

Chapter 2: System Analysis

2.1. Feasibility Analysis

Feasibility analysis is a critical step in the development process, assessing whether the proposed solution is practical and achievable within current technological, economic, operational, and legal constraints. In the context of this project, the analysis explores whether the system can be built effectively using available resources, tools, and methods. It considers the technical infrastructure needed to support real-time web scraping and machine learning integration, the cost efficiency of using open-source libraries, the ease of use for end users, and compliance with ethical and legal standards. A favourable outcome across all feasibility dimensions justifies the continuation and further investment into system development.

2.1.1. Technical Feasibility

The project involves building a web application that scrapes product data from Indonesian e-commerce platforms specifically Tokopedia and Blibli, it integrates machine learning techniques to process and filter the retrieved data based on user input, providing tailored product recommendations.

- **Data Acquisition:** Product information is acquired through web scraping technique utilizing available APIs. The system parsed and retrieved essential attributes, including product names, pricing data, customers ratings, image URLs, and direct link to product page.
- **Data Processing:** The data is processed to enhance accuracy and relevance. Product names are aligned with user search queries using the FuzzyWuzzy library, which implements Levenshtein Distance for string similarity. Prices are standardized using the MinMaxScaler from scikit-learn to normalize numerical values across datasets.
- **Filtering Mechanism:** Users specify a target budget along with a tolerance percentage. Based on these inputs, the system calculates an acceptable price range. For instance, a budget of IDR 50,000 with a 20% tolerance results in a range of IDR 40,000 to IDR 60,000. The product list is then filtered to include only items within this range, and the number of displayed results is limited according to the user-defined maximum.
- **System Architecture:** This application is built entirely with Python. Data retrieval and processing are handled via backend scripts, while the frontend

is implemented using Streamlit, a lightweight and responsive Python framework ideal for prototyping data applications. As data is processed in-memory, there is no reliance on traditional backends like Flask or Django and external databases.

- **Scalability:** Although the application is not currently planned for deployment, it is designed to handle larger datasets efficiently through asynchronous scraping and API rate-limiting. Its modular structure allows future scalability if needed.

2.1.2. Economic Feasibility

- **Development Costs:** The use of open-source tools and libraries such as Python, Streamlit, FuzzyWuzzy, and MinMaxScaler from scikit-learn, significantly reduces the cost of development. This project doesn't require any commercial software or paid licenses, making it highly cost-effective.
- **Operational Costs:** Since the application runs locally and processes data in-memory, ongoing operational expenses are minimal. There is currently no need for cloud hosting, database services, or dedicated servers.
- **Revenue Potential:** While the system is presently a non-commercial tool aimed at enhancing user purchasing decisions and aiding market analysis, it holds future potential for monetization. Revenue channels could include advertising, paid features for real-time alerts, or partnerships with e-commerce platforms.

2.1.3. Operational Feasibility

- **User Interaction:** The system offers an intuitive interface via Streamlit. Users can input product names, specify their budget and tolerance level, and define the number of desired results. In response, the system presents a filtered list of relevant products that match the criteria. This functionality aims to simplify product comparison and assist users in making informed choices.
- **Maintenance:** Regular updates and maintenance may be required to ensure the system remains functional and accurate, especially as e-commerce platforms frequently update their webpage structures or modify their API endpoints. Failure to adapt to these changes could lead to broken scraping logic or incomplete data retrieval.

2.1.4. Legal and Ethical Feasibility

- **Legal Compliance:** The system adheres to responsible scraping practices. It respects the terms of service of target e-commerce platforms and avoids actions that may violate their usage policies.
- **Privacy and Ethics:** The application does not collect or retain any personal user information, ensuring adherence to privacy standards and ethical guidelines in software design.

Conclusion: The system demonstrates strong feasibility from technical, economic, operational, and legal standpoints. It leverages cost-efficient technologies, offers a user-friendly interface, and is built with adaptability and scalability in mind while respecting ethical and legal constraints.

2.2. Requirement Analysis

Requirement analysis is a foundational stage in the system development life cycle that focuses on understanding what the system should do and how users will interact with it. This section outlines the functional requirements through business process modelling and detailed use case descriptions. It illustrates how the system supports users in searching for products, filtering results, analysing data, and exporting findings. The analysis aims to capture both user needs and system capabilities, ensuring that the platform is intuitive, efficient, and scalable. This clarity of requirements helps reduce development risks and provides a roadmap for future enhancements.

