

CODEBOX



*A project report submitted to
Rajiv Gandhi Pradyogiki Vishwavidhyalaya, Bhopal
in partial fulfillment for the award of
the degree of
Bachelor of Technology
in
Information Technology*

DEPARTMENT OF INFORMATION TECHNOLOGY

**SUSHILA DEVI BANSAL COLLEGE OF TECHNOLOGY
INDORE- 453331**

2023 - 2024

CODEBOX



*A project report submitted to
Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal
in partial fulfillment for the award of
the degree of
Bachelor of Technology
in
Information Technology*

PROJECT GUIDE

Mrs.Namrata Sharma Bhatt

SUBMITTED BY

Hemant Kumar Kashyap(0829IT211027)

Pooja Singhad (0829IT211046)

DEPARTMENT OF INFORMATION TECHNOLOGY

SUSHILA DEVI BANSAL COLLEGE OF TECHNOLOGY

INDORE- 453331

2023 - 2024

ACKNOWLEDGEMENT

I am thankful to all faculty members, providing their valuable time and guidance elaborating view of studying the project details and getting the right vision for its implementation. I pay my immense gratitude to Prof. Mrs. Namrata Sharma Bhatt for her continuous and deliberate discussion on the topic and indeterminable burden taken by her in helping me throughout conducting the project.

- Hemant Kumar Kashyap (08291T211027)
- Pooja Singhad (08291T211046)

SUSHILA DEVI BANSAL COLLEGE OF TECHNOLOGY
INDORE, 453331



CERTIFICATE

This is to certify that **HEMANT KUMAR KASHYAP(0829IT211027), Pooja Singhad (0829IT211046)**. have completed their project work, titled “**CODEBOX**” as per the syllabus and have submitted a satisfactory report on this project as a part of fulfillment towards the degree of “**BACHELOR OF TECHNOLOGY**” (Information Technology) from **RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL.**

HEAD OF THE DEPARTMENT

PROJECT GUIDE

DIRECTOR

SUSHILA DEVI BANSAL COLLEGE OF TECHNOLOGY
INDORE, 453331



CERTIFICATE

This is to certify that **HEMANT KUMAR KASHYAP(0829IT211027), Pooja Singhad (0829IT211046)** have completed their project work, titled “**CODEBOX**” as per the syllabus and have submitted a satisfactory report on this project as a part of fulfillment towards the degree of “**BACHELOR OF TECHNOLOGY**” (Information Technology) from **RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL.**

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Our project revolves around the development of an advanced desktop application that aims to revolutionize the way users manage files. Similar in concept to GitHub, our application offers a rich array of features that cater to every aspect of file management and user interaction, promising a robust, efficient, and highly user-centric platform.

Detailed Features:

File Management:

File Storage: Users can securely store their files, ensuring data integrity and accessibility,**File Upload and Deletion:** Uploading and deleting files are simplified processes, allowing users to effortlessly manage their digital assets.

Download:

Efficient Download: Downloading shared files is a seamless process, promoting effective collaboration.

Notifications:

Admin Announcements: Administrators can post notifications and announcements, keeping users informed about updates, changes, or important events.

Feedback Mechanism:

User Feedback: A dedicated feedback feature allows users to provide comments and suggestions, driving continuous improvement ,**Feedback Analysis:** Administrators can review and analyze feedback to enhance the application.

Note-Taking: Users can create and manage personal notes within the application .

User Profiles:

Personalized Profiles: Each user has a customizable profile where they can manage personal information, preferences, and settings.

Our desktop application places great emphasis on a user-friendly interface, top-tier security, and a seamless user experience. We aim to provide a comprehensive solution that caters to the diverse needs of modern users. This project represents an evolution in digital workspace management and is poised to become an indispensable tool for professionals and individuals alike.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF TABLES	ii
	LIST OF FIGURES	iii
	LIST OF SYMBOLS	iv
1.	Introduction	
1.1.	Purpose	
1.2.	Scope	
1.3.	Problem in existing system	
1.4.	Statement of Problem	
2.	System Requirement Analysis	
2.1	Introduction	
2.1.1	Purpose	
2.1.2	Document Conventions	
2.1.3	Intended Audience and Reading Suggestions	
2.1.4	Product Scope	
2.1.5	References of SRS	
2.2	Overall Description	
2.2.1	Product Perspective	
2.2.2	Product Functions	
2.2.3	User Classes and Characteristics	
2.2.4	Operating Environment	
2.2.5	Design and Implementation Constraints	
2.2.6	User Documentation	
2.2.7	Assumptions and Dependencies	
2.3	External Interface Requirements	
2.3.1	User Interfaces	
2.3.2	Hardware Interfaces	
2.3.3	Software Interfaces	

2.3.4 Communications Interfaces

2.4 Functional Requirement

2.4.1 System Feature 1

2.4.2 System Feature 2 (and so on)

2.5 Nonfunctional Requirements

2.5.1 Performance Requirements

2.5.2 Safety Requirements

2.5.3 Security Requirements

2.5.4 Software Quality Attributes

2.6 Project Plan

2.6.1 Team Members

2.6.2 Division of Work

2.6.3 Time Schedule

3. Analysis

3.1. Methodology Used (*If Structural*)

3.2. Use Case diagram (*Also include use case specifications*)

3.3. ER Model

DFD

3.4. Process Specification

3.5. CFD

3.6. STD

4. Design

4.1. Architectural Design

4.1.1. System Architecture Diagram

4.1.2. Description of Architectural Design

4.2. Database Design

4.2.1. Data Dictionary

4.2.2. Normalization

4.3. Component Design

4.3.1. Flow Chart

4.4. Interface Design

4.4.1. Screenshots

INTRODUCTION

1.1 PURPOSE

The purpose of your file management system project is to provide users, especially students and professionals, with a user-friendly and efficient solution for managing digital files and note-taking. The key purposes and goals of your project include:

1. **Effective File Management:** To offer a convenient and organized way for users to upload, categorize, and manage their digital files. This facilitates better data organization and access.
2. **Efficient Note-Taking:** To provide users with a platform to create, edit, and save notes. This helps users capture and organize their ideas, information, and study materials.
3. **Enhanced Productivity:** To contribute to users' productivity by making it easier to find and access their files and notes, ultimately saving time and reducing frustration.
4. **Security and Privacy:** To prioritize the security and privacy of user data by implementing access control and robust security measures.
5. **Collaboration Support:** To enable users to collaborate with others by sharing files and notes, promoting teamwork and knowledge exchange.

1.2 SCOPE

The scope of your file management system project outlines the boundaries and specific functionalities that the project will encompass. It defines what is included and what is not. Here's the scope of your project:

Project Scope:

1. **User Registration and Authentication:**
 - Users can register for an account with a username and password.
 - The system enforces user authentication for security.
2. **File Management:**
 - Users can upload, categorize, and manage digital files (documents, images, etc.).
 - Files can be organized within the system.
3. **File Download and Access:**
 - Users can download their uploaded files or access them as needed.
 - Access control mechanisms are implemented to ensure that users can only access their own files.
4. **Note Creation and Editing:**
 - Users can create, edit, and save notes within the system.
 - The system provides a basic text editor for note management.
5. **Search and Retrieval:**
 - Users can search for files and notes using criteria like file name, type, keywords, and dates.

- The system returns search results efficiently.

6. User Profile Management:

- Users can view and update their profile information, including their username and password.

7. Security and Access Control:

- Robust security measures are implemented to protect user data.
- Passwords are securely stored using hashing and salting techniques.
- Users have varying levels of access to their own data.

8. Collaboration and Sharing (Optional):

- Users can collaborate by sharing files and notes with other users.
- Collaboration features support teamwork and information exchange.

9. User-Friendly User Interface:

- The system offers a user-friendly interface created using Java Swing, ensuring an intuitive and visually appealing user experience.

10. Data Backup and Recovery (Optional):

- The system includes data backup mechanisms to prevent data loss.
- Recovery procedures are in place to restore the system in case of data corruption or hardware failures.

11. Educational Use (Optional):

- The system is designed to support educational institutions by providing tools for educators and students to manage course materials and collaborate.

12. User Documentation:

- User documentation, such as user guides and manuals, is available to assist users in using the system effectively.

The project's scope focuses on creating a comprehensive file management system with note-taking capabilities. It emphasizes user-friendliness, data security, and efficient organization, making it valuable for individual users and potentially educational institutions. Additional features, such as collaboration and data backup, are optional and can be included based on project priorities and requirements.

1.3 PROBLEM IN EXISTING SYSTEM

In the traditional approach to file management and collaboration, users heavily rely on a combination of disparate tools and methods to meet their needs. File management typically involves local storage on personal devices, making it challenging to access files from multiple locations and collaborate effectively. User search is limited to personal networks and physical directories, and file sharing relies on physical media or email attachments. Notifications are typically communicated through email, phone calls, or in-person discussions, which can lead to delays and inefficiencies. Feedback is often conveyed verbally or through email, lacking a structured system for user input.

Note-taking may involve physical notebooks or various digital note-taking applications, and user profiles are typically not well-defined, especially in personal file management.

1.4 STATEMENT OF PROBLEM

In the increasingly digital world, individuals, students, and professionals face challenges related to efficient file management, note-taking, and collaboration on digital resources. The existing solutions often lack user-friendliness, robust security, or the flexibility to cater to diverse needs. Users struggle with disorganized files, limited access control, and inadequate tools for note-taking and knowledge sharing.

This project aims to address these issues by developing a user-friendly file management system with integrated note-taking capabilities. The system will prioritize data security, offer advanced search features, and, optionally, support collaboration. By providing a comprehensive solution, the project seeks to enhance productivity, organization, and knowledge management for individual users and, optionally, educational institutions.

Key Problem Areas to Address:

1. **Inefficient File Management:** Users struggle with disorganized files, making it challenging to locate and access resources when needed.
2. **Limited Note-Taking Tools:** Existing solutions lack efficient and user-friendly note-taking features, hindering information capture and organization.
3. **Data Security and Access Control:** Users require robust data security measures to protect their files and notes, with the ability to control access to their data.

SYSTEM REQUIREMENT ANALYSIS

2.1 INTRODUCTION

System Requirement Analysis is a pivotal phase in the life cycle of system and software development. It is the systematic approach to gathering, documenting, analyzing, and managing the needs and expectations of stakeholders in order to define a clear roadmap for project success. This introductory overview encapsulates the significance, stages, collaborative nature, and functions of requirement analysis in ensuring that software and systems are effectively designed and deployed to meet business and user objectives.

2.1.1 PURPOSE

The purpose of your file management system project is to provide an organized and efficient way for users to manage their digital files and take notes. Here are some key purposes and benefits of your project:

1. **File Organization:** Users can easily upload, categorize, and manage their files in one central location, making it convenient to find and access their documents, images, and other digital assets.
 2. **Efficient File Retrieval:** The search functionality allows users to quickly locate specific files based on file names, types, keywords, or other criteria, saving time and reducing frustration.
 3. **Data Security:** The system ensures that user files and data are stored securely. User authentication and access control measures protect sensitive information.
 4. **Collaboration:** Users can share files with others, promoting collaboration and teamwork. This feature is particularly useful in an educational setting.
 5. **Note-Taking:** Users can create and store notes within the system, allowing them to jot down ideas, reminders, and important information in an organized manner.
 6. **Accessibility:** The desktop application ensures that users can access their files and notes from their own devices, enhancing accessibility and flexibility.
 7. **User-Friendly Interface:** The user interface is designed to be user-friendly, making it easy for individuals, including students and educators, to navigate and utilize the system.
 8. **Learning Opportunity:** For you and your team, this project serves as a valuable learning experience. You gain practical knowledge in software development, database management, and user interface design.
 9. **Contribution to Academic Environment:** If used in an educational setting, your project can contribute to the efficiency of file management, note-taking, and collaboration among students and faculty, enhancing the overall academic environment.
- Overall, the project's purpose is to simplify and enhance the way users manage their digital resources and notes, providing a valuable tool for organizing, accessing, and collaborating on files and information.

2.1.2 DOCUMENT CONVENTIONS

A Java Swing-based desktop application with MySQL database integration designed for file management and note-taking. The project aims to provide users with a user-friendly interface for uploading, organizing, searching for files, and creating and saving notes. It also focuses on user security, efficient data access, and an organized approach to digital resource management. This project contributes to improved file organization, efficient data retrieval, and enhanced user productivity, particularly in an educational context. It provides a valuable learning experience for the development team and contributes to the academic environment.

2.1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

Intended Audience:

The intended audience for CodeBox project can be diverse, as the system's features have broad applicability. The primary audience segments include:

1. End Users: Users who will directly interact with the Codebox , including students, professionals, or individuals looking for a better way to organize digital files and take notes.
2. Educational Institutions: Educational institutions, including colleges, universities, and schools, that may consider adopting the system to facilitate better file management, collaboration, and note-taking among students and faculty.
3. Educators and Faculty: Teachers, professors, and instructors who can use the system to share resources and manage course materials, enhancing the teaching and learning experience.
4. System Administrators: IT administrators responsible for deploying and maintaining the application within an educational institution or other organizations.
5. Developers and Technical Personnel: Software developers and technical teams interested in the project's technical aspects and implementation details.

Reading Suggestions:

1. User Documentation: Provide user guides and manuals to help end users navigate and use the application effectively. These documents should include step-by-step instructions on tasks such as file upload, search, and note creation.
 2. Technical Documentation: For developers and technical personnel, offer detailed technical documentation that covers the system architecture, database schema, API endpoints (if applicable), and coding conventions. This documentation can help other developers understand the project's structure and contribute to its development or maintenance.
 3. Educational Institutions: When targeting educational institutions, create a comprehensive document or proposal that highlights the benefits of using your system in an educational setting. Explain how it can enhance collaboration, file management, and note-taking among students and faculty.
 4. Training Materials: Develop training materials for system administrators who will be responsible for deployment, configuration, and maintenance. These materials should cover installation procedures, server requirements, and troubleshooting guidelines.
 5. Online Resources: Consider creating a project website or a knowledge base with FAQs, tutorials, and user support forums. These resources can be beneficial for all user categories and can provide ongoing support.
 6. Case Studies and Use Cases: Present case studies and real-world use cases that demonstrate how your file management system can address specific needs or challenges. These can be especially useful for educational institutions and educators looking for practical solutions.
- By tailoring your documentation and reading suggestions to each audience segment, you can effectively address the needs and interests of a diverse range of users and stakeholders.

