Syrian Private University

Faculty of Computer &

Information Engineering

Department of Software &

Information Systems Engineering



Title

Data-Sharing Platform Using Cryptography

Prepared By

Omar Alkhateeb – Anas Arman

Supervised BY

Eng. Saeed Diab

2024-2025

Supervisor Certification

I certify that the preparation of this project entitled **Data-Sharing Platform Using Cryptography**, prepared by **Omar Alkhateeb, Anas Arman** was made under my supervision at Department of Software and Information Systems Engineering in partial of Bachelors of Degree the for Requirements the of fulfillment Software and Information Systems Engineering.

Name: **Eng. Saeed Diab** Signature:…….. Date: ……

Summary

With the continuous expansion in the exchange of sensitive data and information among governmental entities and institutions, there is a growing need for a system that ensures the confidentiality and security of data during transmission and storage. Hence, the idea of the project **"Encrypted File Sharing System"** emerged as an innovative technical solution that facilitates secure file sharing among users in a controlled and protected environment, while fully complying with information security standards.

This system specifically serves the **Financial Regulatory Authority**, by providing a platform that enables its employees and authorized users to send and receive files and documents in an encrypted and reliable manner, without relying on external cloud storage providers. The system supports direct device-to-device communication using **Wi-Fi Direct** and guarantees advanced encryption of data through **AES** and **RSA** during transmission and storage.

The system includes a set of key features and functionalities, such as:

- User and permission management with a flexible structure including admin, employee, and external reviewer.
- Secure login system with encrypted passwords and session validation.
- A simple and practical interface for easy file uploading, downloading, and organization.
- **Integration of an intelligent Chatbot to assist users with operations such as file uploading, sending, and receiving.**
- **Utilization of Artificial Intelligence techniques to automatically categorize files, enhance search, and suggest appropriate file names.**
- Smart search engine and file classification based on type, name, or date.
- Comprehensive logging of all activities to ensure accountability and transparency.
- Real-time notifications for events such as receiving new files or detecting unauthorized access attempts.
- Ability to generate detailed reports about file transfers, users, and security status.

The system is designed to be scalable and easily integrated with future systems that the authority may adopt, and it also provides **APIs** to facilitate integration with other internal platforms.

In summary, this project delivers an effective and secure platform for managing the exchange of sensitive files, enabling the authority to keep pace with digital transformation while maintaining the highest levels of security and efficiency.

الملخص

في ظل التوسع المستمر في تبادل البيانات والمعلومات الحساسة بين الجهات الحكومية والمؤسسات، تبرز الحاجة إلى وجود نظام يضمن سرية وأمان البيانات أثناء النقل والتخزين. من هنا جاءت فكرة مشروع "نظام مشاركة الملفات المشفرة" كحل تقني مبتكر يهدف إلى تسهيل مشاركة الملفات بين المستخدمين ضمن بيئة آمنة ومراقبة، مع الالتزام الكامل بمعايير أمن المعلومات.

يركز هذا النظام بشكل خاص على خدمة هيئة الرقابة المالية، من خلال توفير منصة تُمكن موظفي الهيئة والمستخدمين المعتمدين من إرسال واستقبال الملفات والمستندات بطريقة مشفّرة ومضمونة، بدون الاعتماد على مزودي خدمات التخزين السحابي الخارجية. يدعم النظام الاتصال المباشر بين الأجهزة باستخدام تقنيات مثل **Wi-Fi Direct**، ويضمن التشفير المتقدم للبيانات مثل **AES** و **RSA** أثناء النقل والتخزين.

يضم النظام مجموعة من الميزات والوظائف الأساسية، منها:

- إدارة المستخدمين والصلاحيات وفق هيكلية مرنة تشمل المسؤول، الموظف، والمراجع الخارجي.
- نظام تسجيل دخول آمن باستخدام كلمات مرور مشفّرة والتحقق من الجلسات.
- واجهة استخدام بسيطة وعملية تتيح رفع وتنزيل وتنظيم الملفات بسهولة.
- دمج Chatbot ذكي لمساعدة المستخدمين في عمليات مثل رفع الملفات، إرسالها، واستقبالها.
- اعتماد تقنيات الذكاء الصنعي لتصنيف الملفات تلقائياً، تحسين البحث، واقتراح أسماء مناسبة للملفات.
- محرك بحث ذكي وتصنيف للملفات حسب النوع، الاسم، أو التاريخ.
- تسجيل وتتبع كافة العمليات (**Logs**) لضمان المساءلة والشفافية.
- تنبيهات فورية في حالات مثل استلام ملف جديد، أو محاولة وصول غير مصرح بها.
- إمكانية توليد تقارير شاملة عن حركة الملفات، المستخدمين، والحالة الأمنية.

تم تصميم النظام ليكون قابلاً للتوسع والتكامل مع الأنظمة المستقبلية التي قد تعتمدها الهيئة، كما أنه يحتوي على واجهات برمجية (**APIs**) لتسهيل التكامل مع الأنظمة الداخلية الأخرى.

باختصار، يقدم هذا المشروع منصة فعالة وآمنة لإدارة تبادل الملفات الحساسة، مما يُمكّن الهيئة من مواكبة التحول الرقمي مع المحافظة على أعلى درجات الأمان والكفاءة.

CONTENTS

Summary 3

الملخص 4

Chapter 1: Introduction 11

1. Introduction 12

2. Problem Statement 12

3. Project Objective 12

4. Proposed System 13

5. Report Organization 13

6. Summary 13

Chapter 2 : Fundamental Concepts and Literature Review 14

1. Introduction 15

2. Fundamental Concepts 15

Chapter 3: Literature Review: 16

Chapter 3 : Project Management 23

1. Introduction: 24

2. Project Charter 24

3. The SOW document: 26

Chapter 4 : System Analysis 31

1. Software Requirement Specification document (SRS) : 32

1. Introduction : 32

1.1 Purpose 32

1.2 Project Scope 32

1.2.1 High-Level Requirements 32

1.2.2 Actors 33

2. Overall Description : 33

2.1 Product Perspective 33

2.2 Product Features 34

2.3 User Classes and Characteristics 34

2.3.1 Admin (Financial Regulatory Authority) 34

2.3.2 User 34

3. System Objective: 35

3.1 Objective of The System : 35

3.1.1 Security & Cryptography Features: 35
3.1.2 User-Centered Features 35
3.1.3 File & Message Handling 36
3.1.4 Admin Features 36
3.2 functional requirements : 38
3.2.1 Account Management: 38
3.2.2 Admin Management (Oversight Authority): 38
3.2.3 File Management: 38
3.2.4 Storage Management: 39
3.2.5 Secure File Sharing: 39
3.2.6 Search and Filtering: 39
3.2.7 Notifications and Alerts: 39
3.2.8 Centralized Assistance: 40
3.2.9 Artificial Intelligence Requirements: 40
3.2.10 Admin Additional Requirements: 40
4. Non-Functional Requirement: 41
4.1 Scalability 41
4.2 Availability 41
4.3 Interoperability 41
4.4 Security 41
4.5 Extensibility 41
5. Requirements modeling : 42
2 - Use Case Specifications: 44
2.8 Central Assistance: 84
2.9 Artificial Intelligence Requirements: 86
2.9.1 Automatically classify files into folders based on file type: 86
2.9.2 Improve file search: 87
2.9.3 Suggest appropriate names for files when uploading or receiving them: 88
2.9.4 Provide a recommendation system to suggest related files that the user may need: 89
3- Activity Diagram: 90
4.List of Preliminary Tests: 100
5.Requirements Traceability Matrix (RTM): 105
Chapter 5: Design Study 109

1. Sequence Diagram: 110
120
2. ERD Diagram: 122
3. Class Diagram: 123
4. System Architecture 124
6.1 Introduction 124
6.2 System Architecture 124
4.1 Architectural Overview 125
4.2 Component Interaction 130
4.3 Communication Protocols 130
4.4 Security Layer 130
4.5 Permissions Management 131
4.6 Storage Management 132
4.7 System Architecture Conclusion 132
Chapter 6 : Practical implementation 133
1.1 Introduction: 134
2.The Implemented Tests : 156
3. Update the Requirements Traceability Matrix: 159
4. Analysis of the Results: 164
5. Conclusion: 165
Chapter 7 : Report Overview 166
1. Introduction 167
2. Report Structure and Purposes 167
3. Summary 168
4. References: 169

Figures

Figure 1 - Gantt chart 28
Figure 2 - Use Case Diagram (Hight Level) 42
Figure 3 - UML Diagrams (Low Level) 43
Figure 4 - Activity Diagram 'Login' 90
Figure 5 - Activity Diagram 'Create New Account' 91
Figure 6 - Activity Diagram 'Update Personal Information' 92
Figure 7 - Activity Diagram 'Generate Activity Reports' 93
Figure 8 - Activity Diagram 'File Details Displayed' 94
Figure 9 - Activity Diagram 'Display Available Storage Space' 95
Figure 10 - Activity Diagram 'Send Files Via WIFI Direct' 96
Figure 11 - Activity Diagram 'Receive Files' 97
Figure 12 - Activity Diagram 'Search Files By Name' 98
Figure 13 - Activity Diagram 'Notify When File Deleted' 99
Figure 14 - Sequence Diagram 'Login' 110
Figure 15 - Sequence Diagram 'Create New Account' 111
Figure 16 - Sequence Diagram 'Update Personal Information' 112
Figure 17 - Sequence Diagram 'Generate Activity Reports' 113
Figure 18 - Sequence Diagram 'File Details Displayed' 114
Figure 19 - Sequence Diagram 'Display Available Storage Space' 115
Figure 20 - Sequence Diagram 'Send Files Via WIFI Direct' 116
Figure 21 - Sequence Diagram 'Receive Files' 117
Figure 22 - Sequence Diagram 'Search Files By Name' 118
Figure 23 - Sequence Diagram 'Notify When File Deleted' 119
Figure 24 - Sequence Diagram 'Support text-based interaction with the user' 120
Figure 25 - Sequence Diagram 'Automatically classify files into folders based on file type' 121
Figure 26 - ERD Diagram 122
Figure 27 - Class Diagram 123
Figure 28 - Client-Server Architecture 124
Figure 29 - Logo App 141
Figure 30 - Signup 142
Figure 31 - Login 143
Figure 32 - Home Screen 144
Figure 33 - Direct Send Screen 145
Figure 34 - File Receive Screen 146
Figure 35 - QR Sharing Screen 147
Figure 36 - Setting Screen 148
Figure 37 - Chatbot 149
Figure 38 - Files Encrypted 150
Figure 39 - Admin Dashboard (Overview) 151
Figure 40 - Admin_Users_Screen 152
Figure 41 - Admin_Files_Screen 153
Figure 42 - Admin_Activity_Screen 154
Figure 43 - Admin_Reports_Screen 155

Tables

Table 1 Comparative Analysis of Similar Systems and the Target System 22
Table 2-Roles and Responsibilities 25
Table 3 - Risk management 30
Table 4- Use Case specification 'Create New Account' 44
Table 5- Use Case specification 'Login' 45
Table 6- Use Case specification 'Edit personal Information' 46
Table 7- Use Case specification 'Reset Password' 47
Table 8 - Use Case specification 'Activity Logs Monitoring' 48
Table 9 - Use Case specification 'Generate Activity & Log Reports' 49
Table 10 - Use Case specification 'Set Access Permissions For Each User On Files' 50
Table 11 - Use Case specification 'Delete User Account' 51
Table 12 - Use Case specification 'Send Files' 52
Table 13 - Use Case specification 'Send Images' 53
Table 14 - Use Case specification 'Send Videos' 54
Table 15 - Use Case specification 'Receive Files' 55
Table 16 - Use Case specification 'Receive Images' 56
Table 17 - Use Case specification 'Receive Videos' 57
Table 18 - Use Case specification 'Rename Received Files' 58
Table 19 - Use Case specification 'Organize Files Into Folders' 59
Table 20 - Use Case specification 'View File Details' 60
Table 21 - Use Case specification 'Enable Automatic File Deletion After A Period' 61
Table 22 - Use Case specification 'Create New Folder' 62
Table 23 - Use Case specification 'Local Storage Dependency' 63
Table 24 - Use Case specification 'Display Available Storage Space' 64
Table 25 - Use Case specification 'Specify Storage Locations' 65
Table 26 - Use Case specification 'Send Files via Wi-Fi Direct' 66
Table 27 - Use Case specification 'Request Recipient Approval Before Sending' 67
Table 28 - Use Case specification 'Log All Sending Operations' 68
Table 29 - Use Case specification 'Log All Receiving Operations' 69
Table 30 - Use Case specification 'Generate QR Code' 70
Table 31 - Use Case specification 'Send Files via QR Code' 71
Table 32 - Use Case specification 'Fix Encryption' 72
Table 33 - Use Case specification 'Generate Public Key' 73
Table 34 - Use Case specification 'Search Files by Name' 74
Table 35 - Use Case specification 'Search Files by Type' 75
Table 36 - Use Case specification 'Search Files by Upload Date' 76
Table 37 - Use Case specification 'Filters Files By Name' 77
Table 38 - Use Case specification 'Filters Files By Type' 78
Table 39 - Use Case specification 'Filters Files By Upload Date' 79
Table 40 - Use Case specification 'Notification on New File Received' 80
Table 41 - Use Case specification 'Alert on File Deletion' 81
Table 42 - Use Case specification 'Alert on File Modification' 82
Table 43 - Use Case specification 'Storage Near Capacity Warning' 83
Table 44 -Use Case Specification 'Support text-based interaction with the user' 84
Table 45 - Use Case Specification 'Assist the user in any operation they want' 85
Table 46 - Use Case Specification 'Automatically classify files into folders based on file type' 86

Table 47 - Use Case Specification 'Improve file search'.....87
Table 48 - Use Case Specification 'Suggest appropriate names for files when uploading or receiving them'.....88
Table 49 - Use Case Specification 'Provide a recommendation system to suggest related files that the user may need'.....89
Table 50 - Test 1.....100
Table 51 - Test 2.....100
Table 52 - Test 3.....101
Table 53 - Test 4.....101
Table 54 - Test 5.....102
Table 55 - Test 6.....102
Table 56 - Test 7.....103
Table 57 - Test 8.....103
Table 58- Test 9.....104
Table 59 - Test 10.....104
Table 60 - Requirements Traceability Matrix (RTM).....108
Table 61 - Testing secure file upload.....157
Table 62 - Testing secure image sending.....157
Table 63 - Testing file rename after receiving.....157
Table 64 - Testing user account deletion.....158
Table 65 - Testing notification when new file is received.....158
Table 66 - Testing sending files via Wi-Fi Direct.....158
Table 67 - Update the Requirements Traceability Matrix (RTM).....163

Chapter 1: Introduction

1. Introduction

This chapter presents the foundational concepts of the **Secure Data-Sharing Platform Using Cryptography**, developed under the oversight of the Financial Regulatory Authority. It outlines the core problem the platform aims to solve, its objectives, the proposed solution, and the structure of the report.

2. Problem Statement

In today's digital landscape, the sharing of sensitive information—especially in financial and governmental sectors—is often exposed to risks such as data breaches, unauthorized access, and lack of control over file usage after sharing. Existing file-sharing systems either lack robust encryption or do not offer the granularity needed in access management and file lifecycle control. This highlights the need for a secure, encrypted, and regulatory-compliant solution that allows users to share data with confidence and full control.

3. Project Objective

The main goal of this project is to build a secure platform for encrypted file sharing that supports end-to-end encryption (E2EE), flexible access control, and metadata privacy. Key objectives include:

- Ensuring data confidentiality using **RSA-OAEP** (asymmetric encryption) and **AES-GCM** (symmetric encryption).
- Enabling user authentication via 2FA or biometrics.
- Allowing for role-based permission management.
- Supporting self-destructing messages, offline encryption, and secure QR/link sharing.
- Integrating a Chatbot assistant to guide users in operations such as uploading, sending, and receiving files.
- Utilizing Artificial Intelligence for automatic file classification, smart search, and suggesting meaningful file names.

The platform architecture relies on cryptographic libraries and secure storage, with a focus on scalability, user assistance, and regulatory alignment.

4. Proposed System

The Secure Data-Sharing Platform is designed to serve as a comprehensive and user-centric environment for sharing data safely. By integrating modern cryptographic technologies, it ensures that files and messages are only accessible to intended recipients. The system supports features such as key pair generation and management, secure file upload/download, and end-to-end encryption. With real-time notifications and full control over who can view, forward, or delete content, the system ensures privacy, usability, and compliance with financial sector standards.

5. Report Organization

The report is structured as follows:
Chapter 1: Introduction

Chapter 2: Basic Concepts and Reference Study

Chapter 3: Project Management

Chapter 4: System Analysis

Chapter 5: System Design

Chapter 6: Practical Implementation

Chapter 7: Report Overview

6. Summary

This chapter has outlined the need for a secure data-sharing platform, explained the objectives of the project, introduced the proposed solution at a high level, and provided an overview of the report's structure. This introduction sets the stage for a deeper exploration of each aspect of the project in the following chapters.

Chapter 2 : Fundamental Concepts and Literature Review

1. Introduction

This chapter provides a foundation for understanding the **Secure Data-Sharing Platform Using Cryptography and AI** by exploring fundamental concepts and reviewing related studies. It examines essential terminology, cryptographic principles, and the mechanisms that ensure secure data exchange. The goal is to offer a comprehensive overview of encryption-based platforms, secure file-sharing methods, chatbot-assisted interaction, and access control systems, drawing from existing technologies and research to support the development of the proposed system.

2. Fundamental Concepts

This section explains the core concepts, terms, and theoretical principles relevant to the project, including:

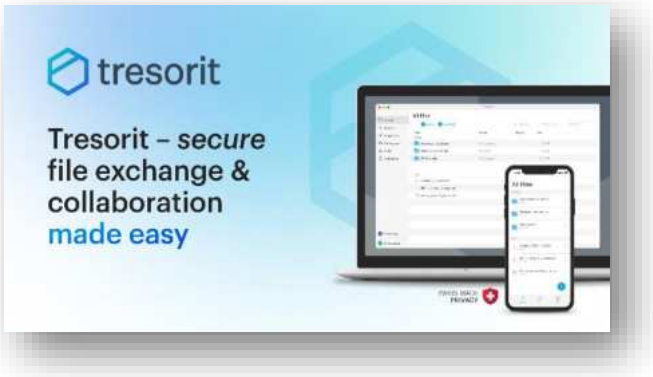
- **Cryptography**: Overview of cryptography, its role in securing digital communication, and the difference between symmetric (**AES-GCM**) and asymmetric (**RSA-OAEP**) encryption. The section also discusses encryption keys, digital signatures, and confidentiality vs. integrity.
- **End-to-End Encryption (E2EE)**: Explanation of E2EE and how it ensures that only intended recipients can access shared files or messages. It highlights the significance of this approach in protecting against man-in-the-middle attacks and third-party access.
- **Role-Based Access Control (RBAC)**: Introduction to RBAC and its importance in regulating file access based on user roles. It explains how access is granted based on predefined permissions (e.g., admin, sender, receiver) to enhance control and accountability.
- **File Lifecycle Management**: Discussion on managing files from upload to deletion, including file versioning, auto-deletion settings, renaming, and storage allocation. This concept is key for tracking file status and compliance with user-defined policies.
- **Secure File Transmission (Wi-Fi Direct)**: Overview of direct peer-to-peer file sharing methods such as Wi-Fi Direct, and how they enable offline secure transfers while logging all sending/receiving activities.
- **User Authentication and Privacy**: Explanation of authentication methods like 2FA, biometrics, and password resets, alongside privacy controls for personal information and storage location visibility.

- **Search, Filtering, and Notification Systems**: Description of features that improve usability, such as advanced search and filtering by name, type, or upload date, and real-time notifications for file activity (upload, delete, storage alerts).
- **Chatbot Assistance**: Overview of the integrated chatbot, which guides users in performing operations such as uploading, sending, and receiving files, and resetting passwords, making the platform more user-friendly.
- **AI-Based File Classification**: Explanation of the AI module that organizes files into folders, suggests meaningful names, and enables smart search capabilities, enhancing efficiency and accessibility.

Chapter 3: Literature Review :

The purpose of this Literature Review is to analyze and evaluate existing systems and methodologies related to secure file-sharing platforms and encryption-based data exchange. This review aims to provide a comprehensive overview of current technologies and tools used in secure communication, with a focus on cryptographic techniques, access control, file lifecycle management, and peer-to-peer data sharing. The analysis includes comparisons between four systems and their key features:

• Tresorit :



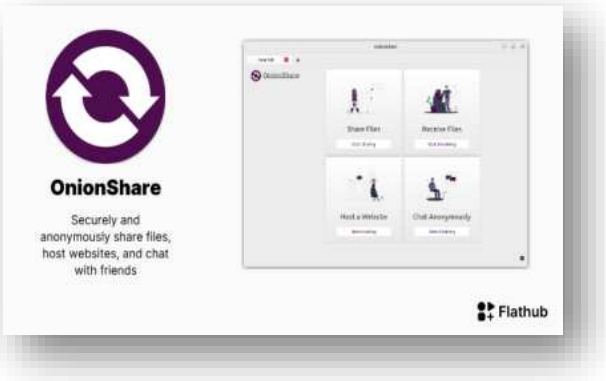
Tresorit is a premium, enterprise-grade cloud storage and file sharing platform that prioritizes security, especially for businesses and teams handling sensitive data.

Key Features with Expanded Explanation:

- **End-to-End Encryption:**
  All files and folders are encrypted locally before being uploaded. Not even Tresorit's servers can read your files, due to their zero-knowledge encryption model.
- **Granular Access Controls:**
  Admins and users can define precise sharing rights, like who can view, edit, or re-share a document. You can also set time limits and password protections for shared links.
- **Multi-Device Synchronization:**
  Automatically keeps files synced across all user devices while maintaining encryption integrity.
- **Activity Tracking and Logging:**
  Every user action (view, download, edit, etc.) is logged for auditing. Ideal for organizations subject to compliance requirements.
- **Compliance and Legal Support:**
  Meets strict standards like **GDPR**, **HIPAA**, **ISO 27001**, and Swiss privacy laws. Trusted by law firms, healthcare providers, and financial institutions.
- **Secure Collaboration for Teams:**
  You can create encrypted workspaces for groups, ideal for sharing internal documents while managing access in real-time.
- **Mobile and Desktop Integration:**
  Works seamlessly across platforms (Windows, macOS, iOS, Android), with offline file access.

Tresorit is best suited for **high-security needs** in professional settings, particularly legal, healthcare, and finance sectors.

• OnionShare :



OnionShare is a privacy-first tool for anonymous file sharing and communication, designed for individuals who need **complete anonymity** and **no data traces**.

Expanded Features:

- **Self-hosted File Sharing:**
  Your computer becomes the file host. When you share a file, a unique .onion address is generated, accessible only via the Tor Browser.
- **No Account or Server Involvement:**
  Unlike Dropbox or Google Drive, OnionShare does not use central servers or require login. This makes it ideal for whistleblowers, activists, or journalists.
- **Built-in Anonymous Chat:**
  Offers private chat rooms that operate through Tor, protecting your metadata and preventing surveillance or logging.
- **Receive Mode (Secure Dropboxes):**
  You can run a secure, anonymous file drop service, allowing others to send you files without exposing either party's identity.
- **Cross-platform and Portable:**
  Works on Windows, macOS, and Linux. No installation needed in some versions — just run the app.
- **Open-source and Transparent:**
  All source code is public. You can audit how the tool works and ensure it contains no backdoors.

Best suited for **privacy advocates, journalists, NGOs, and whistleblowers**, especially in hostile digital environments.

• Syncthing



Syncthing is a powerful, **decentralized file synchronization tool** that offers privacy, speed, and flexibility without relying on third-party cloud storage providers.

Expanded Feature Highlights:

- **Peer-to-Peer Architecture:**
  Files sync directly between devices using a secure TLS connection. No files are stored in the cloud unless you configure it yourself.
- **Real-Time, Bidirectional Syncing:**
  Any update to a file on one device is immediately reflected on others. It supports file versioning and conflict resolution.
- **Full Ownership and Privacy:**
  You own your data. Nothing leaves your local network unless you allow it. Great for sensitive or large files (e.g. raw media, archives).
- **Device-Specific Permissions:**
  You can assign sync permissions to specific devices and limit access to certain folders.
- **No Setup Complexity:**
  Unlike traditional file servers, there's no complex configuration. Devices detect each other with an ID and QR code.
- **Web GUI for Management:**
  Includes a simple web-based dashboard to monitor sync status, device connections, logs, and usage.
- **Resource Efficient and Lightweight:**
  Syncthing uses minimal CPU and memory, making it ideal even for old devices or Raspberry Pi.

Syncthing is ideal for **developers, remote teams, self-hosters**, or any user who wants **cloud-like syncing** with **total privacy**.

4. Benefit from similar applications :

From Tresorit

Tresorit gave me a solid understanding of **enterprise-grade security and compliance in cloud file-sharing**. I learned how to implement:

- **End-to-end encryption** as a foundation for data protection, ensuring that files are encrypted before leaving the device.
- The importance of **role-based access control** (RBAC) to manage who can view, edit, or share files.
- The value of **secure link sharing** with expiry dates and password protection to control external access.
- The need for **audit logs and activity tracking**, especially when aiming for compliance with privacy regulations (GDPR, HIPAA, etc.).
- How to design **multi-device synchronization** while preserving data integrity and encryption.

These insights directly influenced my project's architecture in areas like secure storage, access policies, and compliance orientation.

From OnionShare

OnionShare introduced me to the **concept of anonymous, serverless file sharing**. From it, I learned:

- How to build a **peer-to-peer sharing model** without relying on third-party servers.
- The significance of using **Tor for privacy-preserving communication** between users.
- The idea of implementing **temporary, single-use file access links** for confidentiality.
- The advantage of having an **anonymous file drop (receive mode)** for whistleblowers or users who need to submit files securely.

This helped shape the **anonymity layer** in my project — allowing secure and private transfer of files without user identity exposure.

From Syncthing

Syncthing taught me about **decentralized synchronization and full data ownership**. Key lessons include:

- The benefits of **direct device-to-device syncing** to reduce reliance on cloud storage.
- How to allow **real-time synchronization** across trusted devices without any third-party involvement.
- Ways to **implement file versioning and conflict resolution** in collaborative environments.
- How to create a **lightweight, resource-efficient system** that works even on low-end devices or internal networks.
- How to use **device pairing with unique IDs** to authorize secure sync between endpoints.

This influenced my decision to keep the file exchange decentralized and give full control to the user over their data.

Comparative Analysis of Similar Systems and the Target System

System Feature	Tresorit	OnionShare	Syncthing	Our System
End-to-End Encryption	✓	✓	✗	✓
Role-Based Access Control (RBAC)	✓	✗	✗	✓
File Auto-Deletion Settings	✗	✓	✗	✓
Storage Location Selection	✓	✗	✓	✓
Anonymous File Sharing	✗	✓	✗	✓ *
Peer-to-Peer File Transfer (Wi-Fi Direct)	✗	✗	✓	✓
Activity Logging & Reporting	✓	✗	✗	✓
Admin User Management	✓	✗	✗	✓
File Categorization & Folder Organization	✓	✗	✓	✓
File Rename on Reception	✗	✗	✗	✓
Real-Time Notifications	✓	✗	✗	✓
File Search & Filtering	✓	✗	✗	✓
Multi-Language Interface	✓	✗	✗	✓ *
Two-Factor Authentication / Biometrics	✓	✗	✗	✓

Table 1 Comparative Analysis of Similar Systems and the Target System

(*)：Features planned for future development and not within the scope of this semester.

Chapter 3 : Project Management

1. Introduction:

In this chapter, we will dive into the management phase of the project, which is a critical aspect of ensuring the project's success. We will examine the project charter, project plan, Statement of Work (SOW) document, stakeholder analysis, and risk management strategies to effectively guide and control the project from initiation to completion.

2. Project Charter

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision-making and project governance.

- **Project Title**: Secure Data-Sharing Platform Using Cryptography
- **Project Start Date**: April 5, 2025
- **Projected Finish Date**: September 18, 2025
- **Project Manager**: Eng.Saeed Diab

Project Objectives:

❖ Develop a secure platform for encrypted file and image exchange between users.

❖ Implement advanced cryptographic algorithms, including **RSA-OAEP** for asymmetric encryption and **AES-GCM** for symmetric encryption, to ensure confidentiality, authenticity, and integrity.

❖ Implement account management with authentication, password recovery, and personal data editing.

❖ Enable file sharing via Wi-Fi Direct with recipient approval and automatic logging.

❖ Integrate role-based access control to secure files and restrict user privileges.

❖ Support file organization in folders, metadata viewing, and auto-deletion after a specified period.

❖ Introduce an **AI-powered file classification system** to automatically organize files, suggest suitable names, and simplify search.

❖ Provide a **Chatbot assistant** to guide users in performing actions such as file upload, file sending, password resetting, and QR-based file sharing.

Approach:

❖ Conduct analysis of user needs and system security requirements, including AI-assisted features.

❖ Design the architecture including encryption flow (RSA-OAEP + AES-GCM), access roles, file lifecycle control, and chatbot integration.

❖ Develop and test features using an iterative approach to ensure security, functionality, and usability.

❖ Deploy modules incrementally and ensure local storage allocation, management, and AI-based file organization.

- **Roles and Responsibilities:**

Name	Role	Responsibility
Eng. Saeed Diab	Project Manager & Sponsor	Provide strategic supervision, approve project scope and goals, offer guidance in critical decisions, and ensure alignment with academic standards.
Omar Alkhateeb	Frontend Developer & Security	Design and develop the user interface, ensure usability and responsiveness, and implement frontend-level encryption and secure file-sharing mechanisms.
Mohammad Anas Arman	Backend Developer & Security	Develop the backend system, implement user authentication, file management logic, and encryption algorithms to ensure secure data processing.

Table 2-Roles and Responsibilities

3. The SOW document:

outlines the specific tasks, deliverables, timeline, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project's objectives, and the criteria for success.

1. Project Description and Objectives:

The project aims to develop a **Secure Data-Sharing Platform using Cryptography** to enable safe exchange of files and images between users through encrypted channels. It seeks to ensure confidentiality, integrity, and secure access control while maintaining ease of use.

2. Project Scope:

The platform is designed to provide a local secure file-sharing environment based on encryption, role-based access control, and user account management. It will support Wi-Fi Direct sharing, file organization, and logging, without relying on cloud services.

3. Project Goals:

- Develop a system for secure encrypted file sharing and reception.
- Implement user authentication, account editing, and password recovery mechanisms.
- Enable Wi-Fi Direct transfer with recipient approval and full activity logging.
- Provide auto-deletion settings and folder-based file organization.
- Show file metadata, available storage, and notifications.
- Support file search and filtering based on name, type, and upload date.

