menuju sistem yang lebih terdesentralisasi.

2.1.2. Integrasi *Blockchain* dan *Smart contract* pada Manajemen Kompensasi

Sebagai solusi atas kerentanan basis data terpusat, teknologi *blockchain* mulai diimplementasikan dalam arsitektur penggajian. Ahmed dan Shakir (2023) membuktikan bahwa pemanfaatan *blockchain* dalam sistem *payroll* mampu memberikan transparansi mutlak dan menghilangkan *Single Point of Failure* (SPOF) yang sering terjadi pada basis data relasional [4]. Lebih jauh, Verma (2025) mengeksplorasi penggunaan *smart contract* untuk mengotomatisasi kompensasi

melalui integrasi dengan platform manajemen sumber daya manusia komersial seperti Workday. Penelitian tersebut menunjukkan bahwa *smart contract* dapat mengubah aturan kebijakan kompensasi perusahaan menjadi eksekusi deterministik yang berjalan secara mandiri tanpa perantara [6]. Meskipun demikian, Lamberty dkk. mencatat bahwa dalam industri modern (Industri 4.0), penerapan pembayaran terprogram (*programmable payments*) sering kali terhambat oleh masalah interoperabilitas dan tingginya friksi pengalaman pengguna saat berinteraksi dengan jaringan *blockchain* publik [5].

2.1.3. *Stablecoin* dan *Streaming payment* sebagai Solusi Likuiditas Pekerja

Keterlambatan penerimaan upah pada sistem konvensional menciptakan fenomena *payday effect*, di mana pekerja sering kali kehabisan likuiditas sebelum siklus gajian berikutnya tiba. Parker dan Parker menegaskan bahwa konvergensi infrastruktur *blockchain* dan *stablecoin* menghadirkan paradigma baru berupa kompensasi waktu nyata (*real-time compensation*) atau *streaming payment*. Transaksi menggunakan *stablecoin* memungkinkan pergerakan nilai dengan biaya rendah dan berkecepatan tinggi, yang berpotensi membebaskan pekerja dari ketergantungan pada fasilitas pinjaman berbunga tinggi [2].

Dukungan terhadap solusi ini juga diperkuat oleh analisis Liao dan Caramichael (2022), yang menyoroti potensi pertumbuhan *stablecoin* sebagai terobosan dalam masa depan sistem pembayaran yang stabil dan tidak terekspos pada volatilitas harga aset kripto konvensional [7]. Selain itu, Aquilina dkk. (2025) menambahkan bahwa ekosistem *Decentralized Finance* (DeFi) terbukti mampu mereplikasi fungsi intermediasi keuangan tradisional. Hal ini membuka jalan bagi implementasi *Earned Wage Access* (EWA) secara terdesentralisasi, di mana pekerja dapat menarik upah yang telah menjadi haknya secara instan dari dana yang dikelola *smart contract* tanpa harus menunggu persetujuan bank perantara [8].

2.2. Celah Penelitian

Berdasarkan tinjauan pustaka di atas, dapat diidentifikasi beberapa celah

penelitian (*research gap*) sebagai berikut:

1. **Pada aspek arsitektur sistem:**

   Meskipun sistem penggajian berbasis *cloud* (Web2) telah membawa peningkatan signifikan dalam efisiensi dan otomasi proses penggajian [1], infrastruktur ini masih gagal memitigasi ancaman manipulasi internal (*ghost workers*) secara mutlak karena sifatnya yang masih terpusat dan dapat diubah oleh administrator sistem. Faruk dan Khan [3] secara eksplisit menyatakan bahwa sistem penggajian terpusat tetap rentan karena seorang administrator tunggal dapat mengubah data upah tanpa terdeteksi secara langsung, dan bahwa penerapan *blockchain* terbukti dapat mengeliminasi *ghost workers* serta meningkatkan akuntabilitas tata kelola organisasi.

2. **Pada aspek fungsionalitas kompensasi:**

   Integrasi *smart contract* pada manajemen kompensasi, seperti yang diteliti oleh Verma [6] dalam konteks integrasi *blockchain* dengan platform Workday, berfokus pada otomasi kebijakan perusahaan seperti bonus berbasis kinerja dan *vesting* saham. Demikian pula, Parker dan Parker [2] menguraikan potensi *blockchain* dan *stablecoin* dalam mempercepat siklus pembayaran menuju model kompensasi *real-time*. Namun, kedua penelitian tersebut belum mengintegrasikan mekanisme *streaming payment* secara langsung sebagai solusi teknis untuk menyelesaikan masalah likuiditas pekerja (*Earned Wage Access*/EWA), di mana karyawan dapat mengakses upah yang telah diperoleh kapan saja secara kontinu tanpa harus menunggu siklus penggajian.

3. **Pada aspek interaksi dan pengalaman pengguna (UX):**

   Sebagian besar literatur yang mengusulkan *blockchain* untuk penggajian belum menyelesaikan hambatan teknis terbesar bagi adopsi arus utama. Ahmed dan Shakir [4] mengidentifikasi bahwa biaya adopsi *blockchain* yang terus meningkat, termasuk kewajiban membayar biaya transaksi komputasi (*gas fee*), merupakan salah satu kendala paling signifikan yang menghambat penggunaan massal teknologi ini. Pada sisi pengalaman pengguna, Aquilina et al. [8] turut menegaskan bahwa

ekosistem terdesentralisasi menghadapi tantangan serius berupa *user experience* yang kompleks. Hambatan ganda ini, biaya gas dan kerumitan teknis validasi identitas kepegawaian digital di jaringan publik, belum dijawab secara terpadu oleh literatur yang ada.

Celah penelitian inilah yang menjadi landasan bagi penulis untuk melakukan rancang bangun sistem *payroll* terdesentralisasi dengan mekanisme *streaming payment*. Penelitian ini membedakan diri dari literatur terdahulu dengan memperkenalkan integrasi *Account abstraction* (ERC-4337) untuk memungkinkan transaksi tanpa biaya (*gasless*) bagi pekerja, serta penggunaan *Soulbound Token* (ERC-5192) sebagai mekanisme identitas kepegawaian digital yang tidak dapat dipindahtangankan. Perbandingan posisi penelitian ini dengan penelitian-penelitian sebelumnya dapat dilihat pada Tabel 2.1.

Tabel 2.1. Tabel Perbandingan Penelitian Terdahulu (*State-of-the-Art*)

Penulis (Tahun)	**Fokus Penelitian / Domain**	**Penerapan *Smart contract* & Desentralisasi**	**Dukungan *Streaming payment* & *Stablecoin***	**Integrasi Validasi Identitas (SBT) & *Gasless Transaction* (AA)**
Meenugu (2025)	Evolusi teknis sistem *payroll* dari manual ke otomatisasi.	Tidak ada (Fokus pada komputasi awan Web2 tradisional).	Tidak ada.	Tidak ada.
Ahmed & Shakir (2023)	Pemanfaatan *blockchain* untuk keamanan data *payroll* & SDM.	Ada (Sebagai buku besar terdistribusi untuk integritas data).	Tidak ada (Hanya audit rekam jejak, bukan transaksi aset).	Tidak ada.
Verma (2025)	Otomatisasi kompensasi melalui *blockchain* terintegrasi Workday.	Ada (Otomatisasi aturan bonus dan gaji).	Tidak ada (Masih berdasar siklus pencairan tradisional).	Tidak ada.
Parker & Parker (2025)	Konsep teoritis kompensasi *real-time* berbasis *blockchain*.	Ada (Secara konseptual).	Ada (Menggunakan *stablecoin* untuk *real-time pay*).	Tidak ada.
Superfluid (2021)	Protokol *money streaming* generik berbasis *token*, tanpa konteks bisnis spesifik.	Ada (primitif transfer kontinu *on-chain* sebagai infrastruktur umum, dapat diadopsi lintas use-case apa pun).	Ada (mendukung streaming *stablecoin* ERC-20 apa pun secara generik).	Tidak ada (tidak menyediakan mesin pajak, kasbon, *governance* PHK, maupun identitas

				kepegawaian).
Sablier (2022)	Protokol *vesting* dan *streaming payment* untuk distribusi *token* linear atau *cliff*.	Ada (primitif *vesting*/*stream* generik, umum dipakai untuk distribusi *token* proyek atau gaji tim internal Web3).	Ada (mendukung *cliff vesting* dan linear *stream*).	Tidak ada (tidak menyediakan kepatuhan pajak Indonesia, kasbon, maupun sertifikasi ketenagakerjaan).
Penulis (2026)	Rancang bangun sistem *payroll* dengan likuiditas instan (EWA).	Ada (Solidity, *Vault, Streaming payment* yang terotomatisasi secara *trustless*).	Ada (Menggunakan arsitektur *streaming payment* dengan *token* ERC-20 *stablecoin*).	Ada (Menggunakan *Soulbound Token* ERC-5192 dan transaksi *Gasless* ERC-4337).

