Agentic SDLC Fundamental Task

You must build a simple full-stack web application using the support of an **agentic AI tool**. The focus is on both the **technical solution** and the **collaborative development process with the AI**.

Project Description

Project: "Task Manager" Web App

Create a web application that allows users to manage tasks. The app should support basic CRUD operations: Create, Read, Update, and Delete.

Tech Stack

- **Frontend**: React.js (with TypeScript preferred), optional: Material UI or TailwindCSS
- **Backend**: Spring Boot (Java 17+)
- **Database**: In-Memory (H2) or PostgreSQL (optional)
- **Build Tools**:
    - Frontend: Vite or Create React App
    - Backend: Maven or Gradle
- **Communication**: REST API with JSON

Detailed Requirements

Entity: Task

```
Task {
  id: number (auto-generated)
  title: string (required, max 100 characters)
  description: string (optional, max 500 characters)
  status: enum ["TODO", "IN_PROGRESS", "DONE"]
  dueDate: date (optional)
}
```

Functional Requirements

Frontend

- A home page that lists all tasks.
- A form to add a new task.
- Options to edit and delete existing tasks.
- Status should be changeable via dropdown.
- Optional: Sort tasks by status or due date.
- Form validations (e.g., required fields, character limits).
- Display error messages on server/API failures.

Backend

- Expose a REST API with the following endpoints:
    - GET /api/tasks – Retrieve all tasks
    - GET /api/tasks/{id} – Retrieve a task by ID
    - POST /api/tasks – Create a new task
    - PUT /api/tasks/{id} – Update a task
    - DELETE /api/tasks/{id} – Delete a task
- Validate input using Bean Validation (e.g., @NotBlank, @Size)
- Store data in an H2 database (or PostgreSQL)
- Enable CORS to allow frontend-backend communication

Agentic AI Involvement

Development Approach

1. Use the AI to help throughout the complete development phase:
    - Generating code
    - Debugging issues
    - Designing APIs
    - Refactoring
    - Prompting architecture decisions
    - Write Testcases
    - Document the project

Bonus / Optional Features

- Group tasks into categories
- Add search/filter functionality
- Deploy the frontend to Vercel or Netlify
- Deploy the backend to Render, Fly.io or similar

Timeline & Deliverables

- **Duration**: 1–2 working days
- **Deliverables**:
    - Link to git repository

Success Criteria

- The app runs successfully and supports all CRUD features.
- The use of AI tools is clearly documented.
- The code is structured and understandable.
- Technical requirements are met.
- AI usage is reflected upon critically and constructively.