2.2.1. System Business Process Analysis

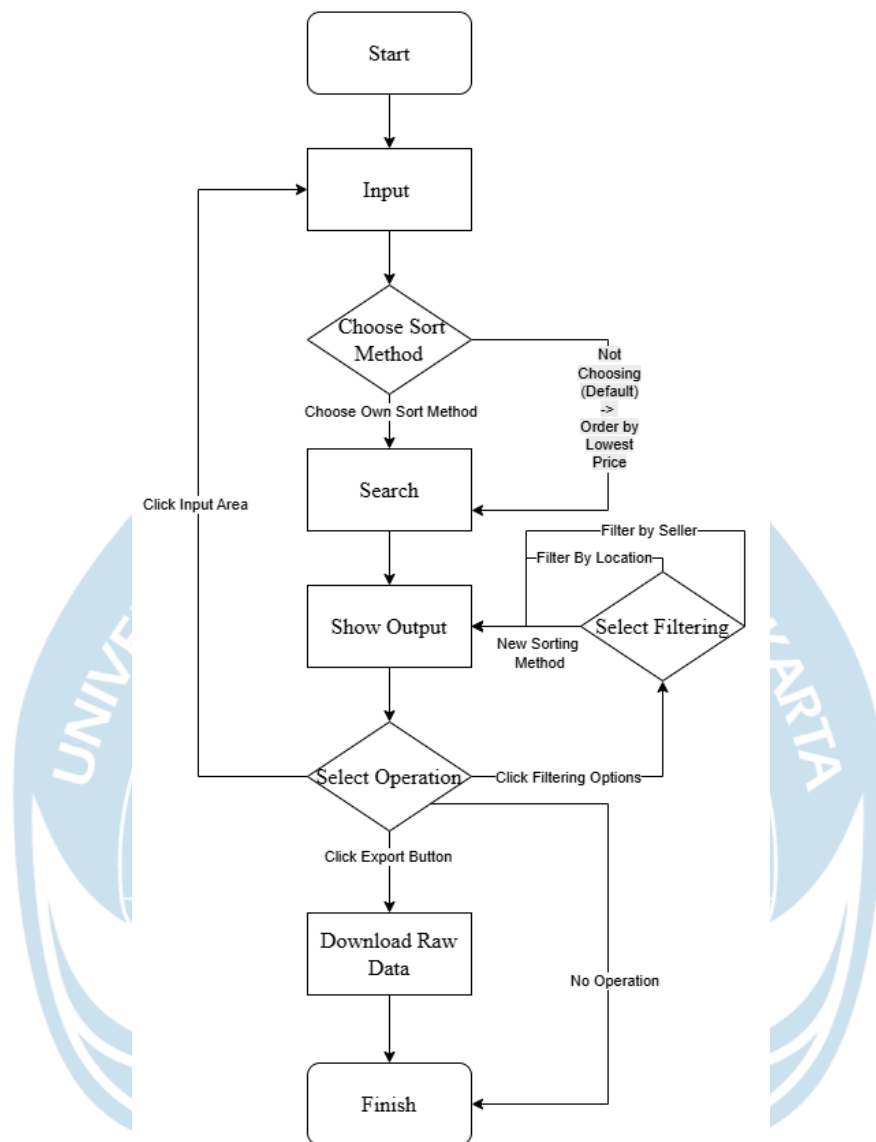


Figure 2 – 1 System Flow Chart

Figure 2 – 1 illustrates the complete workflow of the system designed to streamline product discovery and decision-making for users. The process begins with user input, where the user specifies five key parameters:

- Product Name, the item users wish to search for.
- Target Budget, users ideal price point for the product.
- Price Tolerance Percentage, percentage deviation allowed from the target budget to accommodate price flexibility.
- Maximum Number of Results, the limit on how many products the system should display.
- Sorting Method, the algorithm to sort result based on what users selects.

The sorting method can be selected based on one or more criteria, including lowest or highest price, sales performance, customers ratings, or a combination of these factors. This flexible input mechanism allows users to tailor the search process according to their specific preferences and shopping goals. Additionally, users can specify constraints such as the desired number of product listings to display, helping to focus the search and improve the relevance of the results.