Superfluid dan Sablier adalah dua protokol *streaming payment* yang telah beroperasi secara nyata di industri Web3, bukan sekadar konsep akademik. Superfluid, diluncurkan pada 2021, menyediakan primitif *Super Token* yang memungkinkan saldo penerima bertambah secara kontinu setiap detik berdasarkan *flow rate* yang ditetapkan pengirim, tanpa memerlukan transaksi berulang untuk setiap pencairan [9]. Sablier, diluncurkan pada 2022, berfokus pada distribusi *token* linear atau bertahap (*cliff vesting*) yang umum dipakai untuk *vesting* saham karyawan maupun distribusi *token* proyek Web3 [10]. Keduanya bersifat generik dan agnostik terhadap konteks bisnis: menyediakan infrastruktur dasar untuk mengalirkan nilai dari satu alamat ke alamat lain, tetapi tidak menyediakan logika bisnis spesifik seperti perhitungan pajak penghasilan, mekanisme persetujuan pemutusan hubungan kerja, kasbon, maupun identitas kepegawaian. Ini menjadi celah yang menjadi dasar kontribusi penelitian ini pada lapisan aplikasi.

Tabel 2.1 menunjukkan bahwa penelitian akademik terdahulu mengenai *payroll* berbasis *blockchain* umumnya bersifat konseptual atau berfokus pada aspek parsial (audit trail, otomasi bonus), tanpa integrasi penuh antara *streaming payment*, kepatuhan pajak, dan identitas digital. Perbandingan juga perlu mencakup protokol *streaming payment* yang telah beroperasi secara nyata di industri, yaitu Superfluid [9] dan Sablier [10], karena keduanya adalah rujukan paling relevan dari sisi mekanisme teknis streaming yang diadopsi penelitian ini. Superfluid dan Sablier merupakan primitif (*primitive*) transfer nilai kontinu yang bersifat generik dan agnostik terhadap konteks bisnis: keduanya menyediakan infrastruktur dasar untuk mengalirkan *token* dari satu alamat ke alamat lain secara linear atau bertahap, namun tidak menyediakan logika bisnis spesifik seperti perhitungan pajak penghasilan, mekanisme persetujuan pemutusan hubungan kerja, kasbon, maupun sertifikasi ketenagakerjaan. Penelitian ini tidak mengklaim kontribusi pada level primitif kriptografis atau protokol streaming itu sendiri, karena mekanisme akrual linear yang diimplementasikan pada fungsi startStream() dan claimSalary() secara konseptual sejalan dengan pendekatan yang telah mapan pada Superfluid dan Sablier. Kontribusi penelitian ini terletak pada lapisan aplikasi (application *layer*)

di atas primitif tersebut, yaitu integrasi *streaming payment* dengan *governance* ketenagakerjaan (persetujuan pemutusan hubungan kerja dua pihak), kepatuhan regulasi Indonesia (perhitungan PPh21 skema Tarif Efektif Rata-rata sesuai PMK 168/2023 dan BPJS), fasilitas kasbon terintegrasi, serta identitas kepegawaian berbasis *Soulbound Token*, yang secara spesifik menjawab kebutuhan penggajian pada segmen SME dan *startup* dengan struktur kompensasi non-standar sebagaimana diidentifikasi dari ketiga narasumber penelitian (Subbab 3.2.2), sebuah kombinasi yang belum ditemukan baik pada penelitian akademik terdahulu maupun pada protokol *streaming payment* yang telah beroperasi di industri.

Sebagai penegasan, contoh sistem serupa yang telah beroperasi nyata di industri dan menjadi rujukan langsung penelitian ini adalah Superfluid [9] dan Sablier [10], keduanya protokol *streaming payment* berbasis EVM yang telah digunakan secara luas untuk kasus penggunaan seperti *vesting token* dan pembayaran kontinu. Namun demikian, sebagaimana diuraikan pada paragraf sebelumnya, keduanya bersifat primitif generik tanpa logika bisnis penggajian, sehingga posisi penelitian ini adalah membangun lapisan aplikasi (*application layer*) di atas prinsip *streaming* yang telah mapan tersebut, bukan mengklaim originalitas pada mekanisme *streaming* itu sendiri.

2.3. Landasan Teori

2.3.1. Web2: Arsitektur Terpusat

Web2 merujuk pada generasi kedua World Wide Web yang didominasi oleh platform terpusat, di mana satu penyedia layanan mengendalikan penyimpanan data, autentikasi pengguna, dan logika aplikasi sepenuhnya di server milik mereka sendiri [24]. Model ini memungkinkan pengembangan aplikasi yang cepat dan pengalaman pengguna yang mulus, namun menempatkan kepercayaan penuh pada satu entitas: pengguna tidak memiliki kendali langsung atas data mereka, dan penyedia layanan dapat mengubah, membatasi, atau menghentikan akses kapan saja.

Ilustrasi konkret dari kerentanan tersebut tampak pada sistem *payroll* konvensional berbasis Web2, yang pada umumnya menyimpan seluruh data gaji

dan riwayat transaksi karyawan dalam satu basis data relasional yang dikelola satu perusahaan penyedia software HR. Apabila basis data ini diretas atau seorang administrator melakukan manipulasi data, tidak tersedia jejak audit independen yang dapat diverifikasi pihak luar untuk membuktikan data asli sebelum perubahan terjadi. Ketiadaan mekanisme verifikasi independen semacam ini menjadi salah satu pendorong utama kebutuhan akan arsitektur alternatif yang lebih transparan. Karakteristik inilah yang membedakan Web2 secara mendasar dari paradigma Web3 yang dibahas pada subbab berikut.

2.3.2. Web3: Arsitektur Terdesentralisasi

Web3 adalah paradigma lanjutan dari World Wide Web yang mengintegrasikan teknologi *blockchain* untuk mendesentralisasikan kendali data, komputasi, dan *governance* ke banyak partisipan yang tersebar, alih-alih bergantung pada satu penyedia layanan tunggal [24]. Alih-alih server terpusat, Web3 menggunakan jaringan *node* yang tersebar dan *smart contract* untuk menjalankan logika aplikasi secara transparan dan dapat diverifikasi publik.

Penerapan prinsip tersebut tercermin pada penelitian ini, di mana seluruh logika inti penggajian, mencakup perhitungan saldo *stream*, klaim gaji, dan persetujuan PHK, dijalankan pada *smart contract* CompanyVault yang ter-*deploy* di jaringan Base, bukan pada server milik Payana. Setiap transaksi tercatat secara permanen dan dapat diverifikasi oleh siapa pun melalui *block explorer* BaseScan. Pendekatan ini menghilangkan kebutuhan untuk mempercayai klaim sepihak dari operator platform, karena kebenaran data dapat ditelusuri langsung ke sumbernya. Sifat dapat diverifikasi inilah yang menjadi landasan bagi berbagai standar teknis Ethereum yang diadopsi pada penelitian ini, sebagaimana akan diuraikan pada subbab-subbab berikutnya.

2.3.3. Decentralized Finance (DeFi)

Decentralized Finance (DeFi) adalah kategori layanan keuangan yang dibangun di atas jaringan *blockchain* publik, memungkinkan fungsi intermediasi keuangan tradisional seperti pinjaman, pertukaran aset, dan penghimpunan dana

dijalankan melalui *smart contract* tanpa lembaga keuangan perantara [25].

Prinsip DeFi tersebut diadaptasi pada penelitian ini dalam bentuk fasilitas kasbon, yaitu uang muka gaji yang didanai langsung dari *vault* perusahaan dan dilunasi secara otomatis melalui *smart contract*. Mekanisme ini tidak memerlukan bank maupun fintech pihak ketiga sebagai perantara pinjaman, sehingga proses pengajuan dan pelunasan kasbon berlangsung sepenuhnya di dalam sistem yang sama dengan penggajian. Pendekatan ini konsisten dengan karakteristik DeFi yang menghilangkan lapisan intermediasi finansial konvensional. Dengan demikian, likuiditas darurat bagi karyawan dapat diakses tanpa proses persetujuan kredit yang panjang sebagaimana lazim pada lembaga keuangan tradisional.

2.3.4. Sistem Penggajian Konvensional dan *Earned Wage Access* (EWA)

Sistem penggajian (*payroll*) secara konvensional didefinisikan sebagai proses administratif yang mencakup perhitungan kompensasi karyawan, pemotongan pajak dan iuran wajib, serta distribusi dana secara periodik, umumnya bulanan, melalui infrastruktur perbankan tersentralisasi [1]. Evolusi teknis sistem ini telah bergerak dari pencatatan manual berbasis kertas, menuju sistem berbasis basis data terpusat, hingga otomasi cerdas berbasis *cloud* computing pada dekade terakhir, namun karakteristik intinya tetap mempertahankan siklus pembayaran periodik (*batch* payment cycle) yang kaku [1].

Earned Wage Access (EWA), atau dikenal pula sebagai *on-demand pay*, merupakan model kompensasi yang memungkinkan karyawan menarik sebagian gaji yang secara proporsional telah mereka hasilkan (*accrued* wages) sebelum tanggal penggajian resmi tiba, tanpa menunggu siklus bulanan penuh [2]. Model ini muncul sebagai respons terhadap keterbatasan sistem penggajian konvensional yang tidak fleksibel terhadap kebutuhan arus kas jangka pendek karyawan, sebuah kesenjangan yang secara khusus berdampak pada pekerja dengan struktur kompensasi non-standar seperti honorarium per sesi, insentif berbasis kinerja, atau pekerja lepas (freelancer) sebagaimana diidentifikasi pada narasumber penelitian ini.

