### ****Objective:****

Develop a Task Manager application using the MERN stack (MongoDB, Express, React, Node.js) combined with MUI (Material-UI) components. The application will serve as a tool to manage tasks, offering full CRUD (Create, Read, Update, Delete) functionalities

### ****Features and Requirements:****

#### ****1. Task Management API****

Implement the following API endpoints:

* **GET /tasks:** Retrieve all tasks from the database.
* **POST /tasks:** Create a new task. The request should include a title, description, deadline, and an optional file upload (PDF).
* **DELETE /tasks/:id:** Delete an existing task by its ID.
* **PUT /tasks/:id:** Update an existing task by its ID. This includes editing the title, description, status (TODO or DONE), deadline.

#### ****2. Task Schema****

Create a Task schema in MongoDB with the following fields:

* **Title:** String, required, represents the name of the task.
* **Description:** String, required, provides details about the task.
* **Status:** Enum (values: TODO, DONE), default is TODO, represents the task's completion status.
* **Linked File:** Blob, optional, stores a file (PDF) associated with the task.
* **Created On:** Date, required, auto-generated when the task is created.
* **Deadline:** Date, required, indicates the date by which the task should be completed.

#### ****3. Task UI/UX****

Design the front-end of the application to include the following features:

* **Task List View:**
  + Display tasks in a tabular format with columns for Title, Description, Deadline, Status, and Actions.
  + Include buttons for actions such as "Mark as Done," "View File," "Edit," and "Delete."
  + Implement a download option for any associated PDF files.
* **Add Task Button:**
  + Provide a prominent "Add Task" button, which opens a modal allowing users to enter the task details (title, description, deadline) and upload an optional PDF file.
  + Since the task form may include a PDF file, submit the form as formData instead of JSON. formData allows you to append both text data (e.g., title, description, deadline) and the optional PDF file in a single request.
* **Task Status Rendering:**
  + Tasks should display as "In Progress" if they are ongoing (from the creation date until the deadline).
  + If a task is marked as "DONE" before the deadline, it should change to "Achieved" once the deadline passes.
  + Tasks not marked as "DONE" by the deadline should automatically display as "Failed."

### ****Sample Dataset:****

Create the following sample tasks for testing:

* **Title:** Study TypeScript
* **Description:** Read the documentation and make notes.
* **Linked File:** (optional) Any PDF file.
* **Created On:** 16/08/2024
* **Deadline:** 19/08/2024

### ****UI Rendering Example:****

Given the sample dataset:

* **From 16/08/2024 to 19/08/2024:** The task will be displayed as "In Progress."
* **On 20/08/2024:**
  + If the task is marked as DONE before the deadline, it will display as "Achieved."
  + If the task is not completed, it will display as "Failed."

### ****Guidance:****

* Utilize MUI components to create a responsive and visually appealing user interface.
* Follow best practices for structuring your codebase, including separating concerns between the front-end and back-end.
* Implement form validation to ensure required fields (title, description, deadline) are filled out correctly.
* Use React hooks and state management effectively to handle the task data and UI updates.

