Full-Stack Engineer Take-Home Assignment

Overview

Build a Task Management Dashboard that demonstrates your full-stack engineering skills across both frontend and backend. This assignment is designed to be completed in **6-8 hours**, and the expectation is that you **try to stay within that range**. If you exceed the time, that is acceptable, but you should still aim for the target window.

When you start working, you must make an initial commit so we can track when you began. A recommended commit message is:

Initial commit: work started, beginning 6-8 hour window

We encourage the use of any AI tool

The Challenge

Create a web application for managing personal tasks, with a focus on user experience, clean architecture, and robust backend functionality.

Core Requirements

1. User Interface (Required)

- Task List View: Display tasks in a clean, organized layout
- Task Creation: Form to add new tasks with validation
- Task Management: Edit, delete, and mark tasks as complete
- Filtering: Filter by status (all, pending, completed)

2. Data Management (Required)

- API Integration with your backend
- Error Handling with meaningful feedback
- Loading Indicators during API calls
- Data Persistence stored in a static JSON file
- Each task must include:
  - title (required)
  - description (optional)
  - priority (High, Medium, Low)

    - due date (optional)
    - status (pending or completed)
    - created timestamp
    - updated timestamp

3. User Experience (Required)

- Intuitive navigation
- Visual feedback (loading, success, error states)
- Keyboard accessibility
- Clean and professional design

Technology Stack

- React with Typescript for the frontend
- Node.js or any language you prefer for the backend

Backend Requirements

Data Source

Use a static JSON file. No database is required.

Required API Endpoints

```
GET /api/tasks        – Get all tasks
POST /api/tasks       – Create a new task
PUT /api/tasks/:id    – Update a task
DELETE /api/tasks/:id – Delete a task
```

Data Format Example

```
{
  "tasks": [
    {
      "id": "1",
      "title": "Sample Task",
      "description": "This is a sample task",
      "priority": "medium",
      "status": "pending",
      "dueDate": "2024-01-15",
      "createdAt": "2024-01-01T10:00:00Z",
      "updatedAt": "2024-01-01T10:00:00Z"
    }
  ]
}
```

Note

The original API component example has been removed. You are responsible for designing and implementing the backend based on the requirements above.

Bonus Features (Choose 1-2)

- Real-time search
- Data visualization
- Advanced filtering
- Task categories
- Drag and drop
- Export/import
- Dark mode

Deliverables

1. Source Code

- Submit via GitHub only
- Provide read access to the email shared in a separate channel
- Use separate frontend and backend directories
- Include:
  - A clear README
  - package.json files

    - Static JSON data
- Exclude:
    - node_modules
    - build artifacts
    - sensitive information

2. Documentation

README.md

- Overview and key features
- Setup and installation instructions
- How to start both backend and frontend
- API documentation
- Scripts and commands
- Technology choices, reasoning, and trade-offs

PRODUCT_DECISIONS.md

- UX choices
- Feature prioritization
- Accessibility considerations
- Performance criteria

Time Expectations

- Aim for **6-8 hours** of focused work
- It is acceptable if the work takes slightly longer
- You must make an initial commit such as: Initial commit: work started, beginning 6-8 hour window

AI Usage Policy

AI tools are highly encouraged. However, you will be expected to understand and explain all generated code and decisions during the interview.

Getting Started Tips

1. Plan for 30 minutes
2. Build backend first
3. Implement CRUD before enhancements

4. Use any tools, including AI
5. Test endpoints early
6. Document decisions as you go
7. Validate that backend and frontend run together

Questions?

Reach out to your interviewer if anything is unclear.