JIGE 7 (3) (2026) 2875-2897
JURNAL ILMIAH GLOBAL EDUCATION
ejournal.nusantaraglobal.ac.id/index.php/jige
DOI: https://doi.org/10.55681/jige.v7i3.6721

Design and Build a Web-Based State Asset Monitoring Dashboard System at the State Wealth and Auction Office (KPKNL) Using the Laravel Framework

Cantika Berliana Rahmasari[1], Rizky Parlika[1]
[1.] Faculty of Computer Science, National Development University "Veteran" East Java
**Corresponding author email: cantikaberlianarahmasarii@gmail.com, rizkyparlika.if@upnjatim.ac.id*

Article Info

Article history:

Received June 19, 2026
Approved August 20, 2026

Keywords:

Laravel, UI/UX, Figma, Blackbox Testing, Function Point

ABSTRACT

So that the supervision of state assets at the KPKNL is still manual. As a result, the data available for decision-making is not always up-to-date, and reporting is time-consuming, not to mention a lack of consistency in record-keeping. Using the waterfall development method, this study developed a web-based dashboard system to address these problems using Laravel and Bootstrap. The interface is designed in Figma, starting from three core pages (Home, Login, and Dashboard), and expanded to 13 functional pages. Available features include real-time asset monitoring, interactive asset charts and tables, asset photo uploads, dropdown filters, report export to PDF and Excel, user management based on RBAC, asset location maps, and user FAQs. To validate functionality, black box testing is used, while to measure system quality, CFP and RCAF Function Point approaches are used. The UI/UX evaluation is conducted by interviewing users, the assessment that the interface is easy to understand, the flow is clear, and the appearance is considered attractive but does not detract from functionality.

How to cite: Rahmasari, C. B., & Parlika, R. (2026). Design and Build a Web-Based State Asset Monitoring Dashboard System at the State Wealth and Auction Office (KPKNL) Using the Laravel Framework. *Jurnal Ilmiah Global Education*, *7*(3), 2875–2897. https://doi.org/10.55681/jige.v7i3.6721

INTRODUCTION

To government agencies, public demands for faster, more transparent, and more accountable public services continue to increase along with the rapid development of information technology. Society expects public institutions to provide services that are not only accurate and reliable but also accessible in a timely manner through digital platforms. This condition encourages government agencies to continuously improve and modernize their information systems in order to support effective governance and improve service quality. The implementation of digital technology in public administration is therefore not merely an innovation but has become a necessity to ensure efficiency, transparency, and accountability in government operations (Rahayu et al., 2023).

KPKNL, as an implementing unit under the Directorate General of State Assets (DJKN), holds a strategic role in managing and supervising state-owned assets that represent significant economic value. The management of these assets requires comprehensive, accurate, and up-to-date information to support administrative processes and decision-making activities. In this

context, transparency and accountability become essential aspects of asset governance. Without proper monitoring and reporting mechanisms, the management of state assets may encounter various challenges that affect data quality, reporting accuracy, and institutional accountability. Consequently, the availability of an integrated information system becomes increasingly important to support effective asset management and strengthen public trust in government institutions (Fahrezi et al., 2022).

Interviews conducted with users at KPKNL provide an overview of several issues encountered in the existing system. These problems are not limited to a single aspect but encompass both technical and operational dimensions. From a technical perspective, users reported system bugs during data entry activities that occasionally disrupt the recording process. In addition, the absence of a feature for uploading asset photographs limits the completeness of asset documentation. Visual documentation is important because it provides supporting evidence regarding the physical condition of assets and facilitates verification processes when needed. The lack of such functionality reduces the effectiveness of asset monitoring and may lead to difficulties in validating asset information.

From an operational perspective, users indicated that several work processes remain time-consuming, particularly those involving the input of asset values and location information. Repetitive manual data entry increases the workload of employees and may contribute to data entry errors. Furthermore, notification mechanisms that are not optimally designed can interrupt users' workflow and reduce productivity during daily operations. Another issue highlighted by users is the absence of visual representations of asset conditions within the dashboard. Without visual summaries and graphical information, users must spend additional time reviewing detailed records to obtain an overall understanding of asset status. These limitations collectively affect the speed of reporting, reduce the accuracy of information management, and make it difficult for organizational leaders to obtain timely information required for decision-making processes (Hendartie et al., 2023; Febriyanti & Kesiman, 2021).

To address these challenges, Laravel was selected as the primary framework for system development. The selection of Laravel was based on several considerations related to maintainability, scalability, and development efficiency. Laravel adopts the Model-View-Controller (MVC) architecture, which separates application logic, user interface components, and data management functions into distinct layers. This architectural approach enables developers to organize source code more systematically, simplifies maintenance activities, and facilitates future system enhancements. The clear separation of responsibilities within the application structure also contributes to improved code readability and development productivity (Pratama, 2021; Dewi et al., 2023).

In addition to its architectural advantages, Laravel provides a variety of built-in features that support the development of modern web applications. The Eloquent Object Relational Mapping (ORM) feature simplifies database interactions and allows developers to manage data efficiently without writing extensive SQL queries. The Blade template engine facilitates the creation of dynamic and reusable user interface components, thereby improving consistency throughout the system. Laravel also includes an authentication mechanism that supports secure user access management. These features contribute to faster development processes while ensuring that the resulting application remains secure, maintainable, and aligned with organizational requirements (Uminingsih & Ichsanudin, 2022).

Several previous studies have examined the implementation of web-based information systems in various organizational environments. However, existing studies generally focus on basic asset recording and management functions and have not extensively incorporated real-time data visualization features. The availability of real-time visualization is particularly important because it enables users and decision-makers to monitor asset conditions more efficiently and identify issues promptly. The lack of visual analytics in previous systems limits the ability of organizations to transform data into actionable information that can support strategic planning and operational control (Emanuella et al., 2022).

Research involving the use of Laravel has also demonstrated positive outcomes in the development of information systems, including warehouse management and inventory-related

applications. These studies indicate that Laravel can support the creation of efficient and reliable web-based systems. Nevertheless, most implementations have been conducted within private-sector or non-governmental environments. As a result, there remains an opportunity to apply Laravel in the context of government asset management, particularly in institutions that require high levels of accountability, data security, and reporting accuracy (Febriana, 2022).

Within the context of KPKNL, no integrated platform has been identified that combines multiple essential functionalities in a single system. Existing solutions generally address only specific aspects of asset management and do not comprehensively integrate dashboard-based monitoring, role-based access control (RBAC), asset condition visualization by division, asset photo uploads, asset location mapping, and frequently asked questions (FAQ) features. The absence of such integration may lead to fragmented information management and reduced operational efficiency. Therefore, the development of a comprehensive asset monitoring dashboard is expected to address these limitations and provide a more effective solution for asset management activities (Hardiansyah et al., 2022).

The system interface is designed using Figma as a prototyping and interface design tool. The use of Figma enables the creation of visual designs that can be evaluated and refined before implementation. The design process adopts a user-oriented approach, emphasizing usability and ensuring that system interactions align with users' daily activities and expectations. Rather than focusing solely on visual aesthetics, the interface design aims to facilitate intuitive navigation, reduce user effort, and support efficient task completion. By prioritizing user experience, the system is expected to achieve higher levels of acceptance and usability among its intended users (Putra et al., 2021; Santoso, 2024).

The development process follows the waterfall methodology because of its structured and systematic nature. Each phase, including requirements analysis, system design, implementation, testing, and deployment, is clearly defined and documented. This sequential approach facilitates project management and ensures that development activities proceed according to established objectives. Comprehensive documentation produced throughout the development lifecycle also supports future maintenance and system enhancement activities. To evaluate system functionality, Black Box Testing is employed to verify whether each feature performs according to specified requirements. In addition, the Function Point method is utilized to estimate system complexity and provide an objective measure of the application's functional size (Nurmaharani, 2023).

Based on these considerations, this study develops a web-based state asset monitoring dashboard system for KPKNL that integrates data visualization, role-based access control, asset location mapping, asset documentation, and reporting features within a single platform. The system is developed using the Laravel framework and supported by a user-oriented interface designed with Figma. Through the integration of these components, the proposed system is expected to improve monitoring effectiveness, support decision-making processes, enhance reporting efficiency, and contribute to more accountable and transparent management of state assets (Rifai et al., 2023).

METHOD

This research uses the waterfall software development method, which is carried out sequentially and systematically (Prasetyo, 2023. The overall research flow is depicted in the flowchart in Figure 1.

