Faculty of Engineering Department of Informatics Engineering-Software and Information Systems Engineering

Government Document Management system

{Wathiq}

Junior project

Prepared by

Bushra alshaabani Najat bostaty Salam almasri

Supervised by

Dr.Riad sonbol

Eng. Raghad alhossny

2025

كلية الهندسة

قسم الهندسة المعلوماتية

هندسة البرمجيات ونظم المعلومات

نظام إدارة الوثائق الحكومية

وثق

مشروع فصلي

إعداد

بشرى الشعباني **نجاة بسطاطي** **سلام المصري**

إشراف

د. رياض سنبل

م. رغد الحصني

2025

Abstract

In the modern era of digital transformation, government institutions face increasing pressure to manage, store, and secure vast amounts of documents efficiently. Traditional paper-based archiving methods are time-consuming, error-prone, and lack transparency.

This project presents the Government Document Management System as a smart solution that leverages Optical Character Recognition (OCR) and Artificial Intelligence (AI) technologies to digitize paper documents, extract textual content automatically, and enrich them with structured metadata. The system provides a secure and user-friendly environment for uploading, classifying, and retrieving documents, while maintaining detailed audit logs to ensure accountability and transparency.

By enabling automated classification, and comprehensive reporting, the system aims to enhance efficiency in government workflows, reduce reliance on manual archiving, and support the broader vision of digital governance.

Ultimately, this project contributes to strengthening institutional performance, improving citizen services, and laying the foundation for future integration with advanced AI-driven analytics.

ملخص

في عصر التحول الرقمي، تواجه المؤسسات الحكومية ضغوطًا متزايدة لإدارة وتخزين وتأمين كميات هائلة من الوثائق بكفاءة. إن الطرق التقليدية للأرشفة الورقية تستهلك وقتًا كبيرًا، وتفتقر إلى الدقة والشفافية.

يقدّم هذا المشروع نظام إدارة الوثائق الحكومية كحل ذكي يعتمد على تقنيات التعرف الضوئي على الحروف (OCR) والذكاء الصناعي لتحويل الوثائق الورقية إلى صيغة رقمية، واستخراج محتواها النصي تلقائيًا، وإثرائها ببيانات وصفية منظمة. يوفر النظام بيئة آمنة وسهلة الاستخدام لرفع الوثائق وتصنيفها واسترجاعها، مع تسجيل جميع العمليات في سجلات تدقيق لضمان الأمان والشفافية.

من خلال التصنيف الآلي، والتقارير الشاملة، يسعى النظام إلى رفع كفاءة العمل الحكومي، تقليل الاعتماد على الأرشفة اليدوية، ودعم رؤية التحول الرقمي.

في النهاية، يساهم المشروع في تعزيز الأداء المؤسسي، تحسين الخدمات المقدمة للمواطنين، ووضع الأساس للتكامل المستقبلي مع تحليلات متقدمة مدعومة بالذكاء الصناعي.

Table of contents

- Abstract ..... p. i
- Chapter 1: Introduction
  - 1.1 Objective ..... p. 1
  - 1.2 Project Scope ..... p. 2
    - 1.2.1 Phase 1 (MVP) – Core Functions ..... p. 3
    - 1.2.2 Phase 2 – Advanced Features ..... p. 4
    - 1.2.3 Phase 3 – Expansion and Integration ..... p. 5
  - 1.3 Technical Scope ..... p. 6
    - 1.3.1 Frontend ..... p. 7
    - 1.3.2 Backend ..... p. 7
    - 1.3.3 Database ..... p. 8
    - 1.3.4 OCR Services ..... p. 8
    - 1.3.5 Security ..... p. 9
    - 1.3.6 Integration ..... p. 9

- Chapter 2: Project Management
  - 2.1 Introduction ..... p. 10
  - 2.2 Project Charter ..... p. 11
  - 2.3 Project Objectives ..... p. 14
  - 2.4 Project Approach ..... p. 15
  - 2.5 Stakeholders ..... p. 16
  - 2.6 Statement of Work (SOW) ..... p. 17

- 2.7 Project Plan (Gantt Chart) ..... p. 25
- 2.8 Risk Management ..... p. 26
- 2.9 Work Breakdown Structure (WBS) ..... p. 27
- 2.10 Summary ..... p. 28

- Chapter 3: Fundamental Concepts and Literature Review
- 3.1 Introduction ..... p. 29
- 3.2 Fundamental Concepts ..... p. 30
- 3.3 Literature Review ..... p. 34
- 3.4 Summary ..... p. 39
- Comparative Analysis Table ..... p. 40

- Chapter 4: System Analysis
- 4.1 Introduction ..... p. 41
- 4.2 Software Requirement Specification (SRS) ..... p. 42
- System Description ..... p. 42
- Inputs and Outputs ..... p. 43
- Performance Requirements ..... p. 44
- Interface Requirements ..... p. 45
- Security Requirements ..... p. 46
- Maintenance Requirements ..... p. 47
- SRS Version History ..... p. 48
- Purpose ..... p. 49

- Project Scope ..... p. 50
- High-Level Requirements ..... p. 51
- Actors (User, Manager, System Admin) ..... p. 52
- General Description ..... p. 53
- Product Perspective ..... p. 54
- Product Features ..... p. 55

List of Tables

- Table 1: A brief analysis of similar systems and the target system ..... p. 40
- Table 2: Project Charter ..... p. 11–12
- Table 3: Project Schedule ..... p. 24
- Table 4: Project Resources ..... p. 23
- Table 5: SRS Version History ..... p. 48

List of Figures

- Figure 1: Gantt Chart ..... p. 25
- Figure 2: Work Breakdown Structure (WBS) ..... p. 27
- Figure 3: Use Case Diagram ..... p. 42
- Figure 4: Sequence Diagrams ..... p. 45–55

chapter1
Introduction

1. introduction:

1.1 Objective

The Government Document Management System aims to establish a secure and efficient digital environment for managing official documents. The system allows users to upload paper-based or scanned documents, convert them into searchable digital formats using Optical Character Recognition (OCR) and Artificial Intelligence (AI), and enrich them with structured metadata. It also provides audit logs to ensure transparency, supports reporting and enhances government workflow efficiency by reducing reliance on manual archiving.

1.2 Project Scope

The project scope defines the boundaries of work and the functions covered by the system. It is divided into three main phases:

1.2.1 Phase 1 (MVP) – Core Functions

- Upload documents in PDF and image formats.
- Extract text using OCR.
- Manage metadata (add, update, delete).
- Record all operations in audit logs.
- Provide basic text search within documents.

1.2.2 Phase 2 – Advanced Features

- Generate detailed reports.
- Improve user interface.
- Support semantic search.
- Manage user roles (Admin, Manager, User).
- Detect duplicate documents before storage.

1.2.3 Phase 3 – Expansion and Integration

- Integrate with other government systems via APIs.
- Enable AI-based document classification.
- Provide predictive analytics for decision support.
- Enhance security and encryption.
- Improve performance and scalability to handle larger volumes of documents.

1.3 Technical Scope

The system's scope includes all essential aspects required to build an effective and secure governmental document management platform that supports users and administrators.

1.3.1 Frontend:

Design and development of user interfaces using React.js, supported by modern libraries for a smooth and interactive user experience.

1.3.2 Backend:

Using .NET Framework to implement the system's logic, enabling efficient management of documents, users, and permissions.

1.3.3 Database:

Using the non-relational MongoDB database to store documents, metadata, and operation logs in a flexible and secure manner.

1.3.4 OCR Services:

An advanced OCR (Optical Character Recognition) engine was integrated to convert images and scanned documents into digital text that is searchable and indexable, which is the core feature that provides high efficiency in information archiving and retrieval.

1.3.5 Security:

The system implements robust security protocols, including data encryption, secure user authentication, and protection against unauthorized access to ensure the confidentiality of sensitive documents.

1.3.6 Integration:

The architecture is designed to support seamless Integration between the OCR engine, the document storage system, and the database, ensuring a unified and consistent data flow across all system modules.

chapter2
Project management

1. introduction:

In this chapter, we will examine the project management phase, which includes: Project Charter, Project Plan, (Statement of Work - SOW), Work Breakdown Structure (WBS), Project Management Strategies and Risk Management Strategies, and Analysis of Project Performance. The project is designed to guide and control the project effectively. The project is a preliminary step towards completion.

2. project charter:

The project charter is a formal document that requests official authorization to initiate the project. It serves as a reference point throughout the project, clearly stating its purpose and establishing a foundation for decision-making and project management.

Project title	**Government Document Management system**
Project start date	25 - 10 - 2025
Project end date	6 - 2 - 2026
Project supervisor	Dr.Riad sonbol Eng. Raghad alhossny
Project methodology	Scrum methodology

- **project objectives:**
  - Establish a secure digital system for managing government documents efficiently.
  - Digitize paper-based documents using OCR technology to extract searchable text.
  - Manage metadata to facilitate classification and organization of documents.
  - Provide advanced search mechanisms (textual and semantic) for fast and accurate information retrieval.
  - Enhance transparency and accountability through audit logs.
  - Generate periodic and analytical reports to support decision-making.

- Integrate with other government systems via APIs.
- Improve performance and scalability to meet the needs of large institutions.

- **Project Approach:**

The Government Document Management System will be developed gradually in phases using the Scrum methodology. Each phase focuses on specific priorities to ensure efficiency and scalability:

1. Account Management:

Develop secure login and registration for users (employees, administrators) with role-based permissions.

2. Document Management:

Enable uploading, editing, and deleting documents, linking them to user accounts.

3. Metadata Handling:

Add, update, and remove metadata to facilitate classification and organization of documents.

4. OCR Text Extraction:

Process scanned documents and images to extract text content for search and indexing.

5. Search Functionalities:

Provide both text-based and semantic search to improve accessibility and retrieval speed.

6. Reporting & Auditing:

Generate periodic reports and maintain audit logs to ensure transparency and accountability.

7. Integration & Expansion:

Connect with other government systems via APIs, support AI-based classification, and enhance scalability.

- **Stakeholders:**

stakeholder	Role	Responsibility
Dr.Riad sonbol	Project Manager & Supervisor	Guides the project and provides feedback.
Eng. Raghad alhossny	Project Manager & Supervisor	Guides the project and provides feedback.
Najat bostaty	Software Engineer	Design and implement user interfaces, connect fronted with backend.
Bushra alshaabani	Software Engineer	
Salam almasri	Software Engineer	

3.statement of work(sow):

The Statement of Work is a comprehensive document that defines the scope of work for the project, including specific tasks, deliverables, schedules, and responsibilities. It provides a clear understanding of what needs to be accomplished, the project's objectives, and success criteria.

3.1 project Description:

The Government Document Management System project aims to digitize and manage government documents securely and efficiently. The system provides functionalities for uploading, classifying, and retrieving documents, integrating OCR technology for text extraction, and offering advanced search capabilities. It also supports metadata management, audit logging, and reporting to enhance transparency and decision-making.

3.2 Project Scope:

The system is designed to:

-Digitize paper-based government records.

-Provide secure document storage and retrieval.

-Enable metadata management for classification.

-Support OCR-based text extraction.

-Offer advanced search (textual and semantic).

-Generate reports and maintain audit logs.

-Integrate with other government systems via APIs.

3.3 Project Objectives:

- Enhance transparency in government document management.
- Improve efficiency and reduce reliance on paper records.
- Provide secure access and role-based permissions.
- Support decision-making through accurate reporting.
- Ensure scalability for large institutions.

3.4 Deliverables:

-Project Plan

-SRS Document (Software Requirements Specification)

-Final Project Report

-Frontend and Backend Components of the Government Document Management system

3.5 project Requirements:

Technologies and Tools

- Backend Framework: .NET Core
- Frontend Framework: React.js
- Database: MongoDB
- OCR Tool: Optical Character Recognition service
- Development Tools: Visual Studio, Visual Studio Code, GitHub
- Testing Tools: Postman
- Project Management Tools: Jira

3.6 Assumptions:

- Continuous availability of project team members and stakeholders to provide feedback and guidance.

- Regular academic supervision and monitoring throughout the project phases.

- Access to required technologies and tools (.NET Core, React.js, MongoDB, OCR services).

- Stable internet connection and secure servers for hosting and collaboration.

- Commitment of all team members to follow Scrum methodology and deliver tasks on time.

3.7 project Resources:

Human Resources:

- Doctor Riad sonbol: Project manager.
- Engineer Raghad al hossny: Project manager.
- Bushra alshaabani: SE-Developer (Backend)
- Najat bostaty : SE-Developer (Frontend)
- Salam almasri: SE-Developer (Backend)

3.8 Schedule :

Project title	**Government Document Management system**
Project start date	25 - 10 - 2025
First seminar	
technical committee	
Project end date	6 - 2 - 2026
Final discussion	31 - 1 - 2026

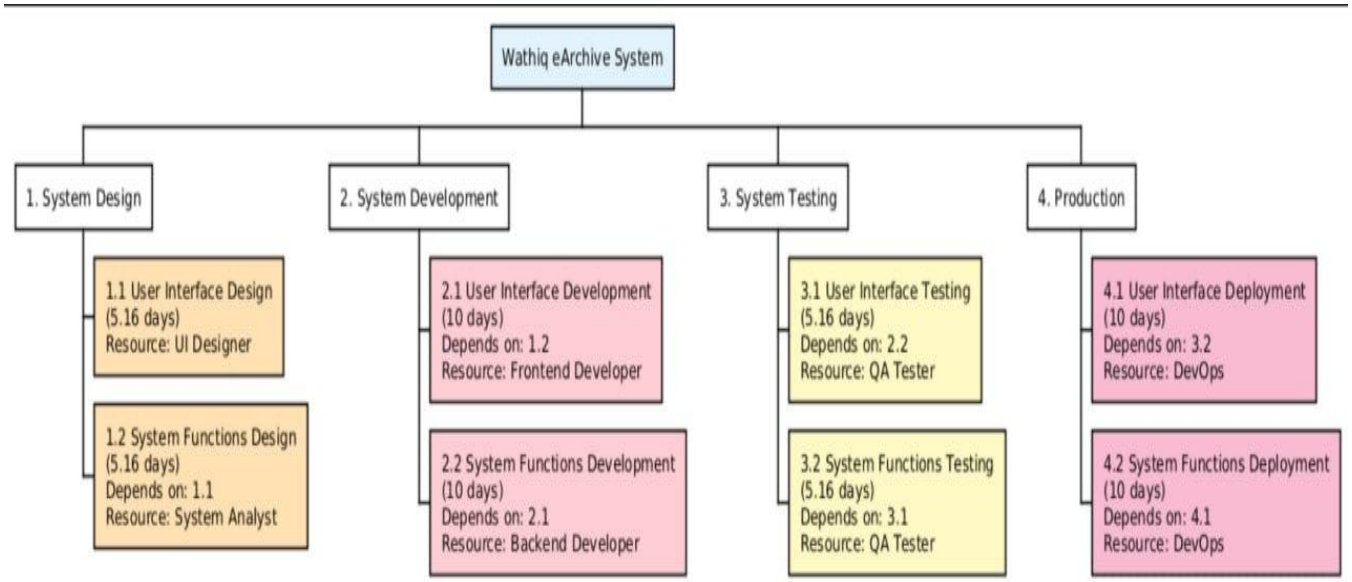
4. WBC "Work Breakdown Structure":

The WBS is a diagram known as the Work Breakdown Structure. It is used in project management to divide large-scale work into smaller, manageable sections that lead to better outcomes.

The WBS is usually represented in a hierarchical format that shows the sequential breakdown of activities and tasks within the project. The main project (parent project) is placed at the top, followed by levels of sub-elements branching from it. These are further divided into smaller, more detailed work elements until each task is clearly defined.

The WBS provides a comprehensive view of the project's structure and details. It helps clearly define the scope of work and assign specific responsibilities to each element in the project. It is also used to facilitate planning, organize resources, estimate costs, and monitor progress.

Work Breakdown Structure



5. Summary:

Effective project management is a fundamental pillar for developing successful software systems. It ensures that projects are completed within scope and schedule, while considering available resources. Strong project management also supports the delivery of high-quality software services that offer innovative solutions, achieve objectives, and meet stakeholder expectations. In this way, solid management bridges ambitious ideas with tangible results.

chapter3
Fundamental Concepts and Literature Review

1. Introduction:

This chapter presents a literature-based review aimed at analyzing systems similar to our project, Wathiq – The Governement Document Management System. It explores a selection of digital platforms that offer services partially or fully aligned with the objectives of our system. The goal is to identify key features and technologies used in these platforms, examine their strengths and weaknesses, and extract insights that can help improve and develop our own system.

2. Fundamental Concepts

This section provides definitions and explanations of the key concepts and terms related to our project:

• Electronic Archiving:

The process of converting paper-based or scanned documents into digital files that can be stored, retrieved, and processed. It aims to improve information accessibility and reduce reliance on manual archiving.

• Document Management:

Includes uploading, editing, deleting, and verifying documents, as well as organizing them and linking them to metadata for easier classification and retrieval.

• Metadata:

Additional information attached to a document, such as title, type, department, and creation date. Metadata improves search accuracy and speeds up access to documents.

• Optical Character Recognition (OCR):

A technology used to extract text from images and PDF files, making documents searchable and enabling text-based processing within the system.

• Audit Log:

A comprehensive record that tracks all user actions within the system, such as uploading, editing, and deleting documents. It enhances transparency and security.

• Role-Based Access Control:

A mechanism that defines what each user can do in the system based on their role (Admin, Manager, User), ensuring data protection and task organization.

• User Interface:

A flexible and user-friendly design that allows users to interact with the system efficiently, tailored to their specific roles and available functionalities.

3. Literature Review:

The purpose of this section is to analyze and evaluate various document management and digital archiving systems. This review aims to provide a comprehensive overview of existing platforms in the field, drawing insights from well-known solutions where available. By understanding current trends and identifying similarities and differences, the study helps inform the

development of the proposed system, Wathiq. The analysis includes comparisons of four systems and their key features:

• **OpenKM**

An open-source enterprise document management system used in both public and private sectors.

Advantages:

- Comprehensive document lifecycle management (upload, classification, archiving, versioning).
- Built-in workflow engine for process automation.
- OCR support for searchable text extraction.
- Integration with external systems (ERP, CRM, SharePoint).
- Role-based access control and encrypted storage.
- Collaboration tools including comments and notifications.
- Scalable and multi-platform compatibility.

Disadvantages:

- Complex installation requiring technical expertise.
- Outdated user interface compared to commercial systems.
- Performance issues with large repositories.

- Customization requires Java programming skills.

- No built-in advanced AI features.

Core Features:

OCR, Workflow, Metadata, Versioning, Audit Log, API, Multi-language support.

• Mayan EDMS

A free and open-source electronic document management system suitable for small to medium organizations.

Advantages:

- Fully open-source under GPL license.

- OCR integration via Tesseract.

- Advanced full-text search using Elasticsearch or Whoosh.

- Version control and audit trail.

- Fine-grained role-based access control.

- Modular and extensible architecture.

Disadvantages:

- Setup and configuration require technical knowledge.

- Basic and less intuitive user interface.

- No built-in AI classification or semantic search.

- Manual maintenance and updates.

- Resource-intensive for large-scale archives.

Core Features:

OCR, Metadata, Full-text search, Versioning, Workflow, API, Multi-language support.

• Laserfiche

A commercial enterprise content management system widely adopted by government and educational institutions.

Advantages:

- Visual workflow designer for complex automation.
- AI-powered document recognition and classification.
- Automatic metadata extraction.
- High security and compliance with international standards.
- Modern and user-friendly interface.
- Flexible cloud and on-premise deployment.
- Integration with Microsoft 365, SAP, and other platforms.
- Built-in analytics and reporting tools.

Disadvantages:

- High licensing and subscription costs.
- Advanced customization requires training or vendor support.
- Demands strong infrastructure for deployment.

