Task July 2024

Develop a RESTful API (using any backend framework of your choice that supports strong typing). This API will allow users to manage notes, incorporating authentication, interfacing with an external database and Sentiment Analysis API. You can assume that all notes are written in English. The server should listen on port 3500.

API Functionality:

1. **User Authentication:**
    - Implement JWT for secure user authentication.
    - Provide endpoints for user registration, login and profile retrieval.
        - POST /auth/register: Register a new user
            - body: username, password
        - POST /auth/login: Login a user
            - body: username, password
        - GET /auth/profile: Retrieve user profile
2. **Notes Management:**
    - Allow authenticated users to create and read notes.
    - Each note should include a title, body, and timestamps for creation.
    - Enable users to subscribe to other users' notes for viewing.
    - Include routes for listing users and notes, considering the user's subscriptions.
        - POST /notes: Create a new note
            - body: title, body
        - GET /notes: Retrieve notes
        - GET /notes/:id: Retrieve a specific note by ID
        - GET /users: List all users
        - POST /subscribe/:id: Subscribe to a user's notes
3. **Sentiment Analysis Integration:**
    - Integrate with the MeaningCloud Sentiment Analysis API (https://www.meaningcloud.com/developer/sentiment-analysis , its free, **no credit card needed**) to analyze the sentiment of the note's body.
    - Automatically analyze the sentiment of each newly created note and store the results in an array of meaningful values.
    - Include the latest sentiment analysis result when retrieving a user's profile.

Database:

Utilize a database system of your choice to store user and notes data. Options include MongoDB Atlas, a local database setup, or any other database that suits your needs. Ensure that relationships are properly managed with foreign keys or indexes.

If you are unable to use one of the mentioned database options, you may create your own class with a "DB" implementation using any data structure you prefer. This should mimic database operations to handle data storage and retrieval effectively.

Documentation:

Provide comprehensive API documentation (you can use tools like Swagger or Postman).

Bonus Challenge (Optional):

1. Enhance the robustness of your API by adding comprehensive unit tests
2. Implement WebSocket functionality for real-time updates when a note is added.

Evaluation Criteria:

The assignment will be evaluated based on the clarity and structure of the code, the efficiency and correctness of API usage, and the overall performance of the application. Ensure all functionalities are implemented as described without deviating into additional features unless specified. Make sure you handle edge cases (if applicable).

Development Considerations:

While assistance from AI tools like ChatGPT is permitted for conceptualizing and structuring your code, the ability to independently articulate and justify your implementation will be crucial during the technical interview. This process will assess your understanding of the technologies used and your problem-solving strategies. **Important notice** - AI tools such as ChatGPT will not be permitted during the technical interview.

Submission Instructions:

After finishing the assignment, please send an email to reisner@bigvu.tv to report that you are done. Please make sure to add a readme file to your project with a detailed step by step instructions on how to build and run your application. Please share the code through GitHub as a **private** repository and share access to: reisner@bigvu.tv, ben@bigvu.tv, tal@bigvu.tv