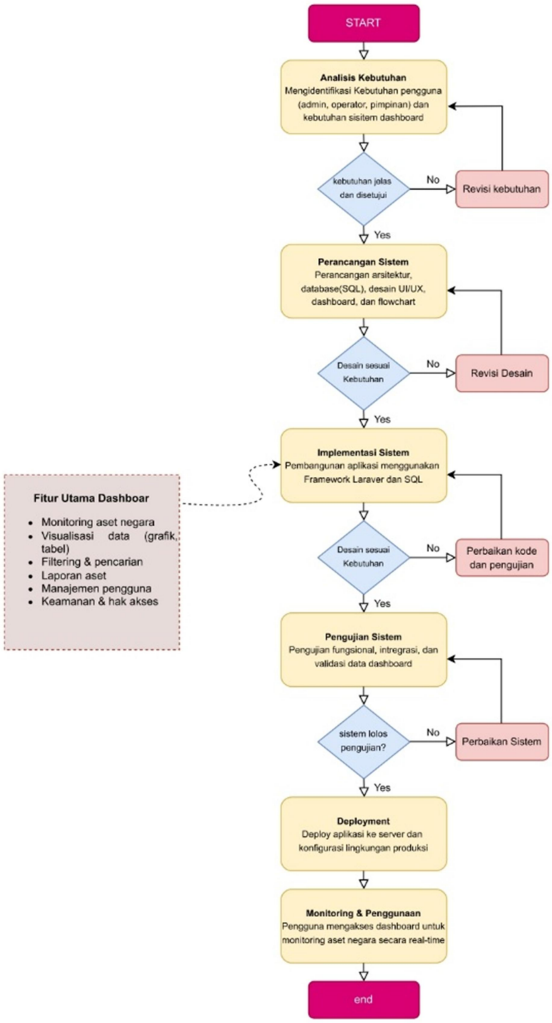


Figure 1. Research Flow Flowchart

The stages of the research in Figure 1 include five main phases that are carried out sequentially:

Needs Analysis

The needs analysis was carried out through three groups of direct users, namely admins, operators, and KPKNL leaders. Data were collected using interviews and observations and compiled into a Software Requirements Specification (SKPL) document (Endra et al., 2021). Requirements that are still unclear or have not been approved are revised before development continues (Khasbulloh & Karim, 2023).

System Planning

System architecture, databases (CDM and PDM), user interfaces designed using Figma, and UML diagrams including Use Case, Activity, Class, Statechart, Sequence, and Collaboration Diagrams are included in the design stage (Desma & Harry, 2022). Designs that do not meet user needs are subsequently revised before implementation.

System Implementation

System development was carried out using the Laravel framework and MySQL database, starting from dashboard development and core application modules (Syafitri et al., 2023). Several key features implemented include real-time asset monitoring, visualization of asset condition data by division, asset photo uploads, asset location maps, dropdown filters, detailed reports in PDF and Excel formats, role-based access control (RBAC) user management, and FAQ and help

features (Muhammad et al., 2022). If bugs or errors are identified during implementation, code corrections and retesting are performed.

System Testing

Black box testing was employed to verify that each system feature functions according to the specified requirements (Firmansyah & Herman, 2023). The Function Point method was used to evaluate software quality and estimate system complexity. Systems that fail to meet testing criteria are revised and tested again.

Deployment

The deployment stage includes installing the application on a production server and configuring the production environment so that users can access the system for real-time state asset monitoring (Febriani et al., 2023).

RESULT AND DISCUSSION

This section presents the results of the research process, beginning with user requirement identification, literature review, UML design development, and UI/UX design using Figma as the primary output of the system design phase.

User Requirement

Interviews were conducted directly with users at KPKNL, including three groups: administrators who manage the system, operators who input and update asset data daily, and leaders who require concise information to support decision-making. A total of 15 respondents consisting of staff members and organizational leaders completed questionnaires to validate the interview findings. The results indicated that 93% of respondents considered a real-time monitoring dashboard to be the most critical system requirement (Siburian & Latifah, 2023).

The interviews also revealed more specific requirements. Users expressed the need to input data and upload asset photographs directly through the website without complicated procedures. Information regarding asset conditions in each division, whether active or damaged, should be presented graphically within the dashboard. Detailed reports based on asset condition and asset type are also required. Furthermore, respondents proposed the inclusion of asset location maps, dropdown-based navigation, FAQ features, user guides, and help-contact information (Rofi & Amalia, 2022). These requirements are summarized in the functional requirements listed in Table 1.

Table 1. List of User Requirements for the KPKNL Monitoring Dashboard System

Yes	Requirements	Description	UI/UX Implementation (Figma)
1	Login & Authentication	The user logs in to the system using a username and password. Each role, namely admin, operator, and leader, has different access rights according to the assigned role.	Login Page (Figure 7): two-column layout, the left column displays the visual branding of KPKNL Surabaya (Ministry of Finance logo, tagline, city illustration), right column contains the username form, password, and Login button.
2	Dashboard Monitoring	Displays a real-time summary of country asset data in the form of card metrics, graphs, and interactive tables.	Main Dashboard Page (Figure 8): a two-panel layout with a dark blue sidebar containing 6 navigation menus, the content area displays the Total Assets, Active Assets, Damaged Assets card metrics, as well as the Visualization, Asset Report, and Add Assets shortcuts.
3	Asset Management	Admins can add, change, delete, and view asset data along with their categories, locations, and statuses, including uploading photos of assets directly through the website.	Asset Monitoring Page (Figure 11): a table with Asset Name, Category, Location, Condition, and Status columns, with a search feature. Active Asset Table Visualization Page (Figure 13): additional Asset Images fields for upload and visual identification.
4	Data Visualization	Asset data is displayed in bar charts, pie/doughnut charts, and	Asset Status Chart Visualization Page (Figure 12): Total Assets, Active Assets,

		interactive tables, including the number of assets per condition (damaged/active) per division.	Damaged Assets, Asset Status Percentage Chart chart, and Recent Assets table (category, date, status).
5	Filters & Search	Users can filter data by division, asset type, condition, and year through dropdown.	Filtering Page (Figure 14): two-column design with 6 dropdown parameters: Asset Category, Asset Status, Tool Condition, Location, Date of Acquisition, and Asset Value, as well as the Search button in the bottom right corner.
6	Asset Report	Operators can print asset reports in PDF or Excel format based on specific conditions, asset types, and periods.	Asset Report page (Figure 15): form with Asset Name, Description, Location, Date of Acquisition, and Asset Value fields, with an Export PDF button in the upper right corner.
7	User Management	Admin manages user accounts: add, modify, delete, and set RBAC-based roles.	User Management Page (Figure 17): form to change account data with Password, Username, and PIN fields, as well as the Change button as the main action.
8	Security & Access Rights	RBAC ensures that each user only accesses the features that are appropriate to their role. Admins have full access, Operators manage asset data, Leaders can only read data.	Integrated on the Login Page (Figure 7) for authentication validation and the User Management Page (Figure 17) for role settings.
9	Asset Location Map	Interactive maps display the physical location of assets on the monitoring page to make it easier to identify the geographical distribution of assets.	Integrated on the Asset Monitoring Page (Figure 11) and the Asset Description Page (Figure 16) which display location details side by side with asset photos.
10	FAQ & Help	Provide FAQs, usage procedures, and help contacts so that users can operate the system independently.	FAQ page (Figure 18): the interactive accordion format contains 3 common questions that can be clicked to display the answers.

From the results of the analysis in Table 1, ten main functional needs were successfully identified. Five of them are high-priority Login & Authentication, Dashboard Monitoring, Asset Management, User Management, and Security & Access Rights which are the main basis for the development of the system. Medium priority needs include Data Visualization, Filters & Search, and Asset Reports. Asset Location Maps and FAQ & Assistance are categorized as low priority but are still implemented at the request of users from the interview results (Negoro, 2023).

Literature Studies

The literature search was carried out with two objectives: to develop the theoretical foundation of the research as well as to recognize approaches that are relevant to the problem at hand. The sources used as references are Sinta indexed journals 1, 2, and 3, as well as international journals from IEEE and ScienceDirect, with a limit on the year of publication starting in 2021. The activities carried out in this study include web-based monitoring systems, Laravel frameworks, UI/UX design using Figma, government asset management, black box testing, measurement using Function Points, and software quality testing (Parlika, 2023).

A number of reoccurring problems in the literature were investigated. It is dominating study of Web-based information system, its MVC structure makes it easier to manage the code when the system starts growing and its built-in security features are satisfactory for the needs of the app in an instance. The ability to make many iterations on the design without having to begin from scratch, and for team members to collaborate on the design makes Figma a popular protocol choice for UI/UX design research. When talking about functional testing, blackbox testing is

most frequently cited; when you don't have to understand the code you simply need to see what the system does when you send it what you're asking it to do. The interactive dashboard is sometimes referred to as the solution, because it is easy to read the data at the management level. The term waterfall is widely used in the public sector literature, and is not as prevalent in other sectors, because projects initiated by governments typically have a closed project scope when the procurement process starts. Table 2. Summary of the literature.