- Closed-source limits deep customization.

- Learning curve for new users.

Core Features:

OCR, AI classification, Workflow, Metadata, Audit Log, Reporting, API, Mobile access.

• DocuWare

A commercial cloud-based document management and workflow automation system.

Advantages:

- Cloud-based and scalable infrastructure.

- AI-powered data capture and indexing.

- Drag-and-drop workflow designer.

- Strong security and regulatory compliance.

- Intuitive and responsive web interface.

- Native integration with Microsoft Office, SAP, and others.

- Mobile apps for remote access.

Disadvantages:

- Subscription-based licensing can be costly.

- Limited customization compared to open-source systems.

- Heavy reliance on stable internet connectivity.

- Complex setup for on-premise deployment.

- Closed-source architecture restricts internal modifications.

Core Features:

OCR, AI indexing, Workflow, Metadata, Versioning, Audit Log, API, Mobile access.

Summary

The Government Document Management System is designed to digitize governmental records using OCR and structured metadata, ensuring secure access and transparency through audit logs.

From the literature review, OpenKM and Mayan EDMS are closest in terms of open-source architecture and core functionalities, while Laserfiche and DocuWare provide advanced features such as workflow automation, security, and enterprise integration.

This analysis highlights the strengths and limitations of existing systems, guiding the development of the Government Document Management System as a secure, efficient, and user-friendly solution tailored to public sector needs without the high costs of commercial platforms.

Below is a summary of the key services offered by the proposed Government Document Management System.

Table 1: A brief analysis of similar systems and the target system.

System Feature	Mayan EDMS	OpenKM	DocuWare	Laserfiche	Our System
OCR Support	✓	✓	✓	✓	✓
AI-Based Classification	✗	✗	Limited	✓	✓*
Workflow Automation	Basic	✓	✓	✓	✓
Metadata & Search	✓	✓	✓	✓	✓*
Semantic / AI Search	✗	✗	Limited	✓	✓*

Security & Access Control	✓	✓	✓	✓	✓
Integration APIs	✓	✓	✓	✓	✓
Analytics & Reporting	✗	**Limited**	✓	✓	✓ *
Multi-Language Support	✓	✓	✓	✓	✓ *
User-Friendly Interface	**Basic**	**Basic**	✓	✓	✓

(*) means features that will be achieved in the future, outside the scope of this semester.

Basic → Feature exists but with limited or simple functionality.

Limited → Partially supported or requires external integration

chapter4
System Analysis

1. Introduction

This chapter focuses on the detailed analysis of the Government Document Management System. The main objective is to study the needs of public institutions and the administrative context in which they operate, helping to identify the essential requirements and core functionalities that the proposed system must include. The chapter explores mechanisms for digital archiving, document management, and access control, along with a comprehensive analysis of use cases and workflows. This provides a clear understanding of how users interact with the system and how their administrative goals can be achieved through it. This analysis forms the foundation for designing and developing an effective software solution tailored to governmental document management.

2. Software Requirement Specification (SRS):

The Software Requirement Specification is a fundamental document in the development process of the Government Document Management System. It is used to describe the functional and non-functional requirements of the proposed system in a clear and structured manner.

The specification includes the following key elements:

1. System Description: The system aims to digitize, organize, and manage governmental documents within a secure institutional environment, supporting search, classification, and workflow.

2. Inputs and Outputs: Inputs include digital or scanned files, while outputs consist of searchable archived documents, administrative reports, and audit logs.

3. Performance Requirements: The system must process documents efficiently and support a large number of users without delays.

4. Interface Requirements: The system should provide integration interfaces with other governmental platforms, along with a simple user interface tailored to different roles.

5. Security Requirements: The system must offer access control mechanisms, data encryption, and activity logging to ensure document integrity.

6. Maintenance Requirements: The system should be upgradable and extensible, allowing future enhancements without compromising core performance.

The SRS document facilitates effective communication between the development team and stakeholders, ensuring a shared understanding of the system's requirements. It also serves as a reference throughout the development process to verify that all specifications are properly implemented, contributing to the system's success and institutional alignment.

3. Software Requirement Specification(SRS) Version 0.1:

Date	Change Reason	Version
20/11/2025	Initial document creation	0.1
25/11/2025	Added login and two-factor authentication requirements	0.2
2/12/2025	Modified access control requirements	0.3
10/12/2025	Updated integration requirements with governmental systems	0.4

1. Introduction

1.1 Purpose

This document defines the requirements of the Government Document Management System, which is designed to digitize paper-based and scanned documents and convert them into searchable and processable digital data. The system enables governmental institutions to organize documents, improve accessibility, and ensure security and transparency through intelligent tools based on OCR and artificial intelligence technologies.

1.2 Project Scope

The Government Document Management System is designed to provide a secure and flexible institutional platform that allows governmental entities to upload, classify, and automatically extract metadata from documents. It offers multi-level user interfaces and a comprehensive audit log. The system ensures easy access to documents and reduces reliance on manual archiving, contributing to improved administrative efficiency.

1.2.1 High-Level Requirements

- Account Management: This function allows users to log in using email or username and password, securely log out, and supports two-factor authentication (2FA) for sensitive users.

- User Management: The system enables the creation of multiple roles and permissions (System Admin, Directorate, Department, Employee), defining what each role can perform regarding document operations.

- Document Management: Users can upload documents in various formats, edit metadata, delete or archive documents based on permissions, and integrate with scanning devices.

- Content Extraction: The system automatically performs OCR to extract text from images and PDF files, supporting Arabic and English with automatic language detection.

- Document Classification: The system applies intelligent classification algorithms to identify document types (e.g., decision, correspondence, invoice, report) and suggest appropriate folders or departments for archiving.

- Metadata Management: The system extracts key metadata from the text such as document number, date, sender/receiver, names, and signatures, with manual editing available when needed.

- Search and Navigation: The system provides full-text search and semantic search using AI, along with filter-based search (type, date, department, source) and result sorting by most recent.

- Document Lifecycle Management: The system supports various document states (Draft, Under Review, Archived, To Be Deleted), with secure trash handling before permanent deletion.

- Audit Logging: All user actions on documents are recorded to ensure transparency and traceability.

- External System Integration: The system can send and receive documents from other governmental platforms such as HR or correspondence systems, and supports data import/export in JSON and XML formats.

1.2.2 Actors:

- User: The user is the individual responsible for uploading documents to the system, whether a department employee or directorate staff. They can upload files, edit metadata, delete or archive documents based on permissions, and search using full-text or semantic tools. The user can also track the document's lifecycle status and download either the original or OCR version.

- Manager: The manager is a user with extended permissions, such as reviewing uploaded documents, approving or rejecting them, managing user permissions within their department or directorate, and accessing activity

reports and audit logs. They can also adjust suggested classifications and route documents to appropriate sections.

- System Admin: The system administrator holds the highest level of access. They configure the system, create roles and assign permissions, manage integration settings with external governmental platforms, and monitor overall system performance. The admin also handles backup operations and ensures compliance with security and reliability standards.

2. General Description

2.1 Product Perspective

The Government Document Management System described in this document is an intelligent and innovative solution that offers a range of benefits to governmental institutions. It provides a secure and flexible digital environment that enables entities to upload, classify, and automatically extract textual content from documents using OCR and artificial intelligence technologies. The system enhances administrative efficiency, promotes transparency through a comprehensive audit log, facilitates easy access to documents, and reduces reliance on manual archiving—supporting digital transformation in the public sector in an effective and organized manner.

2.2 Product Features

- Account Management:

Login and logout using email or username and password, with support for two-factor authentication (2FA) for sensitive users (for all users).

- Document Management:

Upload documents in multiple formats (PDF, images), edit metadata, delete or archive documents based on permissions, and download either the original or OCR version (for users).

Integration with scanning devices to import documents directly (for users).

Automatic OCR to extract text from images and PDFs, supporting Arabic and English with automatic language detection (for users).

- Document Classification:

Apply intelligent classification algorithms to identify document types (e.g., decision, correspondence, invoice, report) and suggest appropriate folders or departments for archiving (for users).

- Metadata Management:

Extract key metadata from the text such as document number, date, source, names, and signatures, with manual editing available when needed (for users).

- Search and Navigation:

Full-text search within document content and semantic search using AI, along with filter-based search and result sorting by most recent (for users).

- Document Lifecycle Management:

Track document status across stages (Draft, Under Review, Archived, To Be Deleted), with secure trash handling before permanent deletion (for users and managers).

- Request and Review Management:

Review uploaded documents, approve or reject them, and route them to appropriate departments (for managers).

- System Administration:

Create roles and assign permissions, configure integration with external governmental systems, manage backups, and monitor overall system performance (for system admin).

2.3 User Categories and Their Characteristics

2.3.1 User

- Responsibilities: Upload documents to the system, edit metadata, delete or archive documents based on permissions, and search using full-text or semantic tools. The user can also track the document's lifecycle status and download either the original or OCR version.

- Privileges: Upload documents, edit metadata, use search and classification tools, download files, and view document status.

2.3.2 Manager

- Responsibilities: Review documents uploaded by users, approve or reject them, manage user permissions within the department or directorate, and route documents to appropriate sections.

- Privileges: Approve documents, adjust classifications, access audit logs, and manage user permissions.

2.3.3 System Admin

- Responsibilities: Configure the system, create roles and assign permissions, set up integration with external governmental platforms, manage backups, and monitor overall system performance.

- Privileges: Full control over system settings, user management, performance monitoring, and ensuring security and reliability.

3. System Features

3.1 Functional Requirements

3.1.1 Account Management

- Requirement 1.1: The system allows users to log in using username or email and password.

- Requirement 1.2: The system allows users to log out securely.

- Requirement 1.3: The system supports two-factor authentication (2FA) for users with sensitive access.

3.1.2 Document Management

- Requirement 2.1: The system allows users to upload documents in multiple formats (PDF, images).

- Requirement 2.2: The system allows users to edit document metadata.

- Requirement 2.3: The system allows users to delete or archive documents based on permissions.

- Requirement 2.4: The system allows users to download the original or OCR version of the document.

- Requirement 2.5: The system integrates with scanning devices to import paper documents.

3.1.3 Content Extraction

- Requirement 3.1: The system performs automatic OCR to extract text from documents.

- Requirement 3.2: The system supports Arabic and English with automatic language detection.

- Requirement 3.3: The system extracts key metadata such as document number, date, source, and signatures.

3.1.4 Classification and Search

- Requirement 4.1: The system classifies documents by type using intelligent algorithms.

- Requirement 4.2: The system suggests appropriate folders or departments for archiving.

- Requirement 4.3: The system enables full-text search within document content.

- Requirement 4.4: The system enables semantic search using artificial intelligence.

- Requirement 4.5: The system supports filter-based search (type, date, department, source).

- Requirement 4.6: The system allows sorting search results by most recent.

3.1.5 Document Lifecycle Management

- Requirement 5.1: The system supports various document states (Draft, Under Review, Archived, To Be Deleted).

- **Requirement 5.2:** The system moves documents to a secure trash before permanent deletion.

3.1.6 Audit Logging

- **Requirement 6.1:** The system records all user actions on documents.

3.1.7 System Administration

- **Requirement 7.1:** The system allows the admin to create roles and assign permissions.

- **Requirement 7.2:** The system allows configuration of integration with external governmental systems.

- **Requirement 7.3:** The system allows backup management and performance monitoring.

3.2 Non-Functional Requirements

- **Scalability**

  - The system must be scalable to support an increasing number of users without performance degradation.

  - The system must support adding new languages easily without deep structural changes.

- **Availability**

  - The system must maintain operational readiness to provide continuous document management services.

- **Security**

  - Data must be encrypted during transmission and storage.

  - Passwords must be stored securely using encryption.

- **Usability**

  - The user interface must be simple, responsive, and easy to understand for all user categories.

- **Maintainability**

  - The code must be modular and well-documented.

  - The system must support future updates without affecting core functionalities.

4. System Requirements

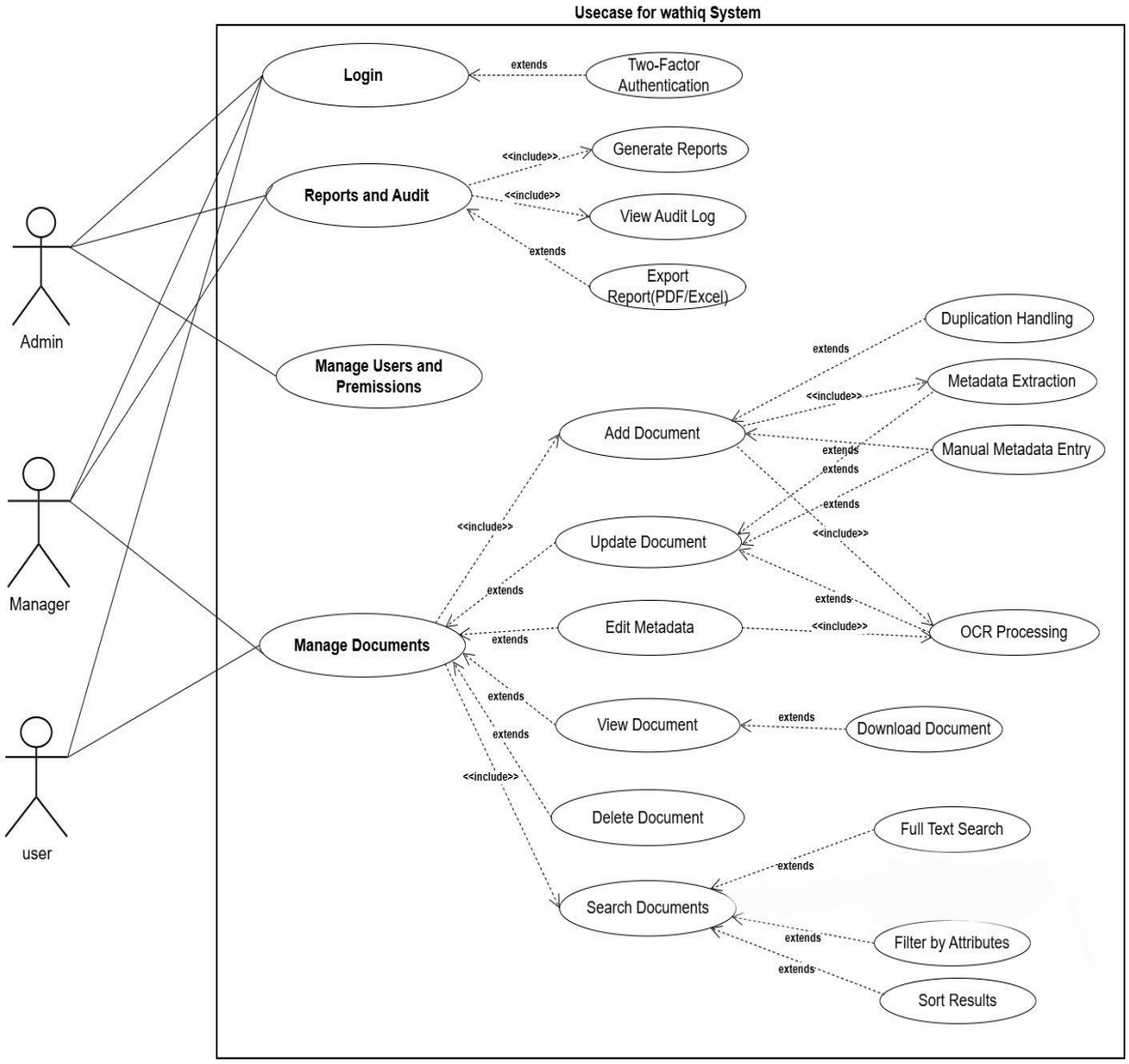
		req tilte	description	category
2	RE-FR-1	يمكن للمستخدم رفع وثائق بصيغ مختلفة (PDF، صور...).		Functional
3	RE-FR-2	يتم تخزين الوثائق في نظام تخزين آمن		Functional
4	RE-FR-3	يمكن للمستخدم تعديل بيانات الوثيقة (العنوان، القسم، النوع...)		Functional
5	RE-FR-4	يمكن للمستخدم حذف أو أرشفة الوثيقة نهائيًا حسب الصلاحيات		Functional
6	RE-FR-5	يقوم النظام تلقائيًا بعملية OCR لاستخراج النص من الصورة أو PDF		Functional
7	RE-FR-6	يدعم النظام اللغة العربية والإنجليزية مع إمكانية اكتشاف اللغة تلقائيًا		Functional
8	RE-FR-7	يطبّق النظام خوارزمية تصنيف تلقائي لتحديد نوع الوثيقة (مثل: قرار – مراسلة – فاتورة – هوية – تقرير...).		Functional
9	RE-FR-8	ستخرج النظام البيانات الأساسية (Metadata) من النص مثل: (رقم الوثيقة / رقم الصادر أو الوارد-تاريخ الإصدار-الجهة المرسلة أو المستقبلة-الأسماء والتواقيع.....)		Functional
10	RE-FR-9	يمكن للنظام اكتشاف التكرار أو الوثائق المتطابقة		Functional
11	RE-FR-10	يمكن للنظام اقتراح مجلد أو قسم أرشفة مناسب بناءً على محتوى الوثيقة.		Functional
12	RE-FR-11	يمكن للمستخدم البحث بالكلمات داخل النص المستخرج (Full Text Search).		Functional
13	RE-FR-12	يمكن للمستخدم البحث حسب فلاتر (نوع الوثيقة، التاريخ، القسم، الجهة...).		Functional
14	RE-FR-13	يقدم النظام إمكانية فرز النتائج حسب الأحدث		Functional
15	RE-FR-14	يمكن للمستخدم تنزيل الوثيقة الأصلية أو نسخة OCR منها حسب الصلاحية.		Functional
16	RE-FR-15	يمكن للمستخدم تسجيل دخول بواسطة البريد أو اسم المستخدم وكلمة المرور.		Functional
17	RE-FR-16	يدعم النظام التحقق الثنائي (2FA) للمستخدمين الحساسين.		Functional
18	RE-FR-17	يمكن النظام من إنشاء أدوار وصلاحيات (مدير النظام – مديرية – قسم – موظف) و تحديد ما يمكن لكل دور فعله (رفع، تعديل، حذف، اعتماد...)		Functional
19	RE-FR-18	كل عملية يقوم بها المستخدم تُسجّل في سجل التدقيق (Audit Log)		Functional
20	RE-FR-19	إدارة دورة حياة الوثيقة عبر كل حالاتها :(Draft-Under Review-Archived-To Be Deleted)و عند الإتلاف، تُنقل الوثيقة إلى سلة محذوفات آمنة قبل الحذف التام.		Functional
21	RE-FR-20	يقدم النظام تقارير بعدد الوثائق حسب النوع، القسم، السنة، المستخدم و تقارير نشاط المستخدمين خلال فترة محددة.		Functional
22	RE-FR-21	يمكن للمستخدم إعادة تعيين كلمة المرور		Functional
23	RE-FR-22	يدعم للمستخدم تنزيل الوثيقة حسب الصلاحية		Functional
24	RE-NF-23	يدعم النظام النسخ تلقائي اليومي للملفات وقاعدة البيانات و إمكانية استرجاع الوثائق من النسخ الاحتياطية بسهولة.		Functional
25	RE-NF-24	يجب ألا تتجاوز الاستجابة 5 ثوانٍ و النظام قادر على معالجة 10 مستخدمين متزامنين على الأقل دون بطء ملحوظ.	performance	non-Functional
26	RE-NF-25	النظام يجب أن يحقق نسبة تشغيل 99% ≥ (Uptime).	Reliability	non-Functional
27	RE-NF-26	واجهة بسيطة وسهلة لموظفين غير تقنيين.	Usability	non-Functional
28	RE-NF-27	تصميم متجاوب (Responsive) لجميع الأجهزة.	Usability	non-Functional

5. Requirements Modeling

5.1 Use Case Diagram

Use case diagrams represent the behavior of the system and help in capturing system requirements.

