Journal of Engineering, Electrical and Informatics

E-ISSN: 2809-8706
P-ISSN: 2810-0557

(Research Article)

Design of a Web Server Based Monitoring System for Aviation Security Equipment Maintenance with an Interactive Dashboard and Telegram Notification Integration at UPBU Silampari

Emmalitna Caroline Sembiring [1*], Eriyandi [2], Febria Roza [3]

[1-3] Program Studi Teknik Navigasi Udara, Politeknik Penerbangan Indonesia, Indonesia
***** Coresponding Author**:** emmalitnacaroline@gmail.com

Abstract: The Silampari Airport Operating Unit (UPBU Silampari) is a Class III airport under the Directorate General of Civil Aviation that plays a strategic Role in the South Sumatra region. In its operations, maintenance activities for aviation security equipment such as the X-Ray Baggage Scanner, Walk Through Metal Detector (WTMD), Hand Held Metal Detector (HHMD), and CCTV are still conducted manually using paper-based forms. This process poses several challenges, including the risk of data loss, delays in inspections, and difficulties in retrieving historical maintenance records, particularly due to the limited number of airport electronic technicians. This study aims to design and develop a web server–based monitoring system for the maintenance of aviation security equipment, equipped with an interactive dashboard and Telegram notification integration. The system is designed to support routine maintenance monitoring that is digitally documented, structured, and in accordance with Directorate General Decree KP 241 of 2014 and the principles of electronic government efficiency as stated in Presidential Instruction No. 1 of 2025. The research employs the Waterfall Software development model, which includes stages of requirements analysis, system design using Unified Modeling Language (UML), Implementation with PHP, MySQL, HTML, and CSS, and testing using the Black Box Testing method. The resulting system design is expected to facilitate technicians in recording, storing, and monitoring equipment conditions in real-time while improving the efficiency and accuracy of maintenance activities at UPBU Silampari.

Keywords: Aviation Security Facilities; Interactive Dashboard; Maintenance; UPBU Silampari; Web Server.

Received: January 19, 2026
Revised: January 31, 2026
Accepted: February 25, 2026
Published: February 28, 2026
Curr. Ver.: February 28, 2026

1. Introduction

The Airport Operator Unit (UPBU) Silampari is one of the airports that has a strategic role in the South Sumatra region, located in the city of Lubuklinggau, and this airport is classified as a Class III UPBU, which means it is managed by the Central Government through the Directorate General of Civil Aviation and serves small to medium-scale flights. Nevertheless, Silampari Airport is still required to meet the same aviation security standards as other major airports. Several main elements that contribute to the realization of aviation security are the availability and reliability of security facility equipment at UPBU Silampari such as X-Ray Baggage Scanner, Walk Through Metal Detector (WTMD), Hand Held Metal Detector (HHMD), and Closed Circuit Television (CCTV) (ICAO, 2017). These equipment function as the first line of defense in preventing threats that could endanger the safety of passengers, personnel, and aviation infrastructure.

To ensure that all of these security equipment function optimally, routine and documented maintenance activities are required. Routine maintenance aims to examine technical conditions, detect potential damage early, and ensure that the equipment is ready for use according to international security standards. All member states are required to ensure that security facility equipment at airports is always in an operationally feasible condition and that

their inspections are officially recorded (ICAO, 2017). Similar obligations are also stated in the National Aviation Security Program which regulates procedures for inspection of security equipment, procedures for reporting inspection results, and archiving obligations for audit and evaluation purposes (Direktorat Jenderal Perhubungan Udara, 2024).

On the other hand, the Aviation Security Facility Equipment Standards specifically establish technical and operational standards for equipment such as X-Ray, WTMD, HHMD, and CCTV, including requirements for periodic inspections and their documentation (Direktorat Jenderal Perhubungan Udara, 2014). These guidelines also serve as the main reference in the implementation of equipment maintenance at UPBU Silampari.

Observations at Silampari Airport show that maintenance document management is still carried out manually using paper forms or simple spreadsheets. When required in digital form, these documents are usually only photographed or scanned manually. This unstandardized process creates various problems, especially when a sudden audit occurs. Technicians often experience difficulties in tracing old maintenance data, which has the potential to cause blame between parties when discrepancies occur. In addition, physical documents are very vulnerable to damage, loss, or inaccurate recording. As a result, the audit process becomes inefficient due to the absence of digital data backup.

Therefore, an innovative digital system is needed that can assist technicians in monitoring and managing maintenance data efficiently while also providing automatic notifications. This web system does not completely replace the manual system, but functions as a supporting system that provides digital data backup. The development of a web server-based maintenance monitoring system becomes a strategic solution to these problems. This system, if equipped with an interactive dashboard, can display equipment conditions in real-time and in a structured manner. Integration with the Telegram notification service will also help remind technicians of inspection schedules or routine maintenance. In addition to supporting the paperless principle, this system increases accountability because all data are stored digitally, documented, and easily accessible at any time.

Previous research conducted by D. Awalludin, D. Apdian, and V. Kristiani discussed the analysis and design of a web-based dies production daily report information system. The research results show that web-based systems are able to reduce recording errors and increase efficiency in daily reporting. However, this research still focuses on the manufacturing industry and has not yet been applied to equipment maintenance activities in the airport environment.

Further research conducted by Nadilla Oktavia and Devi Nurmalia (2022) developed a web-based daily maintenance information system for medical equipment. The results of the study show that web-based systems can simplify the maintenance and monitoring process of equipment in a more structured manner. However, the developed system has not yet been equipped with real-time notification features and has not been implemented in airport operational facilities.

