

**OPERATING SYSTEM**

# Project Title: Enterprise-Grade File Management System

**Course name:** Operating System

**Course code:** CSE-316

**Section:** K23WA

|  |  |
| --- | --- |
| **Submitted To: -** | **Dr. Gurbinder Singh Brar** |

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll No.** | **Registration No.** |
| **Ankit Kumar** | **48** | **12318930** |
| **Akshat Mahla** | **46** | **12321409** |
| **Bailapudi Jaswanth Kumar** | **47** | **12320659** |

# DECLARATION

This is to declare that , Bailapudi Jaswanth Kumar, a student of Bachelor of

Technology under CSE discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own work and is authentic data.

Bailapudi Jaswanth Kumar (Reg. no: 12320659)

Department of Computer Science and Engineering

Lovely Professional University, Punjab

# Project Title : Enterprise-Grade File Management System

## 1. Project Overview: -

**Project Goals:**

The primary goal of this project is to design and implement a robust and secure enterprise-grade file management system. This system will provide a centralized and controlled environment for organizations to store, manage, and share digital assets while adhering to stringent security requirements. Key objectives include:

* **Secure Storage:** Implement a system that ensures the confidentiality and integrity of stored files through strong encryption mechanisms.
* **Strong Authentication:** Establish a multi-layered authentication process, including password-based login and two-factor authentication (2FA), to verify user identities and prevent unauthorized access.
* **Granular Access Control:** Develop a comprehensive access control model that allows administrators to define precise permissions for users and groups on specific files and folders, ensuring the principle of least privilege.
* **Secure File Operations:** Enable authorized users to perform essential file operations (reading, writing, sharing, metadata retrieval) in a secure manner, preventing unauthorized modifications or data breaches.
* **Threat Detection and Mitigation:** Integrate proactive security measures to detect and mitigate common security threats such as buffer overflows and malware attacks, ensuring the system's resilience.
* **Efficient File Management:** Provide a user-friendly interface and efficient backend infrastructure for organizing, searching, and retrieving files effectively.
* **Auditability:** Maintain a detailed audit log of all system activities, including user logins, file accesses, modifications, and administrative actions, to facilitate security monitoring and incident response.

**Expected Outcomes:**

Upon successful completion of this project, the following outcomes are expected:

* **A fully functional and secure file management system:** Capable of storing, managing, and sharing files within an enterprise environment.
* **Implementation of strong authentication mechanisms:** Including password-based login with robust password policies and a functional twofactor authentication system.
* **A comprehensive access control model:** Allowing for the definition and enforcement of granular permissions on files and folders based on user roles and groups.
* **End-to-end encryption of data:** Ensuring data confidentiality both at rest and in transit.
* **Secure implementation of core file operations:** Reading, writing, sharing, and metadata retrieval will be protected against unauthorized access and manipulation.
* **Integration of security threat detection and mitigation techniques:** Including measures to prevent buffer overflows and detect potential malware uploads.
* **A detailed audit logging system:** Providing a comprehensive record of system activities for security analysis and compliance purposes.
* **Documentation:** Comprehensive documentation covering the system architecture, design decisions, implementation details, user guides, and security protocols.

**Project Scope:**

This project will encompass the following key areas:

* **System Architecture Design:** Defining the overall architecture of the file management system, including its components, interactions, and data flow.
* **Authentication and Authorization Module Development:** Implementing password-based authentication with secure password storage and the integration of a two-factor authentication mechanism. Developing the access control model and its enforcement mechanisms.
* **Secure Storage Implementation:** Choosing and implementing appropriate encryption techniques for data at rest and ensuring secure data handling practices.
* **Secure File Operations Development:** Implementing the core file operations (read, write, share, metadata retrieval) with built-in security checks and preventative measures.
* **Threat Detection and Mitigation Implementation:** Integrating mechanisms for detecting and mitigating buffer overflows and implementing basic malware scanning capabilities for uploaded files.
* **Audit Logging Implementation:** Developing a robust audit logging system to track relevant system activities.
* **User Interface (UI) Development (Basic):** Creating a basic user interface for interacting with the file management system (uploading, downloading, browsing, sharing). The focus will be on functionality and security rather than advanced UI/UX features.
* **API Development:** Designing and implementing secure APIs for programmatic interaction with the file management system.
* **Security Testing:** Conducting thorough security testing, including vulnerability scanning and penetration testing, to identify and address potential security weaknesses. 1
* **Documentation:** Creating comprehensive technical and user documentation.