Implementasi EWA konvensional pada umumnya masih bergantung pada

penyedia jasa pihak ketiga yang menjembatani perusahaan dan penyedia dana talangan, sehingga tetap mewarisi keterbatasan infrastruktur perbankan tradisional berupa latensi settlement dan biaya transaksi antar-lembaga keuangan [2]. Penelitian ini memposisikan *streaming payment* berbasis *blockchain* sebagai pendekatan alternatif yang mengeliminasi ketergantungan pada perantara keuangan tersebut, dengan mengalihkan mekanisme akrual dan pencairan dana sepenuhnya ke logika *smart contract* yang berjalan otonom.

Situasi tersebut dapat digambarkan melalui skenario seorang karyawan yang menerima gaji setiap tanggal 25, namun pada tanggal 10 anaknya mendadak sakit dan membutuhkan biaya berobat mendesak. Pada sistem konvensional, karyawan tersebut umumnya terpaksa berutang karena gajinya belum cair. Dengan EWA, karyawan dapat menarik sebagian gaji yang telah dihasilkan sejak tanggal 1 hingga 10, bukan meminjam gaji masa depan, kapan pun dibutuhkan tanpa harus menunggu tanggal 25. Mekanisme inilah yang menjadi dasar perancangan fitur klaim gaji fleksibel pada penelitian ini.

2.3.5. *Wallet*, *Embedded wallet*, *Seed phrase*, dan *Non-custodial*

Wallet (dompet digital) adalah alat yang menyimpan kunci privat pengguna dan digunakan untuk menandatangani transaksi di jaringan *blockchain*. *Non-custodial* berarti pengguna sendiri yang memegang kendali penuh atas kunci privat tersebut, berbeda dengan *custodial wallet* seperti pada exchange, di mana penyedia layanan yang menyimpan kunci atas nama pengguna [26]. Kunci privat *non-custodial* umumnya direpresentasikan sebagai *seed phrase*, yaitu rangkaian 12 atau 24 kata yang jika hilang berarti akses ke aset di dalamnya hilang permanen, tanpa mekanisme pemulihan seperti pada sistem konvensional. *Embedded wallet* adalah pendekatan yang menyematkan *wallet non-custodial* langsung ke dalam aplikasi, memungkinkan pengguna *login* dengan metode familiar seperti email dan OTP, sementara *wallet* beserta kunci privatnya dibuat dan dikelola secara transparan di latar belakang.

Penerapan konsep tersebut terlihat pada alur *login* penelitian ini, di mana karyawan cukup memasukkan alamat surel dan kode OTP sebagaimana aplikasi

konvensional pada umumnya. Mereka tidak pernah melihat, mencatat, atau mengelola *seed phrase* secara langsung sepanjang proses tersebut. *Wallet non-custodial* mereka dibuat secara otomatis oleh Privy pada saat *login* pertama kali, namun kepemilikan kunci privatnya tetap sepenuhnya berada di tangan pengguna, bukan di tangan Payana. Pendekatan ini memungkinkan karyawan awam memperoleh manfaat kepemilikan aset *non-custodial* tanpa harus memahami kompleksitas teknis di baliknya.

2.3.6. *Gas fee*

Gas fee adalah biaya yang harus dibayarkan pengguna untuk mengeksekusi transaksi atau operasi komputasi di jaringan *blockchain* berbasis EVM, dihitung berdasarkan kompleksitas komputasi yang dibutuhkan (diukur dalam unit gas) dikalikan harga gas saat itu [27]. Mekanisme ini diperlukan untuk mencegah penyalahgunaan sumber daya jaringan yang terbatas, namun menjadi salah satu hambatan adopsi terbesar bagi pengguna non-teknis, karena mengharuskan mereka memiliki dan mengelola *cryptocurrency* hanya untuk dapat berinteraksi dengan aplikasi.

Persoalan tersebut dapat diilustrasikan melalui skenario tanpa mitigasi, di mana seorang karyawan yang hendak mengklaim gaji harus lebih dulu memiliki ETH pada *wallet* miliknya untuk membayar biaya gas. Kebutuhan semacam ini tidak masuk akal secara praktis bagi karyawan yang justru sedang membutuhkan uang, sekaligus mengharuskan mereka memahami cara memperoleh *cryptocurrency* terlebih dahulu. Penelitian ini mengeliminasi hambatan tersebut melalui penerapan *Account abstraction* (ERC-4337), di mana komponen *Paymaster* mensponsori seluruh biaya gas atas nama karyawan. Dengan demikian, klaim gaji dapat dilakukan tanpa karyawan perlu memiliki maupun memahami konsep *cryptocurrency* sama sekali.

2.3.7. *Smart contract* dan Bahasa Pemrograman Solidity

Smart contract adalah program komputer yang tersimpan dan dieksekusi di atas *blockchain*, yang secara otomatis menjalankan, mengendalikan, atau

mendokumentasikan peristiwa dan tindakan sesuai dengan persyaratan yang telah ditentukan sebelumnya dalam kodenya [6]. Berbeda dengan perjanjian konvensional yang bergantung pada penegakan oleh pihak ketiga (pengadilan, notaris, atau lembaga keuangan perantara), *smart contract* menegakkan dirinya sendiri (self-executing) melalui logika kode yang tidak dapat diintervensi begitu di-*deploy* ke jaringan, sebuah properti yang secara langsung relevan dengan kebutuhan otomasi distribusi kompensasi tanpa intervensi manual pada setiap siklus penggajian.

Solidity adalah bahasa pemrograman bertipe statis (statically-typed) yang dirancang khusus untuk menulis *smart contract* yang berjalan di atas EVM, dengan sintaksis yang terinspirasi dari bahasa pemrograman berorientasi objek seperti C++, Python, dan JavaScript. Solidity mendukung konsep pewarisan (inheritance), pustaka (libraries), dan tipe data kompleks (struct), yang pada penelitian ini dimanfaatkan untuk memodelkan entitas bisnis seperti EmployeeStream, SeveranceVault, dan SalaryAdvance sebagai struct terpisah di dalam satu kontrak CompanyVault. Sistem pada penelitian ini diimplementasikan menggunakan Solidity versi 0.8.26, yang menyertakan pemeriksaan overflow dan underflow aritmetika secara otomatis pada level bahasa sejak versi 0.8.0, mengeliminasi kelas kerentanan yang sebelumnya harus ditangani secara manual menggunakan pustaka tambahan seperti SafeMath pada versi Solidity terdahulu.

Pola desain *Checks-Effects-Interactions* (CEI) merupakan konvensi keamanan standar dalam pengembangan *smart contract* yang mewajibkan urutan eksekusi sebagai berikut: seluruh validasi kondisi (checks) dilakukan terlebih dahulu, diikuti oleh seluruh perubahan status internal kontrak (effects), dan interaksi dengan kontrak eksternal atau transfer dana (interactions) ditempatkan paling akhir. Pola ini secara sistematis mencegah kerentanan *reentrancy*, yaitu kondisi di mana kontrak eksternal yang dipanggil dapat memanggil balik fungsi asal sebelum status internal selesai diperbarui, sehingga berpotensi menguras dana kontrak melalui pemanggilan berulang. Seluruh fungsi kritis pada kontrak CompanyVault penelitian ini, termasuk claimSalary() dan _autoRepayAdvance(), dirancang mengikuti pola ini secara ketat.

Cara kerja *smart contract* dapat diilustrasikan seperti mesin penjual otomatis versi digital untuk urusan keuangan perusahaan. Begitu syaratnya terpenuhi, misalnya telah tiba waktu gajian dan karyawan berhak menerima, mesin ini secara otomatis mencairkan dana tanpa memerlukan staf HR menekan tombol transfer secara manual. Tidak ada satu pihak pun, termasuk pemilik perusahaan, yang dapat mengubah aturan pencairan tersebut secara sepihak setelah kontrak diaktifkan di *blockchain*. Sifat otomatis dan tidak dapat diubah inilah yang menjadi dasar kepercayaan pada seluruh alur penggajian yang dirancang pada penelitian ini.

2.3.8. *Stablecoin* (Standar ERC-20) dan *Streaming payment*

Ethereum *Request* for Comment (ERC) adalah kategori proposal standar teknis dalam ekosistem Ethereum yang mendefinisikan aturan, antarmuka, atau perilaku yang harus dipatuhi agar suatu *smart contract* dapat berinteraksi secara seragam dengan *smart contract* lain, dompet digital, maupun aplikasi pihak ketiga di jaringan yang sama. Setiap ERC diberi nomor unik sesuai urutan pengajuannya sebagai Ethereum Improvement Proposal (EIP), dan setelah disetujui menjadi rujukan standar yang diadopsi luas oleh pengembang. Penelitian ini mengadopsi tiga standar ERC: ERC-20 untuk *token* fungibel (*stablecoin*), ERC-4337 untuk *Account abstraction*, dan ERC-5192 untuk *Soulbound Token*, yang masing-masing dijelaskan pada subbab berikut.

ERC-20 adalah standar teknis interoperabilitas *token* fungibel (fungible *token*) di jaringan berbasis EVM [12], yang mendefinisikan sekumpulan fungsi wajib seperti transfer, balanceOf, dan approve, sehingga *token* yang mengimplementasikan standar ini dapat berinteraksi secara seragam dengan dompet digital, bursa, dan kontrak pihak ketiga mana pun tanpa memerlukan integrasi khusus. *Stablecoin* adalah kategori *token* ERC-20 yang nilainya dipatok (pegged) terhadap aset stabil, umumnya mata uang fiat seperti Dolar Amerika Serikat atau Rupiah, sehingga menghadirkan properti stabilitas harga yang tidak dimiliki oleh *cryptocurrency* volatil seperti Bitcoin atau Ethereum [7]. Stabilitas ini menjadikan *stablecoin* sebagai instrumen pembayaran yang layak untuk kompensasi tenaga kerja, karena karyawan tidak perlu menanggung risiko fluktuasi nilai tukar antara

waktu penerimaan dan waktu penggunaan dana [2].

