*PROJECT TITLE: FILE MANAGEMENT SYSTEM*

*SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT*

1. **INTRODUCTION**
   1. **PURPOSE**

This Software Requirements Specification (SRS) document provides a comprehensive overview of the requirements for the File Management System (FMS). The purpose of this document is to:

* **Define System Requirements**: Clearly outline both the functional and non-functional requirements necessary for the development of the FMS, ensuring all stakeholders have a shared understanding of the system's capabilities.
* **Establish User Roles**: Identify and describe the various user roles within the system, including Admin, Regular User, and Guest User, along with their specific permissions and interactions with the system. This helps in tailoring the user experience to meet diverse needs.
* **Guide Development and Testing**: Serve as a foundational reference for developers and testers throughout the software development lifecycle. By detailing requirements, this document aids in ensuring that the final product meets the specified criteria and functions as intended.
* **Support Project Management**: Provide a clear framework for project planning and management, enabling effective resource allocation, timeline estimation, and risk assessment. This ensures that the project is delivered on time and within budget.
* **Facilitate Communication**: Act as a communication tool among stakeholders, including project managers, developers, testers, and end-users. By outlining requirements, it promotes transparency and fosters collaboration among all parties involved.
* **Ensure Compliance and Security**: Address key security and compliance requirements, ensuring that the system adheres to relevant data protection regulations and best practices. This is crucial for maintaining user trust and safeguarding sensitive information.
* **Enhance User Experience**: Identify usability requirements that focus on creating an intuitive and user-friendly interface. This ensures that users can efficiently manage their files, thereby enhancing overall satisfaction and productivity.

By providing a detailed description of the File Management System's requirements, this document aims to lay the groundwork for a robust, secure, and user-centric system that meets the needs of its users while supporting the organization's operational goals.

**1.2 SCOPE**

The File Management System (FMS) is designed to provide a comprehensive solution for users to effectively manage their files in a secure, efficient, and user-friendly manner. The scope of the FMS includes the following key functionalities and features:

* **File Upload and Storage**: Users will have the ability to upload files of various types and sizes, ensuring that they can easily store their documents, images, and other media. The system will support multiple file uploads simultaneously and provide drag-and-drop functionality for enhanced user experience.
* **User Authentication and Authorization**: The FMS will implement robust user authentication mechanisms, including account registration, login, and password recovery. Role-based access control will ensure that different user roles (Admin, Regular User, Guest User) have appropriate permissions and access levels, safeguarding sensitive information.
* **File Organization**: Users will be able to create, rename, and delete folders to organize their files efficiently. The system will support a hierarchical folder structure, allowing users to categorize their documents for easy retrieval.
* **File Management**: Users can perform various file management operations, including viewing file details, editing file names, and moving files between folders. The interface will be designed to facilitate quick access to commonly used functions.
* **File Sharing and Collaboration**: The FMS will enable users to share files with other users or external collaborators via unique links. Users can set permissions for shared files, determining whether recipients can view or edit the content. This feature will enhance collaboration and streamline workflows.
* **Search and Filtering Capabilities**: The system will include advanced search and filtering options to help users quickly locate files by name, type, or date. This functionality will significantly improve efficiency, especially for users with extensive file collections.
* **Version Control**: The FMS will track changes to files, allowing users to maintain a history of versions. This feature will enable users to revert to previous versions if needed, ensuring data integrity and facilitating collaboration.
* **User Profile Management**: Each user will have a personal profile where they can view and edit their information. The profile will display uploaded files and folders, providing a quick overview of the user's activities within the system.
* **Compliance and Security Measures**: The FMS will incorporate necessary security protocols to protect user data, including encryption of sensitive information and adherence to data protection regulations (e.g., GDPR). Regular security audits and updates will be conducted to ensure ongoing compliance.
* **Platform and Device Compatibility**: The system will be developed as a web-based application, ensuring compatibility across various devices and browsers. This approach will allow users to access their files from desktops, laptops, tablets, and smartphones.

**Exclusions**

The scope of the FMS does not include:

* Advanced file editing features (e.g., document editing tools).
* Integration with third-party applications outside of basic file sharing.
* Offline access capabilities.

