

**MINOR PROJECT ON:**

## **Personalized Notes Management System**

**Minor Project Submitted in Partial Fulfillment of the Requirements for  
the degree of**

**Master of Computer Application**

**Submitted by:**

<b>Preeti Mitra</b>	<b>13071023025</b>
<b>Sumanta Paul</b>	<b>13071023044</b>
<b>Sumit Mishra</b>	<b>13071023045</b>

**Under the guidance of  
Mr. Surojit Roy  
Assistant Professor, Dept. Of MCA  
Techno Main Salt Lake, Kolkata-91**



**Techno Main Salt Lake  
EM 4/1 Salt Lake City, Sector V  
Kolkata 700091**

**Year - 2024**

(Maulana Abul Kalam Azad University of Technology)

Techno Main, Salt Lake

FACULTY OF MCA DEPARTMENT

### **Certificate of Recommendation**

This is to certify that Preeti Mitra, Sumanta Paul, Sumit Mishra have completed his Minor project (MCAN-381) work titled “**Minor project on: Personalized Notes Management System**”, under the direct supervision and guidance of **Asst. Prof. Surojit Roy**. We are satisfied with his work, which is being presented for the partial fulfillment of the degree of Master of Computer Application (MCA), Maulana Abul Kalam Azad University of Technology, Kolkata-700064.

---

Asst. Prof. Surojit Roy  
Assistant Professor, Dept. of MCA  
Techno Main Salt Lake, Kolkata-91  
Date:

---

Prof. (Dr.) Shiladitya Chowdhury  
HOD MCA Department  
(Techno Main, Salt Lake)  
Date:

(Maulana Abul Kalam Azad University of Technology)

Techno Main, Salt Lake  
FACULTY OF MCA DEPARTMENT

### **Certificate of Approval \***

The foregoing Minor project is hereby approved as a creditable study of Master of Computer Application (MCA) and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or any statement made, opinion expressed or conclusion therein but approve this Minor project only for the purpose for which it is submitted.

---

---

---

Signature of the examiners

Final Examination for  
Evaluation of the Project

\* Only in case the Minor project is approved

## PREFACE

In today's fast-paced digital world, efficient management of information is crucial. Our project, **NoteNest** Notes Management System, aims to provide a streamlined and user-friendly platform for managing personal and professional notes. This web-based application allows users to add, edit, search, and delete notes with ease, ensuring that important information is always at their fingertips.

The system is designed with simplicity and functionality in mind, catering to users who need a reliable tool to organize their thoughts, tasks, and ideas. By integrating essential features such as note creation, modification, search capabilities, and deletion, our Notes Management System offers a comprehensive solution for note-taking and management.

This project not only enhances productivity but also ensures that users can efficiently manage their notes in a secure and organized manner. Whether for personal use or professional purposes, our Notes Management System is an indispensable tool for anyone looking to keep their information well-organized and easily accessible.

As a minor project for academic purposes, this initiative demonstrates the practical application of programming languages like PHP, JavaScript, and tools like MySQL, Bootstrap, HTML, and CSS. It not only highlights the technical capabilities of the development team but also showcases an understanding of real-world business operations.

This preface serves as a reflection of our dedication to creating impactful digital solutions and our commitment to fostering a deeper understanding of the technological frameworks that drive modern systems. We believe **NoteNest** will contribute significantly to enhancing efficiency and precision in Notes Management.

## Goal of the Minor Project

### Goals

**User-Friendly Interface:** Ensure the website is easy to navigate, allowing users to quickly add, edit, search, and delete notes without any confusion.

**Efficient Note Management:** Implement features that allow users to organize their notes effectively, such as categorization, tagging, and sorting options.

**Search Functionality:** Develop a robust search feature that enables users to find specific notes quickly using keywords or phrases.

**Data Security:** Ensure that all notes are securely stored and protected from unauthorized access. Implement user authentication and encryption where necessary.

**Responsive Design:** Make sure the website is accessible and functional on various devices, including desktops, tablets, and smartphones.

**Performance Optimization:** Optimize the website for fast loading times and smooth performance, even with a large number of notes.

**Scalability:** Design the system to handle an increasing number of users and notes without compromising performance.

**Backup and Recovery:** Implement a backup system to prevent data loss and ensure that users can recover their notes in case of accidental deletion or system failure.

**User Feedback and Support:** Provide a way for users to give feedback and access support if they encounter any issues with the system.

**Accessibility:** Ensure the website is accessible to users with disabilities, following web accessibility standards.

These goals will help you create a comprehensive and efficient notes management system that meets user needs and provides a seamless experience.

## Organization of the Minor Project

**Chapter 1** describes the scope of the project. Determination of software project planning and it is of utmost importance for the project scope to be unambiguous and understandable at the management and technical levels. It describes the data and control to be processed, function, performance, etc. in a nutshell, it describes what the project deals with.

**Chapter 2** deals with the concepts and problem analysis comprising of Cost Analysis, Time Analysis, Team Structure, Software Configuration Management Plan, Quality Assurance Plan and Risk Management.

Cost Analysis involves estimating the cost involved in developing the project. In this case, the project, in consideration, is a minor project. Therefore, only the effort required for developing the project and the development time are estimated.

Time analysis involves an analysis of the time required for completing the different phases of the project.

Team structure describes the organization of the people directly involved in the software project. Certain factors including the difficulty of the problem to be solved, the size of the resultant program(s) in lines of code or function points, the team lifetime, the degree of sociability (communication) required for the project, etc. need to be considered when planning the structure of the team.

Software Configuration Management Plan defines the project strategy for SCM (Software Configuration Management) which, in turn, is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.

Software quality assurance plan is created to define the software team's SQA (Software Quality Assurance) strategy.

Risk management involves a series of steps that help the software team to understand and manage uncertainty.

**Chapter 3** describes about the technologies, which is required to do the project. The prior knowledge on this technology is essential for developing the system.

**Chapter 4** deals with the system requirements for developing the project.

**Chapter 5** describes the design of the project comprising of data design, architectural design, interface design and procedural design.

**Chapter 6** deals with the coding standard that has been followed while developing the project so as to promote easy understanding and readability of the code.

**Chapter 7** involves evaluating the outcomes of various phases of the Project. This analysis helps in understanding the effectiveness, efficiency, and quality of the software being developed.

**Chapter 8** describes the testing strategies that have been used for the project to uncover the errors associated with it. To ensure that the product is of good quality, tests have been conducted systematically and test cases have been designed using disciplined techniques.

**Chapter 9** describes the future scope of the project. It highlights all the possible areas in which the project can be improved later so as to enhance its features and functionalities.

**Chapter 10** describes the conclusion from the project.

**Chapter 11** lists the references and bibliography for the project.

**Chapter 12 Appendix 1** contains all possible screenshots of the GUI of the project.

**Chapter 13 Appendix 2** contains code of the project.

## **Table Information and Purpose**

<b>Table No.</b>	<b>Table Caption</b>	<b>Purpose</b>	<b>Page No.</b>
<b>1</b>	<b>Complexity Table</b>	<b>Calculation of FPA</b>	<b>04</b>
<b>2</b>	<b>Assigning Weights</b>	<b>Calculation of FPA</b>	<b>04</b>
<b>3</b>	<b>Testing Table</b>	<b>Testing</b>	<b>27</b>



## Image Information and Purpose

Image No.	Image Caption	Purpose	Page No.
1	Team Structure	Concepts and Problem Analysis	02
2	ER Diagram	Design	15
3	Level 0 DFD	Design	16
4	Level 1 DFD	Design	16
5	Signup with invalid input	Result Set Analysis	23
6	Signup with valid input	Result Set Analysis	23
7	Homepage	Screenshots	31
8	Signup Page	Screenshots	31
9	Login Page	Screenshots	32
10	Add Note	Screenshots	32
11	Edit Note	Screenshots	33
12	List of Notes	Screenshots	33

## Acknowledgement

We take this opportunity to express our deep gratitude and sincerest thank to our project mentor, **Asst. Prof. SUROJIT ROY** for giving most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

We will like to give a special mention to our colleagues. Last but not the least we are grateful to all the faculty members of MCA department for their support.

Our special thanks to **Asst. Prof. Dr. SHILADITYA CHOWDHURY** (HOD of MCA/BCA of Techno Main Salt Lake) for providing us the opportunity to conduct the project "Petrol Pump Management System".

The project has been developed in a very short period of time. So, error and limitations remain to be found in documentation and in the project. We will gratefully accept from readers and correction, criticism and suggestions for the improvement of our project.

