Kingdom of Saudi Arabia
Qassim university
College of Computer
Department of Computer Science

المملكة العربية السعودية
جامعة القصيم
كلية الحاسب
قسم علوم الحاسب

Digital Notes Management System

Dr.Alaa Al-Sahil

CS383

Group No.5227

PREPARED BY:

1- RAWAN ALFOUZAN - 441203131
2- RAGHAD MESLEH ALHARBI - 441203195
3- JANA ALMISHALI - 441203301
4- ASEEL ALOWAIRDHI - 441203369
5- KADI ALBAKRI - 441203464
6- BILSAN ALMANAEE - 441203277
7- ALJAZI ALEQAB - 441203328
8- LUBNA ALAWAD - 441203288

Table of Contents

01
SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1 Introduction
2 System Overview
3 Requirements Engineering
4 Functional Requirements
5 Non - Functional Requirements
6 External Interface Requirements
7 UML - Use Case Diagram
8 Ethics and Responsibilities

02
SOFTWARE DESIGN DOCUMENT (SDD)

1 Software Design and Document (SDD)
2 Software Architecture Design
3 Process Model
4 Component Design
  4.1 Class Diagram
  4.2 State Diagram
  4.3 Activity Diagram
  4.4 Sequence Diagrams
5 Contributions Table

03
PROJECT MEETINGS NOTES

1 First meeting
2 Second Meeting
3 Third Meeting
4 Fourth Meeting
5 Repository link

Software Requirements Specification (SRS)

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to provide a detailed Software Requirements Specification (SRS) for the Digital Notes Management System. This document describes the system's goals, users, features, constraints, and required functionalities to ensure a clear understanding among all stakeholders.

1.2 Scope

The Digital Notes Management System enables users—including students, teachers, and professionals—to create, categorize, store, search, and manage digital notes efficiently. The system supports tagging, cloud backup, and real-time collaborative editing. It targets individuals and teams who require an organized and centralized method for managing digital notes.

1.3 Document Structure

This document is organized into several sections to clearly define the requirements of the Digital Notes Management System.

- Section 1 provides the introduction, purpose, scope, document structure, constraints, key definitions ,overview and References
- Section 2 presents an overview of the system and its main capabilities.
- Section 3 describes the requirements engineering process, including gathering, analysis, and validation.
- Section 4 outlines the functional requirements of the system.
- Section 5 lists the non-functional requirements.
- Section 6 describes the external interface requirements.
- Section 7 presents the UML Use Case diagram.
- Section 8 discusses ethical and professional responsibilities followed during development.

1.4 Constraints

The requirements engineering process for this system is affected by several constraints:

- Time Constraints: The project must be completed within the academic semester, limiting the depth of requirements exploration.
- Knowledge Constraints: Team members have varying levels of experience in software engineering, which may influence the complexity of the system.
- Tool Limitations: The system design and diagrams must be created using approved tools (Canva, Lucidchart, StarUML), which may limit certain advanced modeling features.
- Scope Limitations: The project focuses on analysis and design rather than full implementation, so requirements are defined at a conceptual level.
- Resource Constraints: No actual server or cloud platform will be deployed; therefore, cloud-related requirements are documented hypothetically.

1.5 Definitions and Acronyms

- SRS: Software Requirements Specification
- User: Any student, teacher, or professional using the system
- Admin: System administrator responsible for user management
- Tag: A keyword assigned to notes for categorization
- Cloud Sync: Synchronizing data across devices
- Collaboration: Real-time shared editing of notes

1.6 Overview

The remaining sections of this document describe the Digital Notes Management System in detail, including its functional and non-functional requirements, interface specifications, UML use case model, and ethical responsibilities followed throughout the development process.

1.7 References

Notion — Official Documentation. https://www.notion.so
Todoist --- https://www.todoist.com

2 SYSTEM OVERVIEW

This section describes the general factors that influence the Digital Notes Management System and its requirements. It explains the product's background, main functions, user characteristics, constraints, assumptions, and its relationship to external systems. This section does not include detailed requirements, which are provided in later sections.

2.1 Product Perspective

The Digital Notes Management System is an independent software platform designed to help users create, organize, and manage digital notes. It operates as a standalone web or mobile application but may integrate with external services such as cloud storage providers for synchronization.
The system is conceptually similar to existing note-taking tools like Notion, Evernote, and Google Keep, but with a stronger focus on structured tagging, collaboration, and academic productivity. It includes internal components such as:

- User Management Module
- Note Creation & Editing Module
- Tagging and Organization Module
- Search Engine Module
- Cloud Sync Module
- Collaboration Module

2.2 Product Features

The major functions provided by the system include:

- Creating, editing, and deleting digital notes
- Organizing notes using tags or categories
- Searching notes by title, content, or tag
- Syncing notes across devices using cloud storage
- Sharing notes with other users
- Real-time collaborative editing
- Commenting on shared notes
- Admin management of users and permissions

These functions support individual productivity as well as group collaboration.

2.3 User Roles and Characteristics

1) Regular User (Student / Teacher / Professional)

- Basic computer and mobile usage skills
- Ability to write and organize notes
- No technical expertise required
- Uses the system for studying, teaching, planning, or work tasks

2) Collaborator

- Similar skills as regular users
- Able to edit or comment on shared notes in real-time
- Familiar with teamwork and shared digital tools

3) Admin (System Administrator)

- Higher technical knowledge
- Familiar with system management tasks
- Responsible for managing user accounts, monitoring shared activities, and ensuring system integrity

These user classes ensure the system suits a wide range of educational and professional users.

2.4 Operating environment

The system is intended to operate on modern web browsers (Chrome, Safari, Edge) and mobile devices running iOS or Android. A stable internet connection is required for cloud synchronization and real-time collaboration. The system operates conceptually without a deployed backend server due to project constraints.

2.5 Design and Implementation Constraints

- The development and design of the system are influenced by the following constraints:
- Tool Constraints: System models must be created using approved tools (Canva, Lucidchart, StarUML).
- Platform Constraints: Cloud synchronization is hypothetical and not linked to an actual server for this academic project.
- Security Constraints: Must follow basic security practices such as password encryption and user authentication.
- Regulatory Constraints: Must comply with data privacy principles—no sharing of personal data without permission.
- Interface Limitations: System interacts only with conceptual cloud services and no external APIs are implemented.
- Hardware Limitations: The system should run on standard laptops and mobile devices.

2.6 Assumptions and Dependencies

The system operates under the following assumptions and dependencies:

- Users have reliable internet access to enable cloud sync and collaboration.
- Users have basic digital literacy and familiarity with note-taking tools.
- Cloud services (hypothetical) are available and functional for syncing.
- Users will use up-to-date browsers or mobile devices.
- The system depends on a stable backend database structure for storing users and notes.
- Real-time collaboration assumes that all users maintain an active connection.

Any changes to these assumptions (like loss of internet, lack of cloud support, or outdated devices) may impact the system's performance or requirements.