Once the input is submitted, the system retrieves product data in real-time from selected e-commerce platforms, Tokopedia and Blibli. The data is then processed by filtering out irrelevant entries, ranking them according to the selected criteria, and presenting them in a clear and comparative format for easy analysis. The results are displayed in an organized interface, highlighting essential product details such as name, price, rating, and sales count.

Users can further interact with the displayed results by dynamically adjusting the sorting algorithm, applying additional filters such as seller name or seller location, or initiating a new search with revised criteria. For greater usability, the system provides the option to export raw, structured data into a downloadable CSV file for further offline analysis or recordkeeping. Additionally, each product listing includes a direct link to the original product page, enabling users to seamlessly proceed to the respective e-commerce platform for more information or to complete a purchase. This end-to-end workflow enhances the user's shopping experience by delivering personalized, accurate, and actionable product recommendations.

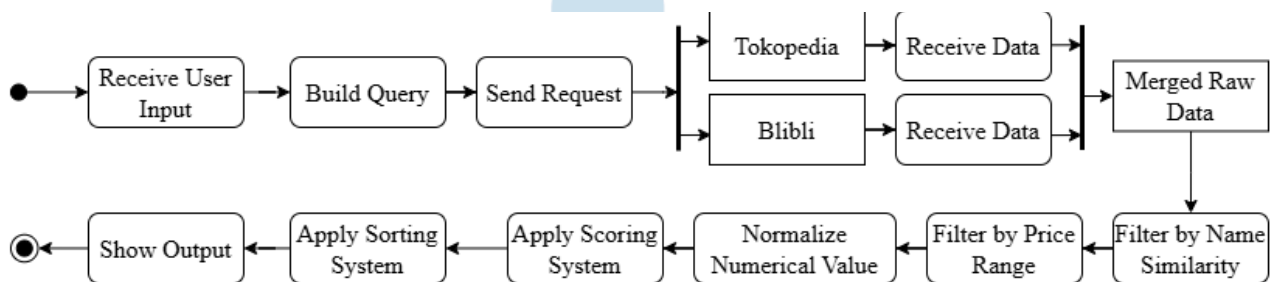


Figure 2 – 2 System Module Activity Diagram

Figure 2 – 2 illustrates the backend workflow of the system that is triggered immediately after a user submits a search query. The process begins by capturing the user's input parameters such as product name, target budget, price tolerance, maximum number of results, and selected sorting preferences. A search query is constructed with the product name as the key parameter. This query is then used to send requests to

selected e-commerce platforms, Tokopedia and Blibli. Each of these platforms is accessed using APIs. The returned raw product data is then collected, merged, and structured into a unified format DataFrame to ensure consistency across platforms for further processing.

Once the data is merged, the system performs a series of preprocessing data. First, fuzzy string matching is applied to the product names to evaluate their textual similarity with the original user query. This ensures that only highly relevant results are retained, even if product titles vary slightly in wording. Products with low similarity scores are discarded. The next step involves filtering out products whose prices fall outside the acceptable range defined by the user's target budget and tolerance percentage. After filtering, all numerical values such as price, sales performance, and customer ratings are normalized using min max normalization, converting the values into a standardized scale between 0 and 1. This normalized data is then scored using a weighted scoring algorithm tailored to the user's sorting preference, whether the users prioritizing price, sales, ratings, or a combination thereof. Finally, the ranked results are returned to the frontend, where they are presented in a clear, interactive, and sortable format for the user to explore and analyse.

2.2.2. Requirement Model Building



Figure 2 – 3 User Case Diagram

The proposed system is a web-based platform designed to assist users in discovering, analysing, and comparing product information across multiple Indonesian e-commerce platforms through a unified and intuitive interface. It addresses the growing need for consumers to make informed purchasing decisions in an increasingly saturated online marketplace, where prices, ratings, and product listings can vary significantly between platforms. By integrating data from various

sources into a single view, the system eliminates the need for manual comparison and repetitive browsing across different sites.

In this system, the user serves as the central actor, engaging with the platform through three primary use cases: searching for products, viewing and interacting with the search results, and exporting the collected data for further analysis. Each of these use cases reflects a distinct part of the user journey, supporting efficient product discovery and decision-making. The following specification tables provide detailed descriptions of each use case, outlining their processes, conditions, and requirements.