Another study by Rivan Haposan, Issa Arwani, and Tibyani (2021) utilized Telegram Bot notification technology in a daily report customer reminder system. The research results show that the use of the Telegram Bot API is effective in providing automatic reminders with a system validation rate reaching 100%. Nevertheless, this study still focuses on the automotive sector and has not been applied to technical activities in the airport environment.

Furthermore, research conducted by Febri Hidayat Saputra et al. (2023) examined the implementation of a website-based electronic equipment controlling system at Sultan Hasanuddin Airport. The results of the study show that web-based systems are able to improve monitoring efficiency and accelerate responses to equipment disturbances. However, the system has not yet been equipped with automatic notification features or analytical dashboards to facilitate comprehensive monitoring.

Research by Rizky Parlika et al. (2020) discussed the implementation of MySQL access and local web servers through the internet network using Ngrok. The research results show that Ngrok enables access to local servers without requiring commercial hosting services and has a fairly good level of security. This technology is relevant to support network integration in environments with limited internet connectivity.

In addition, research conducted by R.U. Ginting et al. (2023) developed a website-based asset inventory information system at PT Dosni Roha. The results of the study show that the system is able to improve efficiency in the process of storing, tracking, and archiving asset data. However, the developed system has not yet been equipped with reminder or notification features related to asset maintenance schedules.

However, based on the results of previous studies and reviews, there has been no research that specifically discusses the three components of monitoring, interactive dashboards, and Telegram notifications in the context of aviation security facility maintenance, especially in Class III UPBU such as Silampari Airport.

Therefore, the main focus of this research is to design a web-based monitoring system capable of providing automatic notifications through Telegram, as a digital solution to the limitations of technician personnel in Class III UPBU in accordance with the KP 241 Year 2014 guidelines.

2. Literature Review

Information systems are a combination of humans, hardware, software, data, and procedures that interact with each other to collect, process, store, and present information in order to support decision-making processes in an organization (Purnawati et al., 2024). In modern organizational operations, information systems play an important role in improving work efficiency, data accuracy, and ease of access to information needed by users. Through computer-based information systems, data that were previously recorded manually can be processed automatically so as to produce reports that are faster, more accurate, and integrated. The implementation of information systems also enables the monitoring of operational activities to be carried out in real-time, making it easier for management to supervise and evaluate organizational performance (Adam & Juliadarma, 2024).

In the context of airport operations, maintenance of aviation security equipment is an important activity that must be carried out in a planned and documented manner to ensure that all equipment can function properly. Maintenance includes inspection, testing, servicing, and recording of equipment conditions so that they continue to meet the established operational standards (Rosa, 2005). Based on the National Aviation Security Program, every airport operator is required to conduct periodic inspections and maintenance of all aviation security facilities in order to ensure a level of security in accordance with international standards established by the International Civil Aviation Organization (ICAO), particularly in Annex 17 concerning Aviation Security (Direktorat Jenderal Perhubungan Udara, 2024). In addition, the technical standards for aviation security facilities also require equipment such as X-Ray, Walk Through Metal Detector (WTMD), Hand Held Metal Detector (HHMD), and Closed Circuit Television (CCTV) to undergo routine inspections and calibration to ensure the reliability of security detection systems (Direktorat Jenderal Perhubungan Udara, 2014).

In the development of information systems, software development methods become one of the important aspects to produce structured systems that meet user requirements. One of the widely used methods is the Waterfall method which consists of several sequential stages, namely requirement analysis, system design, implementation, testing, and system maintenance (Wahid, 2020; Listiyan & Subhiyakto, 2021). In addition, web technologies such as web servers, database management systems, and programming frameworks play an important role in supporting the implementation of web-based information systems that can be accessed through local networks or the internet (Kadir, 2014). The use of interactive dashboards is also increasingly applied in modern information systems because they are able to present data visually and in real-time, thereby making it easier for users to understand information and support decision-making processes more effectively (Hilmi et al., 2025; Ismubandono et al., 2019)..

3. Research Method

This study uses a software development method with the Waterfall model, which is one of the approaches in the Software Development Life Cycle (SDLC) (Wahid, 2020). The Waterfall method was chosen because it has clear and structured stages, making it easier in the process of designing and developing a web-based maintenance monitoring system at UPBU Silampari. Through this approach, each development stage produces outputs that become the basis for the next stage so that the development process can be carried out in a directed and well-documented manner.

The research stages begin with requirement analysis which aims to identify user needs for the system through direct observation in the field as well as collecting information related to the maintenance process of airport security equipment. The next stage is system design which is carried out using Unified Modeling Language (UML) such as use case diagrams and entity relationship diagrams (ERD) to describe user interactions and the structure of the system database. After that, the implementation stage is carried out by writing program code

using web development technologies, followed by the testing stage to ensure that all system functions run properly and meet user requirements.

This research was conducted at Silampari Airport Lubuklinggau during the period of October 2025 to February 2026. Data collection was carried out through direct observation and documentation of maintenance activities of aviation security equipment such as X-Ray, CCTV, Walk Through Metal Detector (WTMD), and Hand Held Metal Detector (HHMD) in accordance with aviation security facility standards (Direktorat Jenderal Perhubungan Udara, 2024; Direktorat Jenderal Perhubungan Udara, 2014). In the system development process, various hardware such as laptops, routers, and smartphones were used as testing media, as well as software such as Next.js, Java Spring Boot, PostgreSQL, DBeaver, IntelliJ IDEA, HTML, CSS, JavaScript, Postman, and Bootstrap to support the process of system development, database management, and system testing.