3 REQUIREMENTS ENGINEERING

3.1 Requirements Gathering Methods

Multiple techniques were applied to ensure that the system requirements reflect real user needs and practical usage scenarios. The following methods were used:

- Brainstorming Sessions: Conducted among team members to generate initial ideas and explore system possibilities.
- Review of Existing Solutions: Analyzed similar applications such as Notion and Todoist to identify common features and usability standards.
- Daily Workflow Observation: Considered typical user routines to determine essential functionalities for efficient note-taking.
- Interviews with Students and Teachers: Short structured meetings were held with a sample of students and instructors to better understand academic note-taking challenges, preferred features, and usability expectations.

3.2 Requirements Analysis

After gathering the data, the requirements were systematically categorized into functional, non-functional, and interface requirements. Each requirement was evaluated based on its feasibility, importance to system performance, and overall impact on user experience. Priorities were assigned to ensure that core functionalities are delivered first.

3.3 Requirements Validation

To ensure accuracy and completeness, all requirements underwent a validation process that included team review meetings, cross-checking with interview feedback, and documentation through agendas and meeting minutes. Requirements were assessed for clarity, consistency, correctness, and completeness before approval.

4 FUNCTIONAL REQUIREMENTS

4.1 User Registration and Authentication:

- The system must allow users (students, teachers, admins) to create accounts with a secure login mechanism, ensuring role-based access control.

4.2 Multimedia Integration:

- The system must support adding and embedding various media types, such as videos, audio recordings, and links, to enhance the notes with diverse content beyond text.

4.3 Search and Categorization:

- The system must support keyword-based search functionality for finding notes, as well as categorizing them by subject or topic.

4.4 Synchronization Across Devices:

- Notes should be synchronized across all devices where the user logs in, ensuring access to updated notes at any time.

4.5 Create Note:

- The system shall allow the user (students, teachers, admins) to create a new digital note with a title and content.

4.6 Edit Note:

- The system shall allow the user (students, teachers, admins) to edit the title, content, tags, and attachments of an existing note.

4.7 Delete Note:

- The system shall allow the user (students, teachers, admins) to delete a note.When deleting, the user can either:
  1. Move to Trash – the note is moved to the Trash and can be restored or permanently deleted later
  2. Permanently Delete – the note is permanently deleted from the trash and cannot be recovered.

4.8 Shared Notes:

- The system shall allow the user (students, teachers, admins) to share a note with other users by granting them view or edit permissions.

5 NON-FUNCTIONAL REQUIREMENTS

5.1 User-Friendly Interface:

- The interface shall be simple, intuitive, and easy to use for students, teachers, and admins without any training.

5.2 System Availability:

- The system shall be available 24/7 with minimal downtime, ensuring users can access their notes at any time.

5.3 Performance Efficiency:

- The system shall load notes and display content within one second, even when the user has a large number of notes.

5.4 Devices Compatibility:

- The system shall work smoothly on different devices including computers, tablets, and smartphones.

6 EXTERNAL INTERFACE REQUIREMENTS

6.1 User Interface

The system shall provide an interactive graphical interface accessible through a web browser or mobile application.
The interface must include:

- A dashboard displaying all notes with title, creation date, and tags.
- A rich-text note editor allowing users to write, format, and save notes.
- A tag-management interface for adding, editing, and deleting tags.
- A responsive layout that adapts to mobile, tablet, and desktop screen sizes.

6.2 Software Interface

The system shall connect to an authentication service to handle:

- Secure login and registration using email and password.
- Password encryption before transmission.
- Error messages for invalid credentials or duplicate email addresses.
- All communication must occur through HTTPS protocols.

6.3 Cloud Database Interface

The system shall connect to a secure cloud-based storage service (e.g., Firebase, AWS, or equivalent) to store:

- User accounts
- Notes
- Tags
- Collaboration updates
- Activity history
- The database must support real-time synchronization for collaborative editing.

6.4 Collaboration & Real-Time Communication Interface

The system shall integrate with a real-time communication service (e.g., WebSockets or Firebase Real-Time Database) to enable:

- Live note editing by multiple users
- Instant updates for changes made by collaborators
- Real-time comments on shared notes

Updates must appear within 0.5 seconds under normal network conditions.

6.5 Search Engine Interface

The system shall integrate with an internal search engine module capable of scanning:

- Note titles
- Note content
- Tags
- Metadata (dates, authors, collaborators)

Search results must be retrieved within 2 seconds.

6.6 Device & Hardware Interface

The system shall operate on:

- Laptops and desktops using modern browsers
- Tablets and mobile devices
- Devices with at least 2 GB RAM and stable internet connectivity
- The system must support input from keyboard, touch screens, and stylus pens when editing notes.

7 UML: USE CASE DIAGRAM

1. Use Case Scenario — Create Note

Primary Actor: User
Supporting Actor: Cloud System
Goal: To create and save a new digital note.
Main Flow:

1. The user selects "Create Note."
2. The system displays the note creation interface.
3. The user enters the note title and content.
4. The user may add multimedia elements (images/audio/video). (include Add Multimedia)
5. The user selects "Save."
6. The system saves the note locally.
7. The system synchronizes the note with the Cloud System. (include Sync Notes)
8. The system confirms that the note has been successfully created.

Alternative Flow:

- 7a. If the Cloud is not available → system stores the note locally and attempts sync later.

2. Use Case Scenario — Edit Note

Primary Actor: User
Supporting Actor: Cloud System
Goal: To modify an existing note.
Main Flow:

1. The user selects a note to edit. (include Select Note)
2. The system displays the note content.
3. The user modifies title, content, or tags.
4. The user may add multimedia. (include Add Multimedia)
5. The user saves the updated note.
6. The system updates the note locally.
7. The system synchronizes updates with the Cloud System. (include Sync Notes)
8. The system confirms that the note has been successfully updated.

3. Use Case Scenario — Move to Trash

Primary Actor: User
Goal: To change the note status to trashed.
Main Flow:

1. The user selects a note. (include Select Note)
2. The user chooses "Move to Trash."
3. The system updates the note status to "Trashed."
4. The system confirms the note has been moved to Trash.

4. Use Case Scenario — Restore From Trash

Primary Actor: User
Supporting Actor: Cloud System
Goal: To restore a trashed note back to active status.
Main Flow:

1. The user views trashed notes.
2. The user selects a trashed note to restore.
3. The system updates the note status to "Active."
4. The system synchronizes the restored note with the Cloud System.
5. The system confirms that the note has been restored.

5. Use Case Scenario — Share Note

Primary Actor: User
Supporting Actor: Cloud System
Goal: To share a note with another authorized user.
Main Flow:

1. The user selects a note to share. (include Select Note)
2. The user chooses "Share Note."
3. The user sets permission: View or Edit.
4. The system uploads shared content to the Cloud System.
5. The Cloud System gives authorized access to the shared user.
6. The system confirms successful sharing.

6. USE CASE SCENARIO — SYNC NOTES

Primary Actor: User

Supporting Actor: Cloud System

Goal: To keep notes updated across devices.

Main Flow:

1. The user triggers Sync or it occurs automatically.
2. The system requests latest note data from Cloud System.
3. Cloud System returns updated items.
4. The system synchronizes notes locally.
5. The system displays success.