Table 2 – 1 Search Item Specification Table

ID	1	Name	Search Item	Priority	High
Participants	Users, the goal it to complete the searching process without missing any required input				
Trigger Condition	User opens the web interface and initiates a search by inputting a product name or other filters.				
Precondition	-				
Postcondition	The system returns a list of products from multiple e-commerce sites based on the input criteria.				
Normal Process	1. User inputs search term such as product name they want to search, budget, price range, sorting method and maximum shown output. 2. System validates inputs. 3. System performs web scraping. 4. Results are returned and displayed.				
Extend Process	-				
Business Rules	Required fields must be provided. Inputs must be sanitized to prevent errors or scraping issues.				
Special Needs	Handle network or scraping failures gracefully. Validate all user inputs.				

Table 2 – 2 View Result Specification Table

ID	2	Name	View Result	Priority	High
Participants	Users, the goal is to explore the retrieved products efficiently, with optional sorting and filtering to help decision-making.				
Trigger Condition	Product search is completed and results are displayed				
Precondition	The system has already performed a search and has data to display.				
Postcondition	The user has viewed or interacted with the result list and optionally navigated to a product page for more details or decision-making.				
Normal Process	1. The system displays search results in a scrollable list. 2. The user browses through the results. 3. The user may click a result to view more information, redirecting to the product page.				
Extend Process	1. User filters seller by location or name. 2. User changes sorting method, for example: from lowest price to highest sales.				
Business Rules	Sorting and filtering must apply only to currently displayed data. Redirects to external product pages must open in a new tab or window to maintain user session.				
Special Needs	Handle cases where no results are found. Interface should support responsive interaction and be optimized for performance when loading large result sets.				

Table 2 – 3 Export Raw Data Specification Table

ID	3	Name	Export Raw Data	Priority	Medium
Participants	Users, the goal is to download the result data for offline use, analysis, or backup.				
Trigger Condition	The user decides to export the data after viewing the results.				
Precondition	A valid set of search results must be available.				
Postcondition	A CSV file containing the raw data is successfully downloaded.				
Normal Process	1. The user clicks the "Export Data" button. 2. The system prepares the data in CSV format. 3. The system initiates a file download for the user.				

Extend Process	-
Business Rules	Export should include visible fields such as product name, price, rating, seller, and link. File must be UTF-8 encoded.
Special Needs	Ensure the export process is fast and works across different browsers. File should be properly named and timestamped to avoid confusion.

2.2.3. Functional Requirements

The web-based product recommendation application is designed to simplify the online shopping experience by efficiently collecting and analysing real-time product data from multiple Indonesian e-commerce platforms. The core functional requirements of the system are as follows:

- **Product Search:** The system must allow users to input product names into a search bar. The search process should include fuzzy string-matching algorithms to handle variations in product names and provide the most relevant results.
- **Price and Tolerance Configuration:** Users can define their target and specify a tolerance range in percentage. Based on these inputs, the system will filter and display products within the specified price range from (100% - n%) up to (100% + n%), ensuring that users can find products that match their budget.
- **Adjust Output Limitations:** Users can define their desired output so the system will only shows based on how many search results users want to see.
- **Real-time Data Collection:** The system uses APIs from e-commerce platforms such as Tokopedia and Blibli to scrape and aggregate real-time product data. This includes product names, prices, sales, customer ratings, seller information, product link, and image link.
- **Seamless Search Interaction:** After a user performs an initial product search, the system allows them to instantly conduct a new search by inputting new product name, budget, price range, sorting type, and desired number of results within the same interface, without navigating away or refreshing the page.
- **Data Export:** The system provides users with the option to export raw scraped product data into CSV format for further analysis. This ensures

transparency and allows users to perform detailed comparisons beyond the recommendations.