**Out of Scope:**

The following aspects are considered outside the scope of this initial project phase:

* Advanced UI/UX Design: Focus will be on core functionality and security.
* Integration with existing enterprise systems (e.g., LDAP, Active Directory) beyond basic authentication.
* Advanced collaboration features (e.g., real-time co-editing).
* Version control and history tracking beyond basic overwriting.
* Data loss prevention (DLP) beyond basic access control and encryption.
* Scalability and high-availability infrastructure design and implementation (beyond basic considerations).
* Mobile application development.
* Compliance with specific industry regulations (e.g., HIPAA, GDPR) in detail (general security principles will be followed).
* Sophisticated malware analysis and sandboxing.

## 2. Module-Wise Breakdown: -

To effectively design and implement the robust and secure file management system, we can divide the project into the following three core modules: **I. Core Backend & Security Module:**

* **Purpose:** This module forms the heart of the file management system. It is responsible for handling all core functionalities related to data storage, retrieval, manipulation, security, and access control. It acts as the central processing unit and data repository for the entire system.
* **Roles:**

o **Authentication and Authorization:**

* + Handles user registration, login, and session management.
  + Implements password-based authentication with strong password policies (e.g., complexity requirements, password hashing with salt).
  + Integrates two-factor authentication (2FA) mechanisms (e.g., TOTP, SMS-based).
  + Manages user roles, groups, and permissions.
  + Enforces the access control model, determining whether a user or group has the necessary permissions for specific file operations.
* **Secure Storage Management:**
  + Manages the underlying storage infrastructure.
  + Implements encryption at rest for all stored files (e.g., AES256).
  + Handles secure file uploads and downloads.
  + Manages file metadata (e.g., creation date, modification date, size, owner, permissions).
* **File Operations Logic:**
  + Implements the core file operations: reading, writing, sharing, and metadata retrieval.
  + Ensures that all file operations are performed securely and in accordance with the enforced access control policies.
  + Manages file locking mechanisms to prevent concurrent modifications and data corruption.
* **Security Threat Detection and Mitigation:**
  + Implements input validation and sanitization techniques to prevent buffer overflow vulnerabilities.
  + Integrates with malware scanning engines to proactively scan uploaded files for malicious content.
  + Implements rate limiting and other security measures to prevent brute-force attacks and denial-of-service attempts.
  + Handles error handling and logging in a secure manner to avoid revealing sensitive information.
* **Audit Logging:**
  + - Records all significant system events, including user logins, file accesses, modifications, permission changes, and security-related events.
    - Ensures the integrity and immutability of audit logs for security monitoring and compliance purposes.
* **API Layer:**
  + - * Provides a secure and well-defined API for other modules

(like the User Interface) to interact with the core functionalities.

* + - * Enforces authentication and authorization at the API level.

1. **User Interface (UI) Module:**
   * **Purpose:** This module provides the user-facing interface that allows users to interact with the file management system. Its primary goal is to offer a user-friendly and intuitive way to manage their files and perform authorized operations.
   * **Roles:**
     + **User Authentication Interface:**
       - Provides login forms for password-based authentication.
       - Guides users through the two-factor authentication process.
       - Handles user registration and password reset functionalities.
     + **File Browsing and Navigation:**
       - Allows users to navigate through the file system hierarchy.
       - Provides functionalities for creating, renaming, and deleting folders.
     + **File Management Interface:**
       - Enables users to upload and download files securely.
       - Provides options for viewing and editing file metadata (where permitted).
       - Facilitates file sharing with other users or groups, adhering to the access control policies.
       - Offers search functionality to locate files based on name, content (if indexed), or metadata.
     + **Access Control Management Interface (Admin Focused):**
       - Provides administrative interfaces for managing users, groups, and their permissions on files and folders.
       - Allows administrators to define and enforce access control policies.
     + **Security Settings Interface (User Focused):**
       - Allows users to manage their security settings, such as enabling/disabling 2FA and potentially viewing their recent activity.
     + **Presentation Layer:**
       - Handles the visual presentation of data and user interactions.
       - Ensures a responsive and accessible user experience.
     + **Communication with Backend:**
       - Communicates with the Core Backend & Security Module via the defined API to perform all file management operations.
2. **Security Monitoring & Management Module:**
   * **Purpose:** This module focuses on proactively monitoring the system for security threats, analysing audit logs, and providing tools for administrators to manage the security posture of the file management system.
   * **Roles:**
     + **Real-time Threat Detection:**
       - Monitors system logs and network traffic for suspicious activities and potential security breaches.
       - Integrates with intrusion detection/prevention systems (IDS/IPS) if applicable.
       - Alerts administrators to potential security incidents.