Penelitian ini menggunakan IDRX, sebuah *stablecoin* yang mengimplementasikan standar ERC-20 dengan nilai dipatok 1:1 terhadap Rupiah Indonesia, sebagai instrumen pembayaran utama pada seluruh alur transaksi sistem, meliputi klaim gaji, kasbon, dan pesangon. Pemilihan *stablecoin* berdenominasi Rupiah dilakukan untuk mengeliminasi risiko volatilitas yang melekat pada *cryptocurrency* konvensional, sekaligus mempertahankan properti transparansi dan settlement instan yang menjadi keunggulan infrastruktur *blockchain* dibandingkan sistem perbankan konvensional [7].

Streaming payment adalah mekanisme distribusi dana yang bersifat kontinu (continuous), di mana penerima memperoleh akrual nilai secara berkelanjutan berdasarkan laju alir (*flow rate*) yang telah disepakati, bukan melalui transfer diskrit pada interval waktu tertentu sebagaimana pada sistem penggajian konvensional [5]. Secara matematis, saldo yang dapat diklaim pada suatu waktu dihitung sebagai hasil kali antara *flow rate* dan durasi yang telah berlalu sejak titik akrual terakhir, sehingga karyawan pada prinsipnya dapat menarik kompensasi yang telah mereka hasilkan kapan pun tanpa menunggu siklus penggajian tetap. Pendekatan trigger-based di mana eksekusi pembayaran dipicu oleh kondisi yang telah diprogramkan sebelumnya, sejalan dengan konsep *programmable payment* yang memungkinkan otomasi penuh atas logika distribusi dana tanpa memerlukan intervensi manual pada setiap transaksi [5], sebuah prinsip yang diimplementasikan pada fungsi startStream() dan claimSalary() penelitian ini (Subbab 4.4.2).

Penerapan konsep tersebut tercermin pada pemilihan IDRX sebagai instrumen pembayaran penelitian ini, yang dapat dianalogikan seperti saldo uang elektronik pada aplikasi dompet digital konvensional, hanya saja berjalan di atas infrastruktur *blockchain*. Nilai 1 IDRX selalu setara dengan 1 Rupiah, sehingga karyawan yang menerima gaji dalam bentuk IDRX tidak perlu khawatir nilainya berfluktuasi sebagaimana harga *cryptocurrency* volatil seperti Bitcoin. Kestabilan nilai ini menjadi syarat mutlak bagi kelayakan *stablecoin* sebagai instrumen kompensasi tenaga kerja. Tanpa kestabilan tersebut, risiko nilai tukar akan berpindah dari perusahaan kepada karyawan, sebuah kondisi yang bertentangan

dengan tujuan penelitian ini.

2.3.9. *Account abstraction* (Standar ERC-4337) dan Transaksi *Gasless*

Pada desain Ethereum konvensional, setiap transaksi harus berasal dari sebuah Externally Owned Account (EOA) yang dikendalikan oleh satu kunci privat berbasis algoritma tanda tangan digital ECDSA, dan setiap transaksi mewajibkan pembayaran biaya komputasi (*gas fee*) dalam bentuk Ether oleh pengirim transaksi tersebut. Persyaratan ini menghadirkan hambatan adopsi yang signifikan bagi pengguna non-teknis, karena mengharuskan mereka memiliki dan mengelola aset kripto hanya untuk dapat berinteraksi dengan aplikasi, sebuah beban kognitif dan finansial yang secara langsung bertentangan dengan prinsip kemudahan penggunaan yang dibutuhkan sistem penggajian konsumer.

ERC-4337 adalah standar *Account abstraction* yang memungkinkan logika validasi transaksi dipindahkan dari protokol inti Ethereum ke lapisan aplikasi (application *layer*), tanpa memerlukan perubahan pada protokol konsensus yang mendasarinya (*hard fork*) [13]. Standar ini memperkenalkan objek transaksi baru bernama *UserOperation* yang diproses melalui mempool alternatif yang independen dari mempool transaksi Ethereum konvensional, kemudian dieksekusi oleh kontrak singleton bernama *EntryPoint* yang memvalidasi dan mengeksekusi setiap *UserOperation* terhadap *smart contract wallet* milik pengguna [13].

Proses klaim gaji *gasless* pada penelitian ini dapat dianalogikan seperti pengiriman paket dengan ongkos kirim ditanggung penerima, hanya saja arahnya dibalik: karyawan cukup menekan tombol "Tarik Gaji" pada aplikasi, sementara pihak Payana melalui komponen *Paymaster* yang menanggung biaya pengiriman transaksi ke jaringan *blockchain*. Karyawan tidak perlu membayar biaya apa pun maupun memahami cara kerja jaringan *blockchain* di baliknya. Seluruh proses validasi dan penyampaian transaksi berlangsung sepenuhnya di latar belakang sistem. Pendekatan ini menjadi salah satu faktor utama yang menjawab hambatan adopsi teknis sebagaimana dirumuskan pada rumusan masalah penelitian.

2.3.10. *Soulbound Token* (Standar ERC-5192) untuk Identitas Digital

Non-Fungible *Token* (NFT) yang mengimplementasikan standar ERC-721 secara konvensional dirancang agar dapat dipindahtangankan (transferable) secara bebas antar-pemilik, sebuah properti yang sesuai untuk representasi kepemilikan aset digital yang dapat diperjualbelikan, namun tidak sesuai untuk representasi atribut yang secara inheren melekat pada identitas seseorang, seperti sertifikat, kredensial, atau riwayat afiliasi kerja, karena sifat dapat dipindahtangankan tersebut memungkinkan kredensial diperjualbelikan secara tidak sah.

Soulbound Token (SBT) adalah kategori *token* non-fungible yang secara permanen terikat pada satu alamat kepemilikan dan tidak dapat dipindahtangankan (non-transferable) setelah diterbitkan. ERC-5192 adalah standar teknis yang mendefinisikan antarmuka minimal untuk mewujudkan properti *soulbound* tersebut sebagai ekstensi dari ERC-721, melalui fungsi locked() yang mengembalikan status penguncian suatu *token*, serta mewajibkan seluruh fungsi transfer bawaan ERC-721 untuk menolak (revert) eksekusinya apabila status *token* yang bersangkutan terkunci [14]. Deteksi dukungan antarmuka ini dilakukan melalui mekanisme ERC-165, sehingga dompet digital dan aplikasi pihak ketiga dapat mengenali dan memperlakukan *token* tersebut secara berbeda dari NFT konvensional [14].

Penelitian ini mengimplementasikan ERC-5192 pada kontrak EmploymentSBT untuk menerbitkan sertifikat ketenagakerjaan digital yang tidak dapat dipalsukan maupun diperjualbelikan, sebagai representasi *on-chain* atas hubungan kerja yang sah antara karyawan dan perusahaan (Subbab 4.4.3). Sifat non-transferable pada standar ini secara langsung memenuhi kebutuhan validasi identitas kepegawaian yang tidak dapat dimanipulasi, karena upaya pemindahtanganan sertifikat akan selalu ditolak oleh kontrak sesuai spesifikasi standar, berbeda dengan dokumen kepegawaian konvensional berbasis kertas atau berkas digital yang rentan dipalsukan.

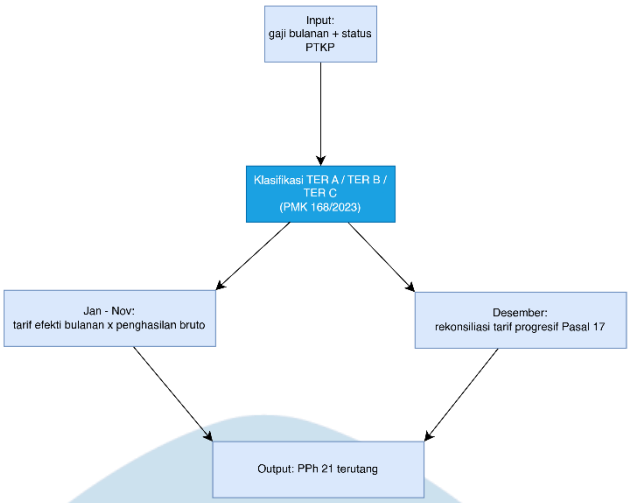
Karakteristik *Soulbound Token* dapat diilustrasikan melalui analogi ijazah atau KTP, yaitu dokumen yang melekat permanen pada identitas pemiliknya dan tidak dapat dijual maupun dipindahtangankan kepada orang lain, berbeda dengan sertifikat tanah yang dapat diperjualbelikan secara bebas. Sertifikat kerja digital

karyawan pada penelitian ini dibuat mengikuti prinsip serupa, sehingga tidak dapat dipalsukan maupun "dijual" kepada pihak yang sebenarnya tidak bekerja di perusahaan tersebut. Setiap upaya pemindahtanganan akan secara otomatis ditolak oleh kontrak sesuai spesifikasi standar ERC-5192. Properti non-transferable inilah yang menjadikan SBT mekanisme validasi identitas yang lebih andal dibandingkan dokumen kepegawaian konvensional berbasis kertas.