**1.3 Definitions, Acronyms, and Abbreviations**

* **FMS**: File Management System
* **UI**: User Interface
* **API**: Application Programming Interface

**1.4 References**

* [Project Charter Document]
* [Wireframes and Design Documents]
* [Technical Architecture Document]

1. **OVERALL DESCRIPTION**

### 2.1 Product Perspective

The File Management System (FMS) will be developed as a standalone web application, ensuring that it is accessible to users through modern web browsers such as Chrome, Firefox, and Safari. This design choice allows for broad accessibility and ease of use across various devices, including desktops, laptops, tablets, and smartphones.

The system will integrate with cloud storage services, such as AWS S3 or Google Cloud Storage, to provide scalable, reliable, and secure file management solutions. This integration will allow users to store large files without compromising performance and will support features such as automatic backups and disaster recovery. The architecture will be designed to facilitate future integrations with other services, enhancing the system’s flexibility.

### 2.2 User Classes and Characteristics

* **Admin**: The Admin user is responsible for managing the overall system. This role includes user management (creating, editing, or deleting user accounts), monitoring system performance, and configuring system settings. Admins will have access to all features of the FMS, allowing them to enforce security policies and ensure compliance with organizational standards.
* **Regular User**: Regular Users will interact with the system to upload, organize, and share their files. They will have the ability to create folders, manage their file storage, and collaborate with others. Regular Users require a straightforward and intuitive interface to facilitate efficient file management, making usability a key focus in the design.
* **Guest User**: Guest Users can access files shared with them but do not have permissions to upload or modify any content. This role is crucial for enabling collaboration with external stakeholders while maintaining security. Guest Users will have a limited interface, providing only the functionalities necessary for viewing shared files.

### 2.3 Operating Environment

The FMS will operate in a web environment, requiring modern web browsers such as Chrome, Firefox, and Safari. This cross-browser compatibility ensures that users can access the application regardless of their preferred browser.

On the server side, the application will utilize Node.js with the Express framework to handle requests and manage server-side logic efficiently. MongoDB will serve as the database, providing a flexible and scalable solution for storing user data and file metadata. This combination of technologies will allow for smooth performance and rapid development cycles.

### 2.4 Design and Implementation Constraints

The development of the FMS will adhere to several critical constraints:

* **Compliance with Data Protection Regulations**: The system must comply with relevant data protection regulations, such as the General Data Protection Regulation (GDPR). This includes implementing necessary data handling practices, ensuring user consent for data processing, and providing users with the ability to manage their data.
* **Responsive Design**: The application will feature a responsive design that adapts to various screen sizes and devices. This ensures that users have a consistent and user-friendly experience, whether they access the system from a desktop computer or a mobile device. Responsive design will involve employing modern CSS frameworks and best practices to optimize layout and functionality.
* **Scalability and Performance**: The system will be designed to handle an increasing number of users and files without compromising performance. This involves planning for load balancing, efficient database queries, and optimizing file storage solutions.
* **Security Measures**: The FMS will incorporate robust security measures, including encryption of data in transit and at rest, to protect user information and files. Regular security assessments will be conducted to identify vulnerabilities and ensure the system remains secure against potential threats.

## 3. Functional Requirements

### 3.1 User Authentication

* **Registration**: Users must be able to register an account by providing necessary information (e.g., username, email, password).
* **Login**: Users must be able to log in using their credentials.
* **Password Recovery**: A password recovery functionality must be available, allowing users to reset their passwords via email verification.
* **Session Management**: The system should maintain user sessions securely, automatically logging out users after a period of inactivity.
* **Two-Factor Authentication (Optional)**: Support for two-factor authentication to enhance account security.

### 3.2 File Upload

* **Multiple File Uploads**: Users can upload multiple files simultaneously to streamline the uploading process.
* **Drag-and-Drop Functionality**: Users should be able to drag and drop files directly into the upload area for convenience.
* **File Size Limitations**: Specify maximum file size limits for uploads (e.g., 100MB).
* **File Type Validation**: The system should validate file types during upload to ensure only permitted formats (e.g., .jpg, .pdf, .docx) are accepted.
* **Progress Indicators**: Display upload progress indicators to inform users about the status of their uploads.