Table 2. Summary of Literature Study

Yes	Year	Author	Key Issues	Solution	Technology
1	2021	Haryuda et al. [11]	The store does not have a prototype UI/UX design for the website	Web-based UI/UX design using the Design Thinking method	Figma, Design Thinking
2	2023	Resorts [13]	The sales system does not yet have an interface that meets the needs of the user	Analyze and design UI/UX sales applications with Design Thinking	Figma, Design Thinking
3	2024	Santo [12]	Web developers lack understanding of interactive UI/UX concepts and techniques	Implementation of UI/UX concepts and techniques in web layout design using Figma	Figma, UI/UX
4	2021	Endra et al. [16]	There is no systematic comparison between Laravel and PHP Native	Analysis of the comparison of Laravel with PHP Native in website development	Laravel, PHP Native
5	2022	Desma & Harry [18]	Hijab sales application is not yet web-based	Construction of web-based sales applications using Laravel and Bootstrap	Laravel, Bootstrap
6	2023	Khasbulloh & Karim [17]	Admission of new students is still manual	Design and build web-based information systems using Laravel	Laravel, MySQL
7	2022	Fahrezi et al. [2]	Testing of the inventory system of goods has not been systematic	Implementation of Black Box Testing on web-based inventory applications	Black Box Testing
8	2022	Resorts [9]	Testing of employee attendance system not yet comprehensive	Blackbox testing with the Top 10 OWASP Attack method	Black Box, OWASP
9	2021	Primary [5]	Software testing has not yet used standardized methods	Blackbox Testing analysis and UML modeling on web-based applications	Black Box, UML
10	2023	Parlika et al. [27]	Software quality has not been quantitatively measured	Quality measurement using Function Point	Function Point, CFP, RCAF

Table 2 summarizes the ten literature reviewed. The most striking similarities were in the choice of technology: Laravel, MySQL, and blackbox testing appeared in most studies, although the methods of application vary. Of the ten literatures, the two most cited in this study are studies on measuring software quality using Function Points, and studies that discuss the limitations and advantages of blackbox testing specifically (Parlika, 2023).

Design (Design)

UML / Diagram Design

The UML notation used for system design is six diagrams (Pangaribuan, 2021). Use Case Diagram One: Stores 3 actors, namely Admin, Operator, and Leader. Users, assets, system configurations are all in the hands of the Admin. Operator operating at the daily level: Inputting data about assets, uploading photos, printing daily reports. Access to it is restricted to reading, with focus on dashboard and real-time reports.

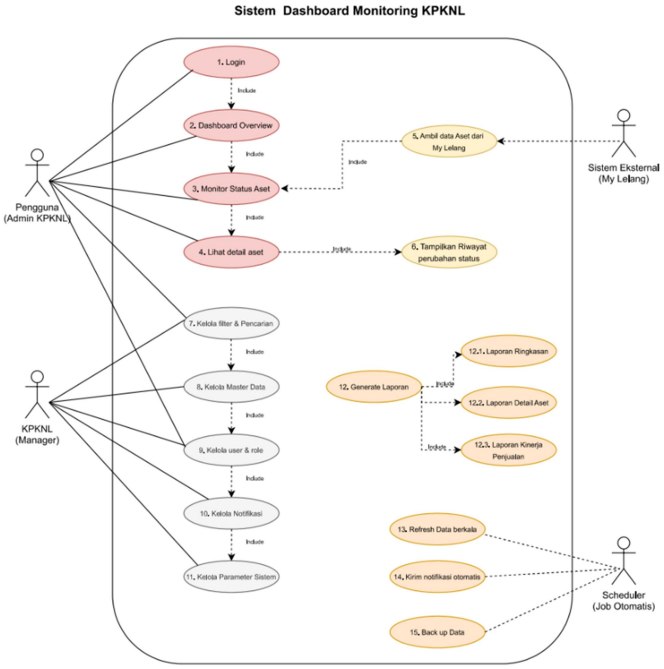
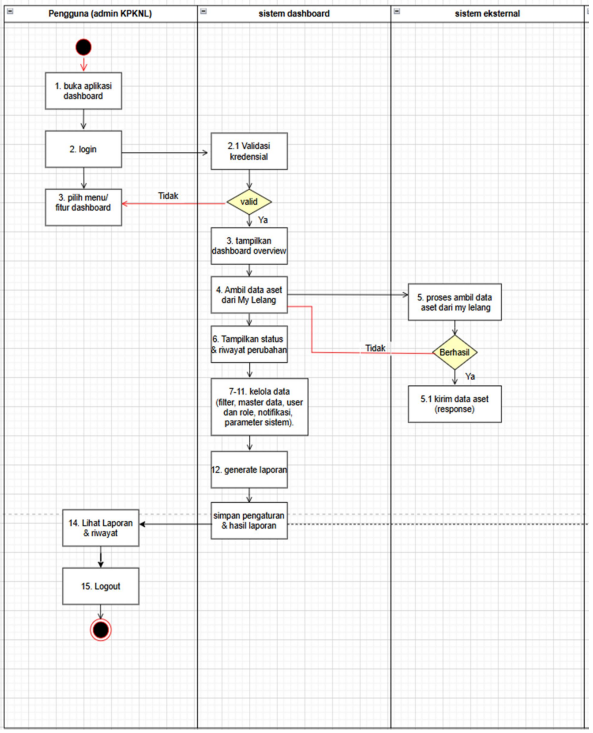


Figure 2. Use Case Diagram of KPKNL Monitoring Dashboard System

The Activity Diagram describes the workflow of the processes related to the main business processes in the system, from the user authentication process, asset data management, photo uploads, creation of detailed reports, the use of asset location maps, to the setting of user access rights. This diagram helps to confirm that all the business logic to be implemented is clearly defined and there is no ambiguity before the implementation is carried out.



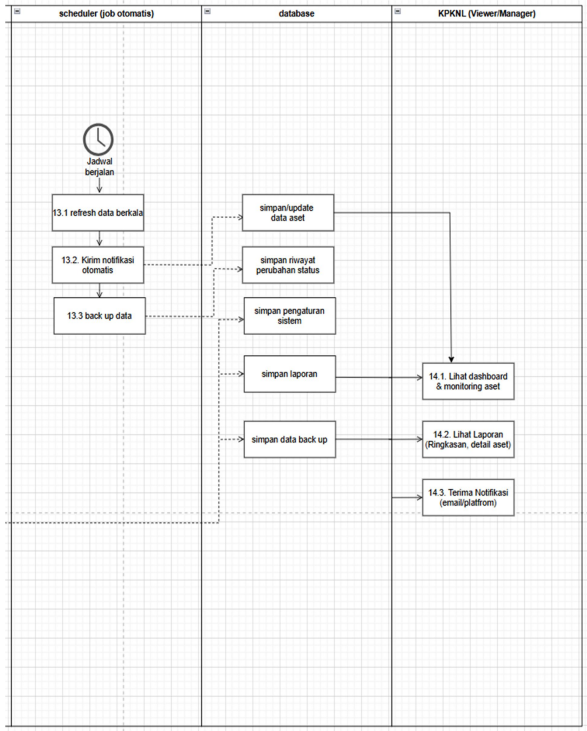


Figure 3. Activity Diagram Main Process Dashboard System

Class Diagram is used to represent the static structure of the system, which includes the main classes of the system, their attributes, operations and the relationships between the classes. The main classes contained in the system include User (id_user, name, email, password, id_role), Role (id_role, nama_role), Asset (id_aset, nama_aset, kode_aset, id_kategori, id_lokasi, value, year, condition, status, foto_path), Category (id_kategori, nama_kategori), Location (id_lokasi, nama_lokasi, address, latitude, longitude), and Report (id_laporan, date, id_user, type, file_path).

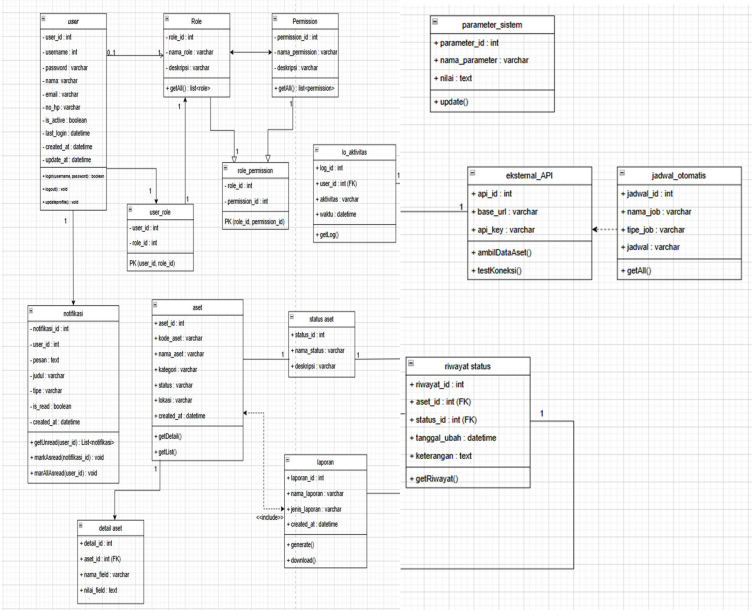


Figure 4. KPKNL Monitoring Dashboard System Diagram Class

The Statechart Diagram in Figure 5 consists of three stages of the asset state flow that are differentiated by color: Preparation (yellow), Execution (blue) and Completion (green). The pipeline starts from the Asset Data Input to the Validation Process. Data that does not pass is returned for correction and revalidation; that passes is immediately saved and the state data moves to the Published state. Entering the implementation stage, the system carries out Asset Monitoring continuously. If there is a change in the state, the data is updated by using the state Upate Data Asset command before proceeding with the monitoring. The system moves to Finished Asset Status once the asset is finished being used.

