Internship Assignment: Quiz Management System

Problem Statement:

You are tasked with developing a **Quiz Management System** named Quizo, a platform where teachers can create, manage, and view their quizzes. This system should allow teachers to log in, create quizzes, edit quizzes, and delete quizzes. You will build a **responsive web application** using **React** for the frontend and **TypeScript** for the backend.

The frontend will use **ShadCN UI components** to create a clean and modern interface, while the backend will handle quiz CRUD operations and simple user authentication. A **SQL database** (MySQL or PostgreSQL) will be used to store quiz data and user credentials.

This assignment is focused on the following features:

1. User login (static credentials, no JWT required).
2. Create, view, update, and delete quizzes.
3. Responsive design.

Requirements:

Frontend:

1. **Login Page:**
    - Create a simple login form with demo credentials (username and password).
    - Validate form inputs and submit credentials to the backend API for authentication.
    - On successful login, redirect to the dashboard where quizzes are displayed.
2. **Dashboard Page:**
    - Display a list of quizzes created by the logged-in teacher.
    - Each quiz should have buttons to **edit** or **delete**.

- Display the quiz title, description, and date created.

3. **Create/Edit Quiz Page:**
   - Form to create a new quiz or edit an existing quiz.
   - Use ShadCN form components for quiz title and description.
   - Basic form validation: ensure title and description are not empty.
4. **Responsive Design:**
   - The application should be mobile-friendly.
   - Use **ShadCN UI components** and Flexbox/Grid layout to ensure proper scaling across devices.

Backend:

1. **Authentication:**
   - Implement simple authentication with static credentials (e.g., hardcoded username/password).
   - **POST /login**: Validate credentials and return a success message upon successful login.
2. **Quiz Management API:**
   - **POST /quizzes**: Create a new quiz with a title and description.
   - **GET /quizzes**: Retrieve a list of all quizzes created by the logged-in teacher.
   - **GET /quizzes/{id}**: Retrieve the details of a specific quiz (title and description).
   - **PUT /quizzes/{id}**: Edit an existing quiz's title or description.
   - **DELETE /quizzes/{id}**: Delete a quiz from the database.

Database:

- Use **MySQL** or **PostgreSQL** for the database.
  - **Users** table to store teacher information (ID, username, password).
  - **Quizzes** table to store quiz details (ID, title, description, teacher_id).

Technical Guidelines:

- **Frontend:** Use **React** to build the application.
  - Use **ShadCN** UI components to create a modern, clean, and responsive design.
  - Use **Axios** or **Fetch API** to interact with the backend APIs.
- **Backend:**
  - Use **TypeScript** for the backend.
  - Implement basic authentication (no JWT required).
  - Handle CRUD operations for managing quizzes.
- **Database:**
  - Use **MySQL** or **PostgreSQL** for the database.

Evaluation Criteria:

- **Functionality:**
  - The application should allow a teacher to log in, view quizzes, create new quizzes, edit quizzes, and delete quizzes.
- **Code Quality:**
  - Code should be clean, modular, and well-commented.
  - Use modern features and best practices.
  - Maintain API documentation for the backend.
- **UI/UX:**
  - The UI should be user-friendly and responsive on mobile and desktop.
  - The design should be intuitive and easy to navigate.
- **Database & API Design:**
  - A simple database schema that allows proper storage and management of quizzes.
  - APIs should follow RESTful conventions (CRUD operations for quizzes).
- **Time Management:**
  - The assignment must be completed in 72 **hours**.

Deliverables:

1. **Codebase:**
    - A link to your GitHub repository containing the full project code.
2. **Instructions:**
    - A **README.md** file with:
        - Project setup instructions (how to run the project locally).
        - API documentation (endpoints and their expected request/response).
3. **Deployed Application (Optional):**
    - If possible, deploy the project using **Vercel**/ **Netlify** for the frontend and **Render**/ **Heroku** for the backend.
    - Provide a live demo URL for evaluation.

Deadline:

You have 72 **hours** to complete and submit this assignment. The deadline is [insert deadline date and time].

Must-Have:

1. Fulfill the **core objectives** (login, quiz CRUD operations).
2. Use **ShadCN UI components** for the frontend.
3. Write **clean, modular code** and follow best practices.

Good-to-Have:

1. Creating a user account for **authentication** purpose (no JWT required).
2. **Dockerize** the application (optional but a plus).

Good Luck!

We look forward to seeing your approach to this challenge. Keep your focus on the core functionality, and don't hesitate to ask if you have any questions.