Mini Project Test: To-Do List Application

Project Overview

Your task is to design and build a **To-Do List Application** with both frontend functionality and backend API integration.

The project should demonstrate your ability to design a user interface, manage application state, and interact with RESTful APIs for data persistence and updates.

You are free to design **any UI of your choice** — make it simple, clean, and user-friendly.

Core Features

1. Create To-Do Items

- Users should be able to add new to-do items.
- Each to-do should include:
    - A title
    - A description (optional)
    - A **due date** and **time**

2. View / Get To-Do Items

- Display a list of all existing to-dos retrieved from the **backend API**.
- Include indicators for:
    - **Completed tasks** (should appear visually distinct — e.g., strikethrough or faded).
    - **Overdue tasks** (tasks with a due **date/time** that has already passed).

3. Update To-Do Items

- Users should be able to edit:
    - The title
    - Description
    - Due date and time

4. Delete To-Do Items

- Users should be able to remove a to-do from the list.
- **Confirm deletion before removing the item permanently.**

5. Mark as Complete / Incomplete

- A toggle feature should allow users to mark a task as completed or undo completion.
- Completed items should appear with a **strikethrough effect** or a **different visual state**.

6. Due Date Notifications

- When the due date and time of a to-do item are reached, a **popup notification** should appear to alert the user.
- You can use:
    - The **browser's built-in alert()**, or
    - A custom modal/toast notification.

7. Overdue Items

- Any to-do with a due date/time that has already passed should:
    - Automatically appear **struck through** or **greyed out**.
    - Be clearly distinguishable as **overdue**.

Technical Requirements

Frontend

- Build the interface using vanilla JavaScript, CSS and HTML.
- The UI should be **clean, functional, and responsive**.
- Handle state management properly.

Backend

- You can use a mock API service (like **JSON Server**, **MockAPI**, **Xano** or a custom backend**).
- The backend should support these endpoints:
    - GET /todos – fetch all to-do items
    - POST /todos – create a new to-do item
    - PUT /todos/:id – update an existing to-do item
    - DELETE /todos/:id – delete a to-do item

API Integration

- All CRUD actions (Create, Read, Update, Delete) must be handled via **API calls**.
- Use **fetch()** or **Axios** for your HTTP requests.
- Handle loading states and display feedback (e.g., “Saving…”, “Deleting…”, “Updated successfully”).

Bonus (Optional Enhancements)

- Add a **search or filter** feature (e.g., show only completed or pending tasks).
- Allow **sorting by due date**.
- Add **dark mode** support.

Deliverables

- A working web app **hosted** or runnable locally.
- Source code on **GitHub** (clearly structured and **documented**).
- A short **README** file explaining:
    - How to run the app
    - Technologies used
    - Key features implemented

Evaluation Criteria

Category	Description	Weight
Functionality	App performs all required actions correctly	40%
API Integration	Proper use of RESTful API calls	20%
UI/UX Design	Clean, responsive, and intuitive interface	20%
Code Quality	Clear, maintainable, and well-documented code	10%
Creativity	Unique design, extra features, and attention to detail	10%

Deadline

You have 12 working days **(from November 10th)** to complete and submit your project.