**VAULT: Virtual Archiving and Updated Logistics Tracking with QR Code Integration for the Provincial Government of South Cotabato**

Rodin F. Suñiga

**Notre Dame of Marbel University**

**MIT 265 – Advanced Software Engineering**

Brenda M. Balala, MIT

March 2025

**Table of Contents**

Contents

[Introduction 4](#_Toc189928059)

[Background of the Study 4](#_Toc189928060)

[Statement of the Problem 5](#_Toc189928061)

[Objectives of the Study 6](#_Toc189928062)

[Concept of the Study 7](#_Toc189928063)

[Conceptual Framework 7](#_Toc189928064)

[System Architecture 8](#_Toc189928065)

[Data Flow and Processing 9](#_Toc189928066)

[Security and Authentication 10](#_Toc189928067)

[Deployment and Scalability 10](#_Toc189928068)

[Methods 11](#_Toc189928069)

[Hardware 11](#_Toc189928070)

[Software 12](#_Toc189928071)

[Frontend 12](#_Toc189928072)

[Backend 13](#_Toc189928073)

[Database 13](#_Toc189928074)

[Procedures 14](#_Toc189928075)

[Design Methodology 14](#_Toc189928076)

[Development Model 15](#_Toc189928077)

[Requirements Analysis 16](#_Toc189928078)

[Functional Requirements 17](#_Toc189928079)

[Non-Functional Requirements 19](#_Toc189928080)

[Requirements Documentation 20](#_Toc189928081)

[System Use Case diagram 20](#_Toc189928082)

[User Stories 21](#_Toc189928083)

[Acceptance Criteria 22](#_Toc189928084)

[System Design 23](#_Toc189928085)

# Introduction

## Background of the Study

Effective document handling is a key factor that should be considered, especially by government organizations that deal with large volumes of information. The conventional practice of handling paper-based files has been known to be both time-consuming and prone to error and delay. This is more relevant, especially for the Provincial Government of South Cotabato, where documents are manually transferred between departments with the following issues: missing files, delayed processing, and the movement of documents, which is also not well tracked. Some of the most essential technological progress is in document management solutions. Several organizations now, including the government, are shifting from manual systems to computer-based systems, which are more convenient and safer. One of these successful pilot initiatives is the Implementation of DSWD Caraga’s Digital Archiving System, which uses OCR for document classification and retrieval operations. This system eliminated the challenges faced by paper-based systems, time-consuming searches for files, and the risks of file loss.

The Provincial Government of South Cotabato experiences similar problems; several offices and departments do not have a localized and automated document tracking system, leading to productivity deterioration throughout the day. Some departments have many papers, and there is movement of the documents, which hampers decision-making. There is also less transparency in the handling of the papers. As has been observed in universities, manual processing of documents involves time and energy, increasing the amount of time spent on administration. To address these issues, this project introduces VAULT: Virtual Archiving and Updated Logistics Tracking, an innovative application with some options as web and mobile document tracking that uses Built-in QR codes. This system aims to improve document management in the provincial government, where paper-based data is digitized, and the real-time monitoring of interactions on the documents. Once again, about other systems, it must be pointed out that real-time tracking guarantees documents’ confidentiality and contributes to increasing the speed of decisions and, if necessary, controlling the flow of documents between departments.

QR codes can thus be considered an efficient way to eliminate this gap and enhance the document transaction process. Users can see the current status of a document and the department in charge of it in real-time by scanning the attached code. This significantly reduces the risk of document loss and enhances accountability through tracking and documenting every document movement at any time. The VAULT system benefits the Provincial Government of South Cotabato and is vital to developing the province’s digitalization. It addresses the need for a secure, efficient, and transparent document management system that enables the employees to track, update, and store the documents. As government organizations proceed to adopt various digital enhancements, VAULT is keen to make document management flexible and expandable.

## Statement of the Problem

1. Manual tracking is inefficient and often leads to delays, negatively impacting overall productivity.
2. The risk of document misplacement or loss is high, and manual data entry increases the likelihood of human errors.
3. Real-time tracking of document status and location is limited, making it difficult to monitor and manage records effectively.

## Objectives of the Study

The primary goal of this research is to create VAULT or Virtual Archiving and Updated Logistics Tracking, a document tracking system with QR Code integration for the Provincial Government of South Cotabato based on the web and mobile platforms. Modernizing document archiving, tracking, and updating aims to provide more efficiency and transparency to the government’s document management problems.

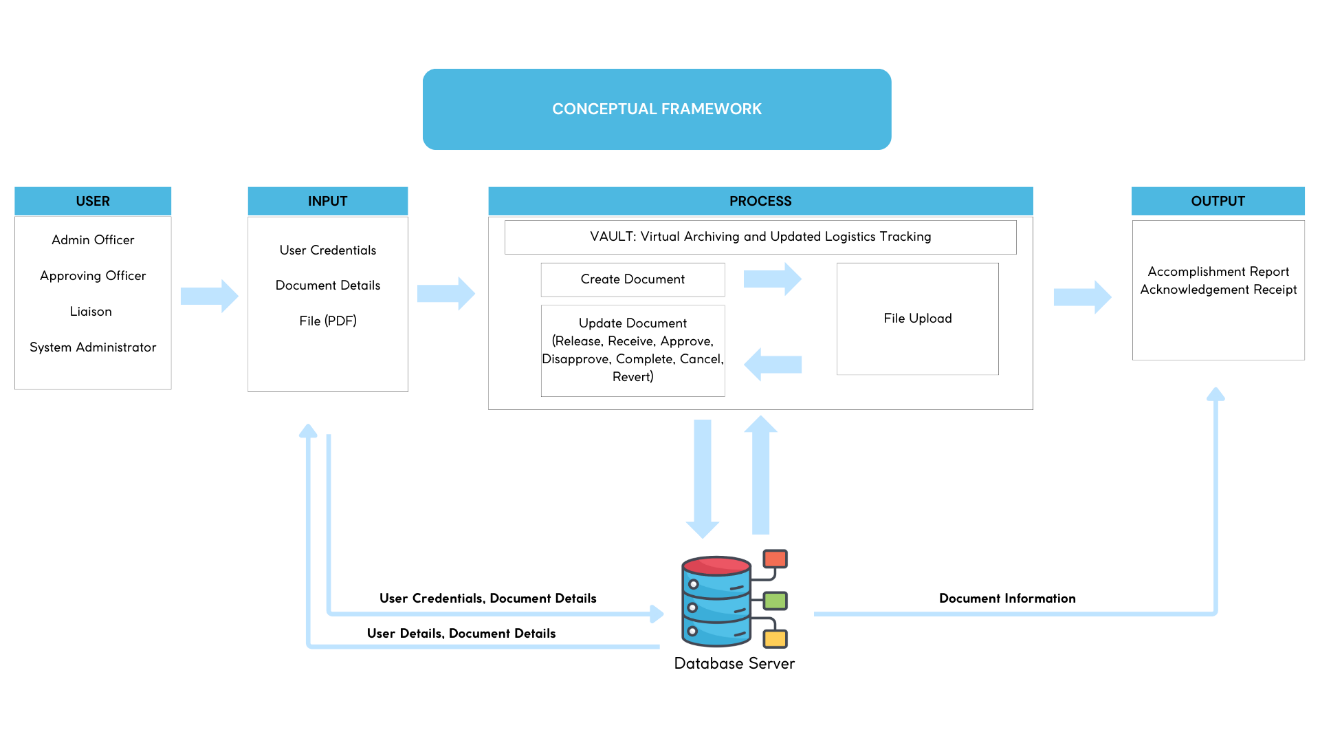
The study’s specific objectives are:

1. Development of the following modules for Web Applications:
   1. Authentication/Login module
   2. Document Management and Tracking
   3. User Management module
   4. Office Management module
   5. Document Type Management module
   6. Reports Generation
      1. Employee’s Accomplishment Report
      2. Acknowledgment Receipt
2. Development of the following modules for Mobile Applications:
   1. Authentication/Login module
   2. Document Management and Tracking
3. Implement the following features in the mobile app for the efficiency of the system
   1. QR Code Scanning for Document Retrieval
   2. In-App Notification

## Concept of the Study

## Conceptual Framework

Figure 1 Conceptual Framework



*Describing content in the figure would appear here*

Users

There are two types of users; the doctor and the patient. Users must first create an account before they can access the main system. They can now access the entire system after creating an account.

Input

Users will register their accounts, which will be saved on the database server, and they will be able to retrieve them by logging in to their accounts to use the system.

Process

**Users**  
 There are four types of users: the Admin Office, Approving Officer Liaison and System Administrator. All account types can create transactions, but only the approving officer and system administrator can perform approvals. Only the system administrator can register users and perform other maintenance tasks.

**Input**

Users will have to provide their username and password to access the system’s features. Document Details are required when creating a document, it will also require a file in pdf format to be uploaded

**Process**

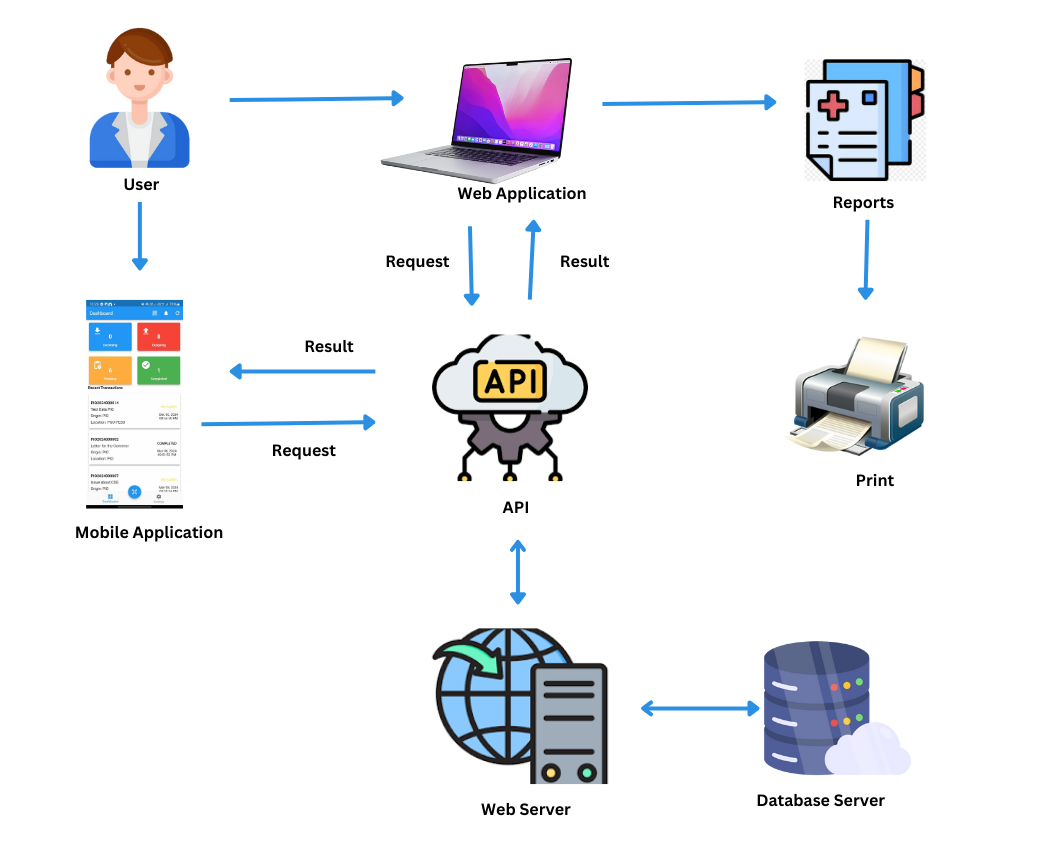
The process starts by creating a document by providing document details, following a file upload. After the creation, the user who created the transaction can now release the document and will have to provide the purpose of the transaction, and other users from different offices will be able to receive it. This action can be performed by all types of user account. During the receiving process, the receiver must provide the liaison’s (or any user account type) authentication code. This process is required to ensure that the released document will be received by the desired office. If necessary, the document has to be approved by the approving office to complete the transaction.

**Output**

The output will be the acknowledgement receipt, accomplishment report and the document’s virtual copy. Which are all saved in the database server and can be retrieve anytime by the users.