* + - **Audit Log Analysis:**
      * Provides tools for administrators to search, filter, and analyse audit logs to identify security incidents, track user activity, and ensure compliance.
      * Generates reports based on audit data.
    - **Security Configuration Management:**
      * Provides interfaces for administrators to configure security settings, such as password policies, 2FA enforcement, and access control rules.
      * Manages and updates malware scanning definitions.
    - **Vulnerability Monitoring:**
      * Integrates with vulnerability scanning tools to identify potential weaknesses in the system.
      * Provides alerts and recommendations for patching and remediation.
    - **Incident Response Tools:**
      * Offers tools and workflows to assist administrators in responding to security incidents, such as isolating affected accounts or revoking access.
    - **Reporting and Analytics:**
      * Generates reports on system usage, security events, and compliance status.
      * Provides insights into potential security risks and areas for improvement.

## 3. Functionalities: -

Here are the key functionalities for each of the three modules of our EnterpriseGrade Secure File Management System, providing examples for better clarity: - **I. Core Backend & Security Module:**

* **Authentication:** Secure login (password hashing), Two-Factor Authentication (TOTP).

*Example: User logs in with password and a code from their authenticator app.*

* **Authorization:** Granular access control based on roles/groups.

*Example: "Marketing Group" has 'read' access to the "Public Docs" folder.*

* **Secure Storage:** Encryption at rest (AES-256).

*Example: Files are unreadable on the server without decryption.*

* **File Operations:** Secure upload (input validation), secure download (authenticated).

*Example: System checks filename for malicious characters during upload.*

* **Threat Detection:** Basic buffer overflow protection, malware scanning.

*Example: Uploaded .exe files are scanned for viruses.*

* **Audit Logging:** Tracking user logins and file access.

*Example: Log shows "User A" accessed "Sales Report" at 10:15 AM.*

1. **User Interface (UI) Module:** 
   * **Authentication:** Login form, 2FA prompt.

*Example: Simple interface for entering username, password, and 2FA code.*

* + **File Management:** Browse folders, upload/download files.

*Example: Users can drag and drop files to upload.*

* + **Sharing:** Share files/folders with specific permissions.

*Example: User shares a document with a colleague allowing them to*

*'edit'.*

* + **Search:** Find files by name.

*Example: Typing "budget" shows all files with "budget" in the name.*

1. **Security Monitoring & Management Module:** 
   * **Monitoring:** Detect suspicious login attempts.

*Example: Alert admin if multiple failed logins occur from one IP.*

* + **Audit Analysis:** View basic audit logs.

*Example: Admin can see who has accessed a particular file.*

* + **Configuration:** Set password complexity rules.

**.**

*Example: Admin can require passwords to have at least 8 characters.*

* + **Incident Response:** Basic account locking.

*Example: Admin can temporarily lock a user account.*

## 4. Technology Used: -

* **Programming Languages:**

**Backend:** o **Python:** For rapid development, extensive libraries (e.g., Django, Flask).

* + **Java:** For robust, scalable enterprise applications (e.g., Spring).
  + **Go (Golang):** For high performance and concurrency.

**Frontend:**

* + **JavaScript:** Essential for interactive web UI.
  + **HTML, CSS:** For structure and styling. o **Frontend Framework/Library:** React, Angular, or Vue.js for a structured UI.
* **Libraries and Tools:**

**Backend:** o **Web Framework:** Django, Flask (Python), Spring (Java), Gin/Echo (Go). o **Database ORM:** SQLAlchemy (Python), Hibernate (Java), GORM (Go). o **Database:** PostgreSQL, MySQL, or a NoSQL option like MongoDB.

