Take Home Challenge: Backend Engg

Challenge: Build a Real-time Quiz Application

Overview:

Develop a backend for a real-time quiz application where users can join a quiz, answer questions, and see real-time results. We prefer that you use NestJS for the backend framework, NATS for real-time event handling, and PostgreSQL for data storage. However, you are free to use Node and other other event solution such as Kafka or RMQ.

Core Features:

1. **User Registration & Authentication**: Implement user registration and login functionality.
2. **Quiz Creation**: Allow authenticated users to create quizzes with multiple-choice questions.
3. **Real-time Quiz Participation**: Users can join a live quiz and answer questions in real-time.
4. **Score Calculation & Leaderboard**: Calculate scores based on correct answers and display a real-time leaderboard using NATS or your preferred choice of event bus.

Technical Requirements:

1. **NestJS/NodeJS**: Use it for the overall application structure.
2. **Events Bus** : Implement NATS for handling real-time events (like new quiz creation, live participation, and score updates).
3. **PostgreSQL**: Utilize PostgreSQL for storing user information, quizzes, questions, and answers.

Fun Element:

Introduce a 'Streak Score' feature where users gain extra points for consecutive correct answers, adding a competitive edge to the quiz.

Test Cases

Unit Tests:

1. **User Service**:
   - Test successful user registration.
   - Test login with correct and incorrect credentials.
2. **Quiz Service**:
   - Test quiz creation with valid and invalid data.
   - Test retrieval of quizzes.

Integration Tests:

1. **Quiz Participation**:
   - Simulate user joining a quiz and submitting answers.
   - Test real-time score calculation and leaderboard updates.
2. **NATS Integration**:
   - Test NATS event publishing and subscribing for quiz events.

End-to-End Tests:

1. **User Flow**:
   - Test the entire flow of user registration, quiz creation, participation, and leaderboard display.
2. **Streak Score Feature**:
   - Test the calculation of streak scores during a quiz.

Submission Guidelines:

- Provide a GitHub repository with your code.

- Include a README with setup instructions and how to run tests.
- Document your API endpoints.

Evaluation Criteria:

- Code quality and structure.
- We are going to check how you implement DTOs, TypeORM/Sequelize, and separation of layers.
- Proper use of tech stack that has been used
- Completeness of features and test cases.
- Documentation and ease of setup.

Possible User Stories:

1. **As a user, I want to register an account, so that I can create and participate in quizzes.**
2. **As a user, I want to log in to my account, so that I can access my quizzes and leaderboard standings.**
3. **As a quiz creator, I want to create a new quiz with multiple-choice questions, so that other users can participate in it.**
4. **As a participant, I want to view available quizzes, so that I can choose one to join.**
5. **As a participant, I want to answer quiz questions in real-time, so that I can compete with others.**
6. **As a participant, I want to view my score and streak score, so that I can see how well I am doing.**
7. **As a user, I want to view the leaderboard, so that I can see where I stand among other participants.**

API Endpoints

User Management

1. **POST /users/register**: Register a new user.
    - Input: Username, password, email.
    - Output: Confirmation of registration.
2. **POST /users/login**: Log in a user.
    - Input: Username, password.
    - Output: Authentication token.

Quiz Management

3. **POST /quizzes**: Create a new quiz.
    - Input: Quiz title, array of questions with options.
    - Output: Quiz ID and confirmation.
4. **GET /quizzes**: List all available quizzes.
    - Output: List of quizzes with titles and IDs.
5. **GET /quizzes/:id**: Get details of a specific quiz.
    - Output: Quiz details including questions and options.

Quiz Participation

6. **POST /quizzes/:id/participate**: Join a quiz.
    - Input: Quiz ID.
    - Output: Confirmation of participation and quiz questions.
7. **POST /quizzes/:id/answer**: Submit an answer for a quiz question.
    - Input: Question ID, selected option.
    - Output: Confirmation of answer submission.

8. **GET /quizzes/:id/score**: Get the current score of the participant in a quiz.
    - Output: Current score and streak score.

Leaderboard

9. **GET /leaderboard**: View the leaderboard for quizzes.
    - Output: Leaderboard showing user rankings and scores.