---

7. USE CASE SCENARIO — MANAGE USERS

Primary Actor: Admin

Goal: To manage user accounts.

Main Flow:

1. Admin selects "Manage Users."
2. System displays user accounts list.
3. Admin selects a specific user.
4. Admin chooses an action:
5. 4.1 Edit User Info (include Edit User Info)
6. 4.2 Delete User (include Delete User)
7. System applies changes.
8. System confirms updates.

---

8. Use Case Scenario — View System Reports

Primary Actor: Admin

Goal: To monitor system performance.

Main Flow:

1. Admin selects "View System Reports."
2. The system retrieves analytics data.
3. The system generates visual/system reports.
4. Admin views or exports reports.

9. USE CASE SCENARIO — LOGIN

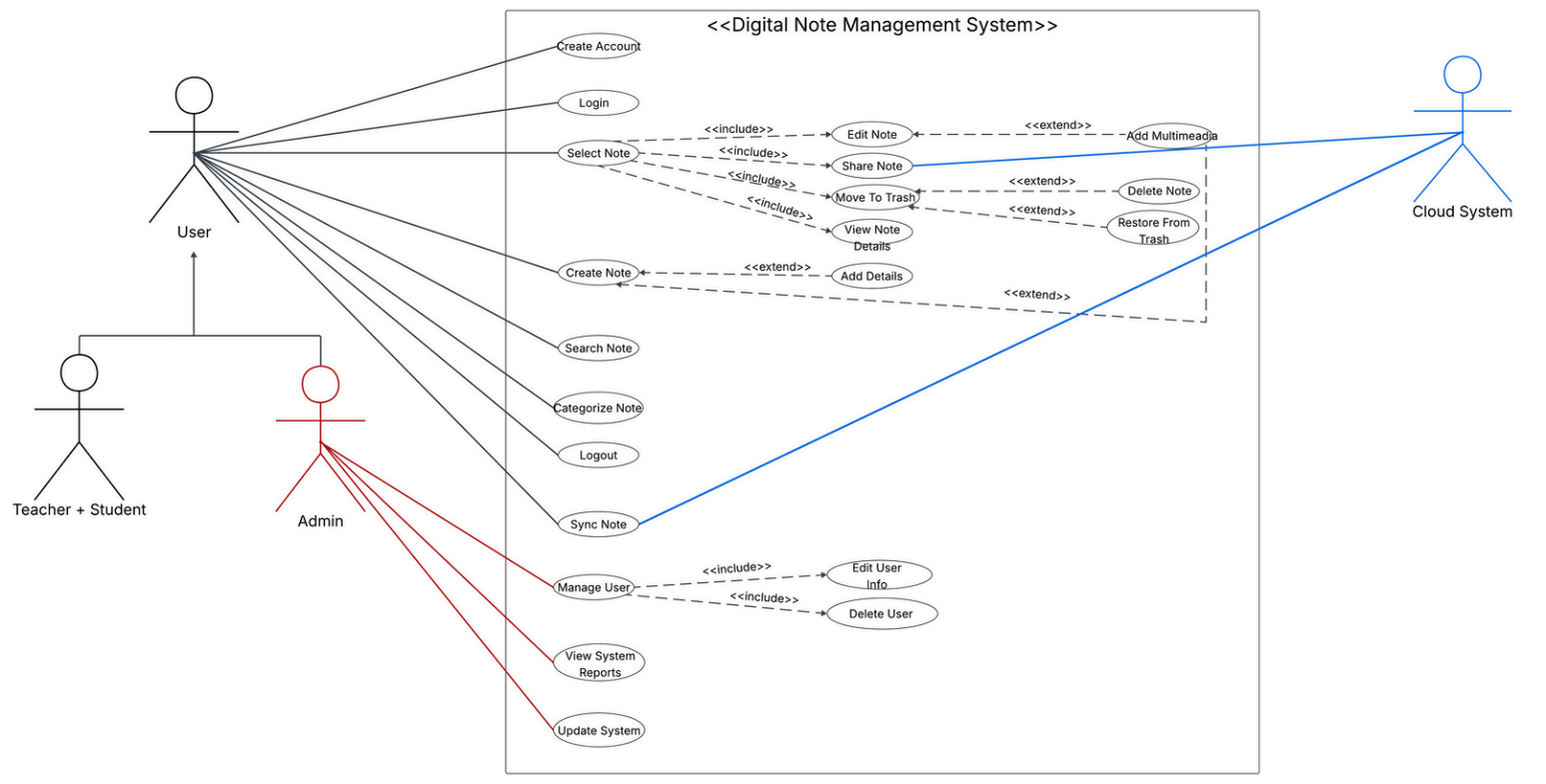
Primary Actor: User / Admin

Goal: To access the system securely.

Main Flow:

1. Actor enters valid credentials.
2. System checks authentication.
3. System applies role-based access control.
4. Actor is redirected to proper dashboard.

7 UML: USE CASE DIAGRAM



DIAGRAM'S LINK

8 SOFTWARE ENGINEERING ETHICS AND RESPONSIBILITIES:

8.1 Ethics in Public

1. Preserve User Privacy
   - Make sure that all notes created by users are treated with confidentiality and that robust measures are in place to stop misuse or illegal access.

2. Assure Data Security
   - To shield users from data breaches or identity exposure, all notes should be stored, encrypted, and transmitted securely.

3. Offer Transparency
   - Explain in detail how data is used, processed, and stored. Users ought to be aware of the data that is gathered and its purpose.

4. Preserve System Dependability
   - Provide a dependable and stable note-taking system that reduces downtime and guards against user data loss.

5. Encourage Usability
   - Make sure that note management features are accessible to people with varying skill levels by designing the system to be user-friendly.

6. Steer clear of harmful use
   - Implement safeguards against system abuse for detrimental, unlawful, or unethical activities.

8.2 SELF Ethics:

1. Produce High-Quality Work
   - Developers shall strive to implement features (like note creation, syncing, and sharing) in a reliable, accurate, and professional manner.

2. Continuous Learning
   - Engineers working on the system shall stay updated with new technologies related to cloud storage, collaboration tools, and secure note management.

3. Produce Clear and Accurate Documentation
   - Engineers must ensure all SRS, SDD, diagrams, and technical reports are written clearly, accurately, and free of misleading information.

8 SOFTWARE ENGINEERING ETHICS AND RESPONSIBILITIES:

8.3Product Ethics

1.Ensuring Product Quality

- The team strives to deliver a reliable and high-quality note-management system. This includes following proper design standards, reviewing requirements thoroughly, and ensuring all features align with user needs and expectations.

2.Protecting User Safety and Data

- The system is designed to protect users from data breaches, unauthorized access, and loss of information. This is achieved by applying essential security practices such as encryption, authentication, and conceptual backup mechanisms.

3.Maintaining Transparency and Honesty

- The team commits to presenting the system's capabilities honestly and without exaggeration. All documented functionalities reflect the actual design and intended behavior of the system, ensuring users are not misled.

4.Practicing Verification and Testing

- Each feature is carefully analyzed and conceptually tested during the design phase. Requirements are validated for feasibility, correctness, and consistency to prevent errors and ensure system integrity.