### System Architecture

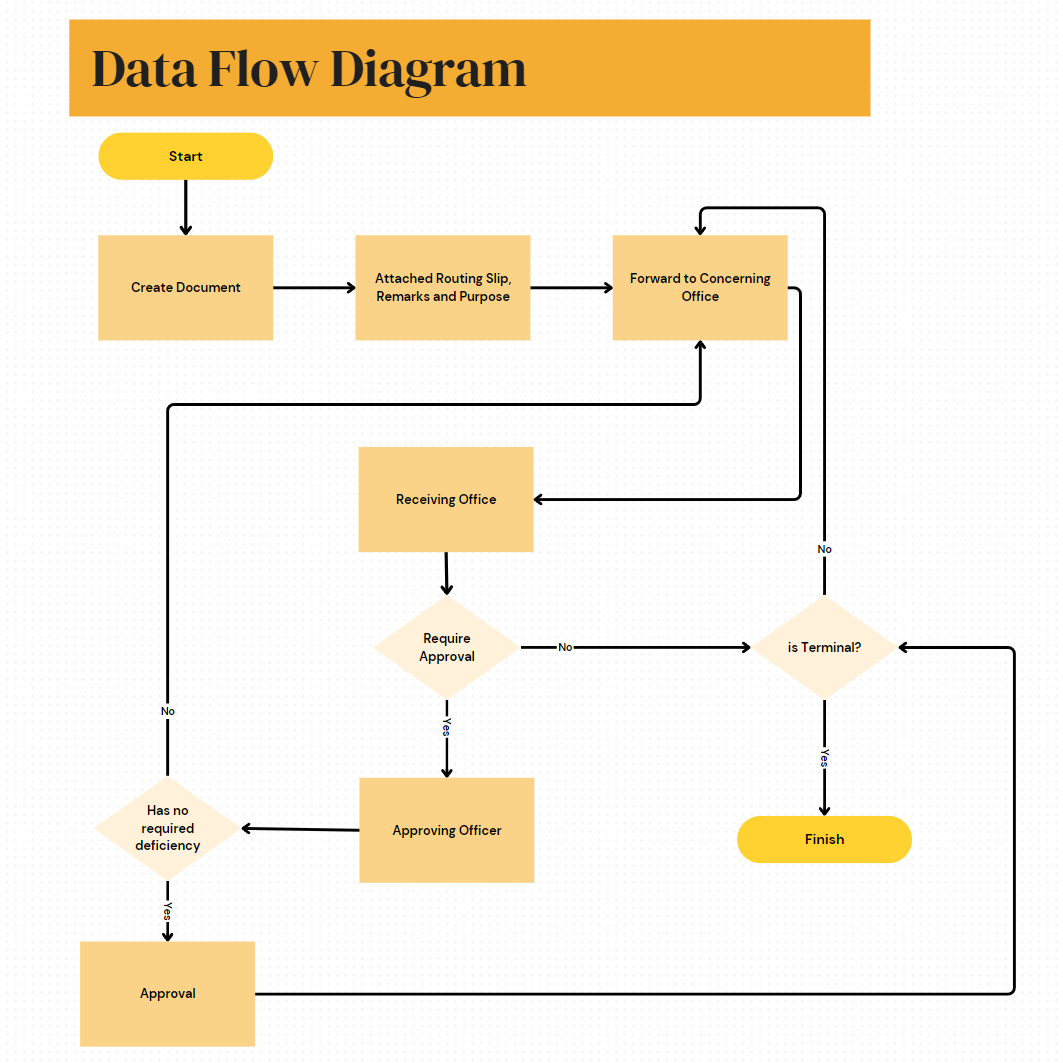
Figure 2 System Architecture



The System Architecture, showcasing the interaction between users, applications, and backend components. Users can access the system via a web or mobile application. The web application enables requests, results viewing, and report generation, while the mobile app provides a user-friendly interface for similar functions. Both applications communicate with the backend through an API connecting them to the web server and database server. The web server processes requests, retrieves data from the database, and delivers responses, ensuring seamless and efficient system operation.

### Data Flow and Processing

Figure 3 Data Flow Diagram



The data flow diagram illustrates the flow of document processing within the VAULT system. The process starts with the creation of a document, followed by attaching a routing slip, remarks, and the document's purpose. The document is then forwarded to the concerning office for processing. Depending on the requirements, it is either sent for approval or passed directly to the receiving office. If approval is needed, the document moves to the approving officer. Upon review, if no deficiencies are found, the document is approved. Finally, the system checks whether the document is terminal; if so, the process concludes. If not, the document may continue in the workflow for further actions.

### Security and Authentication

**JSON Web Tokens (JWT)** - used for stateless authentication between the frontend and backend. Each request to the API must include a valid JWT token in the authorization header. The token contains encoded user identity, role, and permissions, ensuring secure access. Additionally, tokens have an expiration time to enhance security and prevent unauthorized access.

**Role-Based Access Control (RBAC) -** defines user roles such as Admin, Approving Officer, Liaison, and System Administrator. It ensures restricted access based on assigned permissions, preventing unauthorized modifications. Furthermore, it limits user actions based on predefined role privileges to enhance data security.

### Deployment and Scalability

The system will be deployed in Oracle Cloud Infrastructure (OCI), utilizing Microsoft Windows Server 2016 as the operating system for hosting the web application. The deployment environment includes:

* **Compute Instances:** The web and backend services will run on OCI Virtual Machines (VMs) with Windows Server 2016.
* **IIS (Internet Information Services):** The web server will be hosted on IIS, ensuring secure and reliable access to the system.
* **MS SQL Server 2016:** The database will be managed using Microsoft SQL Server 2016, hosted on a dedicated OCI instance to ensure data integrity and high availability.
* **Object Storage:** Documents and files will be stored securely in Oracle Cloud Object Storage, ensuring scalability and easy access.

To support system performance and future growth, the following scalability measures are implemented:

* **Horizontal Scaling of Microservices:** The system follows a microservices architecture, allowing independent scaling of services based on demand. Additional instances of specific services can be deployed on OCI as needed.
* **Load Balancing:** OCI's built-in **Load Balancer** will be used to distribute traffic across multiple instances, ensuring even resource utilization and preventing server overload.
* **Database Optimization:** The **MS SQL Server 2016** database will be optimized with indexing, stored procedures, and query performance tuning to handle large datasets efficiently.
* **Auto-Scaling:** OCI provides **Auto-Scaling Groups**, which can dynamically adjust the number of compute instances based on real-time traffic and resource consumption.
* **Disaster Recovery & Backups:** OCI’s **Backup and Disaster Recovery services** will be configured to ensure data integrity and system availability in case of failures.

# Methods

## Hardware

* PC/Laptop
  + Windows 8 or Higher
  + RAM: at least 8GB
  + Updated Web Browser
* Android
  + Android 7.0 (Nougat) or higher
  + RAM: 8GB or higher

## Software

|  |  |
| --- | --- |
| Visual Studio 2022 | Used for developing, debugging, testing, and deploying applications across various programming languages and frameworks. It provides an integrated environment with built-in version control, cloud services, and advanced development tools. |
| Visual Studio Code | Used as a versatile text editor for coding, with features like syntax highlighting, IntelliSense, Git integration, and support for extensions to enhance development efficiency across multiple languages. |
| Internet Information Services (IIS) | Used for hosting and managing websites, web applications, and services. It supports HTTP/HTTPS protocols and server-side scripting with ASP.NET, enabling secure and scalable web deployment. |

### Frontend

|  |  |
| --- | --- |
| Bootstrap | Used for designing responsive and mobile-first web interfaces. It provides built-in CSS and JavaScript components to create modern, visually appealing, and consistent UI elements across different screen sizes. |
| Flutter | Used for developing mobile and web components of the system with a single codebase. It enables fast development cycles, supports highly customizable UI elements, and ensures natively compiled performance across multiple platforms. |

### Backend

|  |  |
| --- | --- |
| ASP.NET | Used for developing dynamic web applications, services, and APIs. It provides a robust framework with built-in libraries for security, performance optimization, and efficient server-side processing, enabling scalable and maintainable web solutions. |

The system follows a microservices architecture, where functionalities are divided into independent services that communicate through APIs. RESTful APIs are implemented to enable seamless communication between the web, mobile applications, and backend services. This approach enhances scalability, maintainability, and flexibility while ensuring efficient data exchange.

### Database

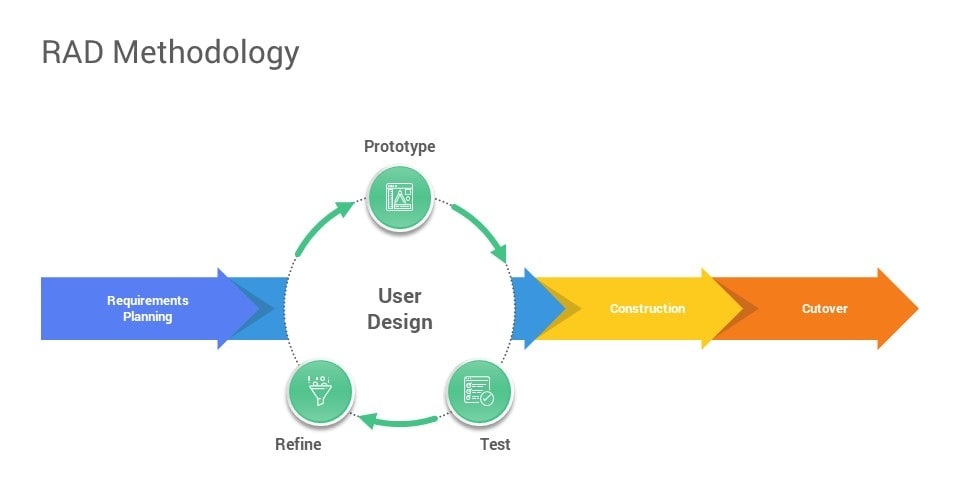
|  |  |
| --- | --- |
| MS SQL Server | The system uses Microsoft SQL Server (MS SQL Server) as its primary Relational Database Management System (RDBMS) to store and manage structured data efficiently. MS SQL Server provides robust support for data integrity, security, and transactional consistency, making it suitable for handling critical business operations. |

# Procedures

This section describes the process of system development, from design to implementation.

## Design Methodology

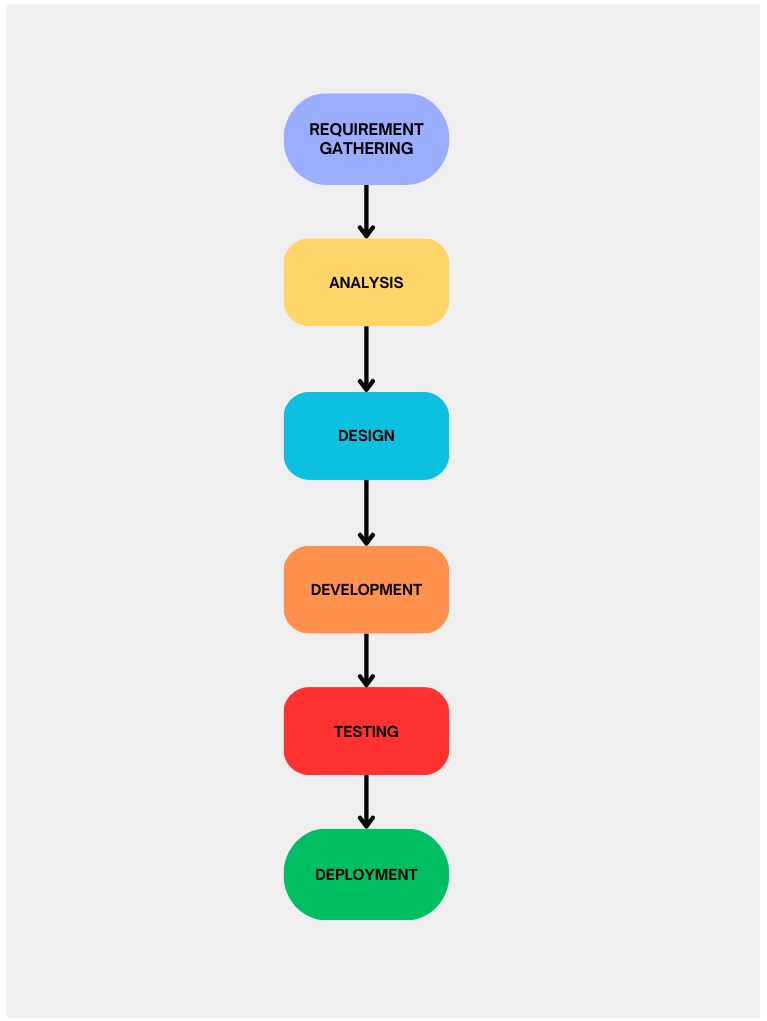
Figure 4 Rapid Application Development (RAD)



Rapid Application Development (RAD), a software development paradigm that prioritizes rapid and iterative development processes, allowing for faster product revisions and more user adaptability. RAD emphasizes end-user involvement in the design and development stages via collaborative workshops and iterative feedback sessions. This technique facilitates the rapid refinement and delivery of systems that successfully satisfy users’ needs. RAD is beneficial for projects involving extensive user engagement and when requirements are expected to change throughout development.

## Development Model

Figure 5 Rapid Application Development (RAD)



**Requirement Gathering** - This initial stage focuses on collecting all necessary information and requirements from stakeholders to understand the project’s scope, objectives, and constraints. It involves discussions with end-users, clients, and other stakeholders to capture their needs and expectations, forming the foundation for the rest of the development process.

**Analysis** - In the analysis stage, the gathered requirements are examined in detail to identify potential challenges, define system specifications, and create a clear project plan. This phase includes feasibility studies, risk assessments, and determining technical and resource needs to ensure the project is viable and aligns with stakeholder goals.

**Design** - The design stage involves creating a blueprint for the software solution, outlining the system architecture, data models, user interfaces, and other technical specifications. This phase focuses on translating the requirements into a structured plan that guides the development team in building the application, ensuring it meets both functional and non-functional requirements.