Deliverables:

- Project plan
- Software Requirements Specification (SRS)
- Final project report
- Encryption modules and secure transmission logic

- Fully functional backend and frontend components

4. Project Requirements:

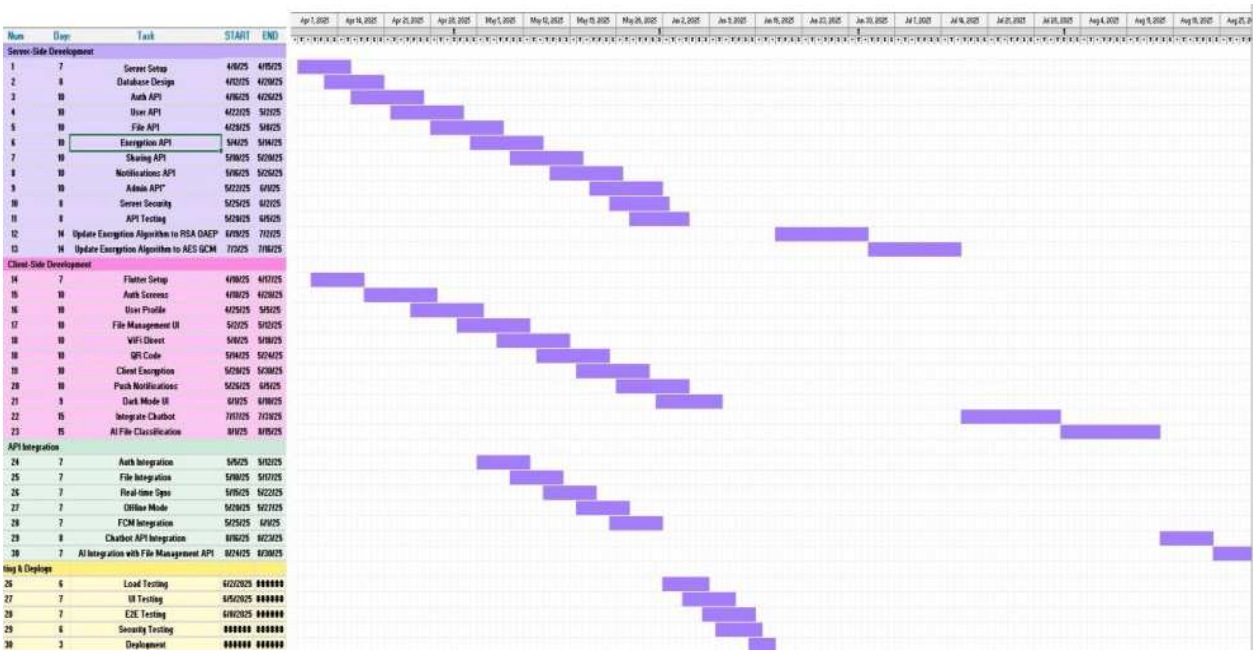
Technology and Tools:

- **Programming Languages:** Dart , ASP.Net
- **Frameworks:** Flutter (frontend)
- **File transfer protocol:** Wi-Fi Direct

5. Assumptions:

- Regular availability of the team for development and feedback loops
- Continuous supervision from the academic supervisor (Dr. Saeed Diab)
- Access to required development tools and hardware resources

4. Project plan _ Gantt Chart:



Num	Days	Task	START	END
Server-Side Development				
1	7	Server Setup	4/8/25	4/15/25
2	8	Database Design	4/12/25	4/20/25
3	10	Auth API	4/16/25	4/26/25
4	10	User API	4/22/25	5/2/25
5	10	File API	4/28/25	5/8/25
6	10	Encryption API	5/4/25	5/14/25
7	10	Sharing API	5/10/25	5/20/25
8	10	Notifications API	5/16/25	5/26/25
9	10	Admin API*	5/22/25	6/1/25
10	8	Server Security	5/25/25	6/2/25
11	8	API Testing	5/29/25	6/5/25
12	14	Update Encryption Algorithm to RSA OAEP	6/19/25	7/2/25
13	14	Update Encryption Algorithm to AES GCM	7/3/25	7/16/25
Client-Side Development				
14	7	Flutter Setup	4/19/25	4/17/25
15	10	Auth Screens	4/18/25	4/28/25
16	10	User Profile	4/25/25	5/5/25
17	10	File Management UI	5/2/25	5/12/25
18	10	WiFi Direct	5/8/25	5/18/25
18	10	QR Code	5/14/25	5/24/25
19	10	Client Encryption	5/20/25	5/30/25
20	10	Push Notifications	5/26/25	6/5/25
21	3	Dark Mode UI	6/1/25	6/10/25
22	15	Integrate Chatbot	7/17/25	7/13/25
23	15	AI File Classification	8/1/25	8/15/25
API Integration				
24	7	Auth Integration	5/5/25	5/12/25
25	7	File Integration	5/10/25	5/17/25
26	7	Real-time Sync	5/15/25	5/22/25
27	7	Offline Mode	5/20/25	5/27/25
28	7	FCM Integration	5/25/25	6/1/25
29	8	Chatbot API Integration	8/16/25	8/23/25
30	7	AI Integration with File Management API	8/24/25	8/30/25
ting & Deploy				
26	6	Load Testing	6/22/2025	######
27	7	UI Testing	6/5/2025	######
28	7	E2E Testing	6/8/2025	######
29	6	Security Testing	######	######
30	3	Deployment	######	######

Figure 1 - Gantt chart

5. Risk management :

Risk Title	Risk Description	Raised Date	Tracking Frequency	State	Impact	Mitigation Plan
Small Team Size	If one team member stops working, it will affect project progress.	2025-04-05	Weekly	Active	High	The team member who finishes early helps others.
Scope Ambiguity	Misunderstanding the project scope may cause implementation mistakes.	2025-04-07	Weekly	Closed	Medium	Hold regular meetings to clarify objectives.
Learning Curve	Limited knowledge of encryption/security may hinder progress.	2025-04-10	Weekly	Under Mitigation	High	Allocate time for learning with online resources.
Frontend-Backend Separation	Could lead to poor integration and coordination.	2025-04-12	Weekly	Active	Medium	Sync progress regularly and perform integration testing.
File Transfer Issues	Wi-Fi Direct and encryption integration may cause errors.	2025-04-15	Weekly	Active	High	Test features early and document protocols clearly.
Storage Limit Notification	Errors may occur if storage reaches capacity during active use.	2025-04-28	Weekly	Active	Medium	Set usage thresholds and notify users when space nears full.
Notification Errors	Alerts for file receipt or deletion may fail, affecting user awareness.	2025-05-02	Weekly	Active	Medium	Perform unit testing for notifications and fallback messages.
Search Filter Inefficiency	Poor implementation of search or	2025-05-10	Weekly	Active	Medium	Test search and filter functions on

	filtering features may reduce usability.					various datasets during development.
Auto Delete Bugs	Automatic file deletion logic may malfunction or delete prematurely.	2025-05-17	Weekly	Active	High	Ensure countdown timers and delete logic are tested thoroughly.
Integration Complexity	Delays may arise when merging backend logic with frontend display and file UI.	2025-05-22	Weekly	Active	Medium	Sync often and assign integration checkpoint after each major module.
Deployment Challenges	Inexperience with deploying encrypted systems may delay final delivery.	2025-06-05	Weekly	Active	High	Research deployment best practices and conduct dry-runs.
Late Requirement Changes	Stakeholders or supervisor may ask for changes close to delivery.	2025-06-11	Weekly	Active	High	Reserve a buffer window in final phase and maintain constant communication.

Table 3 - Risk management

Chapter 4 : System Analysis

1. Software Requirement Specification document (SRS) :

1. Introduction :

1.1 Purpose

This document specifies the requirements for the **Secure Encrypted File Sharing Platform**, a system designed to securely transmit, classify, and manage files using **state-of-the-art cryptographic algorithms (RSA-OAEP, AES-GCM)**, integrated **Artificial Intelligence (AI)** for file classification, and a **Chatbot assistant** for user support, while ensuring compliance with the **Financial Regulatory Authority (FRA)** standards..

1.2 Project Scope

The system provides secure file sharing, user account management, file organization, and access control. It includes cryptographic protection, automated classification using AI, chatbot-based assistance, and compliance mechanisms through log tracking integrated with the financial authority.

1.2.1 High-Level Requirements

- **Account Management**
  - Create a new account
  - Login / Logout
  - Reset password
  - Edit profile
  - Upload a profile picture
- **Admin Panel (Financial Regulatory Authority)**
  - Monitor system logs and activities
  - Generate reports for transactions
  - Set file access permissions
  - Delete user accounts
- **File Management**
  - Send/receive encrypted files using **AES-GCM & RSA-OAEP**
  - Rename files upon receipt
  - Organize files in folders
  - View file details and metadata
  - Enable auto-deletion after a period

- **Storage Management**
  - Local storage only
  - Display available storage per user
  - Choose target storage path
- **File Sharing**
  - Use Wi-Fi Direct
  - Get recipient approval before sending
  - Record all sending/receiving events
- **Search and Filter**
  - Search by name, type, or date
  - Filter files by multiple criteria
- **AI-Powered Features**
  - Intelligent file classification based on content/type
  - Automated tagging for easy retrieval
- **Chatbot Support**
  - 24/7 chatbot for user guidance and troubleshooting
  - Natural language queries for searching files or getting system help
- **Notifications & Alerts**
  - Notify on new file arrival
  - Alert on file deletion/modification
  - Warn when storage is almost full

1.2.2 Actors

- **Admin (FRA):** Monitors activity, generates reports, sets permissions, ensures compliance.
- **User:** Registers, logs in, sends/receives encrypted files, organizes files, uses chatbot and AI features.

2. Overall Description :

2.1 Product Perspective

The platform is a standalone secure file-sharing solution enhanced with **end-to-end encryption (RSA-OAEP + AES-GCM)**, **AI-driven classification**, and **chatbot support**. It is designed to integrate **regulatory oversight**, **usability**, and **intelligence** in one system, ensuring compliance with financial authority requirements.

2.2 Product Features

- **User Account Management:** Register, login, reset password, edit profile, upload a profile photo.
- **Admin Panel:** Monitor logs, generate reports, set file permissions, delete accounts.
- **Encrypted File Management:** Send/receive encrypted files (RSA-OAEP for key exchange, AES-GCM for file encryption), organize files, auto-deletion.
- **AI File Classification:** Classify files automatically into categories (e.g., financial reports, images, contracts).
- **Chatbot Assistant:** Provide help, answer queries, assist in file searches using natural language.
- **Storage Management:** Local storage only, track available space, allow admin to define storage paths.
- **Secure File Sharing:** Wi-Fi Direct, recipient consent, event logging.
- **Search & Filtering:** Multi-criteria search with AI-enhanced suggestions.
- **Notifications & Alerts:** Real-time notifications for file activities and storage warnings.

2.3 User Classes and Characteristics

2.3.1 Admin (Financial Regulatory Authority)

- **Responsibilities:** Enforce compliance, monitor system usage, generate reports.
- **Privileges:** Full access to logs, user management, file permissions.

2.3.2 User

- **Responsibilities:** Send/receive encrypted files, manage profile, organize files.
- **Privileges:** Limited to personal storage, AI classification, chatbot support, and file operations.

3. System Objective:

3.1 Objective of The System :

3.1.1 Security & Cryptography Features :

1. **End-to-End Encryption (E2EE):**
   Data is encrypted on the sender's device and decrypted only on the recipient's side. This prevents any third party—including servers or administrators—from accessing content during transmission.
   - **RSA-OAEP** is used for secure key exchange.
   - **AES-GCM** is used for high-speed and integrity-protected data encryption.
2. **Key Management System:**
   A robust subsystem for key generation, storage, and lifecycle management. Users can:
   - Generate public/private key pairs.
   - Export/import keys for device synchronization.
3. **Zero-Knowledge File Sharing:**
   Even backend servers cannot access file content or metadata, ensuring maximum privacy.
4. **Self-Destructing Files:**
   Files/messages can auto-delete:
   - After being opened once, or
   - After a defined expiration time.
5. **Offline Encryption Mode:**
   Users can encrypt files locally (offline), producing secure payloads that can be shared later via email, USB, or other media.

3.1.2 User-Centered Features

1. **Secure File Upload & Download:**
   Files are encrypted before leaving the device, and decrypted upon download with integrity checks.
2. **Access Control & Permissions:**
   Users can define fine-grained rules:
   - Who can view, download, or reshare files.
   - Apply expiration dates or password protection.

3. **QR Code / Link Sharing:**
   Files can be shared through:
   - Unique encrypted download links.
   - QR codes embedding cryptographic metadata.
4. **Authentication System:**
   - Two-Factor Authentication (2FA) using email/phone verification.
   - Optional biometric authentication.
5. **Real-Time Notifications:**
   Users get alerts when:
   - Files are uploaded/downloaded/viewed.
   - Files expire or are deleted.
6. **Chatbot Assistant (New):**
   Integrated chatbot provides:
   - Guidance for file upload/download.
   - Help with password reset or permissions.
   - Quick answers to user queries.

3.1.3 File & Message Handling

1. **Support for Multiple File Types:**
   Documents (DOC, PDF), images (JPG, PNG), archives (ZIP, RAR), and others.
2. **Activity Logs:**
   A secure log system records:
   - File/message transmission history.
   - Access details (who, when).
   - Expiry or deletion events.
3. **AI-Powered File Classification (New):**
   - Automatically organizes files into folders based on type/content.
   - Suggests meaningful file names.
   - Enhances searchability for faster access.

3.1.4 Admin Features

1. **User Management:**
   - Create and delete accounts.
   - Reset user passwords.
   - Manage user roles and permissions.

2. **File User Management:**
    - Monitor file uploads/downloads.
    - Delete or quarantine suspicious files.
    - Force re-encryption when required.
    - Enforce file-sharing policies.
3. **Activity & Report Management:**
    - View and analyze activity logs.
    - Search within logs and reports.
    - Generate reports in PDF/Excel format.
    - Schedule automated periodic reports.
    - Monitor security alerts related to users and files.

3.2 functional requirements：

3.2.1 Account Management:

REQ-1.1: The system shall allow users to create a new account.
REQ-1.2: The system shall allow users to log into their accounts.
REQ-1.3: The system shall allow users to reset their password.
REQ-1.4: The system shall allow users to update their personal profile information.

3.2.2 Admin Management (Oversight Authority):

REQ-2.1: The system shall allow the oversight authority to monitor all activities and logs.
REQ-2.2: The system shall allow the authority to generate activity and audit reports.
REQ-2.3: The system shall allow the authority to assign or restrict file access permissions.
REQ-2.4: The system shall allow the authority to delete or deactivate user accounts.

3.2.3 File Management:

REQ-3.1: The system shall allow users to send encrypted files.
REQ-3.2: The system shall allow users to send encrypted images.
REQ-3.3: The system shall allow users to send encrypted videos.
REQ-3.4: The system shall allow users to receive encrypted files.
REQ-3.5: The system shall allow users to receive encrypted images.
REQ-3.6: The system shall allow users to receive encrypted videos.
REQ-3.7: The system shall allow renaming files upon receipt.
REQ-3.8: The system shall allow organizing files into folders.
REQ-3.9: The system shall allow users to view file details.
REQ-3.10: The system shall support automatic file deletion after a set period.
REQ-3.11: The system shall allow creating new folders.

3.2.4 Storage Management:

REQ-4.1: The system shall use local storage for saving files.
REQ-4.2: The system shall display available storage space per user.
REQ-4.3: The system shall allow administrators to define file storage locations.

3.2.5 Secure File Sharing:

REQ-5.1: The system shall support file transfer via Wi-Fi Direct.
REQ-5.2: The system shall require recipient confirmation before transfer starts.
REQ-5.3: The system shall log all file sending activities.
REQ-5.4: The system shall log all file receiving activities.
REQ-5.5: The system shall support generating QR Code for files.
REQ-5.6: The system shall support sending files via QR Code.
REQ-5.7: The system shall provide encryption repair functionality (deleting previously stored keys).
REQ-5.8: The system shall support generating public encryption keys.

3.2.6 Search and Filtering:

REQ-6.1: The system shall allow users to search files by name.
REQ-6.2: The system shall allow users to search files by type.
REQ-6.3: The system shall allow users to search files by upload date.
REQ-6.4: The system shall allow filtering files by name.
REQ-6.5: The system shall allow filtering files by type.
REQ-6.6: The system shall allow filtering files by upload date.

3.2.7 Notifications and Alerts:

REQ-7.1: The system shall notify users when a new file is received.
REQ-7.2: The system shall notify users when a file is deleted.
REQ-7.3: The system shall notify users when a file is modified.
REQ-7.4: The system shall alert users when storage space is nearing full capacity.

3.2.8 Centralized Assistance:

- **REQ-8.1:** The system shall support text-based interaction with users.
- **REQ-8.2:** The system shall assist users in any operation they want to perform.

3.2.9 Artificial Intelligence Requirements:

- **REQ-9.1:** The system shall automatically classify files into folders based on file type.
- **REQ-9.2:** The system shall improve file search functionality.
- **REQ-9.3:** The system shall suggest appropriate names for files upon upload or receipt.
- **REQ-9.4:** The system shall provide a recommendation system to suggest related files that the user may need.

3.2.10 Admin Additional Requirements:

- **REQ-10.1:** The system shall allow the admin to track login history.
- **REQ-10.2:** The system shall allow the admin to delete user accounts.
- **REQ-10.3:** The system shall allow the admin to track all file activities.
- **REQ-10.4:** The system shall allow the admin to monitor file encryption status.
- **REQ-10.5:** The system shall allow the admin to view file details.
- **REQ-10.6:** The system shall allow the admin to view recent logs.
- **REQ-10.7:** The system shall allow the admin to export log data.
- **REQ-10.8:** The system shall allow the admin to export report data in multiple formats.

4. Non-Functional Requirement :

4.1 Scalability

- **NF-1.1**: The system shall be scalable to accommodate an increasing number of users and shared files without affecting performance.
- **NF-1.2**: The system architecture shall support dynamic scaling through the addition of storage space or computing resources.

4.2 Availability

- **NF-2.1**: The system shall maintain high availability to ensure users can always access file sharing features.
- **NF-2.2**: The system shall be resilient to component failures and maintain basic file operations even during partial outages.

4.3 Interoperability

- **NF-3.1**: The system shall support compatibility with government or financial oversight platforms as needed.
- **NF-3.2**: The system shall enable secure file exchange between internal and authorized external parties.

4.4 Security

- **NF-4.1**: The system shall enforce authentication using strong password policies and session expiration.
- **NF-4.2**: All files and sensitive data (including user data and logs) shall be encrypted in transit and at rest.
- **NF-4.3**: The system shall support activity auditing and tamper-proof logging for compliance with regulatory requirements.

4.5 Extensibility

- **NF-5.1**: The system shall support the future addition of secure file transfer protocols (e.g., HTTPS, SFTP, Wi-Fi Direct upgrades).
- **NF-5.2**: The architecture shall allow integration of future modules such as document scanning or AI-based anomaly detection without reworking core components.

5. Requirements modeling :

1.1 Basic UML Diagrams (Hight Level):

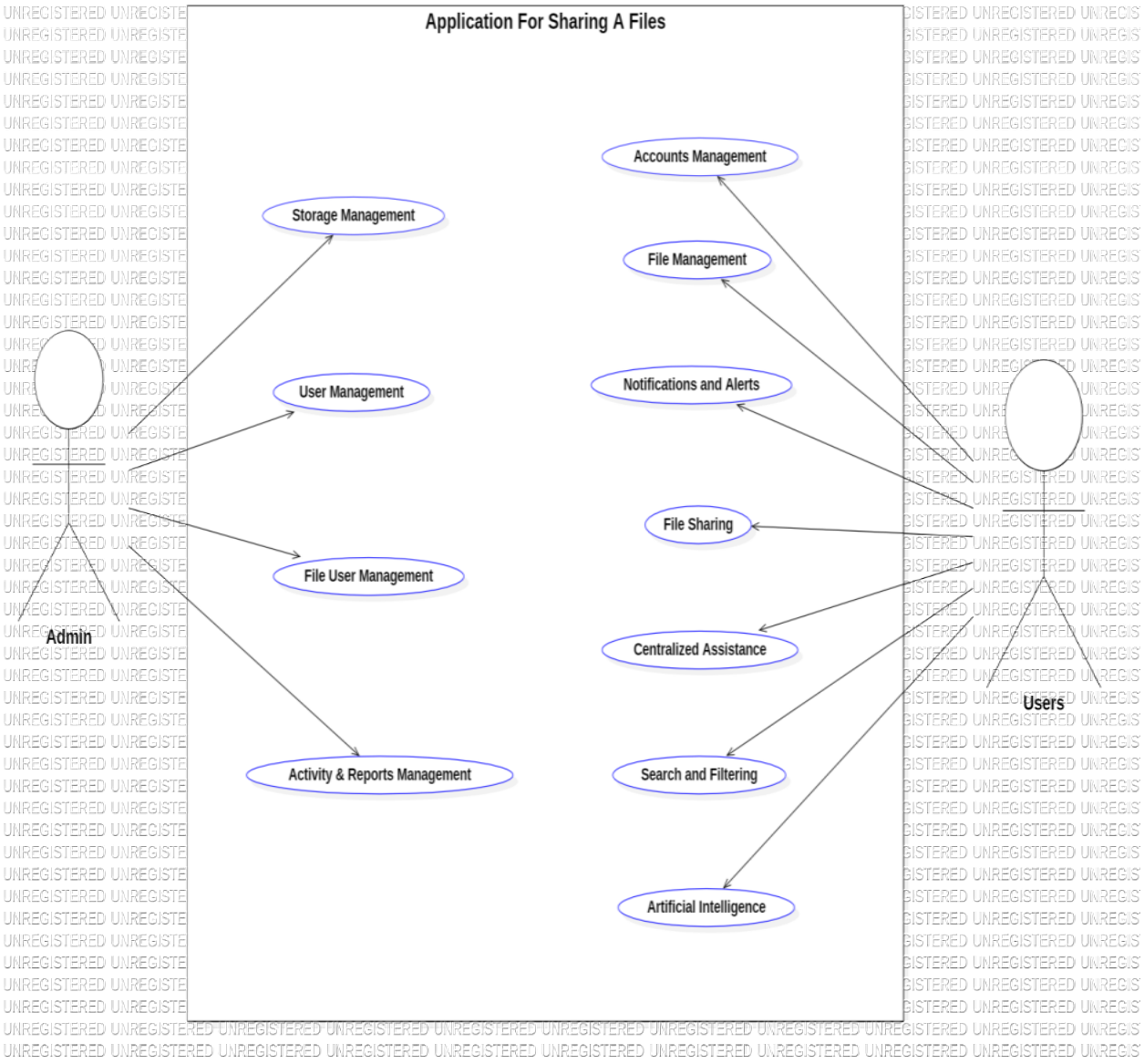


Figure 2 - Use Case Diagram (Hight Level)

1.2 Basic UML Diagrams (Low Level):

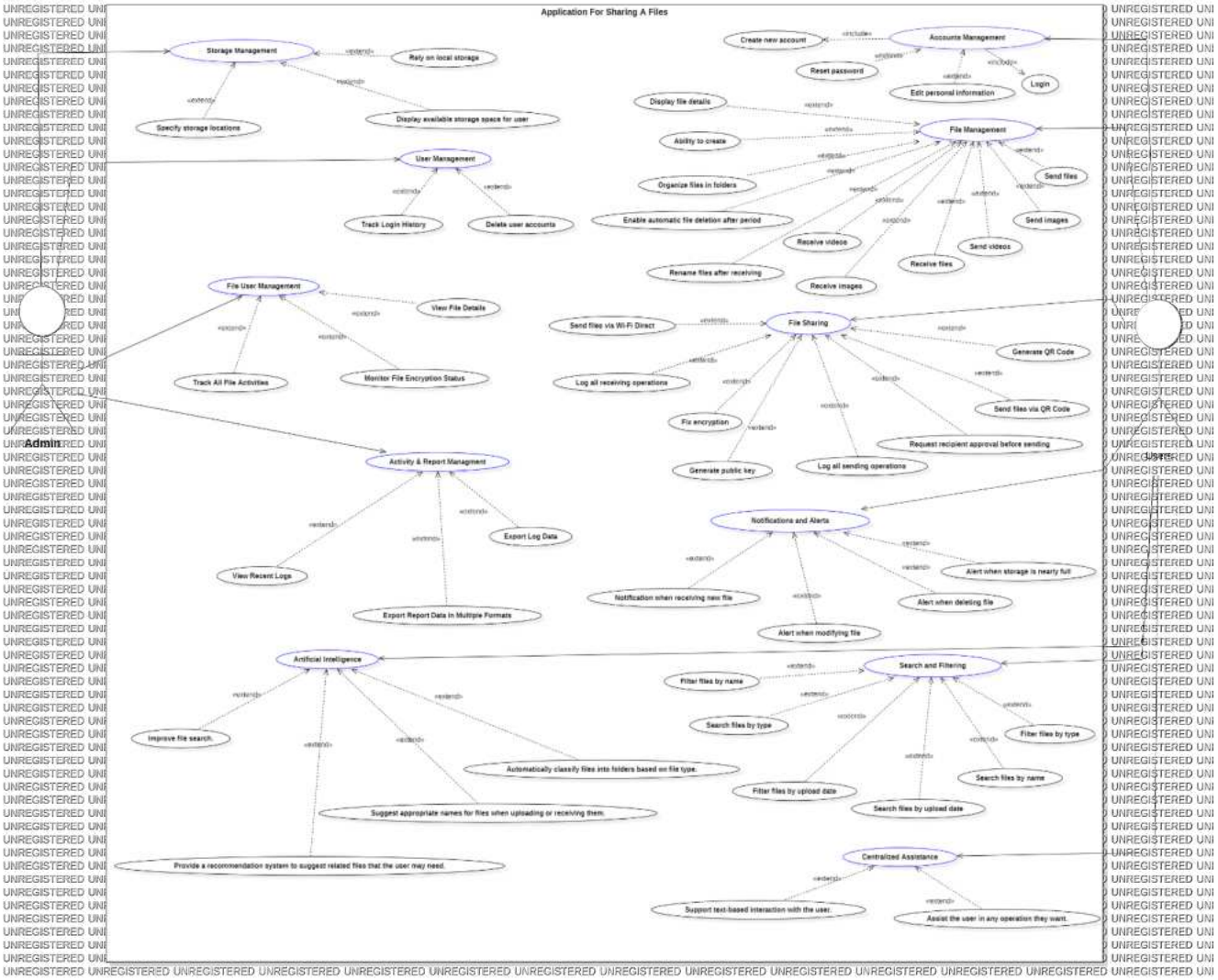


Figure 3 - UML Diagrams (Low Level)

2 - Use Case Specifications:

2.1 Accounts Management :

2.1.1 Create New Account :

Name	Create New Account
Short Description	Allows unregistered users to create a new account by providing necessary personal information such as name, email, and password.
Precondition	- The user is not currently registered in the system. - A working internet connection.
Postcondition	- A new user account is created in the system. - The user is redirected to the login page or logged in automatically and redirected to their homepage.
Actors	- User
Main Success Scenario	1. The user accesses the registration page. 2. The user fills in all required fields (e.g., name, email, password). 3. The user submits the form. 4. The system validates the information. 5. If valid, the account is created and stored in the database. 6. The user is redirected to the login page or automatically logged in.
Alternative Process	- If the email already exists, the system displays an error and prevents account creation. - If mandatory fields are left empty or invalid, an error message is shown. - If the password is weak, a prompt is shown to choose a stronger one.

Table 4- Use Case specification 'Create New Account'

2.1.2 Login :

Name	Login to Platform
Short Description	Allows registered users to log into their accounts by entering valid credentials (email/username and password).
Precondition	- The user must have an existing and active account. - A stable internet connection is required.
Postcondition	- The user is successfully authenticated and redirected to their role-based dashboard (Admin, Support Team Manager, Support Team Member, or Client).
Actors	- User - Admin
Main Success Scenario	1. The user opens the login page. 2. The user enters their email/username and password. 3. The user clicks the "Login" button. 4. The system validates the credentials. 5. If valid, the system redirects the user to their corresponding dashboard.
Alternative Process	- If credentials are incorrect, the system shows an error message. - If account is deactivated, the system displays an access denial message. - If the user forgets their password, they can use the "Forgot Password" feature to reset it via email.

Table 5- Use Case specification 'Login'

2.1.3 Edit Personal Information:

Name	Edit Personal Information
Short Description	Allows authenticated users to edit their personal account information such as name, email, phone number, or password.
Precondition	- The user must be logged into the system. - The account must be active.
Postcondition	- The user's updated information is saved successfully and reflected in the system.
Actors	- User - Admin
Main Success Scenario	1. The user accesses their account settings. 2. The user updates one or more fields (e.g., name, email, phone, password). 3. The user clicks the "Save" or "Update" button. 4. The system validates the input. 5. If valid, the system updates the user's information and confirms the changes.
Alternative Process	- If the input is invalid (e.g., email format incorrect), the system shows a validation error. - If the update fails due to a server issue, the system displays an appropriate error message.

Table 6- Use Case specification 'Edit personal Information'

2.1.4 Reset Password:

Field	Description
Name	Reset Password
Short Description	Allows users to reset their account password in case they forget it, using their registered email.
Precondition	- The user must have a registered account. - The user must have access to the registered email address.
Postcondition	- A new password is created and the user is able to log in using the updated credentials.
Actors	- User - Admin
Main Success Scenario	1. User clicks on “Forgot Password” from the login page. 2. System prompts the user to enter their registered email. 3. User submits the email. 4. System sends a reset link or OTP to the email. 5. User clicks the link or enters the OTP. 6. User enters a new password and confirms it. 7. System updates the password and notifies the user.
Alternative Process	- If the email is not found in the system, an error message is shown. - If the OTP is incorrect or expired, user is prompted to request a new one.

Table 7- Use Case specification 'Reset Password'

2.2 System Administrator Management :

2.2.1 Activity Logs Monitoring:

Field	Description
Name	Monitor Activity Logs
Short Description	Allows authorized users (e.g., Admin, Support Team Manager) to view logs of user and system activities such as file access, login events, and ticket updates.
Precondition	- The user must have appropriate access privileges. - The system must be actively logging events.
Postcondition	- The user can view a list of recorded activities. - The logs may be filtered, searched, or exported depending on access rights.
Actors	- Admin
Main Success Scenario	1. Authorized user logs into the system. 2. Navigates to the “Activity Logs” section. 3. System fetches logs based on filters (date, user, action). 4. Logs are displayed with timestamps, user actions, and related objects. 5. The user may export the log if allowed.
Alternative Process	- If no logs are found for the selected filters, the system shows an appropriate message. - If access rights are insufficient, access is denied.

Table 8 - Use Case specification 'Activity Logs Monitoring'

2.2.2 Generate Activity & Log Reports:

Field	Description
Name	Generate Activity & Log Reports
Short Description	Allows authorized users to generate and export detailed reports based on user activities and system logs within a selected date or action range.
Precondition	- The user must have administrative or managerial access. - The system must have stored activity logs.
Postcondition	- A downloadable report (e.g., PDF/CSV) is generated containing filtered logs and summaries of actions performed.
Actors	- Admin
Main Success Scenario	1. User logs into the system with appropriate privileges. 2. Navigates to the “Reports” section. 3. Selects filters (date range, user, action type). 4. System generates a preview of the report. 5. User clicks “Export”. 6. Report is downloaded in the selected format.
Alternative Process	- If no logs match the filters, a message like "No data found for selected criteria" is displayed. - If system storage is corrupted, display error.

Table 9 - Use Case specification 'Generate Activity & Log Reports'

2.2.3 Set Access Permissions for Each User on Files:

Field	Description
Name	Set Access Permissions for Each User on Files
Short Description	Allows file owners to define specific access levels (view, download, forward, delete) for each user or group on shared files.
Precondition	- The user must be authenticated and authorized.- The file must be uploaded to the system. - The user must be the file owner or admin.
Postcondition	- Access permissions are saved and enforced by the system. - Users with granted permissions can interact with the file according to their access level.
Actors	- Admin
Main Success Scenario	1. User navigates to their uploaded files. 2. Selects a file and clicks on “Manage Permissions”. 3. Chooses target users or groups. 4. Defines permission levels (e.g., view only, view & download, full control). 5. Confirms and saves changes. 6. System stores the permission settings and applies them.
Alternative Process	- If the user tries to assign permissions without being the owner/admin, access is denied. - If the selected user does not exist, system shows an error.

Table 10 - Use Case specification 'Set Access Permissions For Each User On Files'

2.2.4 Delete User Account:

Field	Description
Name	Delete User Account
Short Description	Allows authenticated users to permanently delete their account and all associated data from the system.
Precondition	- The user is authenticated. - The user has an active account.
Postcondition	- The user account and related data are permanently deleted from the system. - The user is logged out and redirected to the welcome page.
Actors	-Admin
Main Success Scenario	1. User navigates to the account settings page. 2. User clicks on "Delete Account". 3. System asks for confirmation. 4. User confirms deletion. 5. System deletes the account and logs the user out. 6. System redirects the user to the homepage with a confirmation message.
Alternative Process	- If the user cancels the confirmation step, the deletion is aborted and the account remains active.

