RUDRATEK

Full-Stack Assignment

Role: Full Stack Developer (React + Node.js)

Time Expectation: 8–10 hours total

Submission Window: 72 hours

AI Tools: Allowed (with disclosure)

Objective

Build a small but complete system that demonstrates your ability to:

- Design clean APIs
- Build a usable frontend on top of them
- Handle state, data, and edge cases correctly
- Explain trade-offs and decisions

This is not a design contest.

This is not about scale.

This is about correctness and structure.

Problem Statement

Build a **Project Tracking System** with:

- A backend API to manage projects
- A frontend dashboard to view and interact with them

Core Entity: Project

A project has:

- `id`
- `name` (required)
- `clientName` (required)
- `status` (`active`, `on_hold`, `completed`)
- `startDate`
- `endDate` (optional)
- `createdAt`
- `updatedAt`

You may add **one extra field** if justified and documented.

Backend Requirements (Node.js)

Required Endpoints

1. **Create Project**

   `POST /projects`

- Input validation
- `endDate` ≥ `startDate`
- Valid status only

1. **List Projects**

   `GET /projects`

   Supports:

- `status` filter
- `search` (name or clientName)
- Sorting by `createdAt` or `startDate`

1. **Get Project by ID**

   `GET /projects/:id`

2. **Update Project Status**

`PATCH /projects/:id/status`

Valid transitions only:

- `active → on_hold | completed`
- `on_hold → active | completed`
- `completed → no transitions`

1. **Delete Project**

   `DELETE /projects/:id`

- Soft delete preferred

Backend Constraints

- Node.js + Express/Fastify
- Async/await
- Clear separation of routes, controllers, and logic
- In-memory DB or simple persistence (SQLite/Postgres/Mongo)

Frontend Requirements (React / Next.js)

Dashboard View

- List projects in a table or cards
- Display key fields
- Fetch data from backend

Filters

- Status filter
- Search by project or client name
- Filters must combine correctly

Project Detail View

- Click project → detail view or side panel

- Show all project data
- Update project status using backend API

States to Handle

- Loading
- Empty list
- No results after filtering
- API errors

Technical Constraints (Frontend)

- Functional components only
- Hooks for state management
- Basic CSS or Tailwind allowed
- No heavy UI abstractions
- Clear component boundaries

AI Usage Policy (Mandatory)

AI tools are allowed.

Include a section in `README.md` answering:

- Which AI tools were used
- For which parts (backend / frontend / debugging)
- What you modified or rejected
- What you understand fully vs partially

In the review round, you must explain everything.

Submission Requirements

1. GitHub repository

2. `README.md` with:
    - Setup instructions (frontend + backend)
    - API documentation
    - Assumptions and trade-offs
3. Optional:
    - Deployed frontend
    - Postman collection
    - Basic tests

Evaluation Criteria

Area	Weight
Backend API correctness	High
Frontend data handling	High
State & edge cases	High
Code structure	High
Reasoning & explanations	Medium
Visual polish	Low

Disqualifiers

- Frontend bypassing backend logic
- Invalid state transitions
- Hardcoded hacks
- Code you cannot explain
- Over-engineering

What This Assignment Is Testing

- Can you connect frontend and backend cleanly
- Can you reason about state across layers
- Can you ship something reliable, not flashy