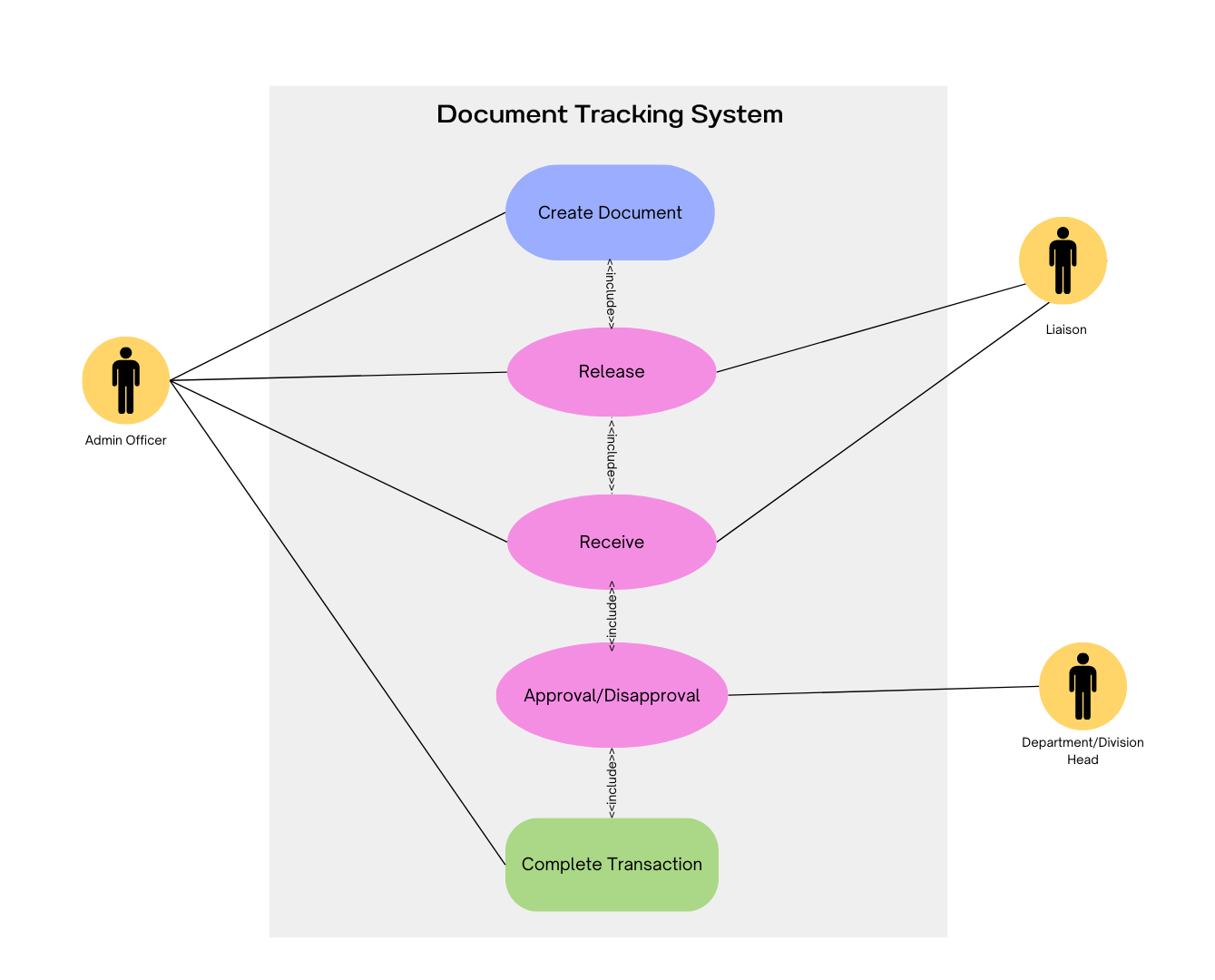
**Development** - During the development stage, the software's actual coding and construction occur. Developers build the system components based on the design specifications, integrating various modules and functionalities. This phase also involves initial testing and debugging to ensure the software works as intended.

**Testing** - The developed software is thoroughly tested in the testing stage to identify and fix any defects, errors, or inconsistencies. This phase involves various testing methods, such as unit testing, integration testing, system testing, and user acceptance testing, to validate that the software meets all specified requirements and functions correctly in different scenarios.

**Deployment** - The final stage involves deploying the tested software into the production environment, making it available for end-users. This phase includes installing the software, training users, and offering support as needed to ensure a smooth transition. After deployment, ongoing maintenance may be required to address any issues and implement future updates or enhancements.

## Requirements Analysis

Figure 6 Use Case of the Current System



### Functional Requirements

|  |  |
| --- | --- |
| **Web Application Modules** | |
| Authentication/Login Module | * The system shall allow users to log in securely with a unique username and password. * The system shall implement session management with automatic logout after a period of inactivity. * The system shall provide change password option. |
| Document Management and Tracking | * The system shall allow users to create, upload, edit, and delete documents. * The system shall enable users release, receive, approve, and disapprove documents. * The system shall track document movement between departments with timestamps and user actions. * The system shall provide real-time document status updates. |
| User Management Module | * The system shall allow System Administrators to create, update, and delete user accounts. * The system shall enable assigning roles and permissions to users. |
| Office Management Module | * The system shall allow the management of office information * The system shall allow System Administrators to add, update, and delete offices. |
| Document Type Management Module | * The system shall allow the creation, updating, and deletion of document types. |
| Reports Generation | * The system shall generate accomplishment reports for the user. * The system shall generate acknowledgment receipts for the received document. |
| **Mobile Application Modules** |  |
| Authentication/Login Module | * The system shall allow users to log in securely with a unique username and password. |
| QR Code Scanning for Document Retrieval | * The system shall allow users to scan QR codes to retrieve document details. * The system shall update document status upon successful QR code scanning. |
| In-App Notifications | * The system shall send real-time notifications for document status updates, approvals, and rejections. * The system shall display notification history within the app |

### Non-Functional Requirements

**High Availability and Fault Tolerance -** The system is designed to maintain 99.9% uptime, ensuring continuous access to document tracking and archiving. It will be deployed in Oracle Cloud Infrastructure (OCI) with failover strategies to prevent downtime. Regular automated backups will be implemented to protect against data loss, while load balancing will distribute traffic across multiple instances to prevent server overload.

**Scalable Architecture to Support High Traffic -** To support high traffic, the system follows a microservices architecture, allowing independent scaling of services. OCI’s Auto-Scaling will dynamically adjust computing resources based on demand, ensuring optimal performance. Additionally, database optimization techniques such as indexing and caching will be applied to improve data retrieval efficiency, especially in high-traffic scenarios.

**Security and Compliance -** Security and compliance measures will be strictly enforced. Role-Based Access Control (RBAC) will restrict system functionalities based on user roles, while JWT authentication will secure API requests and manage user sessions. All sensitive information, including documents and user credentials, will be encrypted in transit and at rest. To enhance accountability, audit logging will track all user actions and system events.

**System Performance -** The system will be designed for high performance, ensuring API response times remain under two seconds under normal load conditions. It will be capable of handling simultaneous transactions without performance degradation while offering optimized search and retrieval functions for quick document access.

For long-term sustainability, the system will follow a modular architecture, allowing for easy updates and feature enhancements. Proper code documentation and structured development practices will be maintained to ensure smooth maintenance and future extensibility.

## Requirements Documentation

### System Use Case Diagram

Figure 7 Use Case of the Propose System



### User Stories

**Admin Officer**

* As an Admin Officer, I want to log in securely using my unique username and password so that I can access the document management system. I need the ability to upload and manage documents to track their status and ensure they reach the correct recipients. Transparency is crucial, so I require access to document history and tracking information. Additionally, I want to receive notifications about document approvals, or rejections actions to stay updated and follow up when necessary. Lastly, I need to generate accomplishment reports on document movements between departments to monitor efficiency and provide updates to management.

**Approving Officer**

* As an Approving Officer, I want to log in securely so that I can review documents assigned to me. I need access to detailed document history so that I can make informed approval decisions. To ensure proper validation, I require the ability to approve or reject documents based on their content. When rejecting a document, I want to add comments or feedback so that the submitting officer understands the reason for rejection and can make necessary revisions.

**Liaison**

* As a Liaison, I want to log in securely so that I can update document statuses while transferring documents between offices. I need the ability to scan QR codes on documents to quickly retrieve document details and update their status in real time. To ensure proper tracking, I want access to the document history so that I can verify its movement and handling. Additionally, I need the capability to generate accomplishment reports on document movements to provide accurate records of my activities and ensure accountability in document transfers.

### Acceptance Criteria

**Web Application Modules**

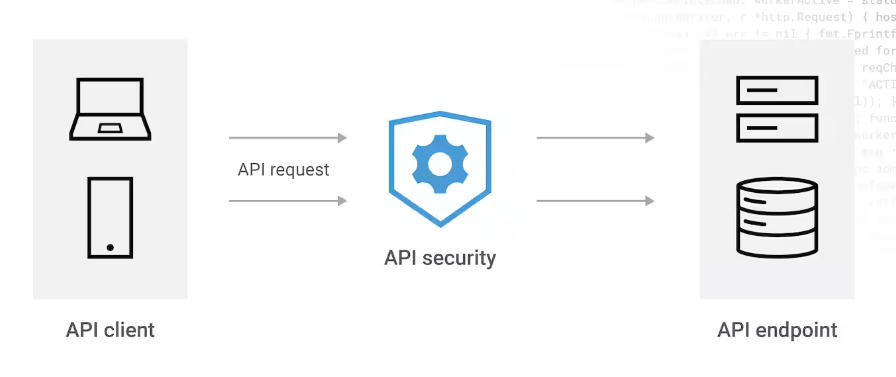
* **Authentication/Login -** Secure login with a unique username and password, session timeout after **15 minutes**, and password change option.
* **Document Management and Tracking -** Users can **create, upload, edit, delete, release, receive, approve, and disapprove** documents with real-time status updates and timestamped tracking.
* **User Management -** System Administrators can **create, update, and delete** user accounts with **role-based access control (RBAC)**.
* **Office Management -** System Administrators can **add, update, and delete** office information.
* **Document Type Management -** Ability to **create, update, and delete** document types.
* **Reports Generation -** Generate **accomplishment reports** and **acknowledgment receipts** in pdf form.

**Mobile Application Modules**

* **Authentication/Login -** Secure login using username and password.
* **QR Code Scanning -** Scan QR codes to **retrieve document details** and update status in real-time.
* **In-App Notifications -** Receive **real-time updates** on document status, approvals, and rejections with **notification history**.

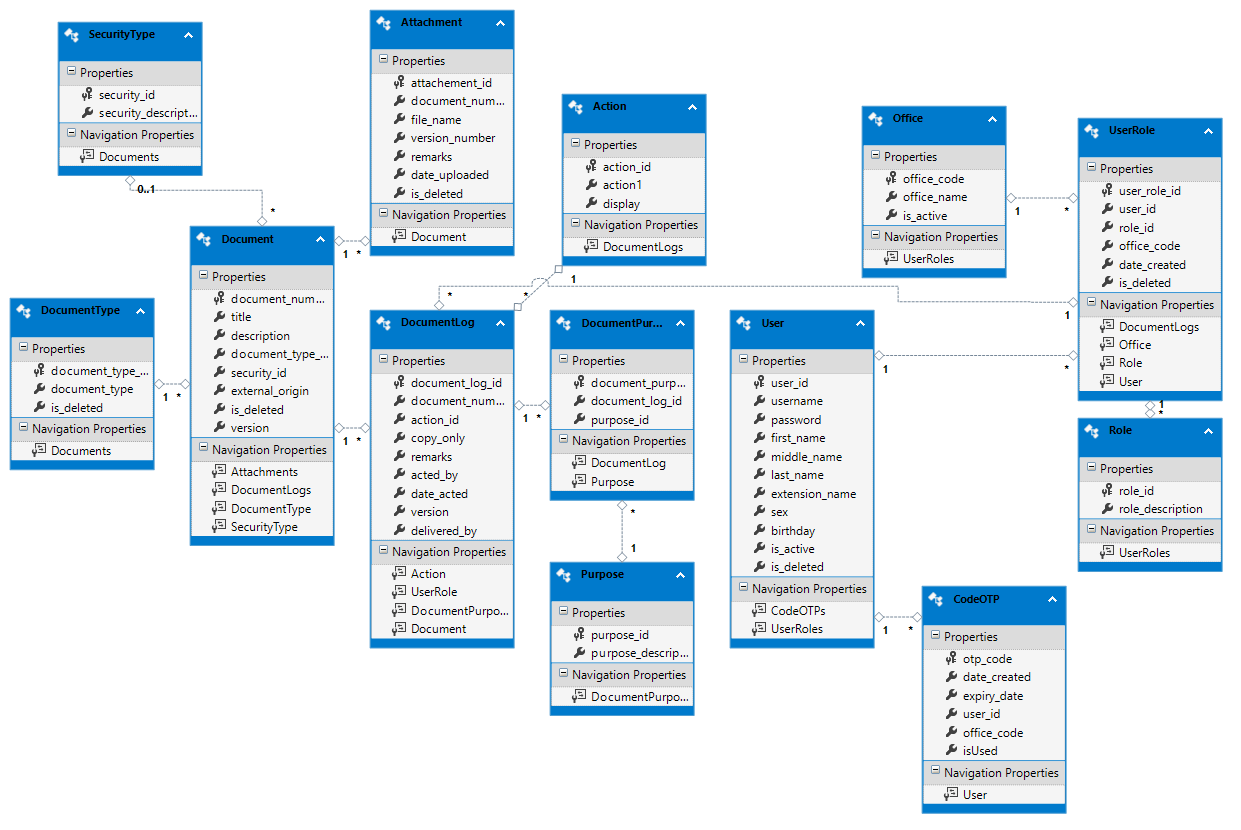
## System Design

### Architectural Design

**Figure 8** Architectural Design

The system follows a Three-Tier Architecture with three layers: Presentation (API Client), Application (API & Web Server), and Data (Database Server). The API Client (web browsers, mobile apps) interacts with the API & Web Server, which handles business logic, security (JWT, OAuth), and request processing. The Web Server retrieves and stores data in the Database Server (MS SQL Server). This design ensures scalability, security, and maintainability, allowing independent upgrades while optimizing performance with security measures like rate limiting and encryption.

### Database Design

**Figure 9** Entity Relationship Diagram

**Data Dictionaries**

The following tables show the list of concept databases. It shows the properties, data types, length and field description to be included in the database to ensure that the different databases capture the required information.

Table 1  
*Document Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| document\_number | varchar(50) | Document number of the document | Primary Key |
| title | varchar(100) | Title of the document | Not Null |
| description | varchar(255) | Description of the document | Not Null |
| document\_type\_id | smallint | Document type of the document | Not Null, Foreign Key from DocumentType Table |
| security\_id | smallint | Security type of the document | Not Null, Foreign Key from SecurityType Table |
| external\_origin | varchar(255) | External origin of the document | Allow Null |
| is\_deleted | bit | Status of document | Not Null |
| version | bit | Version of the document, used to concurrency | Allow Null |

Table 2 shows document logs, which holds every transaction made in every document that has been created.