### 3.3 File Management

* **Create Folders**: Users can create new folders to organize files effectively.
* **Rename Files/Folders**: Users can rename existing files and folders to keep their storage organized.
* **Move Files**: Users can move files between folders easily using a drag-and-drop interface or context menu options.
* **Delete Files/Folders**: Users can delete files and folders from their storage, with a confirmation prompt to prevent accidental deletions.
* **File Versioning**: The system should maintain a version history for files, allowing users to revert to previous versions if necessary.

### 3.4 File Sharing

* **Shareable Links**: Users can generate unique links to share files with others.
* **Permission Settings**: Users can set permissions for shared files, specifying whether recipients can view or edit the shared content.
* **Expiration Dates for Links**: Users can set expiration dates for sharing links, after which access will be revoked.
* **Activity Tracking**: The system should track and display activity related to shared files, including who accessed them and when.

### 3.5 File Search and Filter

* **Search Functionality**: Users can search for files by name, type, or date using a search bar.
* **Advanced Filtering Options**: Users should be able to filter files based on type (e.g., images, documents, videos) and other criteria such as size and upload date.
* **Sorting Options**: Users can sort files by name, date modified, and size to facilitate easier navigation.

### 3.6 User Profile Management

* **View Profile**: Users can view their profile information, including username, email, and account status.
* **Edit Profile**: Users can edit their profile information, including changing their password and updating personal details.
* **Display Uploaded Files**: Users can see a list of their uploaded files and folders, providing quick access to their content.
* **Account Deactivation**: Users should have the option to deactivate their accounts if they choose to leave the platform.

### 3.7 Notifications and Alerts

* **File Upload Notifications**: Users receive notifications upon successful uploads or errors during the upload process.
* **Sharing Notifications**: Users are notified when files are shared with them or when their shared links are accessed.
* **System Updates**: Users should receive alerts about system updates, maintenance schedules, and new features.

### 3.8 Help and Support

* **Help Documentation**: The system should provide access to help documentation or FAQs to assist users with common issues.
* **Support Contact**: Users should be able to contact support for assistance, with options for chat, email, or a ticketing system.

## 4. Non-Functional Requirements

### 4.1 Performance Requirements

* **Concurrent User Support**: The system should efficiently handle at least 100 concurrent users without performance degradation, ensuring smooth operation during peak usage times.
* **File Upload Speed**: File uploads should complete within 5 seconds for files under 10MB. For larger files, the system should provide clear feedback on the upload progress and estimated time remaining.
* **Response Time**: The system should respond to user requests within 2 seconds under normal operating conditions to ensure a seamless user experience.
* **Scalability**: The architecture must be scalable to accommodate an increasing number of users and files without requiring significant re-engineering.

### 4.2 Security Requirements

* **Data Encryption**: User data must be encrypted during transmission (using HTTPS) and at rest (using encryption algorithms) to protect sensitive information from unauthorized access.
* **Role-Based Access Control**: Implement role-based access control (RBAC) to restrict access to system functionalities based on user roles, ensuring that only authorized users can perform specific actions.
* **Regular Security Audits**: The system should undergo regular security assessments and penetration testing to identify and mitigate vulnerabilities.
* **Data Backup and Recovery**: Implement automated backup procedures to ensure data integrity and availability, along with clear recovery processes in case of data loss.

### 4.3 Usability Requirements

* **Intuitive User Interface**: The UI should be designed with user experience in mind, featuring a clean layout, easy navigation, and accessible design elements to cater to users of varying technical abilities.
* **Help Documentation**: Provide comprehensive help documentation within the application, including FAQs, tutorials, and troubleshooting guides, to assist users in effectively utilizing the system.
* **Accessibility Compliance**: The system must comply with accessibility standards (e.g., WCAG 2.1) to ensure that users with disabilities can navigate and use the application without barriers.
* **Mobile Responsiveness**: The application should be fully responsive, providing an optimal user experience on devices of all sizes, from desktops to smartphones.

### 4.4 Reliability Requirements

* **System Uptime**: The system should maintain an uptime of 99.9%, ensuring that users can access the application whenever needed.
* **Error Handling**: Implement robust error handling to provide clear and user-friendly error messages, guiding users on how to resolve issues encountered during interactions with the system.