2.1.4 PRODUCT SCOPE

Project Name: CODEBOX

Product Scope:

1. User Registration and Authentication:

- The system allows users to register for an account and authenticate themselves securely.

2. File Management:

- Users can upload, categorize, and manage their digital files, including documents, images, and other file types.

- The system provides file upload, download, and deletion functionalities.

3. Notes Feature:

- Users can create, edit, and save notes within the system for personal or collaborative use.

4. Search and Retrieval:

- Users can search for files and notes using various search criteria, such as file name, type, keywords, and date.

- The system returns search results efficiently and accurately.

5. User Profile Management:

- Users have the ability to view and update their profile information, including username and password.

6. Security and Access Control:

- The system enforces user authentication and access control to protect user data.
- Passwords are securely stored using hashing and salting techniques.

7. User-Friendly User Interface:

- The user interface is designed using Java Swing to provide a user-friendly and visually appealing experience.

- It offers intuitive navigation and easy access to features.

8. Data Backup and Recovery:

- The system provides data backup mechanisms to prevent data loss and supports recovery in case of data corruption or hardware failures.

9. Collaboration Features:

- Users can share files and notes with other users, facilitating collaboration and teamwork.

10. Documentation and Training:

- The project includes user documentation, such as user guides and manuals, to assist users in using the system effectively.

- Training materials may be provided for system administrators.

11. Performance Optimization:

- The system is optimized for efficient data access and responsiveness, even when handling a large number of files and notes.

12. Educational Use: (Optional)

- The system is suitable for use in educational settings, allowing educators and students to manage course-related materials and notes efficiently.

Out of Scope:

1. Email or messaging features.
2. Integration with external storage services (e.g., cloud storage).
3. Mobile application development (if not part of the project scope).

By defining the product scope, you provide a clear and shared understanding of the project's objectives and deliverables, helping to ensure that the project stays on track and meets its intended goals.

2.1.5 REFERENCES OF SRS

2.2 OVERALL DESCRIPTION

2.2.1 PRODUCT PERSPECTIVE

1. System Overview:

- The CODEBOX is a standalone desktop application developed using Java Swing for the user interface and MySQL for the database. It operates within the user's local environment, independent of external systems.

2. Interaction with External Systems:

- The system interacts with the MySQL database to store and retrieve user data, files, and notes. This interaction is vital for data storage and retrieval operations.

3. Standalone Nature:

- The system operates as a standalone application on the user's desktop or computer, offering a self-contained solution for file management and note-taking.

4. User Workstations:

- Users interact with the system through their individual workstations, where the application is installed. The system is responsible for providing a user-friendly interface for users on these workstations.

5. Data Backup and Recovery:

- Data backup and recovery mechanisms are integral parts of the system, ensuring the security and availability of user data, even in cases of data loss or corruption.

6. User-Generated Data:

- Users generate and manage data within the system, including files and notes, which are stored and organized by the system.

7. Security:

- The system emphasizes data security through user authentication, password hashing, and access control measures. This ensures that only authorized users can access their data.

8. Collaboration and Sharing:

- Users can collaborate by sharing files and notes with other users, facilitating teamwork and information exchange.

9. User Documentation and Training:

- The system provides user documentation, such as user guides and manuals, to help users understand and use the system effectively. Additionally, training materials may be available for system administrators.

10. Scalability:

- While the current version is designed for individual users, future versions may consider adding collaborative features and networked file sharing for larger-scale deployments in educational or organizational settings.

2.2.2 PRODUCT FUNCTIONS

The product functions of CODEBOX project are the specific capabilities and features that the system will provide to its users. These functions are at the core of your project's purpose. Here are the key product functions for your file management system:

1. User Registration and Authentication:

- Users can create an account by registering with their credentials (username and password).
- Authentication ensures that only authorized users can access the system.

2. File Upload and Management:

- Users can upload files from their local devices, categorize them, and organize them within the system.
- The system provides options for adding file descriptions or tags to aid in organization.

3. File Download and Access:

- Users can download their uploaded files or access them as needed.
- Access control mechanisms ensure that users can only retrieve their own files.

4. Note Creation and Editing:

- Users can create and edit notes within the system.
- The text editor allows basic formatting and organization of notes.

5. Search and Retrieval:

- A search feature allows users to find files and notes quickly using criteria such as file name, type, keywords, and dates.
- Search results are presented efficiently, aiding in data retrieval.

6. User Profile Management:

- Users can view and update their profile information, including their username and password.

7. Security and Access Control:

- Robust security measures are implemented to protect user data.

- Passwords are securely stored using hashing and salting techniques.
 - Users have varying levels of access to their own data, and security measures prevent unauthorized access.
8. Collaboration and Sharing:
- Users can share files and notes with other users.
 - Collaboration features support teamwork and information exchange among users.
9. User-Friendly User Interface:
- The system offers a user-friendly interface created using Java Swing, ensuring an intuitive and visually appealing user experience.
10. Data Backup and Recovery:
- The system includes data backup mechanisms to prevent data loss.
 - Recovery procedures are in place to restore the system in case of data corruption or hardware failures.
11. User Documentation:
- User documentation, such as user guides and manuals, is available to assist users in using the system effectively.

2.2.3 USER CLASSES AND CHARACTERISTICS

In CODEBOX project, it's essential to define the user classes and their characteristics to tailor the system to the needs of different users. Here are the primary user classes and their characteristics:

1. Individual Users:

- Characteristics:
 - Individuals from various backgrounds, including students, professionals, and general users.
 - Need a solution for personal file management, note-taking, and data organization.
 - May have varying technical proficiency levels, so the system should be user-friendly

2. System Administrators:

- Characteristics:
 - Technical personnel responsible for system deployment, configuration, and maintenance.
 - Need access to system settings and tools for data management.
 - Focus on ensuring system availability and data security.

3. Collaborative Users (Optional):

- Characteristics:
 - Users who want to collaborate on files or notes with others, such as students working on group projects.
 - Require features for sharing and editing files or notes with multiple collaborators.
 - Collaboration features enhance teamwork and productivity.

4. Technical Support (Optional):

- Characteristics:

- Technical support personnel or helpdesk staff.
- May need access to user accounts to assist with troubleshooting.
- Focus on providing assistance and resolving user issues.

2.2.4 OPERATING ENVIRONMENT

1. Hardware Requirements:

- User Workstations: The system runs on individual user workstations, which should meet the following minimum requirements:

- A modern computer with a multi-core processor.
- Sufficient RAM (typically 4 GB or more).
- Adequate storage space for the application and user data.
- A display with a resolution that supports the Java Swing user interface.
- A standard keyboard and mouse for user input.
- Database Server: If hosting the MySQL database server separately, it should have:
 - A dedicated server or hosting environment.
 - Adequate storage space for the database and backups.
 - Sufficient processing power and memory based on anticipated data load.

2. Software Requirements:

- Operating System: The system is compatible with popular operating systems, including Windows, macOS, and Linux.

- Java Runtime Environment (JRE): Users must have a compatible JRE installed on their workstations to run the Java Swing-based application.

- MySQL Database Server: The server hosting the MySQL database should have the MySQL DBMS installed and properly configured.

- Web Browsers (Optional): If the system provides web-based documentation or support, it should be compatible with standard web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.

- Security Software (Optional): Ensure compatibility with antivirus and firewall software commonly used on user workstations.

3. Network Requirements:

- The system operates on a local network or over the internet, depending on your deployment.
- Users should have a stable and reliable internet connection for remote access (if applicable).

4. Deployment Environment:

- The system is primarily intended for deployment within educational institutions, but it can also be used by individual users.

- Ensure compatibility with the institution's network infrastructure and policies.

5. Scalability Considerations:

- The system should be designed to handle potential growth in the number of users and the volume of data.
- Database architecture should allow for scalability.

By defining the operating environment, you ensure that your file management system can function effectively within the specified hardware and software parameters. This information also helps users and system administrators understand the prerequisites for installing and running the application.

2.2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

Design and implementation constraints in CODEBOX outline the limitations, guidelines, and factors that can impact the design, development, and deployment of your file management system. These constraints help manage expectations and guide the project's development process. Here are some design and implementation constraints for your project:

1. Technology Stack:

- The project is limited to using Java for the user interface (Java Swing) and MySQL for the database. Compatibility with these technologies is crucial for design and implementation.

2. Operating System Compatibility:

- The system is designed to run on various operating systems, including Windows, macOS, and Linux. It must be tested and optimized for compatibility with each of these platforms.

3. Hardware Resources:

- The system's performance is constrained by the hardware resources available on individual user workstations. The system should be optimized to operate efficiently within these constraints.

4. Internet Connectivity (Optional):

- If the system supports remote access or collaboration over the internet, users are required to have a stable internet connection. Slow or unreliable connections may impact performance.

5. Scalability:

- The system's design should consider scalability for future enhancements and increased user loads. However, it is constrained by the existing architecture.

6. Security Measures:

- Implementation must adhere to strict security constraints, including robust password hashing, access control, and encryption, to protect user data and ensure compliance with privacy regulations.

7. User Interface Design:

- Design constraints include adhering to Java Swing for the user interface, which may limit the extent of modern design elements that can be incorporated.

8. Documentation and Training:

- Constraints involve creating user documentation and training materials that are clear and accessible for users with varying technical proficiency levels.

9. Database Constraints:

- The design must conform to the MySQL database management system, and any database-related constraints must be addressed to ensure optimal database performance and data integrity.

10. Deployment Environment:

- The system is primarily designed for deployment within educational institutions, and the implementation should align with the institution's network infrastructure and policies.

11. Compliance and Regulations:

- Adherence to legal and regulatory constraints, such as data privacy and copyright laws, is essential. The system's design and implementation should ensure compliance with relevant legal requirements.

12. Budget and Resource Constraints:

- The project may be constrained by budget limitations and resource availability, influencing the extent of features and functionalities that can be implemented.

Identifying these constraints from the outset of your project is crucial for managing expectations, guiding development efforts, and ensuring that the project is completed successfully while adhering to practical and technical limitations.

2.2.6 USER DOCUMENTATION

User documentation is essential for helping users understand how to CODEBOX effectively. It provides guidance on various aspects of the system, from installation to using its features. Here's an outline of the user documentation that you should consider providing:

1. Installation Guide:

- Provide step-by-step instructions for installing the file management system on different operating systems, such as Windows, macOS, and Linux.

- Include system requirements and any prerequisites, like the Java Runtime Environment (JRE).

2. User Manual:

- A comprehensive user manual should cover all essential aspects of using the system. This can be organized into sections such as:

- Logging in and user profile management.
- Uploading, managing, and downloading files.
- Creating and editing notes.
- Searching for files and notes.
- Security and access control.
- Collaboration and sharing.
- Tips and tricks for efficient use.

3. Frequently Asked Questions (FAQs):

- Compile a list of common questions and provide clear answers to address users' concerns.

- Include solutions to common issues and troubleshooting tips.

4. User Interface Guide:

- Explain the various elements of the user interface and how to navigate through the system.
- Highlight any specific features or functionalities unique to the Java Swing interface.

5. Security and Privacy Guidelines:

- Educate users about best practices for maintaining their account security and protecting their data.
- Describe the system's security measures and how they benefit users.

6. Collaboration and Sharing Instructions:

- Provide detailed instructions on how to collaborate with other users, share files and notes, and manage shared resources.

7. Search and Retrieval Tips:

- Offer tips on effective searching and retrieval of files and notes, including how to use search criteria efficiently.

8. Data Backup and Recovery Procedures (if applicable):

- Explain how users can back up their data and, if needed, recover data in case of data loss or system issues.

9. Glossary of Terms:

- Define technical terms, acronyms, and domain-specific jargon used within the system.

10. Contact Information and Support:

- Provide contact details for user support or a helpdesk, including email addresses, phone numbers, or links to support resources.

11. Updates and Release Notes:

- Inform users about updates, enhancements, and bug fixes in each system release.

12. Access to Online Resources:

- Include links to online resources, such as a project website, knowledge base, or user forums, where users can find additional information and support.

Ensure that the documentation is clear, well-organized, and accessible to users with varying levels of technical proficiency. It should serve as a valuable resource for users to maximize the benefits of your file management system while minimizing potential challenges or issues.

2.2.7 ASSUMPTIONS AND DEPENDENCIES

Assumptions and dependencies in CODEBOX help to identify certain conditions or factors that are considered as given or external to the project but can affect its execution and success. Here are some assumptions and dependencies for your file management system project:

Assumptions:

1. **User Technical Proficiency:** Users are assumed to have basic computer skills and familiarity with file management. The system will be designed to be user-friendly, but a minimum level of technical proficiency is assumed.
2. **Stable Internet Connection (if applicable):** If the system supports remote access and collaboration over the internet, it's assumed that users have access to a stable internet connection for uninterrupted service.
3. **Data Integrity and Availability:** The assumption is that data stored within the system will maintain its integrity and be available when needed. Appropriate backup and recovery mechanisms will be in place to support this assumption.
4. **Hardware and Software Compatibility:** It's assumed that users' workstations meet the specified hardware and software requirements, allowing the system to function as intended.
5. **Legal and Regulatory Compliance:** The project assumes that it will operate within the legal and regulatory framework for data privacy and copyright, with necessary precautions in place to adhere to these requirements.

Dependencies:

1. **Java Development Environment:** The project depends on a Java development environment for implementing the Java Swing user interface. Any changes or issues in the Java environment could affect the project.
2. **MySQL Database Server:** The system depends on the MySQL database server for data storage and retrieval. Any downtime or issues with the database server could impact the system's functionality.
3. **Internet Connectivity (if applicable):** For remote access and collaboration features, the project depends on a reliable internet connection. Any network issues or outages can affect the availability of these features.
4. **Third-Party Libraries and Components:** If the project uses third-party libraries or components, it's dependent on their functionality and maintenance. Updates or changes to these libraries could impact the system.
5. **User Input and Feedback:** The project may depend on user input and feedback for improvements and updates. User participation is essential for enhancing the system based on user needs.
6. **Resource Availability:** The project depends on the availability of resources such as development tools, hardware, and human resources for development and maintenance.
7. **Budget and Funding:** If the project has budget constraints, it depends on the availability of funds for development, hosting, and support.