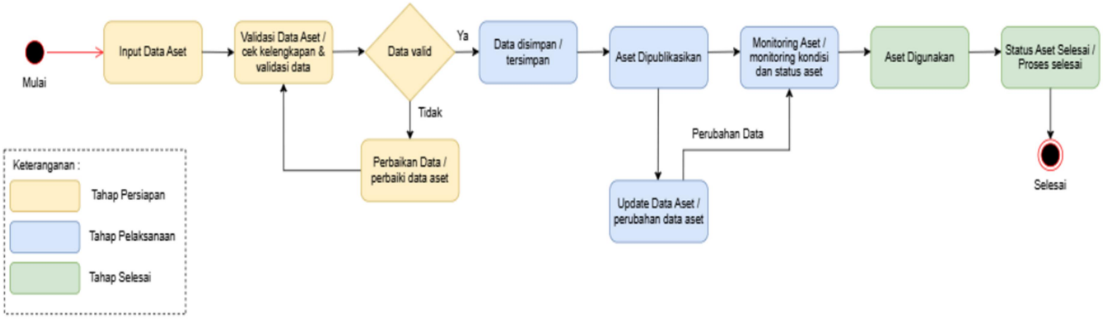
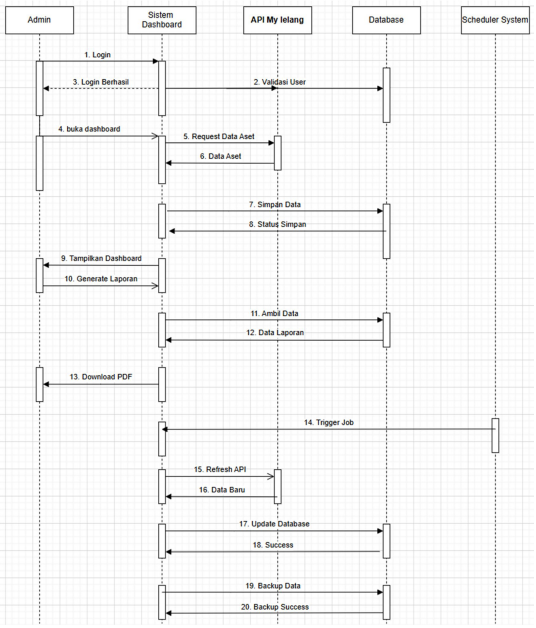


Figure 5. Statechart State Asset Status Diagram

Figure 6 shows the sequence of interactions between the five lifelines, namely Admin, Dashboard System, My Auction API, Database, and Scheduler System, which are shown in the 20 message steps. Flow: Login and Check User using DB, then get data assets from API My Lelang and store it on DB and display on Admin Dashboard. Processes covered in the next scenario include generating the report, downloading to PDF and refreshing the data automatically via trigger job, API, update database, and finally backing up the data through the Scheduler System.

automatic data updates triggered by the Scheduler System through the trigger job mechanism, API refresh, database update, and ended with a data backup process.



.**Figure 6.** Squence Diagram Main Process System Dashboard

The Collaboration Diagram in Figure 7 illustrates the pattern of structural collaboration between five main objects, namely Admin, Dashboard System, Database, My Auction API, and Scheduler System through 14 ordered messages. The Dashboard System object will be a data communication center that connects all other objects both to authenticate asset data from the My Auction API, input data to the Database, create PDF reports for Admins, and update scheduled asset data triggered by the Scheduler System.

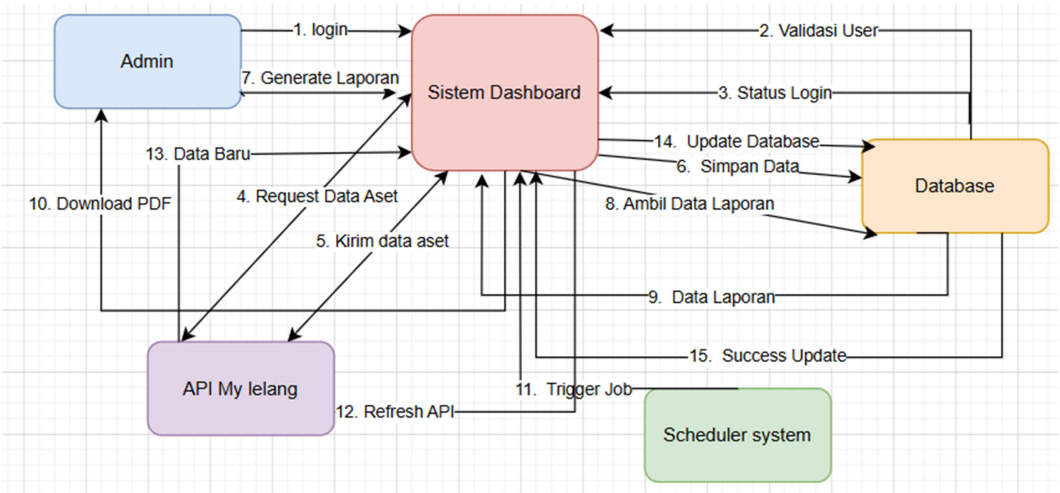


Figure 7. Collaboration Diagram Main Process Dashboard System

UI/UX Designing Using Figma

The UI/UX design is created in Figma, an interactive prototype. The process uses the principles of user-centered design and usability guidelines, and the results of the user requirement stage become a direct reference (Rafida, 2022). An iterative process was performed to trace the design to specific needs generated in the interviews (Selay, 2023).

The system is designed with 13 pages, following the navigation flow: Home → Login → Dashboard → functional sub-pages. Home is the first page that the user will see before going into the authentication process. Once logged on, Dashboard serves as the entry point to all features for asset management and monitoring. The design of each page is presented in the Summary file of components and design concepts.

Table 3. Components and Design Concepts of Each Page in Figma

Title	References	Main Components	Design Description
Home Page Design (Landing Page)	Figure 8	Hero image, Ministry of Finance logo, tagline, START button, agency value icon	Visual storytelling-based welcome page with illustrations of the KPKNL Surabaya building. The dominant color is sky blue with gold accents to reinforce the institutional identity. The START button is the only call-to-action that directs the user to the Login page.
Login	Figure 9	Form username & password, Login button, KPKNL branding, illustration of the city of Surabaya	Two-column layout: the left column displays the institutional identity (Ministry of Finance logo, KPKNL Surabaya text, tagline, city illustrations), the right column contains the authentication form. Designed as simple as possible so that users can instantly find forms without disorientation.
Main Dashboard	Figure 10	Sidebar navigation, greeting users, card metrics, shortcut icons main features	Two-panel layout: a dark blue sidebar (#1A237E) contains 6 main menus, a gray background content area (#F5F5F5) displays card metrics of Total Assets, Active Assets,

			Damaged Assets, as well as the shortcuts Visualization, Asset Report, and Add Assets.
Number of Active Assets	Figure 11	Active asset list table, Asset Name, Category, Location, Condition, pagination	A page that lists all assets in an active state in the form of a structured table. A pagination feature is included to make it easy to navigate large amounts of data.
Total Assets	Figure 12	Asset summary table, Asset Name column, Category, Condition, Amount	A page of the total recapitulation of the country's assets listed in the system. It is designed so that leaders can get a comprehensive overview of all assets managed by KPKNL in one view.
Asset Monitoring	Figure 13	Monitoring table, Asset Name column, Category, Location, Condition, Status, search feature	A real-time monitoring page of asset condition and status. The search feature is placed at the top of the page to make it easier for operators to quickly find specific asset data.
Asset Status Chart Visualization	Figure 14	Card metrics, asset status donut chart, Recent Assets table (category, date, status)	A visualization page that presents a summary of the condition of the asset in the form of card metrics and a donut graph of the percentage of the status of the asset. The Latest Assets table on the right side allows leaders to make data-driven decisions quickly.
Visualization of the Active Asset Table	Figure 15	Detailed active asset table, Asset Image column, Asset Location, photo upload feature	An advanced view of the visualization that includes the Asset Image and Asset Location columns. The asset photo upload feature is integrated directly into this view for visual identification of assets.
Filtering	Figure 16	6 filter dropdown (Category, Status, Condition, Location, Date, Value), Search button	A symmetrical two-column design with six dropdown-based search parameters. The Search button is placed in the lower-right corner as the only action, following the principle of single visual focus.
Asset Report	Figure 17	Report input form (Asset Name, Description, Location, Date, Value), PDF Export button	A manual report creation page that provides a structured input form. The Export PDF button in the upper right corner allows operators to download reports directly in PDF format.
Asset Description (Filtering Details)	Figure 18	Asset photos, data details (Name, Category, Status, Location, Date, Value)	A filtering detail page that combines visual elements of a photo with textual data side by side. Users can verify the physical condition and administrative data of assets at once in a single view.
User Management	Figure 19	Change account form (Password, Username, NIP), Change button	Minimalist design with a single button Change as the main action, according to the principle of single focus. It can only be accessed by Admins to ensure the security of the management of user account data.
FAQ & Help	Figure 20	Interactive accordion, 3 frequently asked questions (add assets, filtering, print reports)	The help page is in a clickable accordion format to display answers. Allows users to find the system usage guide independently without technical assistance.

From Table 3 it appears that each page is designed with a specific purpose and complementary components. The Home → Login → Dashboard flow reflects a natural and predictable user workflow, so users don't have to learn the system from scratch.