2.3.11. Kepatuhan Penggajian di Indonesia: PPh21, BPJS, dan Pesangon

Berbeda dengan protokol *streaming payment* generik seperti Superfluid dan Sablier yang bersifat agnostik terhadap yurisdiksi, sistem penggajian yang beroperasi di Indonesia terikat pada tiga kewajiban regulasi yang saling terpisah: pemotongan Pajak Penghasilan Pasal 21 (PPh21), iuran Badan Penyelenggara Jaminan Sosial Ketenagakerjaan (BPJS Ketenagakerjaan), dan kompensasi pemutusan hubungan kerja (PHK). Ketiganya memiliki basis hukum dan skema perhitungan yang berbeda, sehingga integrasi ke dalam logika *smart contract* memerlukan pemahaman atas aturan yang mendasarinya, bukan sekadar pemotongan persentase tetap.

PPh21 di Indonesia sejak Januari 2024 dihitung menggunakan skema Tarif Efektif Rata-rata (TER) sebagaimana diatur dalam PMK Nomor 168 Tahun 2023 [17]. Skema ini mengklasifikasikan pegawai ke dalam tiga kelompok tarif (TER A, B, dan C) berdasarkan status Penghasilan Tidak Kena Pajak (PTKP), dan menerapkan tarif efektif bulanan pada sebelas bulan pertama, sebelum direkonsiliasi menggunakan tarif progresif Pasal 17 pada bulan Desember [18]. Penelitian Santioso dkk. mencatat bahwa penerapan skema ini di lapangan masih rawan kesalahan pemotongan, khususnya ketika komponen penghasilan tidak rutin seperti bonus turut diperhitungkan, sebuah kompleksitas yang relevan bagi sistem otomatis yang menghitung kompensasi secara kontinu, bukan pada satu titik waktu pembayaran bulanan [18]. Penelitian ini mengimplementasikan skema TER melalui parameter `pph21Bps` yang dapat dikonfigurasi HR Admin per perusahaan pada fungsi `setCompanyConfig()`, alih-alih tarif progresif statis (Subbab 4.4.2).



Gambar 2.1. Alur Perhitungan PPh21 Skema Tarif Efektif Rata-rata (TER) sesuai PMK Nomor 168 Tahun 2023

Iuran BPJS Ketenagakerjaan merupakan kewajiban terpisah yang dibagi antara kontribusi pemberi kerja dan pekerja, dengan besaran yang ditetapkan pemerintah dan dapat berubah dari waktu ke waktu. Karena sifatnya yang dapat berubah mengikuti kebijakan pemerintah, penelitian ini memperlakukan persentase BPJS sebagai parameter yang dapat dikonfigurasi (`bpjsBps`) alih-alih nilai tetap yang di-hardcode pada kontrak, sehingga perusahaan dapat menyesuaikan tanpa memerlukan migrasi kontrak.

Kompensasi PHK di Indonesia diatur dalam PP Nomor 35 Tahun 2021 [19], yang mewajibkan pemberi kerja membayar uang pesangon, uang penghargaan masa kerja, dan uang penggantian hak, dengan besaran yang bergantung pada masa kerja dan alasan pemutusan hubungan kerja. Utama dkk. menekankan bahwa proses pemberian kompensasi ini idealnya melalui mekanisme yang dapat diverifikasi dan tidak dapat disangkal oleh kedua belah pihak, mengingat PHK kerap menjadi sumber sengketa ketenagakerjaan [20]. Penelitian ini menerjemahkan kebutuhan verifikasi tersebut ke dalam mekanisme *governance* dua tahap pada kontrak `CompanyVault`, yaitu `proposeTermination()` yang mencatat alasan pemutusan secara *on-chain* melalui `reasonHash`, diikuti oleh `executeTermination()` setelah periode tunggu terlampaui, sehingga keputusan PHK tidak dapat dieksekusi secara sepihak maupun instan.

Penerapan kepatuhan pajak pada penelitian ini dapat diilustrasikan melalui skenario seorang karyawan dengan gaji sebesar Rp8.000.000 per bulan, yang akan

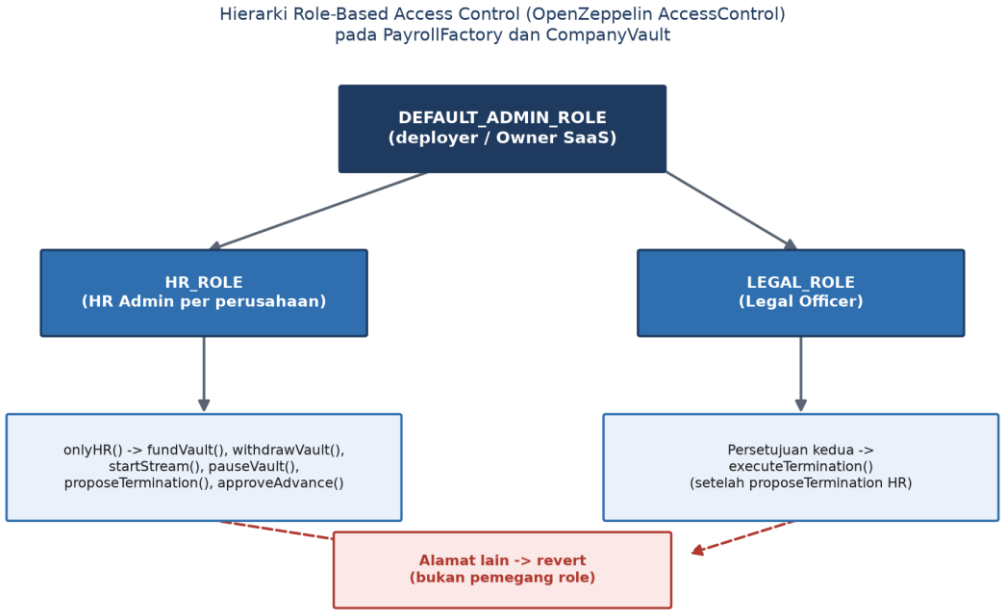
secara otomatis dipotong PPh21 sesuai kelompok tarif TER yang berlaku beserta iuran BPJS Ketenagakerjaan setiap kali mengklaim gaji melalui aplikasi. Proses ini menghasilkan hasil akhir yang serupa dengan slip gaji konvensional, yang juga mencantumkan potongan pajak dan BPJS secara eksplisit. Perbedaannya terletak pada mekanismenya: perhitungan dan pemotongan pada penelitian ini dilakukan otomatis oleh *smart contract*, bukan dihitung manual oleh staf HR. Pendekatan ini mengeliminasi risiko kesalahan hitung manual sekaligus memastikan kepatuhan diterapkan secara konsisten pada setiap transaksi klaim gaji.

2.3.12. Kontrol Akses dan Mitigasi Kerentanan pada *Smart contract*

Kontrak yang menampung dana terkumpul dari banyak pengguna, sebagaimana `CompanyVault` pada penelitian ini, merupakan target yang secara inheren menarik bagi penyerang, karena sifat immutable dan transparan *blockchain* berarti kerentanan yang tereksploitasi tidak dapat diperbaiki secara reaktif setelah dana berpindah tangan. Rezaei dkk., dalam tinjauan literatur sistematis atas insiden serangan *smart contract* dunia nyata, mencatat bahwa kerugian finansial akibat kerentanan kontrak telah mencapai lebih dari satu miliar dolar Amerika Serikat, dengan *reentrancy* dan kegagalan kontrol akses sebagai dua kategori penyebab paling dominan [21]. Temuan ini menjadi dasar bagi penelitian ini untuk menerapkan mitigasi berlapis (*defense-in-depth*), bukan mengandalkan satu mekanisme keamanan tunggal.

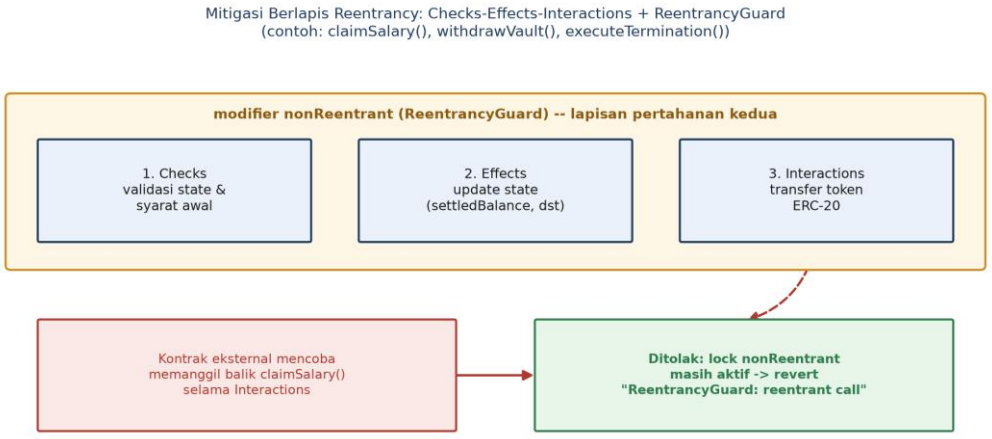
Pada aspek kontrol akses, penelitian ini menggunakan pustaka `AccessControl` dari OpenZeppelin yang mengimplementasikan *Role-Based Access Control* (RBAC), alih-alih pola kepemilikan tunggal (Ownable) yang lebih umum digunakan pada kontrak sederhana. Kalapaaking dkk. menunjukkan bahwa model akses berbasis peran lebih sesuai untuk sistem yang melibatkan lebih dari satu jenis aktor dengan hak yang berbeda-beda, dibandingkan model kepemilikan tunggal yang mengasumsikan satu entitas administratif [23]. Karakteristik ini relevan dengan kebutuhan penelitian ini, di mana setiap perusahaan memiliki HR Admin masing-masing yang harus diisolasi hak aksesnya dari perusahaan lain pada kontrak `PayrollFactory` dan `CompanyVault` yang sama, direalisasikan melalui

modifier `onlyHR()` yang memverifikasi peran pemanggil fungsi sebelum eksekusi (Subbab 4.4.2).