4. Results And Discussion

General Overview

In general, the working principle designed for routine maintenance monitoring activities of security facility equipment is the same as the standard procedures that exist in the security facility equipment unit at Silampari Airport. The difference lies in the maintenance checklist activities carried out by technicians which are usually done manually. The concept of this system can make it easier for technicians by only needing to carry a smartphone and internet access to access the website.

After access is granted, technicians can input maintenance data according to the procedures, then the data will be stored through the database and the data will be automatically saved according to the time of inspection. This system is also equipped with an interactive dashboard to display equipment condition status in real-time as well as a Telegram notification feature as a reminder for maintenance schedules.

The following is an illustration of the current condition at Silampari Airport and the desired condition:

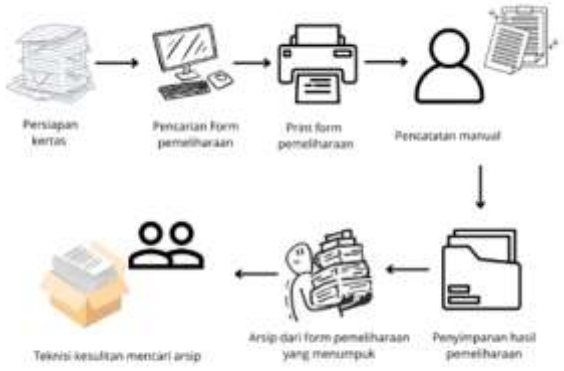

Figure 1. Illustration of Current Conditions.

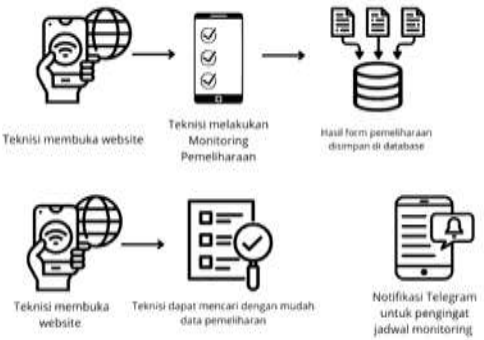

Figure 2. Illustration of Desired Conditions.

DESIGN STAGES

Software Installation Process

Visual Studio Code

a. Visit the official Visual Studio Code website through a web browser, such as Google Chrome or Microsoft Edge, at the link
b. https://code.visualstudio.com/download
c. Click the "Download for Windows" button on the page to start the download process of the Visual Studio Code application.
d. After the download process is complete, the Visual Studio Code application icon will appear in the downloads folder.
e. Click the installer icon to start the Visual Studio Code installation process.
f. After the installer window opens, the user will be asked to agree to the terms and conditions of using Visual Studio Code. Select the option "I accept the agreement", then click the "Next" button.
g. Determine the storage location for the Visual Studio Code installation. Next, the user can browse the desired directory, then click the "Next" button.
h. Next, the system will ask for confirmation to start the installation process. Click the "Install" button to continue.
i. After the "Install" button is clicked, the Visual Studio Code installation process will run for approximately one minute until it finishes on the user's device.
j. Figure IV.6 After all the previous stages are completed, the Visual Studio Code window will open successfully. The user can then create a new file in the application and select the desired programming language to begin the software development process.

Based on the stages that have been explained, it can be concluded that the installation process of Visual Studio Code on the Windows operating system can be carried out easily and successfully. The steps described in this guide can be applied to various types of web browsers on Windows to download and install Visual Studio Code, especially on the Windows 10 operating system.

JAVA

1) Open a web browser, then search for "Download JDK 17" or access directly through the link https://www.oracle.com/Java/technologies to download the Java Development Kit (JDK) from the official Oracle website.
2) Run the JDK installer file by double-clicking it. The system may ask for permission before the installation process begins. Select the "Yes" option to allow the installer to run.
3) Click the "Next" button to start the installation process. On the next screen, an option will be displayed to change the installation path or location. Users can adjust the installation directory as needed.
4) Next, click the "Next" button to continue the installation process. The system will display the installation progress indicator.
5) After the installation process is complete, a screen will appear indicating that the installation has been successful.
6) Next, open Command Prompt and type the command

Java -version to ensure that Java has been successfully installed.

3. Node.js

a. Access the official Node.js download page through the link
b. https://nodejs.org
c. Click the Windows Installer icon to begin downloading the installer file.
d. After the download process is complete, click the downloaded file through the web browser to open and run the Node.js installer.
e. Click the "Next" button in the installer window, then check the License Agreement approval box. Next, continue the installation process by pressing the "Next" button.
f. At the final stage of setup, there is an option to install several additional tools used in compiling native modules. The selection of this option is left to the user; however, in this guide, these additional tools are not included in the installation process.
g. At the final stage, click the "Install" button to start the installation process. During the process, the installer display will show the progress status.

h. After the installation process is complete, the installer will display a notification that the installation has been successful and ask the user to press the "Finish" button to close the installer window.

i. To check the Node.js version as a form of installation verification, open Command Prompt by pressing the Windows key, typing cmd, then pressing Enter. After the Command Prompt window opens, run the command as shown in the following image:

```
1 C:Users\Administrator>nod
2 v12.18
```

Figure 3. Command Prompt Node.js Verification.

i. Installing Node.js on the Windows operating system has included Node Package Manager (NPM). To ensure that NPM can be used properly, run a command to install the React package as a form of testing, namely by using the package installation command through NPM as shown in the following image:

```
1 C:Users\Administrator>npm install -g
2 +      16.13.
3 added 6 packages from 3 contributors in
```

Figure 4. Command Prompt Node.js Verification Complete.