5.Enhancing Usability

- The product is designed to be easy to use and accessible to different user groups, including students, teachers, and professionals. User-friendly interface principles are followed to ensure smooth navigation and a clear, organized experience.

6.Preventing Potential Harm

- The team ensures that the system does not cause harm to users by avoiding design flaws that could lead to data loss, errors, or misuse. Safeguards are proposed to reduce risks and protect users from negative consequences.

8 SOFTWARE ENGINEERING ETHICS AND RESPONSIBILITIES:

8.4 Management :

- Management ethics ensures the project is planned, organized, and executed responsibly, fairly, and transparently. The management team coordinates members, assigns tasks fairly, sets realistic deadlines, and ensures equal participation. Ethical management promotes honest communication, respects differences, handles conflicts respectfully, and documents decisions based on logic and ethics. It also safeguards privacy, protects project files, avoids plagiarism, and properly cites all sources, fostering a fair and positive team environment.

8.5 Judgment :

1. Focus on Key Features
- The system prioritizes essential features that add real value, such as easy note-taking, search functionality, and synchronization across devices.
2. Avoid Overcomplicating the User Experience
- Complex features that could confuse users are avoided, ensuring the interface remains simple and clear.
3. Ensure User Data Protection
- User data is secured, with all information encrypted and protected to ensure privacy and safety. User consent is always obtained for any data usage.

8.6 Professional Ethics :

1. Ethical Conduct and Responsibility
- The team maintained fairness, honesty, accountability, and respectful communication in all project activities.

2. Continuous Learning and Professional Growth
- Members supported each other in understanding requirements, modeling, and documentation while actively improving their skills.

3. Accuracy and Transparency
- Project information was communicated clearly, and documents were regularly reviewed to ensure correctness and consistency.

4. Compliance with Standards, Laws, and Licensing
- All tools and resources were legally used, and documentation followed academic and professional guidelines.

5. Respect for Intellectual Property
- All diagrams, texts, and designs were original or properly cited to uphold copyright and intellectual property rights.

6. Commitment to Quality and Professionalism
- The team focused on producing high-quality work, ensuring that planning, development, and documentation met professional standards.

8 SOFTWARE ENGINEERING ETHICS AND RESPONSIBILITIES:

8.7 Colleagues :

1. Honor the work of your teammates.
   - Give due credit for ideas and implementations, acknowledge the contributions of others, and refrain from misrepresenting their work.

2. Be Honest and Professional in Your Communication
   - Communicate truthful information, voice concerns politely, and uphold integrity when making technical decisions and providing progress reports.

3. Work Together Equitably
   - Avoid giving others an unfair workload, assist team members, and participate equally in tasks.

4. Preserve Technical Honesty
   - Don't minimize risks or conceal mistakes. Report problems that could compromise the safety or quality of the system and offer appropriate fixes.

5. Openly Exchange Knowledge
   - To guarantee a seamless and effective development process, assist colleagues in understanding system components, documentation, and tools.

6. Observe the confidentiality of project information.
   - Keep user data, code, and internal documents private.

8 SOFTWARE ENGINEERING ETHICS AND RESPONSIBILITIES:

8.8Client and Employer Ethics :

1. Accurately Meeting Client Requirements
   - The system is designed to fulfill the client's needs exactly as specified, including note creation, organization, searching, and synchronization, without removing essential features or adding unnecessary complexity.

2. Transparency with the Client and Employer
   - The system clearly presents its real capabilities and limitations without exaggeration or false claims. All functionalities behave exactly as documented.

3. Protecting Client Data
   - The system ensures that all user notes, accounts, and stored information remain secure and confidential, preventing unauthorized access or data misuse.

4. Delivering a Reliable and High-Quality System
   - The system must perform consistently, maintain stability, and provide accurate results that meet the quality standards expected by the client and employer.

5. Avoiding Misleading or Unethical Behavior
   - The system does not misrepresent features, hide important information from users, or perform actions that could deceive the client or employer.

6. Prioritizing the Client's Best Interest
   - All design and technical decisions prioritize usability, data protection, and productivity to ensure the system supports the client's goals effectively.

Software Design Document（SDD）

1 SOFTWARE DESIGN AND DOCUMENT (SDD)

1. INTRODUCTION

1.1 Purpose

The purpose of this Software Design Document (SDD) is to describe the architectural design, components, models, and interactions that define the Digital Notes Management System. This document translates the functional and non-functional requirements from the Software Requirements Specification (SRS) into a detailed technical blueprint that guides developers, designers, and stakeholders. It ensures a shared understanding of how the system will be structured and how its features will be implemented.

1.2 Product Scope

The Digital Notes Management System supports students, teachers, and professionals in creating, managing, organizing, and collaborating on digital notes efficiently. The system includes core features such as note creation and editing, categorization through tags, multimedia integration, searching, sharing, cloud-based synchronization, and real-time collaboration.
This SDD defines how these features are modeled, structured, and designed using architectural patterns, components, diagrams, and process models that outline system behavior and interactions.

2. System Overview

The Digital Notes Management System is designed to provide users with an efficient and organized platform for creating, managing, and sharing digital notes. It supports both individual note-taking and real-time collaboration, allowing multiple users to edit and comment on shared notes simultaneously.

2 SOFTWARE ARCHITECTURE DESIGN

2.1 Architecture Overview

The system uses a layered architecture, allowing users to interact with the interface, while the backend services handle business logic, authentication, and data storage.
Separating responsibilities into layers, making the system easier to maintain, scale, and secure.

2.2 Architecture Pattern

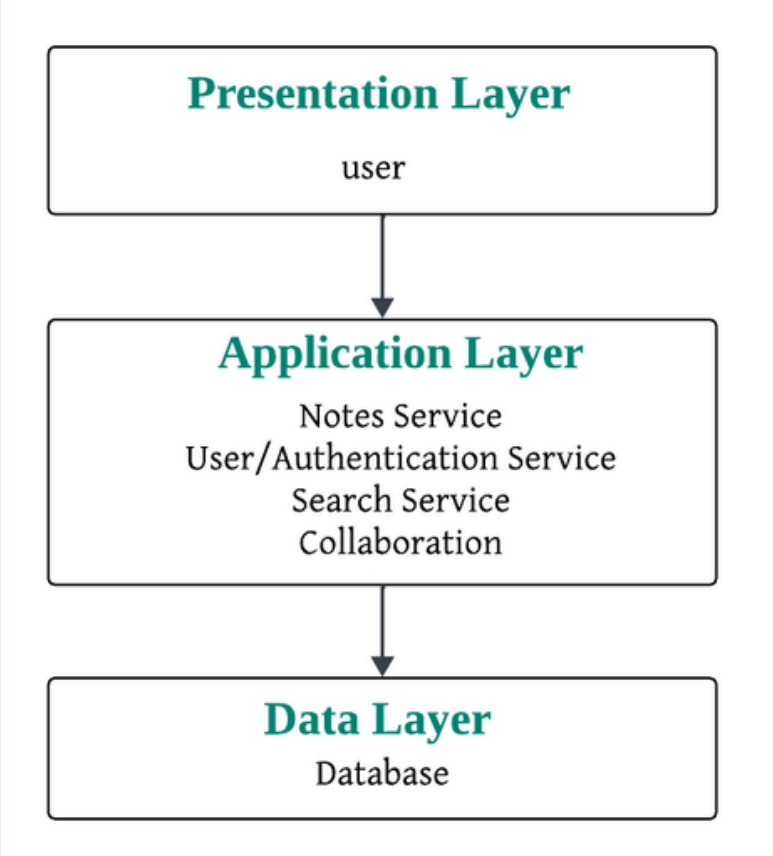
The system is organized into three layers:

1-Presentation Layer:
Handles user interaction through the interface
Collects user input and presents system outputs

2-Application Layer:
Manages backend services including notes management, user accounts, authentication, search functionality, and collaboration features

3-Data Layer:
Stores all users and notes in the database
Ensures data integrity, security, and accessibility to other layers

2.3 Layered Architecture



3. SOFTWARE PROCESS ACTIVITIES AND MODEL

3.1 Selected Software Process Model

The most suitable software process model for the Digital Notes Management System is the Agile (Scrum-based) iterative model. This model supports continuous improvement, flexible updates, and evolving features, which aligns well with the nature of digital note-taking applications.

3.2 Justification for Choosing Agile

1. Flexibility for Evolving Features
The system requires constant enhancements such as improved search, tagging, UI refinements, collaboration features, and cloud synchronization. Agile enables these changes smoothly through short and adaptive development cycles.

2. Supports Continuous User Feedback
Agile encourages releasing small functional parts early and refining them based on user feedback. Since note-taking systems depend heavily on usability and user experience, this feedback-loop is essential.

3. Easier to Implement Major Changes
In Incremental models, adding a major feature often requires modifying earlier components, which slows development. In Agile, refactoring is expected and integrated into the process, making major changes more manageable and less disruptive.

4. Industry Alignment
Modern note-taking applications such as Notion, Evernote, and Google Keep release frequent updates and continuously evolve their features. While these companies do not explicitly state their development models, their update patterns strongly suggest an Agile-like approach, making Agile a realistic and suitable choice for our system as well.

3.3 Agile Software Process Activities

1. Product Backlog Management
All required system features such as note creation, tagging, searching, collaboration, and cloud syncing are listed, prioritized, and continually updated in the product backlog.

2. Sprint Planning
Before each sprint, the team selects a set of high-priority backlog items (notes module, search module, UI updates, etc.) and plans how these features will be designed or refined during the sprint.

3. Iterative Design and Development (Sprint Execution)
During each sprint, the team works on small increments of the system, focusing on iterative improvements to features like the note editor, tagging system, or synchronization components.

4. Sprint Review
At the end of every sprint, the team evaluates the completed increment, reviews how features perform, and checks whether they meet the intended requirements of the Digital Notes Management System.

5. Sprint Retrospective
The team analyzes what went well, what needs improvement, and how the next sprint can be more efficient ensuring continuous enhancement of both the process and the product.

6. Continuous Feedback Integration
User feedback (or stakeholder feedback in academic context) is incorporated into the backlog and influences future sprints. This supports usability improvements and evolving system features.

3.4 Mapping Sprints to System Features

Sprint 1	User registration, login, and main dashboard.
Sprint 2	Note creation, editing, deletion, and tagging.
Sprint 3	Search functionality and note organization.
Sprint 4	Hypothetical cloud synchronization and real-time collaboration.

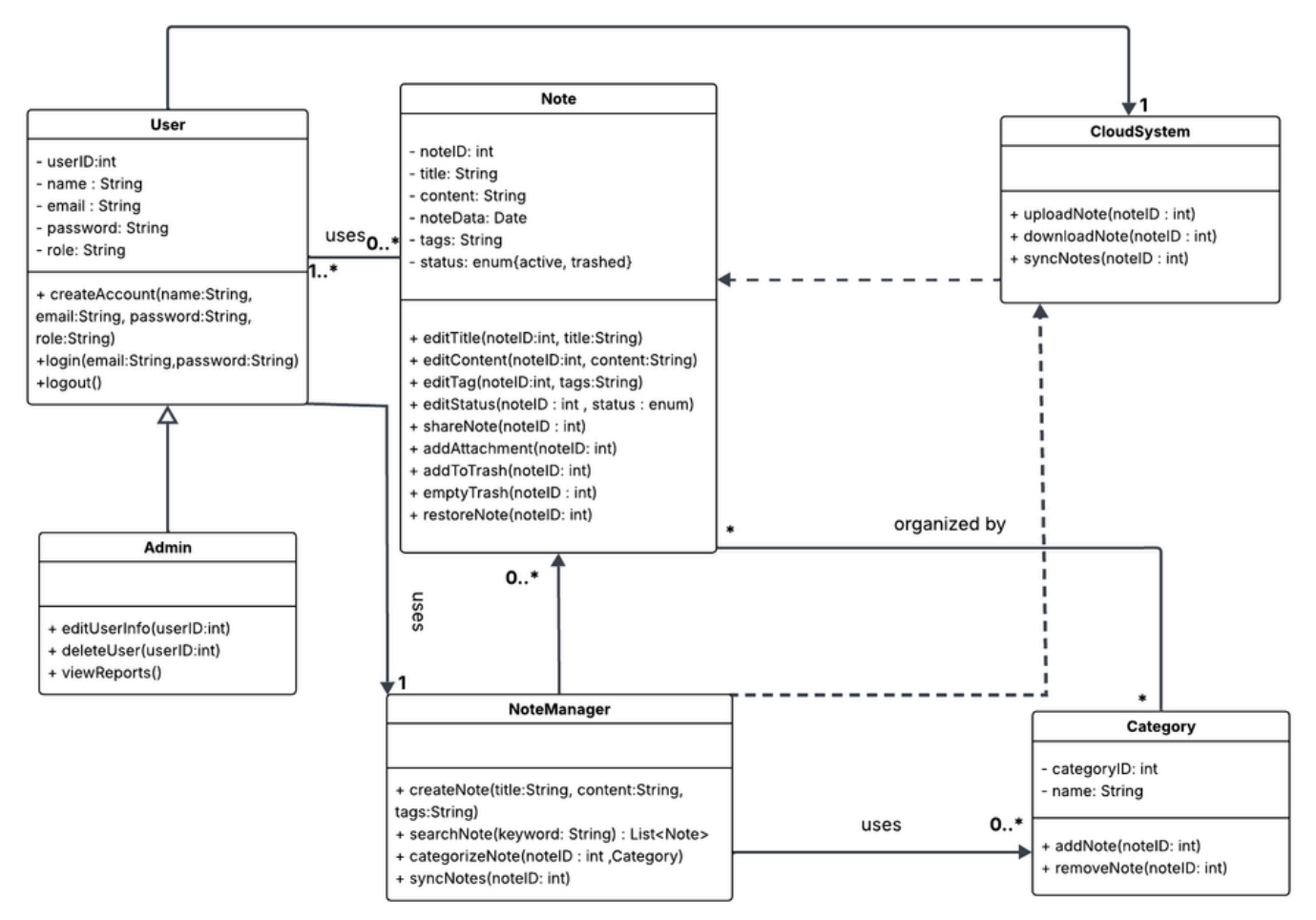
4 COMPONENT DESIGN

4.1 CLASS DIAGRAM

In a digital note management system, a registered user logs in using their email and password. After authentication, the user interacts mainly with the NoteManager, which is responsible for handling note operations. Through the NoteManager, the user can create new notes by providing a title, content, and optional tags. Each created note is stored as a Note object, which keeps information such as the note ID, creation date, tags, and status (active or trashed).