Understanding these assumptions and dependencies is crucial for project planning and risk management. It allows you to proactively address potential issues and ensures that the project operates smoothly under various conditions.

EXTERNAL INTERFACE REQUIREMENTS

2.3.1 USER INTERFACES

Designing user interfaces (UI) for CODEBOX project involves creating user-friendly screens and interactions for tasks like file uploading, searching, downloading, and note-taking. Here are some key user interfaces you should consider for your project:

1. User Registration and Login:

- Allow users to create accounts and log in securely.
- Include fields for username, email, password, and possibly two-factor authentication.

2. Dashboard:

- Provide a central dashboard where users can access key features and navigate to other sections of the system.
- Display an overview of recent activity, file uploads, and notes.

3. File Upload and Management:

- Create an intuitive interface for users to upload files, including options to select files from their devices and specify file descriptions or categories.
- Implement features for organizing, editing, and deleting files.

4. File Search and Discovery:

- Design a search bar or advanced search options to help users find files easily.
- Display search results with relevant file details and the ability to preview or download files.

5. File Preview:

- Allow users to preview supported file types (e.g., documents, images, videos) directly in the browser without downloading them.

6. File Download:

- Implement download options for users to save files to their devices.
- Include features like batch download and download history.

7. Notes Editor:

- Create a user-friendly text editor for taking notes with features like formatting, bullet points, and the ability to save, edit, and organize notes.

8. Notes Management:

- Enable users to view, edit, and delete their notes.
- Organize notes by categories or tags for better accessibility.

9. User Profile:

- Allow users to manage their profiles, update personal information, and change account settings.

10. Sharing and Collaboration:

- If relevant, implement features for sharing files and notes with other users or external parties.

11. Notifications:

- Provide a notification center where users can see alerts about system events, such as file sharing invitations or system updates.

12. Settings and Preferences:

- Allow users to customize their experience by configuring preferences, such as notification settings and default views.

13. Help and Support:

- Include a help center or support section with FAQs, user guides, and contact options for user assistance.

14. Accessibility Features:

- Ensure the UI is accessible to individuals with disabilities, including features like screen reader compatibility and adjustable text sizes.

15. Responsive Design:

- Ensure that the UI is responsive, adapting to various screen sizes and devices, including mobile phones and tablets.

16. Security Measures:

- Implement security features, such as user authentication, encryption, and permissions management, to protect user data and files.

17. Feedback and Reporting:

- Include options for users to provide feedback or report issues or violations within the system.

Remember that user testing and feedback are essential for refining your user interfaces to ensure they meet user expectations and are easy to use. User-centered design principles should guide your UI development, making the system intuitive and efficient for users to manage their files and notes.

2.3.2 HARDWARE INTERFACES

In CODEBOX project, hardware interfaces are the connections or interactions between the software system and physical hardware components. While much of the functionality is software-based, hardware interfaces play a crucial role in data storage, retrieval, and user interactions. Here are some hardware interfaces to consider for your project:

1. Storage Devices:

- Hard Drives and Solid State Drives (SSDs): Hardware interfaces are needed to read from and write to storage devices for file storage and retrieval.

- Network Attached Storage (NAS): If you are using NAS for data storage, the system should have the necessary network interfaces to access and manage files on the NAS.

2. Input/Output Devices:

- USB Ports: Users may use USB ports to connect external storage devices for file transfer.

- Card Readers: Interfaces for reading memory cards and flash drives.

- Printers and Scanners: If your system supports printing and scanning of documents, hardware interfaces are required for communication with these peripherals.

3. Network Interfaces:

- Ethernet Ports: To connect the system to a wired network for data transfer and access.

- Wi-Fi: For wireless network connectivity, which allows users to access the file management system remotely.

4. Server Hardware:

- Rack Servers or Server Blades: If your system is hosted on dedicated server hardware, you need to ensure the server hardware interfaces are compatible with your software requirements.

5. Backup and Data Redundancy Systems:

- Tape Drives: For backup and archiving purposes, interfaces to connect and manage tape drives may be necessary.

- RAID Controllers: Hardware interfaces for managing redundant storage configurations.

6. Mobile and Tablet Interfaces:

- USB, Lightning, or USB-C ports for mobile devices: If your system supports file transfers with mobile devices, you'll need the appropriate hardware interfaces for connecting these devices to the system.

7. Biometric Devices:

- Fingerprint or retina scanners: If your system has security features, you may need interfaces for biometric devices to enhance user authentication.

8. Other Peripherals:

- Barcode Scanners, Keyboards, Mice, and other input devices: Interfaces for connecting and interacting with peripherals in specific use cases.

9. Hardware Security Tokens:

- Interfaces for hardware security tokens used for multi-factor authentication.

It's important to ensure that the software components of your file management system can effectively communicate with these hardware interfaces. This may require developing drivers, protocols, and APIs to manage interactions and data transfer. Additionally, compatibility with various hardware configurations and operating systems is essential for a seamless user experience.

2.3.3 SOFTWARE INTERFACES

If CODEBOX is a desktop application, the software interfaces will primarily involve interactions and integrations with components and libraries relevant to desktop application development. Here are some software interfaces to consider in the context of a desktop application:

1. Operating System APIs:

- Utilize operating system-specific APIs (e.g., Windows API, macOS API) for tasks like file system operations, window management, and system notifications.

2. GUI Frameworks:

- Depending on your chosen programming language and platform (e.g., Windows Forms, WPF for Windows; Cocoa for macOS; GTK, Qt for cross-platform), you'll need to work with the graphical user interface framework's APIs.

3. File System Interaction:

- Interface with the local file system for file management tasks, such as creating, deleting, moving, and copying files and folders.

4. Database Integration:

- If your application uses a local database, ensure smooth interaction with the database management system (e.g., SQLite, MySQL, or embedded databases like HSQLDB or Derby).

5. User Authentication and Authorization:

- Implement user authentication and authorization within the application, ensuring that only authorized users can access specific features or files.

6. Third-Party Libraries:

- Use third-party libraries for specific functionalities like text editing, image viewing, or document preview, depending on the file types your application supports.

7. Local Storage and Caching:

- Employ local storage mechanisms and caching strategies for improved performance and offline access to files and data.

8. Networking Libraries:

- If your desktop application connects to remote servers or services (e.g., for file synchronization or cloud storage), use networking libraries and APIs for secure communication.

9. Logging and Debugging Tools:

- Implement logging and debugging interfaces to assist in diagnosing and troubleshooting issues within the application.

10. Security and Encryption APIs:

- Use security and encryption APIs to protect user data and files, especially if the application stores sensitive information.

11. Localization and Internationalization:

- Incorporate localization and internationalization APIs to support multiple languages and regional settings for a diverse user base.

12. Automated Update Mechanisms:

- Consider mechanisms for automatic software updates to ensure users have access to the latest features and security fixes.

13. System Tray Integration (if applicable):

- Implement interfaces for adding functionality to the system tray or menu bar to provide quick access to application features.

14. Interprocess Communication (IPC):

- If your application consists of multiple processes or components, implement IPC mechanisms for data exchange and coordination.

15. Graphics and Multimedia Libraries:

- For viewing and handling multimedia files, leverage graphics and multimedia libraries for a better user experience.

16. Printers and Scanners (if applicable):

- If your application supports printing or scanning, use APIs to interact with connected devices.

Your desktop application will need to interface with these components and libraries to provide users with a seamless and feature-rich experience for managing files and taking notes. The specific interfaces you use will depend on the technology stack and platform you choose for development.

2.3.4 COMMUNICATIONS INTERFACES

Communication interfaces in your desktop CODEBOX refer to the methods and protocols used for exchanging data and information between your application and external entities, including users, other systems, or services. Here are some important communication interfaces to consider for your project:

1. User Interface (UI):

- This is the primary interface where users interact with your application. Ensure a user-friendly interface for file management, notes, and other features.

2. User Notifications:

- Implement notifications to inform users about events like file uploads, downloads, changes in shared files, or system updates.

3. File Upload and Download:

- Design interfaces for users to upload files to the system and download them. This may involve secure data transfer protocols (e.g., HTTPS) for uploading and downloading files.

4. Search and Query:

- Create a search interface that allows users to query and retrieve files efficiently based on various criteria.

5. Notes and Text Editing:

- Develop a text editing interface for creating and editing notes. Consider rich-text editors for a more versatile user experience.

6. File Sharing and Collaboration:

- If your system supports file sharing or collaboration features, design interfaces for inviting users to collaborate on files, managing access permissions, and communicating with collaborators.

7. User Registration and Authentication:

- Implement interfaces for user registration and login to establish secure user authentication.

8. APIs for Third-Party Integration:

- If you plan to provide APIs for third-party developers to integrate with your system, design these interfaces according to the data and functionalities you wish to expose.

9. Error Reporting and Logging:

- Implement interfaces for error reporting and system logging. Users and administrators should be able to access logs and report issues or errors.

10. Support and Helpdesk:

- Include interfaces for users to contact support, request assistance, or access a helpdesk for common issues.

11. Feedback and User Surveys:

- Create interfaces for users to provide feedback or participate in surveys to gather insights for future improvements.

12. Social Media and Sharing:

- If relevant, provide interfaces for users to share files or notes on social media platforms.

13. Real-Time Messaging:

- Implement interfaces for real-time messaging or chat functionalities, especially if your system supports collaboration and communication among users.

14. Data Synchronization:

- If your project involves synchronization with other devices or cloud services, design interfaces for data synchronization protocols and services.

15. File Metadata and Tagging:

- Develop interfaces for adding metadata and tags to files, enhancing search capabilities and organization.

16. System Updates and Notifications:

- Enable communication of system updates and important information to users through notification interfaces.

17. Data Backup and Recovery:

- If your system includes backup and recovery features, create interfaces for users to manage backups and restore data.

18. Integration with Cloud Services:

- If your system integrates with cloud storage services, design interfaces for connecting to these services through APIs.

Each of these communication interfaces is vital for ensuring a smooth user experience and efficient interaction with your desktop file management system. Consider the specific functionalities and user requirements of your system to determine the interfaces that are most essential for your project.

FUNCTIONAL REQUIREMENT

The CODEBOX is expected to encompass a range of essential functional requirements. Users should be able to register and authenticate securely, providing a foundation for their interaction with the system. Once logged in, they must have the capability to upload files, efficiently organize them, and navigate their collection. A robust search and retrieval function should enable users to locate files effortlessly, while downloading and sharing files with others must be straightforward. A text editor for creating and editing notes is essential, offering users the ability to organize and categorize their notes. User access control is crucial for administrators to manage user roles and permissions effectively. The system should issue notifications and alerts for file-related events and system updates, and, if required, support file versioning and data backup and recovery. Customization options for settings and preferences are necessary, as are security features like encryption and access controls. Finally, the application should support various file types, enable user feedback, and possibly integrate with cloud services or facilitate offline use and file synchronization, depending on project requirements. These functional requirements collectively shape the core features and behaviors of the desktop file management system, ensuring it meets user needs and project objectives effectively.

2.4.1 SYSTEM FEATURE 1

Certainly, let's consider the first system feature for CODEBOX :

Feature 1: User Registration and Authentication

The User Registration and Authentication feature serves as the foundational gateway to your file management system. It allows users to create accounts, securely log in, and manage their identities within the application. Key elements and functions of this feature include user registration, login, and password reset functionalities. During registration, users will provide essential information, such as their usernames, email addresses, and passwords, and may also set preferences for security features like two-factor authentication. The system should securely store and manage user credentials and authenticate users, providing access to their personal files, notes, and system functionalities upon successful login. This feature ensures that user data and system resources are accessible only to authorized users, forming the cornerstone of secure and personalized interactions within the application. It is an indispensable component for safeguarding user privacy and data integrity in your file management system.

2.4.2 SYSTEM FEATURE 2 (AND SO ON)

Certainly, let's consider the second system feature for your CODEBOX project:

Feature 2: File Upload and Management

The File Upload and Management feature is a core functionality of your file management system, enabling users to seamlessly upload, organize, and manage their files within the application. Users should be able to upload various file types, including documents, images, videos, and more, and organize them into folders or categories of their choosing. The feature should also support file metadata, including file names, types, sizes, and upload dates, allowing users to easily track and access their files. To provide a user-friendly experience, the system should offer functionalities like file preview, editing, deleting, and moving, ensuring that users have full control over their digital assets. This feature is fundamental in simplifying the process of file management and organization, empowering users to efficiently manage their documents and resources while maintaining a structured and easily navigable file hierarchy within the application.

NONFUNCTIONAL REQUIREMENTS

Nonfunctional requirements encompass various aspects critical to the success of your file management system. These requirements define attributes such as performance, security, usability, and compliance. Performance requirements may include expectations for system responsiveness, scalability, and resource utilization, ensuring that the application functions smoothly under varying loads. Security requirements dictate measures for data encryption, access control, and protection against vulnerabilities. Usability requirements focus on the user experience, emphasizing features like a user-friendly interface, accessibility, and responsiveness. Compliance requirements ensure that the system adheres to industry standards, legal regulations, and best practices. Collectively, these nonfunctional requirements serve as the foundation for creating a robust, user-friendly, secure, and compliant file management system.

2.5.1 PERFORMANCE REQUIREMENTS

Performance requirements specify the criteria and expectations related to the speed, efficiency, and resource usage of your file management system. These requirements ensure that the application meets performance-related objectives. Here are some performance requirements you might consider for your project:

1. **Response Time:** Define the maximum acceptable response times for key operations, such as file uploads, downloads, and searches. For example, file downloads should occur within X seconds for files under Y MB.
2. **Scalability:** Specify how the system should handle increasing user loads. For instance, the system should support concurrent access by a minimum of Z users without significant performance degradation.

