## Back End Developer Challenge

### **Objective**:

Develop a web application for managing a simple to-do list with user authentication. The application should allow users to sign up, log in, create, read, update, and delete tasks. Each user should only have access to their own tasks.

### **Requirements:**

**Front-end:**

1. Framework/Library: Use React.

2. Design: Create a simple, user-friendly UI using a CSS framework like Bootstrap or Material-UI.

3. Pages/Components:

- Sign Up Page: Form for new users to create an account.

- Login Page: Form for existing users to log in.

- To-Do List Page: Display a list of tasks with options to add, edit, and delete tasks.

- Task Form Component: Used for adding and editing tasks.

4. State Management: Use React’s Context API or Redux for state management.

5. Routing: Use React Router for navigating between pages.

**Back-end:**

1. Framework: Use Node.js with Express.js.

2. Database: Use MongoDB with Mongoose for data modeling.

3. Authentication: Implement JWT-based authentication.

4. API Endpoints:

- User Registration: POST /api/register

- User Login: POST /api/login

- Get User’s Tasks: GET /api/tasks

- Add Task: POST /api/tasks

- Update Task: PUT /api/tasks/:id

- Delete Task: DELETE /api/tasks/:id

### **Functionality:**

1. User Authentication:

- Users should be able to register with a username, email, and password.

- Users should be able to log in with their email and password.

- Use JWT for maintaining user sessions.

2. Task Management:

- Logged-in users should be able to create, read, update, and delete their tasks.

- Each user should only have access to their own tasks.

### **Bonus Points:**

1. Validation: Implement form validation on both client and server sides.

2. Error Handling: Provide meaningful error messages and handle errors gracefully.

3. Responsive Design: Ensure the application is mobile-friendly.

4. Unit Testing: Write basic unit tests for both front-end components and back-end API endpoints.

5. Deployment: Deploy the application to a cloud service like Heroku, Vercel, or Netlify.

### **Deliverables:**

1. Source Code: A GitHub repository containing the source code for both the front-end and back-end.

2. README: A detailed README file with instructions on how to set up and run the application locally.

3. Demo: A live demo link if the application is deployed.

### **Evaluation Criteria:**

1. Code Quality: Clean, readable, and well-documented code.

2. Functionality: All required features work as expected.

3. UI/UX: The application should be intuitive and user-friendly.

4. Security: Proper implementation of authentication and data protection.

5. Bonus Points: Implementation of additional features and best practices.

Good luck and happy coding!