PostgreSQL

1) Users can download the latest stable version of the PostgreSQL installer that matches the Windows operating system through the official EnterpriseDB website at the link
2) https://www.enterprisedb.com/downloads
3) After the installer file has been successfully downloaded, run the installer by double-clicking it, then follow the installation stages displayed. At the initial stage, click the "Next" button to continue the process.
4) Select the installation directory or folder as the location to install PostgreSQL, then click the "Next" button to continue.
5) Determine the PostgreSQL components to be installed according to user needs, then click the "Next" button.
6) Select the database directory as the storage location for PostgreSQL data, then click the "Next" button.
7) Set a password for the PostgreSQL database superuser account (postgres).
8) Determine the port number that will be used by PostgreSQL. Ensure that the port is not used by other applications. If unsure, use the default setting, which is port 5432, then click the "Next" button.
9) Select the default locale setting that will be used by the database, then click the "Next" button.
10) Click the "Next" button to start the installation process. Wait until the installation process is complete, which may take several minutes.
11) After the installation process is complete, click the "Finish" button to end the PostgreSQL installation process.

5. PostgreSQL Verification

There are several methods that can be used to verify the successful installation of PostgreSQL, one of which is by connecting to the database server using a client application such as pgAdmin or psql. The simplest and fastest method is by utilizing the psql shell. The steps that can be taken are as follows:

a. Accessing the psql Shell
Search for the psql application through the Windows search bar, then open the application. Next, enter the required server information, including the server address (localhost), database name (default postgres), port number (default 5432), username (postgres), and the password that was set during the installation process.

b. Verifying Installation
After successfully logging into the psql shell, run the command SELECT version(); to ensure that PostgreSQL has been installed correctly. If the installation is successful, the system will display information about the PostgreSQL version being used.

IntelliJ IDEA IDE

1) To download and install IntelliJ IDEA, visit the official IntelliJ IDEA website through the link
2) https://www.jetbrains.com/idea/download
3) Click the file with the .exe extension as shown in the related figure to begin downloading the executable file.
4) After the download process is complete, open the downloads folder to ensure that the file has been successfully downloaded.
5) Run the .exe file, then the initial installer display will appear. Click the "Next" button to continue the installation process.
6) Determine the installation location of IntelliJ IDEA. Users can change the installation location by pressing the "Browse" button or using the default location. After that, click "Next" to continue.
7) Check the additional options required, such as creating a desktop shortcut and other settings. After that, click "Next".
8) Click "Next", then the IntelliJ IDEA installation process will begin. At this stage, the system will display the installation progress screen as shown in the related figure.
9) After the installation process is completed successfully, the final installation screen will appear. Select the "Run IntelliJ IDEA" option to open the application.
10) When IntelliJ IDEA is run, users will see the IntelliJ IDEA splash screen.
11) Accept the User Agreement by clicking the "Confirm" button to continue using IntelliJ IDEA.

IntelliJ IDEA Activation

IntelliJ IDEA activation can be carried out through three methods, namely free trial license, premium license, and educational subscription for students which provides free license renewal. The following are the steps for the free trial license:

a. After completing the previous installation steps, an activation display will appear providing two options, namely "Activate IntelliJ IDEA" and "Start Trial". The trial option provides free access to use IntelliJ IDEA for 30 days.
b. After selecting the trial option, the trial period will automatically become active.
c. Click the "Activate IntelliJ IDEA" option, then select "New Registration" if the user does not yet have a JetBrains account.
d. Next, the user will be directed to the login page. If you do not yet have an account, you can create a new account on that page.
e. The registration process can be done by logging in through a Google account, GitHub, or other available methods, or by filling in personal data such as first name, last name, and password to create a new JetBrains account.
f. After successfully logging in, the user will be shown a list of available premium license packages and can select and purchase a package according to their needs.
g. Steps for activating the educational license (Student)
h. The first step that must be done is creating a JetBrains account by following the stages in the free trial license activation process.
i. After the account has been successfully created, access the Education Subscription menu and complete the requested data. The email address used must be an official email issued by a university or educational institution.
j. After all data has been filled in and the requirements are met, continue the process on the Education Subscription page according to the instructions provided.
k. After agreeing to the applicable terms and conditions, the student license will automatically become active on the user account.
l. Next, perform the login process through the same web page using the student account credentials that have been created.
m. After successfully logging in, IntelliJ IDEA will automatically detect the student license in the background if the user logs in using the same account. Click the "Accept" button to begin using the student license.
n. After the activation process is complete, users can begin using IntelliJ IDEA according to their needs.

Note: Educational licenses provide free renewal facilities as long as the student status is still valid.

Database Creation (PostgreSQL)

The initial stage in database creation is designing the database schema. This design aims to ensure that the structure of the stored data is capable of supporting all the needs of the security equipment maintenance monitoring system.