### ****Reference:****

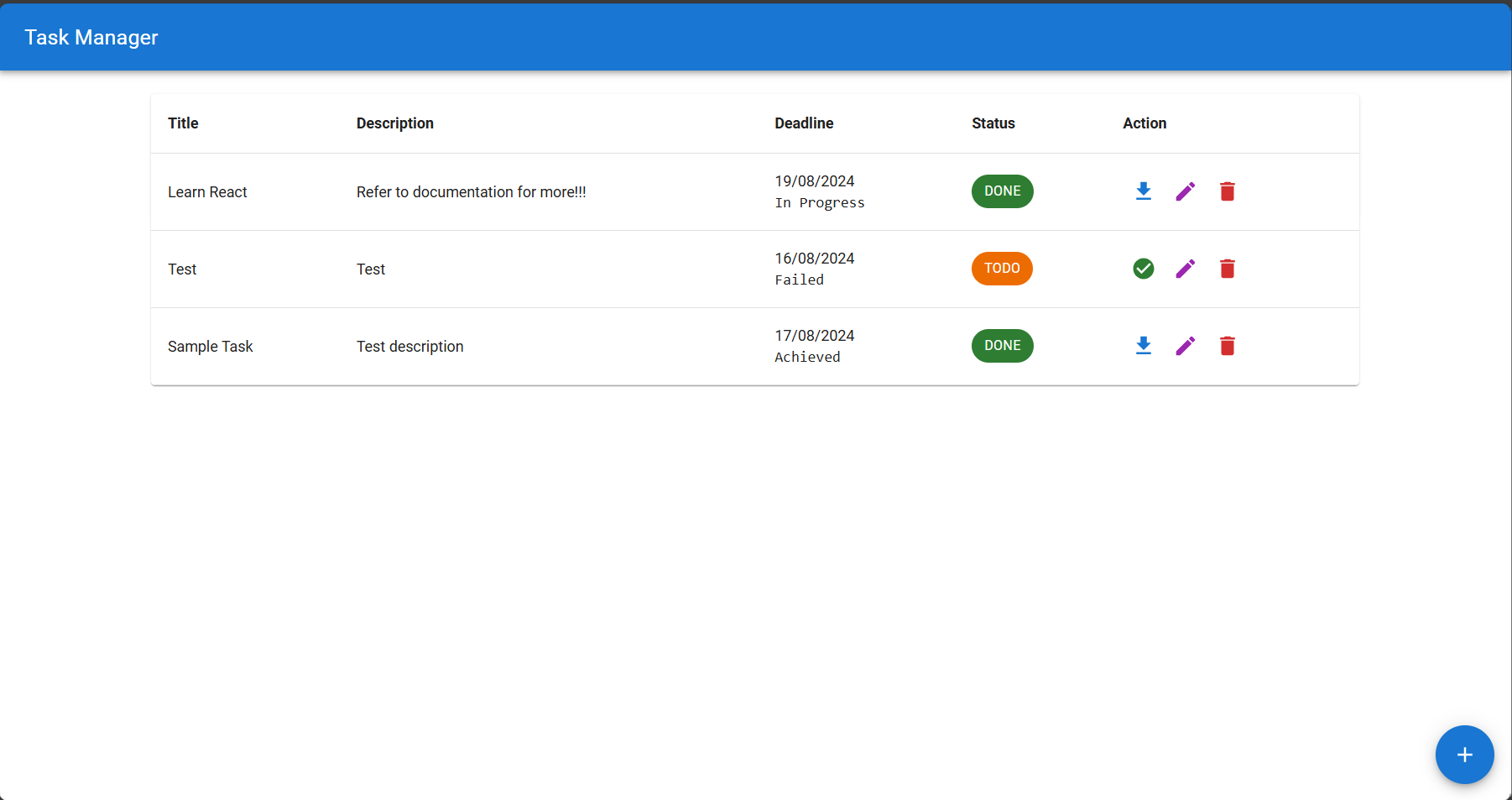
Refer to the attached images for a visual guide on how the final output of the project should look. This includes the task list view, "No tasks found" message, and the "Add Task" modal.