---

---

---

---

---

## INDEX

Minor Project on: Personalized Notes Management System		
Chapter	Statement about the problem	Page No.
1	Scope of the Project	01
2	Concepts and Problem Analysis  2.1 Team Structure 2.2 Function Point Analysis 2.3 Software Configuration Management 2.4 Quality Assurance Plan 2.5 Risk Management	02-08
3	Theoretical Background	09-11
4	Software Requirement Specifications	12-14
5	Design/Solution/Methodology  6.1 Data Design 6.2 Architectural Design 6.3 Interface Design 6.4 Procedural Design	15-20
6	Coding Standard Followed and Assumptions	21-22
7	Result Set Analysis	23
8	Testing	24-26
9	Future Scope of the Project	27-28
10	Conclusion	29
11	References and Bibliography	30
12	Appendix 1	31-33
13	Appendix 2	34-36

# Chapter 1

## Scope of the Project

**Project Overview:** The Notes Management System is a web-based application designed to help users manage their notes efficiently. Users can add, edit, search, and delete notes, ensuring easy access and organization of their information.

**Objectives:** Develop a user-friendly interface for managing notes. Implement functionalities for adding, editing, searching, and deleting notes. Ensure data security and integrity. Provide a responsive design for accessibility on various devices.

**Features:**

- **Add Notes:** Users can create new notes with a title and content.
- **Edit Notes:** Users can modify existing notes.
- **Search Notes:** Users can search for notes using keywords.
- **Delete Notes:** Users can remove notes they no longer need.

**User Authentication:** Secure login and registration system to protect user data.

**Responsive Design:** Ensure the application works well on desktops, tablets, and smartphones.

**Constraints:** Limited budget and resources.

Strict timeline for project completion.

Ensuring data security and privacy.

**Assumptions:** Users have basic knowledge of using web applications.

Internet access is available for all users.

**Risks:** Potential security vulnerabilities.

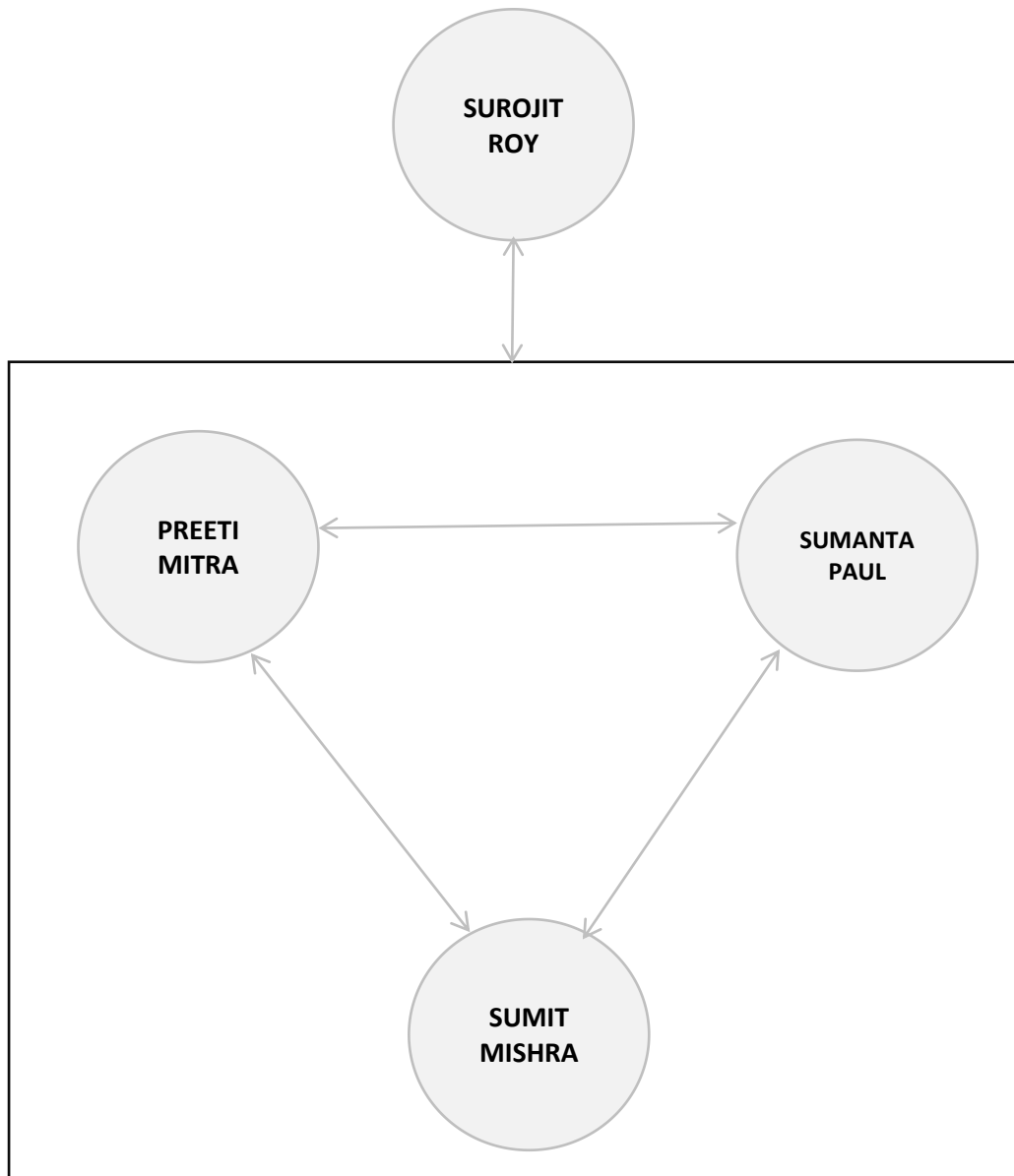
Delays in development due to unforeseen technical challenges.

This scope provides a clear outline of what your project aims to achieve and the boundaries within which it will operate.

## Chapter 2

### Concepts and Problem Analysis

#### 2.1 Team Structure



**Image 1 : Team Structure**

In this project, Democratic Team Structure is followed, as the name implies, does not enforce any formal team hierarchy. This team structure is appropriate for this project since by following this model, the group of engineers can invent better solutions than a single individual as in a chief programmer team. This team structure is suitable for projects requiring less than five or six engineers as for yours.

## 2.2 Function Point Analysis for NoteNest

### 1. Determine the Number of Components

From the **Level 0 DFD**, we can identify the following components:

#### External Inputs (EI):

##### User Inputs:

- Sign Up ( Username, Email, Password, Confirm Password )
- Login ( Email, Password )
- Add Note ( Title, Description )
- Update Note
- Delete Note

**Total EI = 5**

---

#### External Outputs (EO):

##### Output to users:

- Registration Success/Failure Message
- Login Status
- Display Notes List
- Search/Filter Notes

**Total EO = 4**

---

#### External Inquiries (EQ):

##### Query Functions:

- Verify Login Credentials
- Check Existing Username/Email
- Fetch Specific Notes for User

**Total EQ = 3**

---

#### Internal Logical Files (ILF):

##### Stored Data within system.

- Users Table ( Registration Data )
- Notes Table ( Per User Notes )

**Total ILF = 2**

---

#### External Interface Files (EIF):

- None mentioned in the diagram.

**Total EIF = 0**

---

## 2. Assign Complexity and Weights

### Complexity Table:

Function Type	Low	Average	High
External Inputs (EI)	3	4	6
External Outputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6
Internal Logical Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

**Table 1**

### Assigning Weights:

We estimate the complexity based on the details available.

Function Type	Count	Weight	FP
External Inputs (EI)	5	3	15
External Outputs (EO)	4	4	16
External Inquiries (EQ)	3	3	9
Internal Logical Files (ILF)	2	7	14
External Interface Files (EIF)	0	5	0

**Table 2**

## 3. Calculate Unadjusted Function Points (UFP)

**UFP = Sum of all weights**

$$\text{UFP} = 15 + 16 + 9 + 14 + 0 = 54$$

## 4. Determine Value Adjustment Factor (VAF)

The VAF is calculated based on 14 general system characteristics (GSC), such as reliability, performance, usability, etc., rated on a scale of 0–5. Assume an average rating of **3** for each GSC for this example.

$$\text{VAF} = 0.65 + (0.01 \times \text{Total GSC Rating})$$

$$\text{VAF} = 0.65 + (0.01 \times 35)$$

$$\text{VAF} = 0.65 + 0.35 = 1.0$$

## 5. Calculate Final Function Points

$$\text{FP} = \text{UFP} \times \text{VAF}$$

$$\text{FP} = 54 \times 1.0 = 54$$

## 2.3 Software Configuration Management

Changes continuously take place in a s/w project. Changes due to evolution of work products. As the project proceeds, changes due to bug found and fixed and changes due to requirements modification. SCM are the practice and procedure for administering the source codes, producing s/w development, controlling change and managing s/w.configuration. The goal of SCM followed during building the project

- 1. Configuration Control:** Implementing a controlled change process. In the project, our mentor reviewed the changes in our project.
- 2. Configuration status Accounting:** Ensuing that configuration contain all their intended parts and sound with respected to their specifying documents, including requirements and architectural specifications.
- 3. Built Management:** Managing the process and tools used for build. One of the most important steps of a s/w built is the compilations process.
- 4. Environment Management:** Managing the s/w and h/w that host the system.
- 5. Teamwork:** Team work is work perform by a team towards a common goal. Team work is a joint action is more than one person, in which persons contribute with their different skills and express his/her individual interest and opinions to the unity and efficiency to achieve the goal.