High level – Use Case Diagram



5.2 System Features _Use Case Description

- **login (UC-01) - use case specification :**

Name:	**Login**
Actors:	User, Manager, Admin
Pre-conditions:	User is not logged in
Main Flow:	**1.** User enters username/email and password. **2.** System validates credentials. **3.** System grants access.
Alternative Flows:	- extend Two-Factor Authentication: if enabled, system requests additional code. - Invalid credentials → error message.
Post-conditions:	User is logged in with an active session.

- **Search Documents (UC-02) - use case specification :**

Name:	**Search Documents**
Actors:	User, Manager
Pre-conditions:	User is logged in.
Main Flow:	**1.** User enters keyword. **2.** <<include>> Full Text Search. **3.** <<include>> Semantic Search. **4.** <<include>> Filter by Attributes. **5.** <<include>> Sort Results.
Alternative Flows:	- None.
Post-conditions:	Search results are displayed.

- **Add User (UC-0 3) - use case specification :**

1. Use Case Specification: Add User	
Name	Add New User
Actors	Admin
Pre-conditions	The Admin is logged in and has the necessary permissions to add users.
Main Flow	1. The Admin enters the new user's data (username, password, role). 2. The system sends an Add User request to the Controller. 3. The system validates the data and adds the user record to the database. 4. A success confirmation is returned to the UI.
Alternative Flows	Invalid Data Entry: If the entered data is invalid (e.g., duplicate username), the system displays an error message.
Post-conditions	A new user record is created in the database, and a success message is displayed to the Admin.

- **View Document (UC-04) - use case specification :**

2. Use Case Specification: View Document	
Name	View Specific Document Content
Actors	User
Pre-conditions	The User is logged in and has permission to access the document.
Main Flow	1. The User clicks on "View Document". 2. The system sends a request to fetch the document by its ID. 3. The system retrieves the document data from the database. 4. The system renders and displays the document content to the User.
Alternative Flows	Document Not Found: If the document is not found, the system displays a "Document not found" message.
Post-conditions	The document content is displayed to the User on the UI.

- **Delete Document (UC-05) - use case specification :**

3. Use Case Specification: Delete Document	
Name	Delete Specific Document from the System
Actors	User
Pre-conditions	The User is logged in and has the necessary permission to delete the document.
Main Flow	1. The User clicks on "Delete Document". 2. The system sends a delete request with the document ID. 3. The system verifies permissions and removes the document record from the database. 4. The system displays a "Document deleted successfully" message.
Alternative Flows	Permission Failure: If the User does not have delete permission, the system displays a "Permission denied" message.
Post-conditions	The document record is deleted from the database, and a success message is displayed to the User.

- **Edit Document (UC-06) - use case specification :**

4. Use Case Specification: Edit Document	
Name	Edit Existing Document Content and Metadata
Actors	User
Pre-conditions	The User is logged in and has the necessary permission to edit the document.
Main Flow	1. The User clicks on "Edit Document" and enters the new data. 2. The system sends the edit request to the DocumentController. 3. The system updates the document fields in the database. 4. The system displays "Document updated successfully" and then displays the updated document.
Alternative Flows	Update Failure: If the data update fails in the database, the system displays an appropriate error message.
Post-conditions	The document record is updated in the database, and the updated content is displayed to the User.

- **Check Permissions (UC-07) - use case specification :**

6. Use Case Specification: Check Permissions	
Name	Check User Permission for a Specific Action
Actors	User
Pre-conditions	The User is logged in and attempting an action that requires permission.
Main Flow	1. The User requests an action (e.g., edit document). 2. The system sends a permission check request to the AuthController. 3. The system retrieves user roles and document ownership from the database. 4. The result (granted or denied) is returned to the UI.
Alternative Flows	Check Failure: If an error occurs during the check process, the action is denied by default.
Post-conditions	The result of the check (allowed or denied) is determined and displayed to the User.

- **Reset Password (UC-08) - use case specification :**

7. Use Case Specification: Reset Password	
Name	Reset Forgotten Password
Actors	User
Pre-conditions	The User has a registered email address in the system.
Main Flow	1. The User requests a password reset. 2. A verification code is generated and sent to the User's email. 3. The User enters the verification code in the UI. 4. The code is validated by the system. 5. The User enters the new password, which is then updated in the database. 6. The system displays a success message.
Alternative Flows	Invalid/Expired Code: The system displays an error message and prompts the User to try again.
Post-conditions	The password is updated in the database, and a success message is displayed to the User.

- **Upload Document (UC-09) - use case specification :**

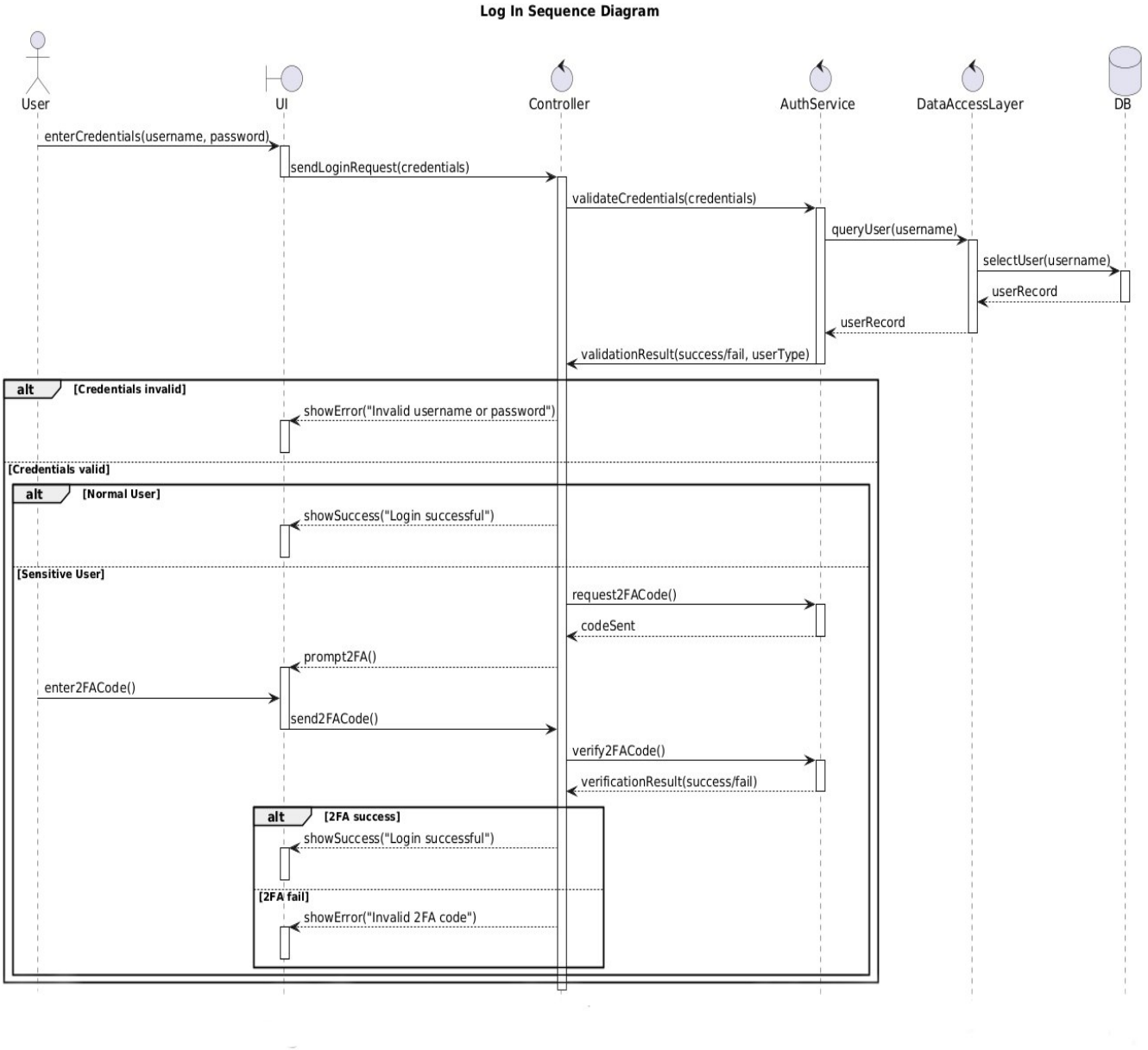
Name:	**Upload Document**
Actors:	User, Manager
Pre-conditions:	User is logged in and has upload permission
Main Flow:	**1.** User selects a file (PDF/Image). **2.** System stores the document securely. **3.** <<include>> OCR Processing. **4.** <<include>> Metadata Extraction. **5.** <<include>> Document Classification. **6.** <<include>> Select Folder.
Alternative Flows:	- Manual Metadata Entry: if OCR fails. - Duplicate Detection: if system detects duplicate.
Post-conditions:	Document is stored, indexed, and searchable.

- **Generate Reports (UC-10) - use case specification :**

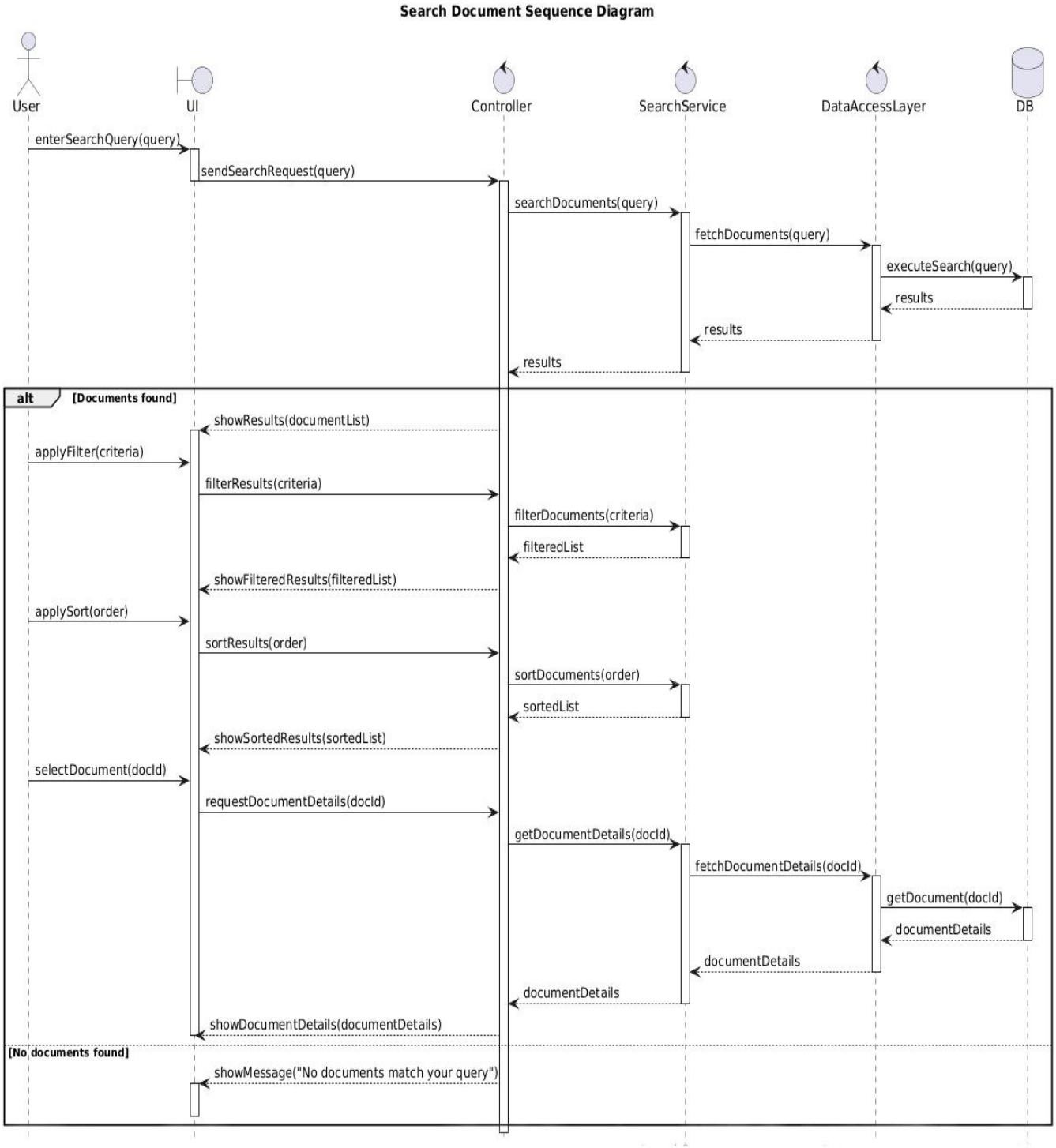
5. Use Case Specification: Generate Reports	
Name	Generate and Display System Reports and Statistics
Actors	User
Pre-conditions	The User is logged in and has access to the Reports page.
Main Flow	1. The User opens the Reports Page. 2. The UI automatically requests the system report. 3. The system fetches statistics (documents, users, activity) from the database via the Data Access Layer. 4. The system displays the numbers and charts to the User.
Alternative Flows	Database Connection Failure: If data retrieval fails, the system displays a "Could not load reports" message.
Post-conditions	The system statistics and charts are displayed on the UI.

5.3 Sequence Diagrams

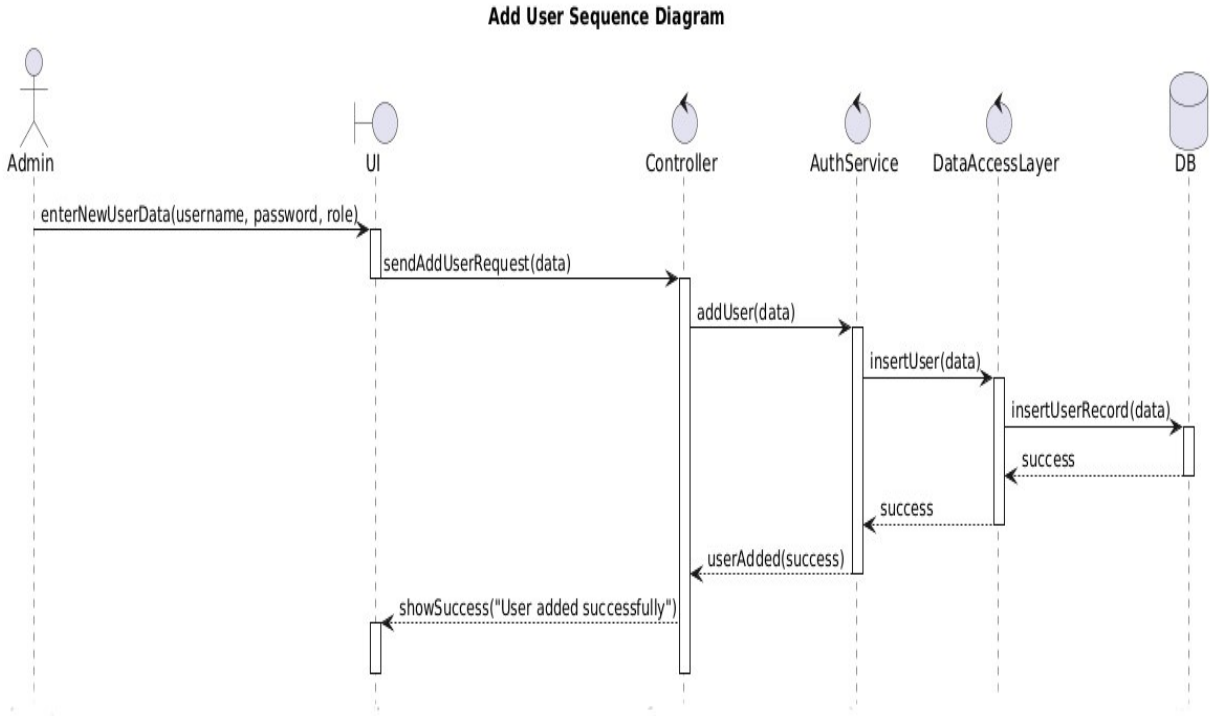
- **Login (UC- 01) – Sequence diagram:**



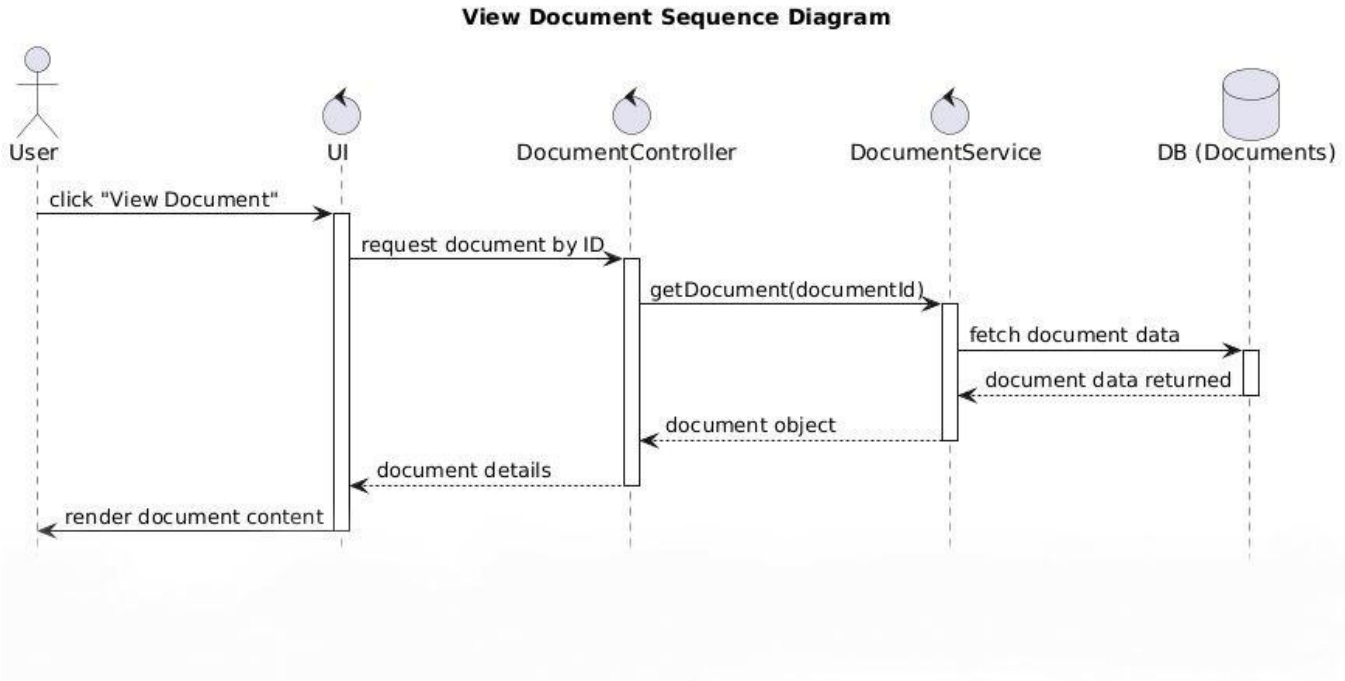
- **Search Document (UC- 02) – Sequence diagram:**