Gambar 2.2. Hierarki Role-Based Access Control (OpenZeppelin AccessControl) pada PayrollFactory dan CompanyVault

Pada aspek *reentrancy*, penelitian ini menerapkan mitigasi dua lapis: pola desain *Checks-Effects-Interactions* pada level penulisan kode, dilapisi dengan modifier `nonReentrant` dari pustaka `ReentrancyGuard` OpenZeppelin pada seluruh fungsi yang memindahkan dana, termasuk `claimSalary()`, `withdrawVault()`, dan `executeTermination()`. Kombinasi ini merupakan penerapan prinsip *defense-in-depth* yang direkomendasikan Rezaei dkk. [21]: CEI mencegah kerentanan pada level logika, sementara `ReentrancyGuard` bertindak sebagai lapisan pertahanan independen yang tetap mencegah eksploitasi apabila kesalahan implementasi CEI luput dari peninjauan kode.



Gambar 2.3. Mitigasi Berlapis *Reentrancy*: Checks-Effects-Interactions dan ReentrancyGuard

Sebagai lapisan mitigasi tambahan pada level operasional, kontrak `CompanyVault` menyediakan fungsi `pauseVault()`/`resumeVault()` dan `pauseStream()`/`resumeStream()` yang mengimplementasikan pola *circuit breaker* (*emergency stop*), yaitu mekanisme yang memungkinkan penghentian sementara fungsi kritis kontrak ketika terdeteksi anomali atau kondisi darurat, tanpa memerlukan migrasi atau penggantian kontrak. Pola ini menjadi lapisan pertahanan terakhir yang melengkapi kontrol akses dan proteksi *reentrancy*, sekaligus menjadi mekanisme respons operasional yang terhubung dengan modul deteksi anomali keamanan pada penelitian ini (Subbab 4.4.4).

Pada aspek arsitektur *Account abstraction*, penelitian ini mengambil pilihan desain yang tidak umum: kontrak `CompanyVault` mewarisi langsung dari `BasePaymaster`, sehingga satu kontrak yang sama menangani baik custody dana penggajian maupun logika *sponsor* gas ERC-4337. Wang dan Chen, dalam analisis keamanan formal pertama atas ERC-4337, menekankan bahwa validasi *UserOperation* oleh *Paymaster* merupakan permukaan kepercayaan (*trust surface*) yang independen dari validasi akun pengguna itu sendiri, dan kegagalan pada lapisan ini dapat dieksploitasi tanpa perlu membobol *smart contract wallet* pengguna [22]. Penggabungan peran *vault* dan *paymaster* pada satu kontrak berarti permukaan risiko keduanya turut menyatu; penelitian ini memitigasi hal tersebut dengan membatasi fungsi *sponsor* gas hanya pada operasi yang telah tervalidasi

onlyHR/`vaultActive`, sehingga logika *Paymaster* tidak menjadi jalur pintas yang melewati kontrol akses *vault* utama.

Pola kontrol akses tersebut dapat diilustrasikan melalui analogi sistem keamanan gedung kantor bertingkat, di mana kartu akses karyawan biasa hanya dapat membuka pintu lift dan ruang kerja, sementara kartu akses manajer HR dapat membuka ruang penggajian, dan hanya pemegang kunci utama yang dapat mengubah siapa saja yang menerima kartu akses jenis tertentu. Sistem Payana menerapkan prinsip yang serupa secara digital, di mana hanya *wallet* yang terdaftar sebagai HR Admin yang dapat memindahkan dana dari *vault* perusahaan. Upaya akses oleh alamat *wallet* lain akan secara otomatis ditolak oleh kontrak melalui modifier onlyHR(). Analogi ini menggambarkan bagaimana kontrol akses berbasis peran diterjemahkan dari konsep keamanan fisik konvensional ke dalam logika *smart contract*.

2.3.13. Teknologi *Blockchain* dan Ethereum *Virtual Machine* (EVM)

Blockchain adalah struktur data terdistribusi yang mencatat transaksi dalam rangkaian blok yang saling terhubung secara kriptografis, direplikasi di seluruh *node* partisipan jaringan, dan divalidasi melalui mekanisme konsensus tanpa memerlukan otoritas tersentralisasi [8]. Karakteristik immutability (data yang telah tercatat tidak dapat diubah secara sepihak) dan transparansi (seluruh riwayat transaksi dapat diverifikasi secara publik) menjadikan *blockchain* sebagai fondasi yang relevan bagi sistem yang membutuhkan jejak audit yang tidak dapat disangkal, sebagaimana dibutuhkan pada proses penggajian dan kepatuhan pajak.

Ethereum, sebagai salah satu implementasi *blockchain* generasi kedua, memperluas fungsi *blockchain* dari sekadar pencatatan transaksi nilai (value transfer) menjadi mesin komputasi terdistribusi dengan status bersama (shared-*state* transactional singleton machine) yang dikenal sebagai Ethereum Virtual Machine (EVM) [11]. EVM menyediakan lingkungan eksekusi sandboxed yang deterministik, artinya kode program (*smart contract*) yang dijalankan akan selalu menghasilkan output identik pada setiap *node* yang mengeksekusinya, sebuah properti yang menjadi prasyarat bagi konsensus terdistribusi [11].

Penelitian ini menggunakan Base, sebuah jaringan *Layer*-2 yang dibangun di atas teknologi Optimistic *Rollup* dan kompatibel penuh dengan EVM (EVM-equivalent). Pendekatan *Layer*-2 memproses sebagian besar komputasi transaksi di luar rantai utama (*off-chain*) dan hanya mengirimkan bukti keabsahan (proof) secara periodik ke jaringan utama Ethereum, sehingga secara signifikan menekan biaya gas dan meningkatkan *throughput* transaksi dibandingkan eksekusi langsung di jaringan utama (*mainnet*), tanpa mengorbankan properti keamanan dan desentralisasi yang diwariskan dari Ethereum sebagai lapisan penyelesaian akhir (settlement *layer*).

Prinsip konsensus EVM dapat diilustrasikan melalui analogi ribuan kalkulator identik yang dioperasikan serentak oleh orang-orang berbeda di seluruh dunia. Ketika *smart contract* Payana menghitung gaji seorang karyawan, seluruh "kalkulator" tersebut menghasilkan angka yang persis sama. Tidak ada satu pihak pun yang dapat secara diam-diam mengubah hasil perhitungan gaji secara sepihak, karena setiap orang dapat memeriksa dan membandingkan hasil tersebut secara independen. Properti deterministik inilah yang menjadi prasyarat bagi kepercayaan terhadap sistem penggajian tanpa otoritas pusat yang tunggal.

2.3.14. EVM Layer 2

Layer-2 (L2) adalah jaringan *blockchain* yang dibangun di atas jaringan utama (*Layer*-1) seperti Ethereum, memproses sebagian besar komputasi transaksi secara *off-chain* dan hanya mengirimkan bukti keabsahan (proof) secara periodik ke *Layer*-1 sebagai lapisan penyelesaian akhir (settlement *layer*). Pendekatan ini secara signifikan menekan biaya gas dan meningkatkan *throughput* transaksi dibandingkan eksekusi langsung di *Layer*-1, tanpa mengorbankan properti keamanan dan desentralisasi yang diwariskan dari jaringan utamanya. Penelitian ini menggunakan Base, jaringan *Layer*-2 berbasis teknologi Optimistic *Rollup* yang kompatibel penuh dengan EVM (EVM-equivalent). Dengan Base, biaya gas untuk sekali klaim gaji jauh lebih murah dibandingkan jika transaksi yang sama dieksekusi langsung di jaringan utama Ethereum, sekaligus tetap mewarisi jaminan keamanan Ethereum sebagai settlement *layer*.

Penerapan *Layer*-2 pada penelitian ini terlihat pada pemilihan jaringan Base sebagai infrastruktur utama sistem Payana. Biaya gas untuk sekali klaim gaji pada jaringan tersebut jauh lebih murah dibandingkan apabila transaksi yang sama dieksekusi langsung pada jaringan utama Ethereum, sementara jaminan keamanan tetap diwariskan penuh dari Ethereum sebagai settlement *layer*. Karakteristik inilah yang menjadikan Base pilihan yang sesuai bagi sistem penggajian yang memerlukan frekuensi transaksi tinggi dengan biaya rendah. Tanpa pendekatan *Layer*-2, model *streaming payment* yang diusung penelitian ini akan sulit direalisasikan secara ekonomis pada skala produksi.