- 3.Throughput: Define the number of transactions or operations the system should be able to handle per unit of time. For example, the system should support X file uploads per minute.
- 4.Resource Utilization: Describe the acceptable levels of CPU, memory, and disk usage under different load conditions to ensure the system operates efficiently.
- 5.Concurrency: Define the maximum number of concurrent users or sessions the system should support without degrading performance. For example, the system should support a minimum of X concurrent search operations.
- 6.Network Performance: Specify network-related performance requirements, such as acceptable latency for file transfers or the maximum allowable network bandwidth usage.
7. Database Performance: If your system uses a database, set requirements for database response times, query execution times, and data retrieval.
8. Load Testing: Define the load testing criteria and scenarios that will be used to validate the system's performance under various conditions.
- 9.Failover and Recovery Time: Specify the acceptable time for system failover in case of hardware or network failures and the recovery time after system disruptions.
- 10.Caching: If your system employs caching, set requirements for cache hit rates and cache update frequencies to improve response times.
- 11.Third-Party Integrations: Ensure that any third-party services or APIs integrated into the system meet performance requirements.

Performance requirements are critical to delivering a responsive and efficient user experience. They guide the design and testing of your system to ensure it can handle the expected workloads and deliver consistent performance, even under challenging conditions.

2.5.2 SAFETY REQUIREMENTS

Safety requirements in a software system typically pertain to ensuring that the application operates without causing harm to users, their data, or the environment. In the context of a desktop file management system, safety requirements may include the following:

- 1.Data Integrity: The system must ensure the integrity of user data, preventing accidental or malicious data corruption or loss. It should include data backup and recovery mechanisms to minimize the risk of data loss.
2. Access Control: Safety requirements may specify that only authorized users have access to sensitive files and system features. Implement robust user authentication and access control to prevent unauthorized access.
3. User Data Protection: The system should protect user data from unauthorized access, tampering, or disclosure. This might involve encryption of sensitive data and secure storage practices.

4. Error Handling: Safety requirements could include robust error handling mechanisms to gracefully handle unexpected situations without crashing the application or exposing vulnerabilities.
 5. User Notifications: Safety requirements may include notifying users about security-related events or breaches, ensuring that they are aware of any potential risks to their data or accounts.
 6. Data Privacy Compliance: If your application handles personal or sensitive data, safety requirements should ensure that the system complies with data privacy regulations, such as GDPR or HIPAA, to protect user privacy.
 7. Secure File Upload and Download: Implement safety mechanisms to scan uploaded files for malware or potential security threats. Ensure secure file transfer protocols for downloading files to prevent malicious code execution.
 8. Secure File Sharing: Safety requirements may involve secure file sharing mechanisms to prevent unauthorized access to shared files and to protect against unauthorized distribution.
 9. Security Updates: The system should be designed to receive and install security updates and patches to address vulnerabilities promptly.
 10. User Education: Include safety requirements for educating users about best practices for using the application securely, including password management, avoiding suspicious links, and recognizing phishing attempts.
 11. Secure Communications: If the application communicates with external services or cloud storage, ensure that data is transmitted securely using encryption and secure communication protocols.
 12. Compliance Auditing: If your application is subject to industry-specific regulations or standards (e.g., SOC 2, ISO 27001), safety requirements may involve regular compliance audits and reporting.
- Safety requirements are crucial to ensuring that your file management system protects user data and operates securely. They help prevent data breaches, data loss, and security vulnerabilities that could lead to harm or loss of user trust. Compliance with industry standards and best practices is often a key component of safety requirements in software systems.

2.5.3 SECURITY REQUIREMENTS

Certainly, here are some security requirements for your file management system project:

1. User Authentication and Authorization:

- Users must authenticate securely using strong passwords and may have the option for multi-factor authentication (MFA).
- Authorization mechanisms should restrict access to files and system features based on user roles and permissions.

2. Data Encryption:

- Implement data encryption for files at rest and during transmission using industry-standard encryption protocols.

3. Secure File Upload and Download:

- Scan uploaded files for malware and verify their integrity.
- Use secure file transfer protocols (e.g., HTTPS) for downloading files to prevent unauthorized access or tampering.

4. Access Control:

- Define user roles and access controls to limit access to files and system functions based on user permissions.
- Implement secure sharing mechanisms for controlled file sharing.

5. Data Backup and Recovery:

- Regularly back up user data to prevent data loss in case of hardware failures or accidental deletions.
- Ensure robust data recovery processes.

6. Secure Coding Practices:

- Adhere to secure coding practices to mitigate common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

7. Security Updates:

- Establish a process for promptly applying security patches and updates to address known vulnerabilities in the system's software and dependencies.

8. Logging and Monitoring:

- Implement comprehensive logging and monitoring to detect suspicious or unauthorized access.
- Protect logs against tampering.

9. Compliance:

- Ensure compliance with relevant data protection laws and industry standards, such as GDPR, HIPAA, or ISO 27001, based on the nature of the data you handle.

10. User Education:

- Provide user education on best security practices, including strong password management, recognizing phishing attempts, and avoiding suspicious links or downloads.

11. Security Audits and Penetration Testing:

- Conduct regular security audits and penetration testing to identify and address vulnerabilities and potential threats.

12. Incident Response Plan:

- Develop an incident response plan to handle security breaches, data leaks, or other incidents. Define roles and procedures for responding to and mitigating security issues.

13. Access Revocation:

- Implement a process for revoking access to the system and user accounts when needed, such as when employees leave an organization.

14. Secure Communication:

- Use secure communication protocols to protect data in transit, especially when communicating with external services or cloud storage providers.

15. Data Masking and Redaction:

- Apply data masking and redaction techniques to protect sensitive data from being exposed, even to authorized users with lower access rights.

16. Secure File Deletion:

- Implement secure file deletion mechanisms that ensure deleted files are unrecoverable.

These security requirements are crucial for ensuring that your file management system protects user data and maintains the confidentiality, integrity, and availability of the application, making it a secure environment for users to manage their files and notes.

2.5.4 SOFTWARE QUALITY ATTRIBUTES

Software quality attributes, also known as nonfunctional requirements, play a critical role in ensuring that your file management system not only functions correctly but also meets user expectations for performance, reliability, and usability. Here are some key software quality attributes for your project:

1. Performance:

- Response Time: Ensure that the system responds quickly to user interactions, such as file uploads, searches, and note creation.

- Throughput: The system should handle a reasonable number of file operations and user requests simultaneously.

- Scalability: Design the system to scale as user loads increase, maintaining performance under heavy usage.

2. Reliability:

- Ensure the system is available and reliable, minimizing downtime and data loss.

- Implement error handling and fault tolerance mechanisms to recover gracefully from unexpected errors or failures.

3. Usability:

- Provide an intuitive and user-friendly interface for efficient file and note management.

- Consider accessibility for users with disabilities to ensure inclusivity.

4. Security:

- Protect user data, files, and notes from unauthorized access and data breaches through strong authentication, access controls, and encryption.

- Implement secure coding practices to mitigate common vulnerabilities.

5. Maintainability:

- Design the system with clean and well-documented code to facilitate ongoing maintenance and updates.

- Support software version control and configuration management practices.

6. Scalability:

- Ensure the system can handle increased workloads and growing data volumes by scaling horizontally or vertically as needed.

7. Interoperability:

- Make sure the system can interface with other software components or services as required, such as third-party APIs, databases, or cloud storage.

8. Portability:

- Develop the application to be platform-agnostic, ensuring it can run on various operating systems and environments.

9. Compatibility:

- Ensure the system is compatible with a range of devices and browsers, allowing users to access it from desktops, mobile devices, and various web browsers.

10. Performance Efficiency:

- Optimize resource usage, such as CPU and memory, to ensure efficient operation, especially on lower-end hardware.

11. Reliability and Availability:

- Ensure high availability and reliability to prevent data loss and system downtime.

12. Adaptability and Extensibility:

- Design the system with adaptability in mind, allowing for future enhancements and extensions to accommodate changing user needs.

13. Data Quality:

- Maintain data quality by implementing validation checks and ensuring accurate and complete data entry.

14. Auditability and Traceability:

- Enable auditing and traceability features to track user activity and system changes for accountability and troubleshooting.

15. Compliance:

- Ensure the system complies with relevant regulations and standards, such as data privacy laws, to protect user data.

16. Feedback and Reporting:

- Implement features for user feedback, error reporting, and monitoring to improve system performance and address issues promptly.

Focusing on these software quality attributes will help you deliver a robust and user-friendly file management system that meets user expectations and industry standards. Each attribute contributes to the overall quality and reliability of the application.

PROJECT PLAN

Creating a project plan is essential for successfully managing and executing CODEBOX . Here's a high-level overview of the steps involved in developing a project plan:

1. Project Initiation:

- Define the project's objectives, scope, and deliverables.
- Identify the stakeholders and their roles.
- Establish a project team with clear responsibilities.
- Develop a project charter to formally authorize the project.

2. Requirement Analysis:

- Gather and document detailed requirements, both functional and nonfunctional, for the file management system.
- Define user stories and use cases to capture user needs and expectations.

3. System Design:

- Create system architecture and design documents, outlining the software and hardware components.
- Design the user interface, file storage, database schema, and security mechanisms.

4. Technology Stack Selection:

- Choose the technology stack, including programming languages, frameworks, and tools.
- Identify third-party components or libraries that will be integrated into the system.

5. Development:

- Start coding the file management system according to the design and requirements.
- Implement features incrementally, following agile or iterative development methodologies.

6. Testing:

- Conduct unit testing to ensure individual components work correctly.
- Perform integration testing to verify the interactions between system components.
- Carry out system testing to validate the overall functionality of the file management system.
- Execute security testing to identify and address vulnerabilities.

7. User Acceptance Testing (UAT):

- Collaborate with users to conduct UAT, ensuring the system meets their needs and expectations.
- Address any issues or bugs identified during UAT.

8. Documentation:

- Create user documentation, including user guides and manuals for the file management system.
- Develop technical documentation for developers and administrators.

9. Training:

- Train end-users and administrators on how to use and maintain the file management system.

10. Deployment:

- Plan the deployment strategy, considering whether the system will be hosted on-premises or in the cloud.

- Deploy the system in a production environment, ensuring all configurations are in place.

11. Monitoring and Maintenance:

- Implement monitoring tools to track system performance and availability.
- Establish a maintenance plan for regular updates, bug fixes, and feature enhancements.

12. User Support:

- Provide ongoing user support through a helpdesk or support team to assist with user queries and issues.

13. Project Management:

- Continuously monitor and control project progress, budget, and resources.
- Address any deviations from the project plan promptly.
- Conduct regular project status meetings with the project team and stakeholders.

14. Project Closure:

- Conduct a final review to ensure all project objectives have been met.
- Obtain formal acceptance from stakeholders.
- Archive project documentation and close out project contracts.

15. Post-Implementation Review:

- Evaluate the project's success and lessons learned.
- Document best practices and areas for improvement for future projects.

16. User Training and Change Management:

- Conduct user training sessions and ensure users are comfortable with the new system.
- Implement change management strategies to help users transition to the new system.

Throughout the project, you'll also need to consider risk management, budget tracking, and communication with stakeholders. Developing a detailed project plan with timelines, milestones, and dependencies will help keep the project on track and ensure its successful completion.

2.6.1 TEAM MEMBERS

Hemant Kumar Kashyap (Team Leader)

Pooja Singhad (Team member)

2.6.2 DIVISION OF WORK

Certainly, based on your technology stack (Java Swing and MySQL) and the roles mentioned earlier, here's how you can divide the project between you and your team member:

Hemant Kumar Kashyap :As mentioned earlier, you will primarily focus on project management, system design, front-end development, documentation, and user training. However, let's break down these responsibilities further:

1. Project Management and Coordination:

- Act as the project manager, overseeing the entire project.

- Define the project plan, tasks, timelines, and priorities.
- Ensure effective communication and coordination with the team member.

2. System Design and UI Development:

- Lead the system design phase, defining the architecture and database structure.
- Create wireframes or mockups for the user interface.
- Develop the user interface using Java Swing, including user registration, login, and UI components.

3. Documentation and User Training:

- Develop user documentation, including user guides and manuals for end-users.
- Create technical documentation for developers and administrators.
- Conduct user training sessions to educate users on how to navigate and use the system effectively.

Pooja Singhad :

Your team member will primarily focus on back-end development, testing, deployment, and ongoing support. Here's a more detailed breakdown:

1. Back-End Development:

- Focus on back-end development, implementing core functionalities like file upload, file management, and search features using Java and MySQL.
- Handle user authentication and access control in the back end.

2. Testing and Quality Assurance (QA):

- Act as the primary tester, conducting thorough testing of the system.
- Perform unit testing, integration testing, and system testing.
- Identify and document issues, bugs, and vulnerabilities for correction.

3. Deployment and Final Configuration:

- Handle the deployment of the system to a production environment, whether it's college servers or cloud hosting.
- Configure the system for production use and ensure it's accessible to users.

4. Final Testing and Bug Fixes:

- Conduct a final round of testing to identify and address any remaining issues, especially those discovered during deployment and initial user interactions.

5. Support and Maintenance (Post-Project):

- Provide user support and address any issues or questions that may arise after deployment.

This division of responsibilities allows you and your team member to work in tandem, with a clear distinction between front-end and back-end development, testing, and deployment tasks. Effective communication is key to ensure the project progresses smoothly and successfully.

2.6.3 TIME SCHEDULE

Creating a time schedule for your project is a crucial step in managing its progress and ensuring you meet your deadlines. Here's a sample time schedule for a college project involving the development of a file management system using Java Swing and MySQL:

Project Duration: Approximately 12 weeks (3 months)

Week 1-2: Project Initiation and Planning

- Define project objectives and scope.
- Create a project plan with tasks, timelines, and responsibilities.
- Identify core features and requirements.
- Choose technology stack (Java Swing and MySQL).

Week 3-4: Requirement Analysis and Design

- Gather and document detailed requirements.
- Design the system architecture and database structure.
- Create wireframes or mockups for the user interface.
- Plan user registration, login, and file management features.

Week 5-8: Development

- Begin coding the system.
- Focus on front-end development (user interface) and back-end development (file management, database integration).
- Implement user registration and login features.

Week 9: User Training Materials and Documentation

- Develop user documentation (user guides and manuals).
- Create technical documentation for developers and administrators.
- Prepare training materials for user training.

Week 10-11: Testing and Quality Assurance

- Conduct thorough testing, including unit testing, integration testing, and system testing.
- Identify and document issues, bugs, and security vulnerabilities.
- Collaborate with your team member for effective testing.