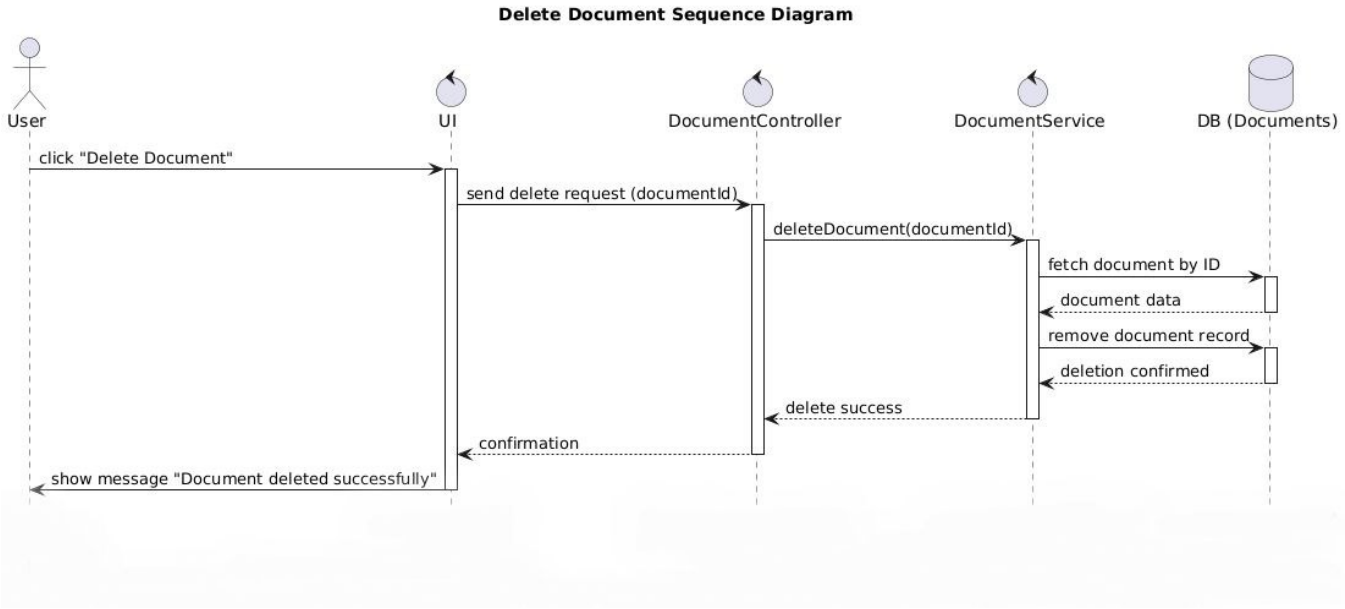
- **add user (UC- 03) – Sequence diagram:**



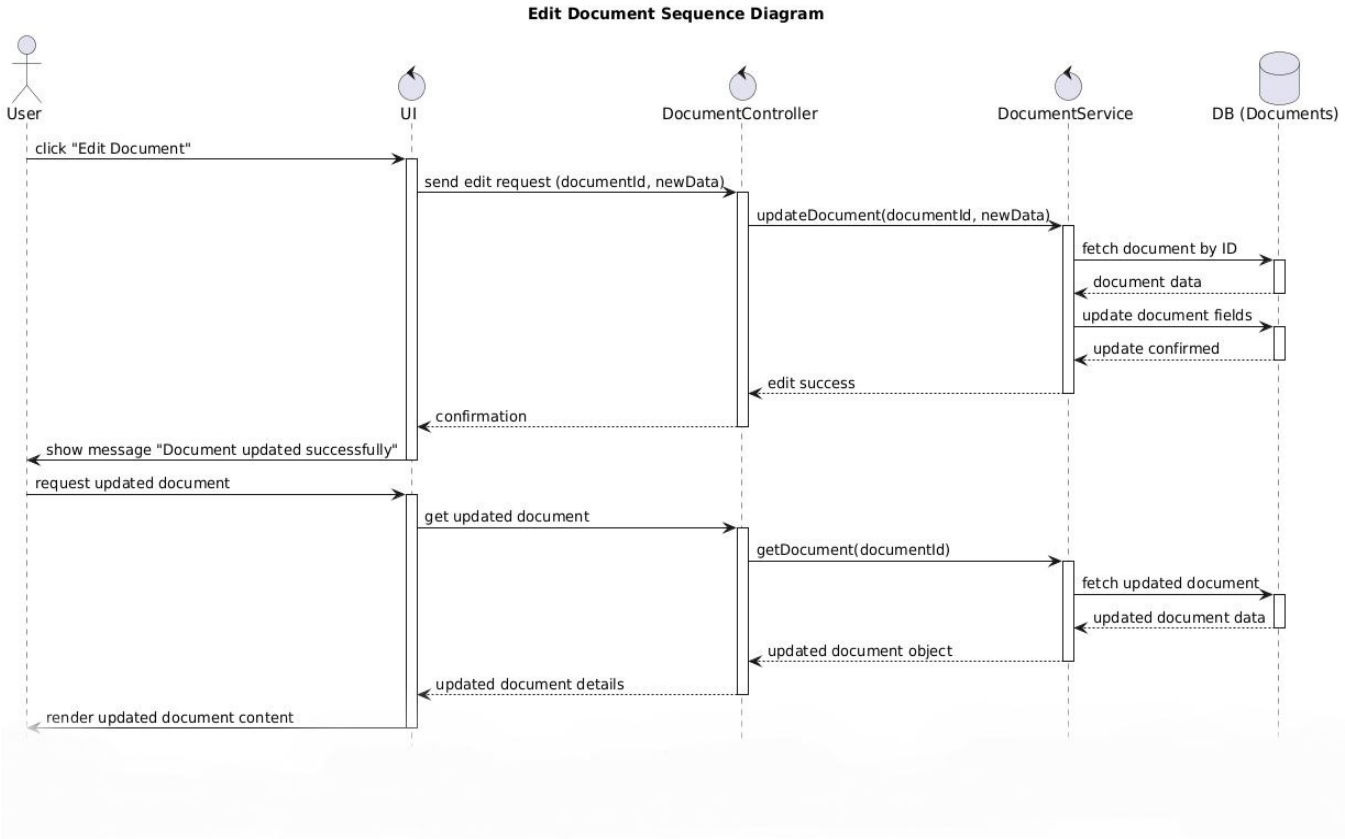
- **View Document (UC- 04) – Sequence diagram:**



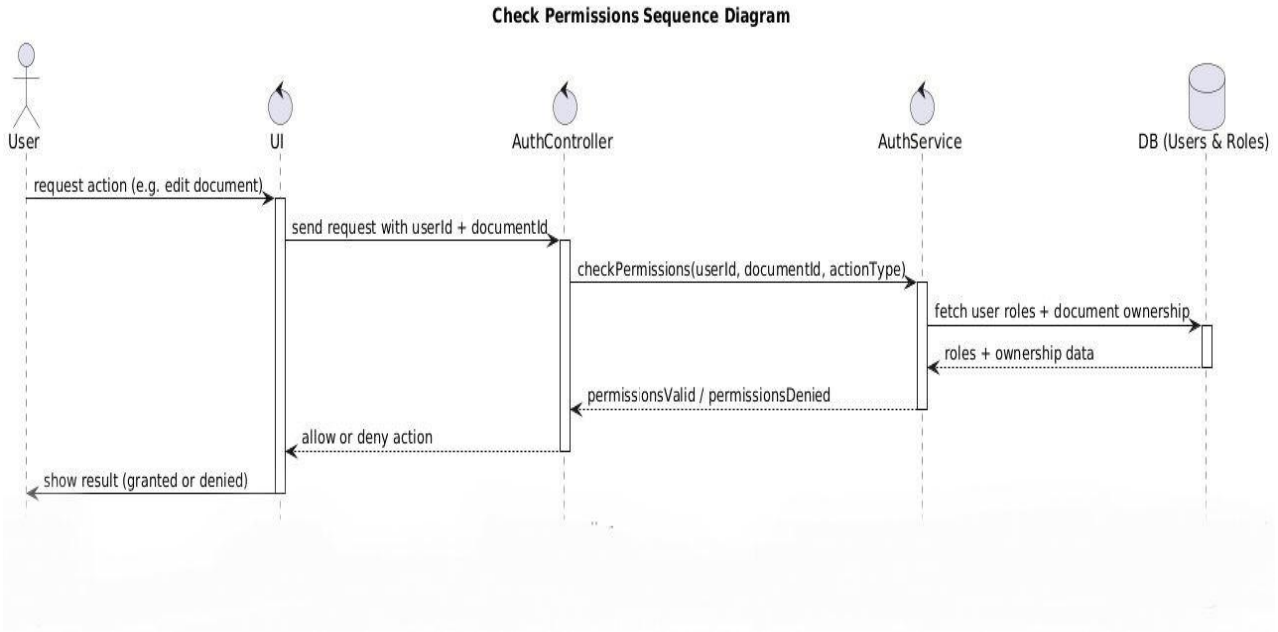
- **Delete Document (UC- 05) – Sequence diagram:**



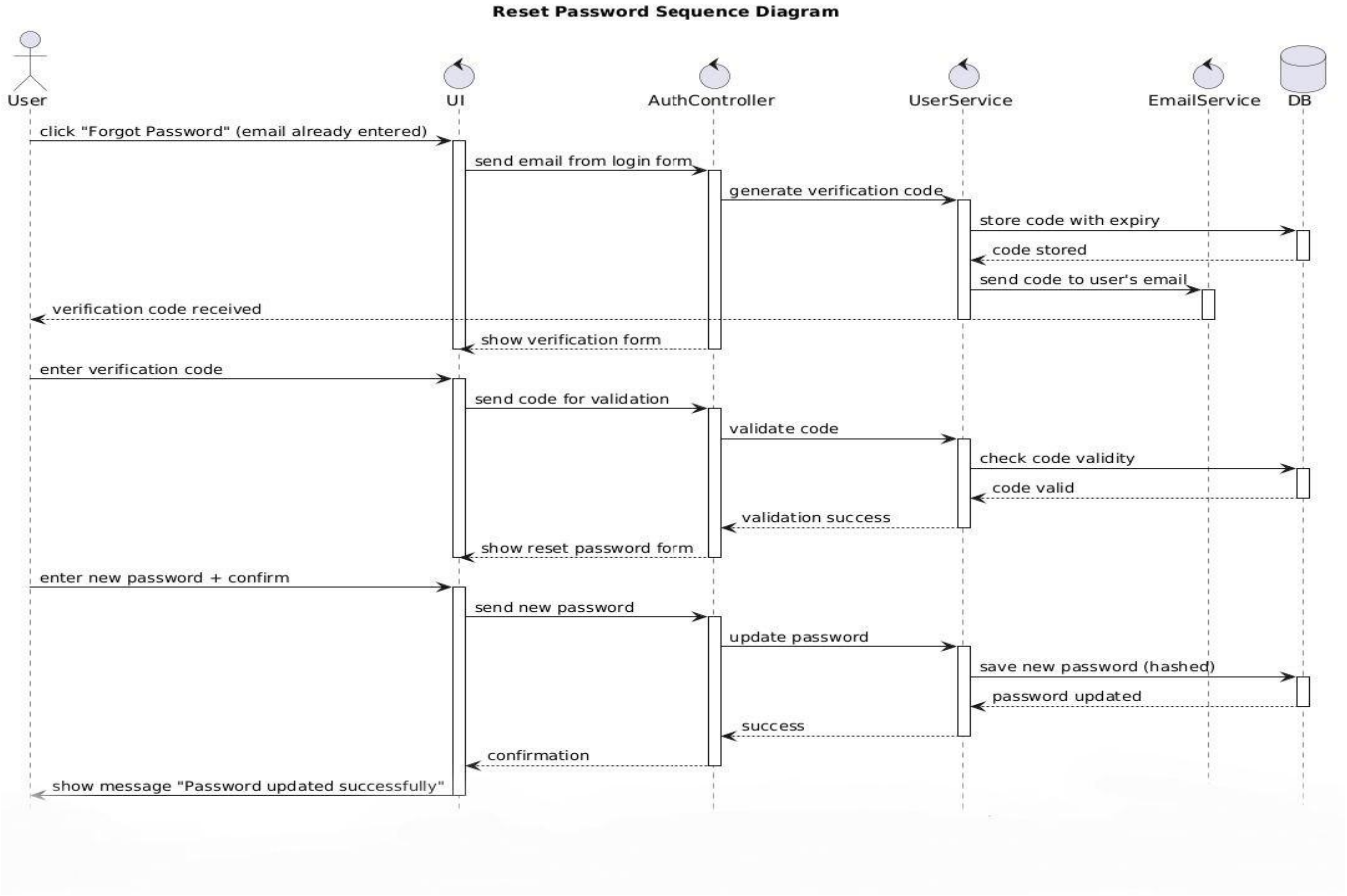
- **Edit Document (UC- 06) – Sequence diagram:**



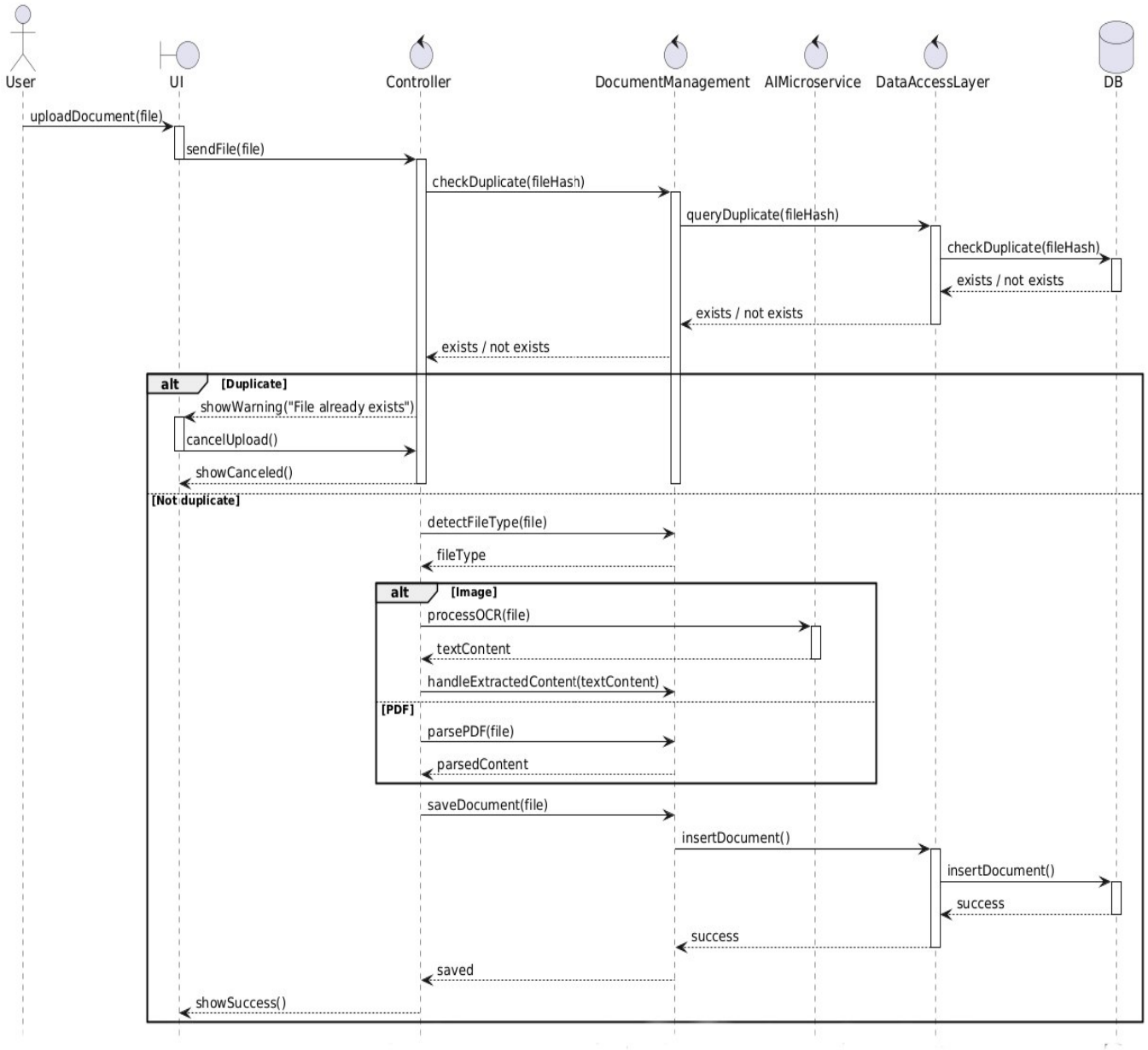
- **Check Permissions (UC- 07) – Sequence diagram:**



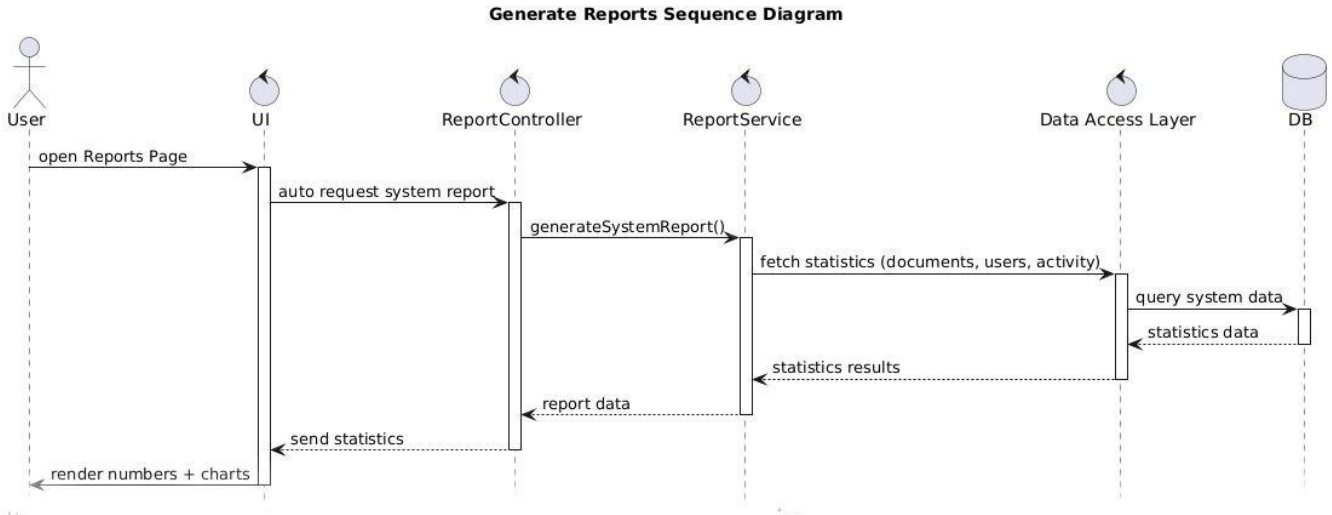
- **Reset Password (UC- 08) – Sequence diagram:**