The design writing of the Home page uses a visual storytelling approach with illustrations of the KPKNL Surabaya building as the main element. This illustration features the icons of the city of Surabaya (Suramadu bridge, city skyline) to reinforce the geographical context of the

agency. The dominant color used is bright sky blue with gold accents in the emphasis text, in line with the visual identity of the Ministry of Finance of the Republic of Indonesia as the parent agency of KPKNL. Using visual focus, the START Button is set in the middle of the page as the only call-to-action that draws the user to the Login page.

Figure 8. Home Page Design (Landing Page)

Figure 8 shows that the Home page has succeeded in building a professional first impression and reflects the institutional identity of KPKNL. Five icons of agency values (Professional, Accountable, Synergy, Transparent, Innovative) are displayed at the bottom of the page as a reinforcement of the image of public services. The results of user interview evaluations stated that the Home display was quite good and attractive, and gave the impression of a system developed specifically for KPKNL Surabaya.

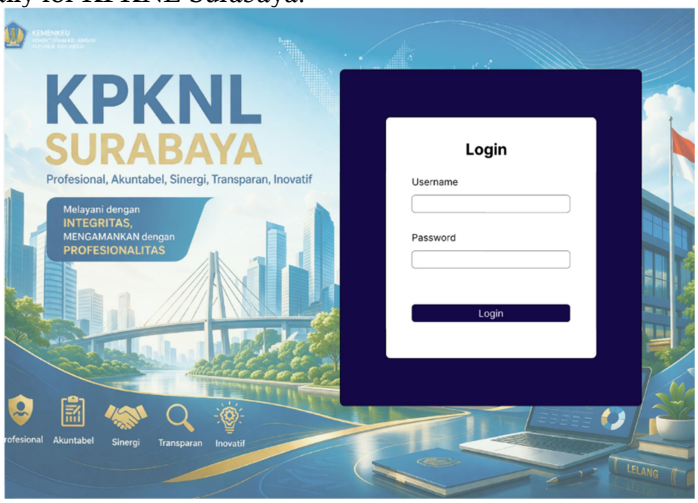


Figure 9. Login Page Design

The visual branding elements in Figure 9 are arranged on the left side and the authentication form is on the right side, using a 2-column layout. The institutional identity of KPKNL Surabaya is presented comprehensively, from the logo of the Ministry of Finance in the upper corner, the text "KPKNL SURABAYA" with a size and color that is in harmony with the visual identity of the Ministry of Finance, the tagline of agency values (Professional, Accountable, Synergy, Transparent, Innovative), and an illustration of the city of Surabaya which displays bridges and skylines of multi-storey buildings as a strengthening of the geographical context. At the bottom there are five icons of agency values that strengthen the image of public services. This visual storytelling approach aims to give the "impression" that the system was developed specifically for KPKNL Surabaya and not a generic system.

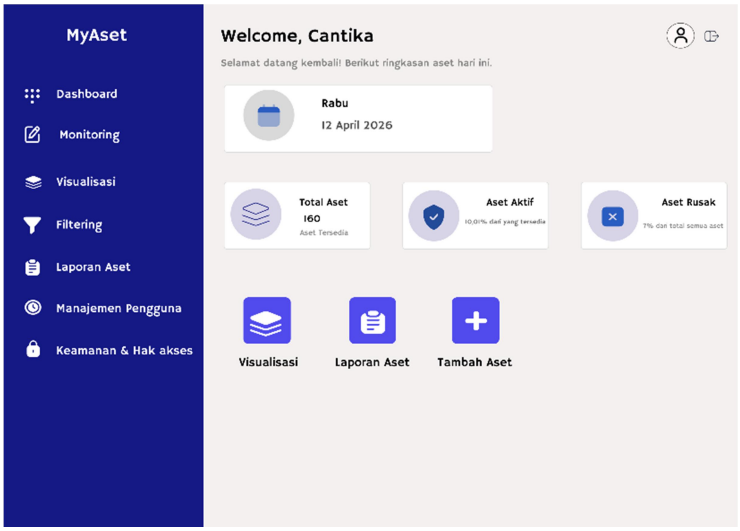


Figure 10. Main Dashboard Page Design

Figure 10 shows a Dashboard view that is designed to present asset condition data in a compact manner without sacrificing readability. Card metrics with relevant icons help users capture information at a glance, without the need to read long text. From the evaluation interview, users stated that the flow of use was clear and adaptation to the system did not take long. There are no complaints about navigation confusion. Overall, the design worked on at Figma is assessed in accordance with the functional needs and usability expectations of users at KPKNL.

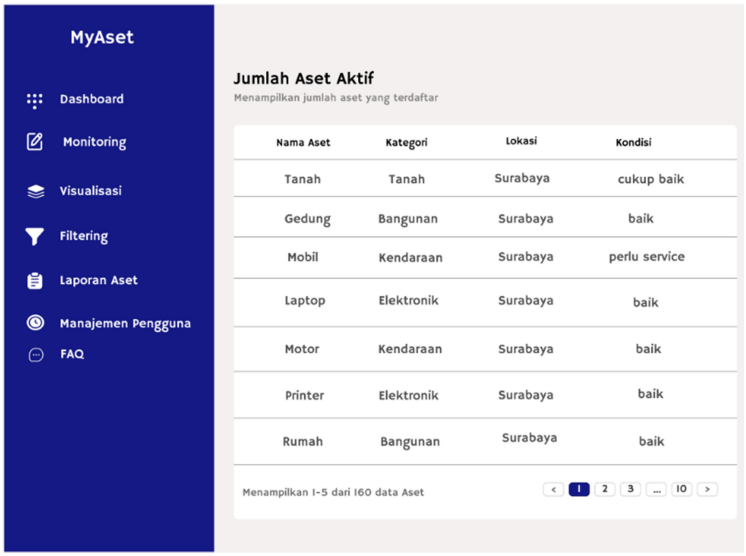


Figure 11. Number of Active Assets Page

Figure 11 shows the Number of Active Assets page which presents a list of assets with active status in the form of a table with the Asset Name, Category, Location, and Conditions columns. This page is equipped with a pagination feature to facilitate the navigation of large amounts of data, so that operators can monitor the condition of active assets in a structured manner.

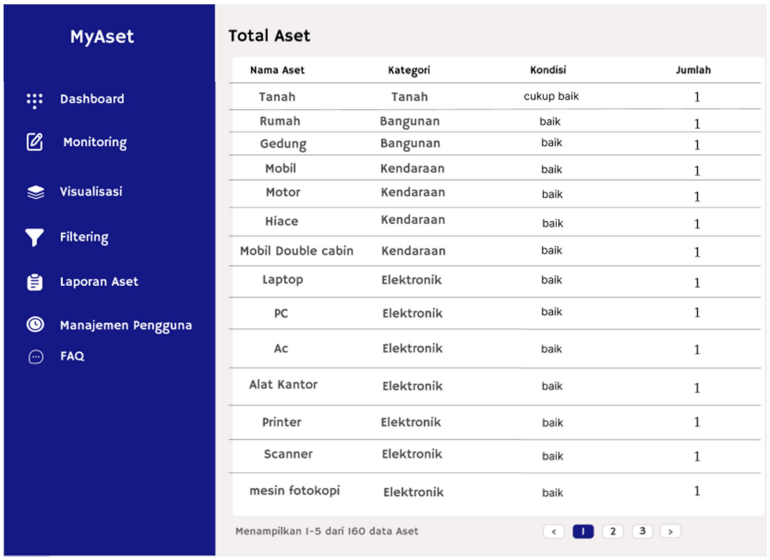


Figure 12. Total Assets Page

Figure 12 shows the Total Assets page which summarizes all the assets of the countries listed in the system. The data is presented in a table with Asset Name, Category, Condition, and Amount columns, allowing leaders to get a comprehensive overview of all assets managed by KPKNL.

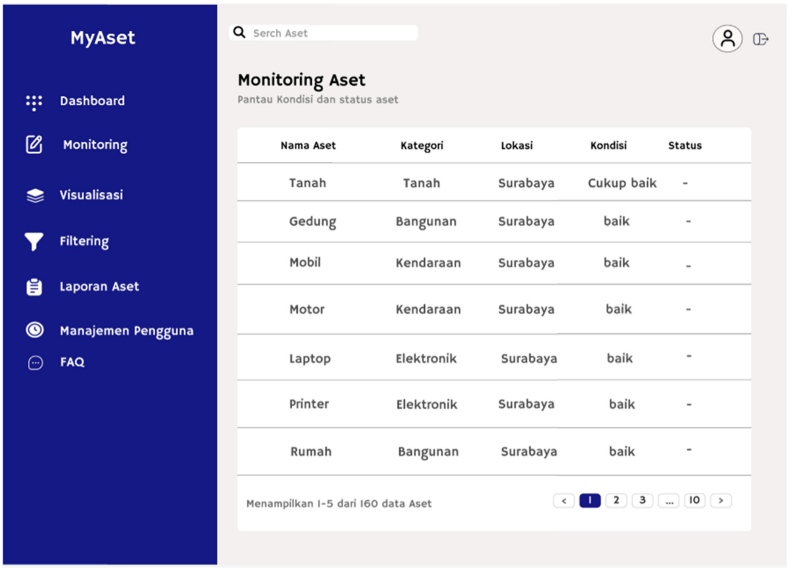


Figure 13. Asset Monitoring Page

Figure 13 shows the Asset Monitoring page which functions to monitor the condition and status of assets in real-time. The table displays the Asset Name, Category, Location, Condition, and Status columns, with a search feature at the top of the page to make it easier for operators to quickly find specific asset data.