## 2.4 Quality Assurance Plan

The Quality Assurance Plan involves defining or selecting standards that should be applied to the s/w development process or s/w product. These standards may be embedded in procedures or processes that are applied during development. Tools that are embedded knowledge of quality standards may support processes.

There are two types of standard that may be established as part of QAP.

**1. Standards Product:** These are standards that apply to the s/w product being developed. They include documents such as the structure of the requirements documents which should be produced, documentation standard such as standard comments header for an object class definition and coding standard which define how a programming language should be used.

**2. Process standard:** These are standards that define which should be followed during the s/w development. They may include definition of specification, design and validation processes and a description of documents which must be generated in the course of these processes.

There is a very close relationship between product and process standard, the product standard applies to the output of the software process and, in many cases, the process standard includes specific process activities that ensure that product standard followed.

## 2.5 Risk Management

### Risk Management Plan for NoteNest

#### 1. Data Security Risks

**Unauthorized Access:** Implement strong authentication and authorization mechanisms to prevent unauthorized access to notes.

**Data Breach:** Use encryption for data at rest and in transit to protect sensitive information.

**Backup and Recovery:** Regularly back up data and have a recovery plan in place to prevent data loss.

#### 2. Operational Risks

**System Downtime:** Ensure high availability and reliability of the system through load balancing and failover strategies.

**Performance Issues:** Optimize the code and database queries to handle large volumes of notes efficiently.

#### 3. Compliance Risks

**Data Privacy Regulations:** Ensure compliance with data protection laws such as GDPR or CCPA, depending on your user base.

**User Consent:** Obtain explicit consent from users for data collection and processing.

#### 4. Development Risks

**Code Quality:** Maintain high code quality through code reviews, testing, and adherence to coding standards.

**Dependency Management:** Keep track of third-party libraries and frameworks to ensure they are up-to-date and secure.

#### 5. User Experience Risks

**Usability Issues:** Conduct user testing to identify and fix usability issues.

**Accessibility:** Ensure the website is accessible to users with disabilities by following web accessibility guidelines.

## **6. Project Management Risks**

**Scope Creep:** Clearly define project scope and manage changes through a formal change management process.

**Resource Allocation:** Ensure adequate resources (time, budget, personnel) are allocated to the project.

## **7. Legal Risks**

**Intellectual Property:** Ensure that all content and code used in the project do not infringe on any intellectual property rights.

**Terms of Service:** Clearly define and communicate the terms of service and privacy policy to users.

By addressing these risks, you can help ensure the successful development and operation of your notes management system.

## Chapter 3

### Theoretical Background

#### HTML5

HTML (HyperText Markup Language) is the standard language for creating web pages. Here are the key points:

**Structure:** HTML structures web content using elements and tags.

**Elements:** Tags like <h1>, <p>, and <a> define headings, paragraphs, and links.

**Attributes:** Tags can have attributes, such as href in <a> for URLs.

**Document Type:** Starts with <!DOCTYPE html>.

**Head and Body:** Divided into <head> (meta-information) and <body> (content).

**Hyperlinks:** Created with <a> tags.

**Multimedia:** Supports images, audio, and video.

**Forms:** Collects user input with <form> tags.

#### CSS (Version 3.0)

CSS, or Cascading Style Sheets, is a language used to style and layout web pages. Here are the key points:

**Purpose:** CSS defines how HTML elements are displayed on screen, paper, or in other media.

**Separation of Content and Style:** It separates content (HTML) from presentation (CSS), making it easier to maintain and update.

**Selectors and Properties:** CSS uses selectors to target HTML elements and properties to define their styles (e.g., color, font-size).

**Cascading Rules:** Styles can cascade from multiple sources, with rules for resolving conflicts.

**Responsive Design:** CSS allows for responsive design, adapting the layout to different devices and screen sizes.

#### JavaScript (12.4.254.15)

JavaScript is a versatile programming language used to add interactivity and dynamic features to web pages. Here are the key points:

**Purpose:** Enhances web pages with interactive elements like animations, form validations, and dynamic content updates.

**Client-Side:** Primarily runs in the browser, allowing for real-time user interactions without needing to reload the page.

**Integration:** Works alongside HTML and CSS to create a complete web experience.

**Versatility:** Can be used for both front-end (client-side) and back-end (server-side) development.

**Popular Uses:** Includes creating interactive forms, displaying timely content updates, and developing games and web applications.

## PHP (Version 8.2.0)

PHP (Hypertext Preprocessor) is a widely-used, open-source scripting language designed for web development. Here are the key points:

**Server-Side:** PHP code is executed on the server, generating HTML to send to the client.

**Dynamic Content:** It can create dynamic web pages, interact with databases, and manage sessions.

**Embedded in HTML:** PHP can be embedded directly within HTML code.

**Versatile:** Supports various databases (e.g., MySQL, PostgreSQL) and can handle file operations, form data, and cookies.

**Free and Open Source:** PHP is free to download and use, with a large community for support.

## XAMPP (Version 8.2.0)

XAMPP is a free, open-source software package that allows users to develop, test, and build websites on a local server. It's a popular PHP development environment that's compatible with Windows, Linux, and Mac OS X:

XAMPP stands for Cross-Platform, Apache, MySQL, PHP, and Perl

**What it's used for:** XAMPP is used for:

Testing web applications locally before deployment

Perfecting code

Transitioning from a local test server to a live serverXAMPP is easy to install and use. It includes the Apache HTTP Server, MariaDB database, and interpreters for PHP and Perl. Users can also install add-in applications like WordPress and Joomla! using Bitnami.

## MySQL Database

MySQL is the world's most popular open source database management system. Databases are the essential data repositories for all software applications. For example, whenever someone conducts a web search, logs into an account, or completes a transaction, a database stores the information so it can be accessed in the future. MySQL excels at this task.

SQL, which stands for Structured Query Language, is a programming language that's used to retrieve, update, delete, and otherwise manipulate data in relational databases. MySQL is officially pronounced "My ess-cue-el," but "my sequel" is a common variation. As the name suggests, MySQL is a SQL-based relational database designed to store and manage structured data. In recent years, however, Oracle added additional support, including for the popular JSON data type.

# **Chapter 4**

## **Software Requirement Specifications**

### **1. Introduction**

#### **1.1 Purpose**

The purpose of this document is to provide a detailed description of the requirements for the Notes Management System. This system will allow users to add, edit, search, and delete notes efficiently.

#### **1.2 Scope**

The Notes Management System will be a web-based application that provides users with the ability to manage their notes. The system will include functionalities such as creating new notes, editing existing notes, searching through notes, and deleting notes.

#### **1.3 Definitions, Acronyms, and Abbreviations**

NMS: Notes Management System

UI: User Interface

CRUD: Create, Read, Update, Delete

#### **1.4 References**

IEEE Standard for Software Requirements Specifications

Project Management Body of Knowledge (PMBOK)

### **2. Overall Description**

#### **2.1 Product Perspective**

The NMS will be a standalone web application accessible via modern web browsers. It will be designed to be user-friendly and responsive.

#### **2.2 Product Functions**

Add Note: Users can create new notes.

Edit Note: Users can modify existing notes.

Search Note: Users can search for notes using keywords.

Delete Note: Users can remove notes from the system.

#### **2.3 User Classes and Characteristics**

Regular Users: Can add, edit, search, and delete their own notes.

Admin Users: Can manage all notes and user accounts.

#### **2.4 Operating Environment**

Web browsers: Chrome, Firefox, Safari, Edge  
Server: Apache or Nginx  
Database: MySQL or PostgreSQL

## **2.5 Design and Implementation Constraints**

The system must be developed using HTML, CSS, JavaScript, and PHP.  
The system should be responsive and accessible on both desktop and mobile devices.

## **2.6 Assumptions and Dependencies**

Users will have internet access to use the system.  
The system will rely on a stable server and database connection.

# **3. Specific Requirements**

## **3.1 Functional Requirements**

FR1: The system shall allow users to create new notes.  
FR2: The system shall allow users to edit existing notes.  
FR3: The system shall allow users to search for notes using keywords.  
FR4: The system shall allow users to delete notes.

## **3.2 Non-Functional Requirements**

Performance: The system should load pages within 2 seconds.  
Usability: The system should be easy to navigate and use.  
Security: The system should ensure that user data is protected and secure.

## **3.3 Interface Requirements**

User Interface: The UI should be intuitive and user-friendly.  
API: The system should provide a RESTful API for integration with other systems.