Table 2  
*DocumentLog Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| document\_log\_id | varchar(100) | Transaction log id of the document | Primary Key |
| document\_number | varchar(50) | Document number of the document | Not Null, Foreign Key from Document Table |
| action\_id | smallint | Action type of the document | Not Null, Foreign Key from Action Table |
| copy\_only | bit | Release type of document | Not Null |
| remarks | varchar(255) | Remarks for the transaction log | Not Null |
| acted\_by | int | User that acted the transaction | Not Null, Foreign Key from UserRole Table |
| date\_acted | datetime | Date and time of the transaction | Not Null |
|  |  |  |  |
| delivered\_by | int | User who delivered the document | Allow Null |
| version | bit | Version of the transaction, used to concurrency | Allow Null |

Table 4 show document type table, it is a reference table which holds a list of document types.

Table 3  
*DocumentType Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| document\_type\_id | smallint | Type id of the document | Primary Key |
| document\_type | varchar(100) | Description of the type of document | Not Null |
| is\_deleted | bit | Status of the record | Not Null |

Table 4 show document purpose table, which holds the purpose of the document that was released or received in transaction log table.

Table 4  
*DocumentPurpose Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| document\_purpose\_id | varchar(100) | Purpose id of document purpose | Primary Key |
| document\_log\_id | varchar(100) | Transaction log of the document | Not Null |
| purpose\_id | smallint | Purpose of the document | Not Null, Foreign Key from Purpose Table |

Table 5 show the action table, action table is a reference table which hold a list of actions that can be perform by every user in the system.

Table 5  
*Action Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| action\_id | smallint | Action id of document | Primary Key |
| action | varchar(100) | Description of action | Not Null |
| display | varchar(100) | Description to be display of action | Not Null |

Table 6 shows the security type table, which holds a list of security types for the document. Each document has a security type: **public**, which is available to everyone with access to the system; **restricted**, which can only be viewed by the office that has a receiving copy; and **confidential**, which can only be viewed by the originating office.

Table 6  
*SecurityType Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| security\_id | smallint | Security Id of document | Primary Key |
| security\_description | varchar(50) | Security description | Not Null |

Table 7 shows the attachment table, which hold the details of the uploaded file. Each attachment may have different versions and is associated to the document created in the system.

Table 7  
*Attachment Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| attachement\_id | varchar(50) | Attachment id of the document | Primary Key |
| document\_number | varchar(50) | Document number of the document | Not Null, Foreign Key from Document Table |
| file\_name | varchar(100) | File name of the uploaded file | Not Null |
| version\_number | smallint | Version number of the uploaded file | Not Null |
| remarks | varchar(255) | Remarks for the file | Not Null |
| date\_uploaded | datetime | Date and time of uploading | Not Null |

Table 8 shows purpose table, this table holds the list of purpose that define the necessary action to be taken by the receiving office.

Table 8  
*Purpose Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| purpose\_id | smallint | Purpose id of the document | Primary Key |
| purpose\_description | varchar(100) | Description of the purpose | Not Null |

Table 9 shows the office table, which holds the list of offices available in the organization.

Table 9  
*Office Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| office\_code | varchar(50) | Office code of the office | Primary Key |
| office\_name | varchar(255) | Name of the office | Not Null |
| is\_active | bit | Status of the office | Not Null |

Table 10 show the user table, which hold the records of the user accounts created in the system. These accounts are used in authentication for accessing the system.

Table 10  
*User Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| user\_id | varchar(50) | User id of the user | Primary Key |
| username | varchar(100) | Username of the user | Not Null |
| password | varchar(100) | Password of the user account | Not Null |
| first\_name | varchar(100) | First name of the user | Not Null |
| middle\_name | varchar(100) | Middle name of the user | Allow Null |
| last\_name | varchar(100) | Last name of the user | Not Null |
| extension\_name | varchar(50) | Extension name of the user | Allow Null |
| sex | varchar(6) | Sex of the user | Not Null |
| birthday | datetime | Birthday of the user | Not Null |
| is\_active | bit | Status of the user account | Not Null |
|  |  |  |  |
| is\_deleted | bit | Status of the user account | Not Null |

Table 11 shows user role table, this table holds the role that assigned to each user. User role is associated to each user account.

Table 11  
*UserRole Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| user\_role\_id | int | User role id of the user | Primary Key |
| user\_id | varchar(50) | User id of the user | Not Null, Foreign Key from User Table |
| role\_id | smallint | Role id of the user | Not Null, Foreign Key from Role Table |
| office\_code | varchar(50) | Office code of the office | Not Null, Foreign Key from Office Table |
| date\_created | datetime | Date and time of user role | Not Null |
| is\_deleted | bit | Status of user role | Not Null |

Table 12 shows role table, it is a reference table that holds different type of role to be assigned in each user. Each role has their own functions and limitations in the system.

Table 12  
*Role Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| role\_id | smallint | Role id of role | Primary Key |
| role\_description | varchar(100) | Description of the role | Not Null |

Table 13 shows the OTP code table, this table holds the created one-time-password for every receiving transaction of the user. Each OTP is uniquely associated to each user for the purpose of identifying the person who delivered the document in a receiving office.

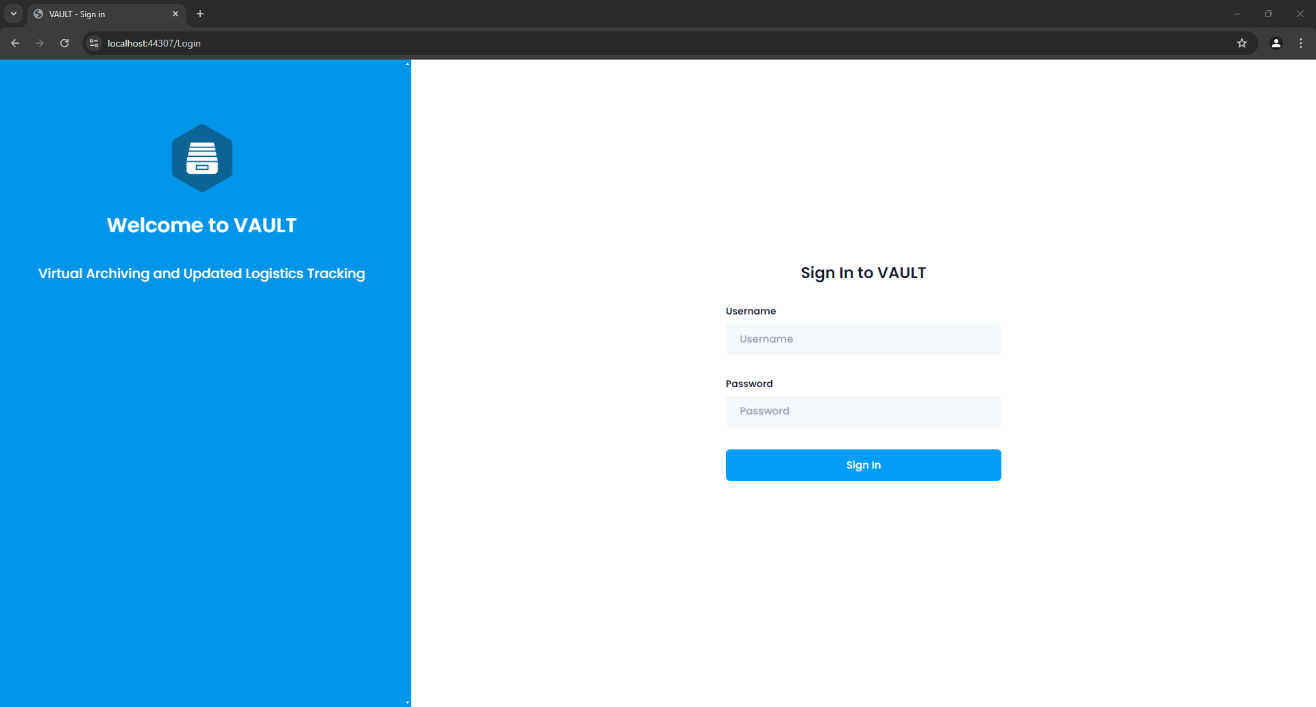
Table 13  
*CodeOTP Table*

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Constraints** |
| otp\_code | varchar(6) | Code of the otp | Primary Key |
| date\_created | datetime | Date and time of the otp | Not Null |
| expiry\_date |  | Expiration date of otp | Not Null |
| user\_id | varchar(50) | User id of user | Not Null, Foreign Key from User Table |
| office\_code | varchar(50) | Office code of the office | Not Null, Foreign Key from Office Table |
| isUsed | bit | Status of otp | Not Null |

### Application Flow

Web Application Modules

Figure 10  
*Login Module*



The login module, illustrated in Figure 10, is responsible for authenticating user credentials to grant access to the system.

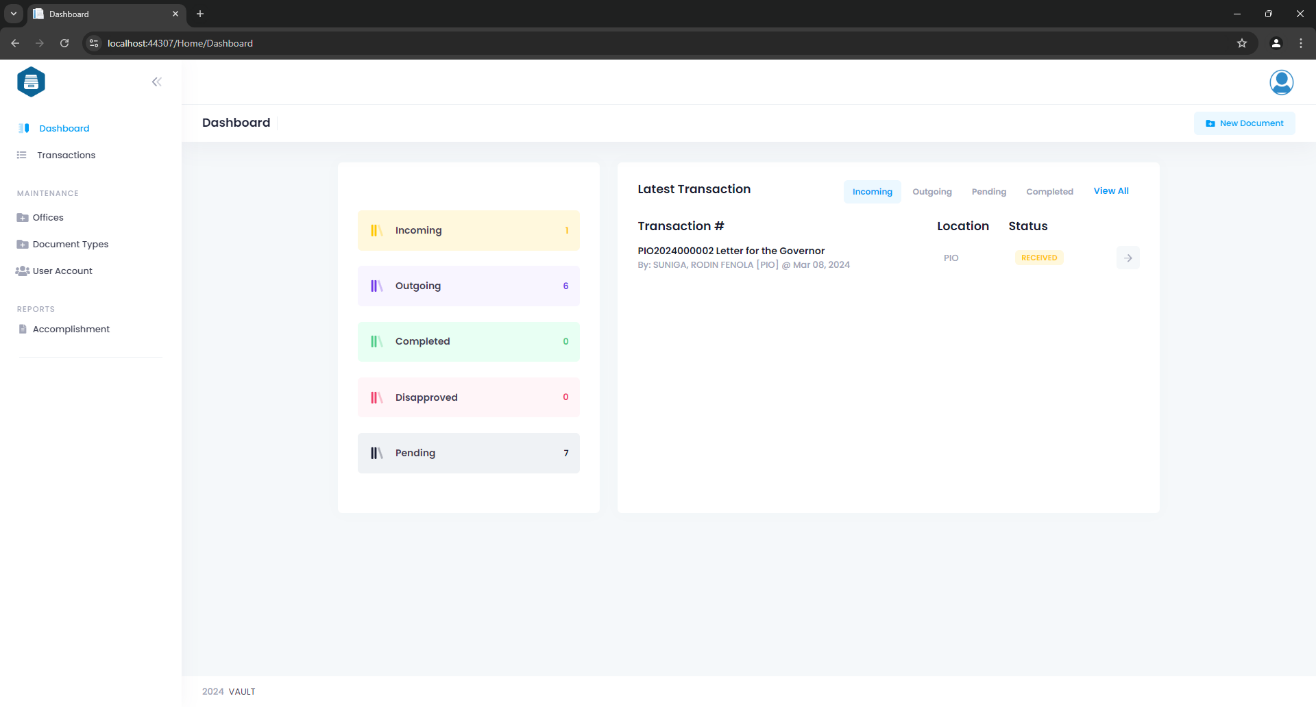


Figure 11  
*Dashboard*

The dashboard page will serve as a landing page of the system, as shown in Figure 11.

Figure 12  
*Create Document*

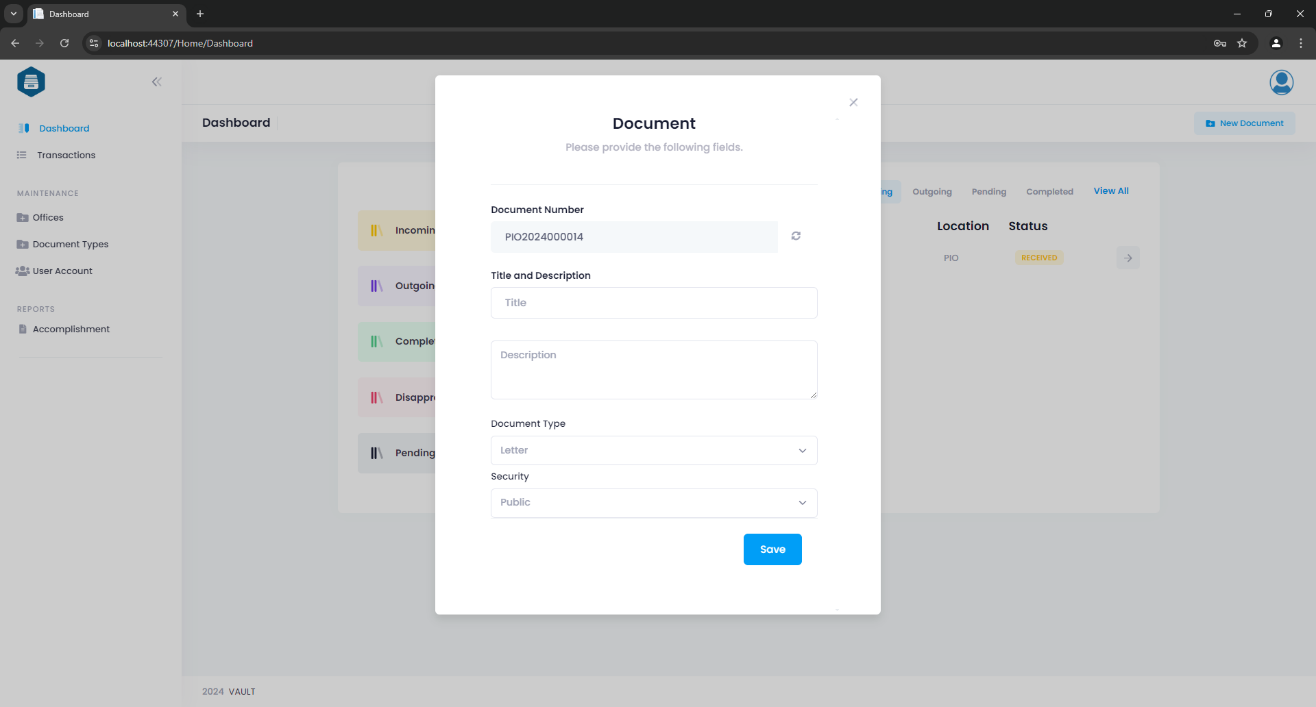


Figure 12 shows Create Document dialog, this allows users to create a new document by entering required details such as document title, description, type, and other relevant information. It includes input fields, dropdown menus, and buttons for saving the document creation process.