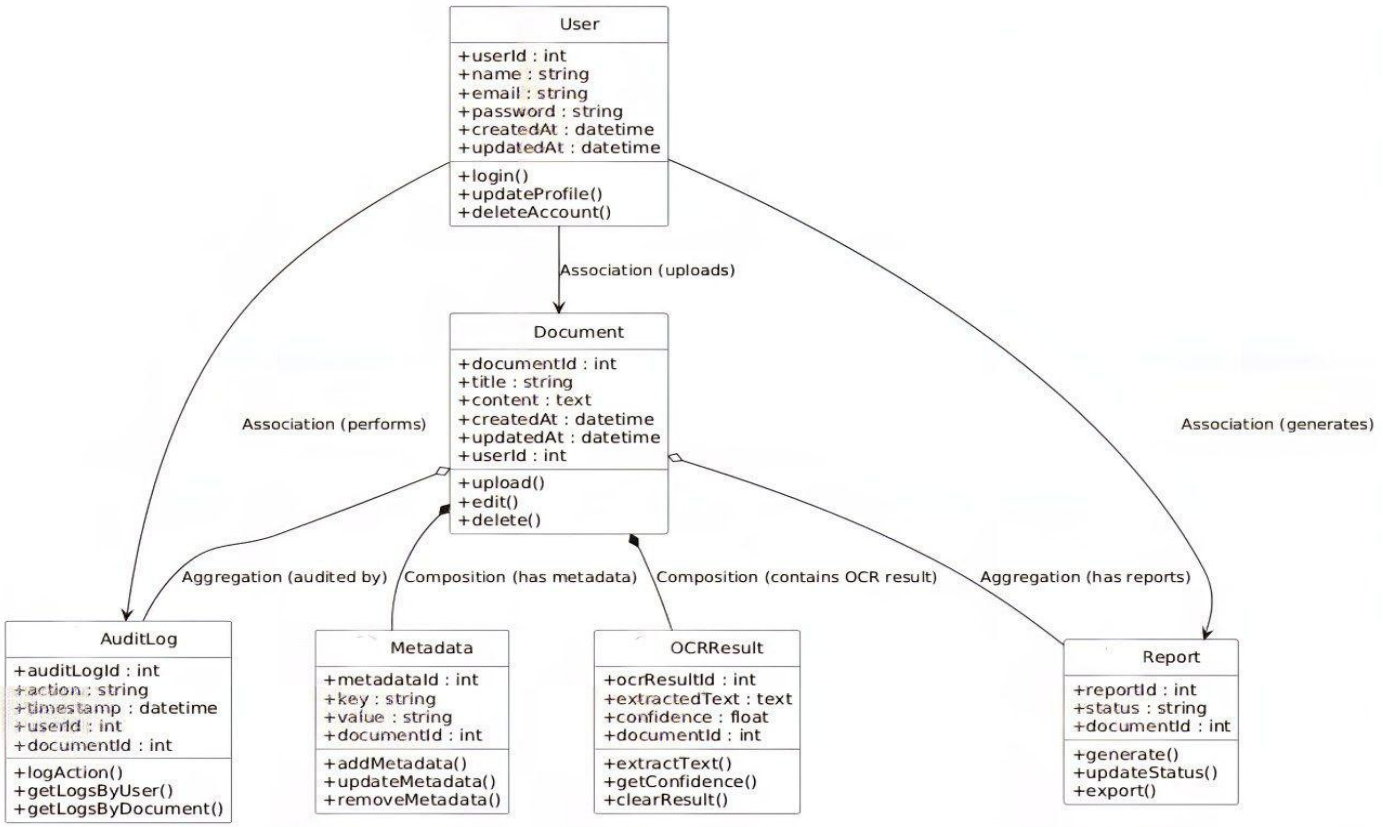
- **Upload Document (UC- 09) – Sequence diagram:**



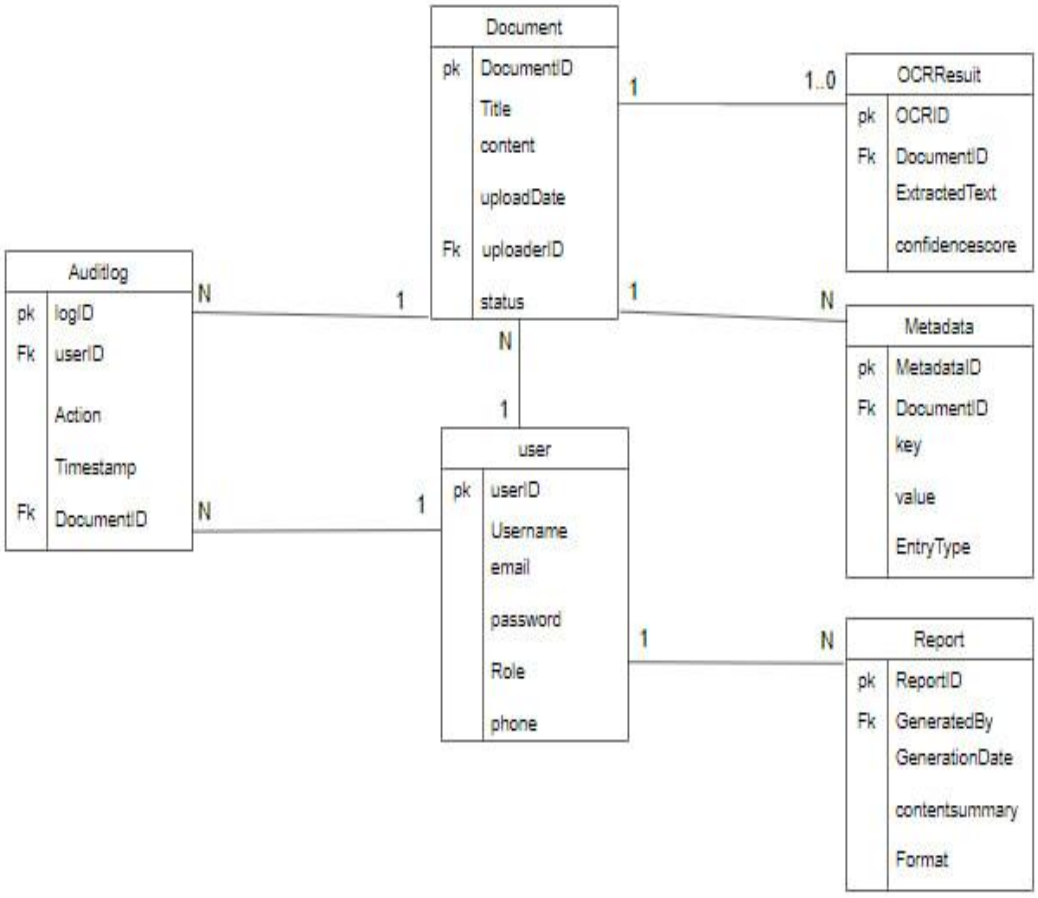
- **Generate General Report (UC- 10) – Sequence diagram:**



5.4 Analysis class diagram



5.5 ERD Diagrams



6. Initial test cases

Key Test Cases (1-15)

ID	Module	Description	Priority
AUTH-001	Auth	Login with valid credentials	P0
AUTH-002	Auth	Login with invalid credentials	P0
AUTH-2FA-003	2FA	Login requires 2FA when enabled	P0
AUTH-2FA-004	2FA	Verify correct 2FA code	P0
RBAC-001	RBAC	User cannot access admin pages	P0
RBAC-002	RBAC	User cannot delete other users' documents	P0
USER-001	Users	Admin creates new user	P0
USER-002	Users	Prevent duplicate email	P0
DOC-UP-001	Documents	Upload valid PDF document	P0
DOC-UP-003	Documents	Reject unsupported file type	P0
DOC-OP-003	Documents	Download document with permission	P0
DOC-OP-006	Documents	Delete document with permission	P0
META-001	Metadata	Add metadata to document	P0
SRCH-001	Search	Search documents by keyword	P0
SRCH-008	Search	Prevent search injection	P0

Key Test Cases (16-30)

ID	Module	Description	Priority
DASH-001	Dashboard	Dashboard totals match actual data	P1
RPT-002	Reports	Export PDF report	P0
RPT-003	Reports	Export Excel report	P0
AUD-001	Audit	Audit log records upload	P1
SEC-001	Security	API rejects requests without token	P0
AUTH-001	Auth	Login with valid credentials	P0
AUTH-002	Auth	Login with invalid credentials	P0
AUTH-2FA-003	2FA	Login requires 2FA when enabled	P0
AUTH-2FA-004	2FA	Verify correct 2FA code	P0
RBAC-001	RBAC	User cannot access admin pages	P0
RBAC-002	RBAC	User cannot delete other users' documents	P0
USER-001	Users	Admin creates new user	P0
USER-002	Users	Prevent duplicate email	P0
DOC-UP-001	Documents	Upload valid PDF document	P0
DOC-UP-003	Documents	Reject unsupported file type	P0

Key Test Cases (31-45)

ID	Module	Description	Priority
DOC-OP-003	Documents	Download document with permission	P0
DOC-OP-006	Documents	Delete document with permission	P0
META-001	Metadata	Add metadata to document	P0
SRCH-001	Search	Search documents by keyword	P0
SRCH-008	Search	Prevent search injection	P0
DASH-001	Dashboard	Dashboard totals match actual data	P1
RPT-002	Reports	Export PDF report	P0
RPT-003	Reports	Export Excel report	P0
AUD-001	Audit	Audit log records upload	P1
SEC-001	Security	API rejects requests without token	P0
AUTH-001	Auth	Login with valid credentials	P0
AUTH-002	Auth	Login with invalid credentials	P0
AUTH-2FA-003	2FA	Login requires 2FA when enabled	P0
AUTH-2FA-004	2FA	Verify correct 2FA code	P0
RBAC-001	RBAC	User cannot access admin pages	P0

Key Test Cases (46-60)

ID	Module	Description	Priority
RBAC-002	RBAC	User cannot delete other users' documents	P0
USER-001	Users	Admin creates new user	P0
USER-002	Users	Prevent duplicate email	P0
DOC-UP-001	Documents	Upload valid PDF document	P0
DOC-UP-003	Documents	Reject unsupported file type	P0
DOC-OP-003	Documents	Download document with permission	P0
DOC-OP-006	Documents	Delete document with permission	P0
META-001	Metadata	Add metadata to document	P0
SRCH-001	Search	Search documents by keyword	P0
SRCH-008	Search	Prevent search injection	P0
DASH-001	Dashboard	Dashboard totals match actual data	P1
RPT-002	Reports	Export PDF report	P0
RPT-003	Reports	Export Excel report	P0
AUD-001	Audit	Audit log records upload	P1
SEC-001	Security	API rejects requests without token	P0

7. Initial Requirement Trackability Matrix (RTM)

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-FR-AU-1.1	The system shall allow users to log in securely	FR(Login)	DFD1	UC-01 Login	AuthService.validateCredentials()	LoginController.cs	TC01_Login_Success	--
RE-FR-AU-1.2	The system shall allow users to reset their password	FR(Reset_Password)	DFD1	UC-08 Reset Password	AuthService.resetPassword()	ResetPasswordController.cs	TC02_Reset_Email_Sent	--
RE-FR-DM-2.1	The system shall allow users to upload documents	FR(Upload_Doc)	DFD2	UC-09 Upload Document	DocumentService.upload()	UploadController.cs	TC03_Upload_Valid_File	--
RE-FR-DM-2.2	The system shall allow users to view documents	FR(View_Doc)	DFD2	UC-04 View Document	DocumentService.get()	DocumentController.cs	TC04_View_Authorized	--
RE-FR-DM-2.3	The system shall allow users to edit document metadata	FR(Edit_Metadata)	DFD2	UC-06 Edit Document	MetadataService.update()	MetadataController.cs	TC05_Edit_Metadata_Valid	--
RE-FR-DM-2.4	The system shall allow users to delete documents	FR(Delete_Doc)	DFD2	UC-05 Delete Document	DocumentService.delete()	DocumentController.cs	TC06_Delete_With_Permission	--
RE-FR-SR-3.1	The system shall allow users to search for documents	FR(Search_Doc)	DFD3	UC-02 Search Document	SearchService.query()	SearchController.cs	TC07_Search_Keyword	--
RE-FR-AC-4.1	The system shall verify user permissions before actions	FR(Check_Perms)	DFD1	UC-07 Check Permissions	AuthService.checkPermissions()	AuthMiddleware.cs	TC08_Permission_Denied	--
RE-FR-RP-5.1	The system shall generate general reports	FR(Gen_Report)	DFD4	UC-10 Generate Report	ReportService.generate()	ReportController.cs	TC09_Report_Generation	--
RE-FR-SA-6.1	The system shall allow the admin to add new users	FR(Add_User)	DFD1	UC-03 Add User	UserService.create()	UserController.cs	TC10_Add_User_Valid	--

Chapter5
System Design

1. Introduction

This chapter provides a comprehensive overview of the system design phase, which includes defining the overall architecture, designing internal components, and detailing each functional unit. It highlights the importance of this phase in building a robust and flexible system that fulfills the goals of governmental document management, responds to user feedback, and contributes to continuous improvement and enhanced system quality.

2. System Architecture

This section presents the system design, which includes defining the overall structure and software architecture, ensuring seamless integration and coordination among all components. This phase serves as the foundational blueprint that determines how data flows between users and the system, and how different modules interact to ensure effective performance, security, and future scalability.

System Architecture

Your system is built on a Three-Tier Architecture within a Client–Server model, consisting of the Presentation Layer, Business Logic Layer, and Data Access Layer, with additional support for AI microservices.

The development of the Government Document Management System was organized based on a Three-Tier Architecture within a Client–Server model. This architecture separates the Presentation Layer, Business Logic Layer, and Data Access Layer, making the development process more structured and maintainable, while enhancing scalability and system reliability. This architecture was fully adopted in the system, using modern technologies such as React for the frontend and Django for the backend, along with support for AI microservices to handle intelligent document classification and entity extraction.

Below is a brief explanation of each layer as represented in the system design:

-Presentation Layer

Receives user requests through the web interface and routes them to the server via the API Gateway.

-Business Logic Layer

Contains core functionalities such as document management, search, auditing, and access control.

-Data Access Layer

Handles database operations using the Repository pattern, ensuring separation between business logic and data.

-AI Microservices

Independent services responsible for intelligent document classification and entity extraction.

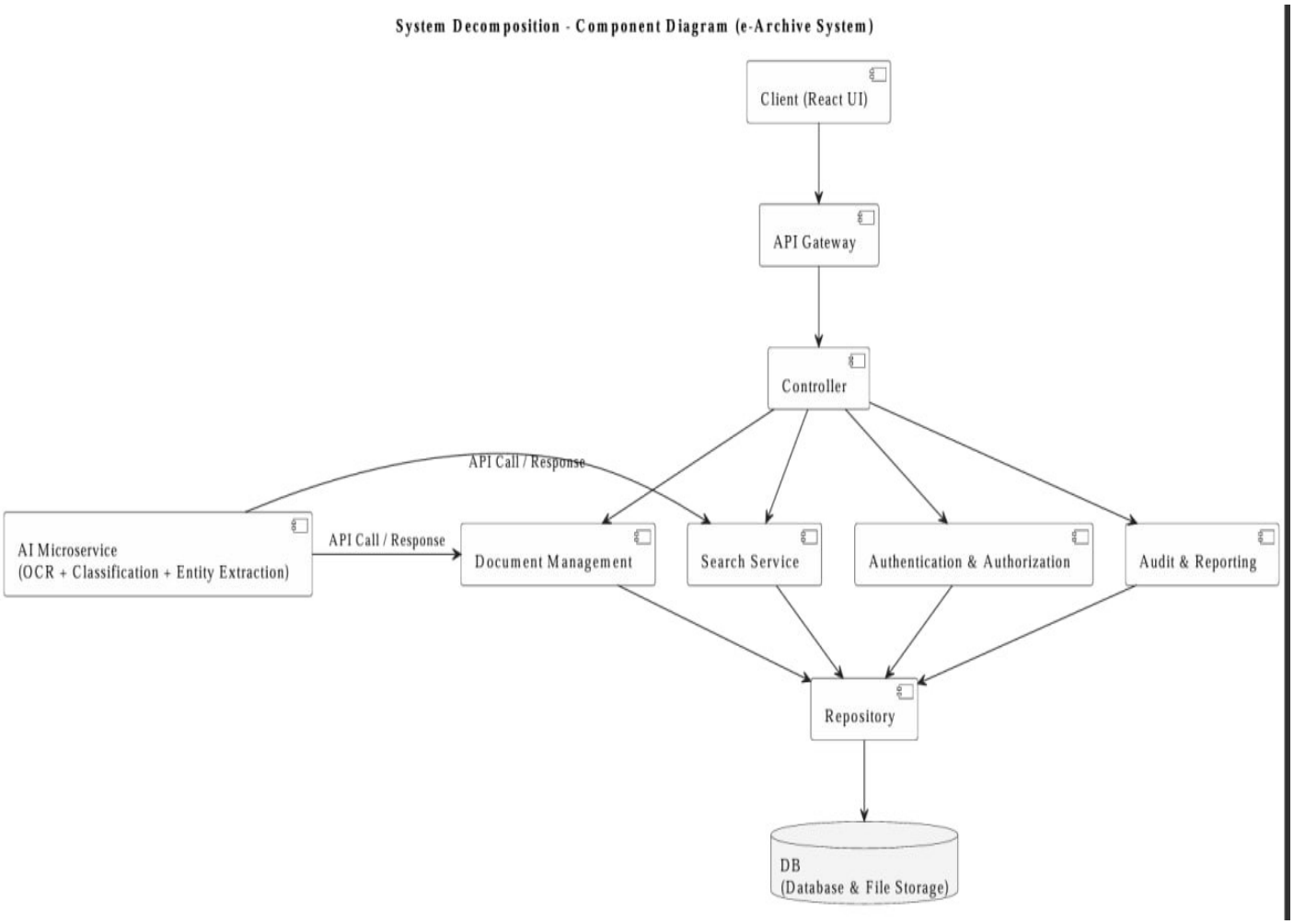
3. Component diagram

This section presents the overall structure of the Government Document Management System, showing how the front-end interfaces interact with the layers to process data and execute operations.

The system consists of five main layers:

Client (Presentation Layer), Presentation Layer, Business Logic Layer, Data Access Layer, and Database Layer.

The system also includes independent AI microservices connected to the Business Logic Layer.



Component Functions

1. Client Layer

- Enables the user to interact with the system through the React UI interface.

- Allows sending requests and performing document-related operations.

2. Presentation Layer

- Controller: Receives requests from the interface and analyzes them.

- Routes the requests to the appropriate internal services.

3. Business Logic Layer

- AI Microservice: Performs OCR, classification, and entity extraction using intelligent techniques.

- Document Management: Handles uploading, editing, deleting, and downloading documents.

- Search Service: Executes search operations and displays results in an organized way.

- Authentication & Authorization: Verifies user identity and manages access permissions.

- Audit & Reporting: Logs all operations and generates usage reports.

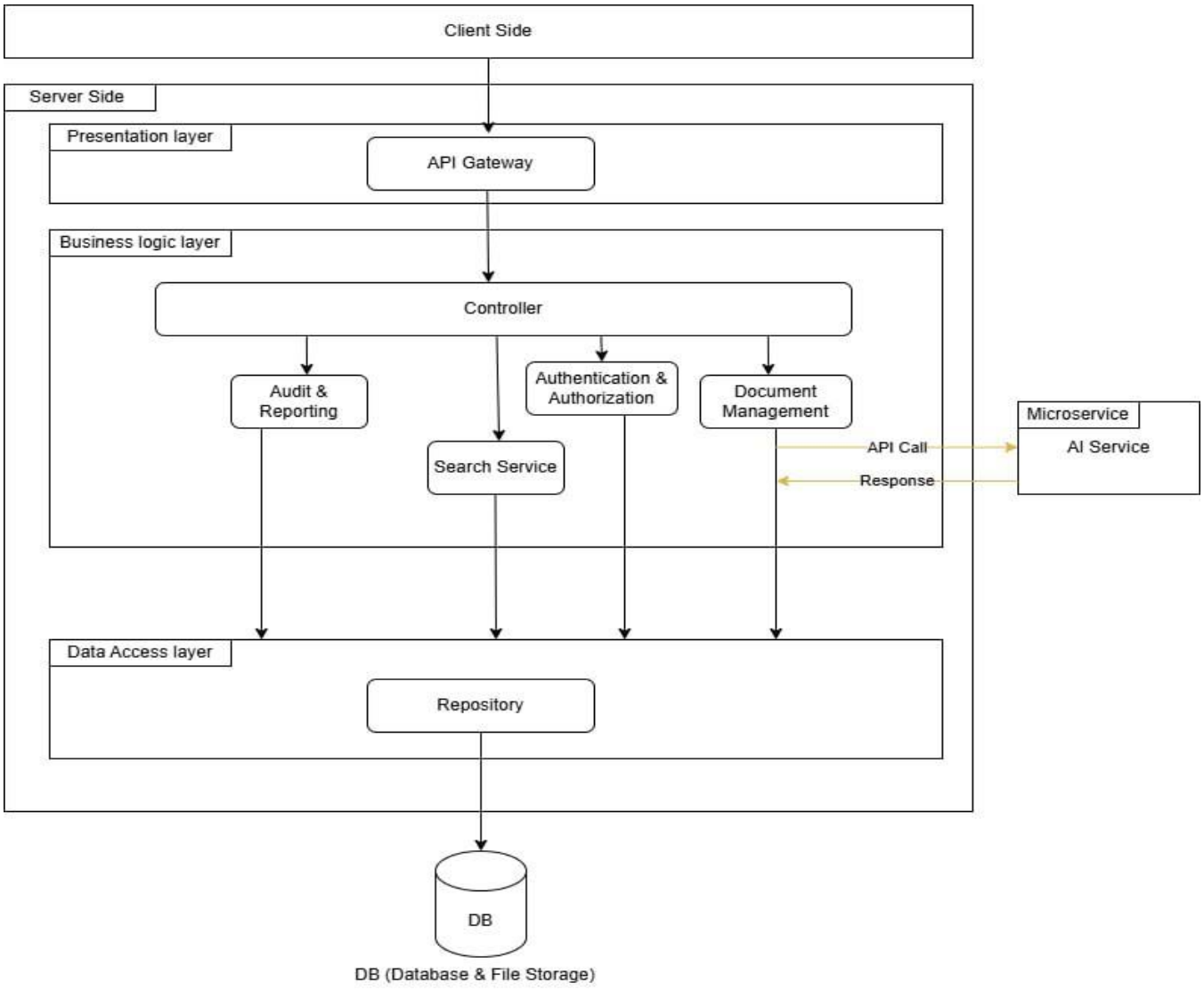
4. Data Access Layer

- Repository: Stores intermediate data and manages structured access to files and services.

