Case Study: Task Management Application with Mocked API

Objective

Build a frontend-only task management application using React and TypeScript. The application must simulate user authentication and task CRUD operations. Backend responses should be mocked within the frontend using tools like MSW, json-server, or custom in-memory mocks.

Deliverables

1. Core Functionality:
    - Login page (mocked auth with JWT or session flag)
    - Dashboard with list of tasks
    - Forms to create/edit tasks (title, description, status)
    - Update/Delete task actions
    - Logout button and auth protection for dashboard

2. State Management:
    - Use **Redux Toolkit**, Zustand, or Context API
    - Manage both authentication state and tasks state

3. Mocking Layer (Required):
    - Use **Mock Service Worker (MSW)** or a similar approach
    - Simulate endpoints like:
        - POST /login – Returns a fake JWT
        - GET /tasks – Returns a list of tasks
        - POST /tasks, PUT /tasks/:id, DELETE /tasks/:id
    - Persist state across reloads using localStorage or sessionStorage

4. UI/UX and Styling:
    - Build a responsive, clean UI
    - Use **Tailwind CSS**, **AntDesign**
    - Include empty state and error views
    - Make it mobile-friendly

5. Documentation:
    - A README.md explaining:
        - How to run the project locally

        - How the mocking works
        - Project structure and libraries used

6. Deploy it on Vercel/Netlify and share the live URL

Bonus (Optional Enhancements)

- Dark mode toggle
- Use Formik for the create/edit task forms with Yup validation
- Unit tests for components

Sample Mock Endpoints to Simulate:

Endpoint	Description
POST /login	Simulates user login
GET /tasks	Fetch list of tasks
POST /tasks	Add a new task
PUT /tasks/:id	Update a task
DELETE /tasks/:id	Delete a task

You can predefine a static user (username: test, password: test123) and fake JWT.

Evaluation Criteria

Area	Details
Functionality	Auth flow and task management features fully implemented
Code Quality	Modular, reusable components, and clean architecture
State Management	Appropriate use of Redux/Context/Zustand for task and auth state
Mock API	Effective use of MSW or similar; clear simulation of backend logic
UI/UX	Clean, responsive design with good UX patterns
Documentation	Easy-to-follow README with mock API explanation
Bonus	Dark mode, filters, unit tests, deployment, Docker, etc.

Technologies Required

Area	Tech Choices
Framework	React (Vite or CRA)
Language	TypeScript
State	Redux Toolkit / Zustand / Context API
Styling	Tailwind CSS / AntDesign
Mock API	Mock Service Worker (MSW) / in-memory mocks
HTTP	Axios or Fetch API

Good luck.

Due date: : 3 days (maximum) from date of receipt