Figure 13  
*Document Details Module*

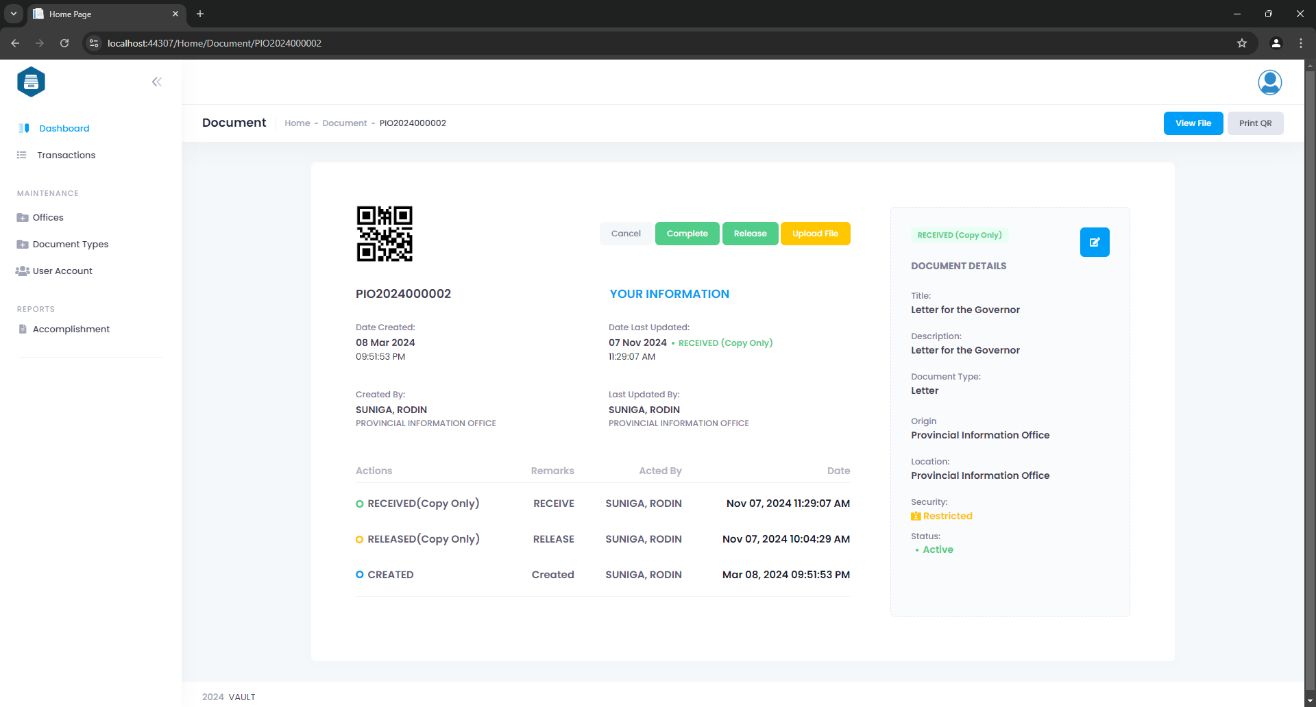


Figure 13 shows Document Details module, this page displays detailed information about a specific document, including its QR code, title, description, type, creator, creation date, origin, location and status. It also provides options for editing, uploading, or printing the document, along with a history log of updates or actions performed.

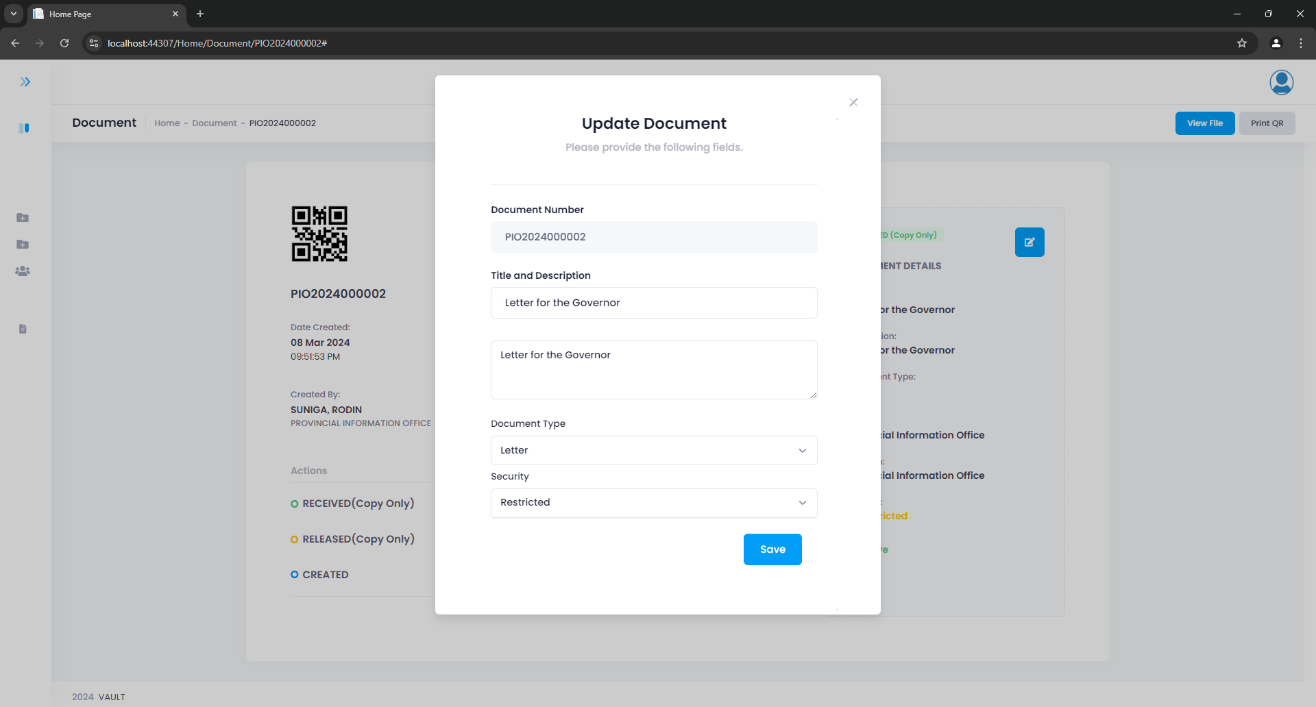


Figure 14  
*Update Document*

This Update Document dialog box shown in Figure 14 allows users to modify the details of an existing document. It includes editable fields for updating the title, type, description, and other attributes. The dialog also provides buttons to save changes or cancel the update process.

Figure 15  
*Transaction Module*

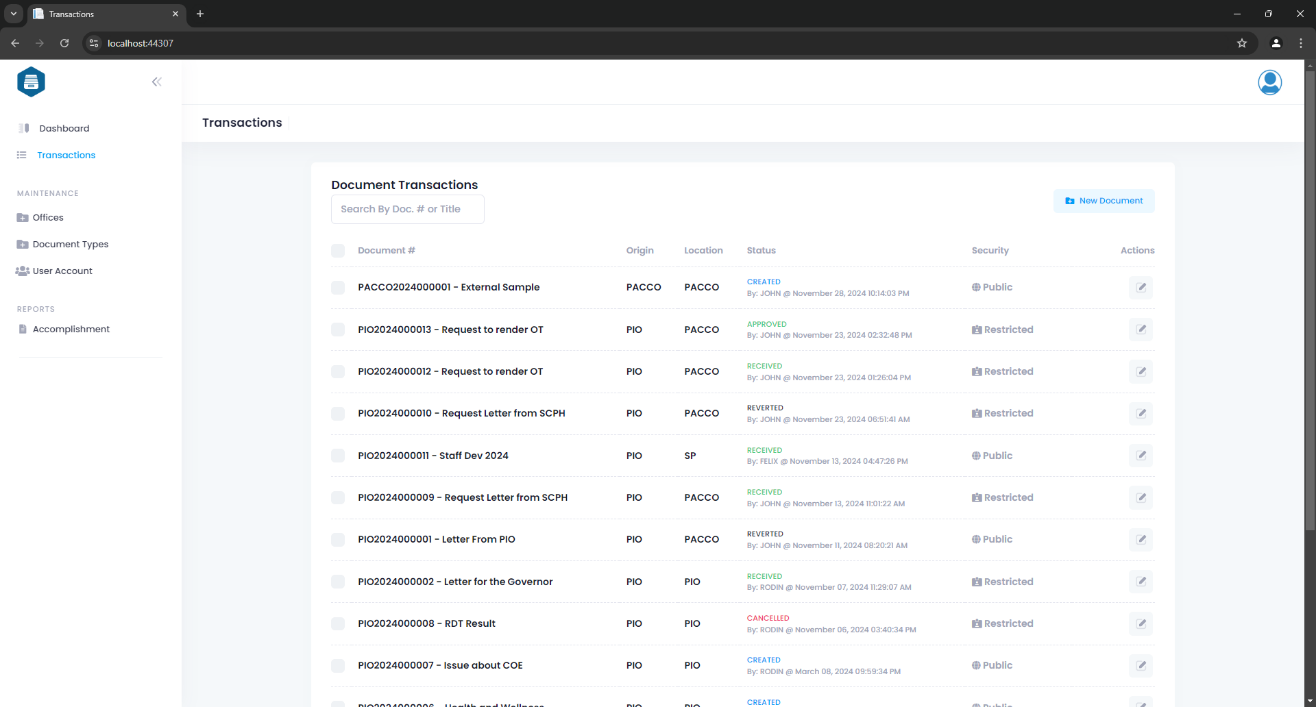


Figure 15 shows Transaction Module, which displays a list of transactions, with the ability to search by document number or title. Users can filter and view specific transactions based on the search criteria. A "Create" button is also available, allowing users to initiate the creation of a new document, providing a seamless experience for managing transactions.

Figure 16  
*Users Module*

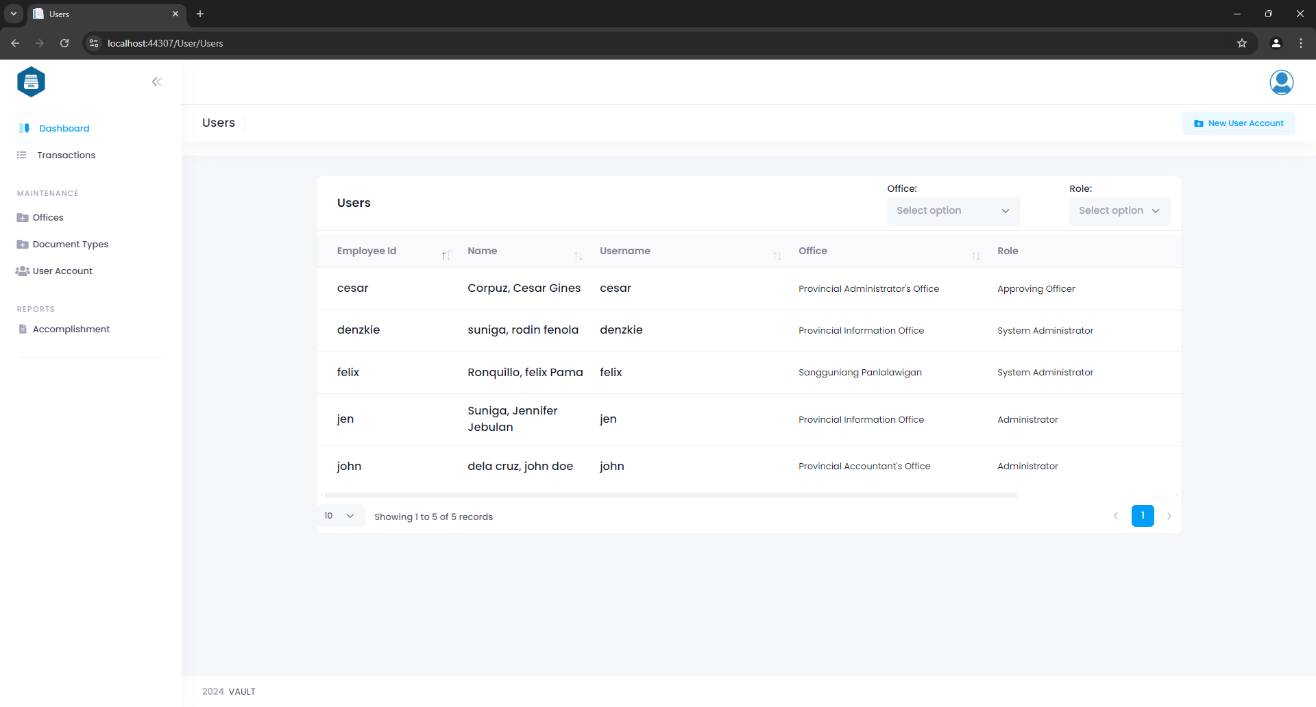


Figure 16 shows Users module. This module displays a list of users, along with their current status (active or inactive). It includes filters to search by office and by role, allowing users to easily refine the list based on specific criteria. Each user entry has options to view and update their details, such as name, role, and other information. Users can update information through an "Edit" button, ensuring efficient management of user profiles.