5. Database Layer

- Database & File Storage: Securely stores metadata and original document files in an organized manner.

Architecture:



Chapter 6
Practical implementation

Introduction

In this chapter, we will highlight the practical aspect of system implementation by presenting the tools and technologies used throughout the development process. This overview aims to clarify the technical structure adopted in building both the front-end and back-end of the system, as well as data management and storage. Each tool will be explained with its features and role in completing the project, reflecting the integration and efficiency of the chosen technologies.

Tools and Technologies Used

1.React.js

React.js is a JavaScript library used to build interactive user interfaces, and is one of the most popular tools for modern web development.

-It is based on the concept of **components**, which makes code reusable and well-organized.

-It offers high performance through the **Virtual DOM**, enabling fast and smooth user interactions.

-In this project, React.js was used to build the **front-end**, designing dynamic and attractive user pages.

2 .ASP.NET Core

ASP.NET Core is an open-source framework developed by Microsoft for building web applications and backend services**.**

-It supports building advanced APIs , facilitating communication between the front-end and the database.

-It is known for its security, high performance, and scalability.

-In this project, it was used to build the back-end server , handling requests and managing data **.**

3. Django

Django is a high-level Python framework used to develop web applications quickly and efficiently.

-It provides built-in tools such as Models , Views , and Templates , reducing the need for repetitive code.

-It is secure, scalable, and easy to maintain.

-Django was used in this project to support backend functionalities and manage data in a structured and safe manner.

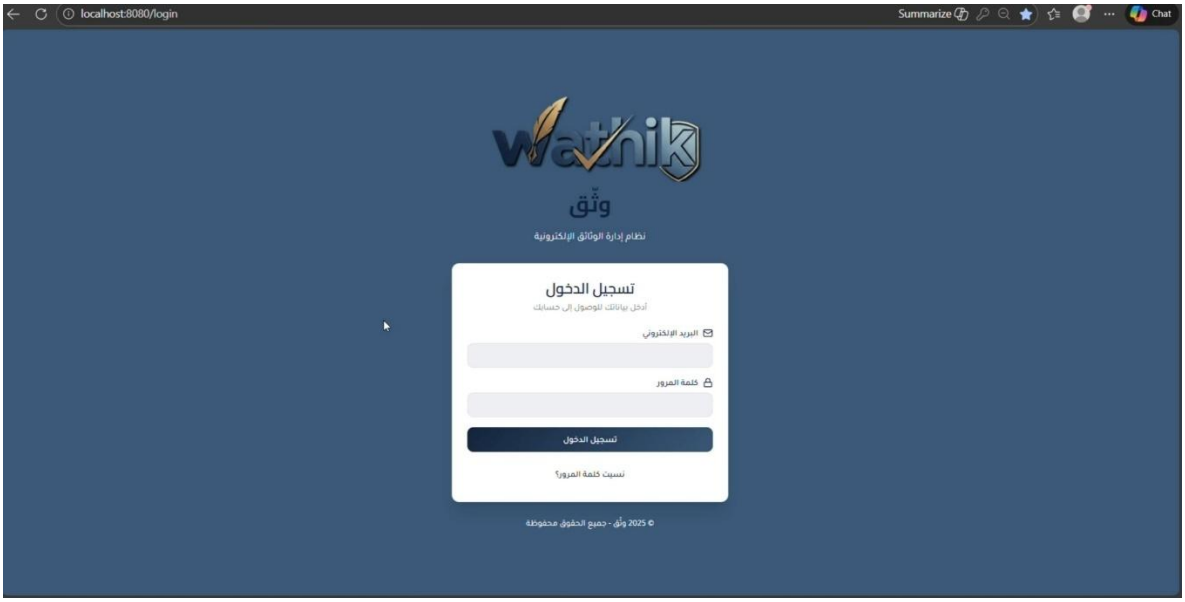
4. Database

A database management system (such as MySQL or PostgreSQL) was used to store and manage project data.

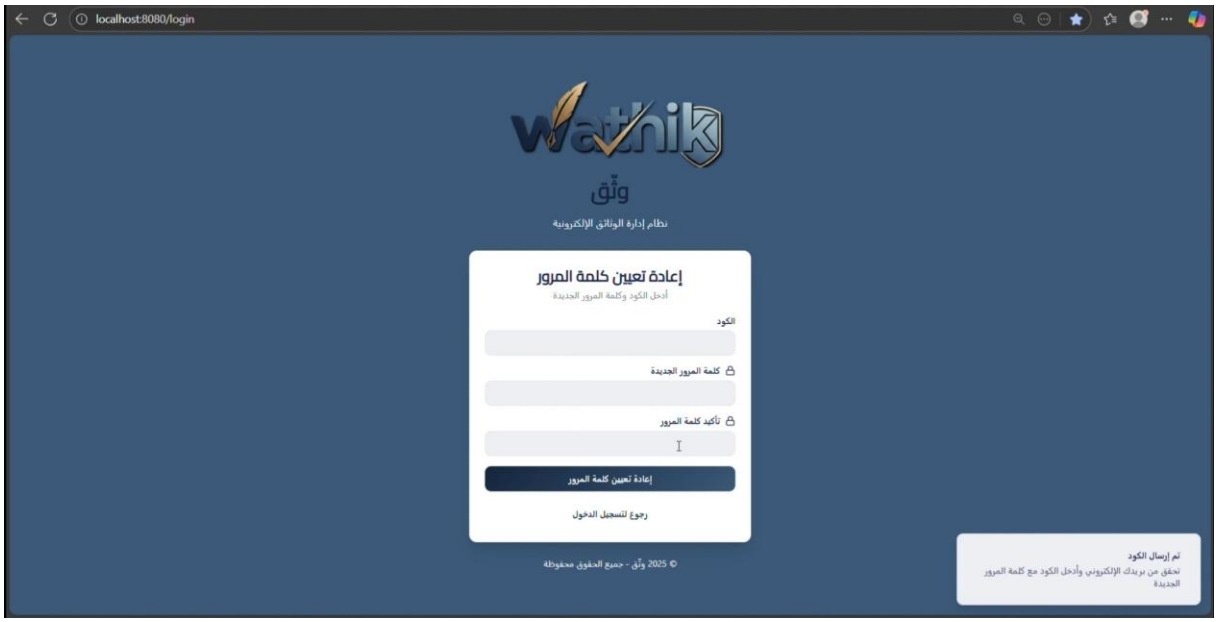
-It stores information about users, products, orders, and shops.

-It is integrated with ASP.NET and Django through APIs, ensuring fast data access and easy management.

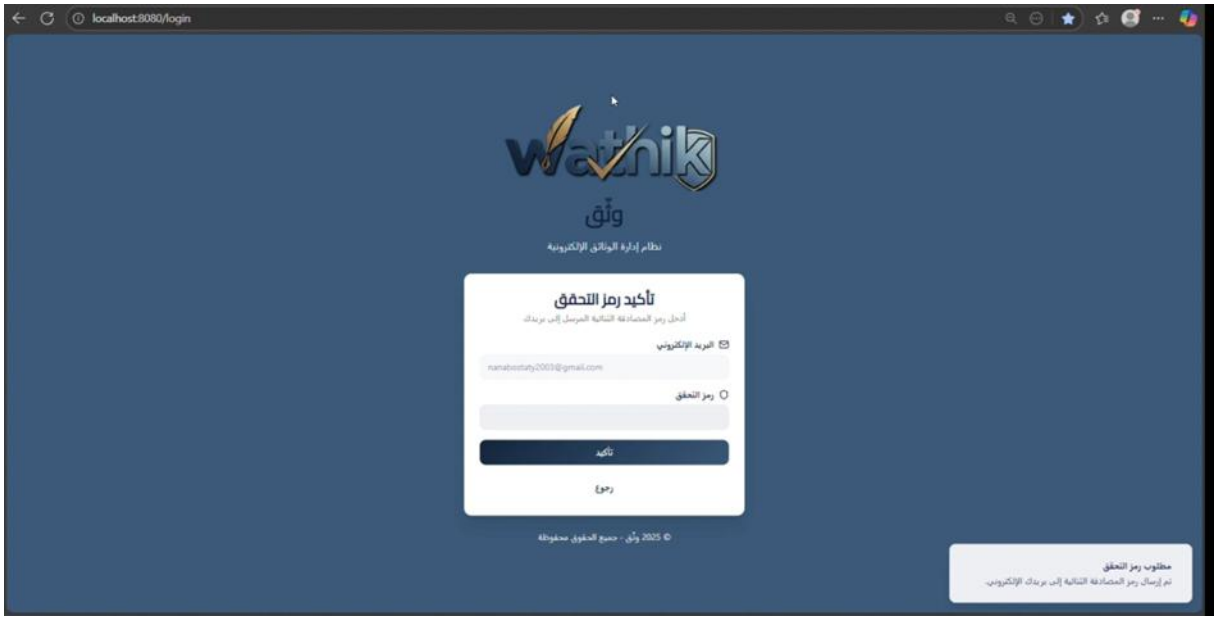
- **System login interface**



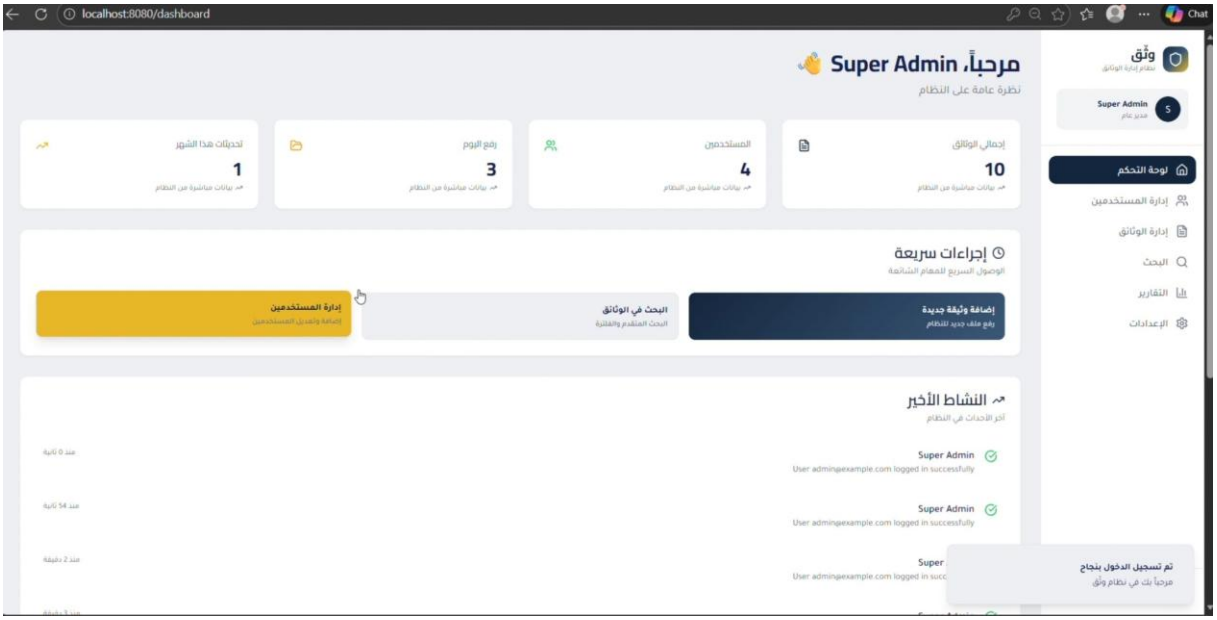
- **Password reset interface**



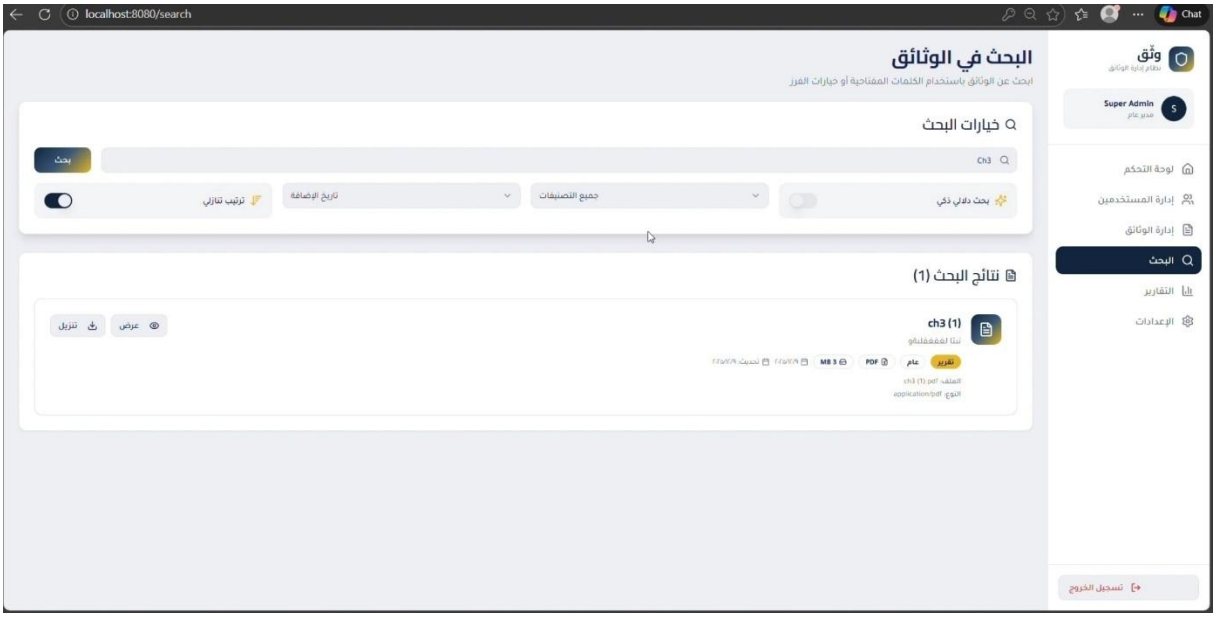
- **Two-factor authentication verification interface**