There are several things that need to be considered for database creation as follows:

a. Entity Identification

Entities are the main objects whose data will be stored in the database. Based on system requirements, the main entities identified include:

1) m_roles: stores system user role data
2) m_users: stores user data
3) user_role_junction: linking table between users and roles
4) master_peralatan: stores master data of aviation security equipment
5) peralatan_uraian: stores equipment details or descriptions
6) uraian_kegiatan: stores types of maintenance activities
7) report_activities: stores maintenance activity reports
8) reports_history: stores report history
9) t_Approval: stores report approval data
10) t_notification: stores notification data including Telegram integration

b. Attribute Determination

Each entity has attributes that represent the information stored. For example, the m_users entity has attributes:

1) user_id
2) email
3) join_date
4) member_id
5) password
6) username

These attributes are used to manage authentication data and the identity of system users.

c. Determining Relationships Between Entities

After the entities and attributes are determined, the next step is determining relationships between entities.

Examples:

1) The report_activities entity has a one-to-many relationship with reports_history, because one activity report can have many change histories.
2) The m_users entity has a many-to-many relationship with m_roles which is realized through the user_role_junction table.

d. ERD Creation

The Entity Relationship Diagram (ERD) in the equipment maintenance monitoring system at UPBU Silampari is designed to support the process of data collection, monitoring, reporting, and approval of aviation security equipment maintenance.The database structure is divided into several main groups so that data management can be carried out in a structured and integrated manner.

The equipment management group is the core of the system that stores all aviation security facility data.The master_peralatan table functions as the main table containing asset information such as facility type, equipment type, location, brand, and report type.The uraian_kegiatan table stores standard maintenance procedure activities, while the peralatan_uraian table acts as a linking table that determines the relationship between equipment and maintenance activities along with the sequence of implementation.

The user management group applies the Role-Based Access Control (RBAC) concept to regulate system access rights.The m_users table stores user account data. The m_roles table defines types of roles in the system. The user_role_junction table connects users with certain roles so that each user has access rights according to their responsibilities.

The monitoring and reporting activity group is used to record dynamic maintenance activities.The report_activities table stores maintenance activity details performed in the field.The reports_history table functions as an archive of verified reports, including technician information, report period, location, and final status.

The validation and notification group supports the approval flow and notification integration.The t_Approval table records the report approval process by authorized parties. The

t_notification table stores notification data that will be sent through Telegram. Overall, this ERD illustrates the system workflow starting from maintenance activities carried out by technicians, activity recording, sending notifications, to the verification process and permanent report storage.

e. Database Schema Design

After the schema design is complete, the next stage is implementing the database using PostgreSQL.

1) Database Creation
   The database is created using a PostgreSQL client application such as DBeaver. This database will serve as a container for storing all monitoring system data.
2) Table Creation
   Table creation is carried out using SQL scripts of the DDL (Data Definition Language) type. This script includes table definitions, data types, primary keys, foreign keys, and other constraints. Example SQL script to create the m_users table:

```
CREATE TABLE "local".m_users (
    user_id uuid NOT NULL,
    email varchar(255) NULL,
    join_date timestamptz(6) NULL,
    member_id varchar(255) NULL,
    "password" varchar(255) NULL,
    username varchar(255) NULL,
    CONSTRAINT m_users_pkey PRIMARY KEY (user_id),
    CONSTRAINT uklxevwjnma30s6actvjsblbp6a UNIQUE (username)
);
```

   The table uses the UUID data type as the primary key to guarantee uniqueness of user data and a unique constraint on the username column to prevent duplication.

Web Application Development (Java Spring Boot and Next.JS)

System development was carried out using a client–server architecture, with Spring Boot as the backend and Next.JS as the frontend.

- Backend (Java Spring Boot)

The backend project was initialized using Spring Initializr with dependencies Spring Web, Spring Data JPA, and PostgreSQL Driver. Next, entities were created to represent database tables using JPA annotations, repositories for data management, services for business logic, and REST API-based controllers to handle requests from the frontend. The configuration of the PostgreSQL database connection was carried out through the application.properties file.

- Frontend (Next.JS)

The frontend project was created using Next.JS. Main pages such as dashboard, equipment list, and data management forms were developed using React components. Data is obtained from the Spring Boot REST API using fetch or Axios, then displayed on the user interface. Add, edit, and delete data functionalities are implemented through user interactions.

Supporting Software

The supporting software used includes PostgreSQL (pgAdmin/DBeaver), JDK and Spring Boot (IntelliJ IDEA/Eclipse), Node.js and Next.JS (Visual Studio Code), and Git and Postman as development and testing tools.

The development process of the Aviation Security Equipment Maintenance Monitoring System at UPBU Silampari begins with the planning stage, namely system requirements analysis which includes security equipment data collection, determination of maintenance schedules, Telegram notification mechanisms, and dashboard interface design so that it is easy to use.

The next stage is database design and implementation using PostgreSQL, which begins with ERD preparation to map data relationships, followed by the implementation of SQL scripts to build maintenance data storage table structures.

Next, the backend implementation stage is carried out using Spring Boot to build business logic, data management, and integration with the Telegram Bot API as an automatic notification medium. On the frontend side, Next.JS is used to develop an interactive dashboard that displays maintenance data dynamically through REST API integration. After the system is built, a testing stage is carried out to ensure all functions run properly, including notification delivery and data synchronization. The final stage is deployment to the production server and system maintenance to maintain stability, fix errors, and adjust the system according to operational needs and applicable aviation security regulations.

DESIGN TESTING

Based on the design results that have been created, testing is then carried out on the system to determine whether the web-based aviation security equipment maintenance monitoring system can operate according to needs and can be accessed via the web on various devices.