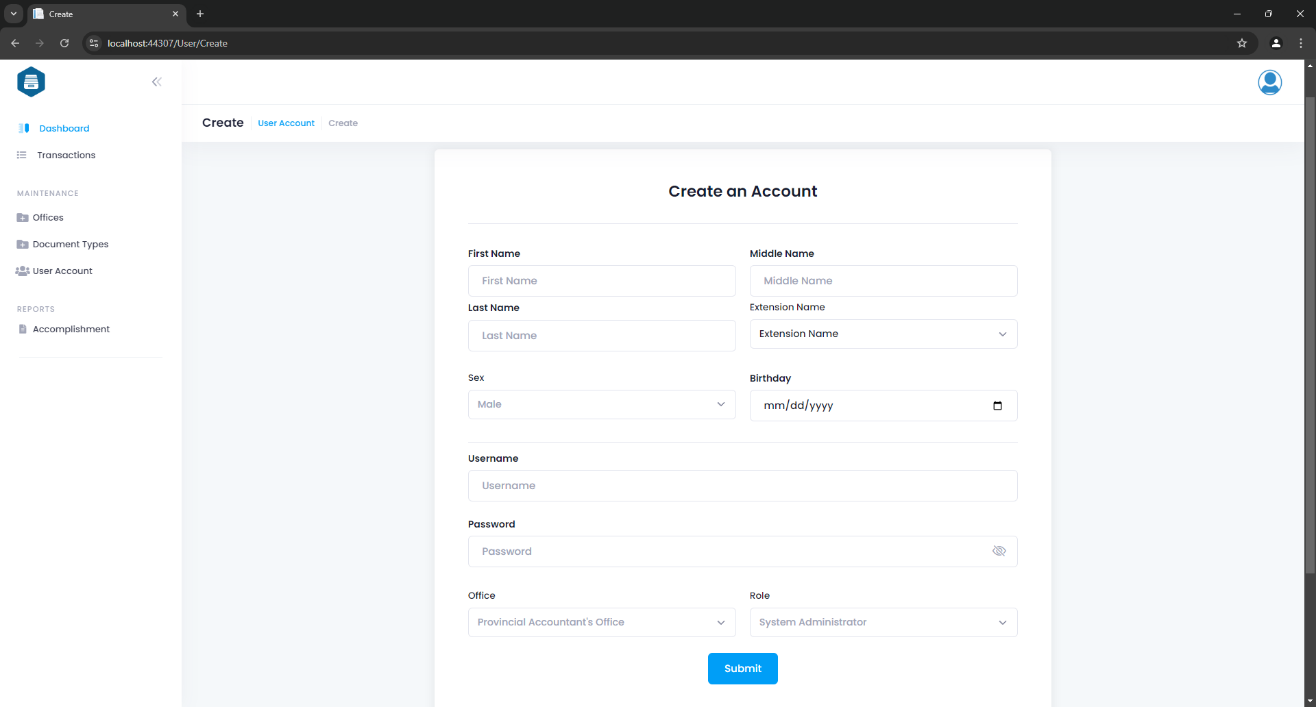


Figure 17  
*Create User*

Figure 17 shows Create User page. This page allows system administrator to create new user accounts by filling in required details such as name, username, password, role, office, and status (active/inactive). It includes input fields for entering user information and a "Save" button to create the user.

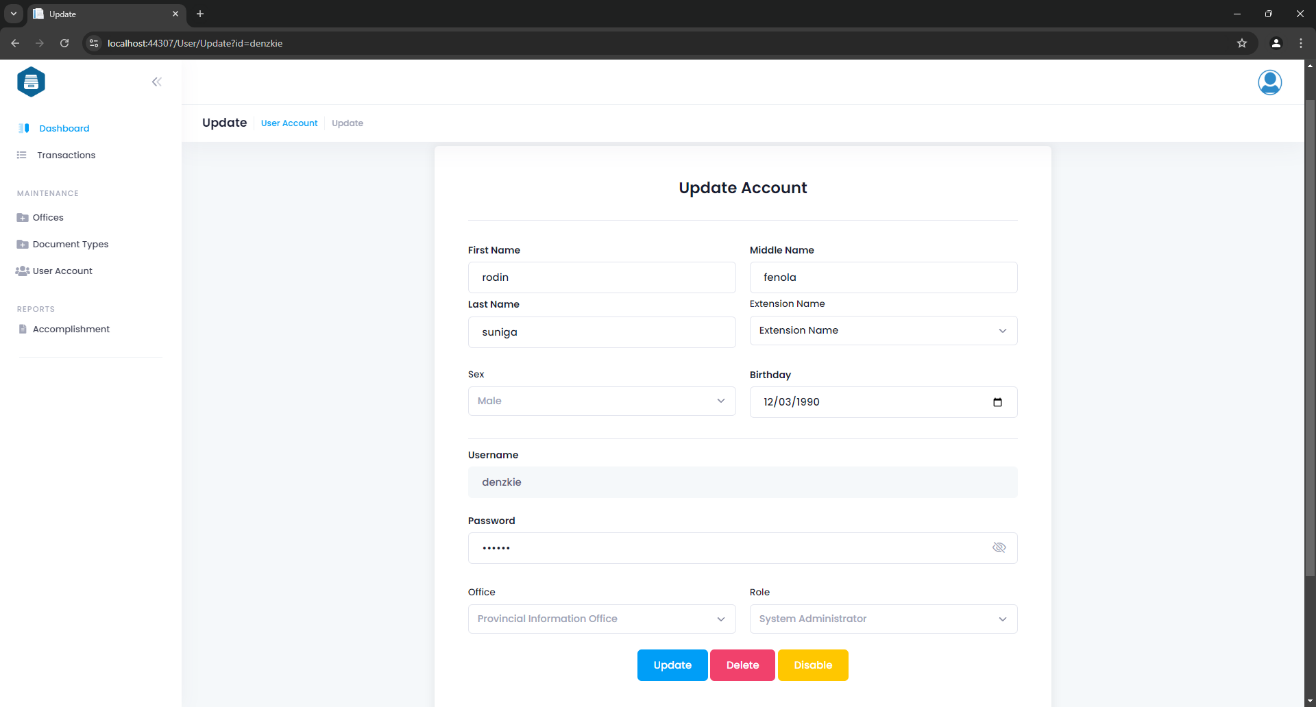
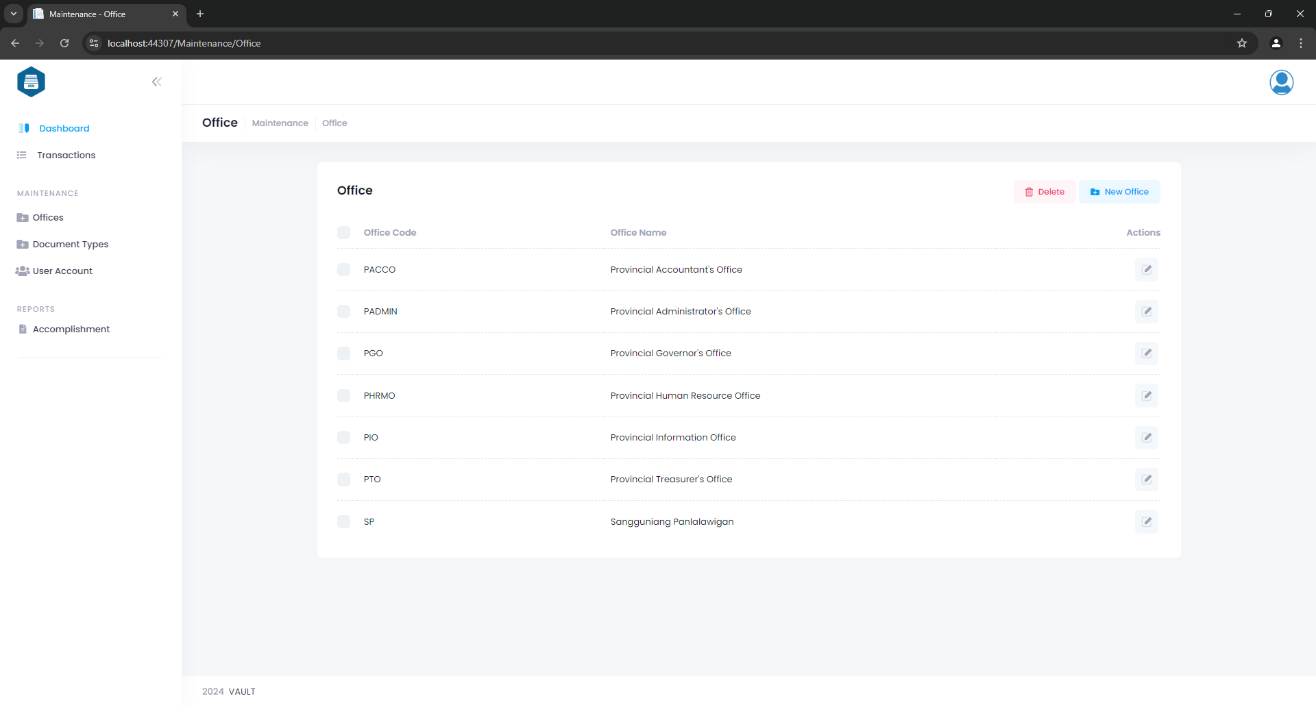


Figure 18  
*Update User*

Figure 18 shows Update User page. This page allows system administrator to update user details, including name, birthday, sex, role, office, and status. It provides options to enable or disable the user account, as well as removing the user from the system. The "Update" button allows the administrator to save any changes made to the user's information.

Figure 19  
*Office Module*



This Office Module as show in Figure 19 displays a list of offices within the organization. Each office entry includes options to delete, update, or view details. The "New Office" button allows administrators to add a new office, while the "Update" with Pencil icon button enables editing of existing office information. The "Delete" button provides the option to remove an office from the system.

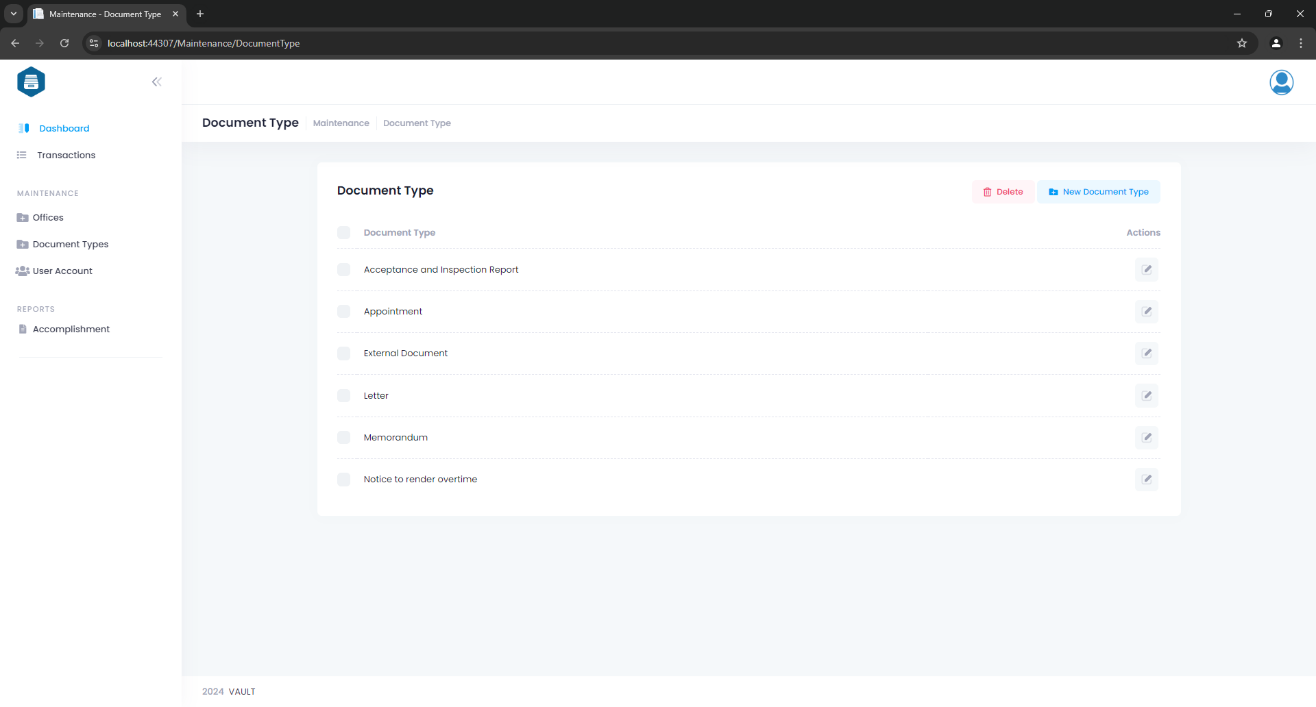


Figure 20  
*Document Type Module*

Figure 20 show Document Type module. This module displays a list of document types used within the system. Each document type entry includes options to delete, update, or view details. The "New" button allows administrators to add a new document type, while the "Update" with Pencil icon button enables editing of existing document type information. The "Delete" button provides the option to remove a document type from the system.