# **4. Software and Hardware Requirements**

## **Hardware Specifications**

### **Server Requirements**

Processor: Intel Xeon or AMD equivalent, 2.4 GHz or higher  
Memory (RAM): 4 GB minimum, 8 GB recommended  
Storage: 10 GB free space, with regular backups  
Network: High-speed internet connection (1 Gbps recommended)

### **Client Requirements**

Processor: Intel Core i3 or AMD equivalent, 1 GHz or higher  
Memory (RAM): 4 GB minimum, 8 GB recommended  
Storage: At least 500 MB free space  
Display: 1280x720 resolution minimum  
Network: Stable internet connection (minimum 512 kbps for web interaction)



## **Software Specifications**

### **Server-Side**

Operating System: Windows 10/11, Linux (Ubuntu 20.04 LTS or CentOS 7)

Web Server: Apache 2.4 or Nginx 1.18

Database: MySQL 8.0 or PostgreSQL 13

Programming Languages: PHP 7.4 or higher

Frameworks: Laravel 8 (optional, if using a PHP framework)

Other Software: OpenSSL, PHPMyAdmin (for database management)

### **Client-Side**

Operating System: Windows 10, macOS Catalina, or Linux

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Other Software: PDF reader (for viewing documentation)

Additional Tools

Version Control: Git

IDE/Code Editor: Visual Studio Code, PHPStorm, or Sublime Text

Project Management: Jira, Trello, or Asana

## Chapter 5

### Design/Solution/Methodology

#### 5.1 Data Design

The Personalized Notes Management System employs a robust data design to handle all operational and transactional data:

##### Entity-Relationship Diagram (ERD)

The ERD outlines the relationships between different entities in your system. Here are the main entities and their relationships:

##### NoteNest User

Attributes: U.No (Primary Key), Username(u), Password, Email(u), date

Relationships: A user can have multiple notes.

##### User's Note Table

Attributes: S.No (Primary Key), Title, des, tstamp

Relationships: Each note belongs to one user.

##### Relationships

One-to-Many Relationship

##### Entity-Relationship Diagram (ERD)

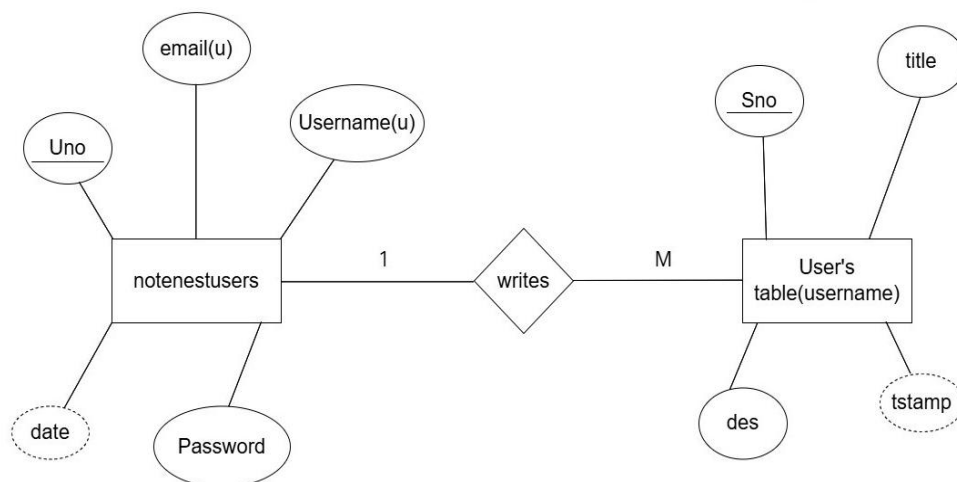


Image 2 : ER Diagram

## Data Flow Diagram (DFD)

### Level 0 (Context Diagram)

User: Interacts with the system to manage notes.

System: Provides functionalities to add, update, search, and delete notes.

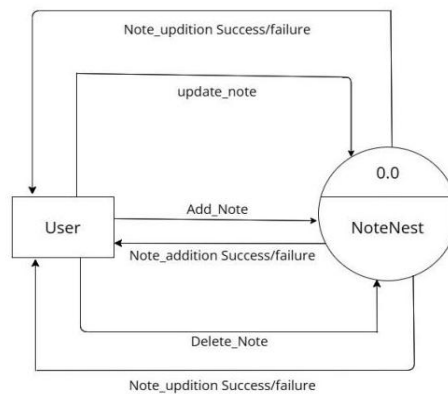


Image 3 : Level 0 DFD

### Level 1 (Detailed Diagram)

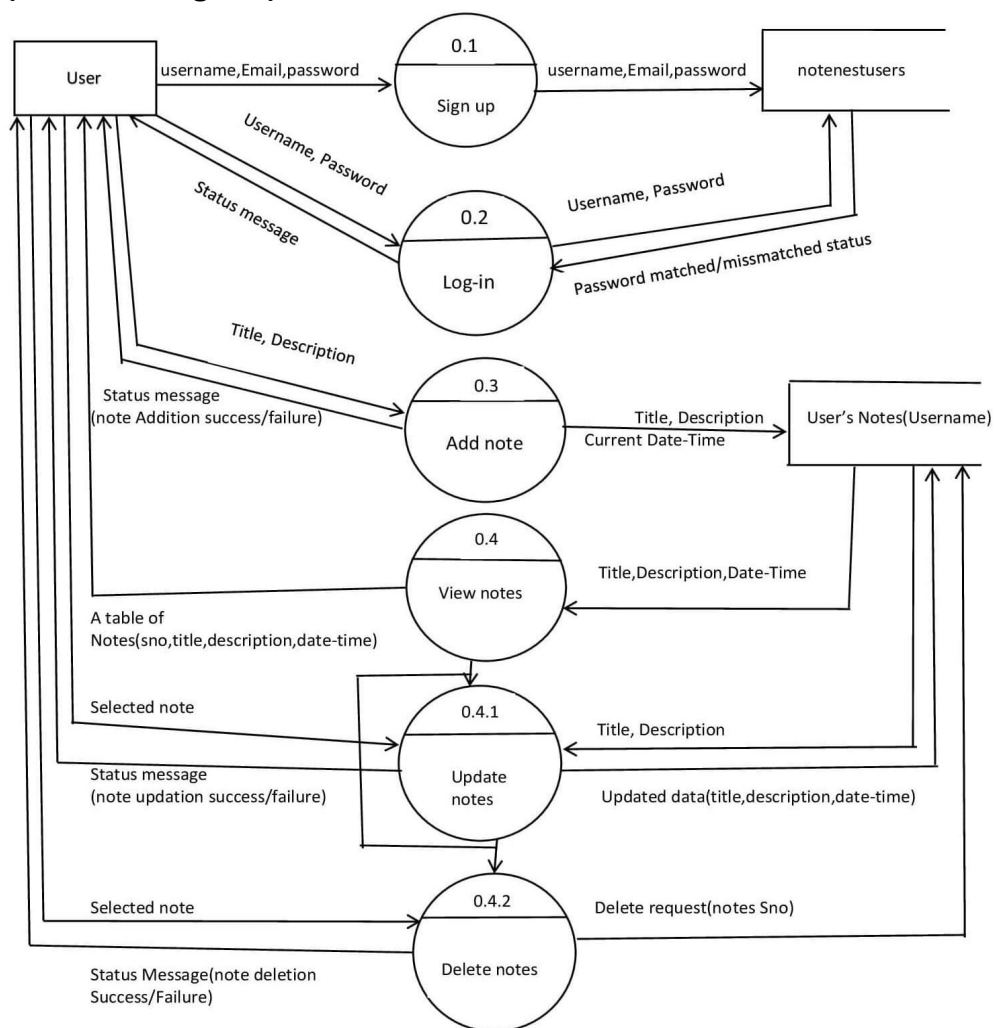


Image 4 : Level 1 DFD

## 5.2 Architectural Design

### 1. Overview

The Notes Management System will be a web-based application designed with a multi-tier architecture to ensure scalability, maintainability, and security. The architecture will consist of the following layers:

Presentation Layer  
Business Logic Layer  
Data Access Layer  
Database Layer

### 2. Layers and Components

#### 2.1 Presentation Layer

Description: This layer is responsible for the user interface and user experience. It handles user interactions and displays information.

Technologies: HTML, CSS, JavaScript, Bootstrap

Components:

User Interface (UI): Web pages for adding, editing, searching, and deleting notes.

Client-Side Scripting: JavaScript for dynamic content and AJAX for asynchronous requests.

#### 2.2 Business Logic Layer

Description: This layer contains the core functionality and business rules of the application. It processes user inputs and interacts with the Data Access Layer.

Technologies: PHP, Laravel (optional)

Components:

Controllers: Handle incoming requests, process data, and return responses.

Services: Implement business logic and rules.

Validation: Ensure data integrity and validation.

#### 2.3 Data Access Layer

Description: This layer manages the interaction between the Business Logic Layer and the Database Layer. It performs CRUD operations.

Technologies: PHP, PDO (PHP Data Objects)

Components:

Repositories: Abstract the database operations.

Data Mappers: Map database records to objects and vice versa.

## 2.4 Database Layer

**Description:** This layer is responsible for data storage and retrieval. It ensures data persistence.

**Technologies:** MySQL or PostgreSQL

**Components:**

**Database Schema:** Tables for users and notes.

**Stored Procedures:** Optional, for complex queries and operations

## 5.3 Interface Design

### 1. Home Page

**Header:** Contains the site logo, navigation links (Notes, About, Contact), and user login/signup buttons.

**Main Section:** Welcome to NoteNest with a brief description and call-to-action buttons (e.g., “Get Started”, “Learn More”).