* + **Authentication:** bcrypt or argon2 for password hashing, libraries for JWT and TOTP (2FA). o **Encryption:** Libraries like cryptography (Python) or built-in Java/Go crypto libraries.
  + **Malware Scanning:** Libraries to interact with ClamAV or other anti-malware tools.
  + **Logging:** Standard logging libraries for the chosen language.

**Frontend:**

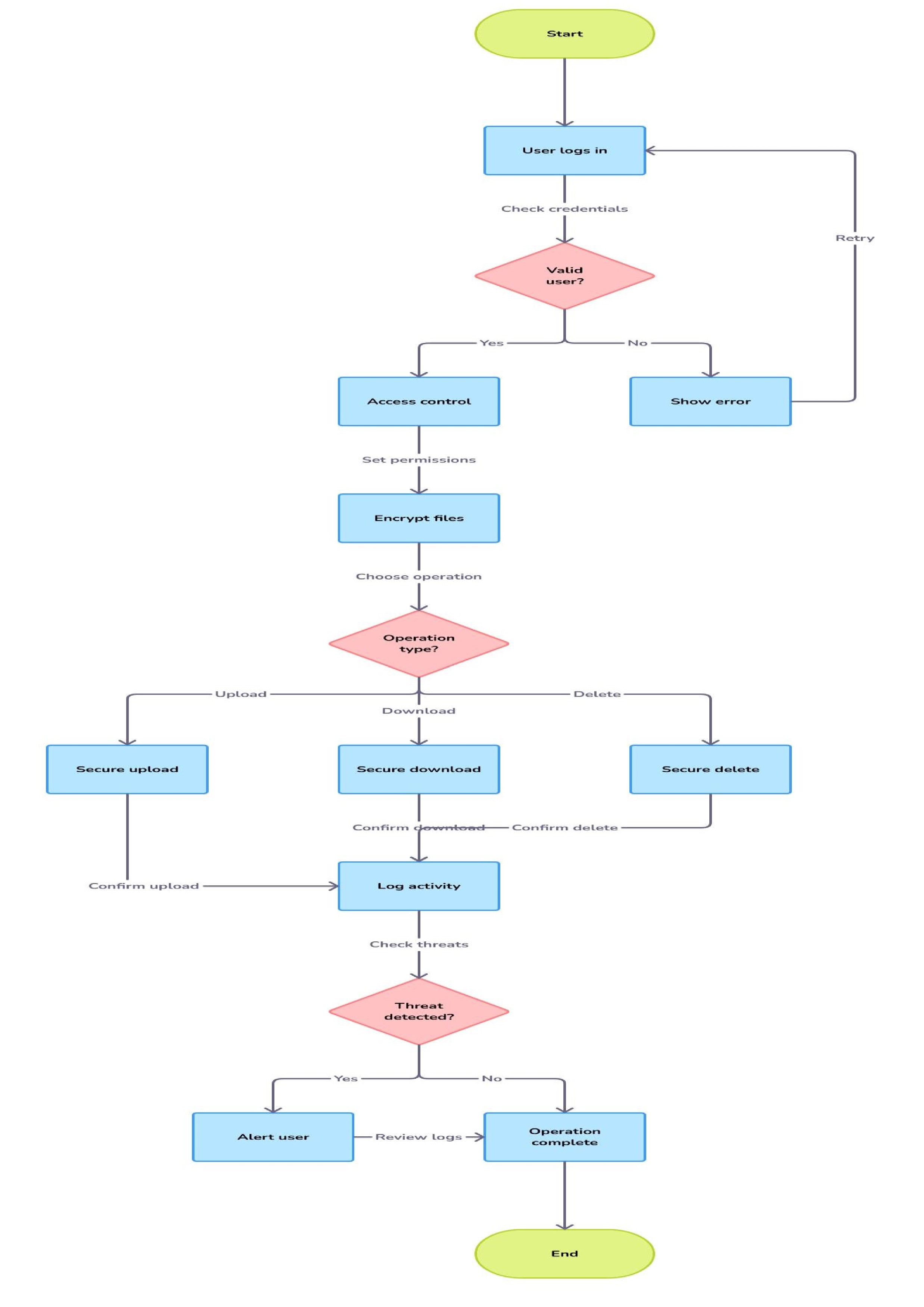
* + **UI Libraries:** Material UI, Bootstrap, Ant Design.
  + **HTTP Client:** Axios or Fetch API.

