

Course Management Application

Objective

The candidate is required to implement a simple application using the MERN stack **with SQLite as the database** (SQLite, Express, React, Node.js) that covers the following modules:

- Register
 - Login
 - Courses CRUD operations (Create, Read, Edit, Delete)
-

Modules to Implement

1. Register

- Endpoint to register a new user.
- Fields: Name, Email, Password, etc.
- Ensure proper validation and error handling (e.g., check if the email is unique).
- Hash the password before saving to the SQLite database.

2. Login

- Endpoint for users to log in.
- Fields: Email, Password.
- The endpoint should return a JWT token if the login is successful.
- Handle errors such as invalid credentials.

3. Courses (CRUD operations)

- **Create:** Endpoint to create a new course.
 - Fields: Course Name, Description, Instructor, etc.
 - **Read:** Endpoint to fetch all courses or a specific course by ID.
 - **Edit:** Endpoint to edit/update course details.
 - **Delete:** Endpoint to delete a course.
-

Assessment Criteria

Naming Structure

- **Consistency:** The candidate should use consistent naming conventions for variables, functions, files, models, and API endpoints.

- **API Routes:** Use plural nouns for collections (`/api/courses`) and singular for single entities (`/api/course/:id`).
- **Clarity:** Names should clearly reflect the data or functionality they represent. For example:
 - User Model: `user.js`
 - Course Model: `course.js`
 - Register Route: `auth/register.js`
 - Login Route: `auth/login.js`

Database (SQLite)

- **Schema Design:** The candidate should design clean SQLite tables for both Users and Courses with proper constraints (PRIMARY KEY, UNIQUE, NOT NULL, etc.).
 - **Data Validation:** Implement validation for the fields, such as email format, password strength, and required fields for courses.
 - **Queries:** Use parameterized queries or an ORM/query builder (e.g., Sequelize, Knex) to prevent SQL injection.
-

Expected Deliverables

1. Backend

- Functional API endpoints for Register, Login, and Courses CRUD.
- Proper handling of JWT authentication and authorization.
- Validations on inputs (using libraries like Joi or Express Validator).
- Appropriate error handling and HTTP response codes.
- SQLite database integration with well-structured tables and queries.

2. Frontend

- A simple React application to interact with the API (e.g., forms for registration, login, and course CRUD).
 - User interface should be clean and easy to navigate.
-

Submission Guidelines:

- Hosted Link (Vercel/Render/Netlify + Backend on Render)
- GitHub Repository (well-structured codebase)
- README file with:

- Tech stack
- Features
- How to run locally
- Screenshots

Assignment Submission Form : https://forms.ccbp.in/assignment-submissions_1426