**Footer:** Includes links to privacy policy, terms of service.

**Main Content Area:** Displays a list of notes with options to view, edit, or delete each note.

**Add Note Button:** Prominently placed to allow users to quickly add a new note.

### 2. Add/Edit Note Page

**Form:** Includes fields for the note title and content, with save and cancel buttons.

**Validation Messages:** Display error messages for required fields or invalid input.

### 3. Search Page

**Search Bar:** Allows users to enter keywords to search for notes.

**Search Results:** Displays a list of notes matching the search criteria, with options to view, edit, or delete each note.

### Design Considerations

**Responsive Design:** Ensure the interface is accessible on both desktop and mobile devices.

**User-Friendly Navigation:** Use clear and intuitive navigation menus.

**Consistent Styling:** Apply a consistent color scheme, typography, and button styles throughout the application.

## 5.4 Procedural Design

Here's a procedural design for Notes Management System, detailing the step-by-step processes for the main functionalities:

### 1. Add Note

#### Steps:

**User Action:** User clicks on the "Add Note" button.

**UI Response:** Display a form with fields for the note title and content.

**User Input:** User fills in the title and content fields.

**Form Submission:** User clicks the "Save" button.

**Validation:** The system validates the input data (e.g., checks for empty fields).

**Database Operation:** If validation passes, the system sends a request to the server to insert the new note into the database.

**Confirmation:** The system confirms the note has been added and displays a success message to the user.

### 2. Edit Note

#### Steps:

**User Action:** User selects a note to edit from the list of notes.

**UI Response:** Display the selected note in an editable form.

**User Input:** User modifies the title and/or content fields.

**Form Submission:** User clicks the "Update" button.

**Validation:** The system validates the updated data.

**Database Operation:** If validation passes, the system sends a request to the server to update the note in the database.

**Confirmation:** The system confirms the note has been updated and displays a success message to the user.

### 3. Search Note

#### Steps:

**User Action:** User enters keywords into the search bar and clicks the "Search" button.

**UI Response:** The system sends a search request to the server with the entered keywords.

**Database Operation:** The server queries the database for notes matching the keywords.

**Results Display:** The system displays the search results to the user, showing a list of matching notes.

**User Interaction:** User can view, edit, or delete notes from the search results.

#### 4. Delete Note

##### Steps:

**User Action:** User selects a note to delete from the list of notes.

**UI Response:** Display a confirmation dialog asking the user to confirm the deletion.

**User Confirmation:** User confirms the deletion.

**Database Operation:** The system sends a request to the server to delete the note from the database.

**Confirmation:** The system confirms the note has been deleted and updates the list of notes to reflect the change.

#### 5. User Authentication (Login/Signup)

##### Login Steps:

**User Action:** User enters their username and password and clicks the “Login” button.

**Form Submission:** The system sends the login credentials to the server.

**Validation:** The server validates the credentials against the database.

**Authentication:** If credentials are valid, the user is authenticated and redirected to the dashboard.

**Error Handling:** If credentials are invalid, an error message is displayed.

##### Signup Steps:

**User Action:** User clicks on the “Signup” button and fills in the registration form.

**Form Submission:** User submits the form.

**Validation:** The system validates the input data (e.g., checks for unique username and valid email).

**Database Operation:** If validation passes, the system sends a request to the server to create a new user account in the database.

**Confirmation:** The system confirms the account creation and redirects the user to the login page.

These procedural designs outline the detailed steps involved in each functionality, ensuring a clear and structured approach to implementing Notes Management System.

## **CHAPTER-6**

### **Coding Standard Followed and Assumptions**

For Notes Management System website, adhering to coding standards and making clear assumptions is crucial for maintaining code quality and ensuring smooth development. Here are some guidelines and assumptions:

#### **Coding Standards**

##### **Naming Conventions:**

Use meaningful and descriptive names for variables, functions, and classes. Follow camelCase for variables and functions, and PascalCase for classes. Constants should be in UPPER\_CASE.

##### **Code Organization:**

Structure your code into modules or components. Keep related functions and classes together. Separate business logic from presentation logic.

##### **Indentation and Spacing:**

Use consistent indentation (e.g., 2 or 4 spaces). Ensure proper spacing around operators and after commas.

##### **Commenting and Documentation:**

Write clear and concise comments explaining the purpose of code blocks. Use docstrings for functions and classes to describe their behavior and parameters. Maintain a README file with project overview and setup instructions.

##### **Error Handling:**

Implement proper error handling using try-catch blocks. Log errors for debugging purposes.

##### **Version Control:**

Use a version control system like Git. Commit changes frequently with meaningful commit messages.



**Testing:**

Write unit tests for critical functions.  
Use automated testing tools to ensure code quality.

**Assumptions****User Authentication:**

Users must be authenticated to add, edit, or delete notes.  
Assume a basic authentication mechanism (e.g., username and password).

**Data Storage:**

Notes are stored in a database (e.g., SQL or NoSQL).  
Each note has a unique identifier, title, content, and timestamp.

**Search Functionality:**

Users can search notes by keywords in the title or content.  
Assume a simple search algorithm for initial implementation.

**User Interface:**

The website has a user-friendly interface with forms for adding and editing notes.  
Assume responsive design for compatibility with various devices.

**Security:**

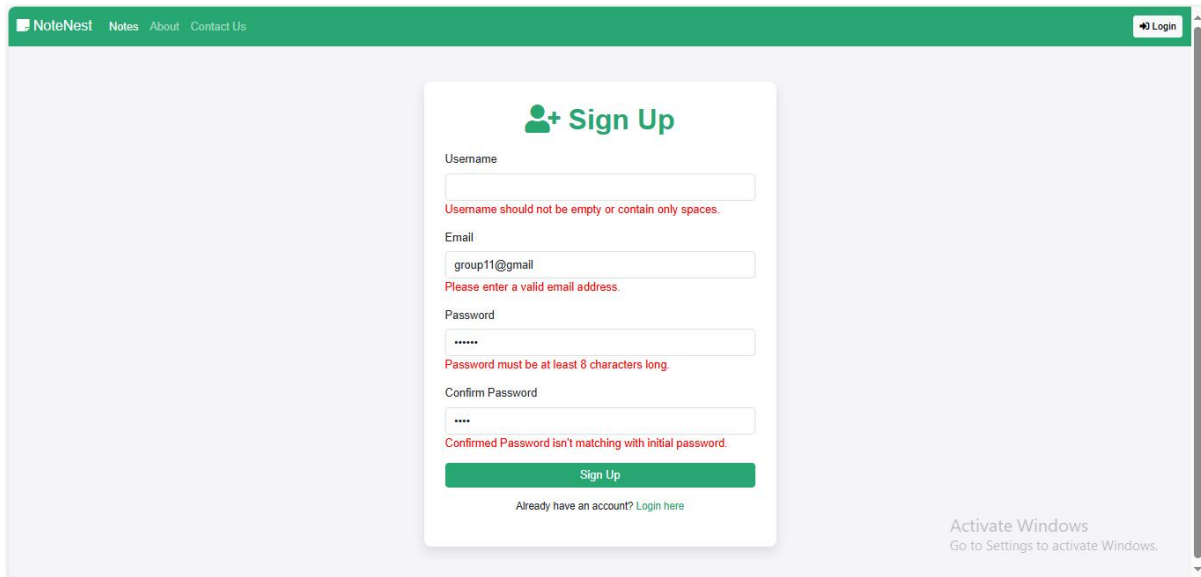
Basic security measures are in place to protect user data.  
Assume HTTPS for secure communication.

**Scalability:**

The system is designed to handle a moderate number of users and notes.  
Assume future scalability options for increased load.