**General:** o Static Code Analysis: Linters for chosen languages.

* **Other Tools:**
* **Version Control:** Git with GitHub, GitLab, or Bitbucket.
* **Containerization:** Docker and Docker Compose.
* **CI/CD:** Jenkins, GitLab CI/CD, GitHub Actions.
* **Cloud Platform (Optional):** AWS, GCP, Azure.

**5**

**. Flow Diagram:**



## 6. Revision Tracking on GitHub

* **Repository Name:** **[Enterprise-Grade-File-Management-System**]
* **GitHub Link:** [[https://github.com/kumarankit79/Enterprise-Grade-FileManagement-System]](https://github.com/kumarankit79/Enterprise-Grade-File-Management-System)

## 7. Conclusion and Future Scope: -

**Conclusion: -**

This project outlines the design and implementation of a robust and secure enterprise-grade file management system. The system prioritizes data security through strong multi-factor authentication, granular access control mechanisms, and end-to-end encryption. By implementing secure file operations and integrating proactive threat detection and mitigation strategies, including buffer overflow protection and malware scanning, the system aims to provide a safe and reliable environment for managing critical digital assets.

The modular design, encompassing a core backend with security focus, a userfriendly interface, and a dedicated security monitoring component, allows for a structured development process and facilitates future enhancements. The detailed functionalities defined for each module provide a solid foundation for building a comprehensive and secure file management solution that addresses the key security challenges faced by modern enterprises.

**Future Scope: -**

While this project establishes a strong base, several enhancements and expansions can be considered for future development:

* **Advanced Threat Detection and Prevention: Integrate** more sophisticated malware analysis techniques (e.g., sandboxing, behavioural analysis), intrusion detection and prevention systems (IDPS), and data loss prevention (DLP) capabilities.
* **Enhanced Collaboration Features:** Implement features like real-time co-editing, version control with detailed history tracking, commenting, and workflow automation to improve team collaboration.
* **Integration with Enterprise Systems:** Seamlessly integrate with existing enterprise identity management systems (e.g., LDAP, Active Directory), content management systems, and other business applications.
* **Scalability and High Availability:** Architect the system for horizontal scalability and implement redundancy mechanisms to ensure high availability and fault tolerance.
* **Advanced Search Capabilities:** Implement full-text indexing and advanced search filters to enable users to quickly and efficiently locate files based on content and various metadata attributes.
* **Mobile Accessibility:** Develop native mobile applications (iOS and Android) to provide secure access to files on mobile devices.
* **Data Governance and Compliance Features:** Incorporate features to support data governance policies, such as data retention rules, legal hold capabilities, and compliance reporting (e.g., for GDPR, HIPAA).
* **Improved User Experience:** Continuously refine the user interface and user experience based on user feedback and usability testing.
* **Integration with Cloud Storage Providers:** Explore options for integrating with or leveraging existing cloud storage infrastructure while maintaining the system's security and control.
* **AI-Powered Security Analytics:** Utilize artificial intelligence and machine learning techniques to analyse audit logs and identify anomalous behaviour or potential security threats more proactively.

## 8. References: -

These are the following references from where I referred my project :-

* Chat-GPT [https://chatgpt.com/share/67e2e0c2-afac-8009-b8cc-](https://chatgpt.com/share/67e2e0c2-afac-8009-b8cc-8ec1a8d5cb39)

[8ec1a8d5cb39](https://chatgpt.com/share/67e2e0c2-afac-8009-b8cc-8ec1a8d5cb39) o Grok

<https://x.ai/>

* Git & GitHub  [https://github.com/nigamgaurvi/my-operating-systemproject/tree/main](https://github.com/nigamgaurvi/my-operating-system-project/tree/main)
* Visily

<https://www.visily.ai/>

* Grammarly

<https://app.grammarly.com/>

* Visual Studio Code <https://code.visualstudio.com/>

**Snapshots Of Coding Section:-**