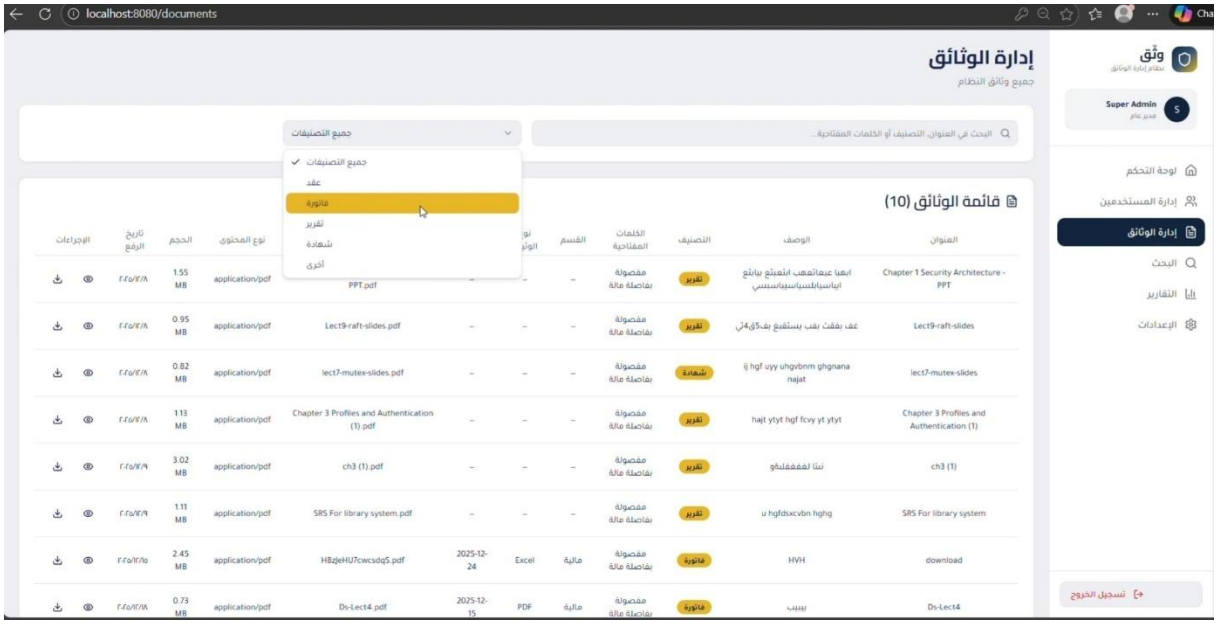
- **Admin dashboard interface**



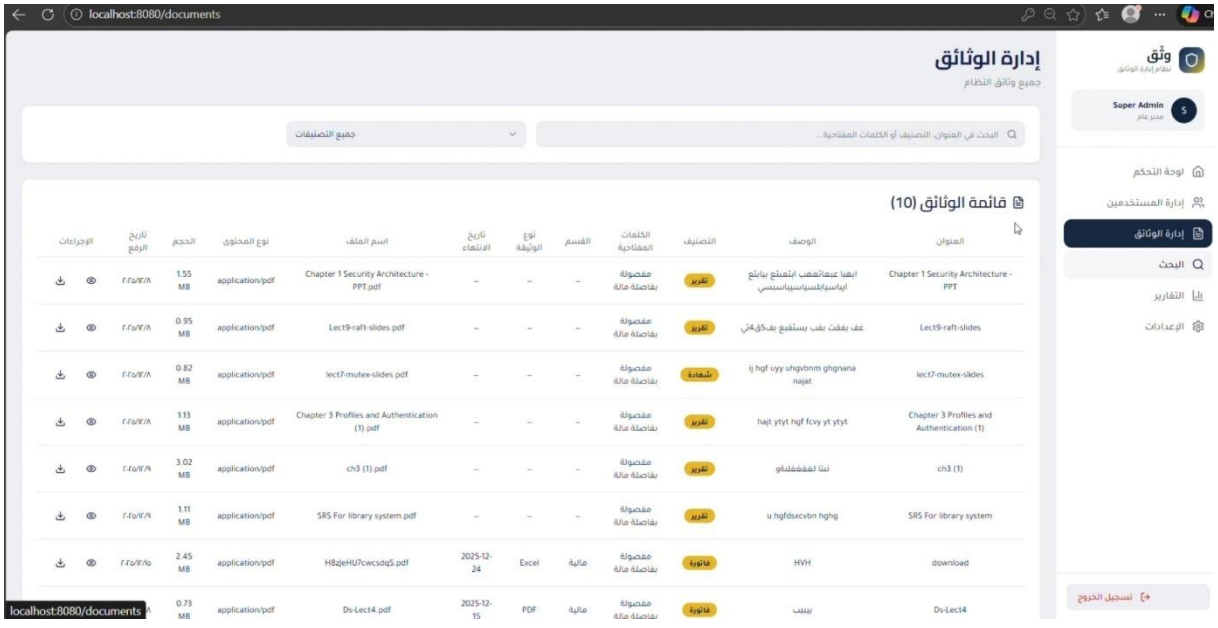
- **Document search interface**



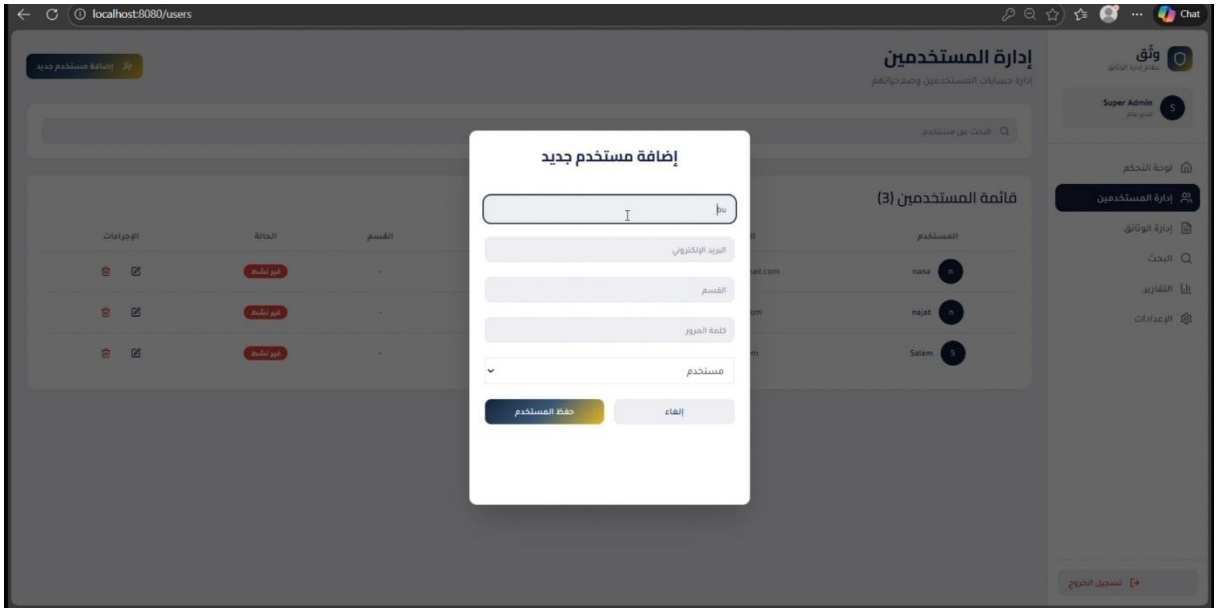
- **Document management interface – display and categorization**



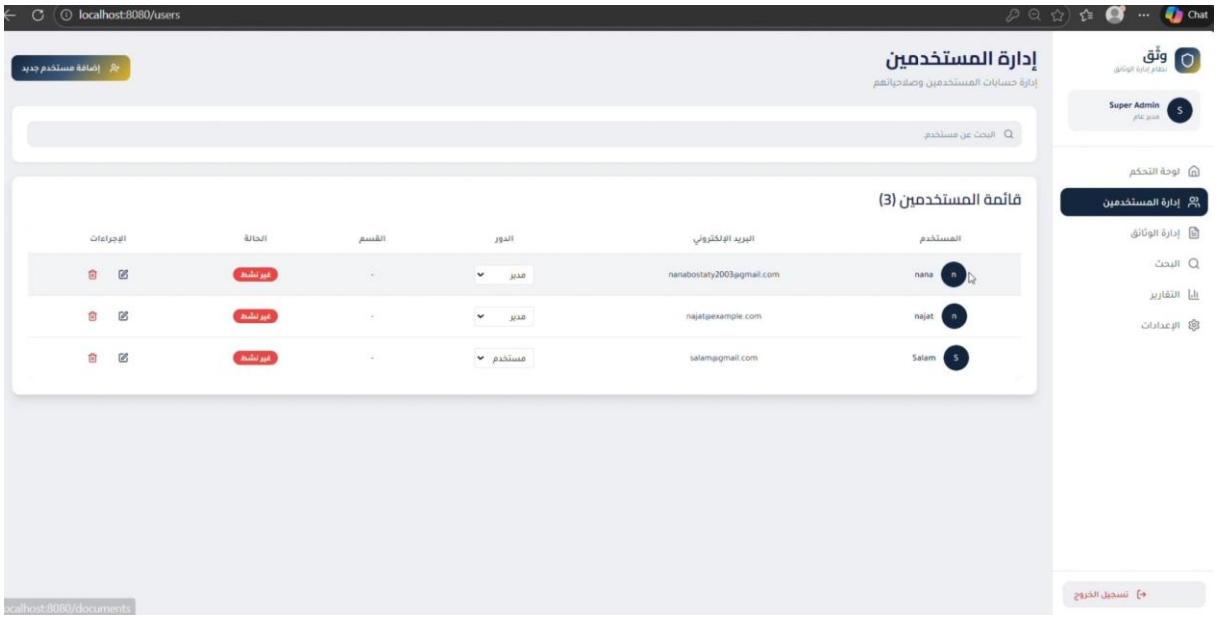
- **Document management interface – document table**



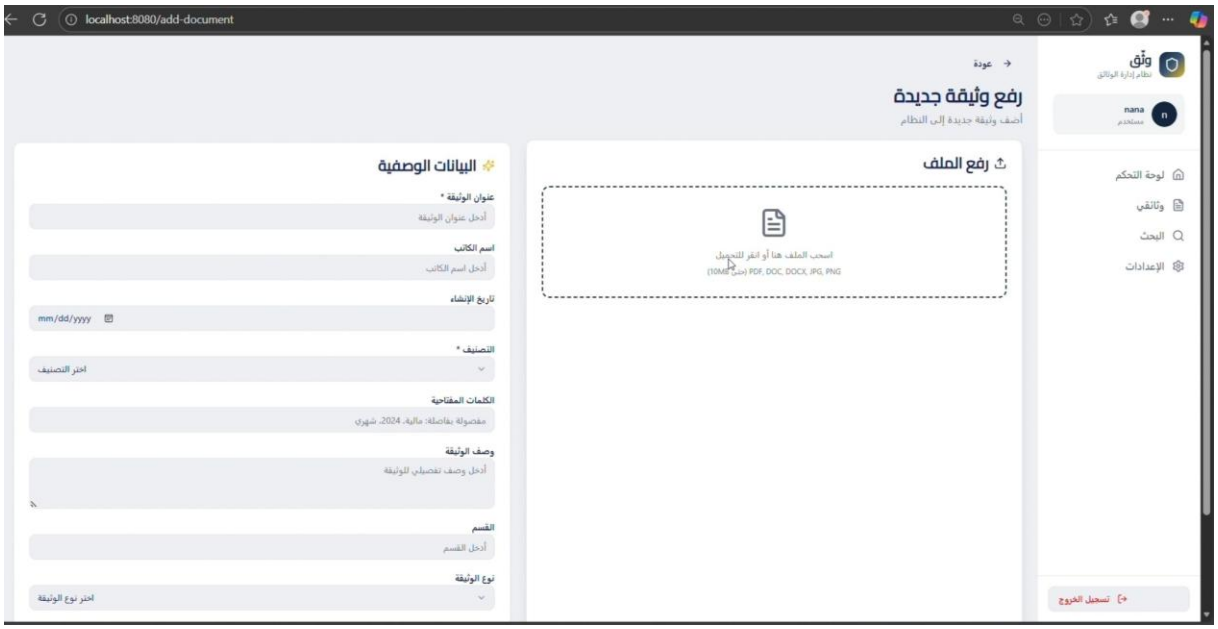
- **Add new user interface**



- **User management interface**

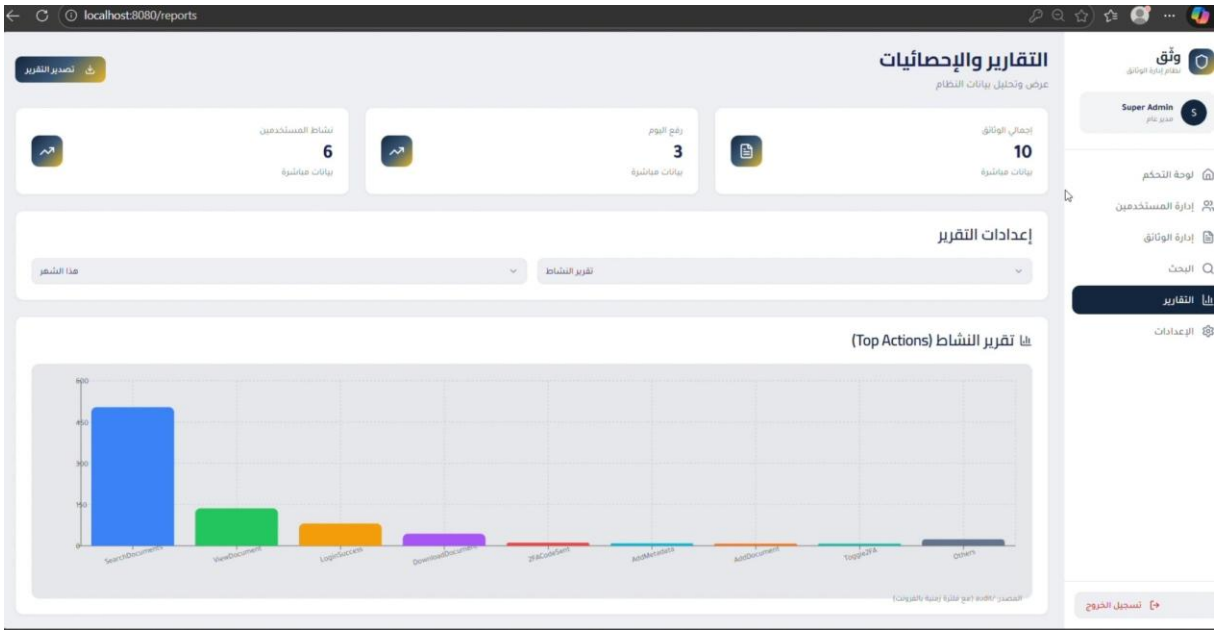


- **Upload new document interface**

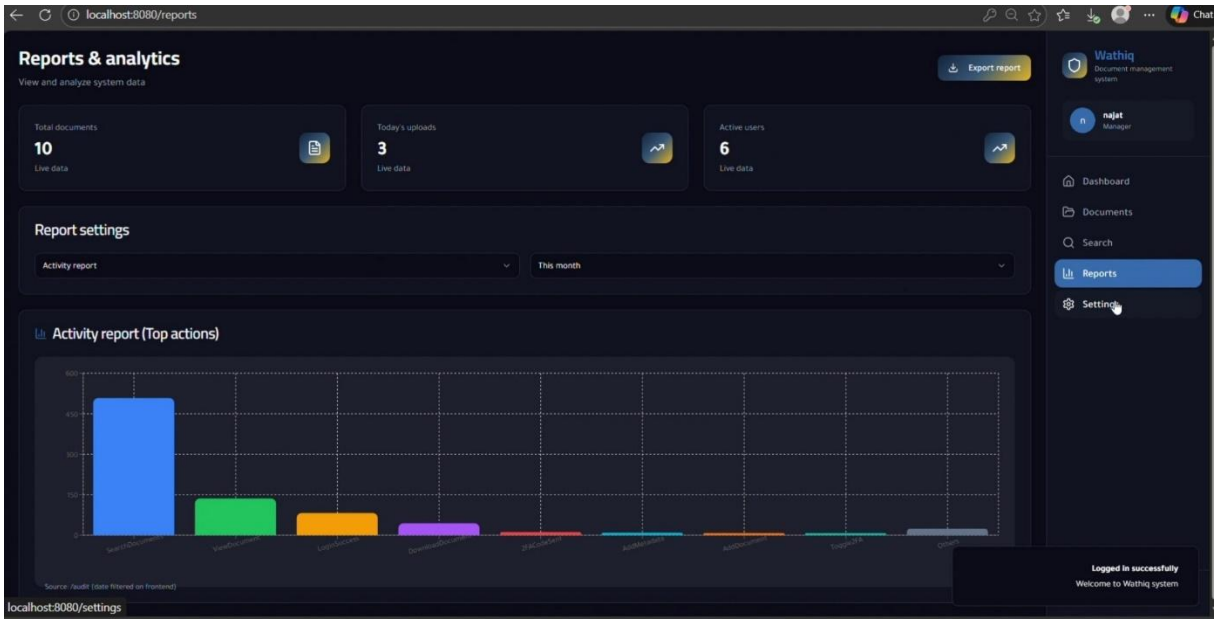


- **Document metadata display interface**

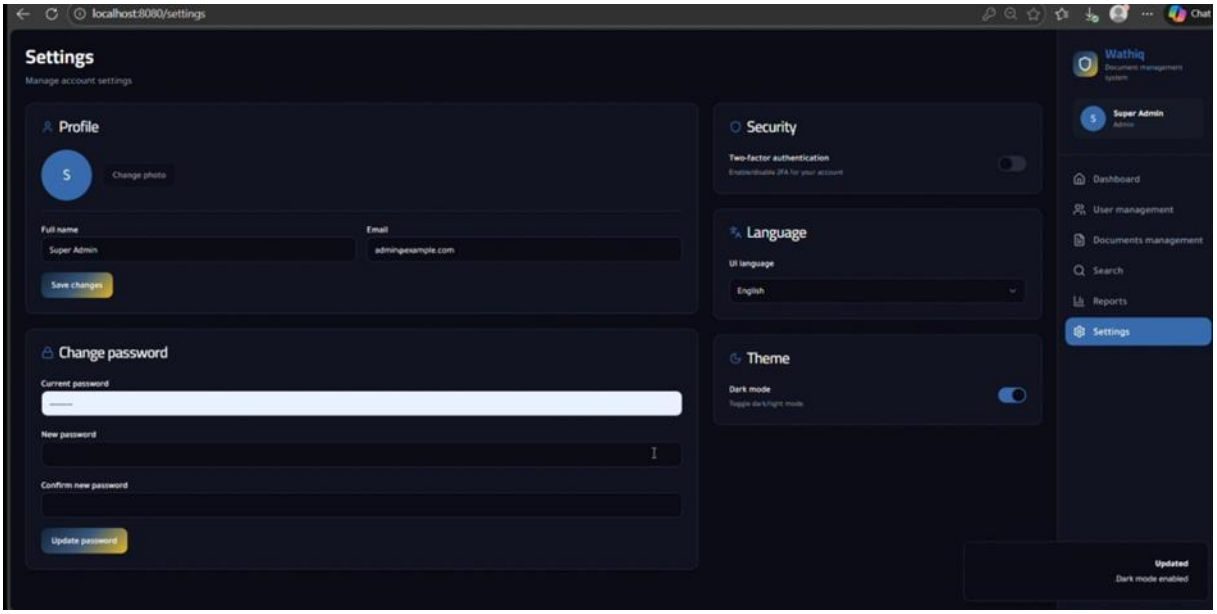
- **Reports and analytics interface – Admin view**



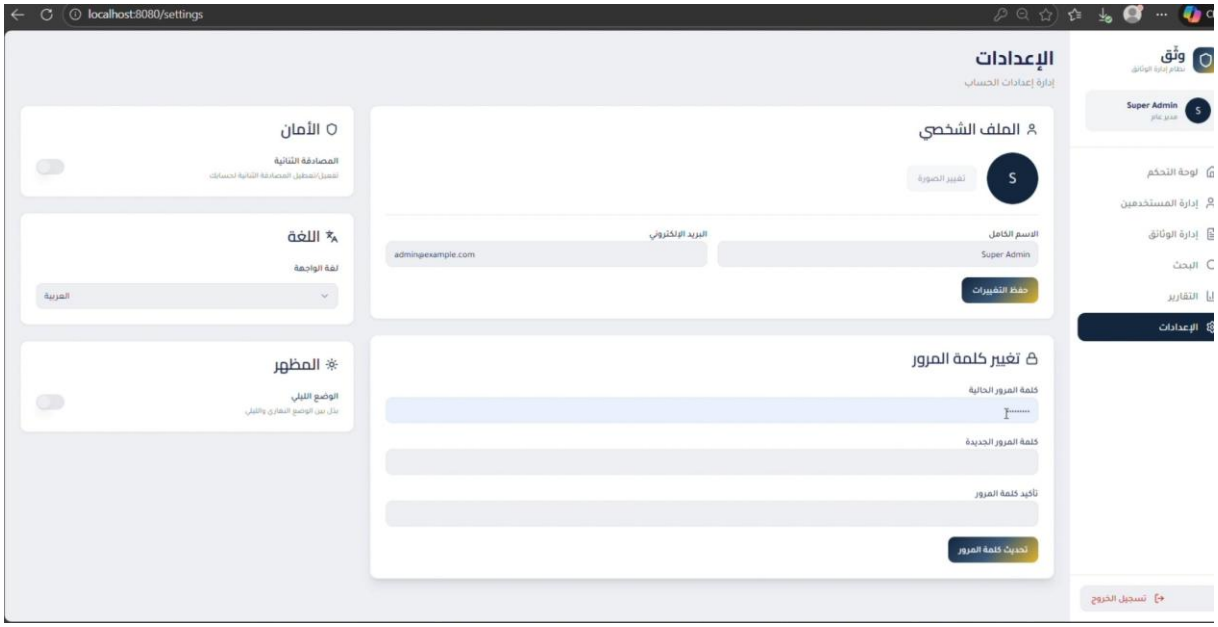
- **Reports and analytics interface – User view**