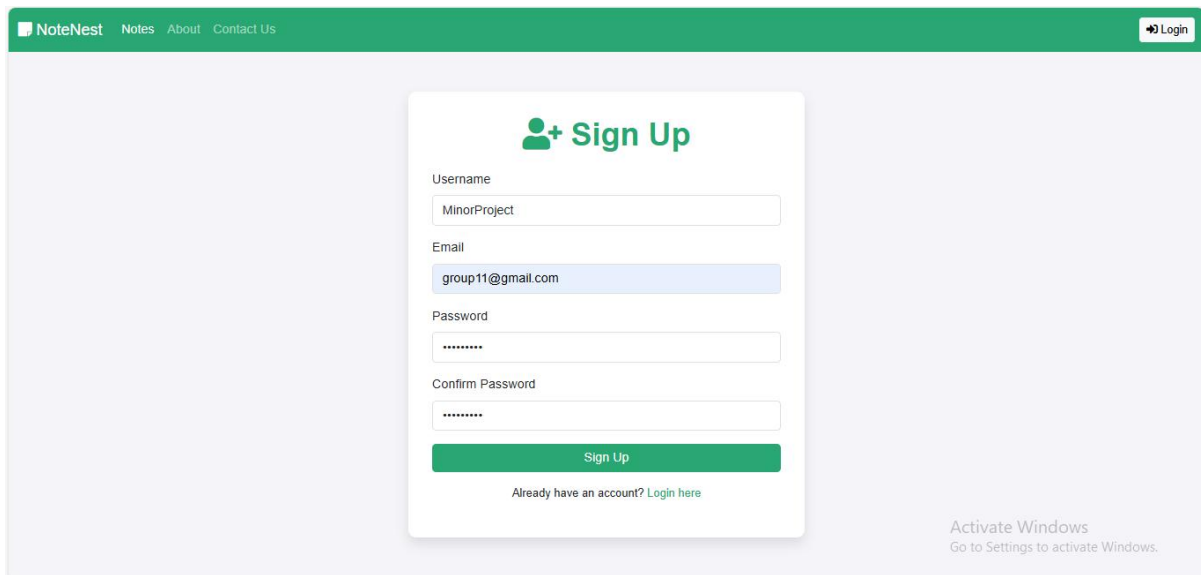
## CHAPTER-7

### Result Set Analysis



The screenshot shows a web application interface for a 'Sign Up' form. The header is green with the 'NoteNest' logo and navigation links: 'Notes', 'About', and 'Contact Us'. A 'Login' button is in the top right. The form is centered and titled 'Sign Up' with a user icon. It contains four input fields: 'Username' (empty), 'Email' (containing 'group11@gmail'), 'Password' (containing 8 dots), and 'Confirm Password' (containing 4 dots). Below each field is a red error message: 'Username should not be empty or contain only spaces.', 'Please enter a valid email address.', 'Password must be at least 8 characters long.', and 'Confirmed Password isn't matching with initial password.' respectively. A green 'Sign Up' button is at the bottom of the form, with a link 'Already have an account? Login here' below it. In the bottom right corner, there is a 'Activate Windows' watermark.

Image 5 : Signup with invalid inputs



The screenshot shows the same 'Sign Up' form as in Image 5, but with valid inputs. The 'Username' field now contains 'MinorProject', the 'Email' field contains 'group11@gmail.com', and both the 'Password' and 'Confirm Password' fields contain 8 dots. The error messages are no longer present. The green 'Sign Up' button and the 'Already have an account? Login here' link remain at the bottom of the form. The 'Activate Windows' watermark is still visible in the bottom right corner.

Image 6 : Signup with valid inputs

## CHAPTER-8

### TESTING

Testing Notes Management System is essential to ensure it functions correctly and meets user expectations. Here's a comprehensive approach to testing your system:

#### Types of Testing

##### Unit Testing:

Test individual components like adding, editing, searching, and deleting notes. Ensure each function works as expected in isolation.

##### Integration Testing:

Test the interaction between different components (e.g., database and user interface). Verify that data flows correctly between the front-end and back-end.

##### Functional Testing:

Validate that the system performs all specified functions. Test scenarios like creating a new note, editing an existing note, searching for notes, and deleting notes.

##### Performance Testing:

Assess the system's performance under various conditions. Conduct load testing to see how the system handles multiple users simultaneously.

##### Security Testing:

Ensure that user data is protected. Test for vulnerabilities like SQL injection, cross-site scripting (XSS), and data breaches.

##### Usability Testing:

Evaluate the user interface and user experience. Ensure the system is easy to navigate and use.

##### Regression Testing:

Re-run tests after changes to ensure new code doesn't break existing functionality. Use automated tests to streamline this process.

#### Test Cases

Here are some example test cases for your Notes Management System:

##### Add Note:

Input: Title, content, and optional tags.

Expected Result: Note is saved and displayed in the list of notes.

**Edit Note:**

Input: Modify the title or content of an existing note.

Expected Result: Changes are saved and reflected in the note.

**Search Note:**

Input: Keyword in the search bar.

Expected Result: Notes containing the keyword in the title or content are displayed.

**Delete Note:**

Input: Select a note and delete it.

Expected Result: Note is removed from the list and database.

**User Authentication:**

Input: Valid and invalid login credentials.

Expected Result: Access granted for valid credentials; error message for invalid credentials.

**Best Practices**

**Automate Testing:**

Use tools like Selenium for UI testing and JUnit for unit testing.

Automate repetitive tests to save time and reduce errors.

**Continuous Integration:**

Integrate testing into your CI/CD pipeline.

Run tests automatically on code commits to catch issues early.

**Test Coverage:**

Aim for high test coverage to ensure all parts of the system are tested.

Use code coverage tools to identify untested areas.

**Documentation:**

Document all test cases and results.

Maintain a test plan outlining the scope, objectives, and schedule of testing activities.

**Test Scenarios:**

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status
TC1	Add a new note	1. Navigate to the "Add Note Page". 2. Enter a title and content. 3. Click "Save".	Note is saved and displayed in the list of notes.	Note is saved and displayed correctly.	Pass
TC2	Edit an existing node.	1. Select a note from the list. 2. Click "Edit". 3. Modify the title and/or content. 4. Click "Save".	Changes are saved and reflected in the note.	Changes are saved and displayed correctly.	Pass
TC3	Search for a note	1. Enter a keyword in the search bar.	Notes containing the keyword in the title or content are displayed.	Relevant notes are displayed.	Pass
TC4	Delete a note	1. Select a note from the list.	Note is removed from the list and database.	Note is deleted successfully.	Pass
TC5	User login	1. Navigate to the login page. 2. Enter valid credentials. Click "Login".	User is authenticated and redirected to the dashboard.	User is logged in successfully.	Pass
TC6	User login with invalid credentials	3. Navigate to the login page. 4. Enter invalid credentials. 5. Click "Login".	Error message is displayed indicating invalid credentials.	Error message is displayed correctly.	Pass
TC7	Add note with missing title	1. Navigate to the "Add Note" page. 2. Leave the title field empty. 3. Enter content 4. Click "Save".	Error message is displayed indicating the title is required.	Error message is displayed correctly.	Pass

**Table 3 : Testing Table**

## **CHAPTER-9**

### **Future Scope of the Project**

The future scope of Notes Management System project can be quite extensive, offering numerous opportunities for enhancement and expansion. Here are some potential areas to consider:

#### **1. Advanced Search and Filtering**

Implement advanced search capabilities, such as full-text search, tag-based filtering, and date range filtering.

Use natural language processing (NLP) to improve search accuracy and relevance.

#### **2. Mobile Application**

Develop a mobile app for iOS and Android to provide users with access to their notes on the go.

Ensure synchronization between the web and mobile platforms.

#### **3. Integration with Other Services**

Integrate with popular productivity tools like Google Drive, Dropbox, and Evernote.

Enable users to import and export notes from/to these services.

#### **4. Enhanced Security**

Implement advanced security features such as two-factor authentication (2FA) and encryption of notes.

Regularly update security protocols to protect user data.

#### **5. AI and Machine Learning**

Use AI to provide smart suggestions, such as auto-tagging notes and recommending related notes.

Implement machine learning algorithms to analyze user behavior and improve the user experience.

#### **6. Customization and Personalization**

Allow users to customize the interface with themes and layouts.

Provide personalized recommendations based on user preferences and usage patterns.

## **7. Offline Access**

Enable offline access to notes, allowing users to view and edit notes without an internet connection.

Ensure changes are synchronized once the connection is restored.

## **8. Voice and Handwriting Recognition**

Implement voice-to-text functionality to allow users to dictate notes.

Integrate handwriting recognition for users who prefer to write notes by hand.

## **CHAPTER-10**

### **CONCLUSION**

In conclusion, Personalized Notes Management System project has the potential to be a highly functional and user-friendly tool for managing notes. By adhering to coding standards and making clear assumptions, you can ensure a solid foundation for development. Comprehensive testing, including unit testing, integration testing, and performance testing, will help maintain the system's reliability and security.

Looking ahead, there are numerous opportunities for enhancing the system, such as implementing advanced search features, adding collaboration tools, developing a mobile app, and integrating AI and machine learning. These enhancements can significantly improve the user experience and expand the system's capabilities.

By continuously iterating and incorporating user feedback, you can create a robust and versatile Notes Management System that meets the evolving needs of its users.



## **CHAPTER-11**

### **REFERENCE AND BIBLIOGRAPHY**

#### **Books and Articles:**

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.