# UI Screenshots

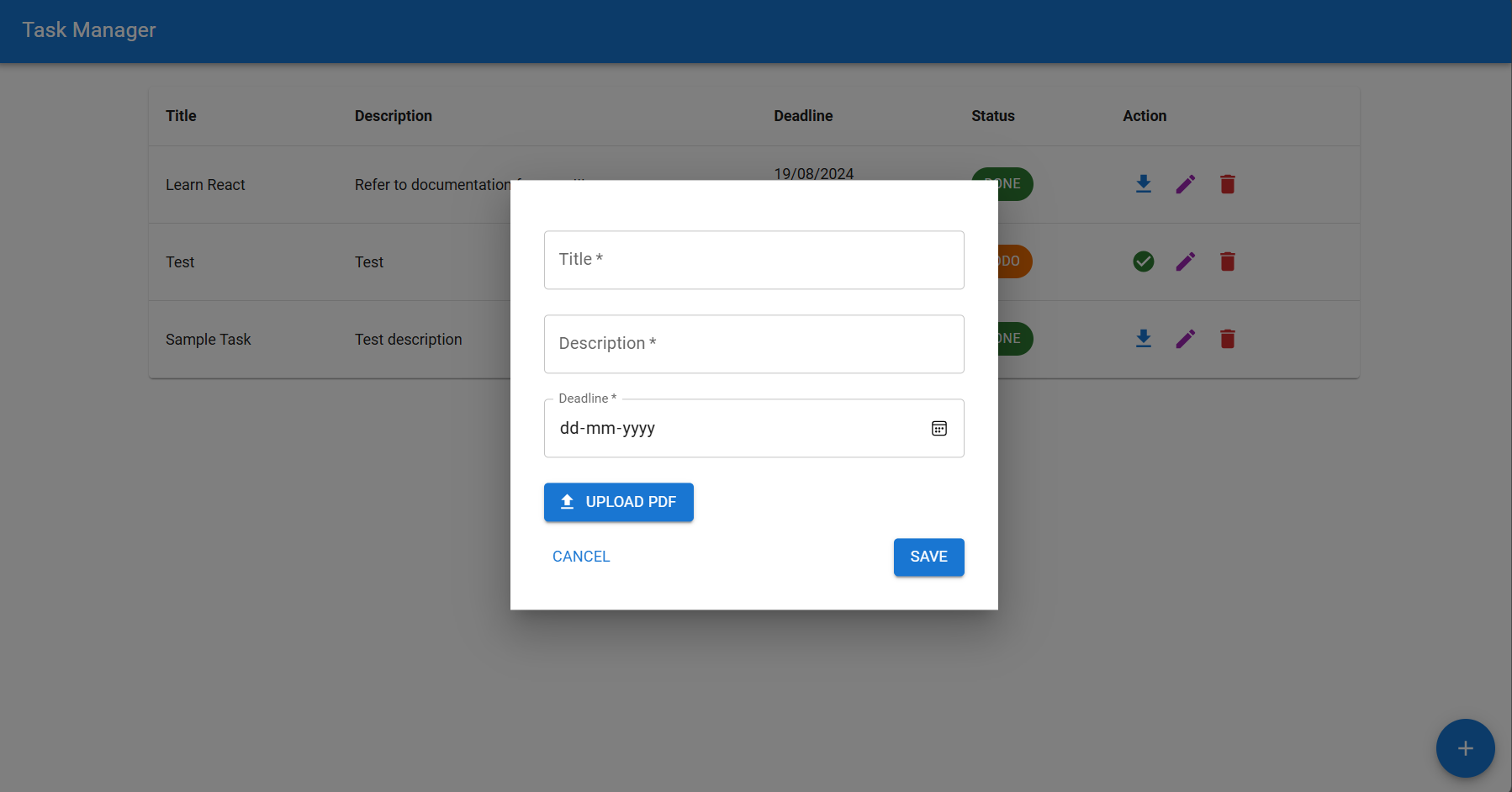
### No Data View

### 

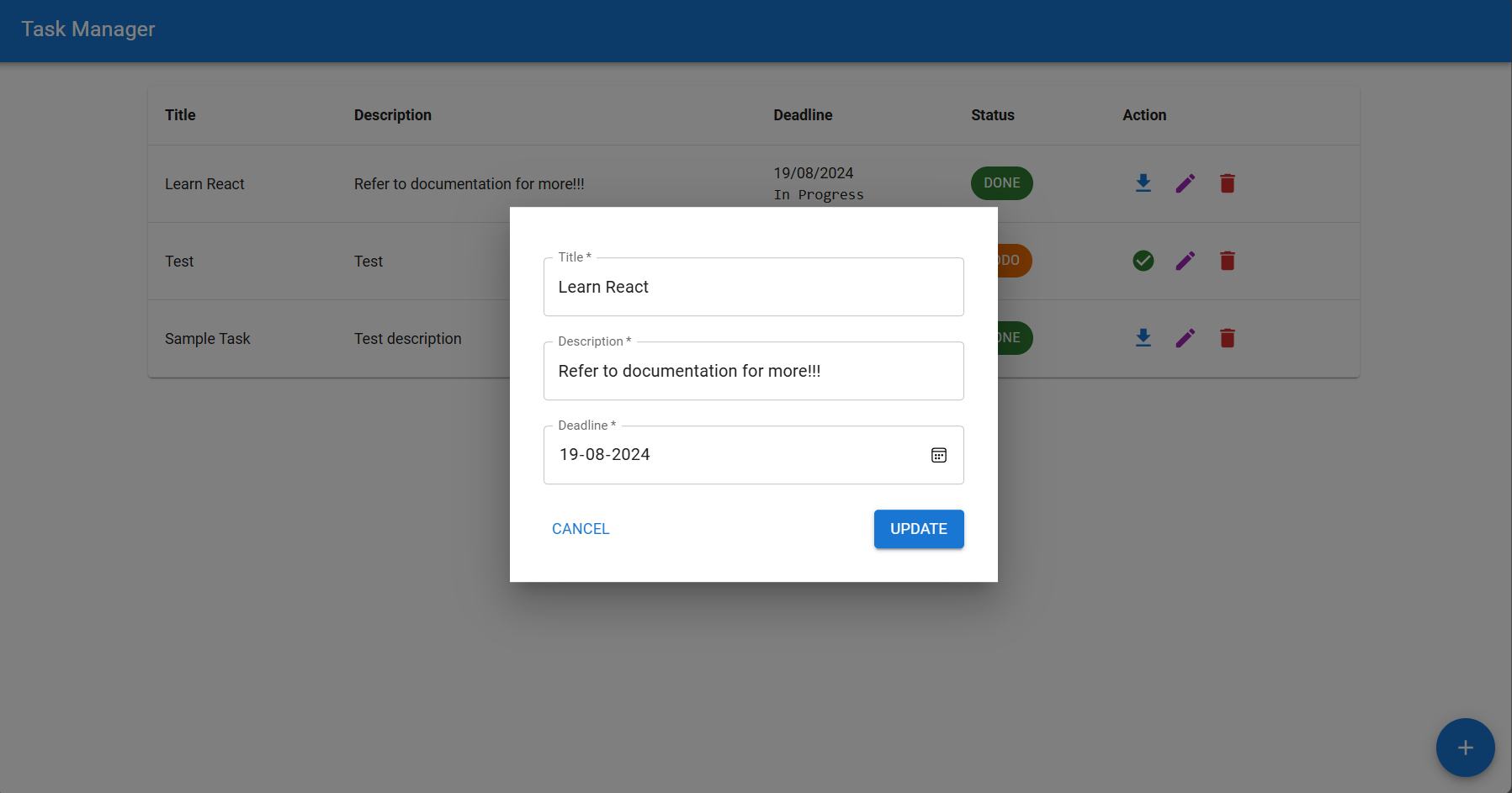
### Table with Data view



### Add Task View



### Edit Task View



### Delete Task