Week 12: Deployment and Final Testing

- Deploy the system to a production environment (college servers or cloud hosting).
- Configure the system for production use.
- Perform a final round of testing to catch any remaining issues.

Week 13: User Training and Support

- Conduct user training sessions.
- Provide ongoing support for users and address any issues that may arise.

Week 14: Project Review, Documentation, and Submission

- Review the project to ensure it meets objectives and requirements.

- Prepare project reports and presentations as needed.
- Submit the project for evaluation.

Week 15: Post-Project Activities (if necessary)

- Address any post-deployment issues or improvements.
- Conduct a post-project review to capture lessons learned.

ANALYSIS

Project analysis involves examining various aspects of your file management system project to ensure its feasibility, requirements, and overall planning. Here's an analysis of your project:

1. Project Feasibility Analysis:

- **Technical Feasibility:** Assess the technical capabilities and expertise of your team to develop the project using Java Swing and MySQL.
- **Economic Feasibility:** Consider the project's budget and resources required for development. Ensure the project is financially viable within your constraints.
- **Operational Feasibility:** Evaluate whether the proposed system aligns with the operational needs and goals of your college.

2. User Needs Analysis:

- Identify the specific needs and expectations of potential users (students, faculty, administrators) for the file management system.
- Conduct surveys or interviews to gather user feedback on desired features and functionality.

3. Requirements Analysis:

- Define detailed functional requirements, including user registration, file upload, search, access control, and note creation.
- Specify nonfunctional requirements, such as security, performance, and scalability.

4. Risk Analysis:

- Identify potential risks and challenges associated with the project, including technical risks, resource constraints, and time limitations.
- Develop a risk mitigation plan to address and minimize identified risks.

5. Technology Stack Analysis:

- Analyze the suitability of Java Swing and MySQL for your project, considering their compatibility, performance, and ease of development.

6. Competitive Analysis:

- Research existing file management systems or similar solutions to understand what features or functionalities are in demand and how your project can differentiate itself.

7. Security Analysis:

- Perform a security analysis to identify potential vulnerabilities in the system and define security measures to protect user data and privacy.

8. Cost-Benefit Analysis:

- Assess the anticipated costs of development, including software and hardware resources, as well as potential long-term maintenance costs.
- Calculate potential benefits, such as improved file management efficiency and user satisfaction.

9. Legal and Compliance Analysis:

- Ensure that your project complies with relevant data protection laws and college policies, particularly concerning data privacy and user consent.

10. User Training and Adoption Analysis:

- Analyze the training needs of users and how you plan to support user adoption.
- Consider the ease of use and user-friendliness of the system.

11. User Interface Analysis:

- Evaluate the user interface design for its intuitiveness, accessibility, and compliance with best practices in UI/UX design.

12. Data Backup and Recovery Analysis:

- Define the data backup and recovery strategy to protect against data loss and ensure business continuity.

13. Sustainability Analysis:

- Assess how sustainable the system is in terms of long-term support, updates, and potential future enhancements.

14. Project Schedule Analysis:

- Review the project schedule to ensure that it is realistic and achievable within the defined time frame.

By conducting a thorough analysis of these aspects, you can make informed decisions, plan effectively, and increase the chances of a successful project implementation. It also helps you anticipate and address potential challenges early in the project lifecycle.

3.1 METHODOLOGY USED (*IF OBJECT ORIENTED*)

Given that you're developing a file management system in Java Swing, an object-oriented approach is a natural fit. Object-oriented programming (OOP) promotes a modular and organized code structure, making it easier to manage and maintain your project. Here's how you can apply OOP methodologies to your project:

1. Object-Oriented Analysis (OOA):

During the analysis phase, you identify and define the main objects and their relationships in your file management system:

- Objects: Identify the key objects in your system, such as User, File, Note, Authentication, and Search.
- Attributes: Determine the attributes (data) associated with each object. For example, a User object might have attributes like username, password, and role.
- Relationships: Define how objects are related. A User has a relationship with Files they upload or Notes they create, and the system has a relationship with the Database.

2. Object-Oriented Design (OOD):

In the design phase, you create a plan for how the objects identified in OOA will be implemented:

- Classes: Create classes that represent the objects identified during analysis. For example, create a User class, File class, Note class, etc.
- Inheritance: Utilize inheritance to model relationships between objects. For instance, you might have a base class (e.g., User) from which other classes like Admin and RegularUser inherit.
- Encapsulation: Implement encapsulation by defining the access level (public, private, protected) for class attributes and methods, ensuring data integrity and security.
- Polymorphism: Utilize polymorphism to allow different classes to have methods with the same name but different behaviors. For instance, you could have a generic "save" method for both files and notes, but each class's implementation would differ.
- Abstraction: Abstract complex systems into manageable and reusable components. Create abstract classes or interfaces for common functionalities like file operations.
- Composition and Aggregation: Model complex objects by using composition (an object contains other objects) and aggregation (an object uses other objects). For example, a User may contain a collection of Files and Notes.

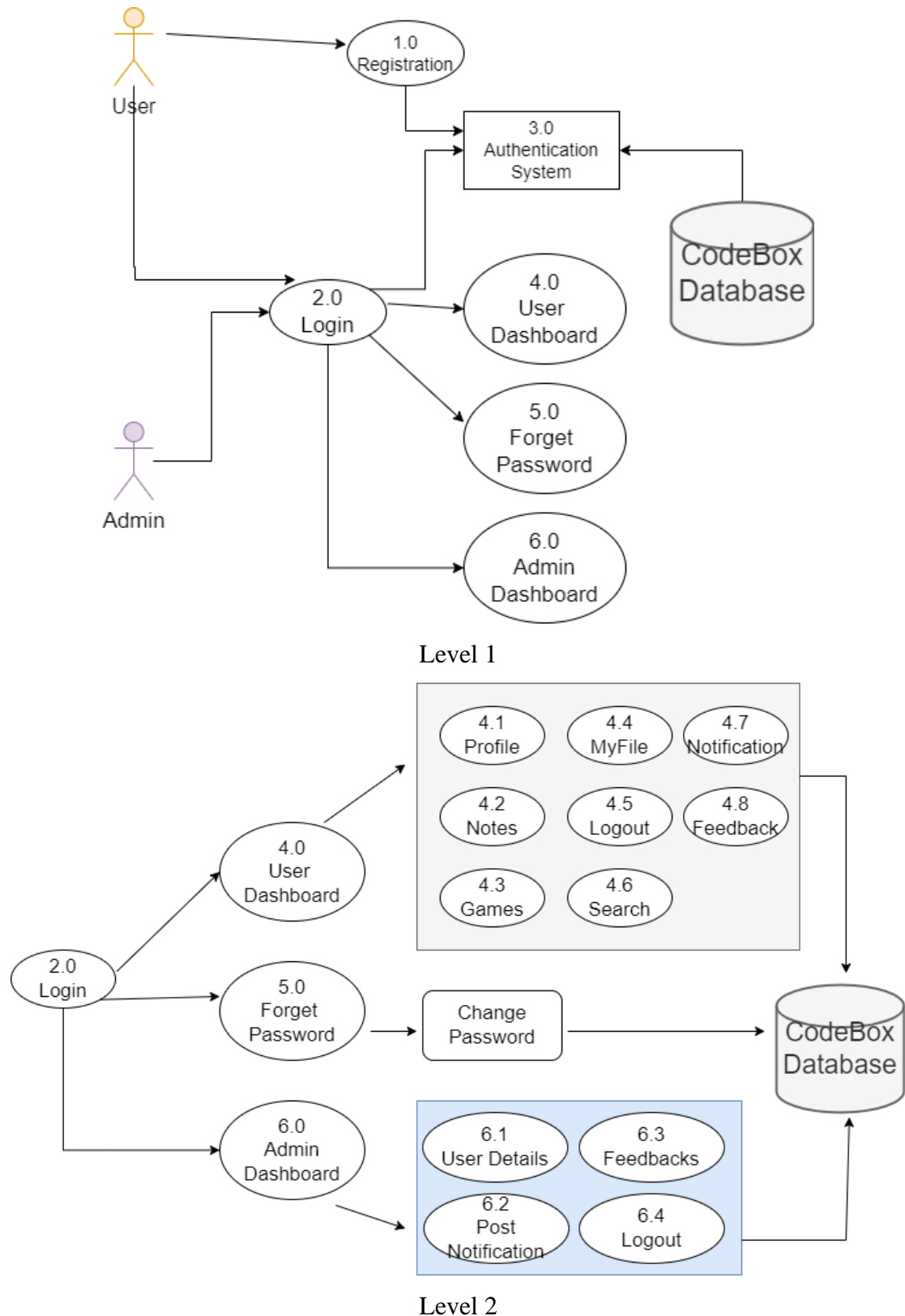
3. Object-Oriented Programming (OOP):

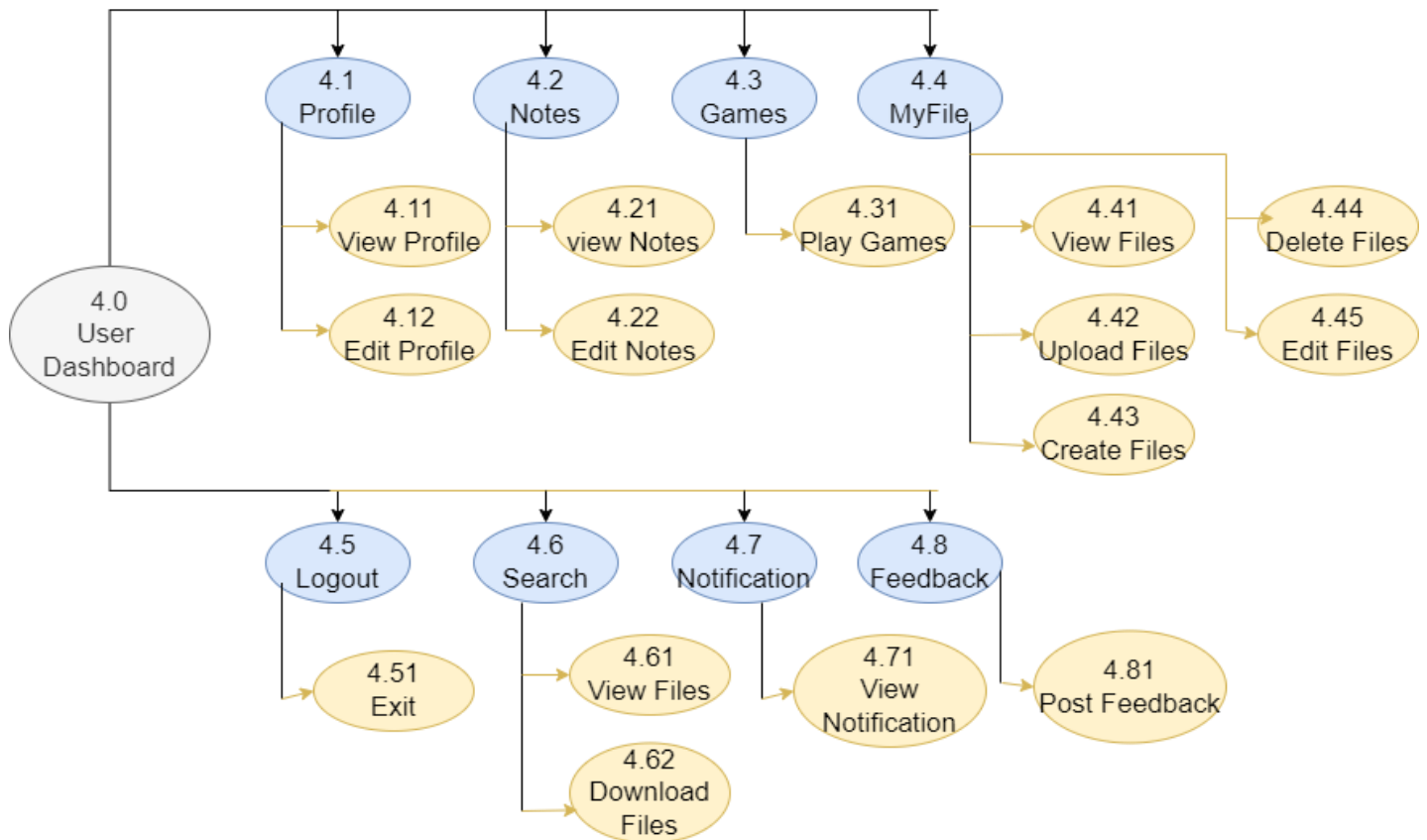
During the implementation phase, you write code using the Java programming language, adhering to the principles of OOP:

- Creating Objects: Instantiate objects from classes, e.g., creating a User object when a new user registers.
- Methods: Implement methods within classes to define their behaviors. For example, a User class might have methods for uploading files, creating notes, and searching for files.
- Inheritance and Polymorphism: Use inheritance and polymorphism to build on or override the behaviors of parent classes and interfaces.
- Encapsulation: Protect class attributes by using access modifiers and accessors (getters and setters).
- Interfaces and Abstract Classes: Implement interfaces and abstract classes to ensure consistency in classes that share common functionality.
- Exception Handling: Implement exception handling to manage errors and exceptions gracefully.