#### **Websites:**

Mozilla Developer Network. (2024). "HTML: HyperText Markup Language." MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTML>

W3Schools. (2024). "CSS Tutorial." W3Schools. <https://www.w3schools.com/css/>

#### **Software and Tools:**

Visual Studio Code. (2024). Microsoft. <https://code.visualstudio.com/>

# CHAPTER-12

## APPENDIX 1

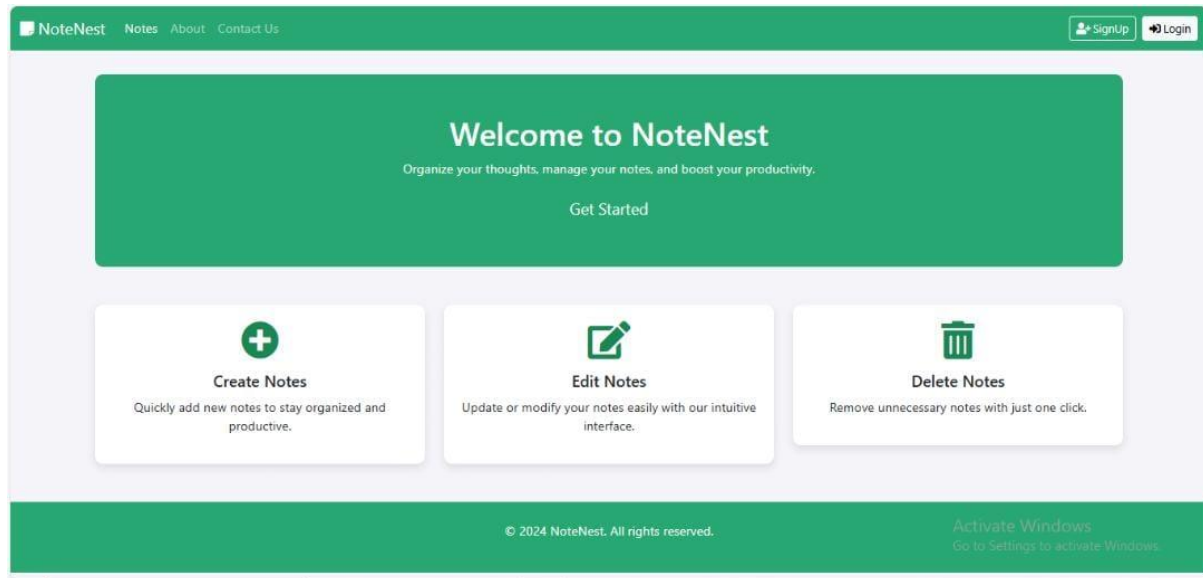


Image 7 : Home Page

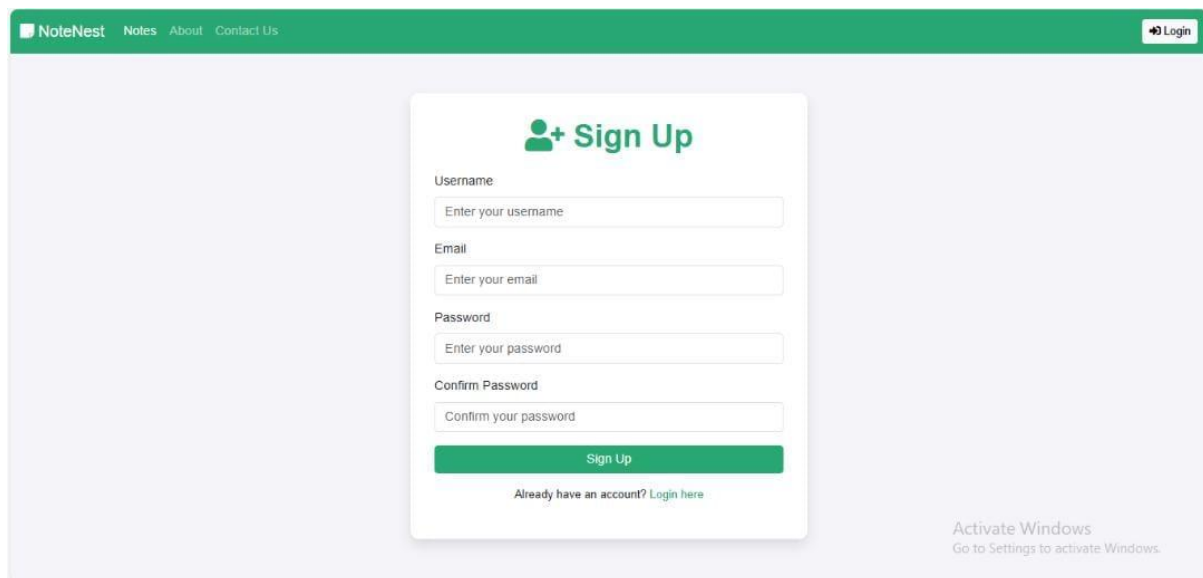
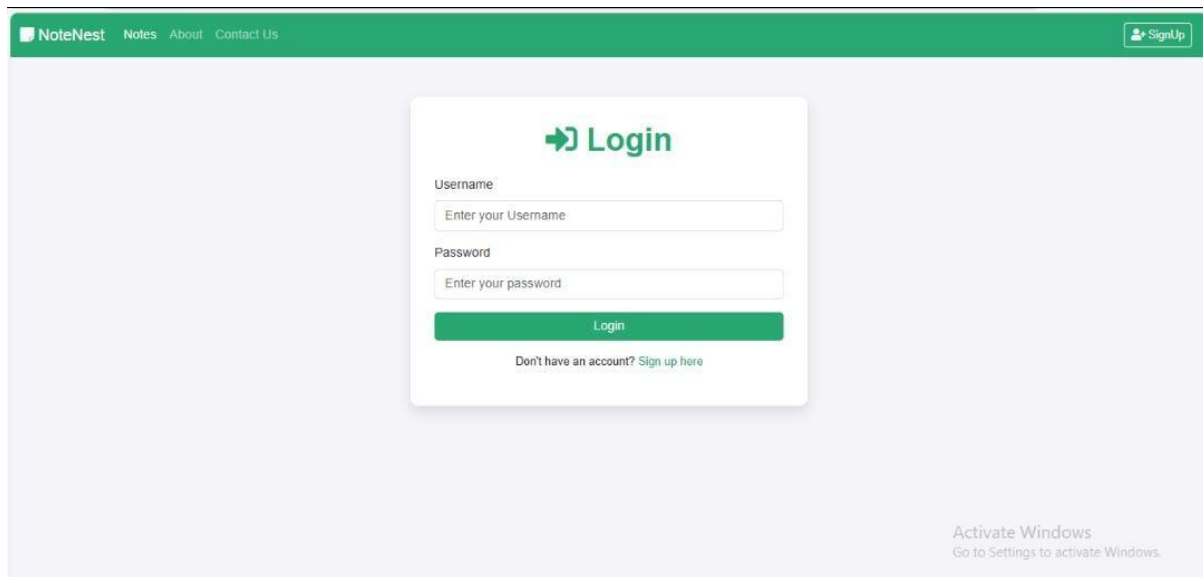
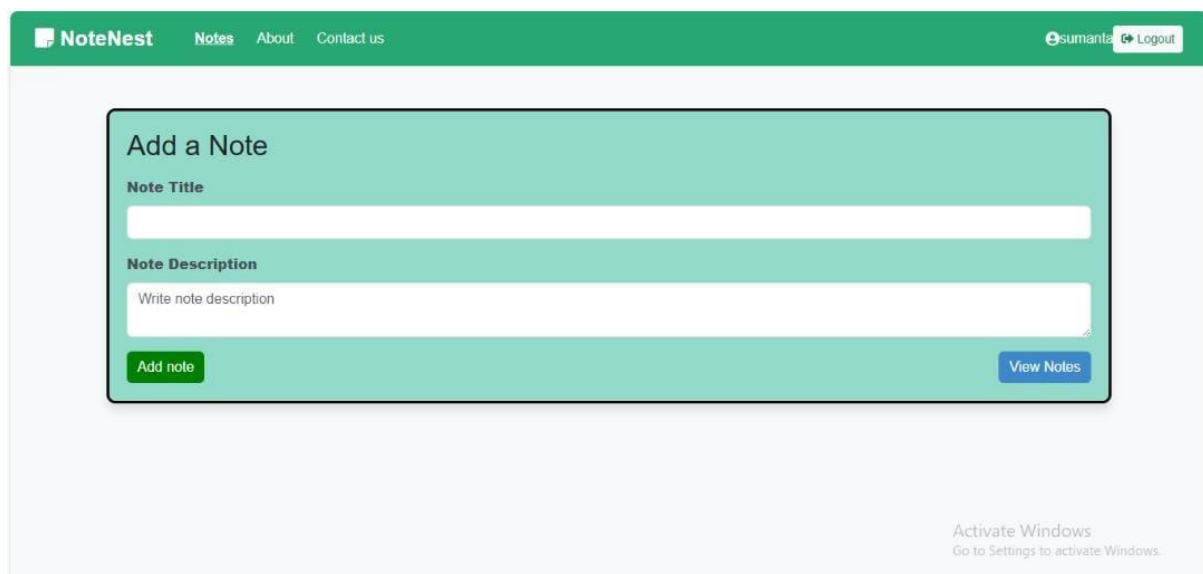


Image 8 : Signup Page



The screenshot shows the login page of the NoteNest application. At the top, there is a green navigation bar with the NoteNest logo and links for 'Notes', 'About', and 'Contact US'. A 'Sign Up' button is located in the top right corner. The main content area features a white login card with a green arrow icon and the word 'Login'. Below this, there are input fields for 'Username' (with placeholder text 'Enter your Username') and 'Password' (with placeholder text 'Enter your password'). A green 'Login' button is positioned below the password field. At the bottom of the card, there is a link that says 'Don't have an account? Sign up here'. In the bottom right corner of the page, there is a message: 'Activate Windows. Go to Settings to activate Windows.'

