Take-home Assignment (Backend E4)

This is a practical round where you'll have to complete a backend project in the given time frame

Some pointers to be noted:

- **Deadline: Check WhatsApp/Mail**
- **Submission:** Share a public link to Github Codebase & Postman Collection with video demonstration of your project over this mailbox
- **POC:** Devansh Agarwal

What are you going to build?

Develop a backend for a real-time quiz game where two players compete against each other. Each player is presented with the same 6 questions in sequence, and they answer these questions independently. The player who scores the highest out of the 6 questions wins the game. The game needs to handle user authentication, real-time question delivery, answer validation, scoring, and state management.

Tech Stack

- **Backend**: NestJS

- **Database**: MongoDB
- **Real-Time Communication**: WebSockets
- **Authentication**: JWT for user authentication

Requirements

1. **User Authentication**:
   - Implement endpoints for user registration and login
   - Securely hash passwords before storing them in MongoDB
2. **Game Session Setup:**
   - Create an endpoint to start a new game session and match two players
   - Once matched, initiate a game session and notify both players using socket [`game:init`]
3. **Question Management**:
   - Pre-store a set of 4 questions in MongoDB. Each question should have a question text, multiple choices, and a correct answer
4. **Real-Time Question Delivery**:
   - Use socket to send questions to each player as soon as they are ready to receive the next question [`question:send`]
5. **Answer Submission and Scoring**:
   - Allow players to submit their answers through socket [`answer:submit`]
6. **Result Calculation**:
   - At the end of the 4 questions, calculate the final scores and determine the winner
   - Send the result to both players and store the session results in MongoDB. [`game:end`]
7. **API Endpoints**:
   - `POST /register`: Registers a new user.
   - `POST /login`: Authenticates a user.
   - `POST /game/start`: Starts a new game session.

Submission Guidelines

1. Commit your code regularly with descriptive commit messages.

2. Include a README file with instructions on how to set up and run the application, including authentication setup and test user credentials.
3. Create Dockerfile to build the server
4. Create docker-compose.yaml must run server and database in it
5. Create kube.yaml -> that can deploy server on kubernetes, (you may use minikube to run it locally)
6. Create kind: Service to access server and database
7. Deploy in on a cloud provider (e.g., Render, AWS, GCP) on minikube and provide a live demo URL.
8. Share the repository URL, postman collection URL and any additional instructions via email.

Assessment Criteria

1. Adherence to clean code principles, efficient database queries, comprehensive error handling, and appropriate HTTP status codes for API responses.
2. Effective use of NestJS features and MongoDB for user authentication, game session management, and database.
3. Correct setup and functionality of login, and real-time game mechanics.
4. Robust handling of potential edge cases and errors to ensure application stability and reliability.
5. Finishing the task well before deadline.