Once notes exist, the user can perform various actions on them. The NoteManager allows the user to search for notes by keyword, categorize notes into different Category objects, and synchronize selected notes with the CloudSystem. At the note level, the user may edit the title or content, update tags, change the status (for example, move a note to trash), delete a note, share it, attach files, or restore a trashed note. When synchronization is requested, the NoteManager calls the CloudSystem to upload or download note data so the user's notes stay consistent across devices.

An Admin is a specialized type of user with additional privileges. The admin can edit user information, delete user accounts, and view system reports to monitor usage and maintain the system. Both regular users and admins eventually log out once they finish managing notes or administrating the system.



DIAGRAM'S LINK

4.2 STATE DIAGRAM

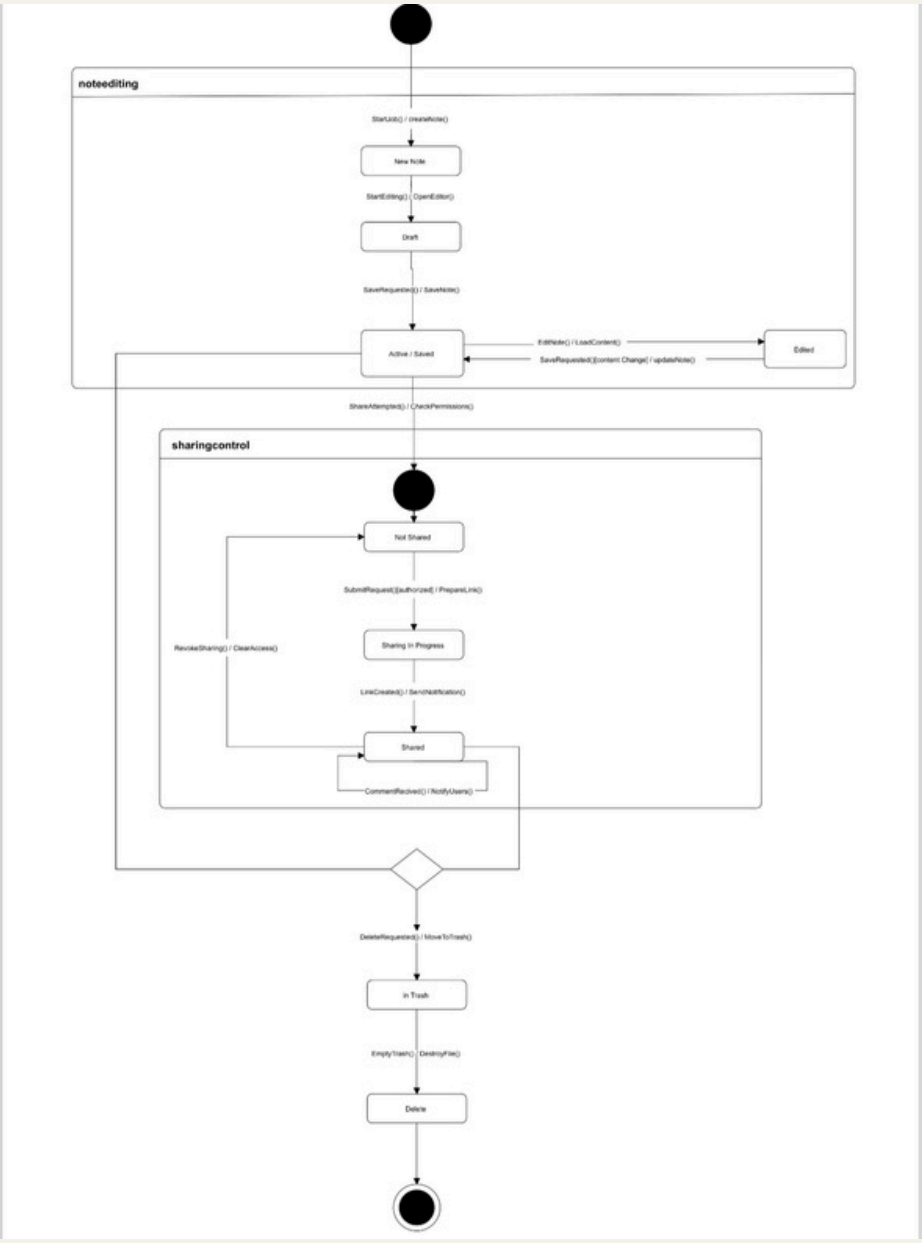
The state diagram describes the lifecycle of a digital note in the system. It shows how a note transitions between different states during creation, editing, sharing, and deletion.

The process begins in the Note Editing composite state. The user first creates a new note (CreateNote()), which moves into the Draft state once editing starts. After saving the content (SaveNote()), the note becomes Active/Saved. If the user edits the note again and saves the updated content, the note transitions to the Edited state.

Next, the note may enter the Sharing Control composite state. By default, every note starts as Not Shared. When the user initiates a sharing request (ShareRequest()), the note moves to Sharing in Progress while permissions and links are prepared. Once sharing is completed, the note becomes Shared, and collaborators may receive updates or add comments. If sharing is revoked, the note returns to Not Shared.

For deletion, the note moves to In Trash when the user performs a delete request (MoveToTrash()). It can stay there temporarily until the user permanently deletes it. When EmptyTrash() is executed, the note enters the final Deleted state.

Overall, the diagram clearly represents each stage a note goes through, reflecting the system's behavior and user interactions accurately.

