Assignment: Full Stack Developer

Instructions: This assignment consists of a single question with three key tasks. Follow all guidelines and adhere to coding standards.

Deadline: To be announced via email.

Submission: Upload all relevant files to GitHub and share the repository link via email before the deadline.

Comments: Include comments in your code wherever necessary.

Citation: Properly cite any external sources used.

Objective:
This assignment will evaluate your skills in full stack development, API integration, authentication, and database management. The task is to create a chatbot using the Gemini API, implement user authentication, and store chat responses in a database.

Assignment Overview:
Task 1: Set Up Authentication
Implement a user authentication system for the chatbot application.

- Next.js and Supabase.
- Allow users to register, log in, and log out securely (e.g., using JWT, OAuth, or session-based authentication).
- Ensure only authenticated users can access the chatbot.

Task 2: Integrate Gemini API for Chatbot
Develop a chatbot in Next.js that integrates with the Gemini API and allows users to upload and chat with a PDF document.

- Set up the Gemini API (refer to the official Gemini API documentation for setup and usage).

PDF Upload & Parsing:

- Allow users to upload a PDF document.
- Use a library like `pdf-parse` or `pdfjs-dist` (on the server) to extract text from the PDF.
-
- Create a frontend interface (e.g., using React, Next.js) for users to interact with the chatbot.

- Ensure the chatbot can process user inputs and return meaningful responses using the Gemini API.

Task 3: Store Responses in a Database
Store all chatbot interactions (user queries and responses) in a database.

- Use a relational database (e.g., PostgreSQL, MySQL) Create a schema to store user IDs, timestamps, user queries, and chatbot responses.
- Implement backend endpoints to save and retrieve chat history for authenticated users.

Requirements:

- **Code Documentation:** Ensure all classes, functions, and API endpoints are well-documented and adhere to coding standards (e.g., PEP 8 for Python, ESLint for JavaScript).
- **Response File:** Save a sample set of user queries and chatbot responses in a `.txt`, `.pdf`, or `.xlsx` file.
- **GitHub Repository:** Push all code (frontend, backend, database scripts) and relevant files to a GitHub repository, then share the link.

Bonus Points:
Earn additional points if you:

- Deploy the application on a platform like Heroku, Vercel, or AWS and provide the live application link in the submission email.
- Add a feature to allow users to view their chat history in the frontend interface.

Deliverables:

- **Source Code**
    - Submit frontend and backend code in appropriate files (e.g., `.js`, `.py`, `.html`, `.css`).
    - Include database schema and setup scripts (e.g., SQL files or MongoDB setup instructions).
- **Sample Queries and Chatbot Responses**
    - Save these in a `.txt`, `.pdf`, or `.xlsx` file.
- **GitHub Repository**
    - Push all code, configuration files, and relevant documentation to the repository.
- **Optional Deployment Link (Bonus)**
    - If deployed, include the live application link in your submission email.

Submission:
Email the GitHub repository link and any relevant files/links to [insert email address].
You may use resources like GitHub Copilot, ChatGPT, or other tools for assistance, but ensure all resources are properly cited.

Note: For any questions or guidance, please contact us at [insert contact email].