2.3.15. RPC dan Jaringan Uji Base Sepolia

Kendala praktis penggunaan RPC dialami langsung pada penelitian ini, yang semula menggunakan layanan RPC dari Alchemy namun harus beralih ke RPC publik Base Sepolia setelah kuota bulanan Alchemy habis di tengah sesi pengujian. Kejadian ini merupakan bukti nyata keterbatasan infrastruktur yang dapat muncul sewaktu-waktu pada penyedia RPC pihak ketiga. Kendala tersebut sekaligus memperkuat argumen kebutuhan NFR-PAYANA-07, yang mensyaratkan mekanisme *retry* dan fallback RPC agar sistem tetap andal ketika satu penyedia mengalami gangguan atau pembatasan. Dengan demikian, pengalaman operasional ini menjadi justifikasi empiris, bukan sekadar asumsi teoretis, bagi kebutuhan non-fungsional tersebut.

2.3.16. *Blockchain Indexer* dan GraphQL (*Framework* Ponder)

Data yang tersimpan langsung di dalam *state blockchain* bersifat sulit dikueri secara efisien untuk kebutuhan aplikasi, karena *node blockchain* pada dasarnya dioptimalkan untuk validasi dan konsensus transaksi, bukan untuk kueri agregat kompleks seperti "tampilkan seluruh riwayat klaim gaji karyawan tertentu dalam tiga bulan terakhir". *Blockchain indexer* adalah lapisan *middleware* yang berlangganan (subscribe) terhadap *event* yang diterbitkan oleh *smart contract*, kemudian menyalin dan menstrukturkan data tersebut ke dalam basis data relasional yang dapat dikueri secara efisien oleh lapisan aplikasi, tanpa aplikasi perlu

berinteraksi langsung dengan *node blockchain* untuk setiap kebutuhan pembacaan data.

Ponder adalah *framework blockchain indexer* sumber terbuka (open-source) berbasis TypeScript yang memungkinkan pengembang mendefinisikan skema basis data dan fungsi penanganan *event* (*event handler*) secara deklaratif, kemudian secara otomatis melakukan *sinkronisasi* historis (backfill) dan *real-time* terhadap *event on-chain* ke dalam basis data PostgreSQL. Penelitian ini menggunakan Ponder versi 0.16.6 untuk mengindeks seluruh *event* yang diterbitkan oleh kontrak PayrollFactory, CompanyVault, dan EmploymentSBT ke dalam 13 tabel skema public, yang kemudian diekspos melalui antarmuka GraphQL bawaan *framework* tersebut.

GraphQL adalah bahasa kueri dan runtime API yang memungkinkan klien meminta struktur data secara presisi sesuai kebutuhan dalam satu permintaan tunggal, berbeda dengan pendekatan REST konvensional yang seringkali mengharuskan multiple round-trip permintaan atau mengembalikan data berlebih (over-fetching) untuk memperoleh struktur data bersarang yang sama. Karakteristik ini dimanfaatkan pada lapisan *frontend* penelitian ini untuk memperoleh data gabungan lintas-entitas, seperti status *stream* beserta riwayat klaim dan saldo pesangon karyawan tertentu, dalam satu kali permintaan kueri, meningkatkan efisiensi pemuatan data pada antarmuka *dashboard* dibandingkan pendekatan REST API konvensional.

Peran Ponder pada penelitian ini dapat diilustrasikan seperti seorang asisten pencatat yang secara terus-menerus menyalin setiap transaksi yang terjadi di *blockchain* ke dalam sebuah buku besar yang rapi dan mudah dicari. Ketika seorang karyawan membuka riwayat klaim gajinya pada aplikasi, data akan langsung muncul dengan cepat dari "buku besar" tersebut, tanpa aplikasi perlu mencari satu per satu ke seluruh riwayat *blockchain* setiap kali halaman dibuka. Pendekatan ini secara signifikan meningkatkan kecepatan pemuatan data pada antarmuka pengguna. Tanpa lapisan indeksasi semacam ini, pengalaman pengguna aplikasi akan jauh lebih lambat dibandingkan aplikasi Web2 konvensional yang menjadi pembanding penelitian ini.

2.3.17. REST versus GraphQL

REST (*Representational State Transfer*) adalah gaya arsitektur API yang mengorganisasikan data ke dalam resource dengan *endpoint* tetap, di mana setiap permintaan bersifat *stateless* dan biasanya mengembalikan struktur data yang sudah ditentukan sebelumnya oleh server, bukan oleh klien [30]. Karakteristik ini kerap mengharuskan klien melakukan beberapa kali permintaan berantai (multiple round-trip) atau menerima data berlebih (over-fetching) untuk memperoleh data bersarang yang saling terkait, sebagaimana dijelaskan pada subbab GraphQL berikut.

Penerapan kedua gaya arsitektur tersebut dapat dilihat pada penelitian ini, yang menggunakan kombinasi keduanya sesuai kebutuhan masing-masing. REST API digunakan untuk operasi *backend* seperti autentikasi dan *relay* transaksi *gasless*, sementara GraphQL Ponder digunakan untuk kueri data gabungan lintas-entitas pada *dashboard*. Sebagai gambaran, status *stream* beserta riwayat klaim dan saldo pesangon seorang karyawan dapat diperoleh dalam satu kali permintaan GraphQL, alih-alih beberapa permintaan REST yang terpisah. Pemilihan ini didasarkan pada karakteristik masing-masing kebutuhan data, bukan preferensi teknis semata.

2.3.18. Application Binary Interface (ABI)

Application Binary Interface (ABI) adalah spesifikasi standar yang mendefinisikan cara berinteraksi dengan *smart contract* dari luar *blockchain*, mencakup nama fungsi, tipe parameter, dan format data yang dikembalikan [32]. ABI diperlukan karena bytecode *smart contract* yang ter-*deploy* di *blockchain* tidak menyimpan informasi nama fungsi secara langsung; aplikasi pihak luar (*backend*, *frontend*) memerlukan berkas ABI untuk mengetahui bagaimana cara memanggil suatu fungsi dan bagaimana menafsirkan hasil yang dikembalikannya.

Pentingnya kesesuaian ABI terbukti nyata pada salah satu cacat yang ditemukan selama pengujian penelitian ini, yaitu ketika fungsi getter Solidity untuk struct yang memiliki field string ternyata mengembalikan tiga output terpisah, bukan satu tuple sebagaimana diasumsikan kode *frontend*. Ketidaksesuaian ini menyebabkan halaman verifikasi sertifikat gagal menampilkan data, meskipun data

pada kontrak itu sendiri sepenuhnya benar. Cacat tersebut hanya terungkap melalui pengujian antarmuka sungguhan, bukan melalui inspeksi kode maupun pengujian unit kontrak secara terisolasi. Temuan ini menegaskan bahwa ketidaksesuaian ABI merupakan kelas kesalahan yang nyata dan berdampak langsung pada pengalaman pengguna, bukan sekadar risiko teoretis.

2.3.19. *Webhook*

Webhook adalah mekanisme di mana satu sistem secara otomatis mengirimkan notifikasi (*payload* data) ke URL tertentu milik sistem lain segera setelah suatu peristiwa terjadi, sebagai kebalikan dari pendekatan *polling* di mana sistem penerima harus berulang kali bertanya "apakah ada yang baru?" pada interval waktu tertentu. Pendekatan ini lebih efisien karena notifikasi dikirim tepat saat peristiwa terjadi, bukan menunggu siklus pengecekan berikutnya.

Penerapan *webhook* pada penelitian ini digunakan untuk menerima notifikasi otomatis setiap kali *event* SalaryClaimed diterbitkan kontrak, yang kemudian diteruskan sebagai notifikasi *real-time* ke antarmuka karyawan. Salah satu cacat nyata yang ditemukan selama pengujian adalah ketidakcocokan *topic hash* pada *webhook* tersebut, yang sempat menyebabkan status keberhasilan klaim gaji tidak pernah ter-broadcast ke antarmuka meskipun transaksi itu sendiri berhasil dieksekusi *on-chain*. Cacat ini menunjukkan bahwa keberhasilan transaksi pada level kontrak tidak serta-merta menjamin keberhasilan notifikasi pada lapisan di atasnya. Temuan tersebut memperkuat pentingnya pengujian *end-to-end* yang mencakup seluruh rantai komponen, bukan hanya pengujian kontrak secara terisolasi.

2.3.20. JSON Web *Token* (JWT) dan API

Application Programming Interface (API) adalah antarmuka yang memungkinkan dua sistem perangkat lunak saling berkomunikasi dan bertukar data mengikuti aturan tertentu. *JSON Web Token* (JWT) adalah standar terbuka untuk merepresentasikan klaim (*claim*) secara ringkas dan mandiri antar dua pihak dalam format JSON yang ditandatangani secara digital, umumnya digunakan sebagai

bearer token untuk membuktikan identitas pengguna pada setiap permintaan API tanpa server perlu menyimpan status sesi (*stateless*) [31].

Penerapan JWT pada penelitian ini digunakan untuk autentikasi API *backend* setelah pengguna berhasil *login* melalui Privy. Setiap permintaan ke *endpoint* yang memerlukan otorisasi, seperti persetujuan kasbon oleh HR Admin, menyertakan JWT pada header permintaan. *Backend* kemudian memverifikasi tanda tangan *token* tersebut untuk memastikan permintaan benar-benar berasal dari pengguna yang telah terautentikasi dan memiliki peran yang sesuai. Mekanisme ini memungkinkan setiap permintaan API diverifikasi secara independen tanpa *backend* perlu menyimpan status sesi pengguna secara terpusat.

