Project Title	Notes Keeping App
Technologies	MERN
Domain	Education
Project Level	Medium
Organization	iNeuron Intelligence Private Limited

Table

Table of Contents

1. Problem Statement:

Design a Notes Keeping Application using the MERN Stack

1.1. Overview of MERN Stack and Notes Keeping App

With the MERN Stack, a JavaScript stack, full-stack online applications may be deployed more rapidly and simply. MongoDB, Express, React, and Node.js are the four technologies that make up the MERN Stack. The development process is supposed to be streamlined and made simpler.

Each of these four powerful technologies offers developers an end-to-end environment in which to work and contributes significantly to the development of web apps.

A user can enter all of his or her notes, work, and logs in a note-keeping web app. In simple words, the note keeping app can be described as a web-based notes taking web app where a user can store all of his significant notes and retrieve them from any location at any time. You might also think of this as a to-do app where the user keeps track of all his notes.

Note: One of the most popular notes app is **Google Keep** to which you can examine to create this project.

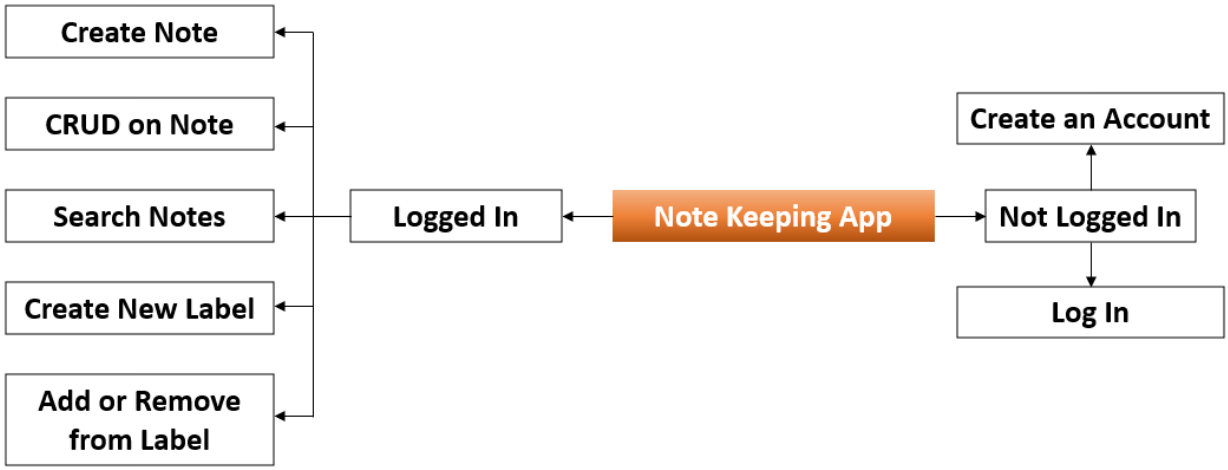
1.1.1 The Notes Keeping App should have the following properties.

These are the features that we can expect to find in your note app-

a. The user should be able to register or log in upon first seeing the website (if already registered).
b. The create note section should be available to logged-in users, from which they can make their notes and add them to the dashboard.

c. The user should have real-time access to the created note in order to conduct CRUD (Create Read Update and Delete) operations.
d. To help the user better understand when the note was written, the creation and modification dates should be given above the note.
e. Every time a user adds a note, a new data-filled card is generated and displayed on the dashboard.
f. The card background and font color should always be modifiable by the user.
g. In order for the user to look for his note using the label that has been provided to it, a search feature is necessary.

Low Level user flow



1.2. Project Objective

• **Note Keeping App:** Everyone is currently extremely busy, which causes them to forget a lot of things. As a result, they need a system where they can record all of their critical notes so they can check them at any moment with a single click.

• **User-friendliness:** Everyone should be able to utilize an app effortlessly, thus we'll need one with an engaging and appealing user interface.

1.3. Scope of The Project

1. Everyone overlooks crucial items in this hectic world, but adopting a note-taking software can help. Many users will benefit from this app by having a complete list of their chores and stuff in one location.

2. The user will feel at ease interacting and using it because to the straightforward but interesting user interface.

1.4. Functional and Non-Functional Requirements: -

1.4.1. Functional Requirements

1. **User Registration:** Users must be able to register for the app using their email address, username, and password. After accessing the app, users must be able to self-register or log in if they already have an account.

2. **CRUD Operation:** After logging in, the user should be allowed to do CRUD operations in his note.

3. **Label:** The label feature allows users to categories newly made notes by dividing them into various groups.

4. **Dashboard:** All of the user's notes will be visible there.

5. **Search:** The software should allow the user to search through their notes.

1.4.2. Non-Functional Requirements

1. **Privacy:** Only users that are logged in can access their dashboard and carry out CRUD operations on them. To stop the misuse of user data, a reliable authentication mechanism is needed.

2. **Robustness:** To avoid losing user data in the event that the user's system crashes, a backup of the user's profile and note data should be kept on a remote server.

3. **Performance:** The app must be lightweight in order to load quickly.

1.5. Use Case Table

Authentication System	Register, Login, Logout	User
Search Note	Search Note	User
User Dashboard	Display Users Notes	User

Table 1. Use Case

2. Project Evaluation Metrics:

2.1. Code:

- You are supposed to write code in a modular fashion
- Safe: It can be used without causing harm.
- Testable: It can be tested at the code level.
- Maintainable: It can be maintained, even as your codebase grows.

- Portable: It works the same in every environment (operating system).
- You have to maintain your code on GitHub.
- You have to keep your GitHub repo public so that anyone can check your code.
- Proper readme file you have to maintain for any project development.
- You should include the basic workflow and execution of the entire project in the readme file on GitHub.
- Follow the coding standards.

2.2. Database:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

2.3. API Details or User Interface:

You have to expose your complete solution as an API or try to create a user interface for your model testing. Anything will be fine for us.

2.4. Deployment:

Implementation of reverse proxy, load balancing, and security group is mandatory for deployed applications.

2.5. Solutions Design:

You have to submit complete solution design strategies in High-level Document (HLD), Low-level Document (LLD), and Wireframe documents**.**

2.6. System Architecture:

You have to submit a system architecture design in your wireframe document and architecture document.

2.7. Optimization of solutions:

Try to optimize your solution on code level, architecture level, and mention all of these things in your final submission.

Mention your test cases for your project.

3. Submission requirements:

3.1. High-level Document:

You have to create a high-level document design for your project. You can reference the HLD form below the link.

Sample link: HLD Document Link

3.2. Low-level document:

You have to create a Low-level document design for your project; you can refer to the LLD from the link below.

Sample link: LLD Document Link

3.3. Architecture:

You have to create an Architecture document design for your project; you can refer to the Architecture from the link below.

Sample link: Architecture sample link

3.4. Wireframe:

You have to create a Wireframe document design for your project; refer to the Wireframe from the link below.

Demo link: Wireframe Document Link

3.5. Project code:

You have to submit your code to the GitHub repo in your dashboard when the final submission of your project.

Demo link: Project code sample link

3.6. Detail project report:

You have to create a detailed project report and submit that document as per the given sample.

Demo link: DPR sample link

3.7. Project demo video:

You have to record a project demo video for at least 5 Minutes and submit that link as per the given demo.

3.8. The project LinkedIn a post:

You have to post your project details on LinkedIn and submit that post link in your dashboard in your respective field.