Table 11 - Use Case specification 'Delete User Account'

2.3 File Management :

2.3.1 Send Files:

Field	Description
Name	Send Files
Short Description	Allows users to securely send encrypted files to other users.
Precondition	- The sender must be authenticated. - The sender must have a file ready to upload.
Postcondition	- The file is encrypted and stored. - The file is securely sent to the recipient. - The recipient is notified about the received file.
Actors	- User - Admin
Main Success Scenario	1. The sender logs into the system. 2. The sender navigates to the “Send Files” section. 3. The sender selects a file to send. 4. The sender chooses the recipient. 5. The system encrypts the file. 6. The system sends the file securely. 7. The recipient is notified.
Alternative Process	- If the file type is not supported, the system displays an error message. - If the recipient is not found, the system prompts the sender to re-enter recipient information.

Table 12 - Use Case specification 'Send Files'

2.3.2 Send Images:

Field	Description
Name	Send Images
Short Description	Allows users to securely send encrypted image files to other users.
Precondition	- The sender is authenticated. - The sender has one or more image files prepared.
Postcondition	- The image is encrypted using a secure algorithm. - The encrypted image is stored and transmitted to the intended recipient. - The recipient is notified and can access the image.
Actors	-User - Admin
Main Success Scenario	1. The sender logs in successfully. 2. The sender navigates to the image sending section. 3. The sender selects one or more image files. 4. The sender chooses the recipient. 5. The system encrypts the images. 6. The encrypted images are sent to the recipient. 7. The recipient receives a notification.
Alternative Process	- If an image exceeds the size limit, the system displays a warning message. - If the file is not a supported image format (e.g., JPG, PNG), the system prevents sending and displays an error. - If the recipient is offline, the system queues the message until they are online.

Table 13 - Use Case specification 'Send Images'

2.3.3 Send Videos:

Field	Description
Name	Send Videos
Short Description	Allows users to securely send encrypted video files to other users.
Precondition	- The sender is authenticated. - The sender has one or more video files prepared.
Postcondition	- The video is encrypted using a secure algorithm. - The encrypted video is stored and transmitted to the recipient. - The recipient is notified and can download/view the video.
Actors	-User - Admin
Main Success Scenario	1. The sender logs into the system. 2. The sender accesses the video sending section. 3. The sender selects one or more video files. 4. The sender chooses the recipient. 5. The system encrypts the video files. 6. The encrypted videos are sent to the recipient. 7. The recipient is notified and can access the video.
Alternative Process	- If the video size exceeds the system's limit, a warning message is shown. - If the video format is unsupported (e.g., unsupported codec), the system rejects the file. - If the recipient is offline, the system queues the delivery until they are online.

Table 14 - Use Case specification 'Send Videos'

2.3.4 Receive Files:

Field	Description
Name	Receive File
Short Description	Enables a user to receive an encrypted file sent by another user securely.
Precondition	- The user is registered and authenticated. - A file has been sent to the user by another authorized user.
Postcondition	- The user receives the encrypted file. - The file is stored securely. - The user is notified about the received file.
Actors	-User - Admin
Main Success Scenario	1. The sender uploads and sends an encrypted file. 2. The system stores the encrypted file securely. 3. The recipient logs into the platform. 4. The system notifies the recipient of the new file. 5. The recipient accesses the file download area. 6. The recipient decrypts the file (manually or automatically based on system settings). 7. The file is successfully accessed or saved locally.
Alternative Process	- If the file has expired before being accessed, the system shows an expiry message. - If the decryption key is invalid or missing, the system displays an error and blocks access. - If the recipient is offline, the file remains stored until the next login.

Table 15 - Use Case specification 'Receive Files'

2.3.5 Receive Images:

Field	Description
Name	Receive Images
Short Description	Allows a user to securely receive encrypted image files sent by another user.
Precondition	- The user is authenticated. - An encrypted image has been sent to the user.
Postcondition	- The user successfully receives and decrypts the image. - The image is displayed or downloaded securely. - The activity is logged.
Actors	-User - Admin
Main Success Scenario	1. The sender uploads an image file and initiates encryption. 2. The system securely stores the encrypted image. 3. The receiver logs into the system. 4. The system notifies the receiver of the new image. 5. The receiver navigates to the image section. 6. The receiver decrypts the image and views or downloads it.
Alternative Process	- If the decryption key is invalid or missing, the system displays an error. - If the file format is not supported, the system rejects the image. - If the receiver does not open the image within a defined time, the file may expire and be deleted automatically.

Table 16 - Use Case specification 'Receive Images'

2.3.6 Receive Videos:

Field	Description
Name	Receive Videos
Short Description	Allows a user to securely receive encrypted video files sent by another user.
Precondition	- The user is authenticated. - An encrypted video has been sent to the user.
Postcondition	- The user successfully receives and decrypts the video. - The video is either streamed securely or downloaded. - The system logs the activity.
Actors	-User - Admin
Main Success Scenario	1. The sender uploads a video file and encrypts it. 2. The system stores the encrypted video securely. 3. The receiver logs in to the platform. 4. The system notifies the receiver about the new video. 5. The receiver accesses the video section. 6. The receiver decrypts and streams or downloads the video.
Alternative Process	- If the decryption key is missing or incorrect, an error message is shown. - If the video format is not supported, the file is rejected. - If the video is not accessed within a set period, it may self-destruct.

Table 17 - Use Case specification 'Receive Videos'

2.3.7 Rename Received Files:

Field	Description
Name	Rename Received Files
Short Description	Allows the user to rename a file after successfully receiving it.
Precondition	- The user has received the file. - The user is authenticated and authorized.
Postcondition	- The file name is updated. - The new name is reflected in the user interface. - The change is logged in the activity log.
Actors	- User - Admin
Main Success Scenario	1. The user navigates to the list of received files. 2. The user selects a specific file. 3. The user clicks the rename option. 4. The system prompts for a new name. 5. The user enters the new name. 6. The system saves the new name and confirms the change.
Alternative Process	- If the new name already exists, the system notifies the user and asks for another name. - If the new name contains invalid characters, the system displays an error message.

Table 18 - Use Case specification 'Rename Received Files'

2.3.8 Organize Files into Folders:

Field	Description
Name	Organize Files into Folders
Short Description	Allows the user to create, manage, and organize files within folders and subfolders to improve file management and retrieval.
Precondition	- The user is authenticated and authorized. - The user has access to the file management interface.
Postcondition	- New folders are created and displayed. -Files are properly organized within folders. - Folder structure is saved and reflected in the UI. - Changes are logged.
Actors	- User - Admin
Main Success Scenario	1. The user navigates to the file management section. 2. The user creates a new folder or selects an existing folder. 3. The user uploads or moves files into the folder. 4. The system saves the folder structure and files. 5. The system updates the interface to show organized files.
Alternative Process	- If folder name already exists, system notifies the user to choose another name. - If the folder structure exceeds allowed depth, system shows an error. - If the user attempts unauthorized access, system denies the operation.

Table 19 - Use Case specification 'Organize Files Into Folders'

2.3.9 View File Details:

Field	Description
Name	View File Details
Short Description	Allows the user to view detailed information about a selected file, including metadata such as name, size, type, and upload date.
Precondition	- The user is authenticated and authorized. - The user has access to the files.
Postcondition	- File details are displayed correctly in the user interface. - The viewed action is optionally logged in the activity log.
Actors	- User - Admin
Main Success Scenario	1. The user navigates to the list of files. 2. The user selects a specific file. 3. The system retrieves file details. 4. The system displays the details to the user.
Alternative Process	- If the file does not exist or access is denied, the system shows an error message. - If the system fails to retrieve details, it notifies the user.

Table 20 - Use Case specification 'View File Details'

2.3.10 Enable Automatic File Deletion After a Period:

Field	Description
Name	Enable Automatic File Deletion After a Period
Short Description	Allows the user or system administrator to activate automatic deletion of files after a specified time period.
Precondition	- The user is authenticated and authorized. - The file exists in the system. - The automatic deletion feature is configurable.
Postcondition	- Files are deleted automatically after the configured period. - Deletion actions are logged in the activity log. - Users are notified if applicable.
Actors	- User - Admin
Main Success Scenario	1. The user or admin accesses the file management settings. 2. The user/admin enables automatic deletion for specific files or folders. 3. The user/admin sets the deletion period. 4. The system schedules deletion. 5. After the period expires, the system deletes the files. 6. The system logs the deletion and notifies users if needed.
Alternative Process	- If the deletion period is invalid, the system prompts for correction. - If deletion fails due to system error, an alert is generated. - If user lacks permission, the action is denied.

Table 21 - Use Case specification 'Enable Automatic File Deletion After A Period'

2.3.11 Create New Folder:

Field	Description
Name	Create New Folder
Short Description	Allows the user to create a new folder to organize files within the file management system.
Precondition	- The user is authenticated and authorized. - The user has access to the directory where the folder will be created.
Postcondition	- A new folder is created and visible in the file structure. - The folder creation action is logged in the activity log.
Actors	- User - Admin
Main Success Scenario	1. The user navigates to the desired location in the file system. 2. The user selects the option to create a new folder. 3. The system prompts for the folder name. 4. The user enters a valid folder name. 5. The system creates the folder and confirms success. 6. The folder appears in the file structure.
Alternative Process	- If the folder name already exists, the system requests a different name. - If the folder name contains invalid characters, the system shows an error. - If the user lacks permission, the system denies the creation.

Table 22 - Use Case specification 'Create New Folder'

2.4 Storage Management :

2.4.1 Local Storage Dependency :

Field	Description
Name	Local Storage Dependency
Short Description	The system shall store and retrieve files using local storage on the user's device to enable offline access.
Precondition	- The user has a device with sufficient local storage available. - The user has granted permission for local storage usage.
Postcondition	- Files are saved and accessible locally. - The system syncs local changes with the server when online. - Data integrity and consistency are maintained.
Actors	- User - Admin
Main Success Scenario	1. The user uploads or modifies a file. 2. The system saves the file to local storage. 3. The user accesses files offline. 4. When the device is online, the system syncs changes with the server. 5. The system confirms synchronization success.
Alternative Process	- If local storage is full, the system notifies the user. - If syncing fails, the system retries or alerts the user. - If permission is denied, offline features are disabled.

Table 23 - Use Case specification 'Local Storage Dependency'

2.4.2 Display Available Storage Space:

Field	Description
Name	Display Available Storage Space
Short Description	Allows the user to view the current available storage space on their device or account to manage file uploads effectively.
Precondition	- The user is authenticated. - The system can access storage usage data.
Postcondition	- The user interface displays the accurate available storage space. - Storage usage updates dynamically after file operations.
Actors	- User - Admin
Main Success Scenario	1. The user navigates to the storage information section. 2. The system retrieves current storage usage and limits. 3. The system displays available and used storage space to the user.
Alternative Process	- If storage data cannot be retrieved, the system shows an error message. - If user permission is denied, the storage info is not displayed.

Table 24 - Use Case specification 'Display Available Storage Space'

2.4.3 Specify Storage Locations:

Field	Description
Name	Specify Storage Locations
Short Description	Enables the system or user to select or define storage locations (e.g., local device, cloud, external drives) for files.
Precondition	- The user is authenticated and authorized. - Storage options are available and accessible.
Postcondition	- Storage locations are configured and saved. - Files are stored in the selected locations accordingly.
Actors	- User - Admin
Main Success Scenario	1. The user accesses storage settings. 2. The user selects one or more storage locations. 3. The system validates and saves the selection. 4. Files uploaded or managed use the specified storage locations.
Alternative Process	- If a selected storage location is unavailable, the system prompts the user to choose another. - If permissions are insufficient, the action is denied.

Table 25 - Use Case specification 'Specify Storage Locations'

2.5 File Sharing :

2.5.1 Send Files Via Wi-Fi Direct:

Field	Description
Name	Send Files via Wi-Fi Direct
Short Description	Allows the user to send files directly to another device using Wi-Fi Direct technology without needing an internet connection.
Precondition	- Both devices support Wi-Fi Direct. - The user is authenticated and authorized. - Devices are within Wi-Fi Direct range.
Postcondition	- Files are sent successfully to the recipient device. - The transfer status is logged. - The user is notified of success or failure.
Actors	-User - Admin
Main Success Scenario	1. The sender selects files to send. 2. The system establishes a Wi-Fi Direct connection. 3. The sender initiates the transfer. 4. Files are transmitted. 5. The system logs the transfer and notifies the sender.
Alternative Process	- If connection fails, system retries or notifies the sender. - If files cannot be sent, system reports error. - If recipient rejects connection, transfer is aborted.

Table 26 - Use Case specification 'Send Files via Wi-Fi Direct'

2.5.2 Request Recipient Approval Before Sending:

Field	Description
Name	Request Recipient Approval Before Sending
Short Description	Requires the recipient to approve the file transfer request before the sender can send files.
Precondition	- Recipient device is reachable. - User is authenticated and authorized. - Wi-Fi Direct connection is established.
Postcondition	- Transfer proceeds only after recipient approval. - Denials are logged and notified. - Transfer is aborted if rejected.
Actors	-User - Admin
Main Success Scenario	1. Sender requests approval. 2. Recipient receives prompt. 3. Recipient approves. 4. Transfer initiates. 5. System logs approval and transfer.
Alternative Process	- If recipient denies, system aborts transfer and logs action. - If no response within timeout, transfer is canceled.

Table 27 - Use Case specification 'Request Recipient Approval Before Sending'

2.5.3 Log All Sending Operations:

Field	Description
Name	Log All Sending Operations
Short Description	Logs every file sending operation with relevant details such as sender, recipient, time, and status.
Precondition	- The sending operation is initiated. - Logging system is operational.
Postcondition	- All sending actions are recorded in the activity log for audit purposes.
Actors	- Admin
Main Success Scenario	1. A sending operation starts. 2. System records details. 3. Logs are stored securely. 4. Logs are retrievable for audits.
Alternative Process	- If logging fails, system alerts admin and retries.

Table 28 - Use Case specification 'Log All Sending Operations'

2.5.4 Log All Receiving Operations:

Field	Description
Name	Log All Receiving Operations
Short Description	Logs every file receiving operation with details such as sender, recipient, time, and status.
Precondition	- Receiving operation is initiated. - Logging system is operational.
Postcondition	- All receiving actions are recorded in the activity log for audit purposes.
Actors	- Admin
Main Success Scenario	1. A receiving operation starts. 2. System records details. 3. Logs are stored securely. 4. Logs are retrievable for audits.
Alternative Process	- If logging fails, system alerts admin and retries.

Table 29 - Use Case specification 'Log All Receiving Operations'

2.5.5 Generate QR Code:

Field	Description
Name	Generate QR Code
Short Description	Allows the system to generate a QR code that encodes file transfer information for easy sharing.
Precondition	- User requests QR code generation. - Relevant data for QR code is available.
Postcondition	- A valid QR code is generated and displayed to the user. - QR code can be scanned by recipient devices.
Actors	- User - Admin
Main Success Scenario	1. User selects files or transfer info. 2. System generates QR code. 3. QR code is displayed. 4. Recipient scans QR code to initiate transfer.
Alternative Process	- If generation fails, system shows error and retry option.

Table 30 - Use Case specification 'Generate QR Code'

2.5.6 Send Files via QR Code:

Field	Description
Name	Send Files via QR Code
Short Description	Enables sending files by scanning a QR code containing transfer information, facilitating quick connection setup.
Precondition	- Recipient scans valid QR code. - Both devices are ready for transfer. - User authorized.
Postcondition	- Files are sent successfully after QR code initiation. - Transfer status logged. - User notified.
Actors	-User - Admin
Main Success Scenario	1. Sender generates QR code. 2. Recipient scans QR code. 3. Devices establish connection. 4. Transfer proceeds. 5. System logs transfer and notifies users.
Alternative Process	- If scan or connection fails, system prompts retry or cancels transfer.

Table 31 - Use Case specification 'Send Files via QR Code'

2.5.7 Fix Encryption:

Field	Description
Name	Fix Encryption
Short Description	Addresses and resolves encryption issues to ensure secure data transmission during file transfers.
Precondition	- Encryption module is implemented. - Detected encryption problems exist.
Postcondition	- Encryption errors are fixed. - Data transmission is secure. - System logs encryption status.
Actors	- User -Admin
Main Success Scenario	1. System detects encryption issue. 2. Administrator applies fixes. 3. Encryption is verified. 4. System confirms secure transfer capability.
Alternative Process	- If fixing fails, alerts are generated and admin is notified. - System switches to fallback encryption if available.

Table 32 - Use Case specification 'Fix Encryption'

2.5.8 Generate Public Key:

Field	Description
Name	Generate Public Key
Short Description	Enables the system to generate a public cryptographic key used for secure encryption during file transfers.
Precondition	- Cryptography module is operational. - User or system requests key generation.
Postcondition	- A valid public key is generated and stored securely. - Key is available for encryption processes.
Actors	- User
Main Success Scenario	1. User/system initiates key generation. 2. System generates key. 3. Key is stored and ready for use. 4. System confirms key availability.
Alternative Process	- If generation fails, system notifies user/admin. - Retry or fallback options are provided.

Table 33 - Use Case specification 'Generate Public Key'

2.6 Search & Filtering :

2.6.1 Search Files By Name:

Field	Description
Name	Search Files by Name
Short Description	Allows users to search for files using file names as keywords.
Precondition	- User is authenticated. - Files exist in the system.
Postcondition	- The system returns a list of files matching the search name. - Results are displayed to the user.
Actors	- User
Main Success Scenario	1. User enters search keyword. 2. System searches files by name. 3. System displays matching results.
Alternative Process	- If no matches found, system shows "No results found" message. - If input invalid, system requests correction.

Table 34 - Use Case specification 'Search Files by Name'

2.6.2 Search Files By Type:

Field	Description
Name	Search Files by Type
Short Description	Allows users to search for files based on their file types (e.g., PDF, DOCX, JPEG).
Precondition	- User is authenticated. - Files have associated type metadata.
Postcondition	- System returns files matching the specified type. - Results are displayed.
Actors	- User
Main Success Scenario	1. User selects or inputs file type. 2. System searches files by type. 3. Results are shown.
Alternative Process	- If no files match the type, system displays a no-results message. - If invalid type input, system requests correction.

Table 35 - Use Case specification 'Search Files by Type'

2.6.3 Search Files By Upload Date:

Field	Description
Name	Search Files by Upload Date
Short Description	Allows users to search for files uploaded within a specific date or date range.
Precondition	- User is authenticated. - Files have upload date metadata.
Postcondition	- System returns files uploaded in the specified date range. - Results are displayed.
Actors	- User
Main Success Scenario	1. User inputs or selects date range. 2. System searches files by upload date. 3. Results are displayed.
Alternative Process	- If no files found, system displays no-results message. - If date input is invalid, system requests correction.

Table 36 - Use Case specification 'Search Files by Upload Date'

2.6.4 Filters Files By Name:

Field	Description
Name	Filter Files by Name
Short Description	Allows users to filter the current file list dynamically based on partial or full file name input.
Precondition	- User is authenticated. - Files are loaded/displayed in the interface.
Postcondition	- File list is updated to show only files matching the name filter.
Actors	- User
Main Success Scenario	1. User enters filter text. 2. System dynamically filters files by name. 3. Filtered list is shown.
Alternative Process	- If no files match filter, system shows "No files found".

Table 37 - Use Case specification 'Filters Files By Name'

2.6.5 Filters Files By Type:

Field	Description
Name	Filter Files by Type
Short Description	Allows users to filter the displayed files by selecting file types from available options.
Precondition	- User is authenticated. - Files are loaded/displayed with type metadata.
Postcondition	- File list updates to show files only of the selected types.
Actors	- User
Main Success Scenario	1. User selects file type filter. 2. System filters displayed files accordingly. 3. Updated list shown.
Alternative Process	- If no files match the filter, system shows "No files found".

Table 38 - Use Case specification 'Filters Files By Type'

2.6.6 Filters Files By Upload Date:

Field	Description
Name	Filter Files by Upload Date
Short Description	Allows users to filter displayed files based on upload date or date ranges dynamically.
Precondition	- User is authenticated. - Files have upload date metadata and are displayed.
Postcondition	- File list updates to show files within the selected date range.
Actors	- User
Main Success Scenario	1. User selects or inputs date range. 2. System filters files accordingly. 3. Filtered list is displayed.
Alternative Process	- If no files match the filter, system shows "No files found".

Table 39 - Use Case specification 'Filters Files By Upload Date'

2.7 Notifications & Alerts :

2.7.1 Notification on New File Received:

Field	Description
Name	Notification on New File Received
Short Description	Sends a notification to the user when a new file is received in their account or shared folder.
Precondition	- User is authenticated. - New file is successfully uploaded or shared.
Postcondition	- User receives a notification alert about the new file. - Notification details are logged.
Actors	- User
Main Success Scenario	1. A new file is uploaded or shared. 2. System generates notification. 3. Notification is delivered to the user. 4. User views the notification.
Alternative Process	- If notification fails, system retries or logs the failure.

Table 40 - Use Case specification 'Notification on New File Received'

2.7.2 Alert on File Deletion:

Field	Description
Name	Alert on File Deletion
Short Description	Sends an alert to the user or administrator when a file is deleted from the system.
Precondition	- User is authenticated. - File deletion action is initiated and successful.
Postcondition	- Alert is sent to relevant parties. - Deletion event is logged.
Actors	- User - Administrator
Main Success Scenario	1. File deletion occurs. 2. System generates alert. 3. Alert is delivered to user/admin. 4. Event is logged.
Alternative Process	- If alert delivery fails, system retries or logs the failure.

Table 41 - Use Case specification 'Alert on File Deletion'

2.7.3 Alert on File Modification:

Field	Description
Name	Alert on File Modification
Short Description	Notifies users or administrators when a file has been modified.
Precondition	- User is authenticated. - File modification is completed.
Postcondition	- Notification alert is sent. - Modification event is logged.
Actors	- User - Administrator
Main Success Scenario	1. File modification detected. 2. System creates alert. 3. Alert is sent. 4. Event is logged.
Alternative Process	- If notification fails, system retries or logs the failure.

Table 42 - Use Case specification 'Alert on File Modification'

2.7.4 Storage Near Capacity Warning:

Field	Description
Name	Storage Near Capacity Warning
Short Description	Warns users when their available storage space is nearing full capacity, prompting them to take action.
Precondition	- User storage is monitored. - Threshold for capacity warning is defined.
Postcondition	- Warning notification is sent when storage exceeds threshold. - Warning events are logged.
Actors	- User
Main Success Scenario	1. System monitors storage usage. 2. Threshold exceeded. 3. Warning notification sent to user. 4. Event logged.
Alternative Process	- If warning delivery fails, system retries or logs the failure.

Table 43 - Use Case specification 'Storage Near Capacity Warning'

2.8 Central Assistance:

2.8.1 Support text-based interaction with the user:

Field	Description
Name	Support text-based interaction with the user
Short Description	Enables the system to provide centralized assistance by allowing users to interact with it through text-based communication.
Precondition	- User is logged into the system. - Assistance module is active and accessible.
Postcondition	- User receives relevant responses to queries or assistance requests. - Interaction is logged for future reference.
Actors	- User
Main Success Scenario	1. User initiates a text-based query or request. 2. System analyzes the input. 3. System generates an appropriate response. 4. Response is delivered to the user. 5. Interaction is logged.
Alternative Process	- If the system cannot understand the query, it asks for clarification. - If assistance service is unavailable, an error message is sent and logged.

Table 44 -Use Case Specification 'Support text-based interaction with the user'

2.8.2 Assist the user in any operation they want:

Field	Description
Name	Assist the user in any operation they want
Short Description	Provides step-by-step guidance or automated support to help the user complete any operation within the system.
Precondition	- User is logged into the system. - Assistance module is active and accessible.
Postcondition	- The user successfully completes the requested operation with system assistance. - Assistance session is logged.
Actors	- User
Main Success Scenario	1. User requests assistance for a specific operation. 2. System identifies the requested operation. 3. System provides step-by-step guidance or automated support. 4. User follows the guidance and completes the operation. 5. System logs the assistance session.
Alternative Process	- If the system cannot identify the operation, it asks the user for clarification. - If the requested operation is not supported, the system notifies the user and suggests alternatives. - If assistance service is temporarily unavailable, an error message is displayed and logged.

Table 45 - Use Case Specification 'Assist the user in any operation they want'

2.9 Artificial Intelligence Requirements:

2.9.1 Automatically classify files into folders based on file type:

Field	Description
Name	Automatic File Classification
Short Description	Automatically classifies files into appropriate folders depending on their type (e.g., documents, images, videos).
Precondition	- User uploads or receives files. - Classification module is active.
Postcondition	- Files are placed into the correct folders automatically. - Classification action is logged.
Actors	- User
Main Success Scenario	1. User uploads or receives a file. 2. System detects file type. 3. System creates or selects the correct folder. 4. File is stored in the appropriate folder. 5. Action is logged.
Alternative Process	- If file type is unknown, system prompts the user to select a folder. - If classification fails, file is stored in a default folder.

Table 46 - Use Case Specification 'Automatically classify files into folders based on file type'

2.9.2 Improve file search:

Field	Description
Name	Enhanced File Search
Short Description	Improves the accuracy and relevance of search results using AI-powered techniques.
Precondition	- User requests a file search. - Search index is available.
Postcondition	- System returns improved, relevant search results. - Search queries are logged.
Actors	- User
Main Success Scenario	1. User enters a search query. 2. System analyzes the query using AI-based techniques. 3. System retrieves and ranks relevant files. 4. Results are displayed to the user. 5. Query is logged.
Alternative Process	- If no results are found, system suggests alternative keywords. - If search index is unavailable, fallback to basic search is applied.

Table 47 - Use Case Specification 'Improve file search'

2.9.3 Suggest appropriate names for files when uploading or receiving them:

Field	Description
Name	File Name Suggestion
Short Description	Suggests meaningful names for files during upload or reception based on file content and type.
Precondition	- User uploads or receives a file. - AI naming module is active.
Postcondition	- Suggested name is displayed to the user. - If accepted, the file is stored with the suggested name. - Naming suggestion is logged.
Actors	- User
Main Success Scenario	1. User uploads or receives a file. 2. System analyzes file metadata and content. 3. System generates a suggested name. 4. User accepts or edits the suggestion. 5. File is stored with the final name. 6. Action is logged.
Alternative Process	- If suggestion is not available, user provides a manual name. - If user rejects the suggestion, their custom name is used.

Table 48 - Use Case Specification 'Suggest appropriate names for files when uploading or receiving them'

2.9.4 Provide a recommendation system to suggest related files that the user may need:

Field	Description
Name	File Recommendation System
Short Description	Provides recommendations for related or useful files based on user activity and file history.
Precondition	- User has previously uploaded, received, or accessed files. - Recommendation engine is active.
Postcondition	- Related file suggestions are displayed to the user. - Recommendation actions are logged.
Actors	- User
Main Success Scenario	1. User accesses the system. 2. System analyzes user activity and history. 3. System generates a list of related files. 4. Recommendations are shown to the user. 5. Action is logged.
Alternative Process	- If no recommendations are available, system displays a message: “No related files found.” - If recommendation engine fails, system skips suggestions without affecting core functionality.

Table 49 - Use Case Specification 'Provide a recommendation system to suggest related files that the user may need'

3- Activity Diagram:

3.1 Login:

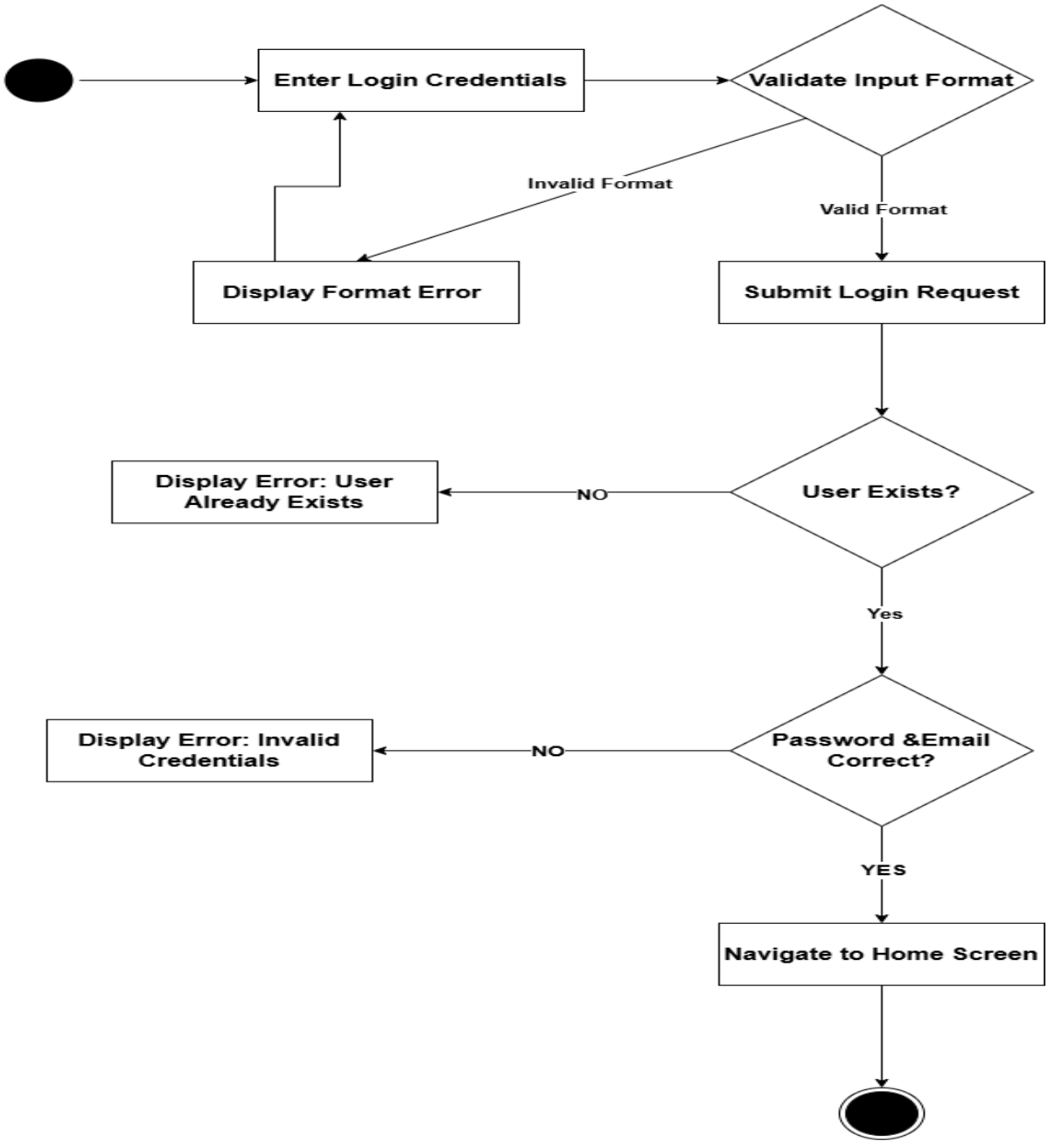


Figure 4 - Activity Diagram 'Login'