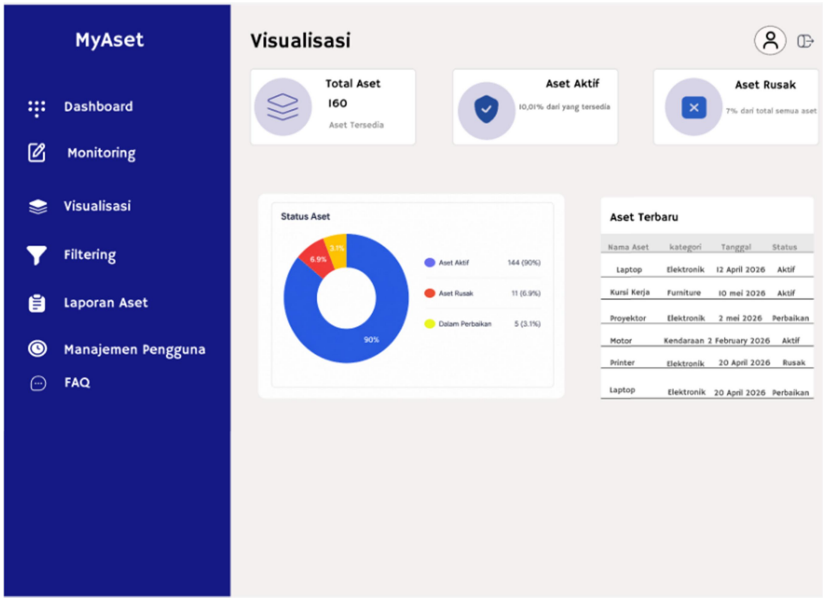


Figure 14. Asset Status Chart Visualization Page

Figure 14 shows the Visualization page that presents a summary of the asset condition in the form of card metrics (Total Assets, Active Assets, Damaged Assets) and a donut graph that visually depicts the percentage of asset status. On the right side, you can see the Latest Assets table with their categories, dates, and statuses, so leaders can make data-driven decisions quickly.

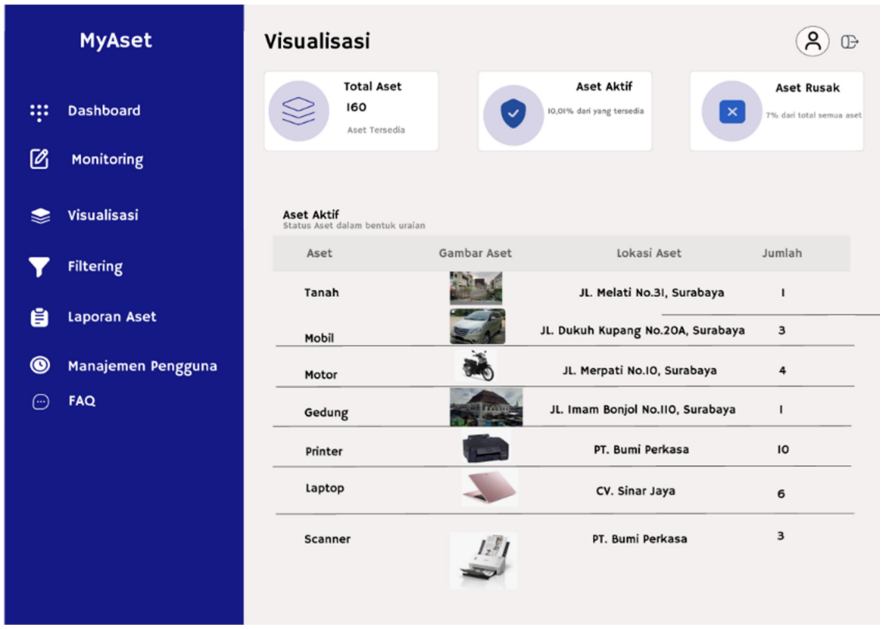


Figure 15. Active Asset Table Visualization Page with Images

Figure 15 shows an advanced view of the Visualization page that presents the details of active assets in the form of a table with the Asset Image and Asset Location columns. The asset photo upload feature is integrated directly into this view, allowing users to visually identify assets while knowing their physical location without having to open a separate page.

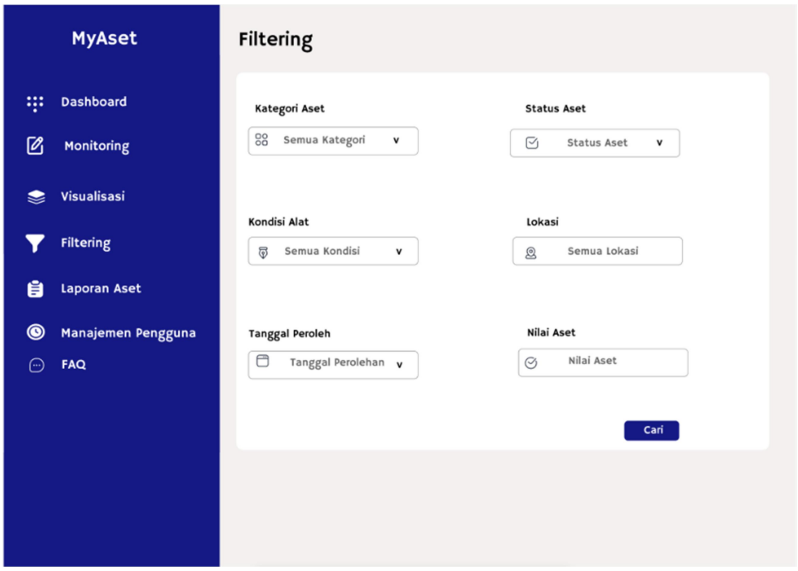


Figure 16. Page Filtering

Figure 16 shows the Filtering page that provides six dropdown-based search parameters, namely Asset Category, Asset Status, Tool Condition, Location, Date of Acquisition, and Asset Value. The symmetrical two-column design makes it easy for users to filter the asset's data specifically as needed, and the Search button is placed in the bottom right corner as the only action on this page.

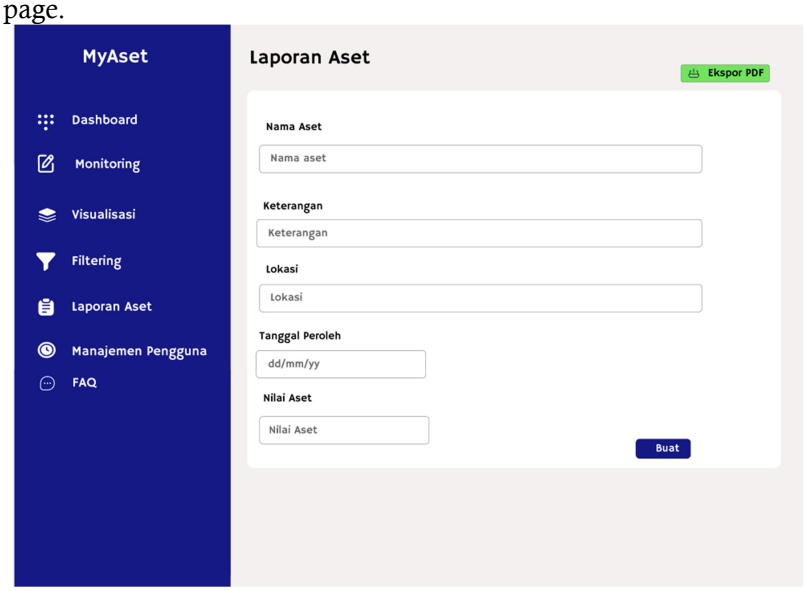


Figure 17. Asset Report Page

Figure 17 shows the Asset Report page that provides an input form for manually generating the report, including the Asset Name, Description, Location, Date of Acquisition, and Asset Value fields. There is an Export PDF button in the upper right corner that allows operators to download asset reports directly in PDF format.

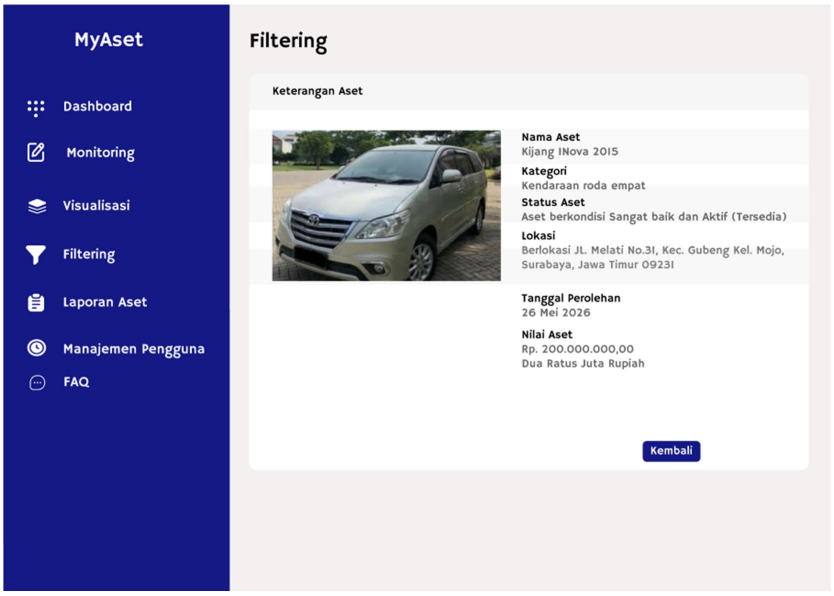


Figure 18. Asset Description Page (Filtering Details)

Figure 18 shows the filtering results detail page that displays the complete information of an asset, including the asset photo, Asset Name, Category, Asset Status, Location, Date of Acquisition, and Asset Value. This view blends the visual elements of the photo with textual data side by side, allowing users to verify the physical condition and administrative data of the asset at once.