Image 9 : Login Page



The screenshot shows the 'Add a Note' page of the NoteNest application. The top navigation bar is green, containing the NoteNest logo, links for 'Notes', 'About', and 'Contact us', and user information 'sumanta' with a 'Logout' button. The main content area is a light blue box titled 'Add a Note'. Inside this box, there is a 'Note Title' section with a text input field, followed by a 'Note Description' section with a larger text area and placeholder text 'Write note description'. At the bottom left of the box is a green 'Add note' button, and at the bottom right is a blue 'View Notes' button. In the bottom right corner of the page, there is a message: 'Activate Windows. Go to Settings to activate Windows.'

Image 10 : Add Note

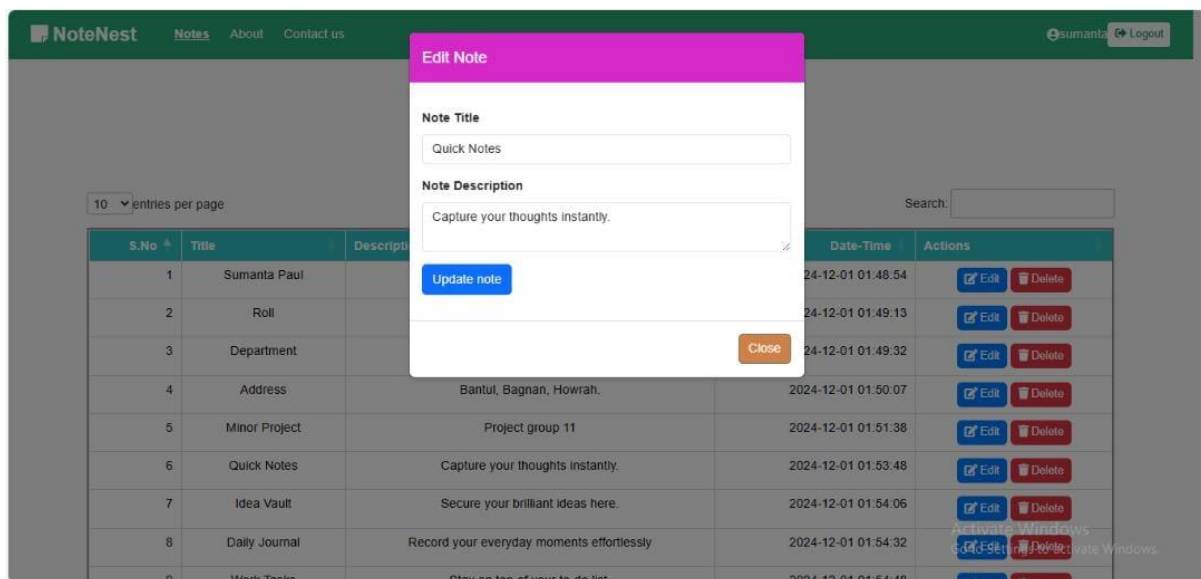


Image 11 : Edit Note

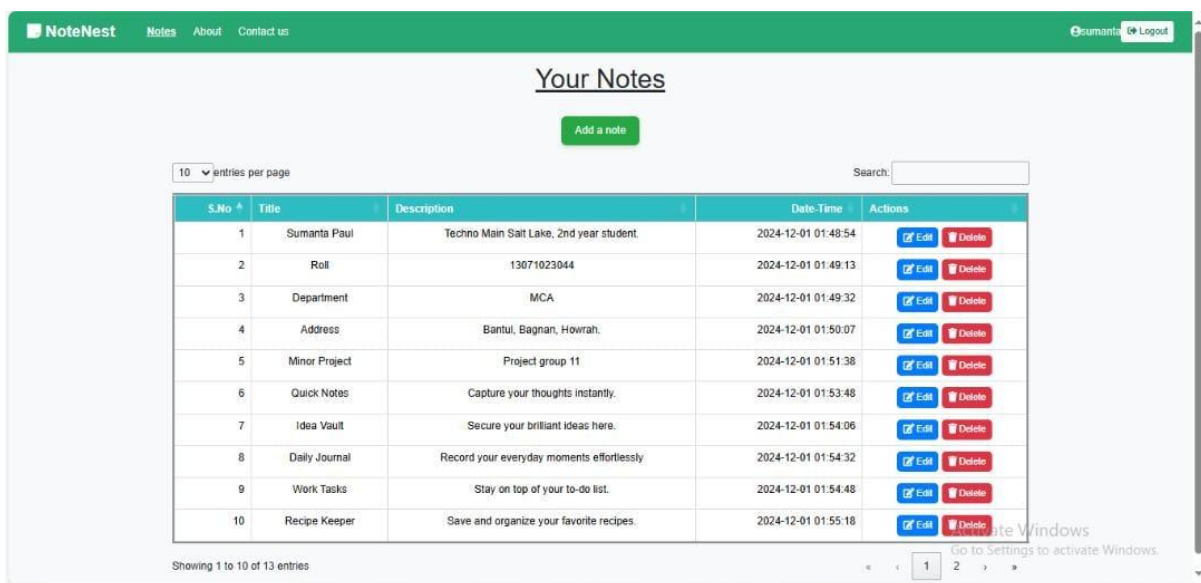


Image 12 : List of Notes

## CHAPTER-13

### APPENDIX 2

#### 1. addNote.php:

```
<?php
include("dbConnect.php");
session_start();
$table = $_SESSION["username"];
// echo $table;

if (empty($_POST["title"])) {
    $error = "<div class='alert alert-warning alert-dismissible fade show' role='alert'
id='alerts'>
        <strong>Warning!!</strong> Please add Note title..
        <button type='button' class='btn-close' data-bs-dismiss='alert' aria-
label='Close'></button>
    </div>";
    echo $error;
} else {
    // Sanitize inputs
    $title = mysqli_real_escape_string($con, $_POST["title"]);
    $desc = mysqli_real_escape_string($con, $_POST["desc"]);

    // Insert query
    $sql = "INSERT INTO $table (title, des, tstamp)
        VALUES ('$title', '$desc', current_timestamp());";
    $result = mysqli_query($con, $sql);

    // Check for success
    if ($result) {
        echo "<div class='alert alert-success alert-dismissible fade show' role='alert'
id='alerts'>
            <strong>Success!!!</strong> Your Note has been inserted sucessfully.
            <button type='button' class='btn-close' data-bs-dismiss='alert' aria-
label='Close'></button>
        </div>";
    } else {
        echo "Error: " . mysqli_error($con);
    }
}
?
```

## 2. UpdateNote.php

```
<?php
include("dbConnect.php");
session_start();
$table=$_SESSION["username"];

if (empty($_POST["titleEdit"])) {
    $error = "<div class='alert alert-warning alert-dismissible fade show' role='alert'>
        <strong>Warning!!</strong> Note title is empty..
        <button type='button' class='btn-close' data-bs-dismiss='alert' aria-
label='Close'></button>
        </div>";
    echo $error;
} else {
    // Sanitize inputs
    $title = mysqli_real_escape_string($con, $_POST["titleEdit"]);
    $desc = mysqli_real_escape_string($con, $_POST["descEdit"]);
    $id = mysqli_real_escape_string($con, $_POST["snoEdit"]);
    // Insert query
    $sql = "UPDATE $table SET title = '$title', des = '$desc' , tstamp= current_timestamp()
WHERE sno = $id";
    $result = mysqli_query($con, $sql);

    // Check for success
    if ($result) {
        echo "<div class='alert alert-success alert-dismissible fade show' role='alert'>
            <strong>Success!!!</strong> Your Note has been updated sucessfully.
            <button type='button' class='btn-close' data-bs-dismiss='alert' aria-
label='Close'></button>
            </div>";
    } else {
        echo "<div class='alert alert-success alert-dismissible fade show' role='alert'>
            <strong>Error!!!</strong> Note cannot be inserted!!.
            <button type='button' class='btn-close' data-bs-dismiss='alert' aria-
label='Close'></button>
            </div>";
    }
}
?>
```

### 3. DeleteNote.php:

```
<?php
include("dbConnect.php");
session_start();
$table=$_SESSION["username"];

if (isset($_POST["delete"])) {
    $sno = mysqli_real_escape_string($con, $_POST["delete"]);

    $sql = "DELETE FROM $table WHERE sno = $sno";
    $result = mysqli_query($con, $sql);

    if ($result) {
        echo "success";
    } else {
        http_response_code(500); // Send error code for AJAX
        echo "Error: " . mysqli_error($con);
    }
} else {
    http_response_code(400); // Invalid request
    echo "Invalid request";
}
?>
```