Admin Interface Testing

User interface testing for the Admin role is carried out to ensure that all management functions of the web-based aviation security equipment maintenance monitoring system can run according to the design. This testing is an implementation of the backend and frontend development stages built using Spring Boot and Next.JS.

At the authentication and access security stage, the system provides a Login page as the main gateway for users. Admin enters username and password which are then verified by the system through matching data in the m_users table. After the authentication process is successful, the system applies the Role-Based Access Control (RBAC) mechanism by referring to the m_roles and user_role_junction tables to ensure that the user has Admin access rights before accessing all system features.

After successfully logging in, the admin is directed to an interactive dashboard that functions as a monitoring center. This page displays a summary of equipment conditions sourced from the master_peralatan table, such as: total number of equipment, equipment in good condition, equipment under maintenance, and equipment detected experiencing disturbances. In addition, equipment condition trend graphs are displayed based on historical data from the reports_history table to help monitor equipment performance periodically.The system also displays real-time anomaly alerts or notifications taken from the t_notification table.

Furthermore, the admin has access to user management features, which allow adding new users to the system. User data such as name, username, email address, and password will be stored in the m_users table, while the assignment of user roles is recorded in the user_role_junction table. This feature ensures that each user has access rights according to their role in the equipment maintenance process.

In system settings and notification integration, Admin can configure general application settings and manage Telegram integration features. The system provides a notification trigger function that will run a process on the backend to send automatic messages to technicians via Telegram Bot. Every notification sending activity is recorded in the t_notification table, so that the entire communication history can be documented properly.

With this testing carried out, it can be known that the Admin interface has been optimally integrated with the database and backend services, and is able to support monitoring processes, user management, and notification delivery according to operational needs at UPBU Silampari.

User Interface Testing

User interface testing is carried out to ensure that the aviation security equipment maintenance monitoring system can be used effectively by technicians as the main users in the field. This testing represents the system testing stage, where functionality is tested from the perspective of users with User access rights.

At the authentication stage, testing was carried out using a technician account with the username elban.The system verifies user data through the m_users table and determines user access rights based on relationships in the user_role_junction table, so that the user is identified as having the technician role. After the authentication process is successful, the system displays a special user interface that is different from the Admin according to the Role-Based Access Control concept.

After entering the system, the user is directed to the technician dashboard which displays a summary of equipment conditions. Statistical information such as total number of equipment, equipment in good condition, equipment under maintenance, and equipment experiencing disturbances is displayed as a form of situational awareness. The navigation menu on the interface side is limited to the main operational features, namely Dashboard, Reports, Approval, and Alerts, so that technicians can focus more on maintenance activities.

Report data is recorded in the report_activities table, while processed reports will be stored in reports_history. Users can choose inspection types based on daily, weekly, or monthly periods, adjusted to the report type attributes in the master_peralatan table. Technicians also fill out a report form that includes facility type, equipment type, work location, and documentation in the form of images as evidence of equipment condition.

After the report is completed, the technician proceeds to the approval submission stage.Through the submission feature, the user fills in the report title and implementation period, which is then stored in the t_Approval table. At this stage, the report status changes to waiting for approval and will then be verified by the Supervisor and Manager.

Based on the results of this test, it can be concluded that the user interface has been able to optimally support technician activities, starting from authentication, maintenance reporting, to the approval submission process according to the designed system workflow.

Supervisor and Manager Role Testing

The Supervisor and Manager roles in the AeroEquip application function as the highest authority in the supervision and validation process of operational maintenance of security equipment at UPBU Silampari. Accounts with the Supervisor and Manager roles, such as Tomikurniawan and erlandgilisaptomi, have full access to the executive dashboard which displays the condition of all airport assets in real-time. Through this dashboard, the Manager can monitor a summary of equipment status, including total number of equipment, operational condition, equipment currently under maintenance, and anomalies detected on the current day. In addition, equipment condition trend graphs are used as analytical tools to assess the stability and performance of security facilities over time.

In the figure shown, the main role of the Supervisor and Manager lies in controlling the approval workflow. Supervisor and Manager have access to the approval menu which displays a list of maintenance reports that have been submitted by technicians and previously verified in the system. At this stage, the Supervisor and Manager review in detail the reports of daily, weekly, and monthly maintenance activities recorded in the system. If there are records that require correction, both roles can return the report to the technician with the available notes.

After the verification process is completed, the Supervisor provides approval status, followed by the Manager providing final approval status (APPROVED FINAL_MANAGER). This status automatically locks the report data so that it can no longer be modified, in order to maintain data integrity and audit requirements.

The three roles can access the entire history of equipment maintenance such as CCTV, X-Ray, HHMD, Walk Through Metal Detector along with their completion status. Approved reports can also be exported into PDF documents as official evidence of maintenance implementation, which is then used for administrative purposes and compliance with aviation security regulations, as shown in the figure.

Telegram Notification Testing

Based on the figure above, the analysis shows that the Telegram notification integration feature in the aviation security equipment maintenance monitoring system at UPBU Silampari has functioned according to the design. This test proves that the system is capable of delivering maintenance information quickly and in real-time to technicians and related personnel through external communication media.

Notifications are sent through the official Telegram Bot named Airport_Equipment_Management_Bot. The use of this Bot allows messages to be received directly on the user's mobile device without dependence on access to the web dashboard, thereby increasing accessibility and responsiveness to equipment conditions in the field.

In this test, there are two types of notifications successfully sent by the system.