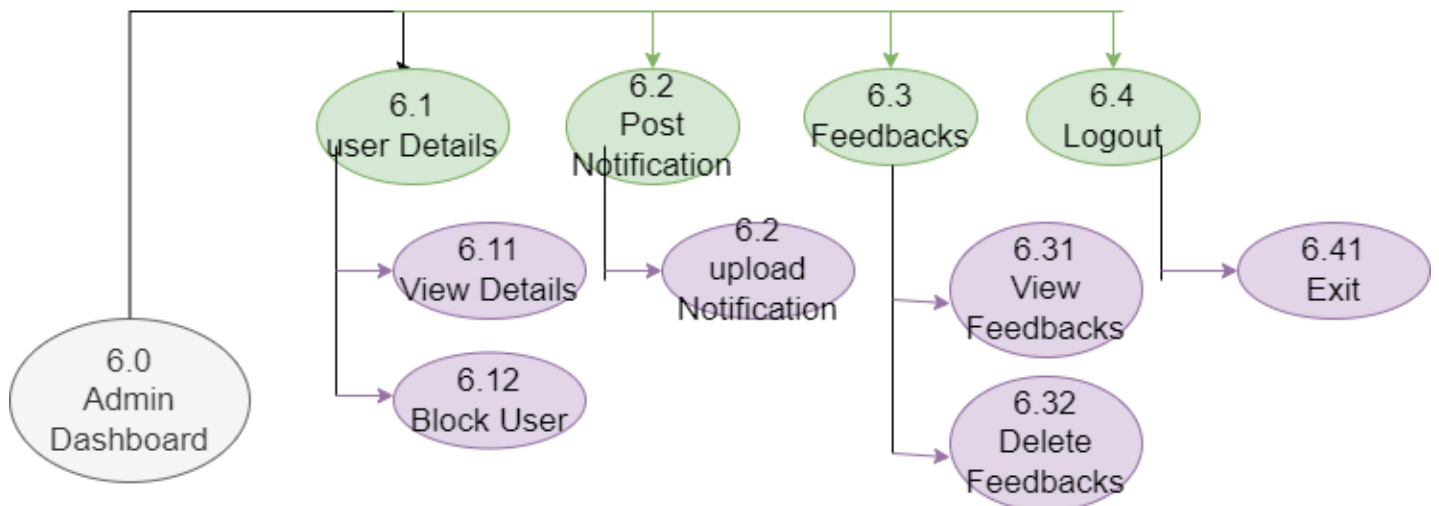
By following an object-oriented approach, you can create a structured and modular file management system that is easier to develop, test, and maintain. OOP principles also make it more adaptable for potential future enhancements and changes in requirements.

3.2 USE CASE DIAGRAMS (ALSO INCLUDE USE CASE SPECIFICATIONS)





Level 2.1



Level 2.2

3.3 ER MODEL

- The relation upon the system is structure through a conceptual ER-Diagram, which not only specifies the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.

- The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the data modeling activity the attributes of each data object noted in the ERD can be described as data object descriptions.

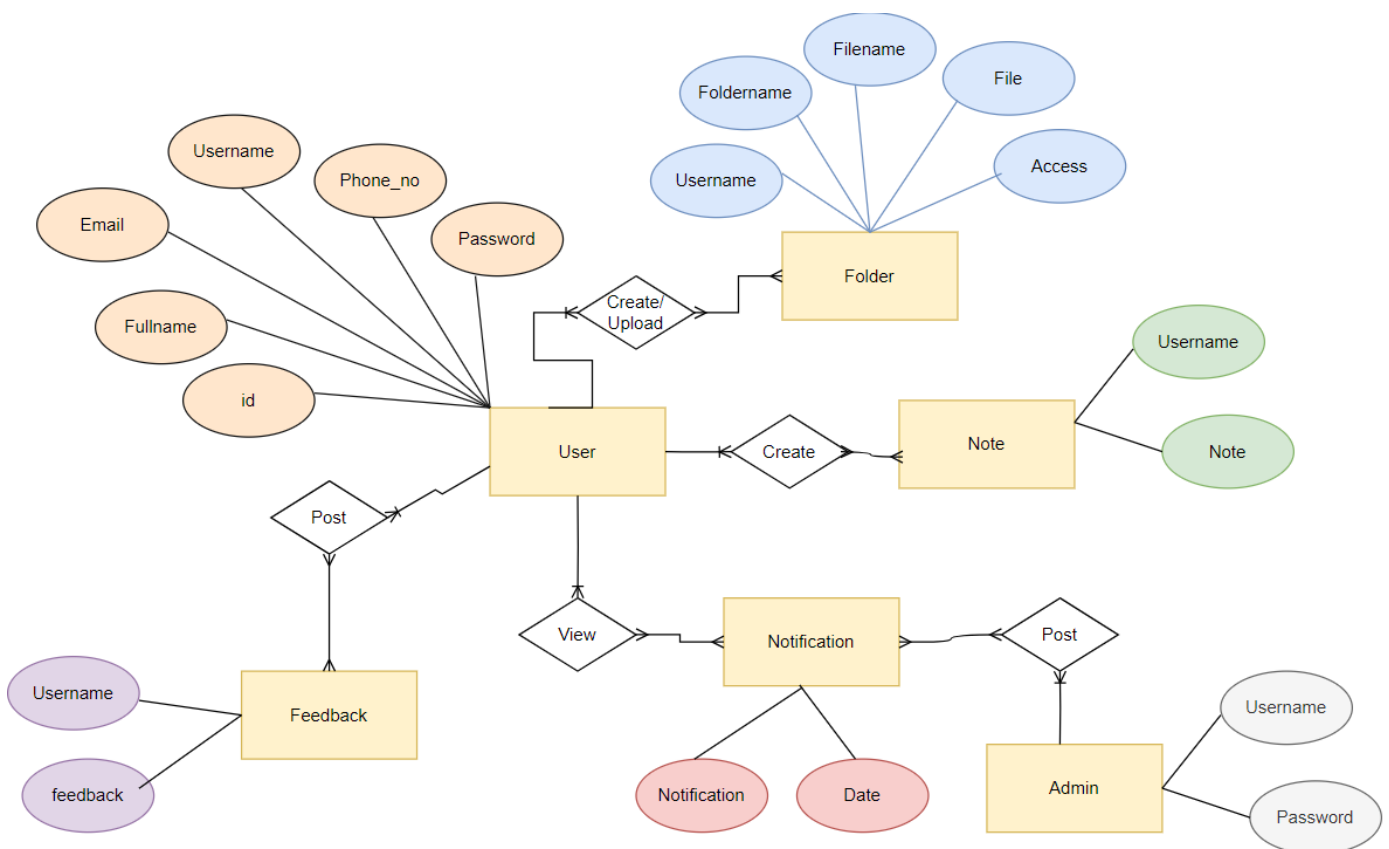
- The set of primary components that are identified by the ERD are

- Data object
- Attributes

The primary purpose of the ERD is to represent data objects and their relationships.

Symbols Used in ER Diagrams

- Rectangles: This Entity Relationship Diagram symbol represents entity types
- Ellipses: This symbol represents attributes
- Diamonds: This symbol represents relationship types
- Lines: It links attributes to entity types and entity types with other relationship types
- Primary key: Here, it underlines the attributes
- Double Ellipses: Represents multi-valued attributes



3.4 PROCESS SPECIFICATION

Process Specification typically involves detailing the processes identified in the Data Flow Diagram (DFD) at a more granular level. Here, I'll provide an example for the "File Upload" process from the File Management System.

Process: File Upload

Input:

User-selected file

User authentication information

Output:

Uploaded file

Confirmation message

Processing Steps:

Authenticate User: Verify the user's credentials and permissions.

Select File: Allow the user to choose a file for upload.

Validate File: Check file type, size, and other relevant parameters.

Store File: Save the file in the designated storage location.

Update Metadata: Update the database with information about the uploaded file.

Notify User: Send a confirmation message to the user.

3.5 CFD

A Control Flow Diagram illustrates the control flow within a system, showing the sequence of processes and the flow of control from one process to another.

Here is a simplified example for the "File Upload" process:

3.6 STD

A State Transition Diagram represents the different states that a system or component can be in and how it transitions from one state to another based on events.

Here's a basic example for the "User Profile" state transitions:

States:

Viewing Profile: Initial state when the user accesses their profile.

Editing Profile: Transition to this state when the user chooses to edit their profile.

Saving Changes: Transition when the user saves changes to their profile.

Viewing Updated Profile: Transition when the changes are saved successfully.

Events:

Edit Profile: Triggers the transition from "Viewing Profile" to "Editing Profile."

Save Changes: Triggers the transition from "Editing Profile" to "Saving Changes."

Changes Saved: Triggers the transition from "Saving Changes" to "Viewing Updated Profile."

4. DESIGN

Sure, in short, the design of your file management system involves:

1. Creating a three-tier architecture (presentation, application, data layers).
2. Designing a user-friendly Java Swing user interface.
3. Defining a MySQL database schema for user accounts, files, and notes.
4. Implementing secure user authentication and access control.
5. Developing components for file management, notes, and search.
6. Incorporating security measures and error handling.
7. Ensuring data backup and recovery strategies.
8. Comprehensive testing and quality assurance.
9. Preparing user and technical documentation.
10. Considering scalability and performance optimization.
11. Using version control for project management.

4.1 ARCHITECTURAL DESIGN

In short, the architectural design of your file management system involves:

1. Presentation Layer: Designing the user interface using Java Swing.
2. Application Layer: Creating controllers/handlers, implementing business logic, and ensuring security.
3. Data Layer: Defining the MySQL database structure and data access.
4. Integration: Establishing communication between layers.
5. Security: Implementing user authentication, authorization, input validation, and secure communication.
6. Scalability and Performance: Planning for potential growth and optimizing performance.
7. Error Handling and Exception Management: Managing errors and exceptions gracefully.
8. Data Backup and Recovery: Planning for data backup and recovery in case of failures.

4.1.1 SYSTEM ARCHITECTURE DIAGRAM

4.1.2 DESCRIPTION OF ARCHITECTURAL DESIGN

4.2 DATABASE DESIGN

Login Table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
FULLNAME	Varchar	Yes	No

EMAIL_ID	Varchar	Yes	No
USERNAME	Varchar	Yes	Yes
PHONENO	Number	Yes	No
PASSWORD	Varchar	Yes	No
CPASSWORD	Varchar	Yes	No
SECURITYQ	Varchar	Yes	No
ANSWER	Varchar	Yes	No

Folder Table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
FOLDERNAME	Varchar	Yes	No
FILENAME	Varchar	Yes	No
FILES	Varchar	Yes	No

Notes table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
NOTES	Varchar	Yes	No

ProfileImage table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
IMAGEFILE	Blob	Yes	No

FeedBack table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
FEEDBACK	Varchar	Yes	No

Notify table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
NOTIFICATION	Varchar	Yes	No
CURRENT_DATE	Varchar	Yes	No

Blocked_User table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
EMAIL_ID	Varchar	Yes	No

Personal_Details table

COLUMN NAME	DATA TYPE	NULLABLE	PRIMARY KEY
USERNAME	Varchar	Yes	Yes
BIO	Varchar	Yes	No
CURRENTROLE	Varchar	Yes	No
COUNTRY	Varchar	Yes	No
EDUCATION	Varchar	Yes	No
LINKEDIN_ID	Varchar	Yes	No

DATA DICTIONARY

NORMALIZATION

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy ie, repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies. Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly Data inconsistency resulting from data redundancy and partial update

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

FIRST NORMAL FORM:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

SECOND NORMAL FORM:

A relation is said to be in second Normal form if it is in first normal form and it should satisfy any one of the following rules

- 1) Primary key is not a composite primary key
- 2) No non key attributes are present
- 3) Every non key attribute is fully functionally dependent on full set of primary key

THIRD NORMAL FORM:

A relation is said to be in third normal form if there exists no transitive dependencies.

Transitive Dependency: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

COMPONENT DESIGN

FLOWCHART

INTERFACE DESIGN

Interface design for your file management system is a crucial aspect of creating a user-friendly and visually appealing application. Here are key considerations for designing the user interface (UI) in your Java Swing-based project:

1. Navigation and Layout:

- Create a well-organized and intuitive layout with menus, toolbars, and navigation controls.
- Ensure consistent placement of UI elements, such as buttons, labels, and input fields, for a coherent user experience.
- Consider using a sidebar or navigation panel for quick access to different sections of the application.

2. Visual Elements:

- Choose a clean and consistent color scheme that aligns with your project's branding or theme.
- Select appropriate fonts and font sizes to ensure readability.
- Use icons and images to enhance the visual appeal and aid in user understanding.

3. User Registration and Login:

- Design registration and login screens with clear and user-friendly input fields.
- Include password strength indicators and validation messages.
- Provide feedback on successful or failed login attempts.

4. File Management:

- Create a user-friendly interface for file upload, download, and management.
- Use table views to display file details (name, type, date) for easy browsing.
- Include options for sorting and filtering files.

5. Notes Feature:

- Design a text editor component for creating and editing notes.
- Consider incorporating basic text formatting options like bold, italic, and bullet points.

6. Search Functionality:

- Implement a search bar for users to search files and notes.
- Display search results clearly and offer filtering options.

7. User Profile:

- Design a user profile section where users can view and edit their account information.
- Include options for changing passwords or profile pictures.

8. Buttons and Actions:

- Use descriptive labels for buttons and actions to make their purpose clear.
- Implement tooltips to provide additional information or guidance.

9. Error Handling:

- Display error messages with clear explanations and suggestions for resolution.
- Highlight fields or areas where errors occur to guide users.

10. Responsiveness:

- Ensure that the UI is responsive and adapts to different screen sizes.
- Consider mobile responsiveness if the application may be used on smaller devices.

11. Accessibility:

- Design the interface with accessibility in mind, ensuring compatibility with screen readers and keyboard navigation.

12. User Feedback:

- Include feedback mechanisms such as progress indicators, success messages, and confirmation dialogs.

13. User Training:

- Offer on-screen guidance, tutorials, or tooltips to assist users in learning how to use the application.

14. Testing:

- Thoroughly test the UI to identify and fix any usability issues, visual glitches, or layout problems.

15. User Documentation:

- Prepare user documentation, including user guides and manuals, to assist users in navigating and using the application.