### 4.5 Maintainability Requirements

* **Code Quality**: The codebase should adhere to industry best practices and coding standards to facilitate maintainability and ease of future enhancements.
* **Documentation**: Comprehensive technical documentation must be provided for the system, including architecture diagrams, API documentation, and deployment instructions, to assist developers in maintaining and updating the system.

### 4.6 Interoperability Requirements

* **Integration with Third-Party Services**: The system should be able to integrate with third-party cloud storage services and authentication providers (e.g., Google, Facebook) to enhance functionality and user convenience.
* **API Standards**: Any APIs provided by the system should follow RESTful principles, ensuring they are easy to use and integrate with other applications.

## **5. Use Cases**

### 5.1 Use Case 1: User Registration

**Actors**:

* 1. Regular User

**Description**:  
A new user registers for an account on the File Management System.

**Preconditions**:

* 1. User navigates to the registration page.
  2. User has a valid email address and a secure password.

**Postconditions**:

* 1. User account is created and stored in the database.
  2. User receives a confirmation email.

**Basic Flow**:

* 1. User accesses the registration page.
  2. User enters required information (username, email, password).
  3. User submits the registration form.
  4. System validates the input data.
  5. System creates a new user account.
  6. System sends a confirmation email.
  7. User receives a notification to check their email.

**Alternative Flow**:

* 1. **Invalid Input**: If the user enters invalid data, the system displays an error message.

### 5.2 Use Case 2: File Upload

**Actors**:

* + Regular User

**Description**:  
A registered user uploads files to their account.

**Preconditions**:

* + User is logged into their account.
  + User has files ready for upload.

**Postconditions**:

* + Uploaded files are stored in the user’s account.
  + User can view and manage the uploaded files.

**Basic Flow**:

* 1. User navigates to the file upload section.
  2. User selects files to upload.
  3. System displays a progress indicator.
  4. System validates the file types and sizes.
  5. Upon successful upload, the system stores the files.
  6. User receives a confirmation message.

**Alternative Flow**:

* + **Upload Error**: If there is an error during the upload, the system displays an error message.

### 5.3 Use Case 3: File Management

**Actors**:

* + Regular User

**Description**:  
A user manages their files by organizing, renaming, and deleting them.

**Preconditions**:

* + User is logged into their account.

**Postconditions**:

* + User’s files are organized as per their actions.

**Basic Flow**:

* 1. User navigates to their file list.
  2. User selects a file or folder to manage.
  3. User chooses to rename, move, or delete the selected item.
  4. System updates the file structure accordingly.
  5. User receives confirmation of the action taken.

**Alternative Flow**:

* + **Deletion Confirmation**: If the user attempts to delete a file, the system prompts for confirmation.

### 5.4 Use Case 4: File Sharing

**Actors**:

* + Regular User

**Description**:  
A user shares files with others by generating shareable links.

**Preconditions**:

* + User is logged into their account.
  + User has files to share.

**Postconditions**:

* + Shared files are accessible to designated recipients.

**Basic Flow**:

* 1. User selects a file to share.
  2. User generates a shareable link.
  3. User sets permissions (view/edit) for the link.
  4. System provides the link to the user.
  5. User shares the link with others.

**Alternative Flow**:

* + **Link Expiration**: User can set an expiration date for the link.

### 5.5 Use Case 5: File Search and Filtering

**Actors**:

* + Regular User

**Description**:  
A user searches for and filters files within their account.

**Preconditions**:

* + User is logged into their account.

**Postconditions**:

* + User finds files that match their search criteria.

**Basic Flow**:

* 1. User accesses the search bar.
  2. User enters search criteria (name, type).
  3. System retrieves and displays matching files.
  4. User applies filters as needed.
  5. User views the filtered list of files.

**Alternative Flow**:

* + **No Results Found**: If no files match the criteria, the system displays a relevant message.

### 5.6 Use Case 6: User Profile Management

**Actors**:

* + Regular User

**Description**:  
A user manages their profile information.

**Preconditions**:

* + User is logged into their account.

**Postconditions**:

* + User's profile is updated with new information.

**Basic Flow**:

* 1. User navigates to the profile management section.
  2. User views current profile information.
  3. User edits details (e.g., password, email).
  4. User submits changes.
  5. System updates the profile and confirms the changes.

**Alternative Flow**:

* + **Invalid Input**: If the user enters invalid information, the system displays an error message.