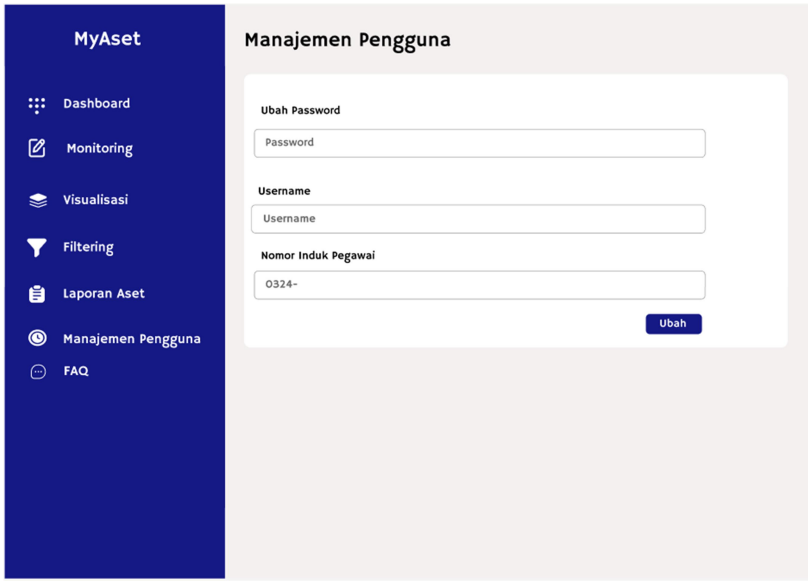


Figure 19. User Management Page

Figure 19 shows the User Management page that provides a form to change account data, including the Password, Username, and Employee Identification Number fields. The design of this page is minimalistized with a single Change button as the main action, following the principle of single focus so that users don't experience confusion when managing their account data.

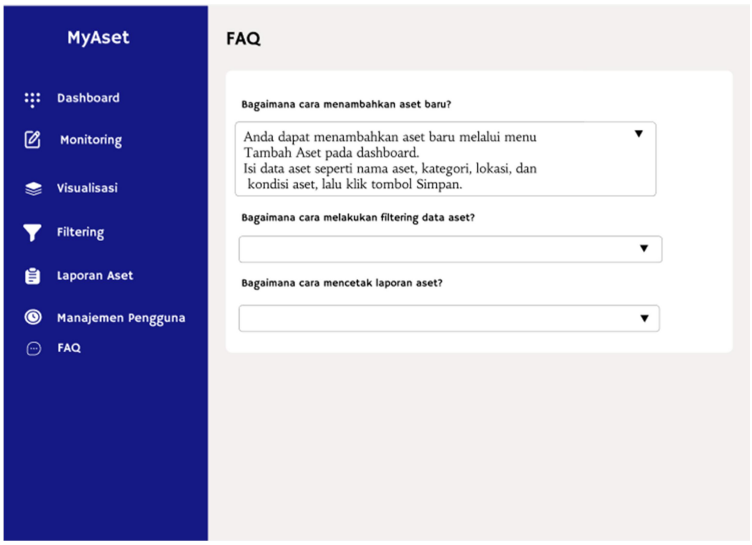


Figure 20. FAQ Page

Figure 20 shows an FAQ page that presents frequently asked questions in accordion format, where each question can be clicked to display its answer. The three questions include how to add new assets, how to filter, and how to print asset reports, so users can find the system usage guide independently without technical assistance.

Development

The system was created using the Laravel framework which is based on MVC (Model View Controller). Wakendalahku: Eloquent ORM for database queries, Blade for rendering views, and built-in authentication for Role Based Access Control (RBAC) (Desma, 2022) Displays are done using Bootstrap to generate responsive views across different screen sizes. Ready-to-use components, interactive tables, card metrics, filter dropdowns, action buttons, and simply cut development time without going far from the Figma Design (Syafitri 2023)

MySQL Database integration with Laravel using Eloquent ORM does not require much additional configuration. The tables covered by the Schema are the asset table, the user table, the category table, the location table, and the report table. The features implemented include: real-time asset monitoring, interactive donut charts and tables per division, photo uploads, dropdown filters, PDF/Excel exports, RBAC-based user management, interactive location maps, and FAQ pages (Syafitri, 2023).

Testing

The test uses the blackbox testing method, which assesses the system from its external behavior without touching the internal code (Pratama, 2021). Each feature is given a specific input and the results are compared to the pre-defined specifications. Full results are in Table 4.

Table 4. Results of the Blackbox Testing KPKNL Monitoring Dashboard System

Yes	Tested Features	Input	Expected Output	Results	Status
1	Login	Valid username and password	Successfully log in to the dashboard page according to the role	The system displays dashboards according to user roles	✓ Valid
2	Login	Incorrect username or password	Error message "Username or password incorrect"	The system displays an error notification	✓ Valid
3	Dashboard Monitoring	Users access the dashboard page	Displays card metrics Total Assets, Active Assets, Damaged	Asset data is displayed in card metrics correctly	✓ Valid

			Assets in real-time		
4	Add Asset Data	Fill out the complete asset data form with photos	The data is stored into the database and appears in the monitoring table	Asset data is successfully saved and displayed	✓ Valid
5	Upload Asset Photos	Uploading image files in JPG/PNG format	The photo is successfully saved and appears in the asset visualization table	A photo of the asset is successfully displayed on the visualization page	✓ Valid
6	Data Visualization	Users access visualization pages	Displays the latest asset condition donut chart and asset table	Graphs and tables are displayed with the corresponding data	✓ Valid
7	Filters & Search	Select the dropdown parameters (Category, Status, Condition, Location)	Displays asset data according to selected filters	The data is successfully filtered and displayed according to the parameters	✓ Valid
8	Export PDF Report	Click the Export PDF button on the report page	Successfully downloaded PDF file containing asset report data	PDF report successfully downloaded with complete data	✓ Valid
9	User Management	Admin changes account data (username, password, PIN)	User data is successfully updated in the database	Saved and confirmed account data changes	✓ Valid
10	FAQ Page	The user clicks on the question on the accordion	Answers to clicked questions appear below	The accordion functions to display the answers correctly	✓ Valid

Based on Table 4, all 10 blackbox testing scenarios produce outputs that correspond to the set specifications, so that all features are declared Valid. These results show that the KPKNL monitoring dashboard system has met all the functional needs identified at the user requirement stage (Firmansyah 2023).

Maintenance

After the system is delivered to users, development enters the maintenance phase, which represents the final stage of the waterfall development lifecycle (Hikmah et al., 2022). The primary objective of this phase is to ensure that the system continues to operate effectively, securely, and in accordance with evolving user requirements and organizational needs (Nurhalizah et al., 2024).

The maintenance strategy for the KPKNL asset monitoring dashboard consists of four main activities. First, corrective maintenance is conducted to address errors that may emerge after the system has been deployed in the production environment. Examples include failures during asset data entry, problems in uploading asset photographs, and notification features that do not function as intended. Corrective maintenance aims to restore normal system functionality and minimize disruptions to users' daily activities.

Second, adaptive maintenance is performed to maintain compatibility with technological developments and infrastructure changes. This activity includes updating Laravel, PHP, and MySQL versions, ensuring that the system remains compatible with server environments and continues to operate efficiently despite changes in supporting technologies. Through adaptive maintenance, the application can benefit from performance improvements, enhanced security mechanisms, and long-term sustainability.

Third, perfective maintenance focuses on improving system functionality and performance based on feedback from users. During system operation, users may identify opportunities for enhancement that can increase efficiency and usability. Examples of proposed improvements include automatic notifications for assets approaching maintenance schedules and integration with centralized question-and-answer services managed by DJKN. These

enhancements are intended to improve user experience and ensure that the system continues to support organizational objectives effectively.

Fourth, preventive maintenance is carried out proactively to reduce the risk of future system failures. Activities include routine database backups, security monitoring, server health checks, and updates of software dependencies before vulnerabilities can affect system operations. Preventive maintenance helps maintain system reliability, protect organizational data, and reduce the likelihood of unexpected downtime (Mely, 2024).

To ensure maintenance activities are implemented consistently, a structured schedule is recommended. System log reviews and database backup procedures should be conducted weekly to support data recovery and operational monitoring. Security patch installations and dependency updates should be performed monthly to maintain system security and software stability. In addition, feature evaluations involving users should be conducted every three months to identify emerging needs and opportunities for system improvement. All maintenance activities should be documented comprehensively so that modification histories can be tracked, audited, and used as references for future development and maintenance efforts (Chairun, 2022).

