MERN TASK EXP. (Wavenet Solutions Pvt Ltd)

Description:

Build a collaborative notes application where users can create, edit, and share notes with role-based access (read/write). Implement user authentication, real-time syncing (polling or WebSocket), and permission control on both frontend and backend. The app should display shared notes, notify collaborators on updates, and maintain a clean, modular full-stack architecture.

Objective:

- Create, update, and share notes

- Collaborate in **real-time** (simulate with polling or use socket.io)

- Control access (read/write/owner) to each note

- Receive notifications when someone updates a shared note

Backend Functional Requirements:

1. Models:

- **User**: { _id, name, email, password (hashed), createdAt }

- **Note**: { _id, title, content, createdBy, collaborators: [{ userId, permission: 'read'|'write' }], lastUpdated }

2. Features:

- **User Auth**: JWT-based login and signup

- **Note APIs**:

    o Create/update/delete note

    o Share note with other users (define permission)

    o Get all notes user has access to

    o Access check logic before every write

- **Notifications**: On note update, notify collaborators (log to DB or socket emit)

3. Bonus Backend Challenges:

- Rate limit APIs per user (middleware)

- Auto-archive old notes (CRON job)

- Efficient querying: Get all notes sorted by last updated, with pagination

Frontend Functional Requirements:

1. Pages:

- **Login/Signup**

- **Dashboard**:

    - My Notes

    - Shared with Me

    - Create New Note

- **Note Editor**:

    - Live editing (polling or WebSocket)

    - Visual indicator of shared access

    - Notification popup on updates

- **Share Modal**:

    - Add users by email

    - Assign permission (read/write)

2. Frontend Challenges:

- Proper state syncing across components

- Controlled components with autosave logic (e.g., save on blur or every X seconds)

- Modal and form validation

- Conditional rendering based on permission

Deliverables:

1. Github Repository of Frontend and Backend.
2. Deployed frontend and backend services on Open Cloud servers. (Vercel, Render, etc..) [**Mandatory**]
3. A short video link demonstrating the application.[**Mandatory**]

Submission:

Share your submissions at a.mitul@wavenetsolutions.in

CC: Divya@wavenetsolutions.in

AI/ML (Wavenet Solutions Pvt Ltd)

Description:

Build a Generative AI (GenAI)-enabled chatbot, the scheme discovery process can be simplified, reducing manual effort in deciphering eligibility criteria and streamlining the application journey.

Functional Requirement:

1. Model Training:
   a. **Dataset Preparation:** Gather and curate relevant datasets, including MSME scheme documents, policy guidelines, FAQs from government sources (both central and state levels).
   b. **Fine-tuning a Large Language Model (LLM):** Select and fine-tune an appropriate pre-trained LLM (Generative AI model) using the prepared MSME scheme datasets.
   c. **Prompt Engineering:** Develop effective prompts to guide the LLM to understand user queries accurately and generate informative and helpful responses.
2. **RAG-Based Architecture (Retrieval-Augmented Generation)**
   a. **Data Ingestion & Indexing:** Scheme documents split into smaller text chunks and embedded in a vector database.
   b. **Context Retrieval:** Relevant chunks fetched via semantic search when the user queries.
   c. **LLM Integration:** AI model uses retrieved chunks as context for generating factual, personalized responses.
   d. **Feedback Loop:** User feedback refines embeddings, improving search relevance and future responses.

Dataset for the Chat-bot (https://we.tl/t-GLtJRoUoYq)

Deliverables:

1. Github Repository of Frontend and Backend.
2. Deployed frontend and backend services on Open Cloud servers. (Vercel, Render, etc..) [**Mandatory**]
3. A short Video link demonstrating the application. [**Mandatory**]

Submission:

Share your submissions at a.mitul@wavenetsolutions.in

CC: Divya@wavenetsolutions.in