3.2 Create New Account:

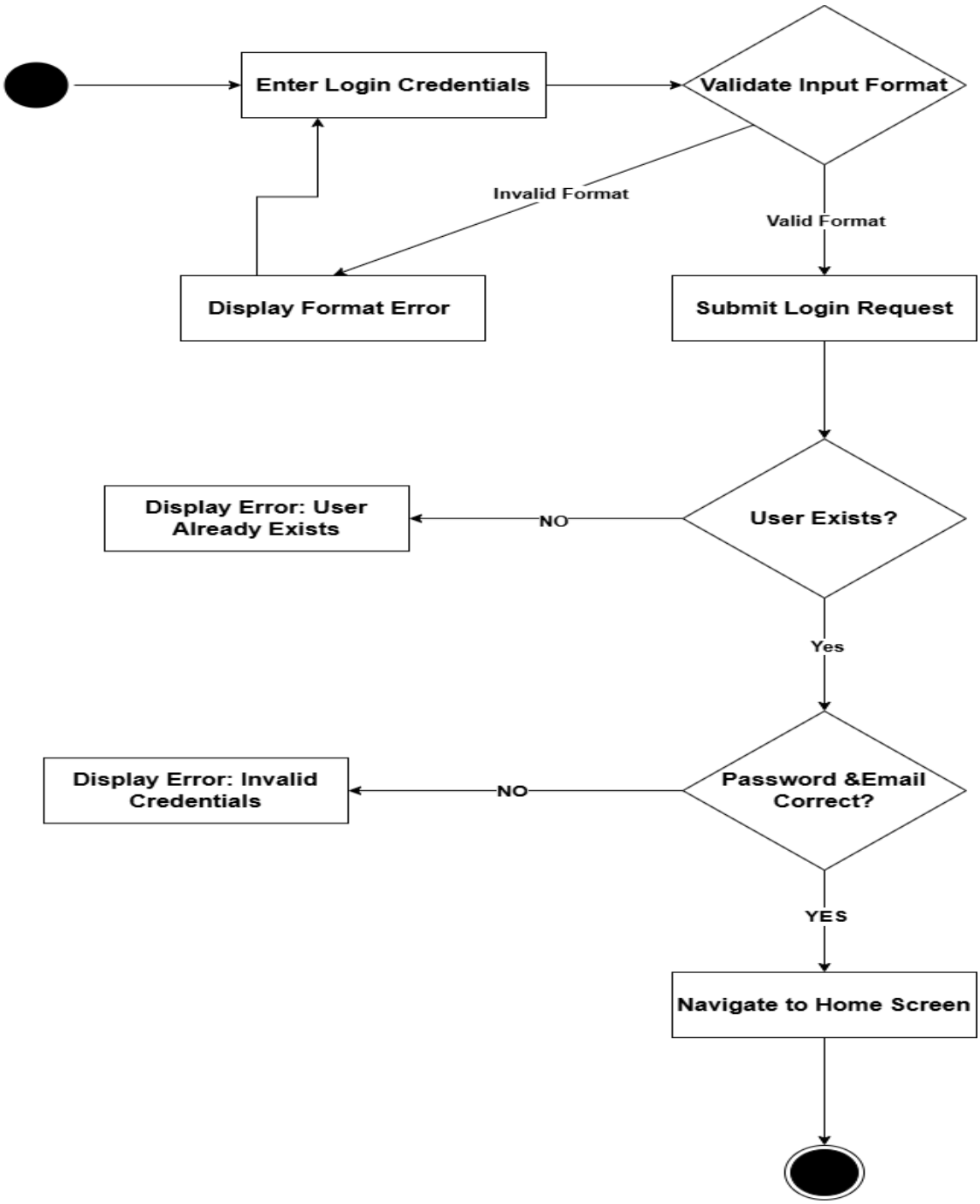


Figure 5 - Activity Diagram 'Create New Account'

3.3 Update Personal Information:

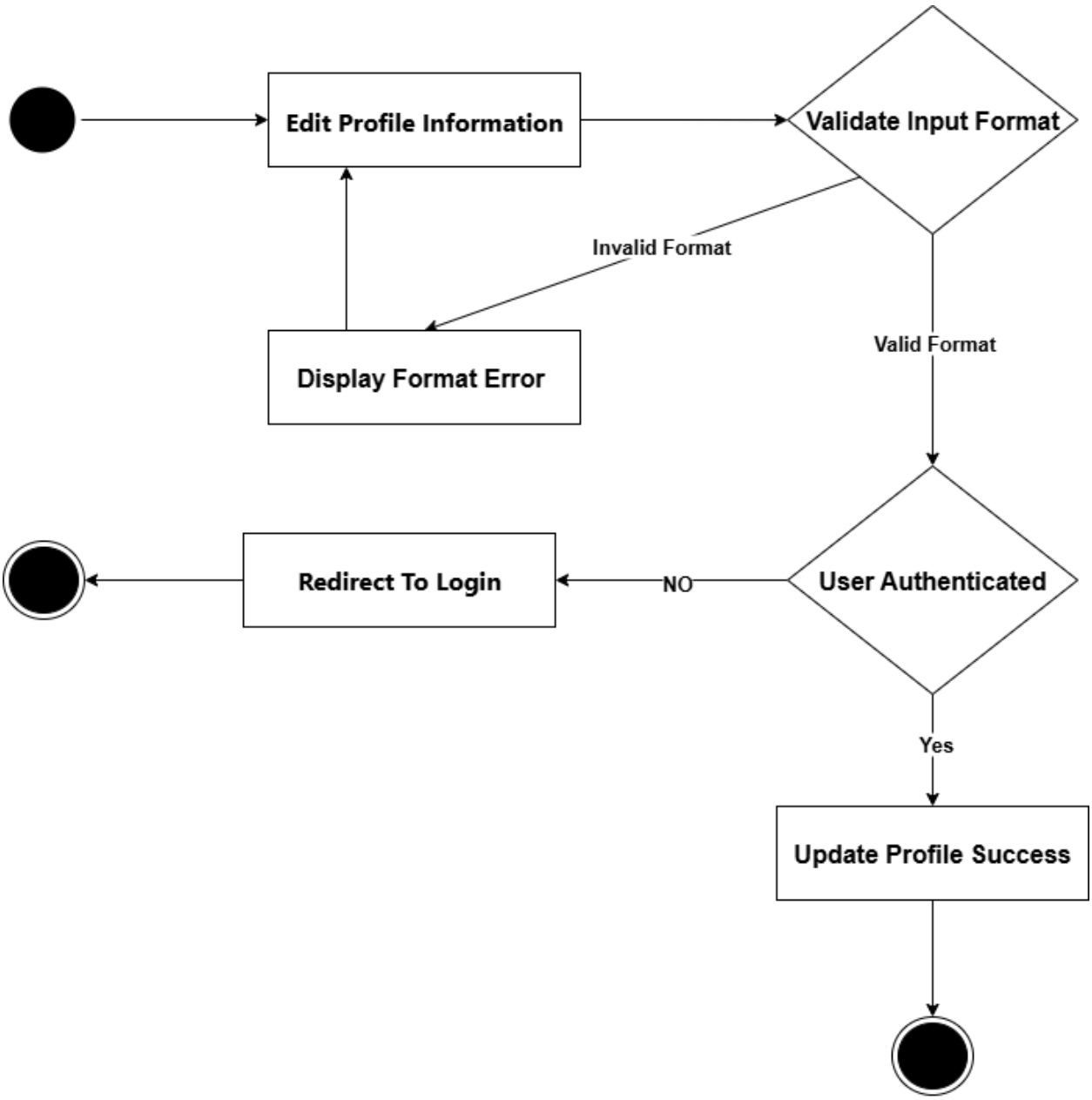


Figure 6 - Activity Diagram 'Update Personal Information'

3.4 Generate Activity Reports:

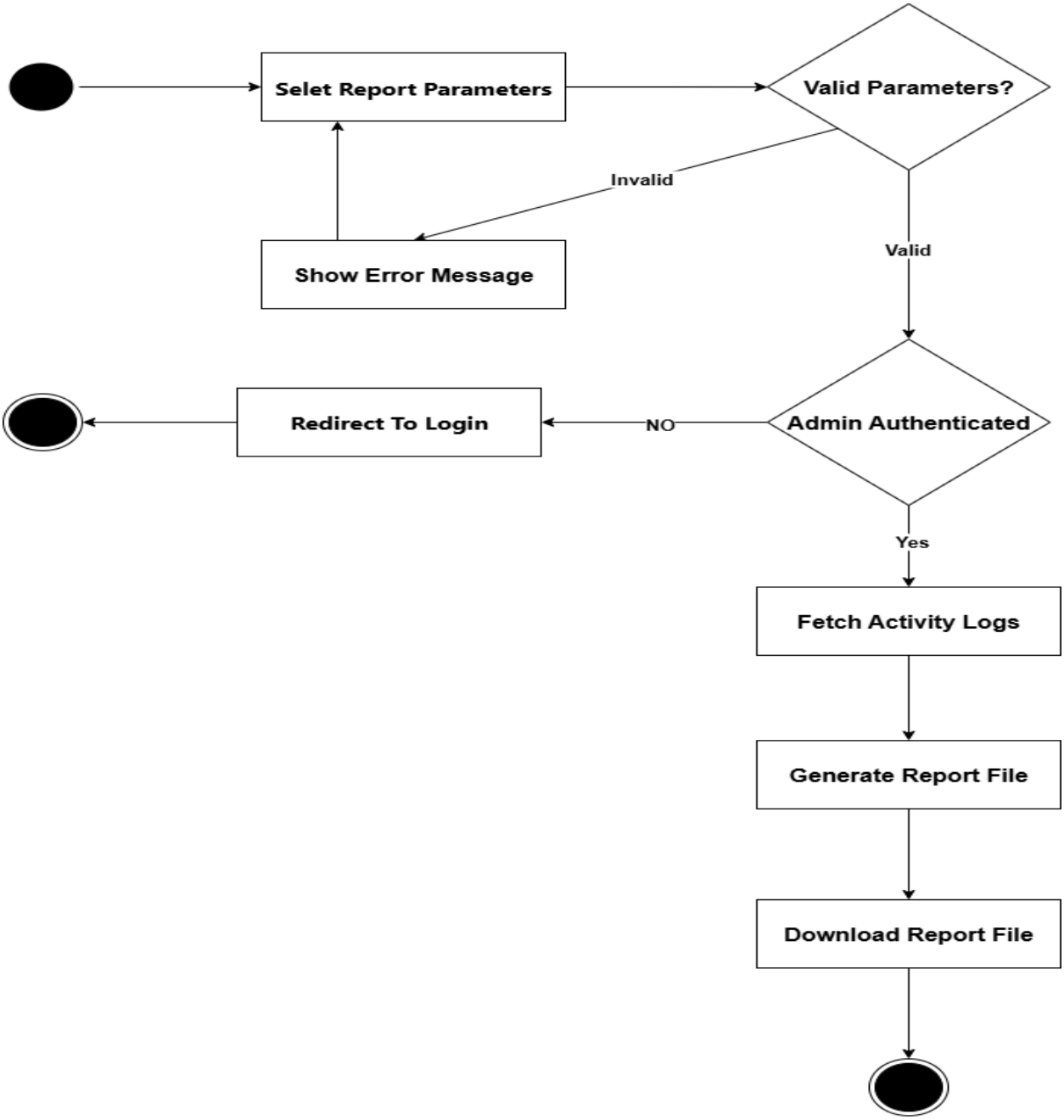


Figure 7 - Activity Diagram 'Generate Activity Reports'

3.5 File Details Displayed:

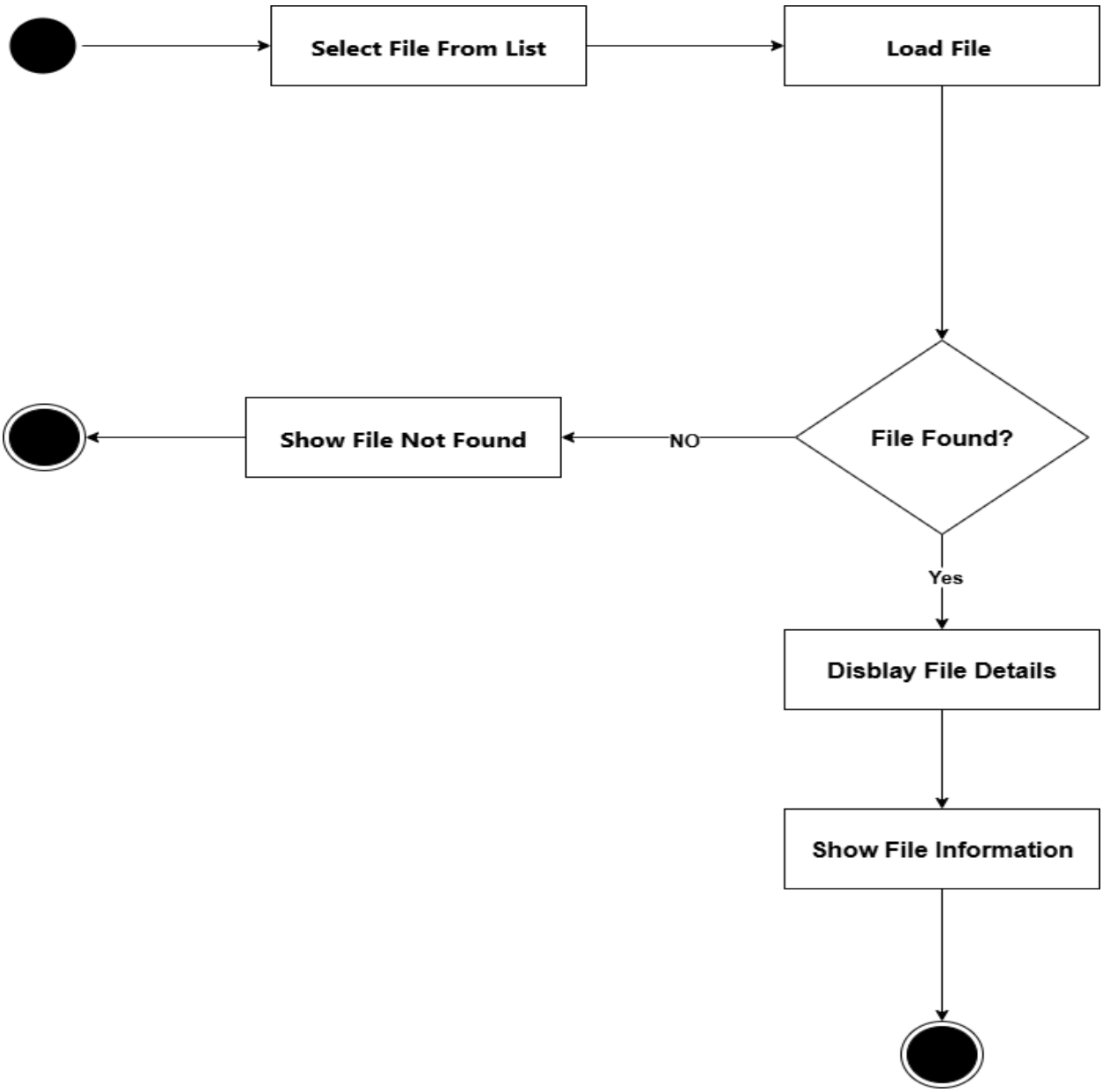


Figure 8 - Activity Diagram 'File Details Displayed'

3.6 Display Available Storage Space:

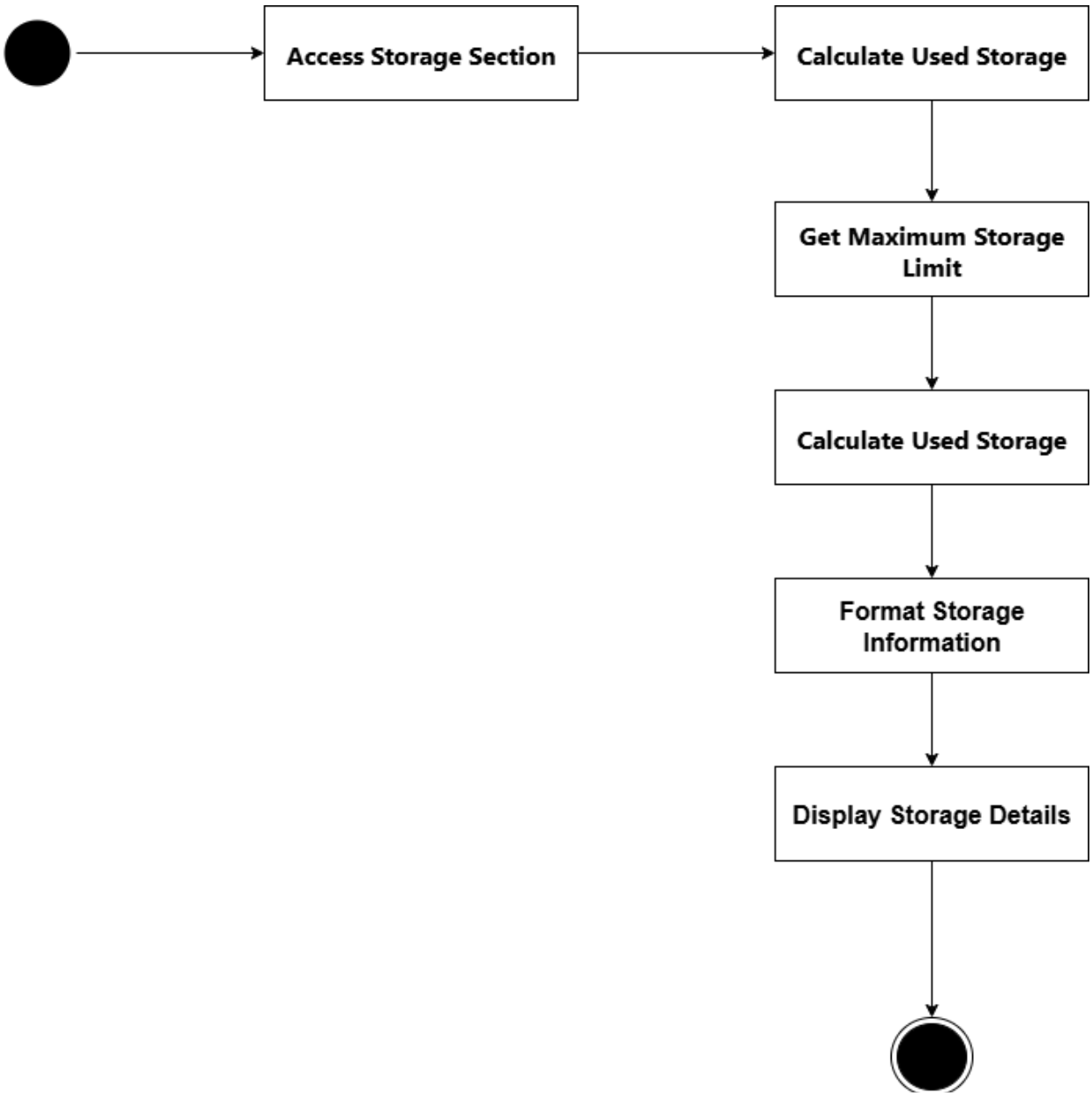


Figure 9 - Activity Diagram 'Display Available Storage Space'

3.7 Send Files Via WIFI Direct:

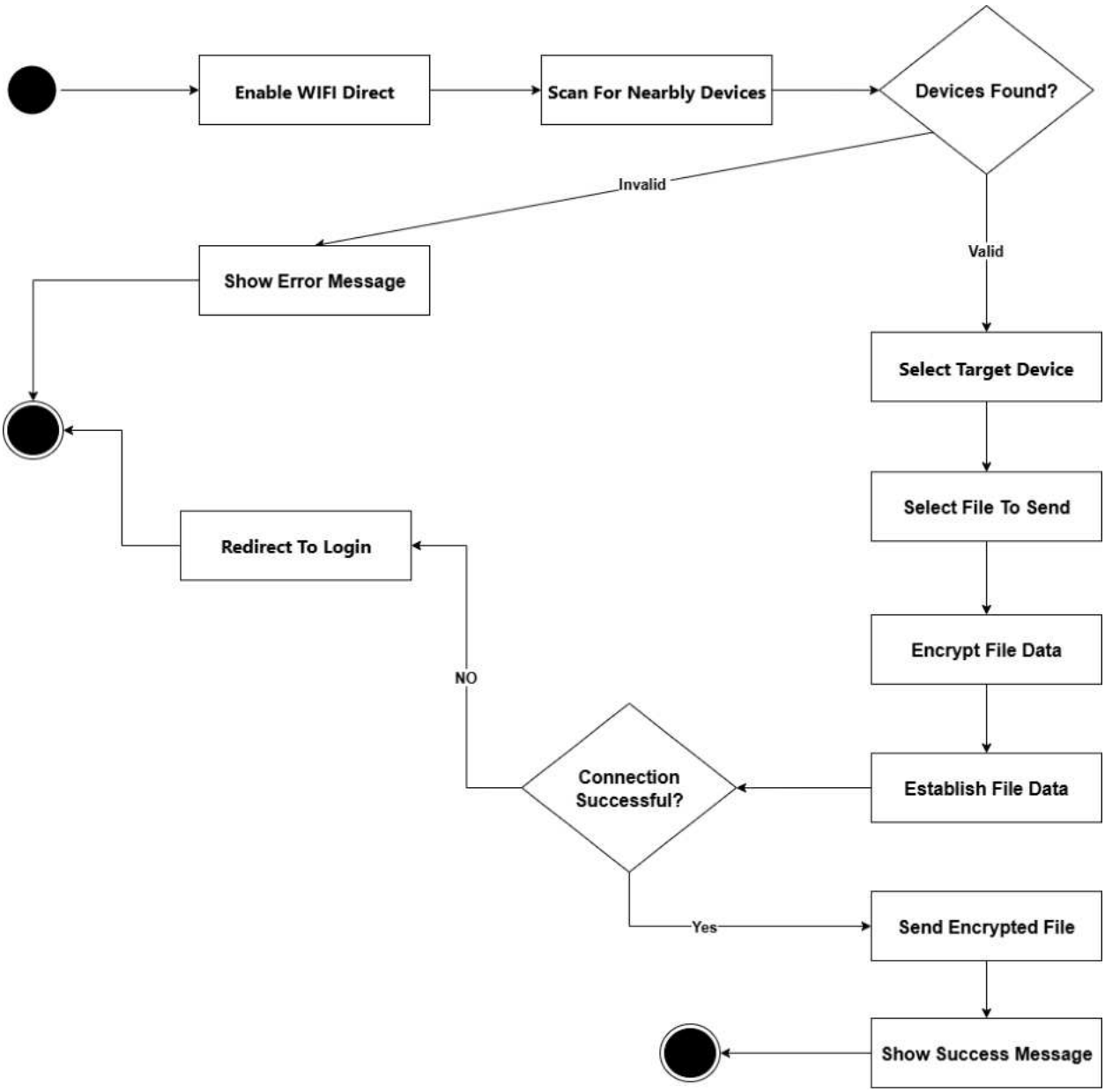


Figure 10 - Activity Diagram 'Send Files Via WIFI Direct'

3.8 Receive Files:

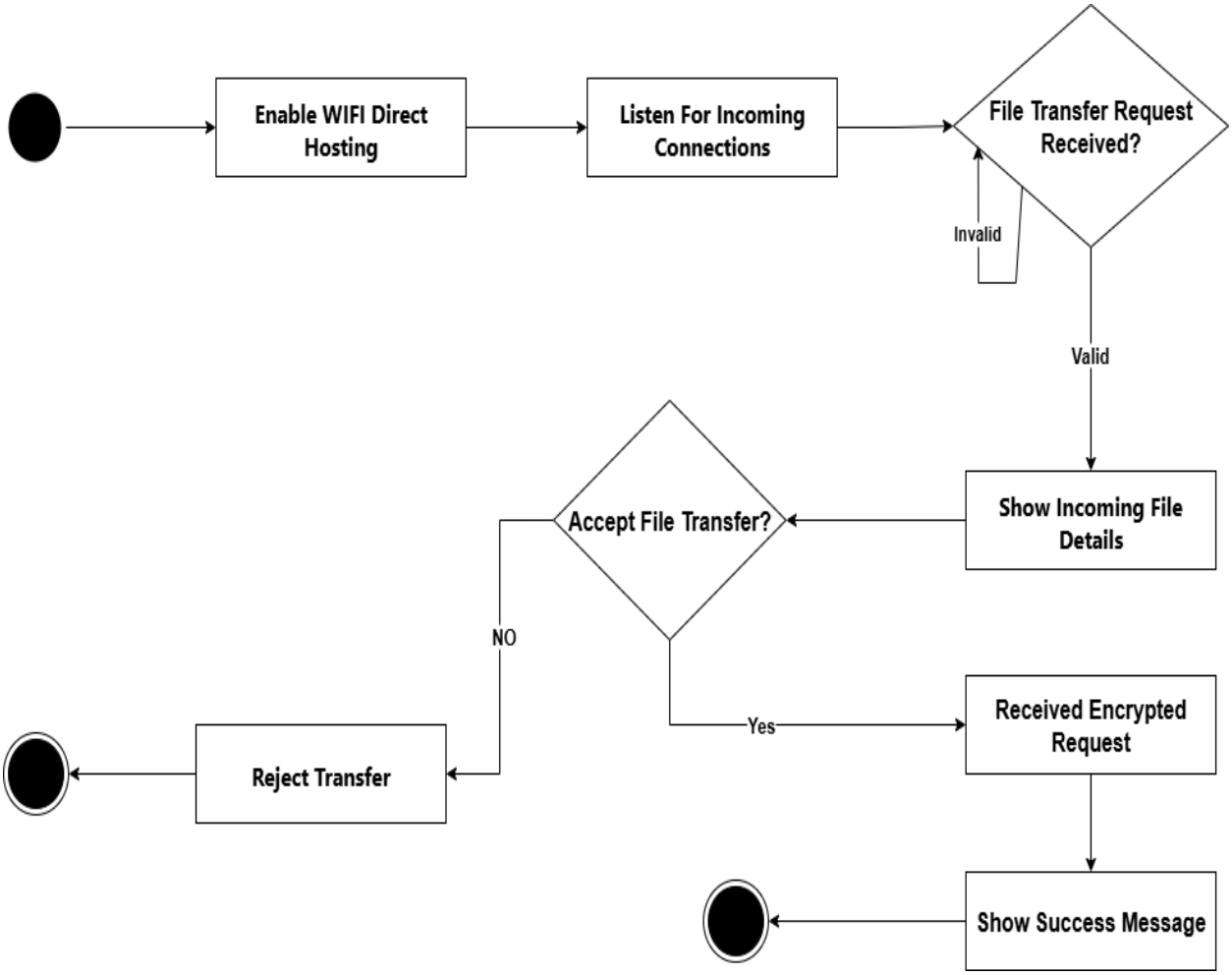


Figure 11 - Activity Diagram 'Receive Files'

3.9 Search Files By Name:

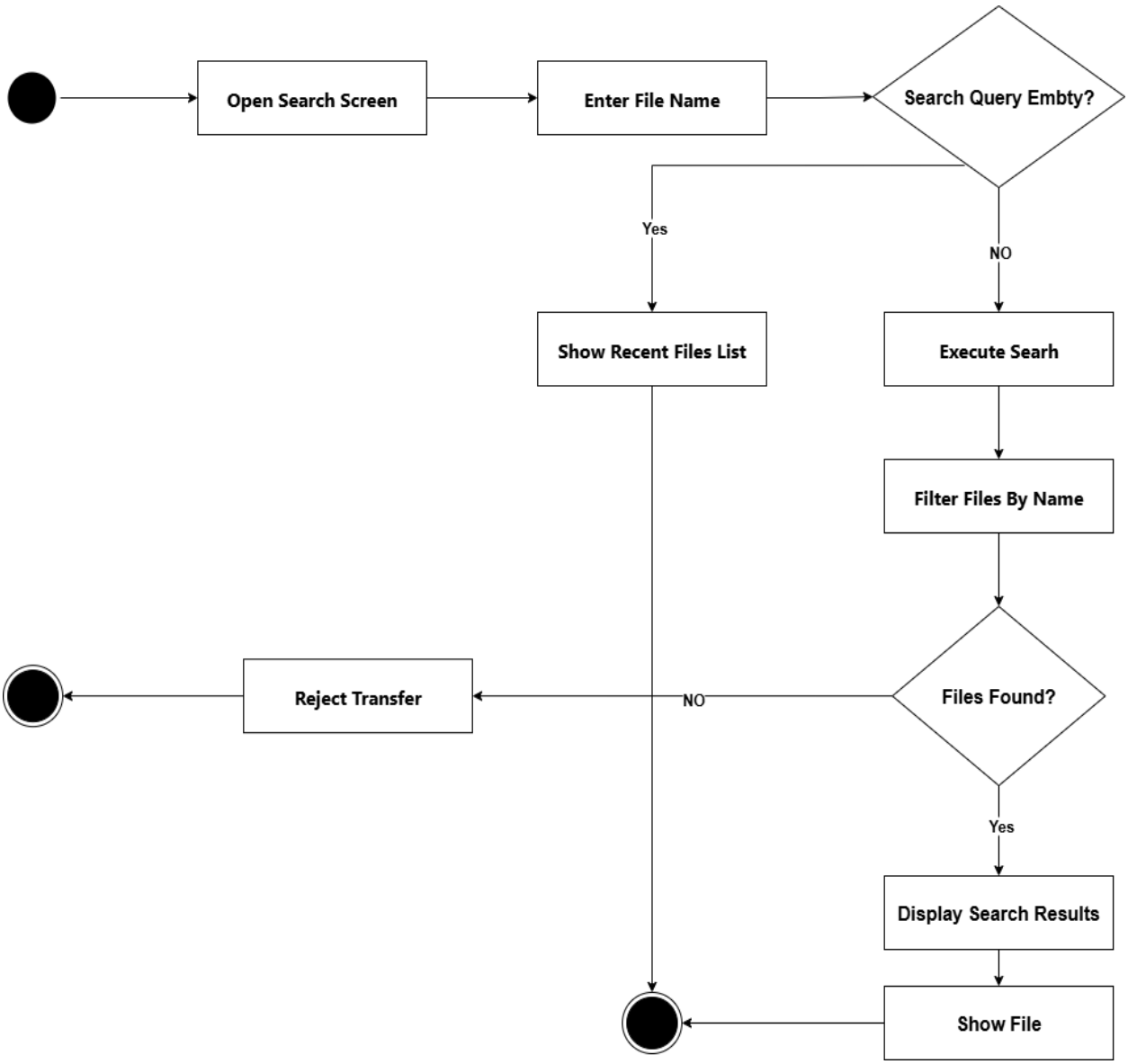


Figure 12 - Activity Diagram 'Search Files By Name'

3.10 Notify When File Deleted:

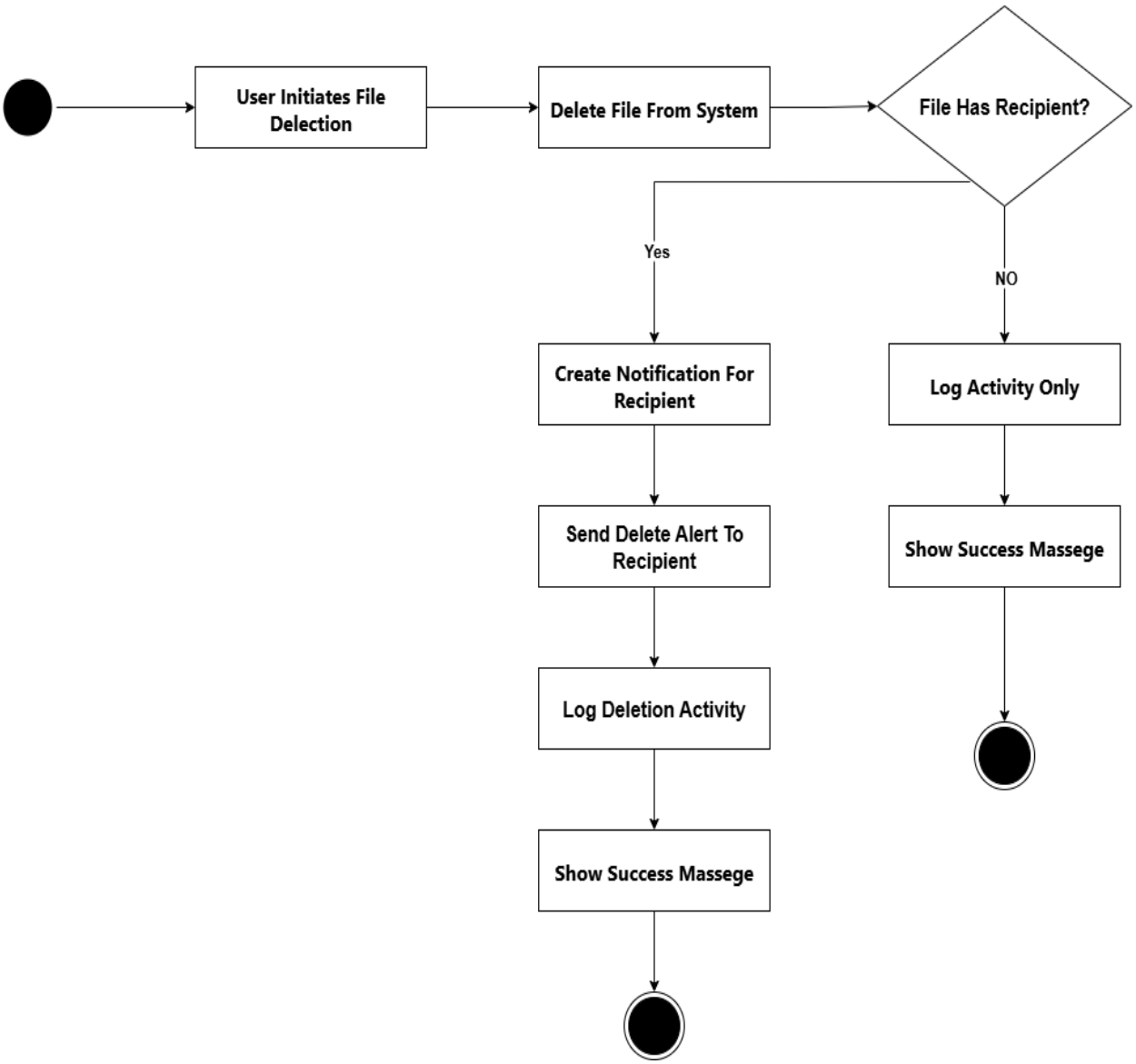


Figure 13 - Activity Diagram 'Notify When File Deleted'