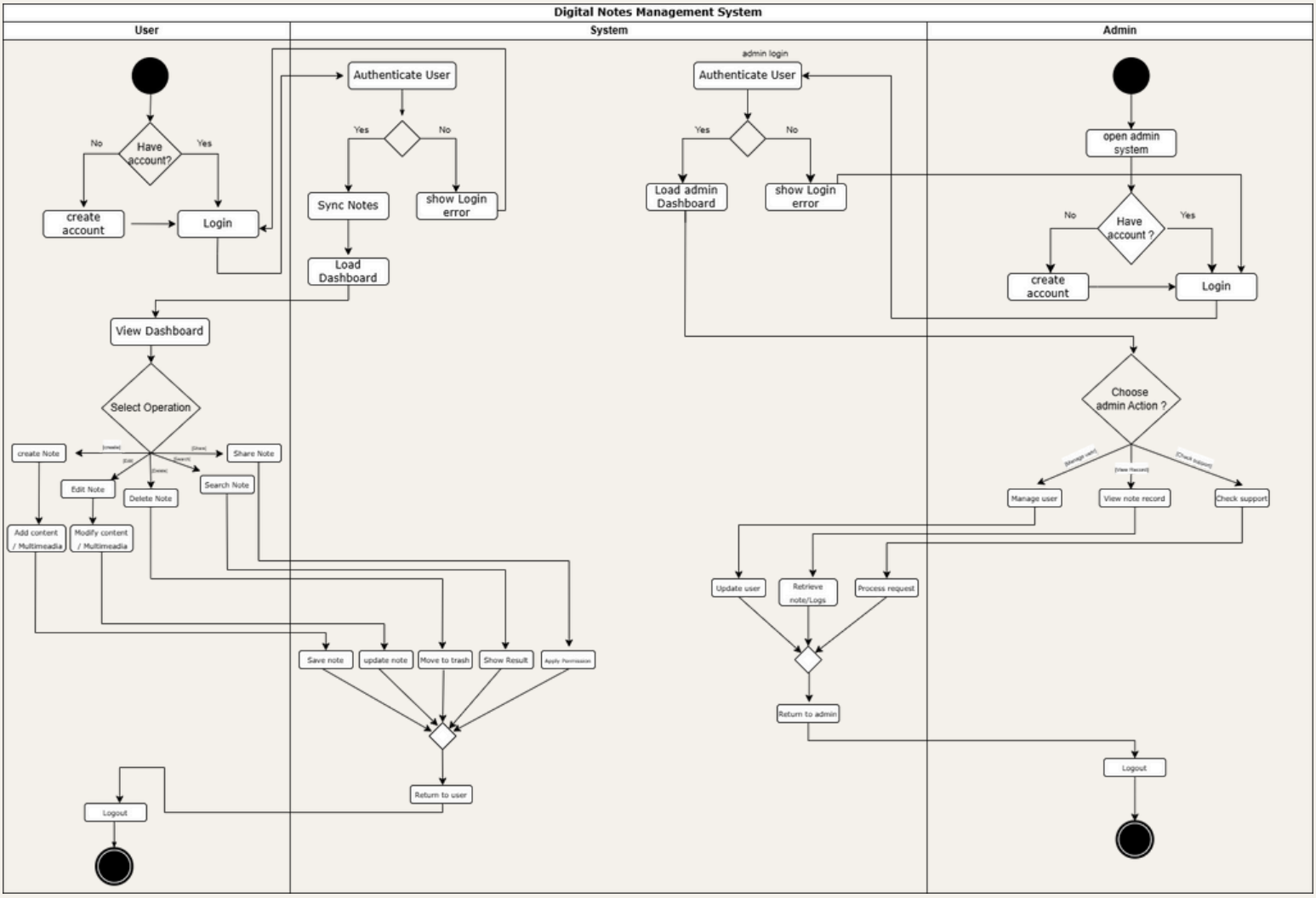


DIAGRAM'S LINK

4.3 ACTIVITY DIAGRAM

In a digital notes management system, a user accesses the application to perform a variety of note-related operations. A user may create an account or log in if an account already exists. After successful authentication, the user is allowed to create new notes, edit existing notes, delete notes, search for notes, or share them with other users. Each operation triggers corresponding system functions such as saving, updating, retrieving, or applying sharing permissions.

An administrator may also access the system to manage user accounts, review note records, or handle support requests. The system authenticates the administrator before allowing administrative actions. Both users and administrators may log out after completing their activities.



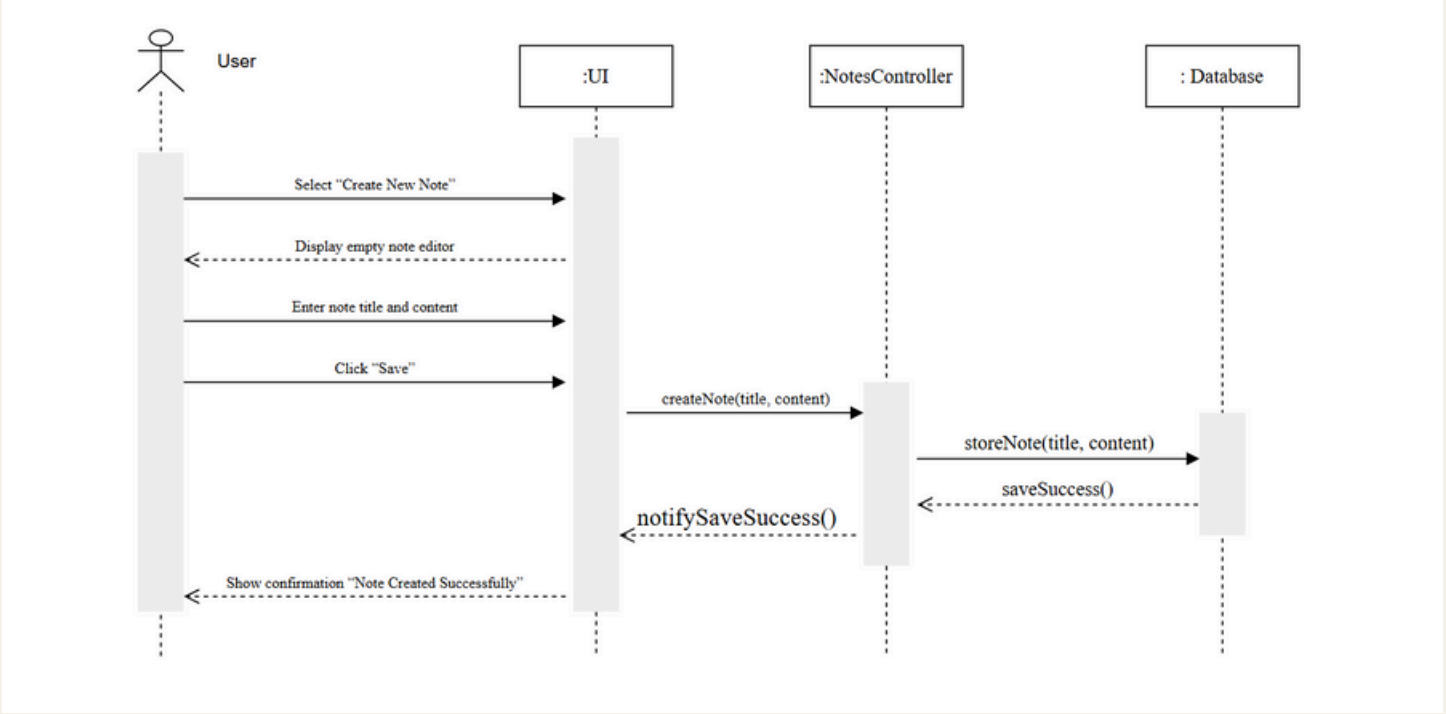
DIAGRAM'S LINK

4.4 SEQUENCE DIAGRAM

CREATE A NOTE

The user begins by selecting “Create New Note.” The system displays an empty note editor where the user enters the note’s title and content. When the user clicks “Save,” the UI sends a createNote request to the Notes Controller.

