Task Manager Application

Overview

Build a full-stack Task Manager application using **Next.js 14+ (App Router)**.
Your app must support CRUD operations, use API routes, and persist data in a JSON file.

Deadline: 1st Dec, 2025 (Monday)

Learning Objectives

- Understand App Router & file-based routing
- Use Server + Client Components correctly
- Build and use API routes
- Manage forms and state
- Persist data with filesystem JSON
- Style using Tailwind CSS

Project Requirements

Setup & Basic CRUD

1. Setup

- Initialize a Next.js project (`create-next-app`)
- Add Tailwind CSS (auto-included)

2. Homepage

- Display a list of tasks
- Each task must have:
    - title
    - description
    - status (pending/completed)

    - creation date

3. Task Creation

- Add a form to create new tasks

4. Update Tasks

- Toggle task status (pending ↔ completed)

5. Delete Tasks

- Implement task deletion

API Routes & Persistence

1. API Endpoints (`/app/api/tasks`)

- `GET /api/tasks` — return all tasks
- `POST /api/tasks` — create a task
- `PATCH /api/tasks/:id` — update status
- `DELETE /api/tasks/:id` — delete task

2. JSON Storage

- Store tasks in a single JSON file
- Read/write using filesystem (`fs/promises`)
- Basic error handling required

3. Frontend Integration

- Replace temporary state with API fetch calls

Finishing Touches

1. Styling

- Make the UI clean and readable
- Responsive design

2. Filtering

- Add filters:
  - All
  - Completed
  - Pending

3. Form Validation

- Validate required fields
- Show user-friendly error messages

4. Loading States

- Add loading indicators when fetching

5. README

- Include setup instructions
- Explain features

Additional Tasks (Bonus Points)

- Edit tasks (title/description)
- Add dark mode
- Add optimistic UI
- Improve error UI

Evaluation Criteria

1. Functionality

- All CRUD operations work
- API routes behave correctly
- JSON persistence works reliably

2. Next.js Usage

- Correct App Router structure
- Server vs Client component usage is appropriate
- Clean routing + API implementation

3. Code Quality

- Clear structure
- No major anti-patterns
- Reasonable component organization

4. UI Quality

- Clean layout
- Works on mobile
- Basic usability handled

5. Error Handling

- API + UI handle errors gracefully
- Validations behave correctly

6. Documentation

- README with setup instructions