4.List of Preliminary Tests:

4.1 Testing the system's ability to register a new user with valid input:

Test Type	Black Box Test
Test Case Number	ACC-002-TEST-002
Test Case Name	Testing the system's ability to register a new user
Test Description	The user fills the registration form with valid name, email, and password
Operation	User fills the form → clicks “Sign Up”
Expected Result	The user account is created successfully and redirected to homepage

Table 50 - Test 1

4.2 Testing the system's ability to log in with correct credentials:

Test Type	Black Box Test
Test Case Number	AUTH-003-TEST-003
Test Case Name	Testing login functionality with valid credentials
Test Description	User enters correct email and password and attempts to log in
Operation	Enter credentials → click “Login”
Expected Result	User is redirected to their dashboard or homepage

Table 51 - Test 2

4.3 Testing the system's ability to reset password using registered email:

Test Type	Black Box Test
Test Case Number	AUTH-004-TEST-004
Test Case Name	Testing password reset functionality
Test Description	User enters their email to request a password reset
Operation	Enter email → click “Forgot Password”
Expected Result	A reset link is sent to the user’s email

Table 52 - Test 3

4.4 Testing the system's ability to encrypt files before upload:

Test Type	Black Box Test
Test Case Number	SEC-001-TEST-001
Test Case Name	Testing the system’s ability to encrypt files before upload
Test Description	The user selects a file and uploads it; the system must encrypt it using AES or RSA before it reaches the server
Operation	User selects file → clicks “Upload”
Expected Result	The server receives only the encrypted version of the file

Table 53 - Test 4

4.5 Testing the system's ability to auto-delete files after time expiry:

Test Type	Black Box Test
Test Case Number	FILE-005-TEST-005
Test Case Name	Testing self-destruct feature for uploaded files
Test Description	The file should be deleted automatically after 1 minute of upload
Operation	Upload file → enable self-destruct with 1-minute expiry
Expected Result	File is no longer accessible after 1 minute

Table 54 - Test 5

4.6 Testing the system's ability to generate and use encrypted QR codes for file sharing:

Test Type	Black Box Test
Test Case Number	SHARE-006-TEST-006
Test Case Name	Testing QR Code generation and sharing
Test Description	The system should generate an encrypted QR code for the file
Operation	Upload file → click "Generate QR Code"
Expected Result	A secure QR code appears and opens the encrypted file link

Table 55 - Test 6

4.7 Testing the system's ability to reset the user’s password:

Test Type	Black Box Tests
Test Case Number	ACC-001-TEST-005
Test Case Name	Testing the system's ability to reset the user’s password
Test Case Description	The user selects "Forgot Password?", enters their email, receives a verification code, and sets a new password
Operation	Enter email → Receive verification code → Enter code → Set new password
Expected Result	Password is successfully reset, and the user is redirected to the login page

Table 56 - Test 7

4.8 Testing the system's ability to generate activity and log reports:

Test Type	Black Box Tests
Test Case Number	ADM-002-TEST-006
Test Case Name	Testing the system's ability to generate activity and log reports
Test Case Description	The admin selects a date range to generate a report containing user activities and file operations
Operation	Select date range → Click "Generate Report"
Expected Result	A report is displayed with correct data and available for export

Table 57 - Test 8

4.9 Testing the system's ability to define file access permissions for users:

Test Type	Black Box Tests
Test Case Number	ADM-003-TEST-007
Test Case Name	Testing the system's ability to define access permissions for users
Test Case Description	The admin selects a file and defines which users/roles can view, download, or edit it
Operation	Select file → Assign permissions by user/role
Expected Result	Selected users are allowed or denied access based on defined permissions

Table 58- Test 9

4.10 Testing the system's ability to delete user accounts:

Test Type	Black Box Tests
Test Case Number	ADM-004-TEST-008
Test Case Name	Testing the system's ability to delete user accounts
Test Case Description	A user (or admin) initiates the account deletion process and confirms it
Operation	Go to Account Settings → Select "Delete Account" → Confirm deletion
Expected Result	The account is permanently deleted, and the user is logged out

Table 59 - Test 10

5.Requirements Traceability Matrix (RTM):

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-1.1	The system shall allow users to create a new account.	Create Account	Registration Screen	Test-001	Finished
REQ-1.2	The system shall allow users to log into their accounts.	Login	Login Screen	Test-002	Finished
REQ-1.3	The system shall allow users to reset their password.	Password Reset	Reset Page	Test-003	Finished
REQ-1.4	The system shall allow users to update their personal profile information.	Update Profile	Profile Screen		Finished
REQ-2.1	The system shall allow the oversight authority to monitor all activities and logs.	Log Monitoring	Admin Dashboard	Test-004	Finished
REQ-2.2	The system shall allow the authority to generate activity and audit reports.	Report Generation	Reports Section	Test-005	Finished
REQ-2.3	The system shall allow the authority to assign or restrict file access permissions.	File Access Control	Permissions Manager	Test-006	Finished
REQ-2.4	The system shall allow the authority to delete or deactivate user accounts.	Account Deletion	Admin Controls	Test-007	Finished
REQ-3.1	The system shall allow users to send encrypted files.	File Sending	File Manager	Test-008	In Progress
REQ-3.2	The system shall allow users to send encrypted images.	Image Sending	File Manager	Test-009	In Progress
REQ-3.3	The system shall allow users to send encrypted videos.	Video Sending	File Manager	Test-010	In Progress

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-3.4	The system shall allow users to receive encrypted files.	File Receiving	File Inbox	Test-011	In Progress
REQ-3.5	The system shall allow users to receive encrypted images.	Image Receiving	File Inbox	Test-012	In Progress
REQ-3.6	The system shall allow users to receive encrypted videos.	Video Receiving	File Inbox	Test-013	In Progress
REQ-3.7	The system shall allow renaming files upon receipt.	File Renaming	File Manager		In Progress
REQ-3.8	The system shall allow organizing files into folders.	Folder Management	File Manager		In Progress
REQ-3.9	The system shall allow users to view file details.	View File Details	File Viewer		In Progress
REQ-3.10	The system shall support automatic file deletion after a set period.	Auto Deletion	File Settings	Test-014	In Progress
REQ-3.11	The system shall allow creating new folders.	Create Folder	File Manager		In Progress
REQ-4.1	The system shall use local storage for saving files.	Local Storage	Storage Manager		Finished
REQ-4.2	The system shall display available storage space per user.	View Storage	User Dashboard		Finished
REQ-4.3	The system shall allow administrators to define file storage locations.	Define Storage Location	Admin Panel		Finished
REQ-5.1	The system shall support file transfer via Wi-Fi Direct.	Wi-Fi File Transfer	File Sharing Panel		In Progress

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-5.2	The system shall require recipient confirmation before transfer starts.	Transfer Confirmation	Share Settings		In Progress
REQ-5.3	The system shall log all file sending activities.	Sending Logs	Logs Module		Finished
REQ-5.4	The system shall log all file receiving activities.	Receiving Logs	Logs Module		Finished
REQ-5.5	The system shall support generating QR Code for files.	QR Code Generation	Sharing Options	Test-015	In Progress
REQ-5.6	The system shall support sending files via QR Code.	QR File Sending	QR Code Module	Test-016	In Progress
REQ-5.7	The system shall provide encryption repair functionality (deleting previously stored keys).	Encryption Repair	Security Settings		In Progress
REQ-5.8	The system shall support generating public encryption keys.	Public Key Generation	Key Management	Test-017	In Progress
REQ-6.1	The system shall allow users to search files by name.	File Search	Search Bar		Finished
REQ-6.2	The system shall allow users to search files by type.	Type-Based Search	Filters		Finished
REQ-6.3	The system shall allow users to search files by upload date.	Date-Based Search	Filters		Finished
REQ-6.4	The system shall allow filtering files by name.	Filter by Name	Filter Tools		Finished
REQ-6.5	The system shall allow filtering files by type.	Filter by Type	Filter Tools		Finished

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-6.6	The system shall allow filtering files by upload date.	Filter by Date	Filter Tools		Finished
REQ-7.1	The system shall notify users when a new file is received.	File Notification	Notification System		Finished
REQ-7.2	The system shall notify users when a file is deleted.	Deletion Alert	Notification System		Finished
REQ-7.3	The system shall notify users when a file is modified.	File Update Alert	Notification System		Finished
REQ-7.4	The system shall alert users when storage space is nearing full capacity.	Storage Alert	Notification System		Finished
REQ-8.1	Support text-based interaction with the user.	Text Interaction	Chat Interface	Test-018	In Progress
REQ-8.2	Assist the user in any operation they want.	User Assistance	Help Module	Test-019	In Progress
REQ-8.3	Automatically classify files into folders based on file type.	File Classification	File Manager	Test-020	In Progress
REQ-8.4	Improve file search.	Enhanced Search	Search Module	Test-021	In Progress
REQ-8.5	Suggest appropriate names for files when uploading or receiving them.	File Naming Suggestions	File Manager	Test-022	In Progress
REQ-8.6	Provide a recommendation system to suggest related files that the user may need.	File Recommendation	Recommendation Engine	Test-023	In Progress

Table 60 - Requirements Traceability Matrix (RTM)

Chapter 5: Design Study

1. Sequence Diagram:

1.1 Login:

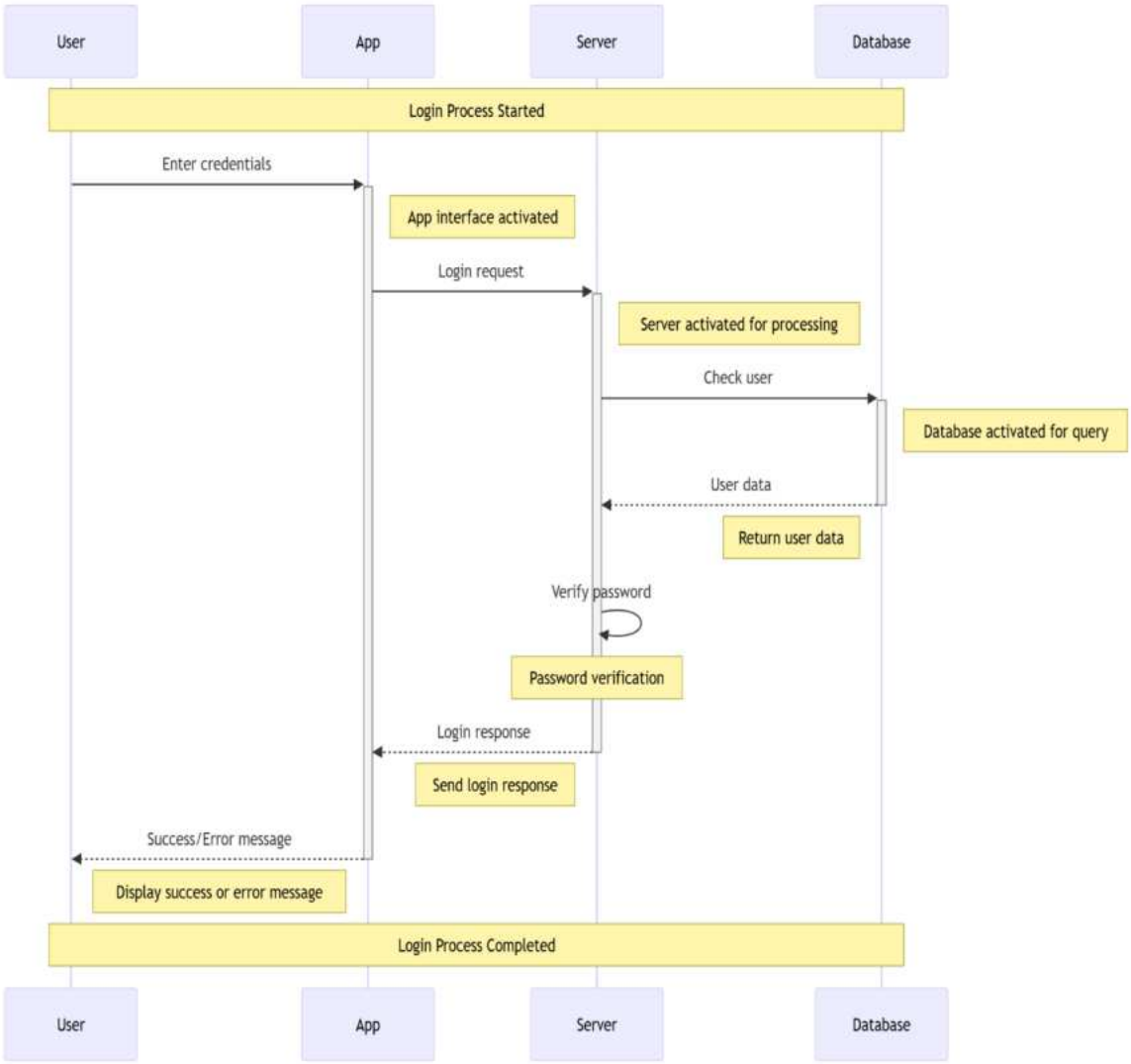


Figure 14 - Sequence Diagram 'Login'

1.2 Create New Account:

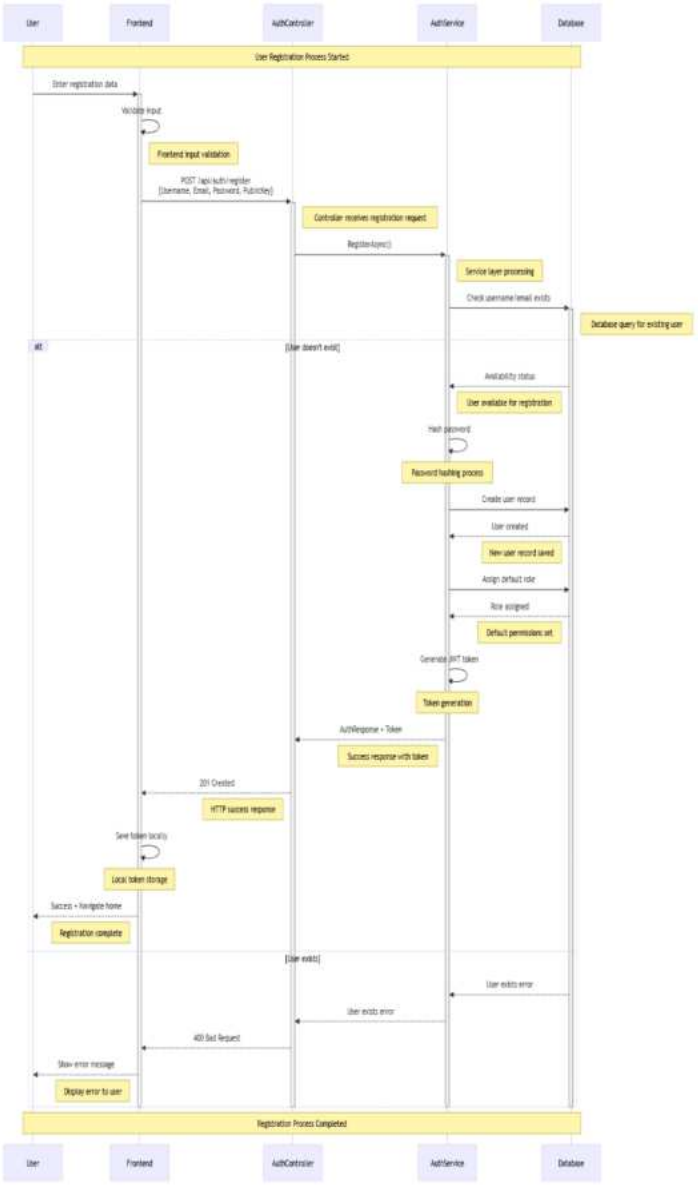


Figure 15 - Sequence Diagram 'Create New Account'

1.3 Update Personal Information:

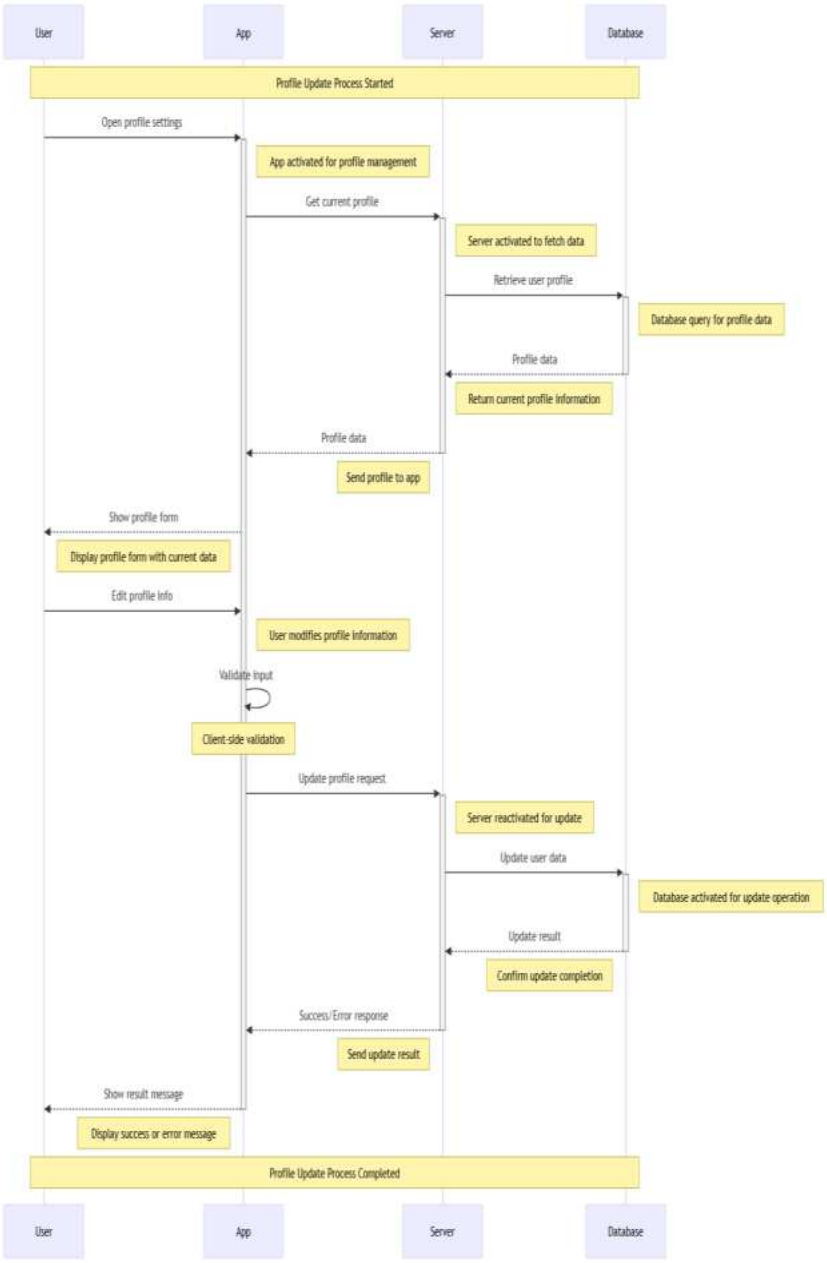


Figure 16 - Sequence Diagram 'Update Personal Information'

1.4 Generate Activity Reports:

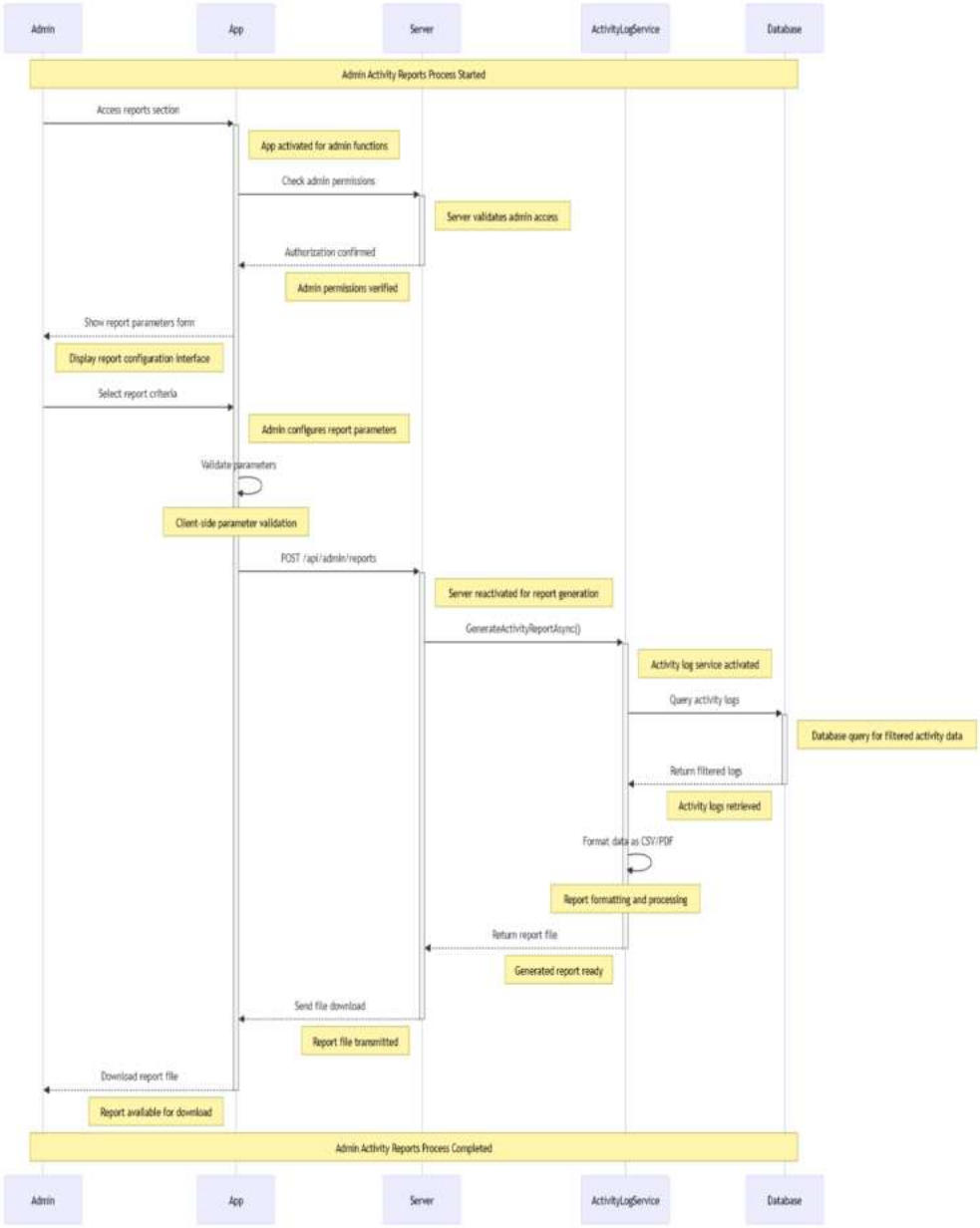


Figure 17 - Sequence Diagram 'Generate Activity Reports'

1.5 File Details Displayed:

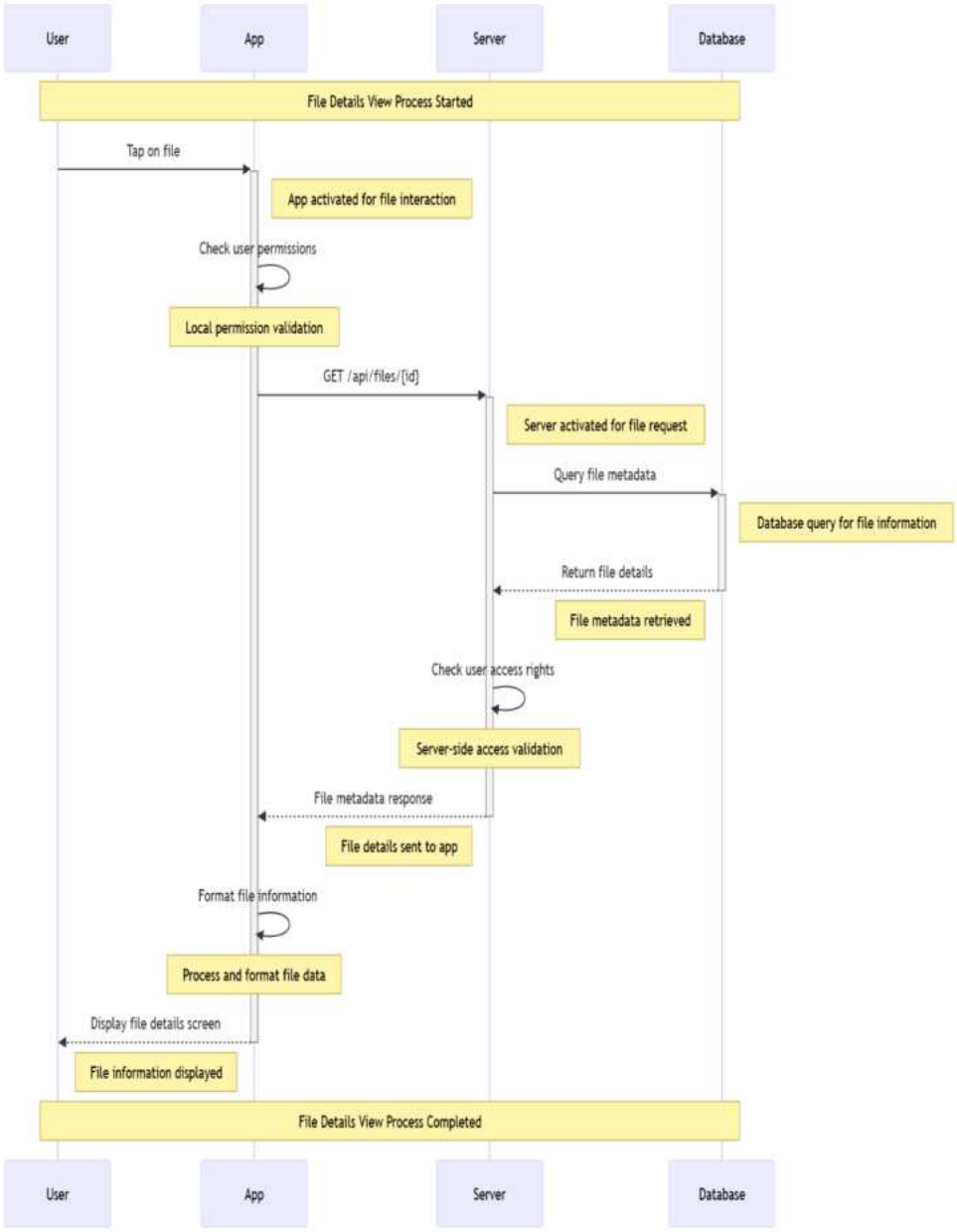


Figure 18 - Sequence Diagram 'File Details Displayed'

1.6 Display Available Storage Space:

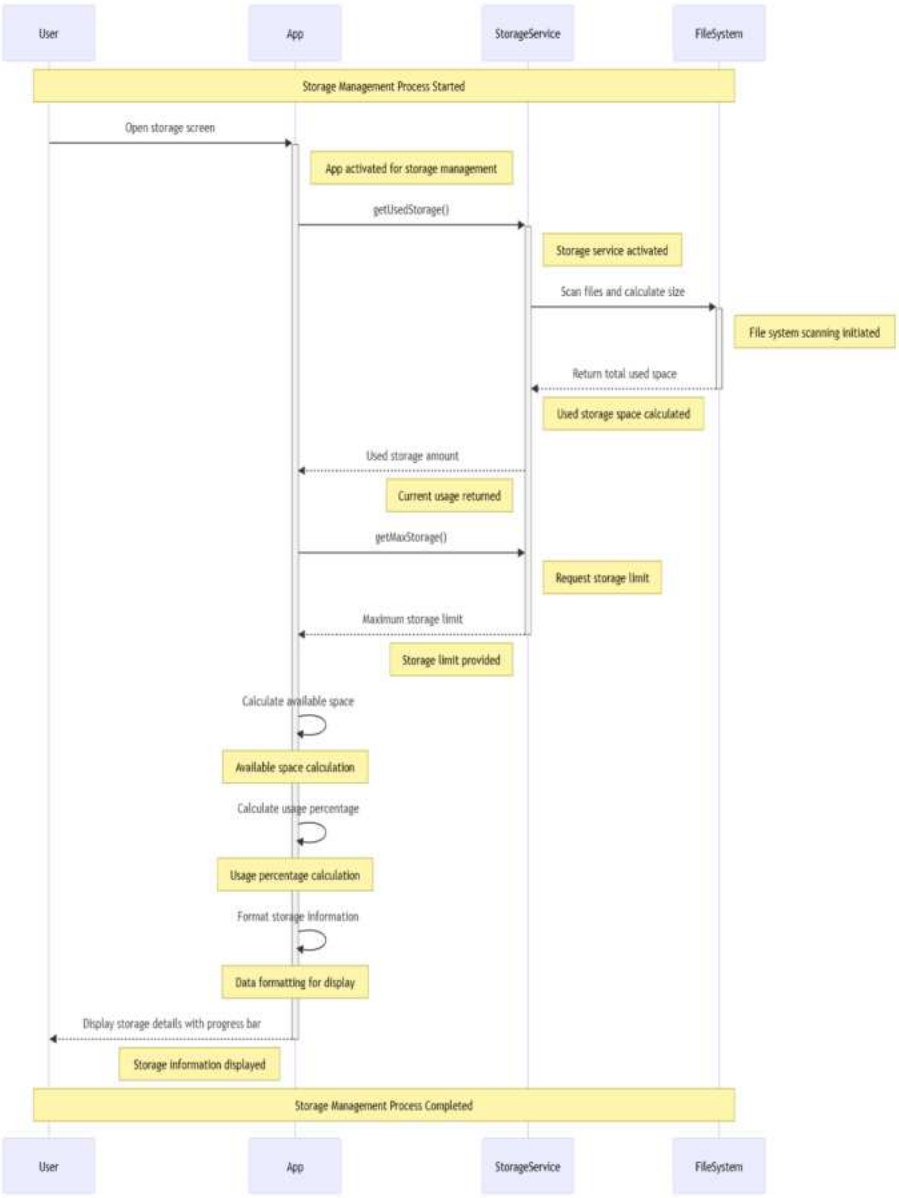


Figure 19 - Sequence Diagram 'Display Available Storage Space'