The controller validates the data and forwards it to the database to store the new note. After the note is successfully saved, the database sends a confirmation back to the controller, which then notifies the UI. Finally, the system displays a success message indicating that the note has been created.



DIAGRAM’S LINK

5 CONTRIBUTIONS TABLE

Member	Contribution
Rawan Alfouzan	• Team meeting documentation • Report presentation • State Diagram design • Functional Requirements specification • Judgament ethics
Raghad Alharbi	• Introduction • System Overview • Requirements Engineering section • Sequence Diagram design • Management ethics
Jana Almishali	• Functional Requirements specification • Activity Diagram design • Non-Functional Requirements specification • Self ethics
Aseel Alowairdhi	• Sequence Diagram design • Software Architecture Design • Software Design Document (SDD) • Profession ethics
Kadi Albakri	• Team meeting documentation • Class Diagram design • UML diagram • Colleagues ethics • Repository
Bilsan Almanaee	• Team meeting documentation • Class Diagram design • UML diagram • Public ethics
Aljazi Aleqab	• Team meeting documentation • Software Process Model • State Diagram design • External Interface Requirements • Client & employer ethics
Lubna Alawad	• Software Process Model • Activity Diagram design • Software Design Document (SDD) • Product ethics

Project Meetings Notes

First Meeting Agenda

Meeting Agenda

Goal: Idea Selection & Review Requirements

Date& Time: 28/10/2025 and 5:30AM

Avenue: Microsoft Teams

Chair : Bilsan Almanaee

Client :

Attendance : All 8 members

Absent with apologies : None

Agenda Items

Area	Goal	Time
Brainstorming Session	All students cooperate in brainstorming.	**2min**
Idea	Idea selection.	**3min**
Review Project Requirments	Confirm understanding of deliverables and grading criteria.	**5min**
Task Management	Task Management.	**10min**
Plan Repository Structure	Decide folder structure and branching strategy.	**2min**
Ethical Principles Identification	Identify ethical principles relevant to the note-taking system.	**3min**
Ethics Topic Assignment	Assign each ethical topic to a team member.	**1min**

Digital Notes Management System

Meeting Number : 1

Meeting Minutes

Goal: Idea Selection & Review Requirements

Date& Time: 28/10/2025 and 5:30AM

Avenue: Microsoft Teams

Chair : Bilsan Almanaee

Client :

Attendance : All 8 members

Absent with apologies : None

Action items :

1. Idea has been selected : “Digital Management System”.
2. Project Requirments reviewed.
3. Meeting Agenda & Minutes Assignment:
   - Bilsan Almanaee → First 28-29 OCT
   - Kadi Albakri → Second 1-2 NOV
   - Rawan Alfouzan → Third 9-10 NOV
   - Aljazy Aleqab → Forth 23-24 NOV
4. Create Google Drive Repository
5. Eight ethical principles selected and assigned: 31 OCT
   - Bilsan → Public
   - Kadi → Colleagues
   - Rawan → Judgment
   - Aljazy → Client and Employer
   - Lubna → Product
   - Aseel → Profession
   - Jana → Self
   - Raghad → Management

Second Meeting Agenda

Meeting Agenda

Goal: Model Selection & SDD and SRS Planning

Date& Time: 1/11/2025 and 4:30AM

Avenue: Microsoft Teams

Chair : Kadi Albakri

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Agenda Items

Area	Goal	Time
Process Model Selection	Choose the most suitable development model for the project.	**3min**
SDD Planning	Identify what sections must be included in the Software Design Document.	**2min**
Requirements-Gathering Plan	Define how functional and non-functional requirements will be collected.	**5min**
Use Case Diagram Planning	Determine the main actor(s) and actions to include in the diagram.	**8min**
Functional & Non-Functional Requirements List	Identify at least eight core system functions and identify at least four quality and performance constraints.	**5min**
External Interface Requirements	Define at least three required interfaces.	**5min**

Digital Notes Management System

Meeting Number : 2

Meeting Minutes

Goal: Model Selection & SDD and SRS Planning

Date& Time: 1/11/2025 and 4:30AM

Avenue: Microsoft Teams

Chair : Kadi Albakri

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Action items :

1. The team discussed and selected the most suitable software process model for the project.
2. The team reviewed and identified the necessary sections to be included in the SDD.
3. The group planned the requirements-gathering approach, deciding how functional & non-functional will be collected.
4. The use case has been discussed, the main actor(s) and key system actions have been identified.
5. Members worked on listing the functional and non-functional requirements.
6. The group identified and defined the external interface requirements.

Third Meeting Agenda

Meeting Agenda

Goal : SRS Progress Check & Kickoff for SDD

Date and Time : 10 / 11 / 2025

10AM - 12PM (2 hours)

Avenue : Group chat

Chair : Rawan Alfouzan

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Agenda Items

Area	Goal	Time
Progress Updates	• Attendees: All 8 members Each member shares what they have started working on. • Overview of completed and in-progress tasks.	**7min**
SRS Draft Review	• Discuss current status of the Software Requirements Specification (SRS) Part. • Identify which sections have been completed and which are still pending.	**10min**
Start of SDD Phase	• Brief introduction to the Software Design Document (SDD).	**8min**
Deadlines and Timeline	• Finalize what needs to be done before next meeting. • Planning next team meeting.	**5min**

Digital Notes Management System

Meeting Number : 3

Meeting Minutes

Date and Time : 10 / 11 / 2025

10AM - 12PM (2 hours)

Avenue : Group chat

Chair : Rawan Alfouzan

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Action items :

1.Reviewed the current draft of the SRS document.

2.Clarified several points related to the SRS structure and content.

3.Officially started work on the SDD phase.

4.Discussed and listed the key diagrams required.

5.Set the date of the next meeting (a week after)

Fourth Meeting Agenda

Meeting Agenda

Goal : SDD Review & Project Review

Date and Time : 23 / 11 / 2025

5pm-6pm (1 hour)

Avenue : Zoom

Chair : Aljazi Aleqab

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Area	Goal	Time
Diagrams Review	• Go through all submitted diagrams related to the SDD. • Discuss accuracy, clarity, and completeness.	**10min**
Full Project Review	• Go through all parts of the project: documents, diagrams, and any deliverables. • Verify completeness and consistency. • Ensure visual and content clarity.	**15min**
Final Formatting & Polish	• Ensure all files are properly formatted and well-organized.	**10min**
Submission Readiness	• Confirm that everything is ready for submission.	**10min**

Digital Notes Management System

Meeting Number : 4

Meeting Minutes

Date and Time : 23 / 11 / 2025

5pm-6pm (1 hour)

Avenue : Zoom

Chair : Aljazi Aleqab

Client : Students and Teachers

Attendance : All 8 members

Absent with apologies : None

Action items :

1.Reviewed all current diagrams in the SDD.

2.Gave feedback and decided on several changes and improvements.

3.Ensured everyone clearly understood the structure and purpose of each diagram.

4.Ensured everything was complete, accurate, and properly formatted.

5.Prepared and rehearsed the presentation.

6.Confirmed submission readiness.

Repository Link

Click Here