# Assignment

## Case Study: Task Management Application with User Authentication

### 1. Objective

Develop a full-stack task management application where users can register, log in, and manage their own tasks. The application must include secure authentication with JWT, perform CRUD operations on tasks, and demonstrate modern development practices in both frontend and backend.

### 2. Technology Stack

**a. Frontend**

- React (Vite/Webpack)
- TailwindCSS
- Redux Toolkit
- Axios
- React Hook Form/Formik + Zod for validation (Optional)
- Jest + React Testing Library for testing (with coverage) (Optional)

**b. Backend**

- Node.js/Express (TypeScript)/NestJs/LoopBack4/Molecular/NextJs
- Prisma ORM
- PostgreSQL/MongoDB/CouchDB/MySQL
- JWT authentication
- Jest + Supertest for testing (with coverage) (Optional)

### 3. Requirements

**Frontend:**

**a. Authentication Workflow**

- User registration with username and password
- Login that returns a JWT token on success
- Frontend stores the token securely (e.g., in localStorage)
- Logout functionality
- Protected API routes that require a valid JWT

### b. Task Management

- Authenticated users can:
    - View their list of tasks
    - Create new tasks
    - Update existing tasks
    - Delete tasks
- Each task must include:
    - title (string)
    - description (optional)
    - status (e.g., pending, completed)
- Users should only see and manage their own tasks

### c. Validation

- All forms must be validated on the frontend using Zod + React Hook Form/Formik
- Backend should also validate request bodies and handle errors gracefully

## Backend:

### a. API Endpoints :

- **Authentication**
    - POST /api/auth/register: Register a new user
    - POST /api/auth/login: Authenticate and return a JWT
- **Tasks**
    - GET /api/tasks**:** Get tasks for the logged-in user
    - POST /api/tasks: Create a new task
    - PUT /api/tasks/:id: Update an existing task
    - DELETE /api/tasks/:id: Delete a task

All task-related routes must require a valid JWT.

### b. Database:

- Use PostgreSQL/MongoDB/CouchDB/MySQL as the database.
- Define database tables for Users and Tasks with relationships.

**Testing (Optional)**

a. Include automated tests for both frontend and backend.
b. Use Jest for unit and integration testing.
c. Backend tests should cover:
    - Auth logic (register/login)
    - Task logic (CRUD operations)
    - Authorization middleware
d. Frontend tests should cover:
    - Form validation
    - UI behavior and component rendering
    - Test coverage report must be generated for both frontend and backend

**4. Deliverables**

1. A public GitHub repository containing both frontend and backend.
2. A complete README.md file with:
    - Local setup instructions
    - How to run the application
    - How to run tests and view coverage
    - API endpoint documentation
3. Extra Points:
    - Unit tests for frontend and backend

Good luck.

**Due date:** 3 days (max.) from date of receipt