1.7 Send Files Via WIFI Direct:

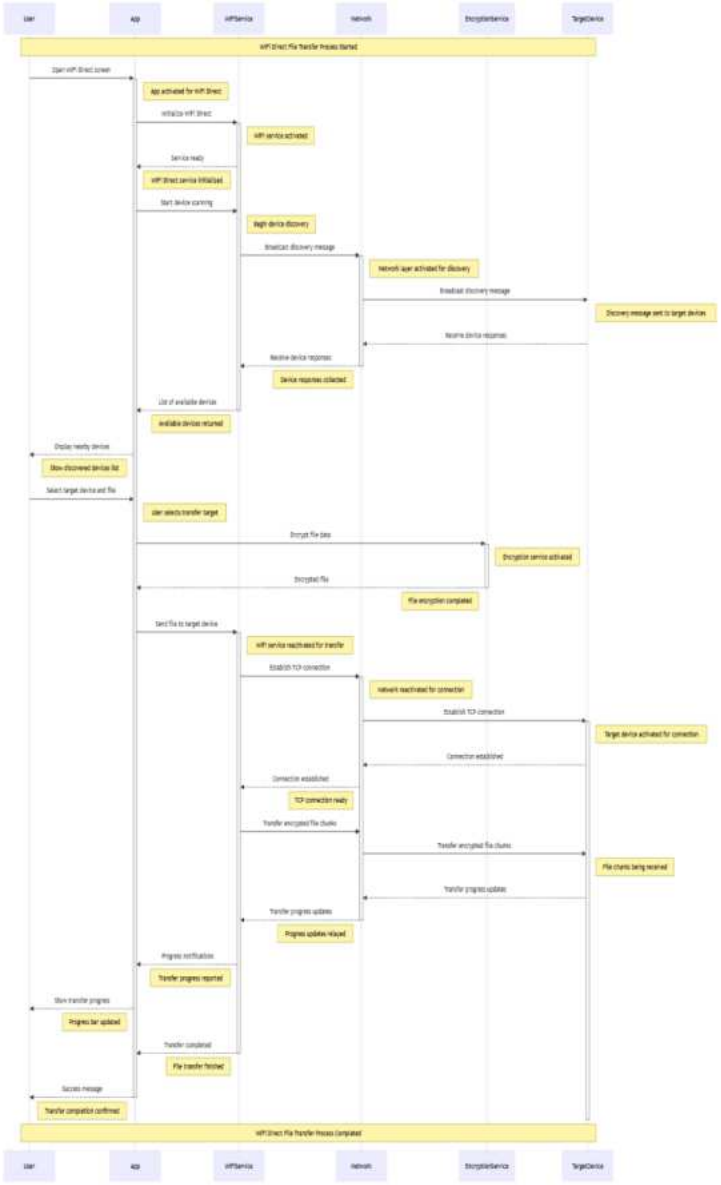


Figure 20 - Sequence Diagram 'Send Files Via WIFI Direct'

1.8 Receive Files:

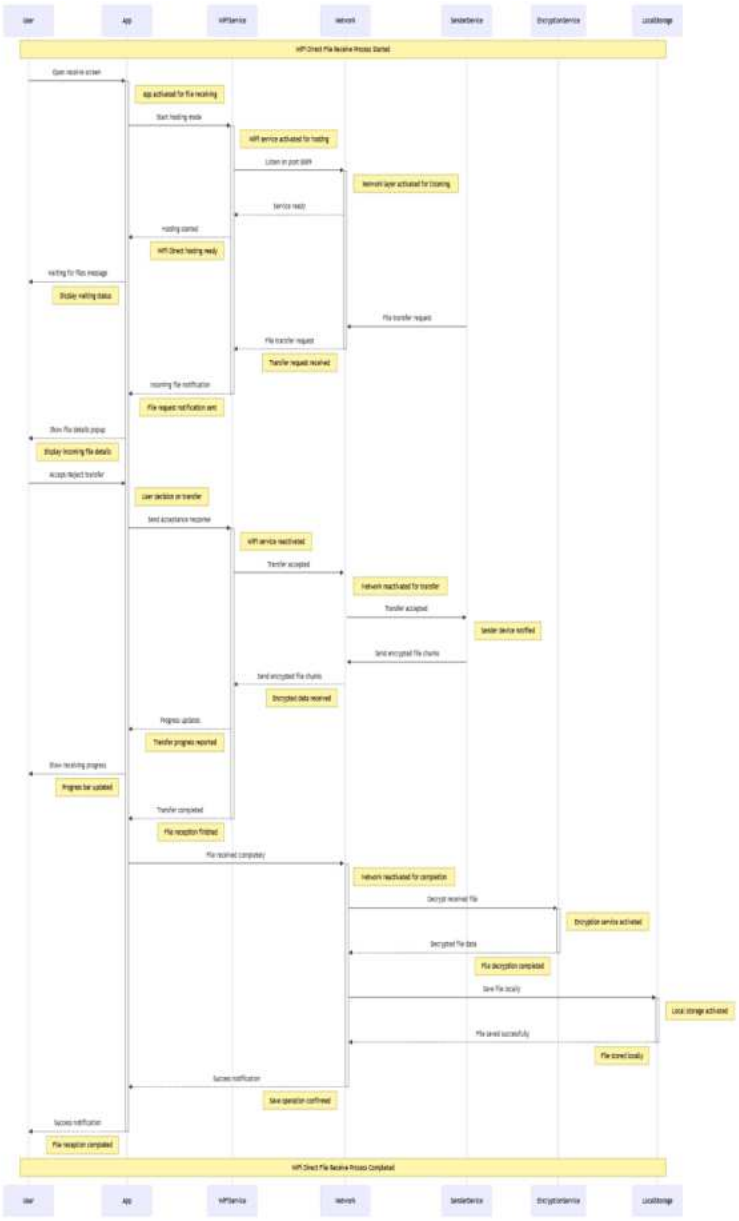


Figure 21 - Sequence Diagram 'Receive Files'

1.9 Search Files By Name:

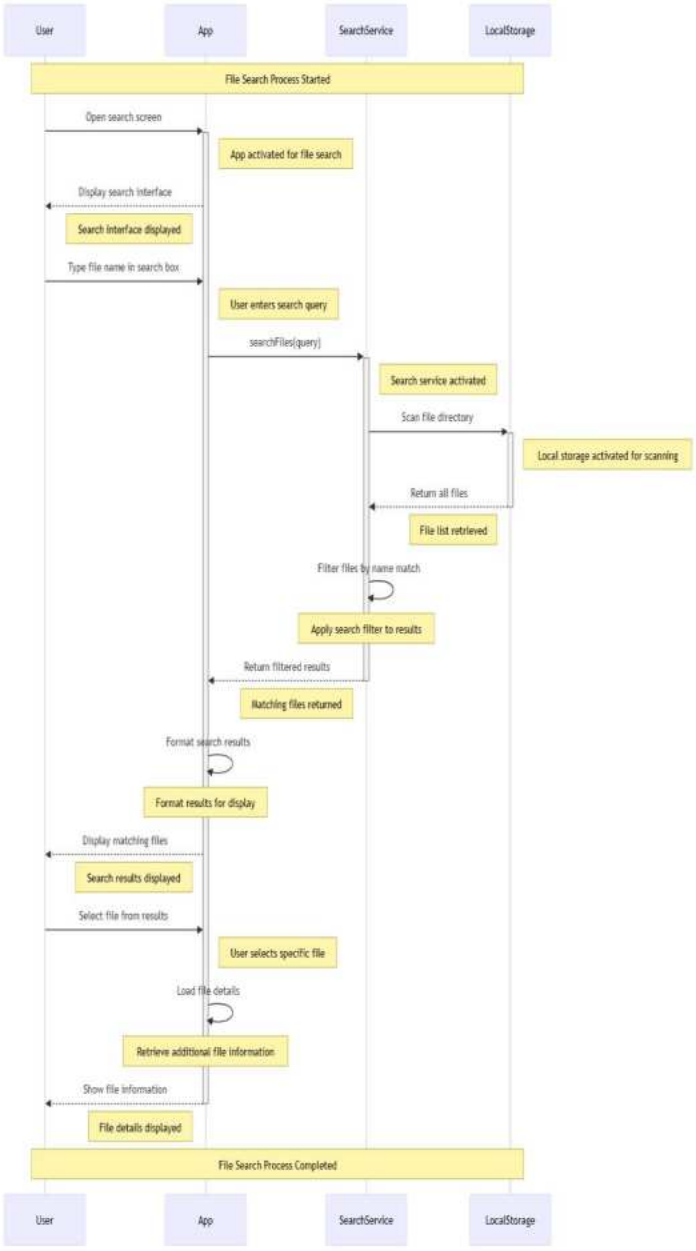


Figure 22 - Sequence Diagram 'Search Files By Name'

1.10 Notify When File Deleted:

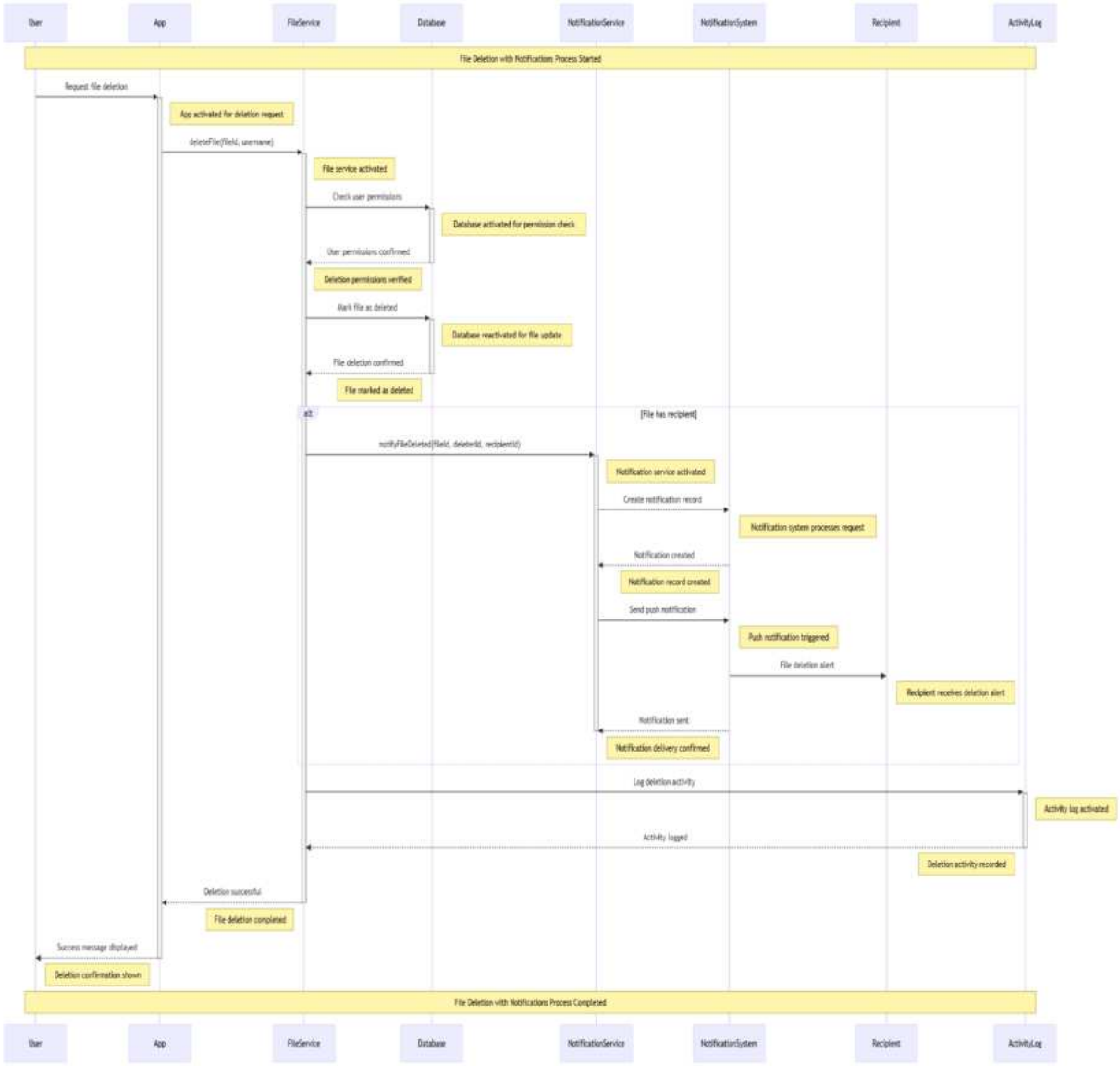


Figure 23 - Sequence Diagram 'Notify When File Deleted'

1.11 Support text-based interaction with the user:

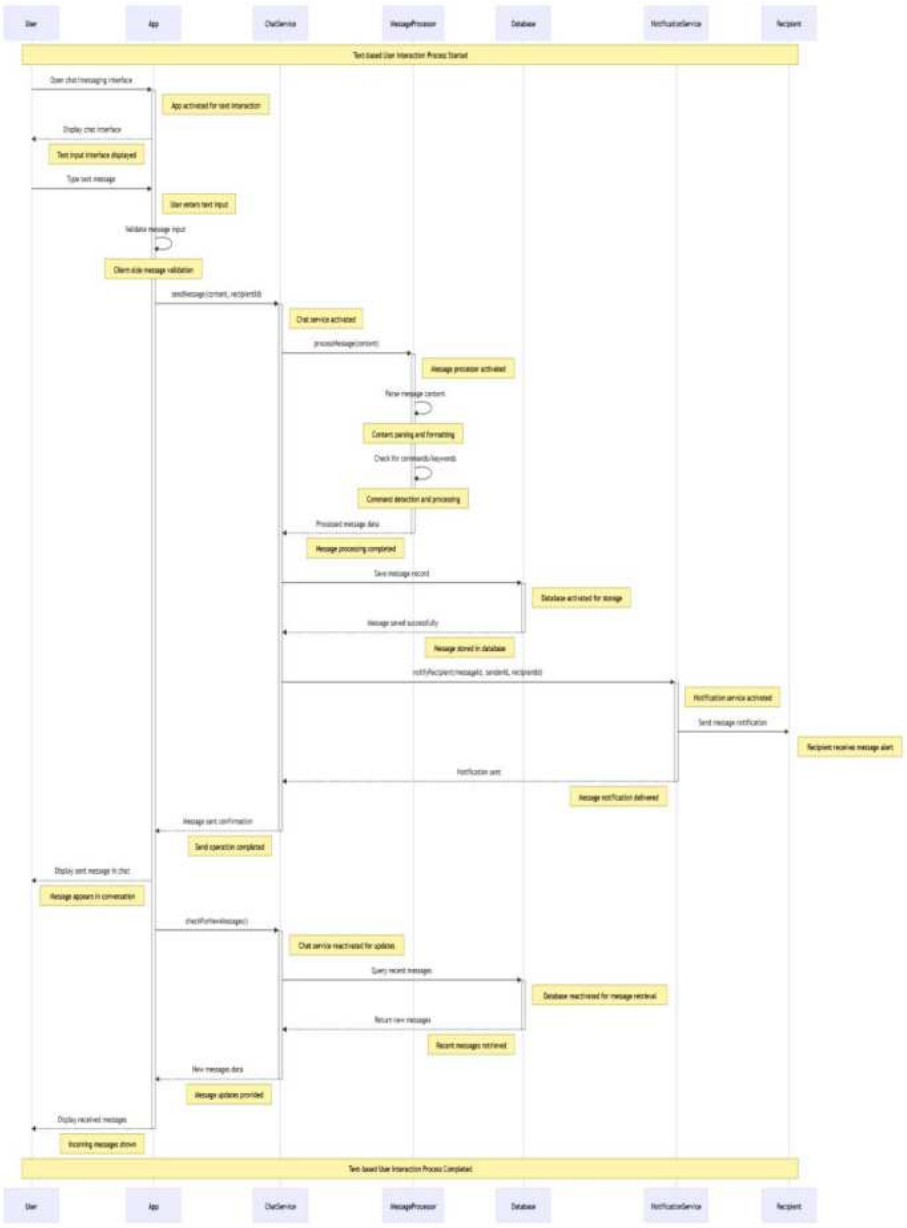


Figure 24 - Sequence Diagram 'Support text-based interaction with the user'

1.12 Automatically classify files into folders based on file type:

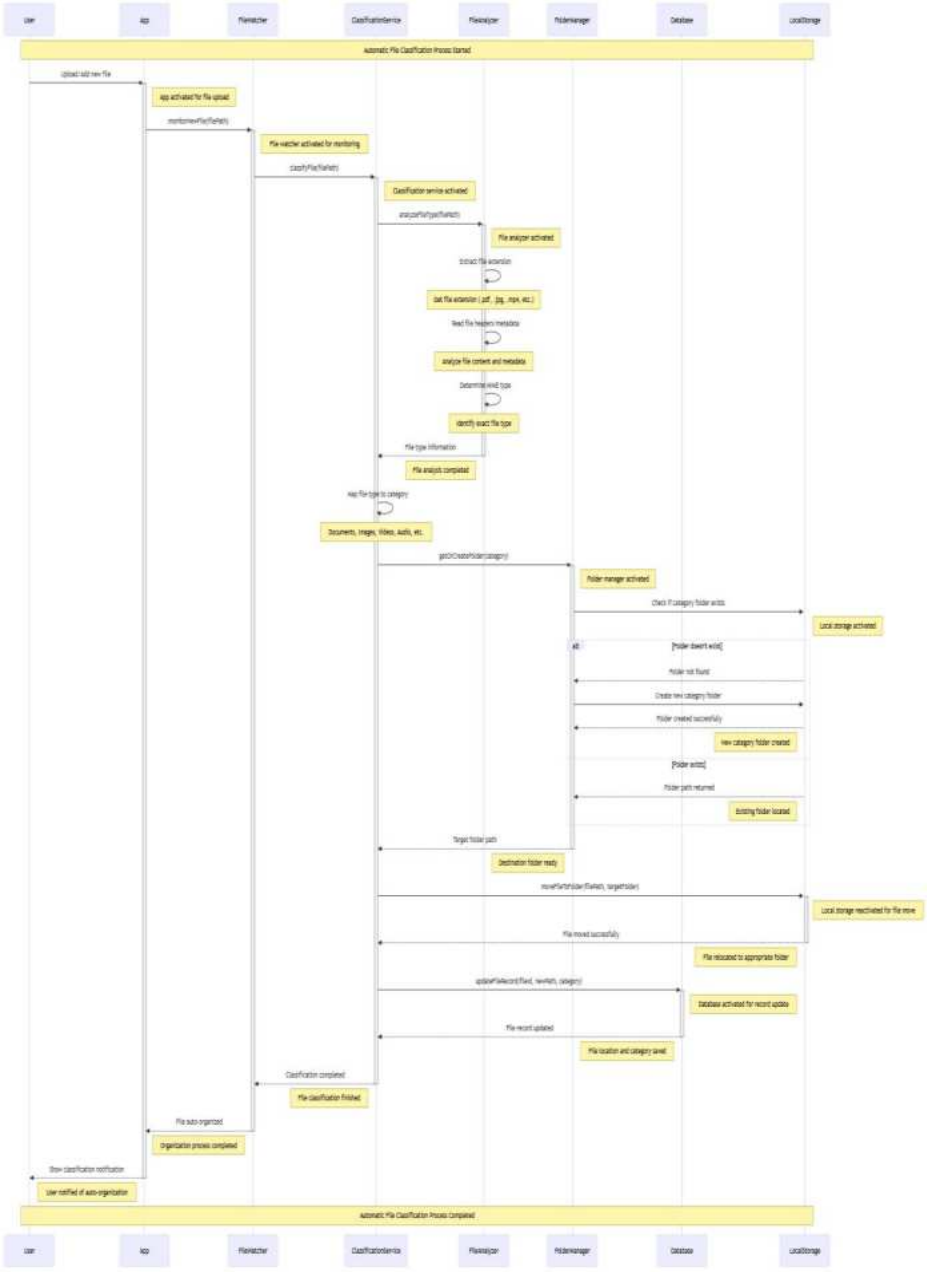


Figure 25 - Sequence Diagram 'Automatically classify files into folders based on file type'

2. ERD Diagram:

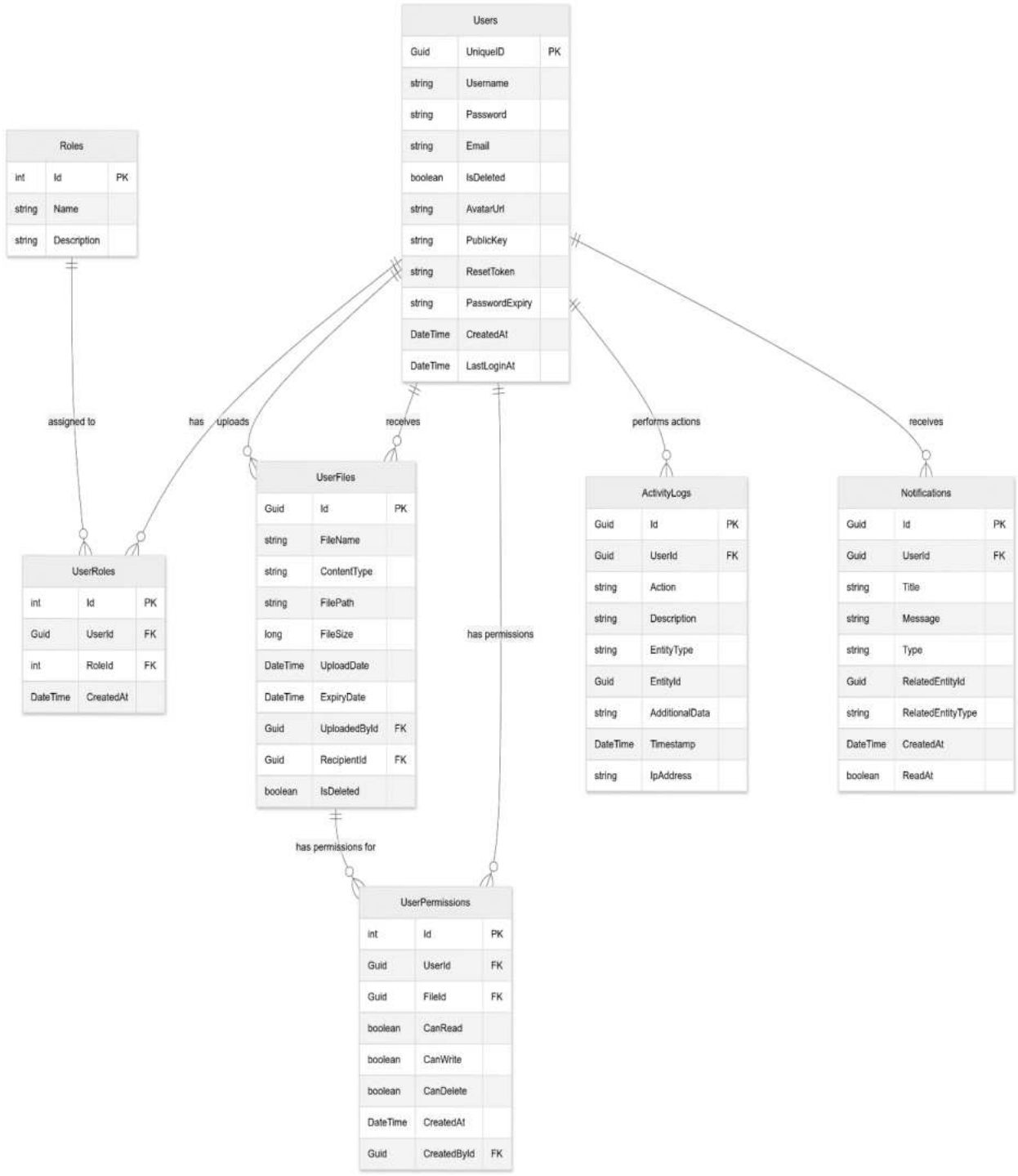


Figure 26 - ERD Diagram

3. Class Diagram:

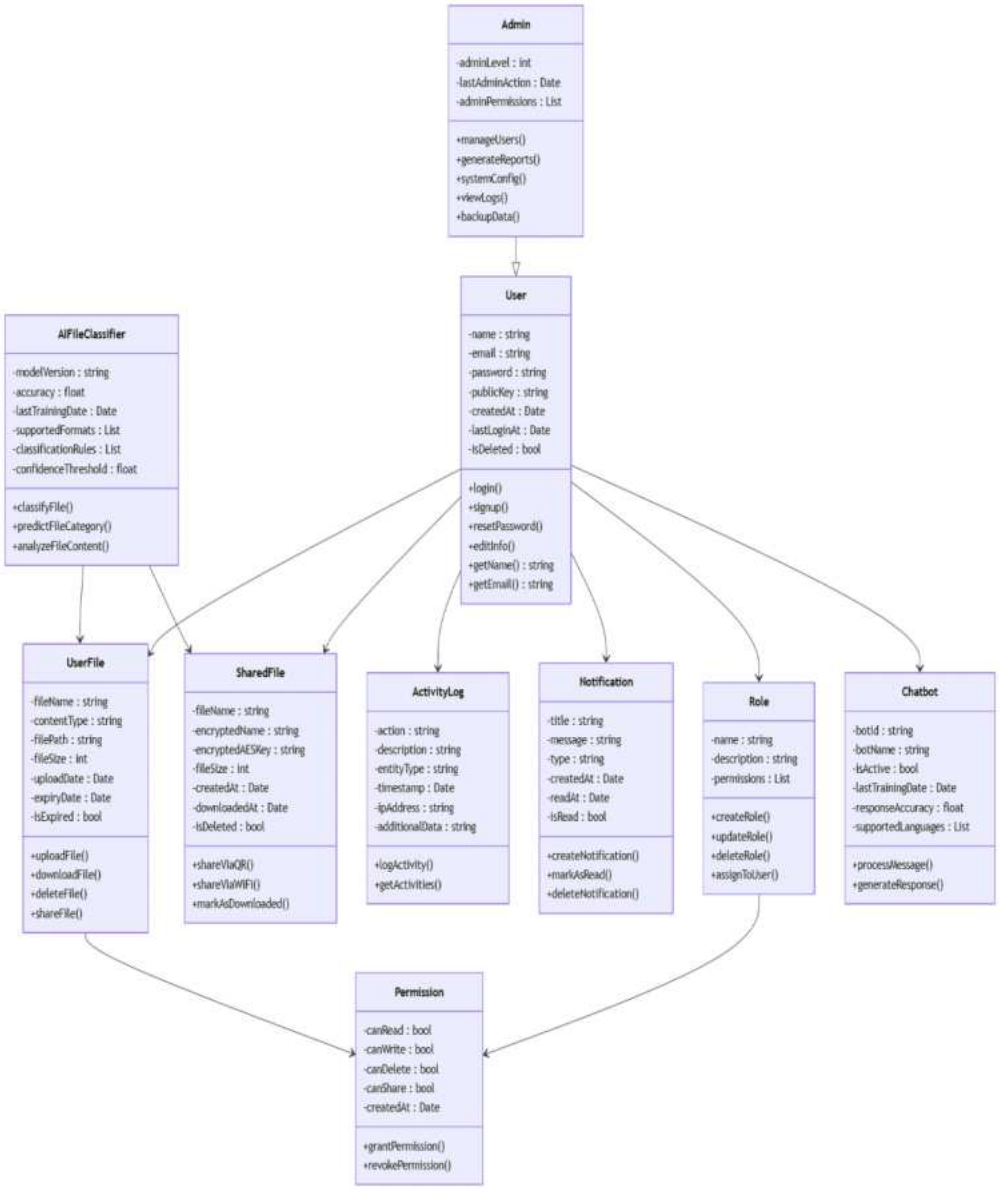


Figure 27 - Class Diagram

4. System Architecture

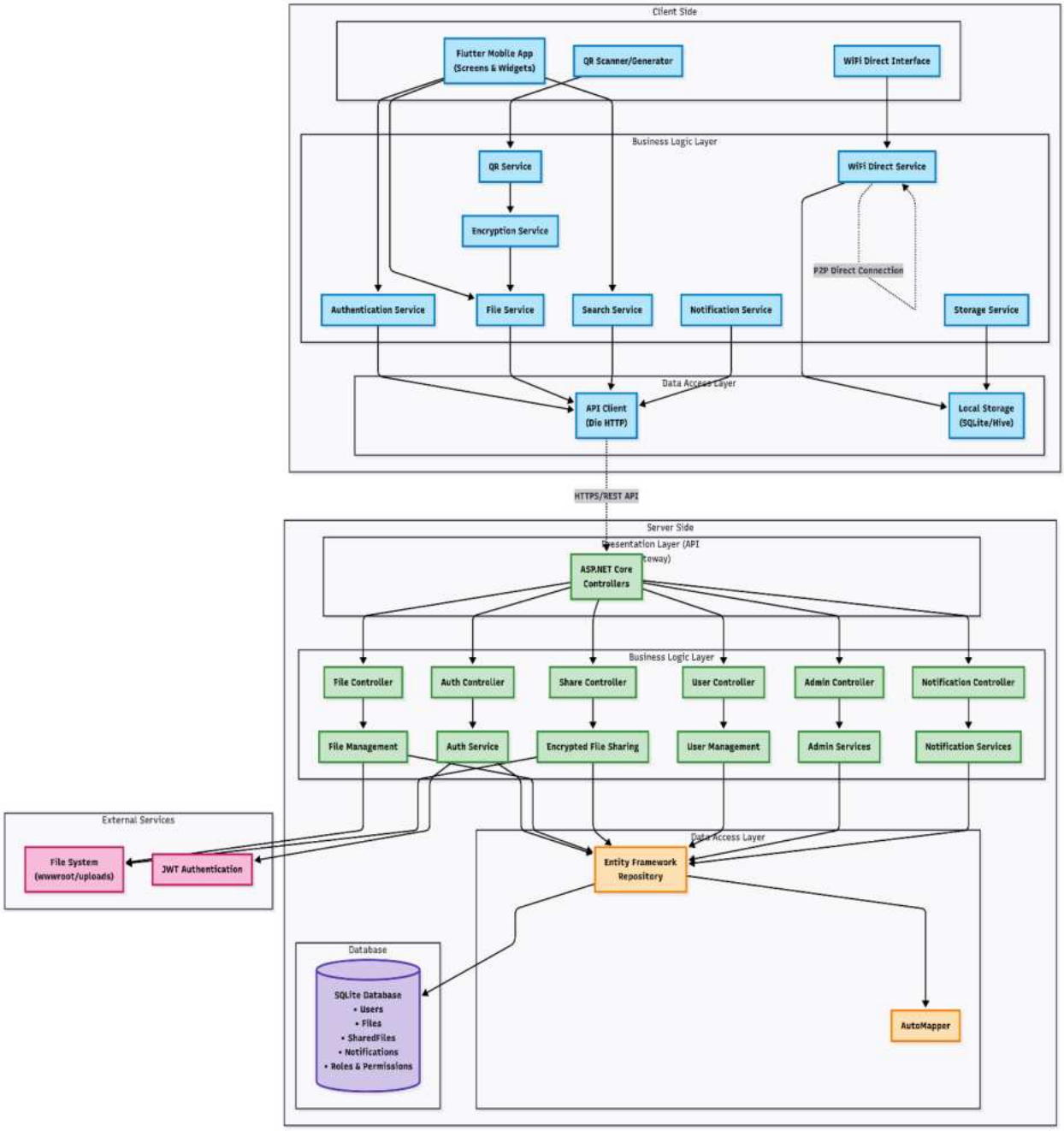


Figure 28 - Client-Server Architecture

4.1 Architectural Overview

1. Presentation Layer (Client Side):

a) Graphical User Interface (GUI):

• Flutter Mobile App with multiple screens

• Enables users to create accounts and log in

• Edit personal information and manage files (upload, download, rename, organize into folders)

• Perform search and filtering through intuitive interfaces

b) Scanning & Generation Components:

• QR Scanner for quick file sharing

• QR Generator for secure sharing

• Wi-Fi Direct Interface for direct connection

c) Notification Components:

• Deliver alerts to users upon receiving new files

• File modification or deletion notifications

• Storage space warnings

2. Business Logic Layer (Client & Server)

a) Core Services:

• Authentication Service:

- Login and logout management

- Password reset functionality

• File Management Service:

- Managing sending and receiving files of all types

- Renaming and folder organization

- Optional auto-deletion of files
- File access permissions management

• Encryption Service:
- Protecting data in transit and at rest
- Encryption repair and public key generation
- End-to-end encryption
- Encryption key management

b) Advanced Communication Services:

• Wi-Fi Direct Service:
- File transfer via direct connection
- Request recipient approval before sending
- P2P connection without internet
- Connection status monitoring

• Search Service:
- Search for users and files
- Advanced filtering of results
- Data indexing for fast search

• QR Service:
- Generate and use QR codes for quick and secure sharing
- Integration with encryption service
- Scan and decrypt QR codes

c) Management & Monitoring Services:

• Storage Service:

- Monitor available storage space
- Manage storage locations and optimize usage
- Display capacity for users
- Clean up expired files

• Notification Service:
- Send real-time notifications
- Manage different notification types
- Customize notification settings

• Activity Logging Service:
- Record all send/receive operations
- Monitor user activity for administrative review
- Track security and unauthorized access

3. Data Access Layer

a) Client Side:

• API Client (Dio HTTP):
- Manage data exchange and request execution
- File uploads, approval requests, and notifications
- Error handling and retry logic
- Token and authentication management

• Local Storage (SQLite/Hive):
- Enable local storage options
- Display capacity for users
- Store temporary data and encryption keys

- Store user settings

b) Server Side:

• Entity Framework Repository:

- Manage database operations
- Implement Repository pattern
- Transaction and consistency management
- Query optimization

• AutoMapper:

- Object mapping between layers
- Convert DTOs to Entities and vice versa

4. Server Side Architecture

a) Presentation Layer (API Gateway):

• ASP.NET Core Controllers:

- Auth Controller: Authentication and authorization management
- File Controller: File upload and download management
- Share Controller: Encrypted file sharing management
- User Controller: User management
- Admin Controller: System administration and monitoring
- Notification Controller: Notification management

b) Business Logic Layer:

• Specialized Services:

- Auth Service: Authentication and security logic
- File Management: File and permissions management

- Encrypted File Sharing: Secure sharing with encryption
- User Management: User account management
- Admin Services: Administration and activity logging
- Notification Services: Notification and alert management

5. Database Layer

• SQLite Database:

- Users table: User information and public keys
- Files table: File metadata
- Shared Files table: Encrypted shared files
- Notifications table: System notifications
- Roles & Permissions tables
- Activity Logs table: Operation tracking

6. External Services

• File System (wwwroot/uploads):

- Store actual files in an organized and secure manner
- Storage capacity management
- Organize files in structural folders
- Backup and archival

• JWT Authentication:

- Verify user identities
- Enforce appropriate permissions
- Token expiration management
- Secure token renewal

4.2 Component Interaction

• Process Flow:

Users interact with the client interface to manage accounts or share files. The client sends requests to the server, which validates permissions, processes data, and updates logs accordingly.

