DentalBee

Full Stack Engineer Offline Task

Objective

Build an end-to-end feature for a note-taking application with integrated audio recording functionality. The goal is to demonstrate your skills in full-stack development, focusing on user authentication, daily note management, in-app audio recording, and testing.

Framework Preference: You are free to use any frameworks you are comfortable with, but we prefer **Django REST Framework** for the backend and **ReactJS** for the frontend.

User Story 1: User Authentication & Daily Notes Management

As a User,

I want to **register, log in, and create daily notes**,

so that I can **manage my daily thoughts securely**.

Requirements:

1. **User Authentication**:
    a. Implement a secure registration and login system.
    b. Use JWT or session-based authentication.
    c. Only authenticated users should have access to create, view, update, or delete notes.
2. **Daily Notes**:
    a. Users should be able to create and manage daily notes.
    b. Each note should include:
        i. **Title** (Text)
        ii. **Description** (Text)
    c. Users should be able to:
        i. **Create** a new note.
        ii. **Edit** an existing note.
        iii. **Delete** a note.
        iv. **View** a list of their notes.

Acceptance Criteria:

- Users can register and log in securely.
- Authenticated users can manage (create, update, delete, view) their notes.
- Backend APIs should be properly secured.
- A basic frontend UI should be provided for managing notes.

DentalBee

User Story 2: In-App Audio Recording & Storage

As a User,

I want to **record audio notes directly in the app**,

so that I can **easily add voice recordings to my daily notes**.

Requirements:

1. **In-App Audio Recording**:
   a. Implement an in-app audio recording feature allowing users to record directly in the app.
   b. The recording should be linked to a specific note.
   c. Do not allow users to upload pre-recorded files (like MP3).
2. **Storage**:
   a. Store audio recordings securely on the server.
   b. Each recording should be tied to the corresponding note.
   c. Ensure that audio files are accessible only to authenticated users.

Acceptance Criteria:

- Users can record audio directly from the app.
- Each recording is stored and linked to its respective note.
- Audio files are securely stored and only accessible by authenticated users.
- The frontend UI should provide a clear interface for recording, saving, and playing audio.

Additional Task Requirements

1. **Automated Testing**:
   a. Write an automated test for at least one aspect of the user stories (e.g., creating a note with a recording).
   b. Use appropriate testing frameworks (e.g., PyTest/Django Test Framework for backend and Jest/React Testing Library for frontend).
2. **Codebase**:
   a. Frontend and backend code should be integrated and managed in a **single GitHub repository**.
   b. Ensure the GitHub repository is **public** and properly organized.

3. **Documentation**:
    a. Provide a detailed **README** file:
        i. Explain any assumptions you made during development.
        ii. Include an overview of the technical design and architecture.
        iii. Provide instructions for running and testing the application.
4. **Dockerization** *(Bonus)*:
    a. Dockerize the application (backend and frontend).
    b. Provide a simple setup method using **Docker Compose** or similar.
    c. Include testing setup in Docker for easy execution.

Submission Instructions

- Host the code in a **public GitHub repository**.
- Send the GitHub repository link in a reply to the provided email.
- Ensure the code is well-structured, modular, and follows best practices.

Evaluation Criteria

1. **Code Quality**: Clean, maintainable code with best practices in both backend and frontend.
2. **Feature Completion**: Successful implementation of all core features, especially the in-app recording.
3. **Testing**: Quality of automated tests and coverage.
4. **Documentation**: Clarity, detail, and completeness of the README.
5. **Dockerization** *(Bonus)*: Ease of setup and testing via Docker.

Technical Guidelines & Suggestions

- **Backend**: If using Django REST Framework, ensure RESTful API design. Feel free to use another backend framework if you prefer.
- **Frontend**: If using ReactJS, focus on creating a clean and intuitive UI.
- **In-App Audio Recording**: Consider using browser APIs (like **Web Audio API**) or libraries (like **React-Mic**).
- **Database**: Use a relational database of your choice (e.g., PostgreSQL, MySQL).
- **Authentication**: Implement a secure and scalable authentication mechanism (JWT is recommended).

Good luck! We look forward to reviewing your submission.