The first type is periodic maintenance warnings, in the form of messages "Monthly Maintenance Warning" which contain information on findings such as problems in sensitivity inspection and equipment performance testing with repair-required status. This information originates from data stored in the t_notification table, especially in the title and information columns, which are generated based on technician maintenance report results.

The second type is an automatic daily reminder, which is sent on a scheduled basis at 08:00 WIB. Greeting messages such as "Good morning" are sent as an implementation of the Trigger Daily Reminder feature, which can be executed by the Admin through the system settings menu. This feature aims to remind technicians to perform inspections and fill in maintenance reports routinely.

In addition, notification messages are also equipped with a direct link to the AeroEquip system. When the link is clicked, the user will be directed to the Login page or web server dashboard to perform further actions such as filling out maintenance reports or approving reports. This indicates good integration between the web system and external notification media.

The success of sending these notifications indicates that the application logic on the backend side has successfully connected with the Telegram Bot API, and the entire external communication pathway testing process can be declared successful. Every message sent is also consistently recorded in the t_notification table, including information on notification title, message content, and delivery time.

Overall, the results of this test show that the Telegram notification feature is capable of functioning as an effective early warning system in supporting the reliability and operational readiness of aviation security facilities at UPBU Silampari.

Web Notification Center Testing

Based on the figure, the notification center page in the AeroEquip system functions as a centralized monitoring module to display system alerts and maintenance information for security equipment at UPBU Silampari. This page is designed to display notification data in real-time, sourced directly from the t_notification table in the database.

Each displayed notification contains important information such as maintenance warning title, description of detected problems (for example disturbances in sensitivity inspection or voltage supply), and equipment operational status indicating repair-required conditions. In addition, the system also displays timestamp indicators with precision according to the created_date attribute, so that users can determine the time when each warning occurred. The use of visual indicators in the form of icons and color differences also helps users identify the priority level of notifications that must be followed up immediately.

The notification management feature on this page allows users to manage incoming information so that it remains organized. Through the mark as read function, users can change the notification status after the information has been understood or followed up.

This action directly updates data in the database through the already_read and read_at attributes, so that the system can record that the notification has been received and processed by the relevant officer.

The test results also show consistent data synchronization between the web notification center page and notifications sent through Telegram. The message content received by users on the Telegram application corresponds to the information displayed on the web interface. This proves that the Spring Boot-based backend has successfully distributed notification data from the same database source to multiple client platforms, namely Next.JS web application and Telegram Bot service.

Overall, the existence of this notification center page not only functions as an information delivery medium but also as a means of managing and tracking system alert follow-up actions. Thus, every anomaly or maintenance condition of security equipment at UPBU Silampari can be well documented and have a clear audit trail, starting from the appearance of notifications to the handling process.

INTERPRETATION OF DESIGN TEST RESULTS

Based on the results of tests conducted on the user side (User/Technician), it can be interpreted that the aviation security equipment maintenance monitoring system at UPBU Silampari has functioned according to the predetermined design. This testing represents the system testing stage, where all main features are tested directly from the perspective of field officers as operational users.

Table 1. Test Results.

No	Feature Tested	Test Scenario	User Input/Action	Expected Output	Test Result	Conclusion
1	User Login / Authentication	User logs in using technician account	Username: elban, Correct password	System verifies credentials in m_users and user_role_junction tables → User successfully logs in	Successful	Login validation and access rights work according to design
2	Role-Based Access Control	User logs in as technician	Access menu after login	System only displays technician menus	Successful	RBAC functions correctly
3	Technician Dashboard	Displays equipment data summary	User opens dashboard	Equipment statistics appear according to database tables	Successful	Dashboard displays real-time data
4	Report Creation	User creates maintenance report	Inspection type, equipment condition, notes, upload photo	Data stored in report_activities table	Successful	Report feature works properly
5	Inspection Type Selection	User selects inspection type	Daily/weekly/monthly	System filters forms based on inspection type	Successful	Filtering works correctly
6	Visual Evidence Upload	User uploads equipment photo	Upload image file	File saved and preview displayed	Successful	Upload works normally
7	Report History	User views past reports	Click report history menu	List of reports displayed with status and dates	Successful	Report history accurate

8	Approval Submission	User submits report	Click submit button	Status becomes Waiting for Approval	Successful	Approval workflow works
9	Report Status Integration	System updates report status	Automatic system process	Status changes Draft → Waiting Approval	Successful	Status integration works
10	Report Data Validation	System retrieves report	User opens report page	Report data displayed correctly	Successful	No data mismatch

During the authentication process, the system successfully verified a user with User access rights using the account with the username elban. Validation was carried out by matching data in the m_users table, as well as determining the user's role based on the relationship in the user_role_junction table. The test results indicate that the authentication mechanism and Role-Based Access Control (RBAC) configuration functioned properly, ensuring that users can only access features according to their authorized permissions. After successfully logging into the system, the user is directed to a technician-specific dashboard that displays a summary of equipment conditions. Equipment statistics information, such as the total number of equipment units, equipment in good condition, equipment under maintenance, and equipment experiencing malfunctions, is displayed correctly based on data stored in the master_peralatan and reports_history tables. In addition, the navigation menu display is limited to the main operational features, indicating that the system is able to adjust the interface according to the user's role, thereby supporting the efficiency of technicians' work activities.