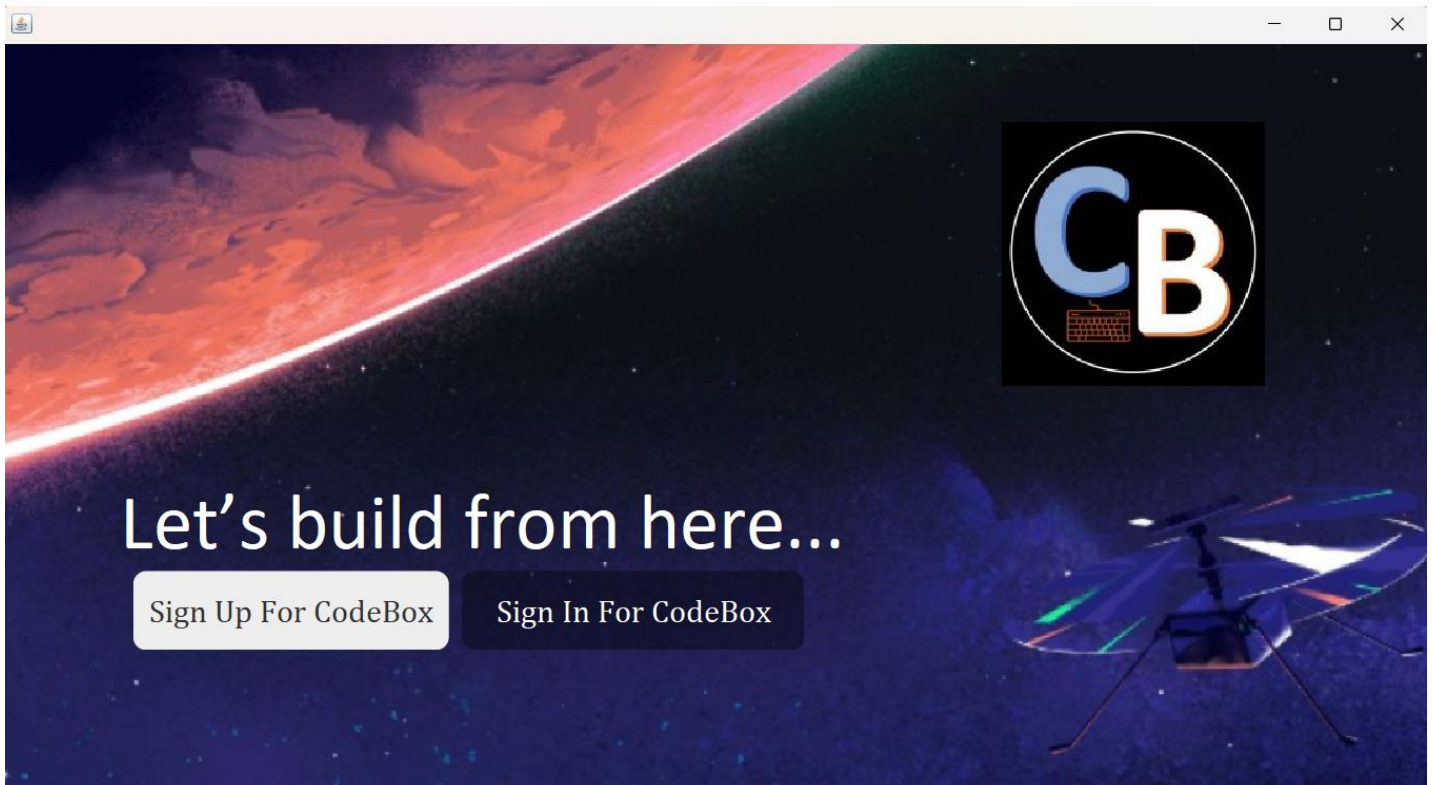
Effective interface design will enhance the user experience, making it easier for users to interact with your file management system. It's important to continuously gather feedback from users during development and incorporate usability testing to refine the design.

SCREENSHOTS

Welcome page

Steps :-

1 . Create new account by clicking on Sign Up for CodeBox Button :



2. When click on Sign Up for CodeBox button User can register for application :

The screenshot shows a web browser window with a blue header and a white registration form titled "Registration Here". The form includes fields for Name (Himanshu), Email_Id (hgmail.com), Username (Enter Username), Password (Enter Password), and Confirm Password. Below these are checkboxes for "Term's and Condition" and input fields for "Enter Security Question :-" and "Answer :-". A blue "Regisiter" button is at the bottom right. A modal message box is displayed in the center, titled "Message", with an information icon and the text "Enter All Detail's" and an "OK" button.

If any field left blank then this error message will shown .

3. When all fields are filled correctly proceed with registration :

The screenshot shows the same "Registration Here" form, but now all fields are filled: Name (Himanshu), Email_Id (hemant2004@gmail.com), Username (Hk@123), Password (12345678), and Confirm Password (12345678). The "Term's and Condition" checkbox is now checked. The "Enter Security Question :-" field contains "Hobby ?" and the "Answer :-" field contains "Book reading". The blue "Regisiter" button remains at the bottom right. A modal message box is displayed in the center, titled "Message", with an information icon and the text "Registered Successfully" and an "OK" button.

4 .When clicking on Sign in for CodeBox Button (User Login page):

CODE BOX

User

Login Here

Username

Enter Username

Password

Enter Password

☐ Show Password [Forget Password?](#)

Log In

8. . User can change there password here :

Back

Reset Password

Hk@123

Search

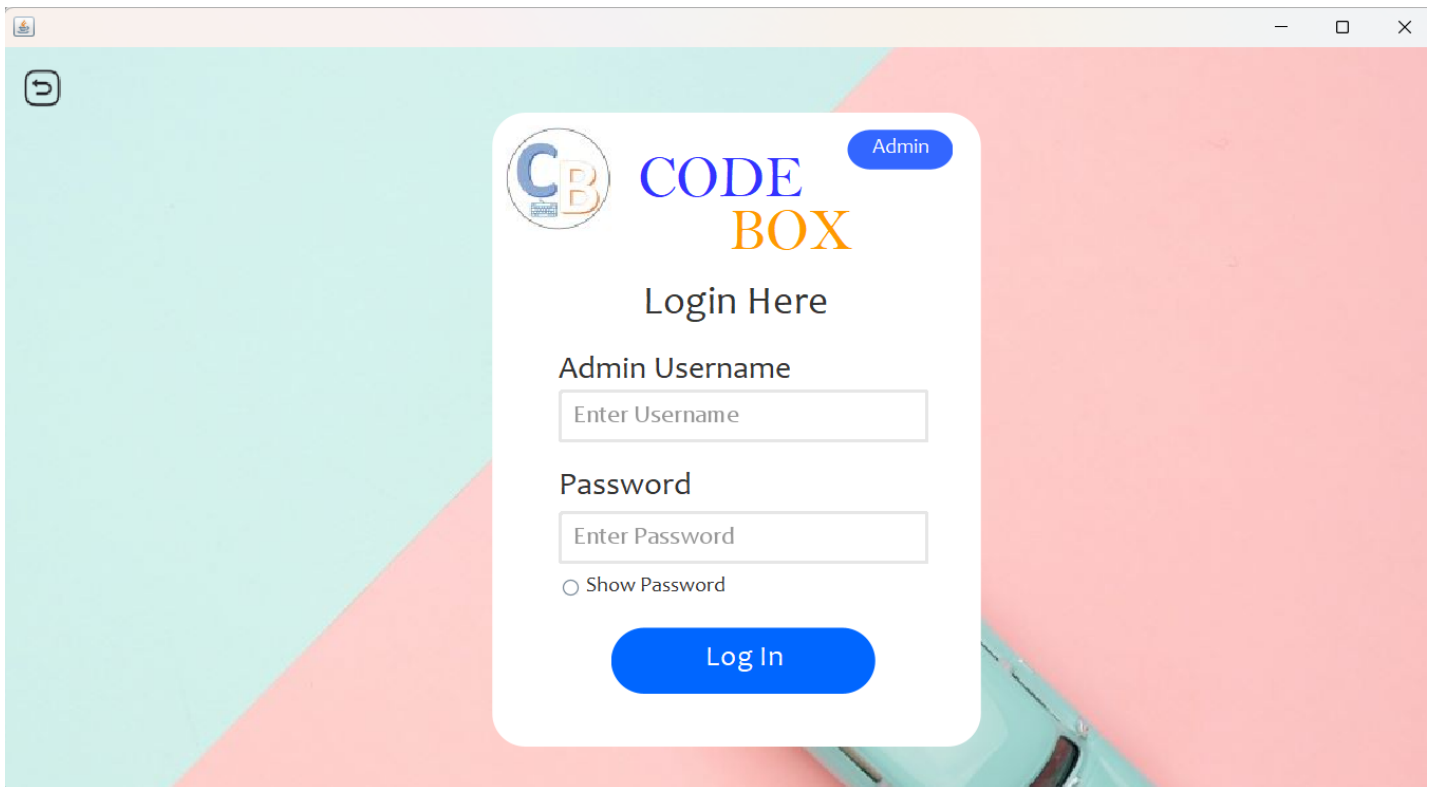
Hobby?

Enter Your Answer

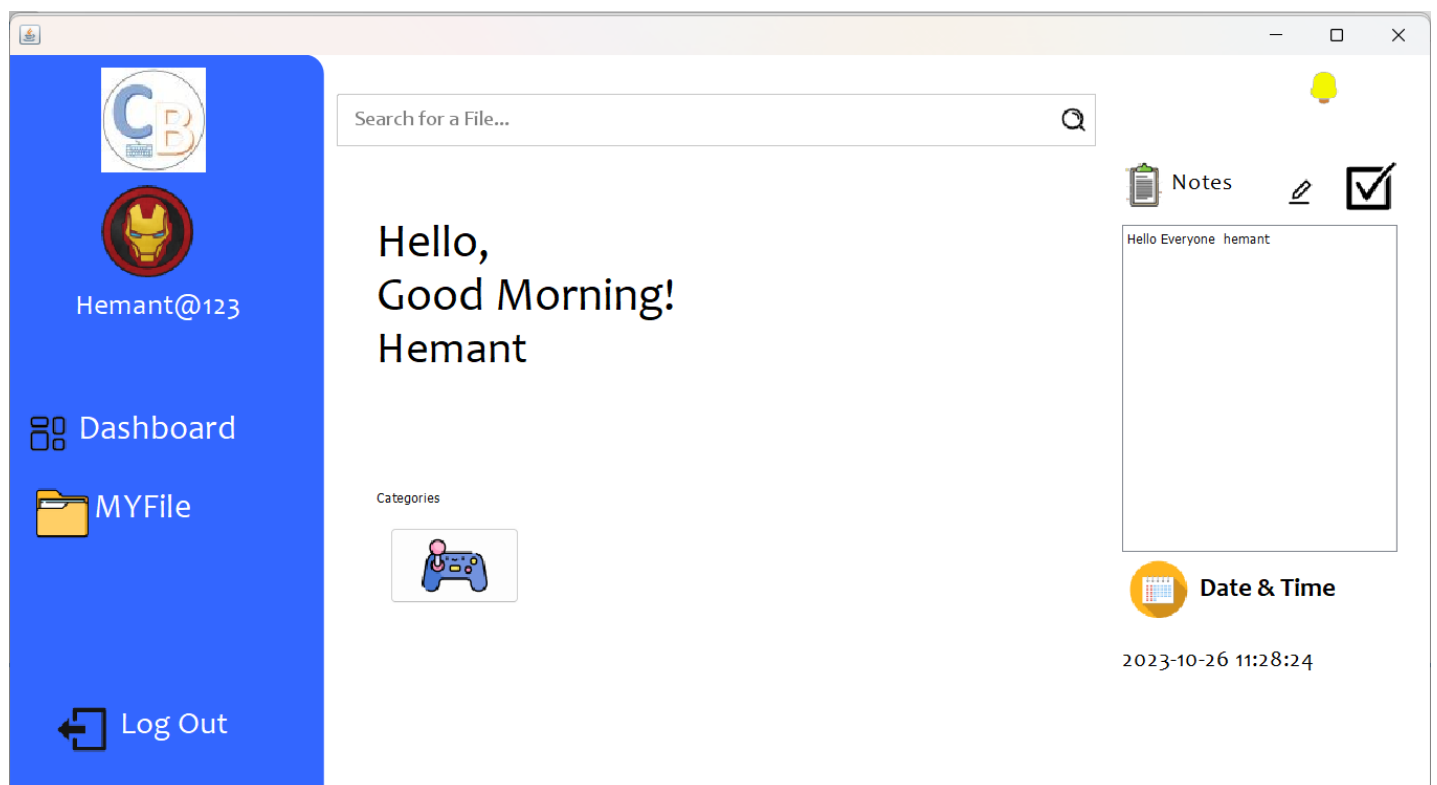
Enter New Password

Submit

9. When Click on User Button The Admin Login Page will appear :




11 . When User Fill all the field correctly He/She will be redirected to User Dashboard : Here Sample User (Hemant@123)



On Dashboard The Notes Function will be shown where User can Modify there notes according to there need .