- **Account settings interface – Dark mode**

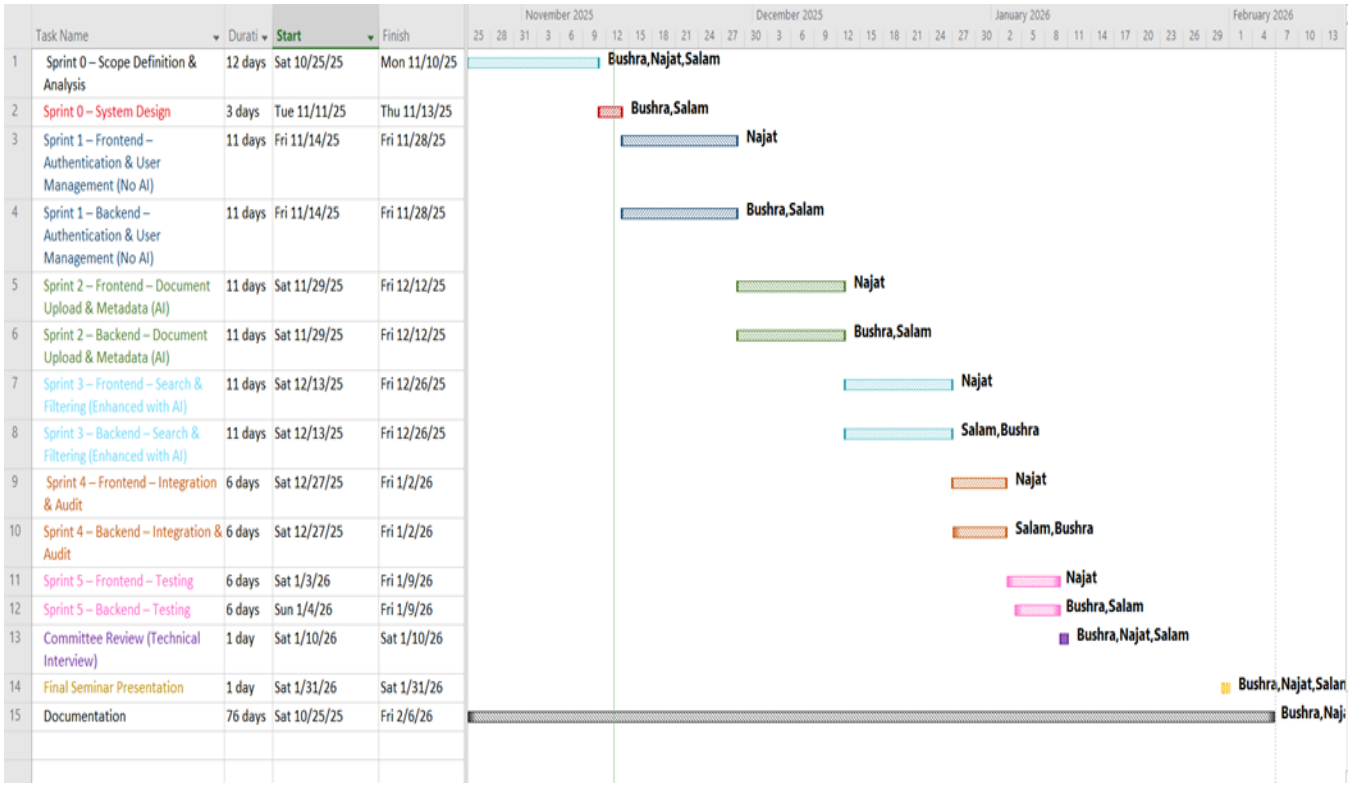


- **Account settings interface – Light mode**



Chapter 7
Project management

1. project plan – Gantt Chart:



	Task Name	Durati	Start	Finish
1	Sprint 0 – Scope Definition & Analysis	12 days	Sat 10/25/25	Mon 11/10/25
2	Sprint 0 – System Design	3 days	Tue 11/11/25	Thu 11/13/25
3	Sprint 1 – Frontend – Authentication & User Management (No AI)	11 days	Fri 11/14/25	Fri 11/28/25
4	Sprint 1 – Backend – Authentication & User Management (No AI)	11 days	Fri 11/14/25	Fri 11/28/25
5	Sprint 2 – Frontend – Document Upload & Metadata (AI)	11 days	Sat 11/29/25	Fri 12/12/25
6	Sprint 2 – Backend – Document Upload & Metadata (AI)	11 days	Sat 11/29/25	Fri 12/12/25
7	Sprint 3 – Frontend – Search & Filtering (Enhanced with AI)	11 days	Sat 12/13/25	Fri 12/26/25
8	Sprint 3 – Backend – Search & Filtering (Enhanced with AI)	11 days	Sat 12/13/25	Fri 12/26/25
9	Sprint 4 – Frontend – Integration & Audit	6 days	Sat 12/27/25	Fri 1/2/26
10	Sprint 4 – Backend – Integration & Audit	6 days	Sat 12/27/25	Fri 1/2/26
11	Sprint 5 – Frontend – Testing	6 days	Sat 1/3/26	Fri 1/9/26
12	Sprint 5 – Backend – Testing	6 days	Sun 1/4/26	Fri 1/9/26
13	Committee Review (Technical Interview)	1 day	Sat 1/10/26	Sat 1/10/26
14	Final Seminar Presentation	1 day	Sat 1/31/26	Sat 1/31/26
15	Documentation	76 days	Sat 10/25/25	Fri 2/6/26

2. Risk management:

Risk Name	Category	Probability	Impact	Mitigation Plan
Unauthorized Access to Documents	Technical	20%	Medium	Implement strict role-based access control (RBAC) and ownership validation for all document-related operations. Conduct regular authorization testing and audits.
Unsecured OCR Callback Endpoint	Security	15%	High	Secure the callback endpoint using API keys or HMAC signatures, restrict access by IP allow listing, and validate all incoming requests before processing.
Running OCR on Duplicate Documents	Performance	10%	Low	Perform duplicate file detection using file hashing before triggering OCR and ensure OCR is executed only for newly created documents.
Delayed OCR Results for Users	Technical	60%	Medium	Use asynchronous processing indicators, polling mechanisms, and clear user notifications to inform users about OCR processing status.
System Performance Degradation When Processing Large Files	Technical	40%	High	Enforce file size and page limits, optimize OCR processing, and utilize background job queues to manage heavy workloads efficiently.
Metadata Inconsistency	Technical	30%	Medium	Define a single source of truth for metadata storage and apply atomic update mechanisms to ensure data consistency across the system.
OCR Failure in Different Deployment Environments	Technical	40%	High	Externalize OCR configurations using environment variables and perform environment-specific testing before deployment.
Redirecting Users to Unauthorized Pages Based on Role	Security	20%	Low	Implement role-based routing logic and validate navigation paths according to user roles during frontend and backend processing.
OCR Failure Due to Missing Language Files	Technical	30%	Medium	Add startup health checks to verify the presence of required OCR language files and provide fallback mechanisms or error notifications if files are missing.

Configuration System (Version Control & Configuration Management)

1. Project Description

The project, **Wathiq**, is a smart electronic archiving system designed to digitize government documents and transform them into searchable and categorized digital data using artificial intelligence techniques.

The system allows users to upload documents, manage metadata, and apply OCR (Optical Character Recognition) to extract text and support future intelligent search and classification features.
During development, the system evolved from a monolithic architecture into a more scalable microservice-based architecture, particularly for the OCR functionality.

2. Git Repository Structure

The project is managed using Git and hosted on GitHub.

The repository contains the complete backend source code of the system.

The OCR functionality was later extracted into a standalone microservice (eArchive.OcrService) to improve scalability and separation of concerns.

Repository link :

https://github.com/salamalmasri33-crypto/Wathiq

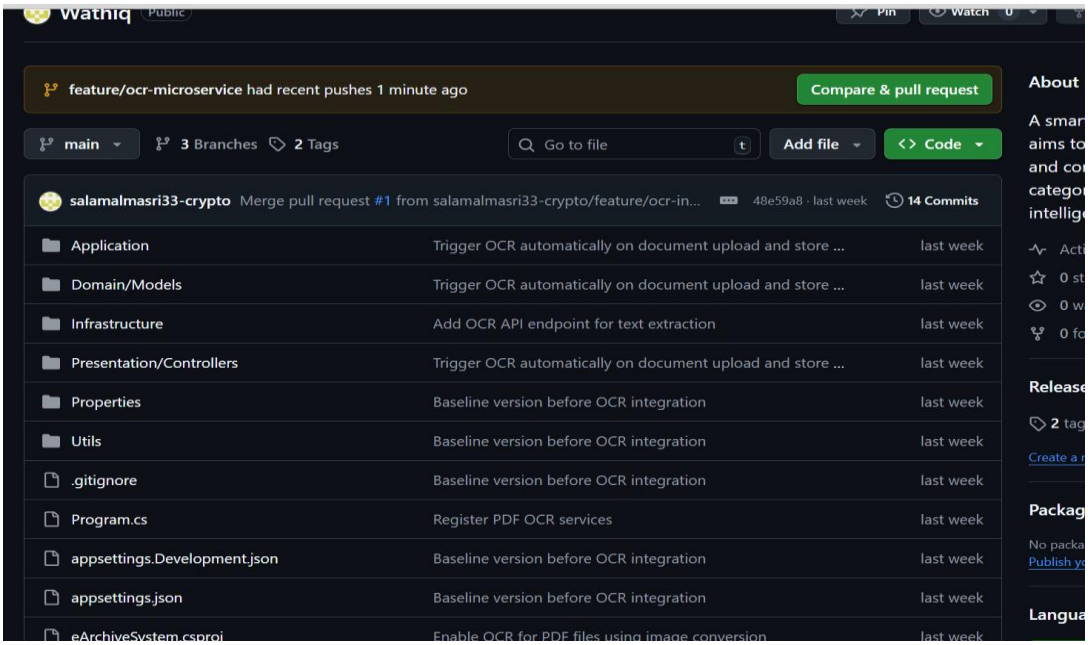
Repository structure highlights:

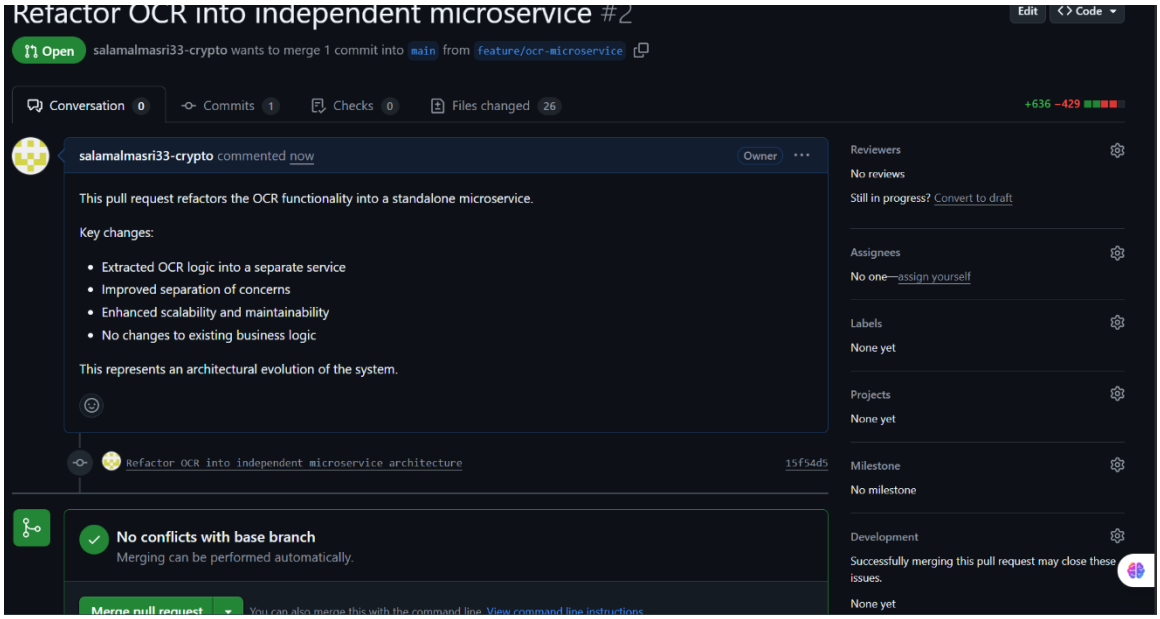
- **Application**: Application layer (DTOs, interfaces, services)
- **Domain**: Core domain models (Document, Metadata, OCR-related entities)
- **Infrastructure**: Persistence, repositories, and technical implementations

- **Presentation**: API controllers
- **eArchive.OcrService**: Independent OCR microservice extracted from the main system

This structure follows clean architecture principles and clearly separates concerns between layers.

.





3. Branching and Merging Strategy

The project follows a **feature-based branching strategy**:

- **main**: Stable production-ready branch
- **feature/ocr-integration**: Initial OCR integration inside the monolithic system
- **feature/ocr-microservice**: Refactor of OCR into an independent microservice

In addition to OCR-related branches, frontend development was implemented using

a dedicated feature **branch** :

• **Wathiq_FrontEnd**: Frontend user interface development and integration

The frontend branch followed the same workflow, where changes were reviewed

and merged into the main branch using Pull Requests after validation.

Branching workflow:

1. Development starts by creating feature branches from the main branch.
2. New features are implemented in dedicated feature/* branches.
3. Each feature branch is merged into main using **Pull Requests (PRs)**.
4. Code changes are reviewed before merging.

Example:

- OCR was first implemented in feature/ocr-integration.
- Later, OCR was refactored into a standalone microservice in feature/ocr-microservice.

- The refactor was merged into main using a Pull Request with no conflicts.

This approach ensured safe evolution of the system architecture.

All changes were merged using Pull Requests after review.

4. Team Members and Responsibilities

The Wathiq project was developed by a **team of three members**, following an agile sprint-based development plan.
Responsibilities were distributed across different sprints, covering frontend, backend, system design, security, AI features, and documentation.

Team Members

- **Salam**
- **Bushra**
- **Najat**

Roles and Responsibilities

Salam – Backend Developer & System Architect

Responsible for the core backend development and architectural decisions, including:

- Designing the overall system architecture
- Backend implementation for user management and authentication
- Document and metadata management (backend)
- OCR integration and refactoring OCR into an independent microservice
- API design and backend logic for dashboards, reports, and audit logs
- Backend security and quality improvements
- Participating in AI-related backend features

- Managing Git branches, Pull Requests, merges, and version tags
- Contributing to documentation and final system integration

Bushra – Backend & System Support

Focused on backend development and system support tasks, including:

- Backend implementation for user management and authentication
- Backend document and metadata management
- Participation in dashboard, reporting, and audit log backend features
- Backend security and quality improvements
- Assisting in AI backend features
- Supporting system testing and integration
- Contributing to documentation and final review activities

Najat – Frontend Developer

Responsible for frontend development and user interface implementation, including:

- Frontend user management and authentication interfaces
- Frontend document management and metadata handling
- Dashboard, reports, and audit log user interfaces
- Frontend security and quality improvements
- Frontend AI feature integration
- Supporting usability testing and UI refinements

Collaboration and Workflow

The team followed a sprint-based development workflow, where:

- Each sprint had clearly defined frontend and backend tasks

- Responsibilities were assigned according to team member expertise
- Progress was tracked using a Gantt chart and sprint milestones
- Git and GitHub were used for collaboration, version control, and integration
- Pull Requests were created for major features and architectural changes
- Code was reviewed before merging into the main branch

This collaborative approach ensured clear responsibility distribution, efficient development, and successful delivery of the system features according to the project plan.

5. Development Workflow

The development workflow followed these steps:

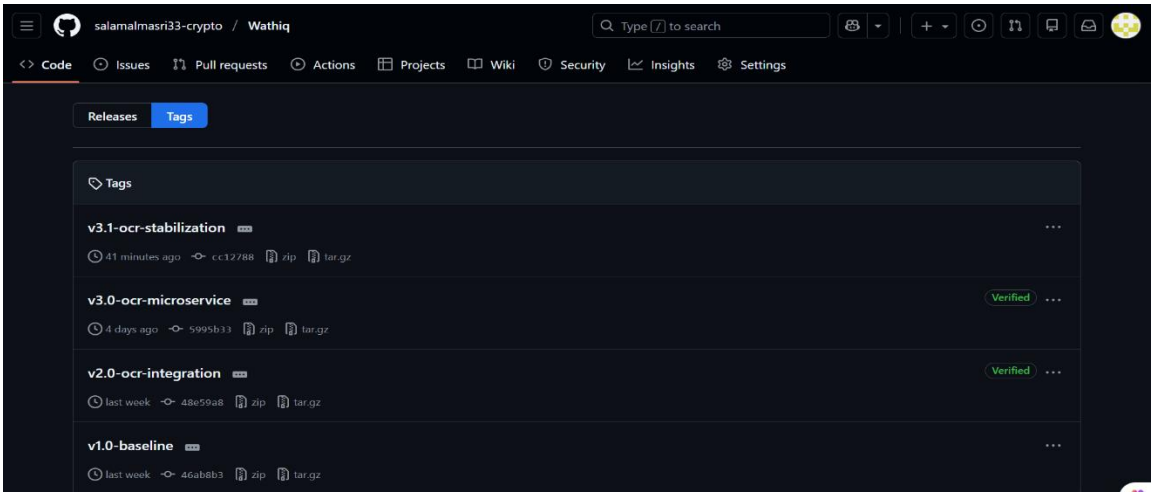
1. Task identification (e.g., OCR integration, refactoring)
2. Creation of a feature branch
3. Implementation and local testing
4. Commit changes with descriptive messages
5. Push changes to GitHub
6. Create a Pull Request
7. Review code changes
8. Merge into the main branch
9. Tag the version to mark a milestone

This workflow ensured traceability, maintainability, and controlled system evolution

6. Tags and Versioning

Git tags were used to mark major project milestones:

- **v1.0-baseline**
  Baseline version before OCR integration.
- **v2.0-ocr-integration**
  OCR functionality integrated directly into the main system.
- **v3.0-ocr-microservice**
  OCR refactored into an independent microservice, improving scalability and maintainability.
- **v3.1-ocr-stabilization**
  Stabilization release focusing on fixing OCR-related issues, improving image handling reliability, and refining metadata extraction without changing the overall system architecture.



Each tag represents a stable snapshot of the system at a specific development stage and clearly documents architectural evolution.

Configuration Management System :

1. Introduction to Configuration Management

Configuration Management is a disciplined process used in software engineering to **systematically manage, track, and control changes** in a software system throughout its lifecycle.

Its main purpose is to ensure that:

- The system remains **consistent and stable**
- Changes are **controlled and traceable**
- Previous system states can be **restored when needed**
- Multiple versions of the system can coexist safely

In this project, Configuration Management was applied using **Git-based version control**, structured branching strategies, and formal version tagging.

2. Configuration Items (CIs)

A **Configuration Item (CI)** is any component of the system that is placed under configuration control.

In the *Wathiq* project, the following were treated as configuration items:

- Source code files
- API controllers
- Domain models (Document, Metadata)
- OCR processing logic
- Application configuration files (appsettings.json)
- Project structure and architecture

- Microservice boundaries

Each change to these items was tracked through Git commits.

The system consists of two main Configuration Items:

1. **eArchiveSystem**
    - Represents the core electronic archiving system.
    - Responsible for document management, metadata storage, user management, and system orchestration.
2. **eArchive.OcrService**
    - Represents an independent OCR microservice.
    - Responsible only for OCR processing and text extraction.
    - Can be developed, tested, and deployed independently from the main system.

Each project is versioned and managed as a separate configuration item within the same solution.

3. Configuration Identification

Configuration Identification defines **what is controlled and how it is uniquely identified**.

In this project, identification was achieved through:

- **Git commits** (unique commit hashes)
- **Branch names** representing development goals

    - feature/ocr-integration
    - feature/ocr-microservice
    - Wathiq_FrontEnd – Frontend user interface development and integration
- **Semantic version tags**:
    - v1.0-baseline
    - v2.0-ocr-integration
    - v3.0-ocr-microservice
    - v3.1-ocr-stabilization

Each tag uniquely identifies a stable configuration of the system.

4. Configuration Control

Configuration Control ensures that changes are **introduced in a controlled and approved manner**.

This was implemented through:

- Feature branches for isolated development
- Pull Requests (PRs) for controlled merging
- Code review before merging into main
- Prevention of direct unstable changes to the main branch

5. Configuration Status Accounting

Configuration Status Accounting is the process of **recording and reporting the state of configuration items over time**.

In this project, this was achieved by:

- Git commit history
- Pull Request history
- Tag history showing system evolution
- Branch comparison (before and after refactor)

6. Configuration Auditing

Configuration Auditing verifies that:

- Changes were properly implemented
- The system matches its documented configuration
- No unauthorized changes exist

In the project:

- Pull Requests acted as configuration audits
- Merged code was verified to work correctly
- OCR refactor did not break existing functionality
- Tags marked only stable, verified versions

7. Configuration Management Applied to Architectural Evolution

A key Configuration Management activity in this project was managing **architectural evolution**.

The transition from:

- **Monolithic OCR integration**
  to **Independent OCR microservice**

was handled as a controlled configuration change.

Using branching, versioning, and tagging allowed:

- Safe experimentation
- Rollback if needed
- Clear documentation of architectural decisions

Conclusion

Configuration Management in the *Wathiq* project ensured that system changes were controlled, traceable, and reversible.
By combining Git version control, structured branching, Pull Requests, and version tagging, the project successfully managed a major architectural refactor without disrupting system stability.