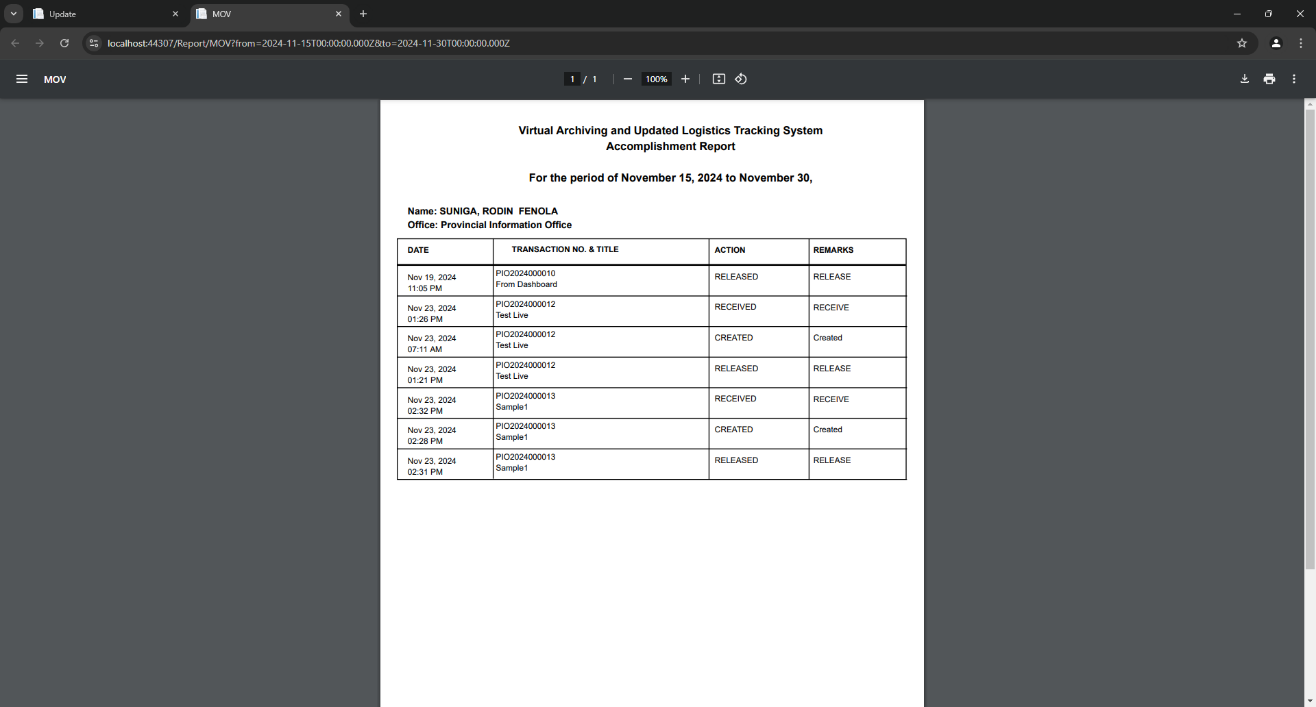


Figure 21  
*Accomplishment Report*

Figure 21 shows the generated Accomplishment Report. This report presents an overview of completed tasks, or activities within a specified timeframe of a specific user. It includes a list of the work accomplished, status and remarks.

Figure 22  
*Acknowledgement Receipt*

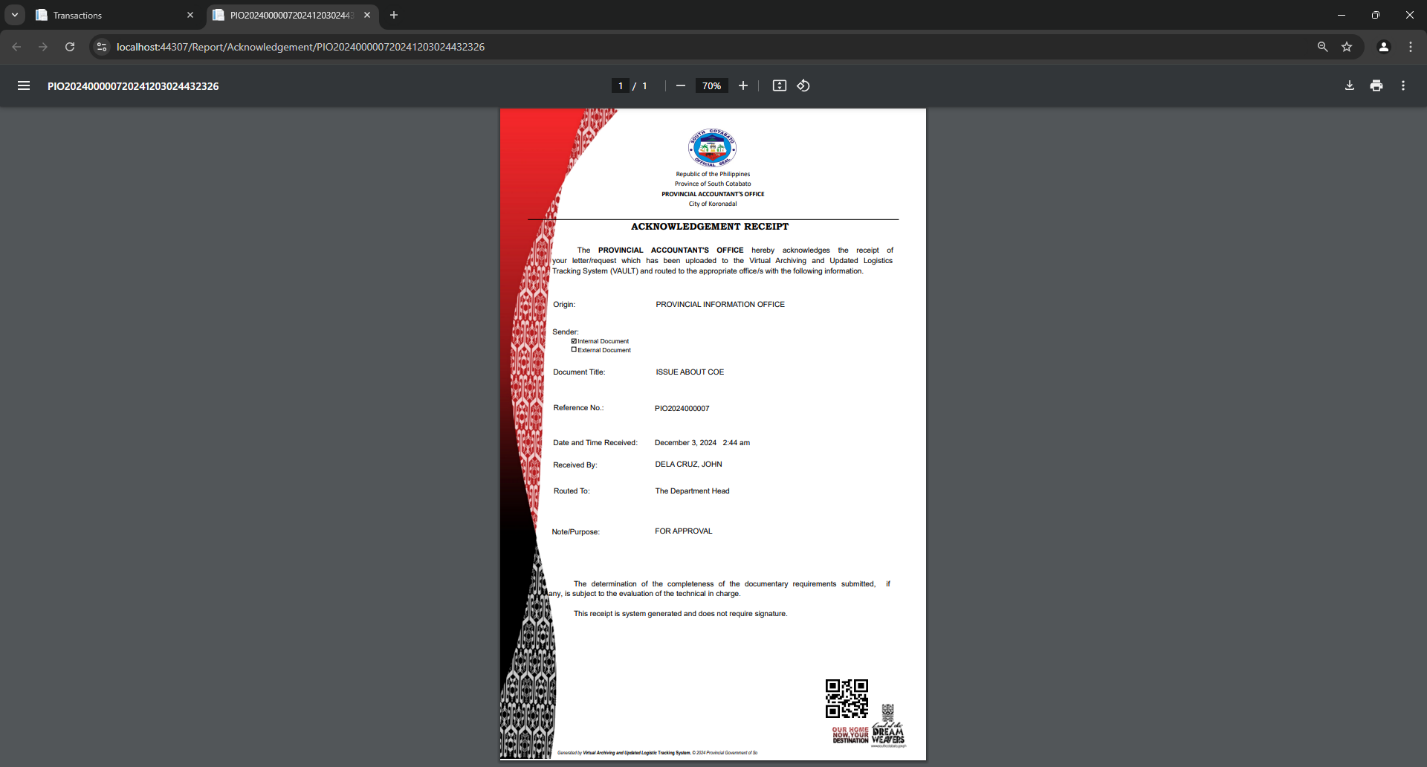
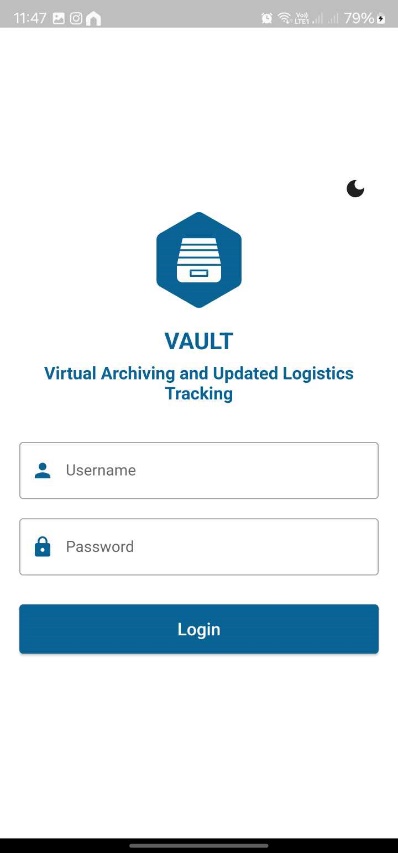


Figure 22 shows the acknowledgement for a transaction or document, confirming that a specific item has been received. It includes key details such as the origin of the document, date of receipt, transaction or document reference number, and other relevant notes.

**Mobile Application Modules**

Figure 23  
*Login Module*

****

The login module for mobile app, illustrated in Figure 23, is responsible for authenticating user credentials to grant access to the system.

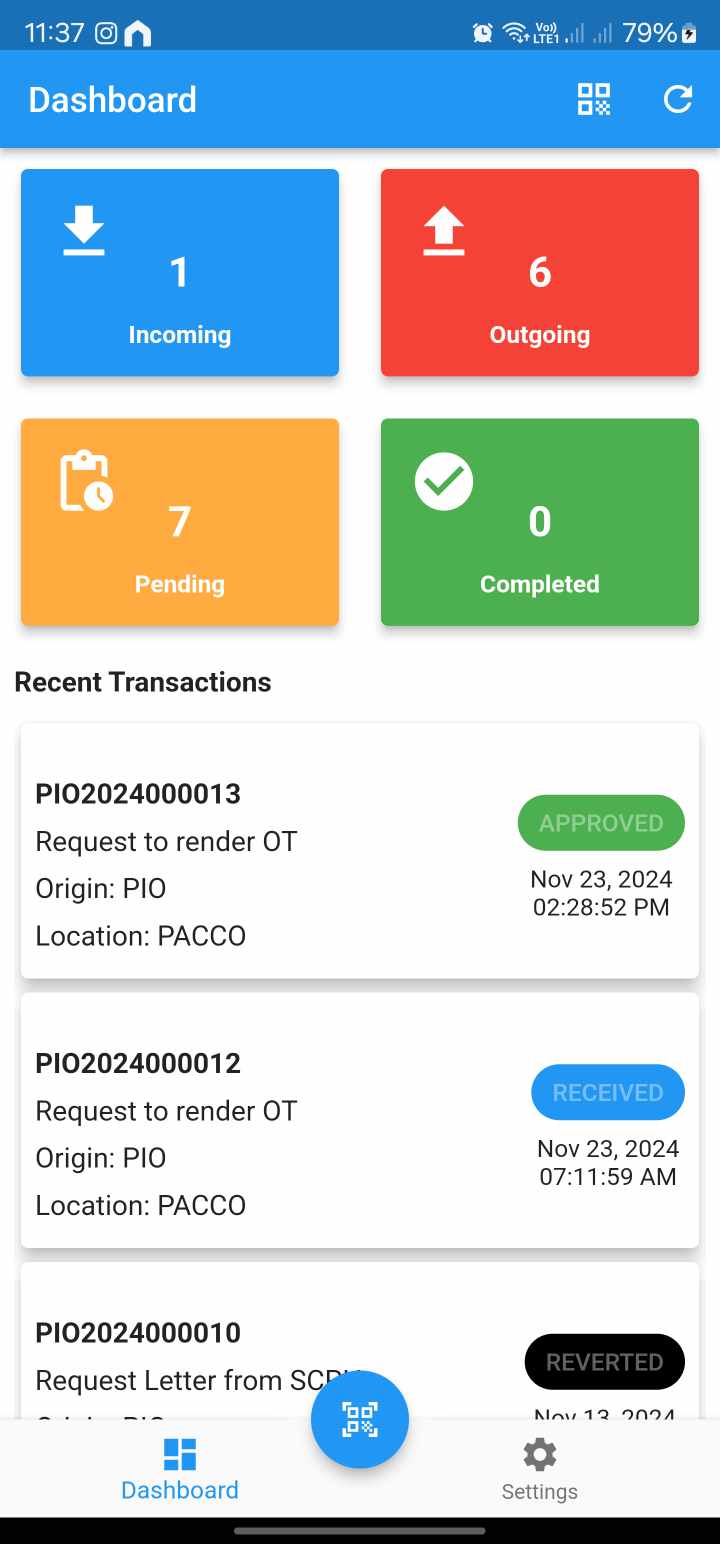
****

Figure 24  
*Dashboard*

The dashboard page will serve as a landing page of the system, as shown in Figure 24. It also shows the recent transactions to provide users with a quick overview of the latest activities, enabling them to track and monitor updates efficiently without navigating through multiple sections of the system.