In the testing of the report management and creation features, technicians were able to successfully create maintenance reports which were stored in the report_activities table. The system was also able to display the history of previously created reports along with their completion status. The selection of inspection types based on daily, weekly, and monthly periods worked in accordance with the report type attributes stored in the master_peralatan table. The visual evidence upload feature also functioned properly as supporting documentation to validate the actual condition of equipment in the field.

At the approval submission stage, technicians successfully submitted reports for verification through the submission feature. The submission data was recorded in the t_Approval table, and the report status automatically changed to Waiting for Approval. This indicates that the approval workflow has been well integrated between technician users and authorized personnel responsible for verification.

Based on the interpretation of the test results, it can be analyzed that the system design has been functionally implemented and is capable of supporting the processes of monitoring, reporting, and submitting approval requests for aviation security equipment maintenance in a structured and integrated manner, in accordance with the operational requirements at UPBU Silampari.

5. Conclusions

Based on the series of research stages including needs analysis, system design, implementation, and testing of the web server–based aviation security equipment maintenance monitoring system with an interactive dashboard and Telegram notification integration at UPBU Silampari, it can be concluded that the developed system has been successfully designed and implemented in accordance with airport operational needs. The system is capable of digitally documenting aviation security equipment maintenance activities in a structured manner and can be easily accessed by technicians. In addition, the integration of automatic notifications through Telegram helps provide reminders related to inspection and routine maintenance schedules, thereby improving discipline and timeliness in the implementation of equipment maintenance.

The implementation of the web-based system has also proven to improve work efficiency compared to the manual recording method previously used. This improvement is reflected in faster data recording processes, safer digital data storage, easier reporting procedures, and the ability to monitor equipment conditions through an interactive dashboard. Therefore, the developed monitoring system can serve as an applicable digital solution to support aviation security facility maintenance activities at UPBU Silampari and has the potential to be implemented in other UPBU units with similar operational needs.

References

Adam, A., & Juliadarma, M. (2024). *Sistem informasi manajemen*.

Awalludin, D., Apdian, D., & Kristiani, V. (2022). Analisis dan perancangan sistem informasi pembuatan daily report produksi dies berbasis web. *Jurnal Algoritma, 18*(2), 342–351. https://doi.org/10.33364/algoritma/v.18-2.868

Direktorat Jenderal Perhubungan Udara. (2014). *KP 241 Tahun 2014 tentang pedoman pengoperasian, pemeliharaan, dan pelaporan fasilitas keamanan penerbangan*.

Direktorat Jenderal Perhubungan Udara. (2024). *Peraturan Menteri Perhubungan Republik Indonesia Nomor PM 9 Tahun 2024*.

Ginting, R. U., Nduru, Y., & Damanik, B. (2023). Sistem informasi inventaris aset berbasis website di PT Dosni Roha Cabang Medan. *Jurnal Mahajana Informasi, 9*(2), 33–40. https://doi.org/10.51544/jurnalmi.v8i2.4683

Hilmi, A. N., Prayitno, E., & Siregar, J. (2025). Perancangan sistem informasi dashboard tiket layanan helpdesk customer care control center (C4) consumer. *Journal of Innovative Future Technology, 7*(2). https://doi.org/10.47080/125t5m48

International Civil Aviation Organization. (2017). *Annex 17: Security—Safeguarding international civil aviation against acts of unlawful interference*.

Kadir, A. (2014). *Pengenalan sistem informasi*.

Listiyan, E., & Subhiyakto, E. R. (2021). Rancang bangun sistem inventory gudang menggunakan metode waterfall studi kasus di CV Aqualux Duspha Abadi Kudus Jawa Tengah. *Konstelasi: Konvergensi Teknologi dan Sistem Informasi, 1*(1), 74–82. https://doi.org/10.24002/konstelasi.v1i1.4272

Oktavia, N., & Nurmalia, D. (2022). Penggunaan sistem informasi daily maintenance alat medik berbasis aplikasi website dalam rangka pemeliharaan alat medik di ruang rawat inap. *Jurnal Kepemimpinan dan Manajemen Keperawatan, 5*(2), 169–176. https://doi.org/10.32584/jkmk.v5i2.1556

Parlika, R., Khariono, H., Kusuma, H. A., Abrori, M. R., & Rofik, M. A. (2020). Implementasi akses MySQL dan web server lokal melalui jaringan internet menggunakan Ngrok. *JIKO (Jurnal Informatika dan Komputer), 3*(3), 131–136. https://doi.org/10.33387/jiko.v3i3.1799

Purnawati, N. W., & Arsan, I. N. A. (2024). *Sistem informasi*.

Rivan Haposan, T., Arwani, I., et al. (2021). Pemanfaatan teknologi notifikasi bot Telegram dalam pengembangan sistem daily report customer reminder berbasis web. *Jurnal Sistem Informasi, Teknologi Informasi, dan Edukasi Sistem Informasi, 2*(2), 64–73. https://doi.org/10.25126/justsi.v2i2.21

Rosa, Y. (2005). Perencanaan dan penerapan preventive maintenance peralatan laboratorium. *Jurnal Teknik Mesin, 2*(2), 106–119.

Saputra, F. H., et al. (2023). Implementasi sistem controlling peralatan elektronika berbasis website di Bandara Sultan Hasanuddin Makassar. *Jurnal Ilmiah Ilmu Komputer, 9*(2), 73–77.

Wahid, A. A. (2020). Analisis metode waterfall untuk pengembangan sistem informasi. *Jurnal Ilmu-Ilmu Informatika dan Manajemen STMIK, 1*(1), 1–5.