12. When Click on Profile Icon Profile Page will Open :


Hemant@123
Hello

India

Name
Hemant

Email ID
kashyaphemant@gmail.com


Username
Hemant@123


Phone no
7803026553


Select Education
Btech

I am currently a *
Student

Folders New +

 Hemant@123

 Html_Files

Html_Files 

Your Files Add Files via Upload Add File New File

Readme.md

Html

Html&CSS

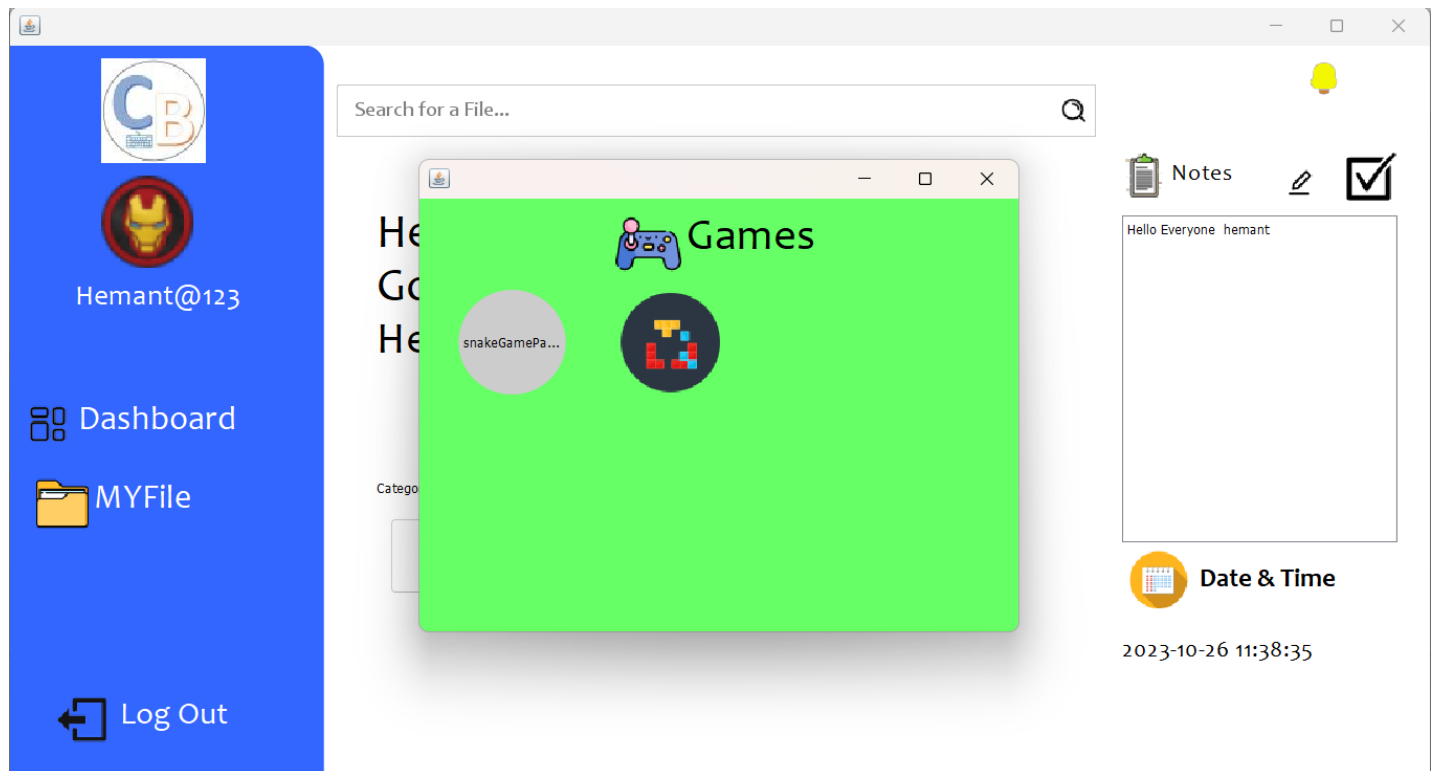
Css

Java

JavaScript

18. When User want to create a new Folder Click on new button and can set the access also Public or Private :

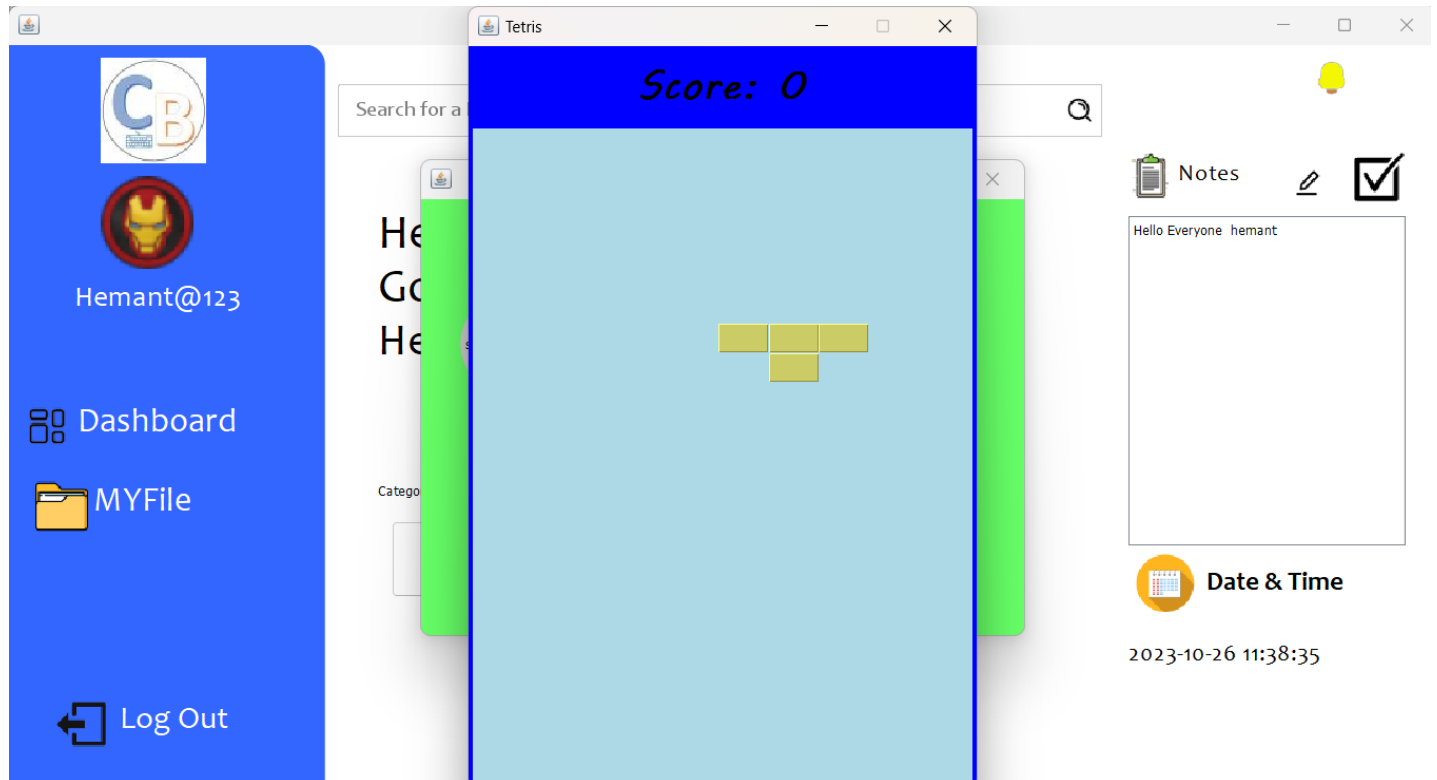
1.Snake 2 . Tetris



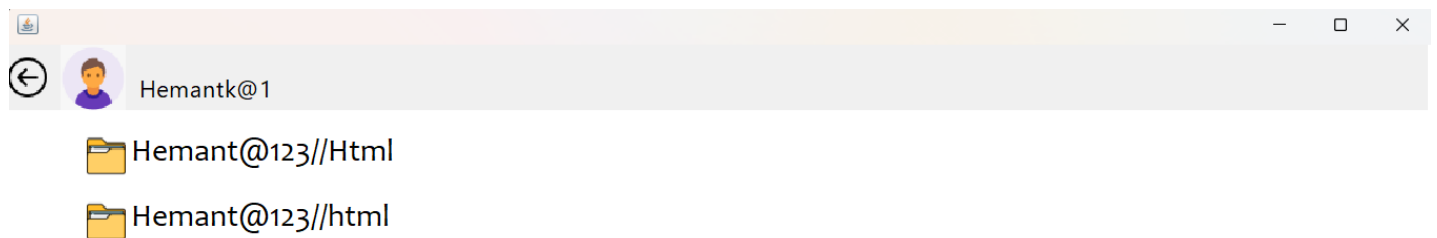
20. The Display of snake game :



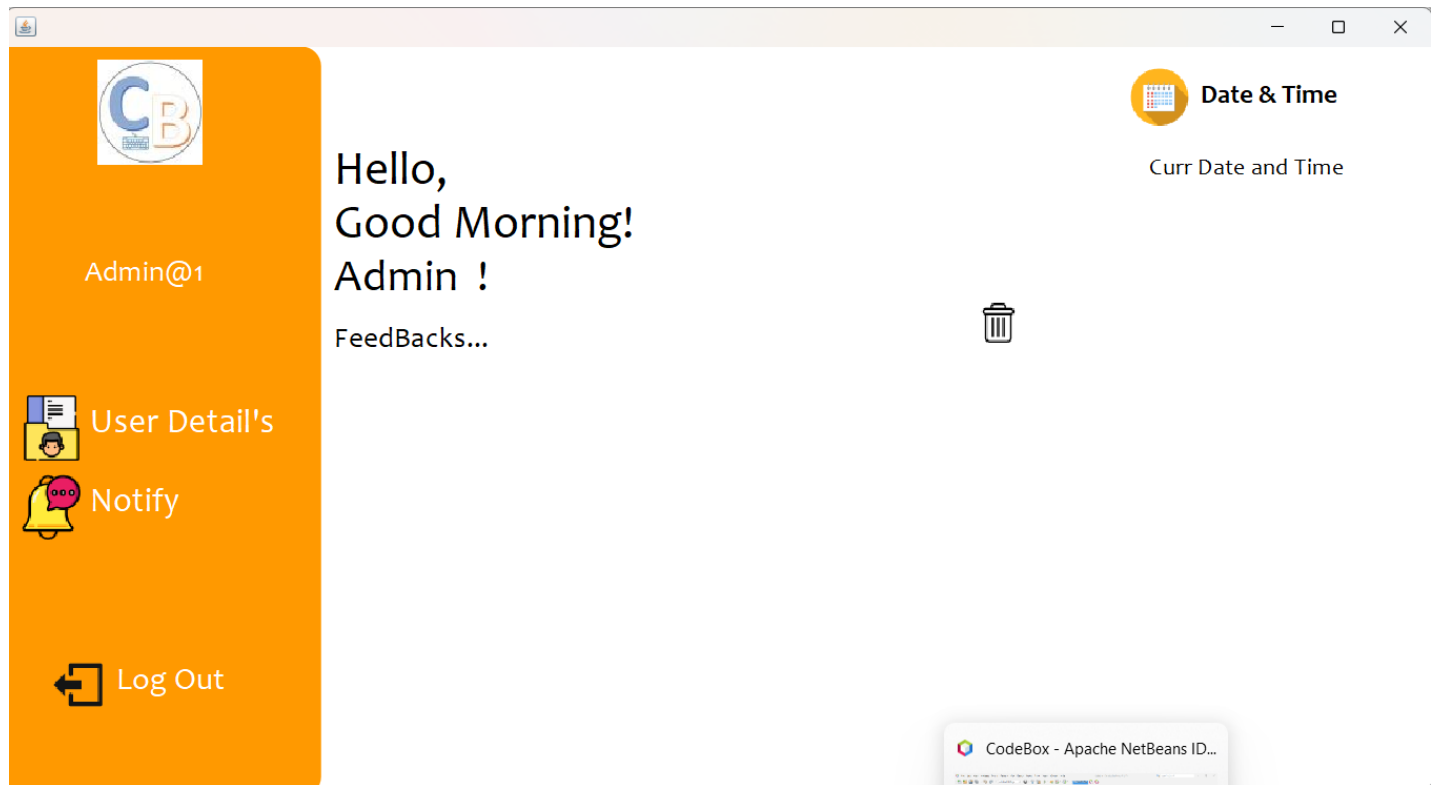
21. This Display of Tetris Game:



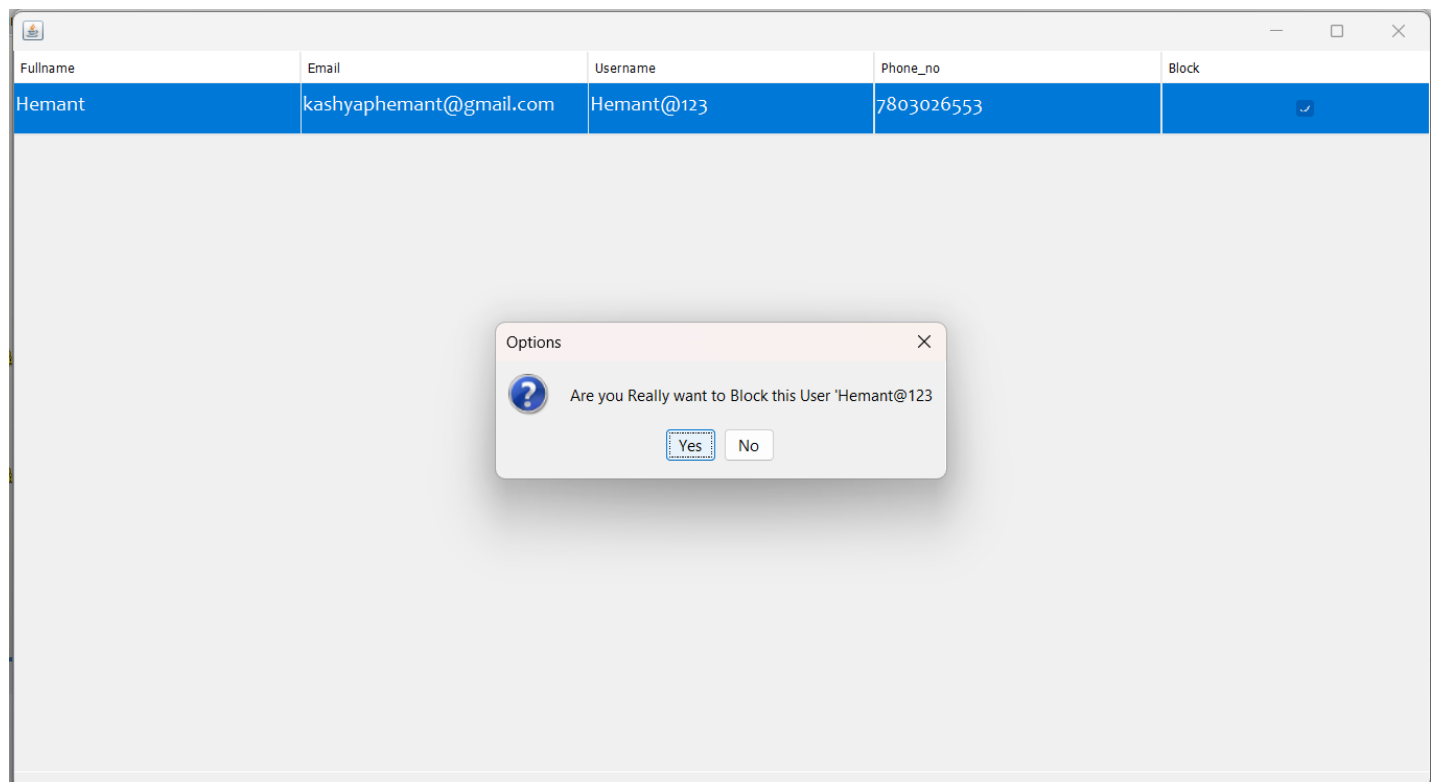
22. By typing something on Search Box in Dashboard User will get all the public file related to that word and User can download the file also :



23. This is the Admin's Dashboard here Admin can see user details feedbacks of User and post notification for all the user :



24. Admin can block some User also to access this application :



25. When click on Notify this will appear and admin can write the message what he wants to give and click on post to post the notification :