2.3.21. Pengujian dan Evaluasi Sistem Terdesentralisasi (DApp)

Aplikasi terdesentralisasi (Decentralized Application, *DApp*) memiliki karakteristik pengujian yang berbeda dari aplikasi konvensional, karena sebagian logika bisnisnya dieksekusi secara immutable di atas *blockchain*: cacat pada *smart contract* yang telah di-*deploy* tidak dapat diperbaiki secara langsung sebagaimana memperbarui kode di server konvensional, melainkan memerlukan mekanisme migrasi atau upgrade kontrak yang kompleks dan berisiko. Karakteristik ini menjadikan pengujian pra-*deployment* sebagai tahap yang jauh lebih kritis pada pengembangan *DApp* dibandingkan pengembangan perangkat lunak konvensional. Foundry adalah toolkit pengembangan dan pengujian *smart contract* sumber terbuka (*open-source*) yang dikembangkan oleh Paradigm dan ditulis dalam bahasa *Rust* untuk kecepatan eksekusi tinggi. Pengujian unit (unit testing) pada *smart contract* umumnya dilaksanakan menggunakan *framework* pengembangan khusus seperti Foundry, yang memungkinkan pengujian ditulis dalam bahasa Solidity itu sendiri dan dieksekusi pada jaringan EVM virtual lokal tanpa memerlukan biaya gas maupun konfirmasi blok riil, sekaligus mendukung pengujian berbasis properti (*property-based testing* atau *fuzz testing*) yang menghasilkan ribuan kombinasi input acak untuk memvalidasi *invariant* kontrak secara otomatis. Pendekatan ini bersifat *white-box*, karena pengujian memiliki akses penuh terhadap struktur internal dan *state* kontrak yang diuji.

Selain pengujian pada level kontrak, evaluasi *DApp* secara menyeluruh memerlukan pengujian integrasi *black-box* yang memverifikasi seluruh rantai komponen dari antarmuka pengguna, lapisan *backend*, hingga transaksi *on-chain* sungguhan, karena keberhasilan eksekusi pada level kontrak secara terisolasi tidak menjamin keberhasilan fungsional sistem secara keseluruhan, sebagaimana dibuktikan oleh temuan cacat integrasi pada penelitian ini yang hanya terungkap melalui pengujian antarmuka sungguhan. Evaluasi kegunaan (*usability* evaluation) turut menjadi komponen esensial pengujian *DApp* yang ditujukan bagi pengguna non-teknis, mengingat kompleksitas konseptual yang melekat pada teknologi *blockchain* berpotensi menjadi hambatan adopsi tersendiri terlepas dari kebenaran fungsional sistem, sebagaimana diukur menggunakan instrumen System *Usability* Scale pada penelitian ini [15], [16].

Penggunaan Foundry pada penelitian ini dapat diilustrasikan seperti uji tabrak kendaraan bermotor di laboratorium sebelum dipasarkan kepada konsumen. Sebelum sistem Payana digunakan oleh perusahaan sungguhan, *smart contract*-nya diuji ribuan kali dengan kombinasi angka gaji dan skenario acak, termasuk kondisi ekstrem seperti gaji bernilai nol maupun pengajuan kasbon yang melebihi batas. Pendekatan ini memungkinkan cacat ditemukan lebih dahulu pada tahap pengujian, bukan pada saat karyawan sungguhan telah menggunakan sistem tersebut. Prinsip pengujian preventif semacam ini menjadi semakin krusial pada pengembangan *DApp*, mengingat sifat *smart contract* yang immutable setelah di-*deploy* ke *blockchain*.

2.3.22. Privy

Privy adalah penyedia infrastruktur *embedded wallet* yang memungkinkan aplikasi menyediakan *login* dengan metode familiar seperti email atau media sosial, sekaligus secara otomatis membuat dan mengelola *wallet non-custodial* di baliknya tanpa pengguna perlu memahami konsep kunci privat atau *seed phrase* [28].
Penerapan Privy pada penelitian ini digunakan sebagai lapisan autentikasi dan penyedia *wallet* bagi seluruh aktor sistem, mencakup karyawan, HR Admin, dan Owner SaaS. Pada saat karyawan memasukkan alamat surel dan kode OTP untuk

pertama kalinya, Privy secara otomatis membuat *wallet non-custodial* di latar belakang dan mengaitkannya dengan akun tersebut. Proses ini membuat pengalaman *login* terasa identik dengan aplikasi Web2 konvensional, tanpa pengguna perlu memahami konsep *wallet* maupun kunci privat. Pendekatan ini menjadi salah satu faktor kunci di balik tingginya skor kegunaan yang diperoleh penelitian ini, sebagaimana akan diuraikan pada Bab V.

2.3.23. Pimlico

Pimlico adalah penyedia infrastruktur ERC-4337 yang menjalankan komponen *Bundler*, yaitu layanan yang mengumpulkan *UserOperation* dari berbagai pengguna dan menyampaikannya ke kontrak *EntryPoint* untuk dieksekusi di *blockchain* [29]. Penerapan *Bundler* dari Pimlico pada penelitian ini digunakan untuk memproses setiap *UserOperation* yang dihasilkan saat karyawan melakukan klaim gaji secara *gasless*. Pimlico menerima *UserOperation* dari *backend* Payana, memverifikasinya melalui *EntryPoint*, kemudian menyampaikannya ke jaringan Base Sepolia sebagai transaksi yang telah disponsori *Paymaster*. Seluruh proses ini berlangsung sepenuhnya di latar belakang, tanpa karyawan perlu berinteraksi langsung dengan infrastruktur *Bundler* tersebut. Ketergantungan pada penyedia infrastruktur pihak ketiga semacam ini turut menjadi pertimbangan dalam keterbatasan penelitian yang diuraikan pada Bab V.

Komponen *Paymaster* pada arsitektur ERC-4337 memungkinkan pihak ketiga, dalam konteks penelitian ini adalah operator platform Payana, untuk mensponsori biaya gas atas nama pengguna, sehingga karyawan dapat melakukan klaim gaji tanpa perlu memiliki saldo Ether maupun memahami konsep *gas fee* sama sekali. Mekanisme ini, dikombinasikan dengan komponen *Bundler* yang mengumpulkan dan menyampaikan *UserOperation* ke *EntryPoint*, menjadi fondasi teknis utama yang memungkinkan pengalaman "zero *gas fee*" yang menjadi salah satu proposisi nilai inti penelitian ini sebagaimana dirumuskan pada Subbab 1.1.

2.3.24. Teknologi Pendukung Implementasi

Selain teknologi inti yang telah dijelaskan pada subbab-subbab sebelumnya,

penelitian ini menggunakan sejumlah *framework* dan tool pendukung pada lapisan *backend* dan *frontend*. Next.js adalah *framework* React yang menyediakan App Router, *server-side rendering*, dan optimasi bawaan untuk aplikasi web modern [33]. Express.js adalah *framework* minimalis untuk membangun REST API di atas *Node*.js, digunakan sebagai fondasi *backend* penelitian ini [34]. Drizzle adalah *object-relational mapping* (ORM) ringan bertipe TypeScript-native yang menjembatani kode aplikasi dengan basis data relasional tanpa mengorbankan kontrol atas kueri SQL yang dihasilkan [35]. PostgreSQL adalah sistem manajemen basis data relasional sumber terbuka yang digunakan untuk menyimpan data *off-chain* aplikasi maupun data hasil indeksasi Ponder [36].

TypeScript adalah superset dari JavaScript yang menambahkan sistem tipe statis, digunakan di seluruh lapisan *backend* dan *frontend* penelitian ini untuk mengurangi kesalahan runtime melalui pemeriksaan tipe saat kompilasi [37]. React adalah pustaka JavaScript untuk membangun antarmuka pengguna berbasis komponen, menjadi fondasi Next.js pada lapisan *frontend* [38]. Tailwind CSS adalah *framework* CSS *utility-first* yang memungkinkan styling antarmuka langsung melalui class HTML tanpa menulis berkas CSS terpisah [39]. shadcn/ui adalah koleksi komponen antarmuka yang dapat disalin langsung ke dalam basis kode (*copy-paste*), bukan diinstal sebagai dependency tertutup, sehingga setiap komponen dapat disesuaikan penuh dengan kebutuhan desain aplikasi [40].

Kombinasi Next.js, React, Tailwind CSS, dan shadcn/ui digunakan untuk membangun tiga portal terpisah bagi Karyawan, HR Admin, dan Owner SaaS, dengan komponen antarmuka yang konsisten namun tetap dapat disesuaikan pada setiap peran. Express.js, Drizzle, dan PostgreSQL, seluruhnya ditulis dalam TypeScript, membentuk *backend* yang menangani autentikasi, *relay* transaksi *gasless*, dan pelaporan kepatuhan. Pemilihan teknologi berbasis TypeScript pada kedua sisi ini memungkinkan tipe data yang konsisten antara *frontend* dan *backend*, mengurangi risiko kesalahan integrasi. Kombinasi tools tersebut dipilih karena kematangan ekosistem dan dukungan komunitas yang luas, sesuai kebutuhan pengembangan skripsi dengan sumber daya terbatas.