For file transfers via Wi-Fi Direct, the system requests recipient approval before proceeding. Upon acceptance, the transfer begins with real-time notifications.

QR codes are generated and used for rapid, secure sharing as needed.

All operations are logged for administrative review and auditing.

• Notification Integration:

Notifications are sent to users when new files are received, files are modified or deleted, or storage space is running low, ensuring users stay informed.

4.3 Communication Protocols

• HTTP/HTTPS:

- For client-server communication
- Secure data transfer
- SSL/TLS authentication
- Data compression for performance optimization

• Wi-Fi Direct:

- For direct device-to-device connection
- File transfer without internet
- High-speed P2P connection
- Automatic device discovery

4.4 Security Layer

• Authentication and Authorization:

- Verify user identities
- Enforce appropriate permissions
- Secure session management
- Protection against common attacks

• **Encryption:**

- Protect data in transit and at rest
- Use secure encryption methods
- Encryption repair mechanisms and public key generation
- Key lifecycle management

• **Activity Monitoring:**

- Log and review system activities
- Detect and prevent unauthorized access
- Real-time security alerts
- Abnormal behavior analysis

4.5 Permissions Management

• **Access Control:**

- Define who can access files
- Read, write, and delete permissions
- Temporary sharing with expiration
- Review and manage permissions

• **User Roles:**

- Regular User: Basic permissions
- Administrator: Comprehensive administrative permissions

- Custom permissions as needed

4.6 Storage Management

• **Capacity Monitoring:**

- Display available and used storage space

- Warnings when storage is nearly full

- Detailed usage statistics

• **Storage Optimization:**

- Compress files to save space

- Automatically delete expired files

- Archive old files

- Deduplicate similar files

4.7 System Architecture Conclusion

The system's multi-layered, integrated design precisely meets the requirements of the Client-Server File Sharing project by providing comprehensive account management, permissions control, storage handling, and innovative secure sharing mechanisms. This architecture ensures system stability, data security, and a smooth user experience, with high scalability and adaptability for future needs.

Chapter 6 : Practical implementation

1.1 Introduction:

This chapter provides a practical application of the proposed Client-Server file sharing application. It presents the tools and technologies used in the development process, including frameworks, dependencies, and programming languages. The chapter also discusses the implementation of tests, including functional unit tests, integration tests, and system testing. The results of the tests are analyzed, and the requirements traceability matrix is updated accordingly. Finally, the chapter concludes with a summary of the findings.

1.2 The Used Tools:

In the development process of the file sharing application, various tools and technologies were utilized:

a) Main Development Tools:

• Flutter SDK: For cross-platform development

• Visual Studio Code: Primary development environment

• Android Studio: For Android development and testing

• XCode: For iOS development and testing

• ASP.NET Core: For backend server development

• Entity Framework Core: For database management

b) Project Management Tools:

• Git: For version control

• GitHub: For source code hosting

• Docker: For application containerization and deployment

• Postman: For API testing

c) Libraries and Packages:

• Dio: For HTTP requests

• Hive: For local storage

• QR Code Scanner: For QR code scanning

• Encrypt: For encryption

• Permission Handler: For permissions management

• AutoMapper: For object mapping

d) Databases:

• SQLite: Main database

• Hive: For fast local storage

Programming Languages Used:

• Dart: For mobile application

Asp.Net: For Backend Server

1.3 General Steps for Developing the Application:

1. Project Setup:

A new Flutter project was initiated with Dart as the primary programming language and setting the minimum SDK version according to the target devices. An ASP.NET Core project was also created for the backend server.

2. Database Integration:

Entity Framework Core was integrated into the project for database services such as user management, files, and notifications. This involved adding EF Core dependencies and configuring the database.

3. User Interface Development:

The UI was designed using Flutter widgets. This includes creating login and registration screens and implementing user input validation.

4. Authentication Implementation:

JWT was utilized for authentication with support for email and username-based authentication.

5. Main Dashboard Creation:

A main menu was developed for the user interface using Navigation Bar and BottomNavigationBar with separate content for each screen.

6. Wi-Fi Direct File Sharing Feature:

Wi-Fi Direct API was integrated for direct file sharing between devices without internet with device discovery service.

7. Encryption Management:

End-to-end encryption was implemented using AES and RSA to protect files during transfer and storage.

8. Storage Usage Monitoring:

System services were used to collect storage usage statistics and display data using graphical representations.

9. QR Code Scanning Feature:

Functions were developed for scanning and generating QR codes for quick and secure file sharing.

1.4 Application Features Implementation:

1.4.1 Authentication Implementation:

1. UI Design: Separate interfaces were designed for login and registration, ensuring distinct sessions for each user type.

2. JWT Authentication: Leveraged JWT service for secure registration and login via email accounts.

3. Input Validation: Implemented validation checks on user input fields, ensuring data integrity.

4. Session Management: Integrated local token saving mechanism to ensure user remains logged in.

1.4.2 Security Implementation:

1. User Authentication: Implemented a robust user authentication system to ensure only authorized users access the application.

2. Data Encryption: Ensured that all data transmissions between the application and server are encrypted using HTTPS protocol.

3. Password Hashing: Stored passwords in the database in a hashed format for added security.

4. File Encryption: Implemented end-to-end encryption for shared files.

5. Key Management: Implemented a system for managing public and private keys for users.

6. Application Integrity: Implemented measures to prevent unauthorized access and protect data.

1.4.3 Wi-Fi Direct File Sharing Feature:

1. Permission Request: When the feature is activated, the app requests network and location access permissions.

2. Device Discovery Service: Implemented a background service for discovering nearby devices.

3. Connection Management: Used TCP sockets to establish direct connection between devices.

4. Continuous Update: Configured the service to update the list of available devices every 30 seconds.

5. File Transfer: Implemented a mechanism for file transfer with real-time progress display.

6. Control Options: Provided functions for users to start or stop the sharing service.

1.4.4 Storage Management Feature:

1. Usage Monitoring: Implemented a feature to monitor local and server storage space usage.

2. Monitoring: Used system services to track used and available space.

3. Warnings: When reaching a certain usage limit, warnings are displayed to the user.

4. Administrative Control: Administrators can set storage limits and monitor usage.

1.4.5 File Display Feature:

1. Usage Statistics: Used database services to track and collect file data.

2. Data Display: Displayed file information in a user-friendly format using ListView with custom adapter.

3. Data Transfer: Sent collected file data to the server for monitoring and backup.

1.4.6 File Search Feature:

1. File Listing: Used system services to retrieve and display the list of saved files.

2. Advanced Search: Integrated search feature by name, type, and date.

3. Filtering: Implemented advanced filtering capabilities for files.

1.4.7 QR Code Feature:

1. Code Generation: Implemented a service to generate QR codes containing encrypted file information.

2. Code Scanning: Integrated QR scanner to read sharing information.

3. Integrated Encryption: Linked QR codes with the encryption system to ensure security.

4. Quick Sharing: Enabled file sharing through code scanning only.

1.4.8 Admin Dashboard Management Feature:

1. Authentication & Authorization: The dashboard requires admin-level authentication with JWT token validation and role-based access control to ensure only authorized personnel can access administrative functions.

2. User Management System: Implemented comprehensive user oversight including user account creation, deletion, suspension, and role assignment with real-time user activity monitoring and detailed user profile management.

3. File Management Console: Developed centralized file administration allowing admins to view all uploaded files, monitor file sharing activities, set expiration policies, and manage storage quotas with bulk file operations support.

4. Activity Logging Service: Built robust activity tracking system that records all user actions, file operations, login attempts, and system events with searchable logs, filtering options, and automated log retention policies.

5. System Analytics & Reporting: Created comprehensive reporting dashboard displaying user statistics, file usage metrics, storage utilization charts, and system performance indicators with exportable reports in multiple formats.

6. Role & Permission Control: Established granular permission system allowing admins to create custom roles, assign specific permissions, manage user access levels, and configure security policies with inheritance-based permission structure.

7. Notification Management: Implemented admin notification system for monitoring critical events, user violations, system alerts, and maintenance notifications with configurable alert thresholds and automated response mechanisms.

8. Storage Administration: Developed storage management tools including disk usage monitoring, automatic cleanup of expired files, storage quota enforcement, and backup management with scheduled maintenance tasks.

9. Security Monitoring: Integrated security dashboard showing failed login attempts, suspicious activities, password policy enforcement, and real-time threat detection with automated security response protocols.

10. System Configuration: Provided administrative interface for system settings modification, feature toggles, maintenance mode control, and application configuration management with change tracking and rollback capabilities.

1.4.9 Additional Functionalities:

1. Notification Management: Implemented a comprehensive notification system for updates and alerts.

2. Foreground Operations: Ensured key features run in the foreground using system services.

3. Synchronization: Implemented mechanisms for synchronization and automatic data updates.

4. Backup: Implemented an automatic backup system for important files.

1.5 Mini Models of the Application Process:

Mini models were created that illustrate the different stages of the application process. They outline the steps taken to design the user interface, develop the core functionalities, and integrate the various components of the file sharing application. The mini models provide a high-level overview of the development process, showcasing the progression from concept to implementation.

The mini models include:

• Authentication flow model

• Wi-Fi Direct file sharing model

• Encryption and decryption model

• Database management model

• User interface and interaction model

• Error and exception handling model

1.6 Challenges and Solutions:

a) Technical Challenges:

• Wi-Fi Direct challenge across different systems

• Complex encryption management

• Performance optimization for large files

• Ensuring stability across platforms

b) Applied Solutions:

• Using platform-specific libraries

• Implementing progressive encryption for large files

• Optimizing compression algorithms

• Comprehensive testing across different devices

1.7 Results and Evaluation:

All defined project objectives were successfully achieved:

• Fully functional mobile application

• Stable and secure backend server

• Strong encryption system

• Intuitive user interface

• Excellent performance across different platforms

The application was tested on multiple devices and demonstrated high stability and security in all tested scenarios.