Figure 25  
*QR Code Scanner*

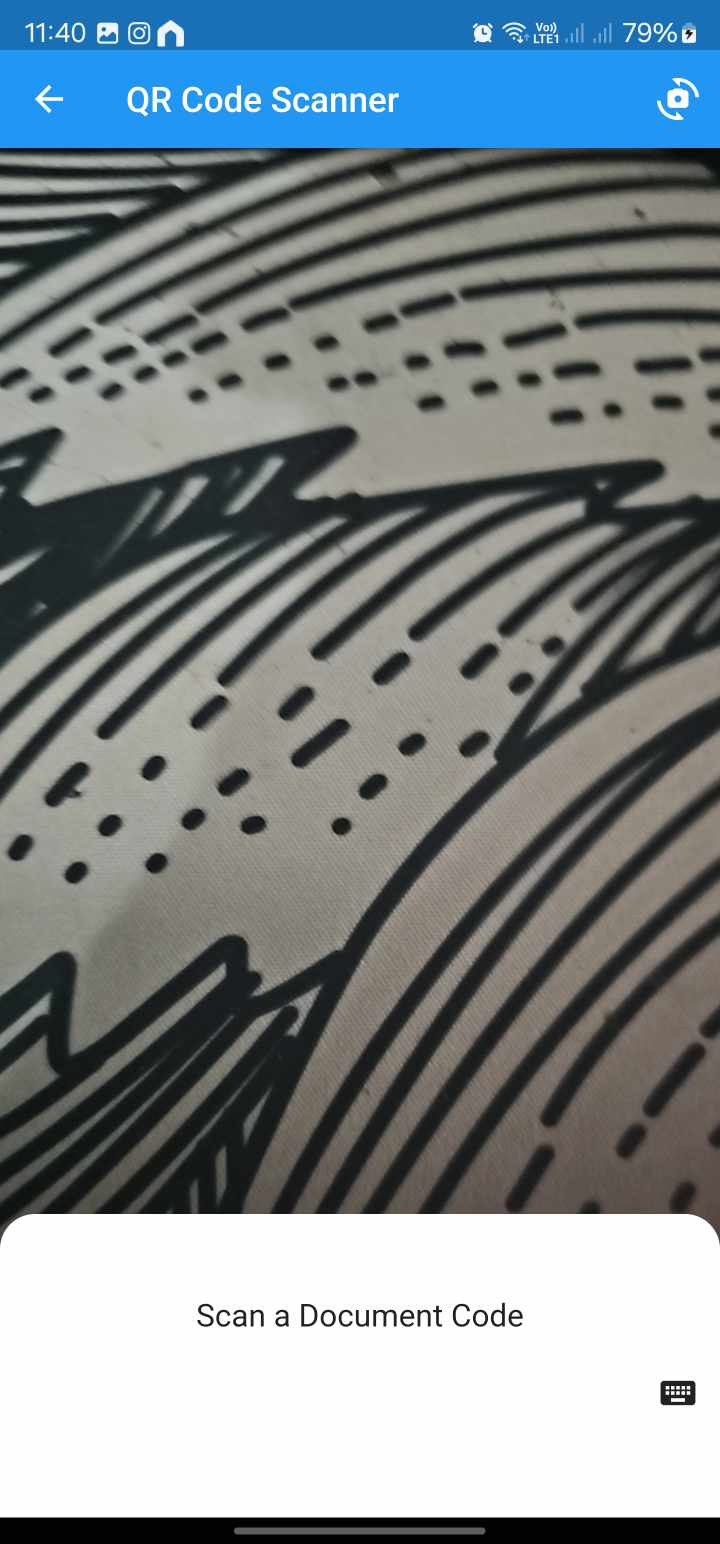
****

Figure 25 show the QR code scanner page. This page enables users to scan QR codes for quick access to document details. It uses the device's camera to capture and decode QR codes, automatically retrieving the associated information. This feature streamlines workflows by providing instant access to relevant data.

Figure 25  
*Details Page*

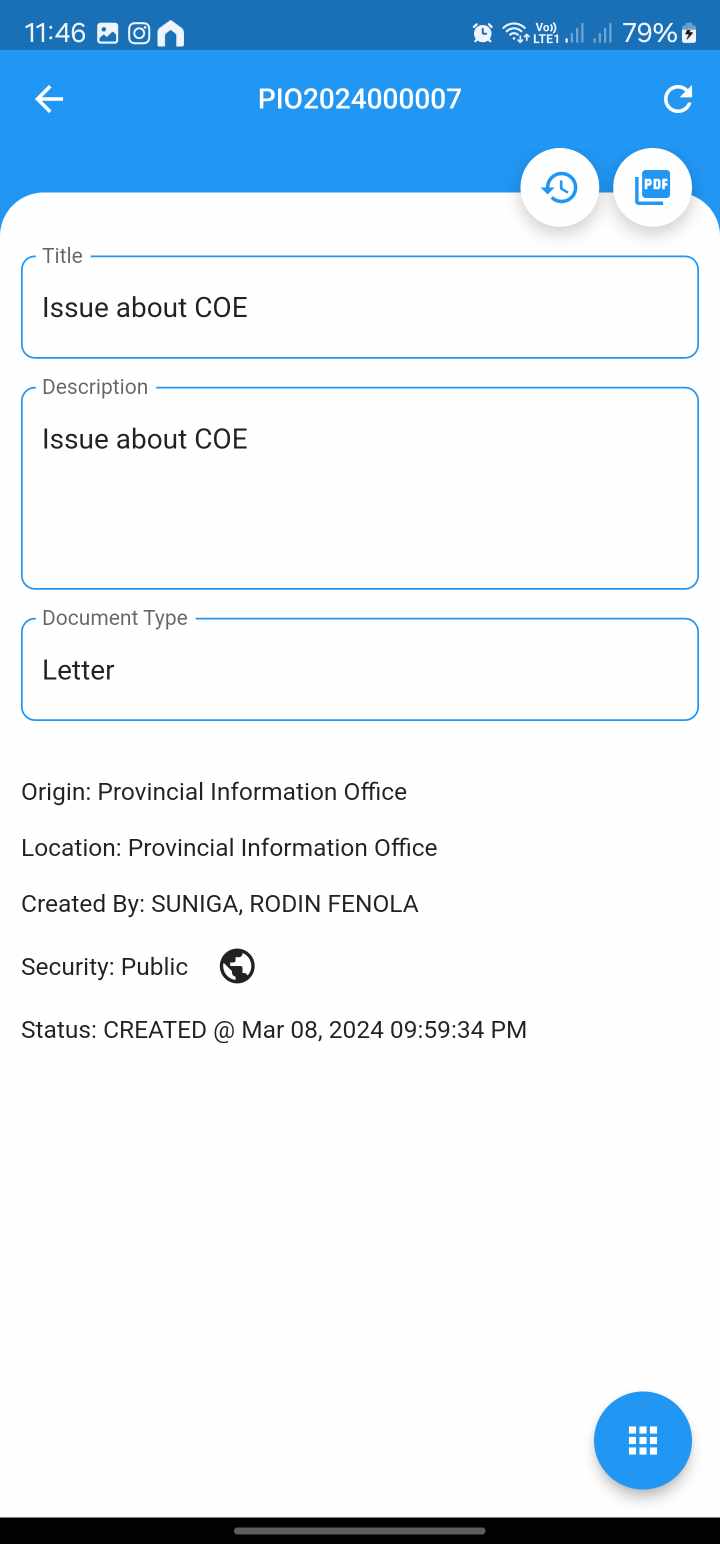


Figure 26 shows Document Details for mobile app, this page displays detailed information about a specific document, including its QR code, title, description, type, creator, creation date, origin, location and status. It also provides action necessary to update the status of the document.

Figure 27  
*Document Logs*

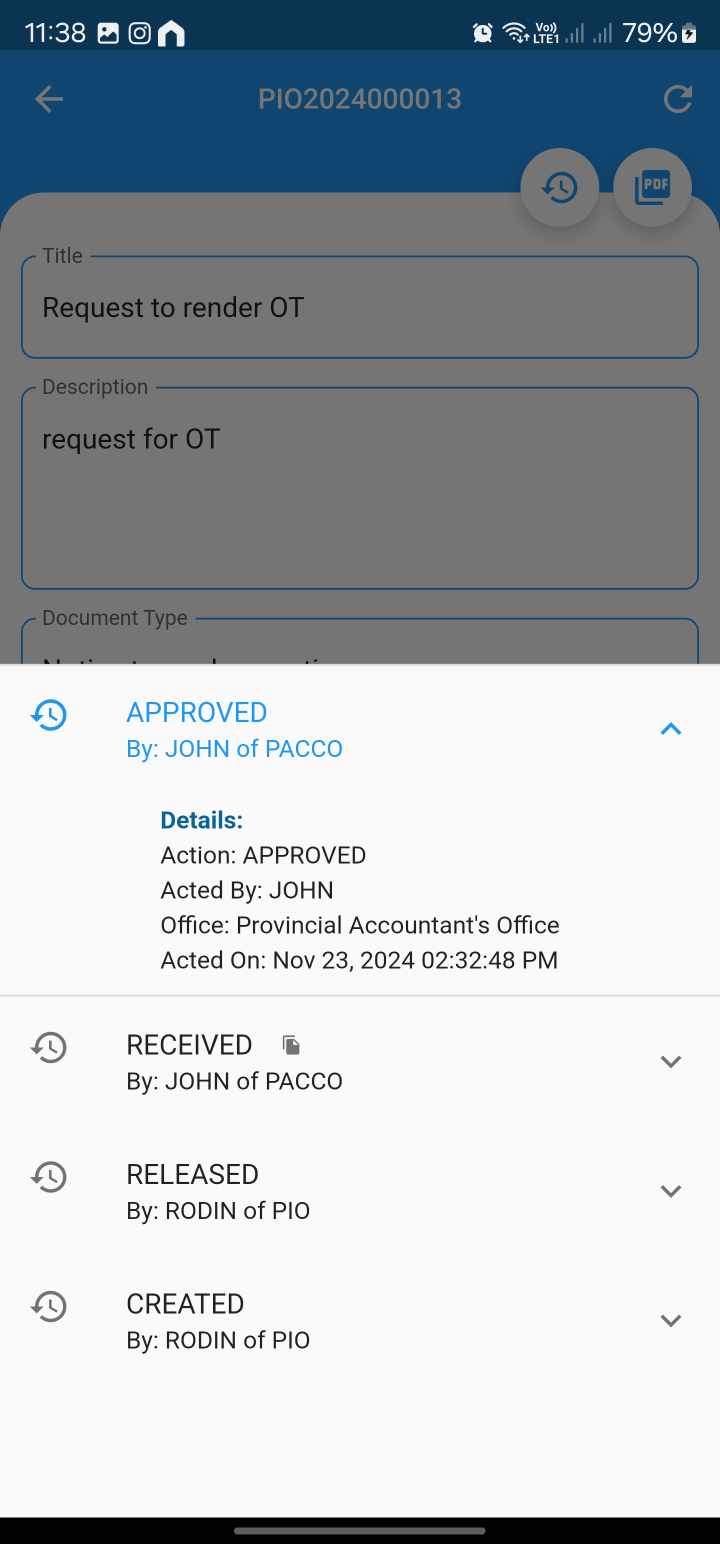
****

Figure 27 show the document logs or history of actions that was performed in the current document. This page displays a detailed history of actions and updates related to a specific document. It includes a timeline of events such as creation, release, receive, approvals.

Figure 28  
*OTP Generator*

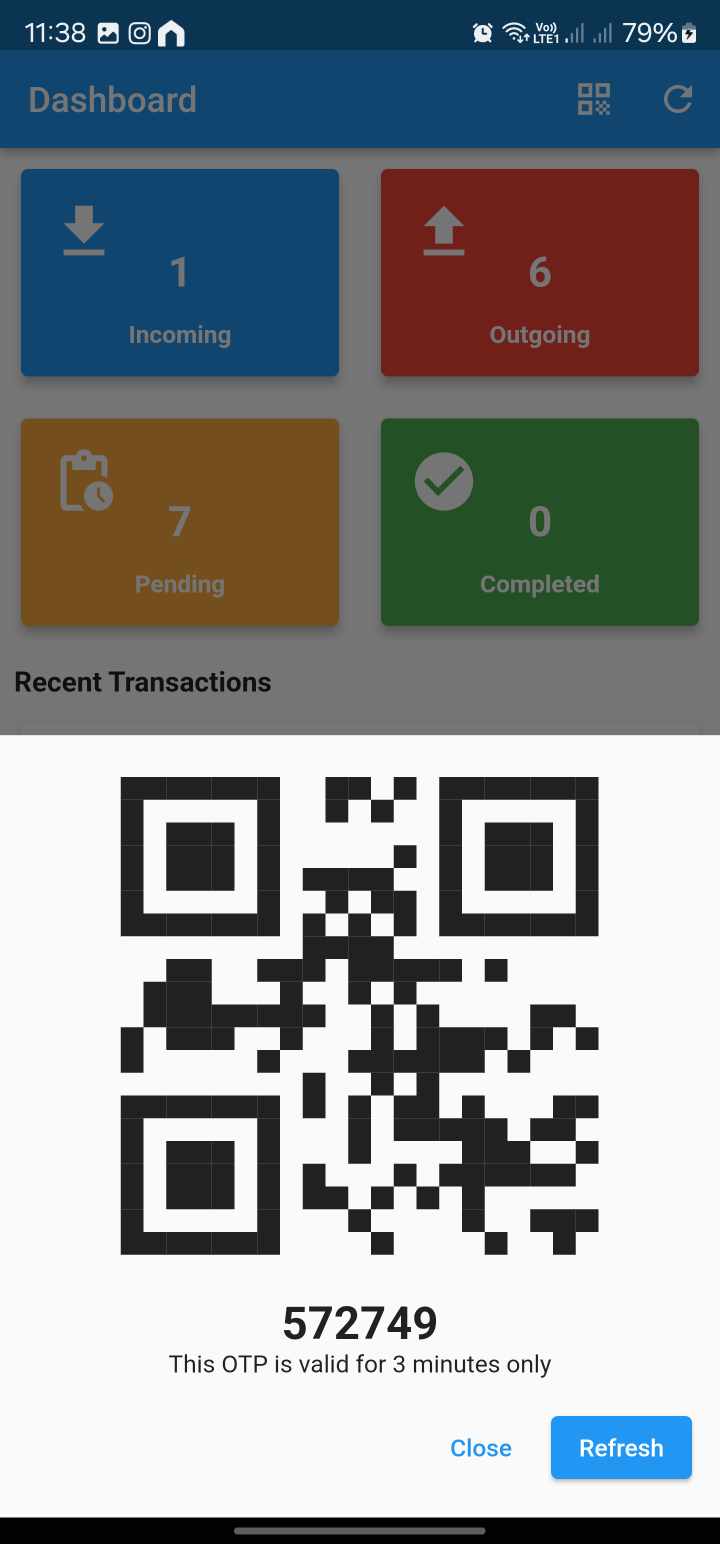
****

Figure 28 show the OTP Generator page. This function generates a one-time password (OTP) for secure user authentication. The OTP is provided by the user who handed the document to the receiving office to perform receive action.

Figure 29  
*PDF Viewer*

****

Shown in Figure 29 is the File Viewer, whish views the lates uploaded file.