CONCLUSION

The web-based state asset monitoring dashboard system for KPKNL was designed and built using Laravel and Bootstrap with the waterfall method. Here are some things that can be concluded from the process undertaken.

The ten functional features developed all refer to the results of interviews and questionnaires of 15 KPKNL respondents. The data collected shows that 93% of respondents consider real-time monitoring dashboards to be the most urgent need. UI/UX designing in Figma results in 13 pages with Home navigation flows, → Login → Dashboard → functional sub-pages. From the evaluation interviews, users assessed that the interface was easy to understand, the flow was clear, and the appearance was in line with the Ministry of Finance's visual identity. Blackbox testing on all features did not find any output that deviated from the specifications.

For further development, three things are worth considering: automatic notifications for assets approaching maintenance schedules, integration with the central DJKN system, and a mobile version to make the system more accessible to all stakeholders in KPKNL.

REFERENCES

Chairun, I. K. D. R. (2022). Repair and maintenance monitoring information system at the mechanical division of PT. XYZ. *Jurnal Responsive Teknik Informatika, 6*(1), 49–61. Retrieved from https://ojs3.lppm-uis.org/index.php/JR/article/view/552/436

Desma, A., & Harry, W. (2022). The utilization of Laravel framework and Bootstrap framework in the development of web-based hijab sales applications. *Media Infortama, 18*(1).

Dewi, F. K. S., Adithama, S. P., & Suhardi, A. T. (2023). Testing of doctor to doctor application using the black box testing method. *CONSTELLATION: Convergence of Technology and Information System, 3*(1), 61–72. https://doi.org/10.24002/constellation.v3i1.7046

Emanuella, G., Sudirman, & Afifah. (2022). Implementation of black box testing on extraordinary websites. *Pusat Penelitian STMIK Kharisma Makassar, 17*(1), 135–148.

Endra, R. Y., Aprilinda, Y., Dharmawan, Y. Y., & Ramadhan, W. (2021). Comparative analysis of PHP Laravel programming language with native PHP in website development. *Expert: Jurnal Manajemen Sistem Informasi dan Teknologi, 11*(1), 48. https://doi.org/10.36448/expert.v11i1.2012

Fahrezi, A., Salam, F. N., Ibrahim, G. M., Rahman, R., & Saifudin, A. (2022). Black box testing on web-based goods inventory application at PT. AINO Indonesia. *OKTAL: Jurnal Ilmu Komputer dan Sains, 1*(1). Retrieved from https://journal.mediapublikasi.id/index.php/oktal/article/view/29

Febriani, E., Imran, B., & Muslim, R. (2023). E-commerce information system for selling rattan handicrafts based on website in Loang Village, Janapria District. *Journal of Computer*

Technology, 1(1). Retrieved from https://ojs.ninetyjournal.com/index.php/COMTECHNO/article/view/46

Febriana, R. (2022). Black box testing of Karawang employee attendance information system with the top 10 OWASP attack method. *Jurnal Ilmiah Wahana Pendidikan, 8*(12), 327–334.

Febriyanti, N. M. D., & Kesiman, M. W. A. (2021). Implementation of black box testing in lecturer management information systems. *JITTER: Jurnal Ilmiah Teknologi dan Komputer, 2*(3). Retrieved from https://ojs.unud.ac.id/index.php/jitter

Firmansyah, M. D., & Herman. (2023). Design and build web-based e-commerce programs in apparel stores using Laravel framework. *JISICOM: Journal of Information Systems and Computer Science*. Retrieved from https://journal.stmikjayakarta.ac.id/index.php/jisicom/article/view/1921

Hardiansyah, Hendayani, M., Herlambang, I. A., Rezki, A. N., & Saifudin, A. (2022). Implementation of black box testing in food ordering applications. *OKTAL: Jurnal Ilmu Komputer dan Sains, 1*(12). Retrieved from https://journal.mediapublikasi.id/index.php/oktal

Hendartie, S., Jayanti, S., & Sutejo, H. (2023). Testing of the new student admission application (PMB) of STMIK Palangkaraya using black box testing. *Jurnal Sains Komputer dan Teknologi Informasi, 5*(2). https://doi.org/10.33084/jsakti.v5i2.5021

Hikmah, A. B., Faqih, H., Hudin, J. M., & Ramdhani, L. S. (2022). Equipment maintenance scheduling information system using waterfall model, *10*(2), 141–145.

Khasbulloh, A., & Karim, A. A. A. (2023). Design and build a web-based new student admission information system using the Laravel framework. *SIMTEK: Jurnal Sistem Informasi dan Teknik Komputer, 8*(1), 17–23. https://doi.org/10.51876/simtek.v8i1.165

Mely, M. (2024). Application of the waterfall method in the development of production equipment schedule maintenance applications, *7*(1), 133–141.

Muhammad, A. F., Nopi, R., & Ariawan, D. R. (2022). Design and build a sales information system at Our Tasikmalaya Bookstore using the Laravel 8 framework. *Jurnal Teknik Informatika dan Sistem Informasi*.

Negoro, R. A., Triayudi, A., & Iskandar, A. (2023). Implementation of e-commerce clothing line using design thinking method and system usability scale. *Jurnal Riset Komputer, 10*(1). https://doi.org/10.30865/jurikom.v10i1.5634

Nurhalizah, R. S., Saksena, I. S., & Bambang, R. B. (2024). Implementation of preventive maintenance in website-based applications (Kostqita case study), *4*(1), 8–14.

Nurmaharani, S. (2023). Analysis and design of UI/UX sales applications. *Infotech Journal, 9*(1), 46–53.

Pangaribuan, I., Zaka, M., & Yunanto, R. (2021). Design of web-based online sales as an entrepreneurship strategy. *International Journal of Entrepreneurship and Technopreneur, 1*(1), 31–36. https://doi.org/10.34010/injetech.v1i1.5467

Parlika, R., Mahendra, R. R., Lutfi, M. R. A. R., Waritsin, R. K., & Ramadhan, H. M. T. (2023). Measuring the quality of student extracurricular data collection website software using function points. *Jurnal Ilmiah Informatika, 11*(1), 1–14. https://doi.org/10.33884/jif.v11i01.5578

Prasetyo, Y. K. D. (2023). Design and implementation of a web-based sales information system using the Laravel framework. *Jurnal Teknik Informatika dan Sistem Informasi, 10*(3), 191–202. Retrieved from https://jurnal.mdp.ac.id/index.php/jatisi/article/view/5023

Pratama, M. S. (2021). Analysis of black box testing software testing methods and UML diagram modeling in veterinary services applications developed with waterfall model. *Jurnal Teknik Informatika Kaputama, 5*(2). Retrieved from https://jurnal.kaputama.ac.id/index.php/JTIK

Putra, D. H., Asfi, M., & Fahrudin, R. (2021). UI/UX design using web-based design thinking method at Laportea company. *Jurnal Ilmiah Teknologi Informasi Terapan, 8*(1).

Rafida, V., Arfyanti, I., & Hidayat, I. (2022). Sales management application at Widya Collection Store web-based. *International Journal of Information Engineering and Electronic Business, 14*(4), 1–10. https://doi.org/10.5815/ijieeb.2022.04.01

Rahayu, W. I., Bintang, J. M., & Pramana, D. A. (2023). Implementation of the Laravel framework in the design of the Roblox programming course registration system application. *Jurnal Teknik Informatika, 15*(1), 9568–9574.

Rifai, F. D., Purwanto, E., & Nurmalitasari. (2023). Laravel-based academic information system for re-registration and submission of student leave. *INFOTECH Journal, 9*(1). https://doi.org/10.31949/infotech.v9i1.5194

Rofi, A. M., & Amalia, R. (2022). Design and build e-commerce in OS Shoe Stores with the waterfall method. *OKTAL: Jurnal Ilmu Komputer dan Sains, 1*(2), 81–90. Retrieved from https://journal.mediapublikasi.id/index.php/oktal/article/view/29

Santoso, M. F. (2024). Implementation of UI/UX concepts and techniques in web layout design with Figma. *Jurnal Teknologi dan Sistem Informasi Bisnis, 6*(2), 279–285.

Selay, A., Andgha, G. D., Alfarizi, M. A., & Wahyudi, M. I. B. (2023). Sales information systems. *ZONAsi Komputer: Program Studi Sistem Informasi Universitas Batam, 13*(3), 232–237. https://doi.org/10.37776/zkomp.v13i3.1461

Siburian, R. O., & Latifah, F. (2023). The application of the waterfall method in the design of a web-based information system at PT. Garuda Inti Sentosa to increase sales. *JISAMAR: Journal of Information System, Applied, Management, Accounting and Research, 7*(4), 972–983. https://doi.org/10.52362/jisamar.v7i4.1252

Syafitri, A., Angraeni, A., & Wibowo, A. (2023). Web-based digital library management information system using Laravel framework. *Jurnal Informatika dan Kesehatan, 2*(2). https://doi.org/10.35473/ikn.v2i2.3699

Umiringsih, M. N., & Ichsanudin, N. (2022). Functional testing of library information system software with black box testing method for beginners. *STORAGE: Jurnal Ilmiah Teknik dan Ilmu Komputer, 1*(2). Retrieved from https://storage.unsa.ac.id