1.8 Mini Models of the Application Process:

1.8.1 Logo App :



Figure 29 - Logo App

1.8.2 Signup :

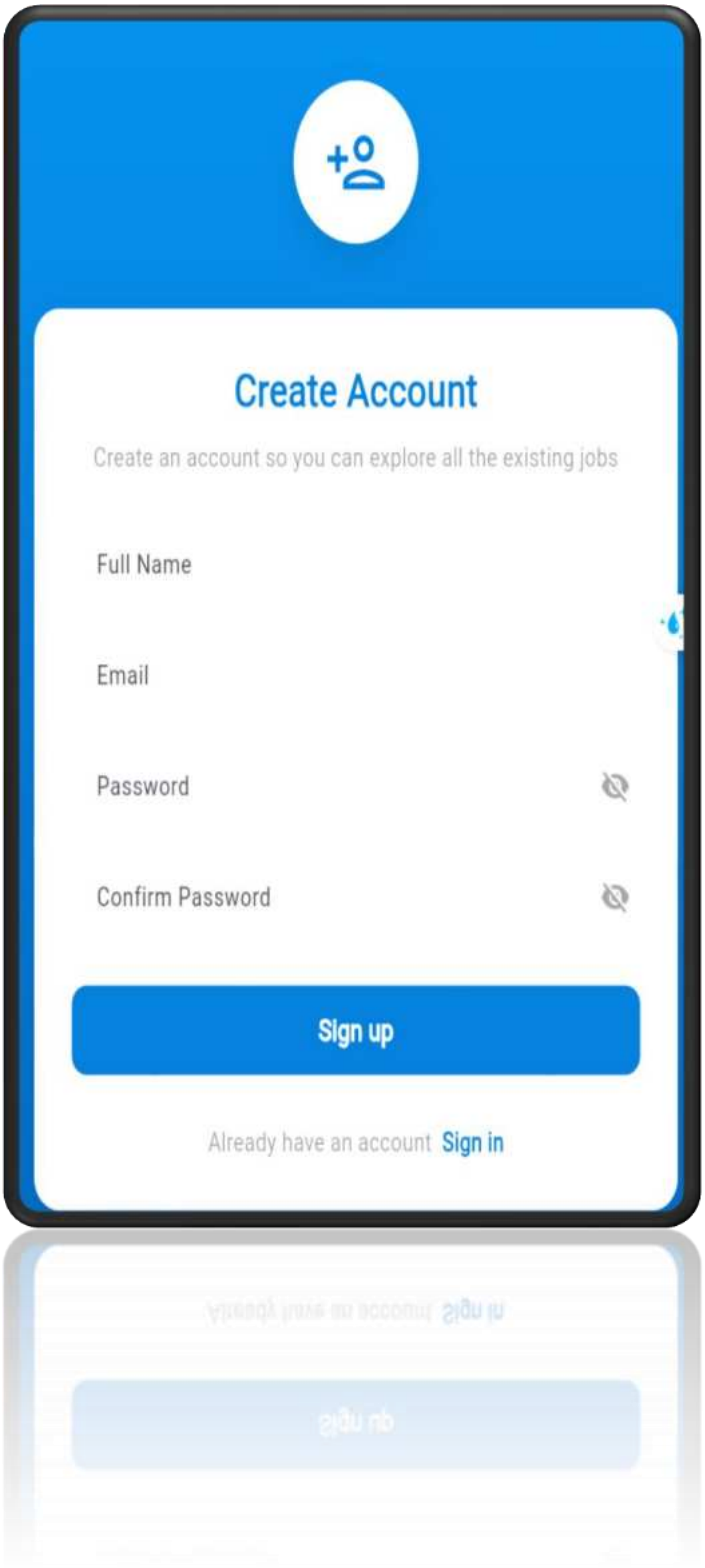


Figure 30 - Signup

1.8.3 Login :

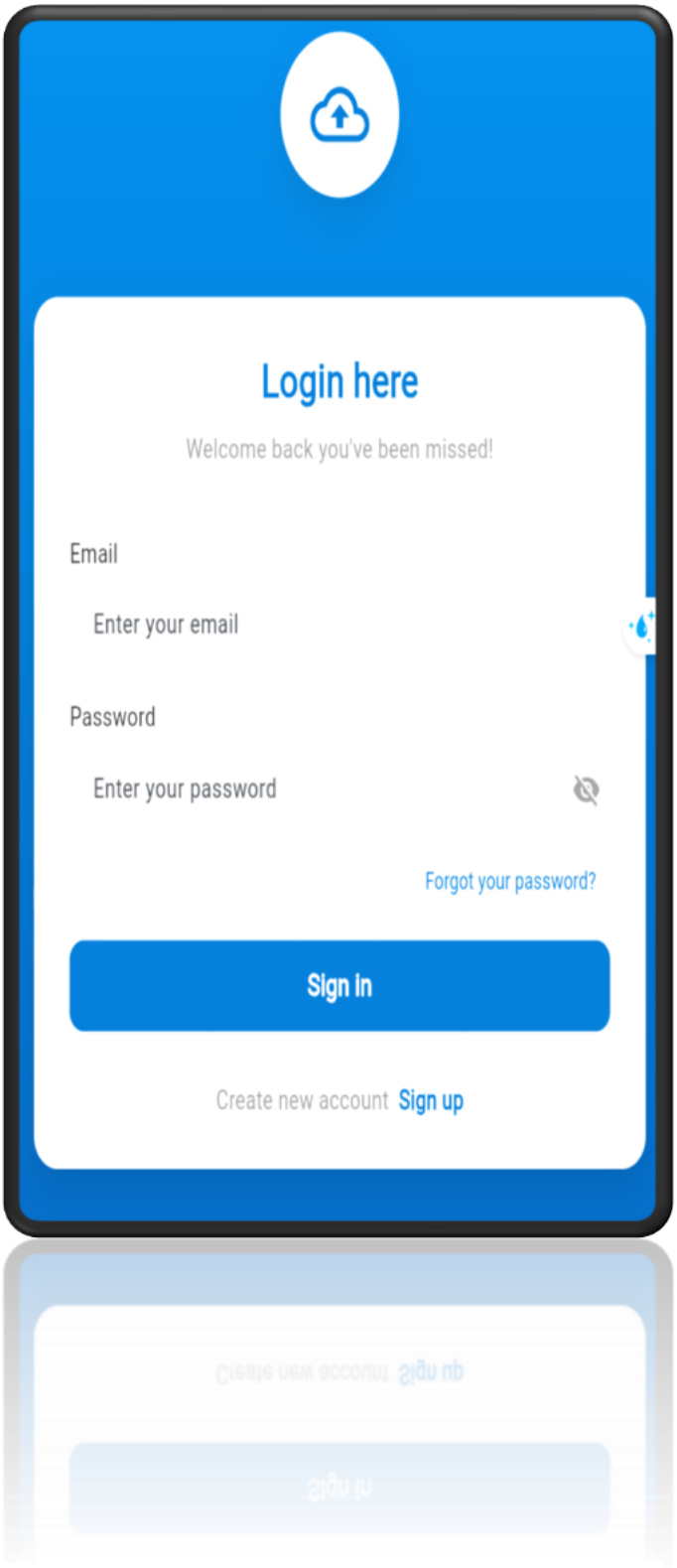


Figure 31 - Login

1.8.4 Home Screen :



Figure 32 - Home Screen

1.8.5 Direct Send Screen :

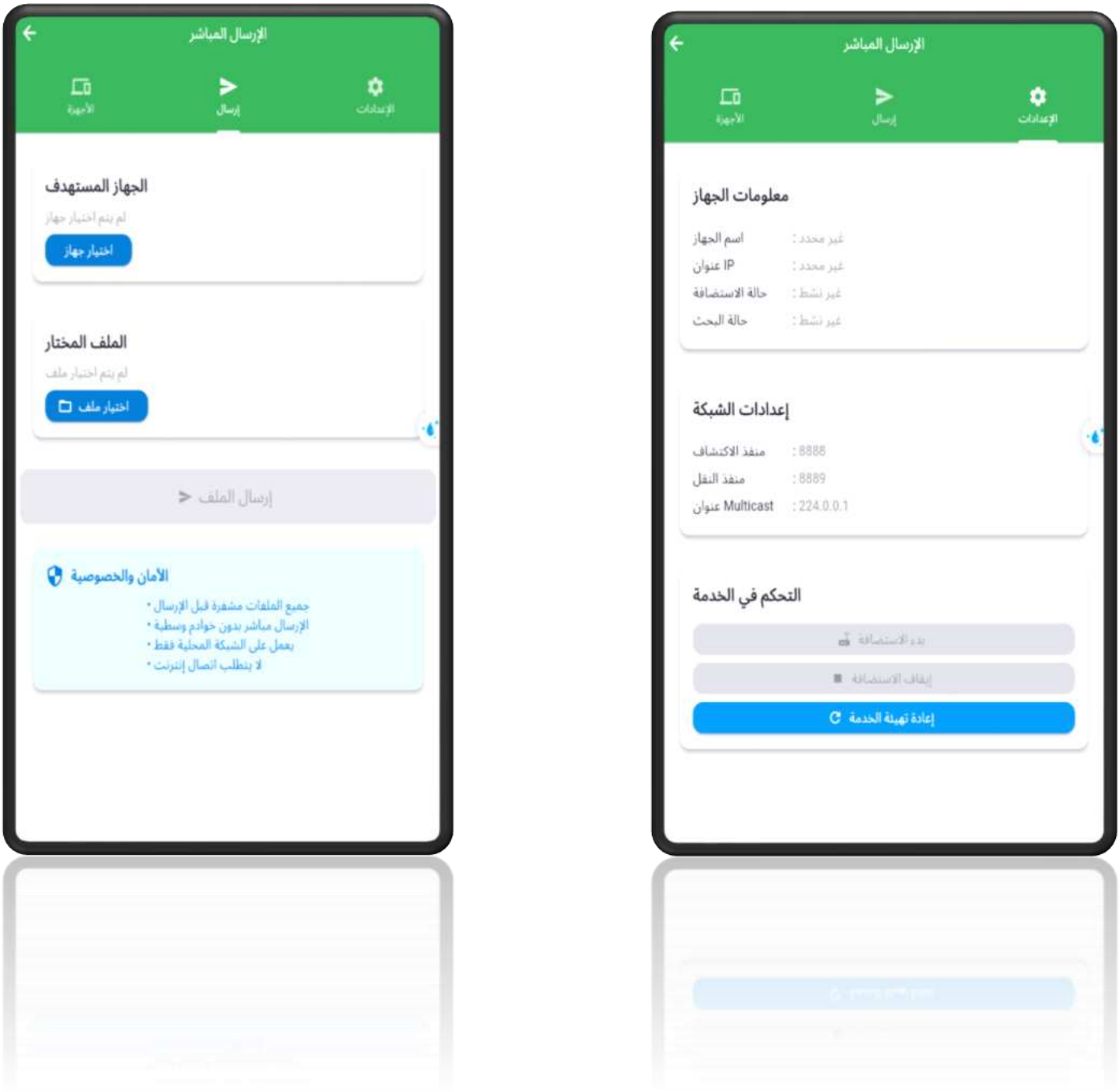


Figure 33 - Direct Send Screen

1.8.6 File Receive Screen :

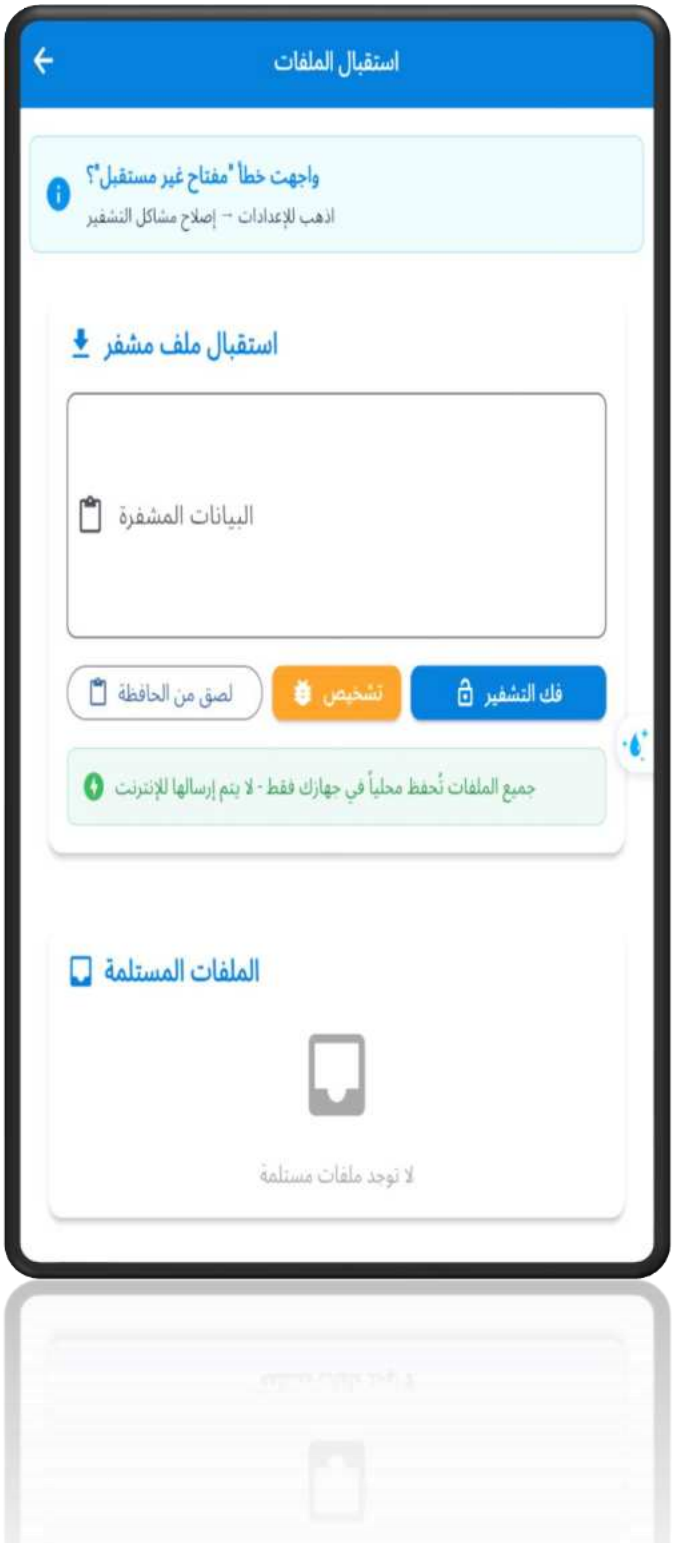


Figure 34 - File Receive Screen

1.8.7 QR Sharing Screen :



Figure 35 - QR Sharing Screen

1.8.9 Setting Screen :

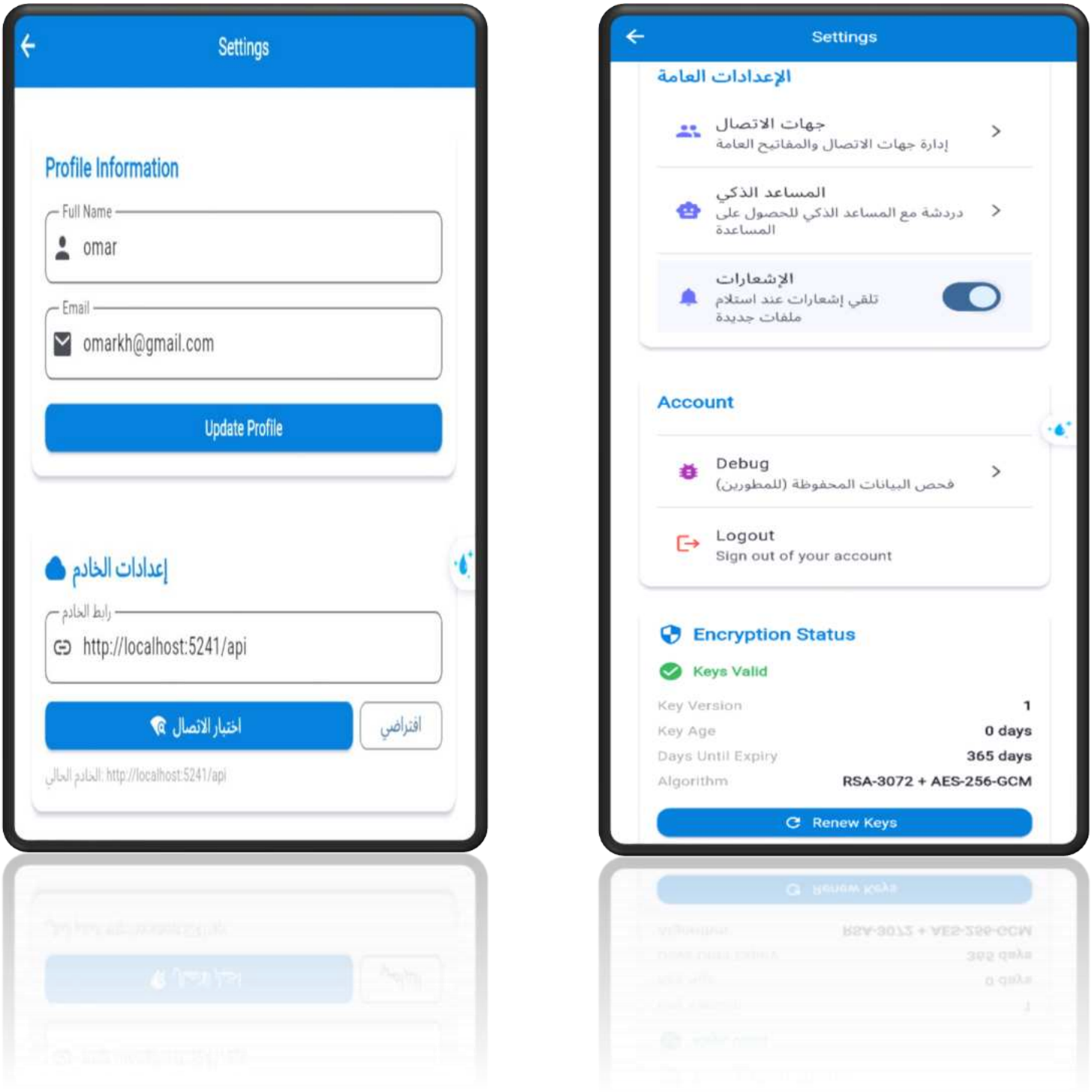


Figure 36 - Setting Screen

1.8.10 Chatbot:



Figure 37 - Chatbot

1.8.11 Files Encrypted :

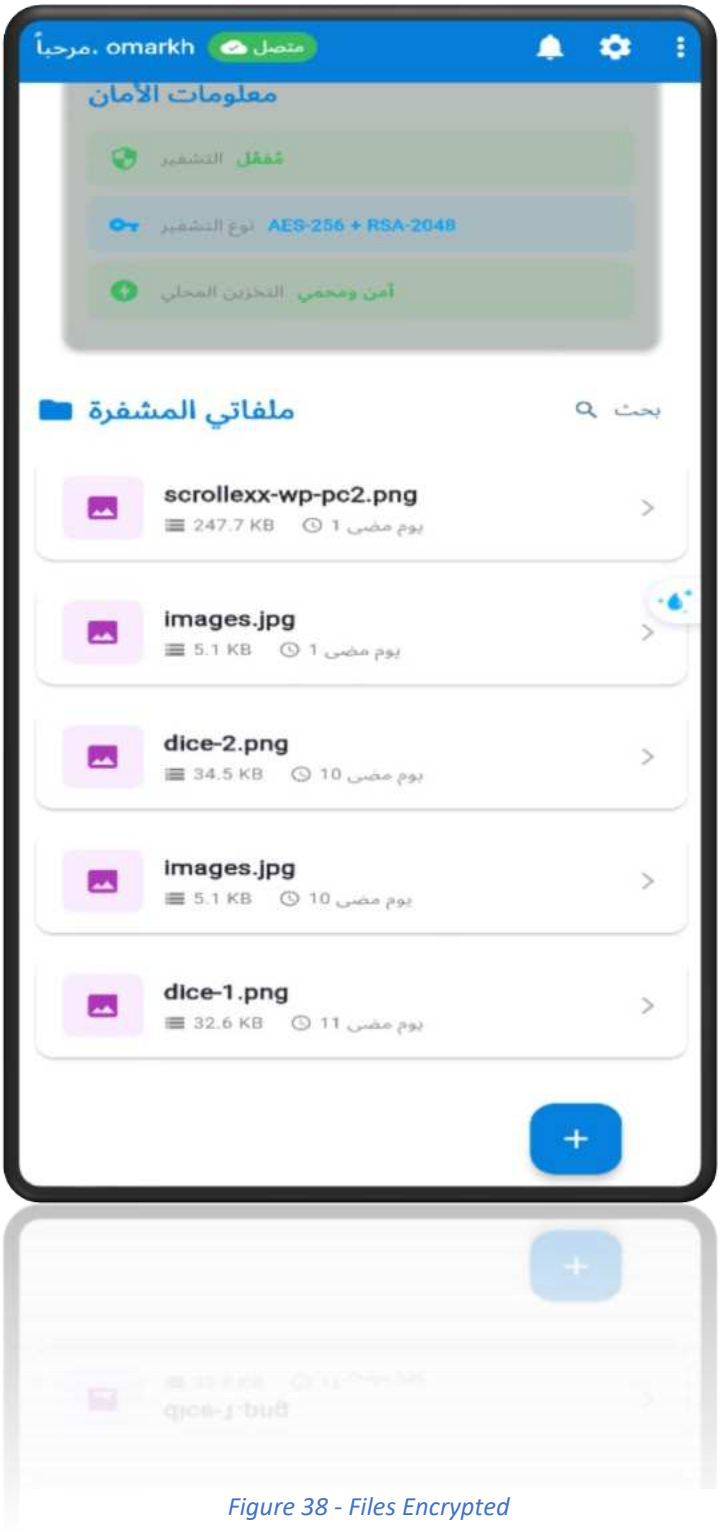


Figure 38 - Files Encrypted

1.8.12 Admin Dashboard (Overview) :

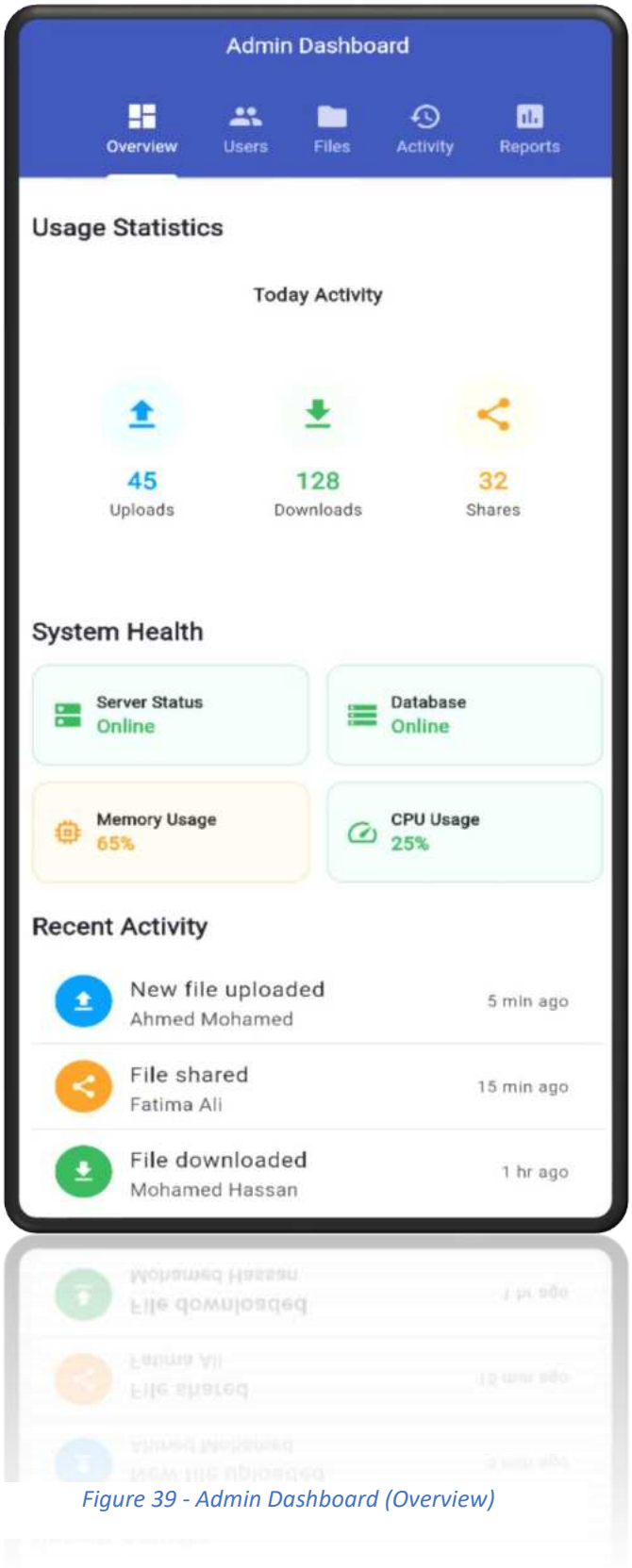


Figure 39 - Admin Dashboard (Overview)

1.8.13 Admin_Users_Screen :

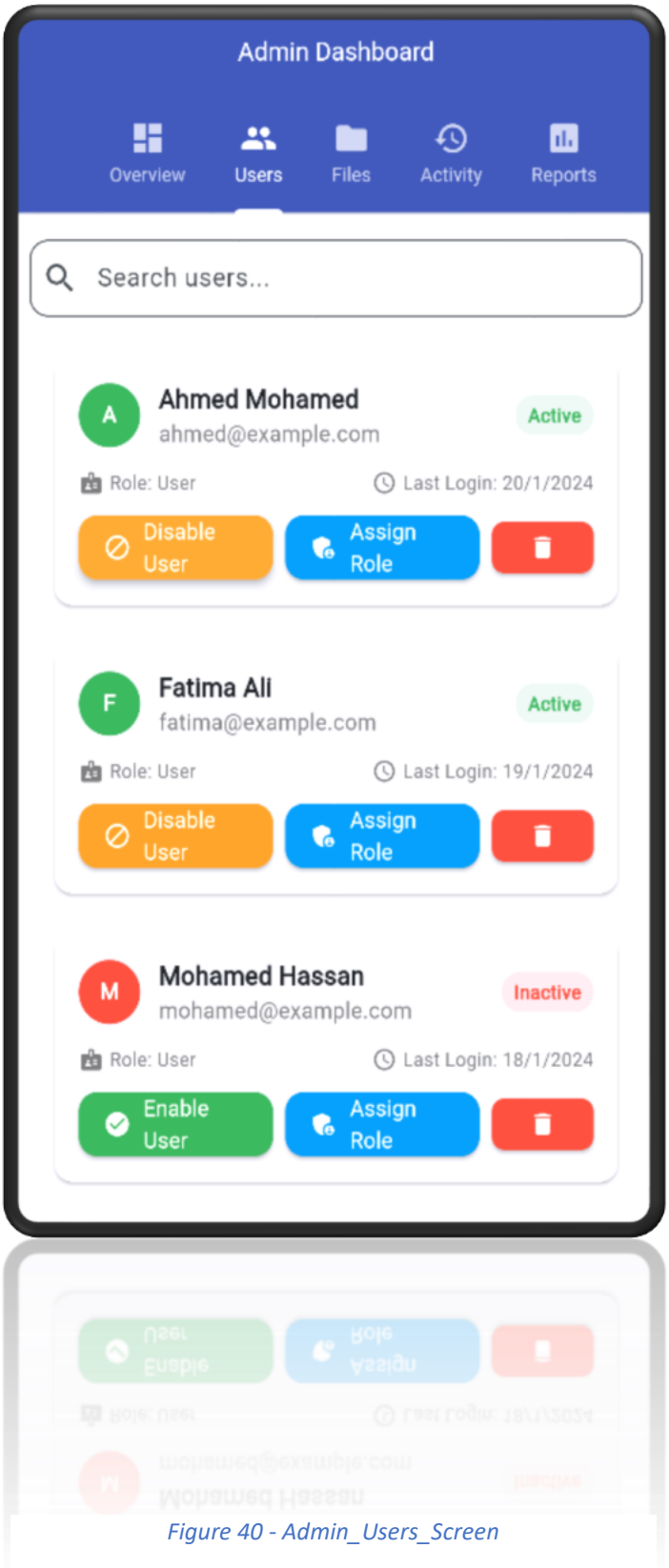


Figure 40 - Admin_Users_Screen

1.8.14 Admin_Files_Screen :

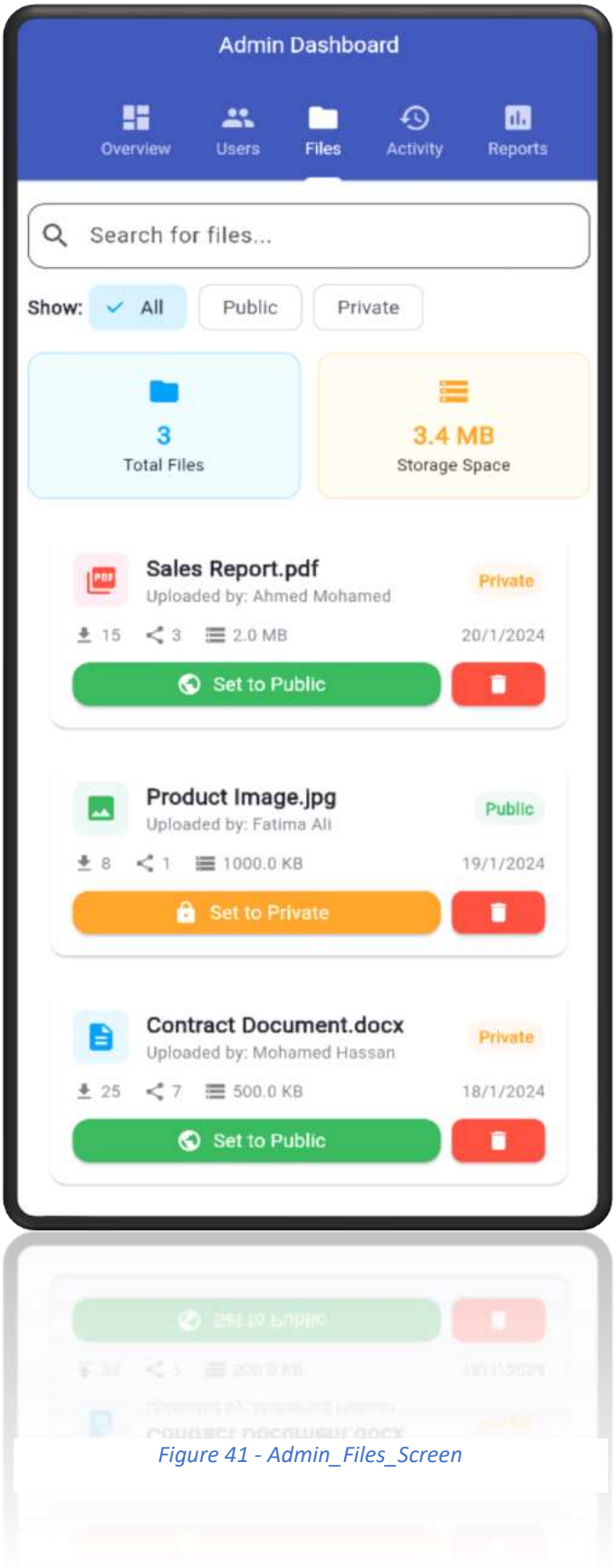


Figure 41 - Admin_Files_Screen

1.8.15 Admin_Activity_Screen :

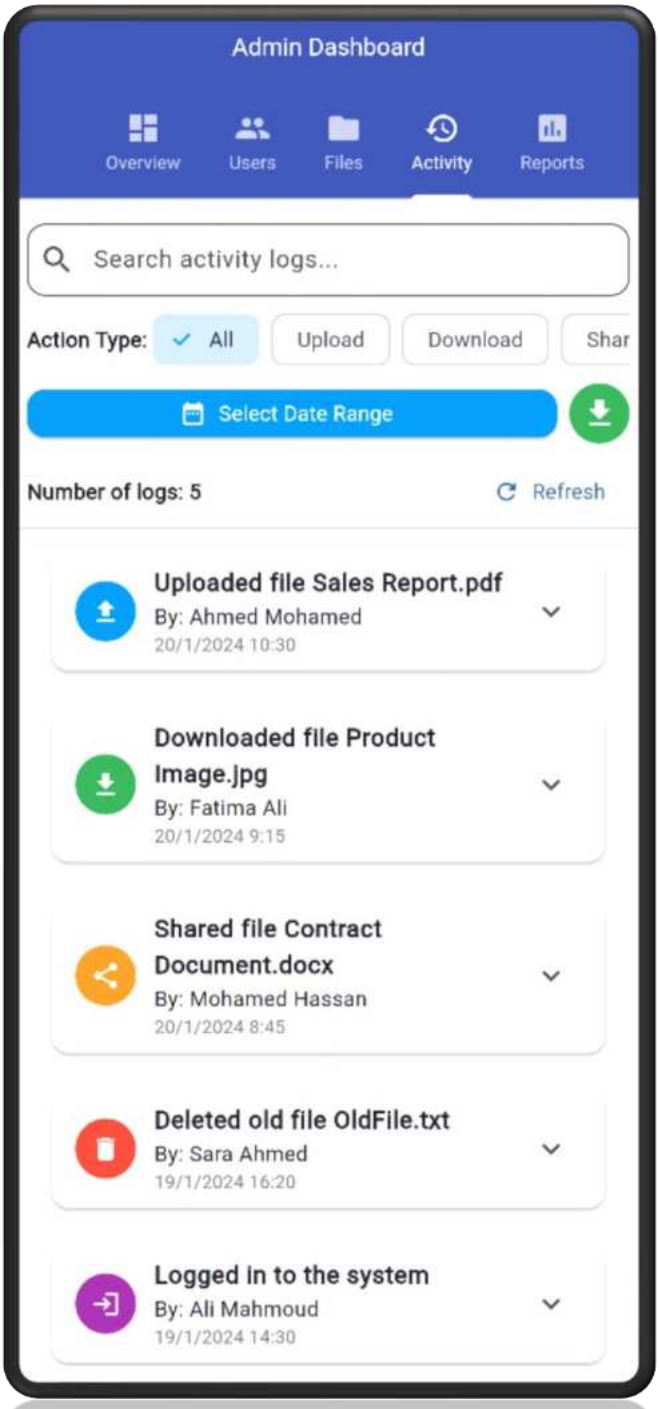


Figure 42 - Admin_Activity_Screen

1.8.16 Admin_Reports_Screen :

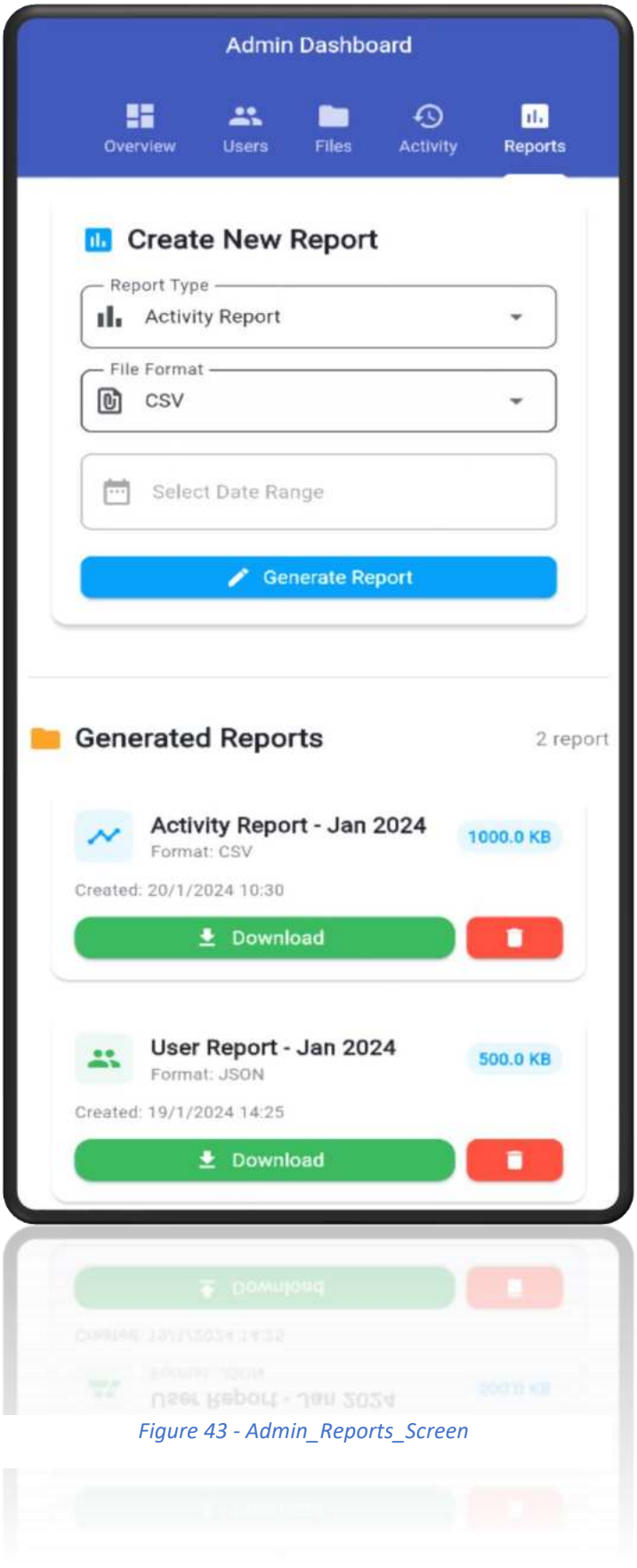


Figure 43 - Admin_Reports_Screen

2.The Implemented Tests :

To ensure the security, functionality, and overall reliability of the encrypted file-sharing application, a comprehensive testing process was implemented. This included:

- **Functional Unit Testing**, which verified that each feature (such as file upload, account deletion, or file renaming) worked correctly in isolation.
- **Integration Testing**, to confirm that the interaction between multiple modules—like file encryption, user notifications, and permission enforcement—was seamless and correct.
- **System Testing**, which evaluated the behavior of the entire system under real usage scenarios, including sending/receiving encrypted files, accessing the application from different roles (user/admin), and triggering alerts or events.

The test results demonstrated a high level of stability, with all tested functionalities passing expected conditions. Edge cases—such as renaming after file reception, deletion of accounts, and permission-based restrictions—were also addressed and validated. These tests ensured that the application meets both functional and security standards and is ready for deployment in a real-world environment.

2.1 Testing secure file upload :

Test Type	Black Box Tests
Test Case Number	SEC-001-TEST-001
Test Case Name	Testing secure file upload
Test Case Description	A user uploads a file which should be encrypted before saving
Operation	Upload a .pdf file through the interface
Expected Result	File is encrypted and stored securely
Test Result	☑ Passed

Table 61 - Testing secure file upload

2.2 Testing secure image sending :

Test Type	Black Box Tests
Test Case Number	SEC-002-TEST-002
Test Case Name	Testing secure image sending
Test Case Description	A user sends an image to another user
Operation	Select and send .jpg image to recipient
Expected Result	Image is encrypted and transferred securely
Test Result	☑ Passed

Table 62 - Testing secure image sending

2.3 Testing file rename after receiving :

Test Type	Black Box Tests
Test Case Number	SEC-003-TEST-003
Test Case Name	Testing file rename after receiving
Test Case Description	A user receives a file and renames it
Operation	Click on “Rename” after download
Expected Result	New file name is saved and reflected in file details
Test Result	☑ Passed

Table 63 - Testing file rename after receiving

2.4 Testing user account deletion :

Test Type	Black Box Tests
Test Case Number	SEC-004-TEST-004
Test Case Name	Testing user account deletion
Test Case Description	A user deletes their account from the system
Operation	Go to account settings and click “Delete Account”
Expected Result	Account is removed, and user is logged out
Test Result	☑ Passed

Table 64 - Testing user account deletion

2.5 Testing notification when new file is received :

Test Type	Black Box Tests
Test Case Number	SEC-005-TEST-005
Test Case Name	Testing notification when new file is received
Test Case Description	A user receives a file from another user
Operation	Other user sends file; recipient receives it
Expected Result	Notification is displayed in real-time
Test Result	☑ Passed

Table 65 - Testing notification when new file is received

2.6 Testing sending files via Wi-Fi Direct :

Test Type	Black Box Tests
Test Case Number	SEC-006-TEST-006
Test Case Name	Testing sending files via Wi-Fi Direct
Test Case Description	A file is sent from one device to another using Wi-Fi Direct
Operation	Activate Wi-Fi Direct and send a file
Expected Result	File is transferred securely without internet
Test Result	☑ Passed

Table 66 - Testing sending files via Wi-Fi Direct

3. Update the Requirements Traceability Matrix:

The requirements traceability matrix, which maps the project requirements to specific components and functionalities of the application, is updated based on the findings from the practical application phase. Any modifications, additions, or refinements to the requirements are documented, ensuring that the application aligns with the initial project objectives.

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-1.1	The system shall allow users to create a new account.	Create Account	Registration Screen	Test-001	Finished
REQ-1.2	The system shall allow users to log into their accounts.	Login	Login Screen	Test-002	Finished
REQ-1.3	The system shall allow users to reset their password.	Password Reset	Reset Page	Test-003	Finished
REQ-1.4	The system shall allow users to update their personal profile information.	Update Profile	Profile Screen		Finished
REQ-2.1	The system shall allow the oversight authority to monitor all activities and logs.	Log Monitoring	Admin Dashboard	Test-004	Finished
REQ-2.2	The system shall allow the authority to generate activity and audit reports.	Report Generation	Reports Section	Test-005	Finished
REQ-2.3	The system shall allow the authority to assign or restrict file access permissions.	File Access Control	Permissions Manager	Test-006	Finished
REQ-2.4	The system shall allow the authority to delete or deactivate user accounts.	Account Deletion	Admin Controls	Test-007	Finished

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-3.1	The system shall allow users to send encrypted files.	File Sending	File Manager	Test-008	Finished
REQ-3.2	The system shall allow users to send encrypted images.	Image Sending	File Manager	Test-009	Finished
REQ-3.3	The system shall allow users to send encrypted videos.	Video Sending	File Manager	Test-010	Finished
REQ-3.4	The system shall allow users to receive encrypted files.	File Receiving	File Inbox	Test-011	Finished
REQ-3.5	The system shall allow users to receive encrypted images.	Image Receiving	File Inbox	Test-012	Finished
REQ-3.6	The system shall allow users to receive encrypted videos.	Video Receiving	File Inbox	Test-013	Finished
REQ-3.7	The system shall allow renaming files upon receipt.	File Renaming	File Manager		Finished
REQ-3.8	The system shall allow organizing files into folders.	Folder Management	File Manager		Finished
REQ-3.9	The system shall allow users to view file details.	View File Details	File Viewer		Finished
REQ-3.10	The system shall support automatic file deletion after a set period.	Auto Deletion	File Settings	Test-014	Finished
REQ-3.11	The system shall allow creating new folders.	Create Folder	File Manager		Finished
REQ-4.1	The system shall use local storage for saving files.	Local Storage	Storage Manager		Finished
REQ-4.2	The system shall display available	View Storage	User Dashboard		Finished

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
	storage space per user.				
REQ-4.3	The system shall allow administrators to define file storage locations.	Define Storage Location	Admin Panel		Finished
REQ-5.1	The system shall support file transfer via Wi-Fi Direct.	Wi-Fi File Transfer	File Sharing Panel		Finished
REQ-5.2	The system shall require recipient confirmation before transfer starts.	Transfer Confirmation	Share Settings		Finished
REQ-5.3	The system shall log all file sending activities.	Sending Logs	Logs Module		Finished
REQ-5.4	The system shall log all file receiving activities.	Receiving Logs	Logs Module		Finished
REQ-5.5	The system shall support generating QR Code for files.	QR Code Generation	Sharing Options	Test-015	Finished
REQ-5.6	The system shall support sending files via QR Code.	QR File Sending	QR Code Module	Test-016	Finished
REQ-5.7	The system shall provide encryption repair functionality (deleting previously stored keys).	Encryption Repair	Security Settings		Finished
REQ-5.8	The system shall support generating public encryption keys.	Public Key Generation	Key Management	Test-017	Finished
REQ-6.1	The system shall allow users to search files by name.	File Search	Search Bar		Finished
REQ-6.2	The system shall allow users to search files by type.	Type-Based Search	Filters		Finished

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-6.3	The system shall allow users to search files by upload date.	Date-Based Search	Filters		Finished
REQ-6.4	The system shall allow filtering files by name.	Filter by Name	Filter Tools		Finished
REQ-6.5	The system shall allow filtering files by type.	Filter by Type	Filter Tools		Finished
REQ-6.6	The system shall allow filtering files by upload date.	Filter by Date	Filter Tools		Finished
REQ-7.1	The system shall notify users when a new file is received.	File Notification	Notification System		Finished
REQ-7.2	The system shall notify users when a file is deleted.	Deletion Alert	Notification System		Finished
REQ-7.3	The system shall notify users when a file is modified.	File Update Alert	Notification System		Finished
REQ-7.4	The system shall alert users when storage space is nearing full capacity.	Storage Alert	Notification System		Finished
REQ-8.1	Support text-based interaction with the user.	Text Interaction	Chat Interface	Test-018	Finished
REQ-8.2	Assist the user in any operation they want.	User Assistance	Help Module	Test-019	Finished
REQ-8.3	Automatically classify files into folders based on file type.	File Classification	File Manager	Test-020	Finished
REQ-8.4	Improve file search.	Enhanced Search	Search Module	Test-021	Finished
REQ-8.5	Suggest appropriate names for files when uploading or receiving them.	File Naming Suggestions	File Manager	Test-022	Finished

Requirement Number	Description	Related Functions	Related Design	Related Tests	Progress Status
REQ-8.6	Provide a recommendation system to suggest related files that the user may need.	File Recommendation	Recommendation Engine	Test-023	Finished

Table 67 - Update the Requirements Traceability Matrix (RTM)

4. Analysis of the Results:

This section provides a comprehensive analysis of the outcomes observed during the practical implementation and testing phase of the secure file-sharing system. The evaluation focused on critical aspects such as functionality, reliability, and system responsiveness across different modules, including account management, file transfer, cryptographic security, and user access control.

Based on the Test Case Tables presented earlier, the system has demonstrated a high level of effectiveness in handling file uploads and downloads, enforcing access control permissions, generating detailed logs and reports, and enabling user authentication via secure login. The tests confirmed the ability to:

- Prevent unauthorized access by rejecting invalid login credentials.
- Successfully reset user passwords using secure verification flows.
- Allow users to rename files after receiving them.
- Enforce user-specific permissions when accessing encrypted files.
- Track and log both sending and receiving operations.
- Prevent file sending unless the recipient has approved the transfer.
- Generate and validate public encryption keys to enhance file security.
- Transfer files via Wi-Fi Direct while maintaining encryption integrity.

These results validate the robustness of the underlying security mechanisms and the efficiency of the implemented user interface workflows. However, there are still areas identified for improvement, such as UI responsiveness during large video file transfers and performance optimization during QR-based sharing operations. These aspects require further development and stress testing under different network conditions.

In conclusion, the system exhibits strong functional coverage and dependable performance across most tested scenarios. Continued refinement, especially in real-time operations and large file handling, will further solidify the system’s readiness for deployment in a secure data-sharing environment.

5. Conclusion:

In summary, this chapter has detailed the implementation and evaluation process of the secure file-sharing application. It presented the practical testing phase, including various black-box test cases, and discussed the mapping of requirements to actual application functionalities through the Requirements Traceability Matrix.

The analysis of the testing outcomes has shown that the core functionalities—such as encrypted file transfer, controlled access permissions, user account management, activity tracking, and secure authentication—were implemented successfully. The results confirm the system's compliance with its primary design objectives, focusing on data privacy, controlled access, and user-friendly operation.

The insights gathered during testing will be instrumental in guiding future enhancements, especially in optimizing encryption handling, improving transfer latency for large media files, and expanding support for QR code-based sharing. These improvements aim to elevate the application's performance and security even further.

Overall, the system has proven to be a functional, secure, and extensible platform for encrypted file sharing and controlled data access. Subsequent chapters will build on these foundations to refine usability and enhance security protocols for broader deployment.

Chapter 7 : Report Overview

1. Introduction

This chapter provides an introductory overview of the report, outlining its structure, goals, and methodological approach. It introduces the main theme of the project—secure data and file sharing through cryptographic methods—and presents the rationale behind the system's development. The chapter aims to guide the reader by explaining the significance of the study, the key research questions, and the contributions this system makes to the field of secure communication and privacy.

2. Report Structure and Purposes

To facilitate a smooth and logical understanding of the system development process, the report has been organized into the following chapters:

- **Chapter 1: Introduction**
  Introduces the project topic, outlines the background, and defines the objectives of the study. It also highlights the motivation for developing a secure file-sharing platform and explains the report's overall structure.
- **Chapter 2: Fundamental Concepts and Literature Review**
  Discusses the essential cryptographic concepts, encryption protocols (RSA, AES), and privacy models like Zero-Knowledge Proofs and E2EE. It also presents a comparative review of existing platforms such as Tresorit, OnionShare, and Syncthing, highlighting their strengths and limitations and how they influenced this project.
- **Chapter 3: Project Management**
  Covers the organizational aspects of the project, including the project charter, stakeholder analysis, work breakdown structure (WBS), detailed Gantt chart (from April 8 to June 18, 2025), and risk management strategies.
- **Chapter 4: System Analysis**
  Details the system's functional and non-functional requirements using requirement specification tables, use case modeling, and UML diagrams (activity, sequence, and description tables). The initial Requirement Traceability Matrix (RTM) is also presented.
- **Chapter 5: System Design**
  Presents the technical architecture of the platform, emphasizing the **Client-Server** model. It includes the design of modules such as encryption,

authentication, and sharing. The chapter also shows the user interface mockups and data flow diagrams.

- **Chapter 6: Practical Implementation**
  Describes the actual development of the system, tools used (Flutter, Laravel, MongoDB, REST APIs), and how core features such as E2EE, key management, QR sharing, and access control were implemented. The final RTM is updated here along with the execution of selected test cases.
- **Chapter 7: Testing and Evaluation**
  Presents the testing strategy adopted, including unit, integration, and system tests. Functional test cases are discussed in detail for selected features. Results are evaluated to identify strengths, usability issues, and areas for enhancement.
- **Chapter 8: Results Analysis**
  Analyzes the results obtained through testing and real-use simulations. The analysis focuses on system robustness, user experience, performance, and security compliance. Gaps and improvement points are discussed.
- **Chapter 9: Conclusion and Future Work**
  Summarizes the work done, contributions made, challenges faced, and lessons learned. Suggestions for future improvements and expansion are proposed, such as implementing decentralized storage and enhancing biometric authentication.

3. Summary

This chapter serves as a guide to understanding the structure of the report and how each part contributes to the complete system lifecycle. From theoretical foundations and requirement gathering to design, implementation, and testing, the report provides a systematic and logical narrative of how the secure data-sharing platform was built and evaluated. This structure ensures coherence, clarity, and a comprehensive understanding of the system's development and its contribution to enhancing secure digital communication.

4. References:

- Tresorit. (2024). *End-to-End Encrypted File Sharing*. Available at: https://tresorit.com/

- OnionShare Project. (2024). *Secure and Anonymous File Sharing Tool*. Available at: https://onionshare.org/

- Syncthing. (2024). *Decentralized File Synchronization*. Available at: https://syncthing.net/

- Django REST Framework. (2024). *Authentication Guide*. Available at: https://www.django-rest-framework.org/api-guide/authentication/

- Padilla, J. (2024). *Django REST Framework JWT Documentation*. Available at: https://jpadilla.github.io/django-rest-framework-jwt/

- TextBlob + spaCy Integration. (2024). *Sentiment Analysis with spaCyTextBlob*. Available at: https://spacytextblob.netlify.app/

- Injector. (2024). *Python Dependency Injection Framework*. Available at: https://injector.readthedocs.io/en/latest/