- **Filtering and Sorting Options:** Users can filter product results by specific attributes such as:
 - **Price:** Displaying products sorted from lowest to highest price.
 - **Sales:** Displaying products sorted by highest sales volume.
 - **Ratings:** Displaying products sorted by highest customer ratings.
 - **Combination of criteria:** Combining two or more criteria with a weighted score to generate recommendations. The system applies a pre-determined weight for each criteria.
- **Machine Learning Integration:** A recommendation engine uses machine learning to rank products based on factors like price and sales performance. The system combines price and sales data with user-defined preferences to generate personalized product recommendations

2.2.4. Non-Functional Requirements

In addition to the functional specifications, the system must meet the following non-functional requirements to ensure performance, reliability, and usability:

- **Performance and Scalability:** The system should be able to handle large amounts of data in real time, ensuring minimal latency in data retrieval and processing.
- **Usability:** The user interface (UI) should be intuitive, easy to navigate, and responsive. The product search, configuration settings, and filtering options should be straightforward, ensuring users can quickly access the information they need.
- **Data Accuracy:** The system must provide accurate and up-to-date product data. Since data is collected through APIs, the application should ensure that it handles discrepancies or data gaps effectively, presenting accurate product details.
- **Reliability:** The system must ensure high availability, with minimal downtime during data retrieval and processing. Regular maintenance and updates should be implemented to address issues such as API changes or data structure updates on the e-commerce platforms.

2.3. Technical Approach Analysis

To develop an effective and efficient product comparison system, several technical approaches were evaluated based on criteria such as development speed, ease of maintenance, scalability, and support for data-centric operations including web scraping and real-time processing. This section presents a comparative analysis of these approaches to justify the selected technology stack and to consider future scalability options.

Table 2 – 4 Technical Approach Comparison Table

Option	Name	Advantages	Disadvantages
1	SSM (Spring + Spring MVC + MyBatis) with MySQL	• Provides a robust, scalable architecture ideal for enterprise-grade applications. • Mature Java ecosystem with extensive documentation and community support. • MyBatis offers fine-grained control over database interactions.	• Complex setup involving significant configuration and boilerplate code. • Longer development time, making it less suitable for rapid prototyping. • Java backend may introduce a steeper learning curve and additional overhead for non-enterprise users.
2	Spring Boot + Thymeleaf + MySQL	• Simplifies traditional Java frameworks through auto-configuration. • Better frontend-backend integration using Thymeleaf templating. • More efficient for moderately complex applications compared to traditional SSM.	• Still relatively heavy compared to Python-based alternatives. • Java ecosystem is not optimized for quick data scraping or experimental development.
3	Python + Streamlit + DataFrame	• Well-suited for data science and scraping tasks due to	• DataFrames are volatile and in-memory, lacking persistent long-term storage

		Python's rich library ecosystem. • Enables rapid development cycles and lower complexity, ideal for prototyping. • Streamlit provides an intuitive, interactive UI with minimal coding. • DataFrames support powerful and flexible in-memory data manipulation and querying.	without database integration. • Does not inherently provide the structured architecture and robustness expected in enterprise applications. • Limited scalability for large datasets that exceed available memory.
--	--	---	--

Conclusion:

Considering the project requirements for a lightweight design, rapid development, and interactive product comparison, the Python-based stack leveraging Streamlit and DataFrames stands out as the most practical and efficient solution. Unlike traditional Java frameworks, which excel in large-scale, enterprise-level applications but demand considerable setup and configuration, the Python approach offers a streamlined workflow tailored for data-centric tasks such as web scraping, preprocessing, and real-time visualization.

Streamlit facilitates the creation of highly interactive, visually appealing user interfaces with minimal code, enabling rapid iteration and immediate feedback during the prototyping phase. This advantage is particularly valuable in research and academic settings, where evolving requirements necessitate flexibility to experiment with various data processing or recommendation algorithms. Moreover, Python's extensive ecosystem of libraries including pandas for data manipulation, scikit-learn for machine learning, and requests for web scraping provides robust support for implementing advanced features without the overhead of integrating multiple languages or complex frameworks.

In summary, the chosen Python and Streamlit approach not only accelerates development and reduces complexity but also ensures that the system remains flexible, maintainable, and accessible. This makes it an ideal foundation for both current project

goals and future enhancements, supporting ongoing innovation in data-driven product comparison and recommendation systems. ethical and legal constraints.

2.4. Chapter Summary

The system analysis establishes the feasibility, requirements, and technical approach for developing a real-time, data-driven product comparison platform. By leveraging Python, Streamlit, and open-source libraries, the proposed system ensures efficient data acquisition, processing, and user interaction, while maintaining scalability, cost-effectiveness, and compliance with ethical standards.

