

Problem Statement: Task Management System

Objective

Develop a **Task Management System** that enables users to manage tasks efficiently. This application will involve building a **backend API** for task management and a **frontend interface** for interacting with the system. The application should support CRUD (Create, Read, Update, Delete) operations with seamless integration between the frontend and backend.

Application Requirements

1. Backend Requirements

1. API Endpoints:

Method	Endpoint	Description
POST	/api/tasks/	Add a new task
GET	/api/tasks/	Retrieve all tasks
GET	/api/tasks/:id	Retrieve a task by ID
PUT	/api/tasks/:id	Update a task by ID
DELETE	/api/tasks/:id	Delete a task by ID

2. Schema:

```
const taskSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
  status: { type: String, enum: ['Pending', 'In Progress', 'Completed'], default: 'Pending' },
  dueDate: { type: Date, required: true },
});
```

o Fields:

- title: Task title (required).
- description: Task description (required).
- status: Task status (Pending, In Progress, or Completed).
- dueDate: Task deadline (required).

3. Status Code Handling:

Status Code	Meaning	When It's used
200 OK	Request succeeded	- For successful retrieval of tasks or a specific task. - For successful update or deletion of a task.
201 Created	Resource created successfully	- When a new task is successfully added.
400 Bad Request	Client sent invalid data	- When validation fails for a field. - Example: Missing title or invalid status.
404 Not Found	Resource not found	- When a task with the specified ID does not exist.
500 Internal Server Error	Server encountered an error	- For unexpected server-side errors. - Example: Database connection issues.

2. Frontend Requirements

1. Technologies:

- **React:** To build the user interface.
- **Axios:** To interact with the backend API.
- **React Router Dom:** To manage navigation between views.

2. Features:

- **View Tasks:**
 - Display all tasks in a table with columns for Title, Description, Status, Due Date, and Actions.
 - Include Edit and Delete buttons in the Actions column.
- **Add Task:**
 - A form to add a new task with fields for Title, Description, Status, and Due Date and use same name(attribute) for form fields.
 - Show a success or error message above the form after submission.
- **Edit Task:**
 - Inline editing in the table when the Edit button is clicked.
 - Save or cancel changes directly within the table row.
 - Show a success or error message above the table after saving changes.
- **Delete Task:**
 - Remove a task from the table using the Delete button.
 - Show a success or error message after deletion.

3. Navigation:

- Use a navigation bar with links:
 - **Task Management:** Displays the task table.
 - **Add Task:** Displays the form for adding a new task.

4. Validation:

- Ensure all fields are filled before submitting the form.
- Validate that the Due Date is not empty.

Deliverables

1. Backend:

- A fully functional backend API with:
 - CRUD operations for tasks.
 - Proper validation and error handling.

2. Frontend:

- A fully functional React application that:
 - Integrates seamlessly with the backend API.
 - Provides an intuitive user interface for managing tasks.
-

Evaluation Criteria

Functionality:

- All CRUD operations work as expected.
 - Seamless integration between frontend and backend.
-

URL to be shared on Toofan

<http://ipaddress>

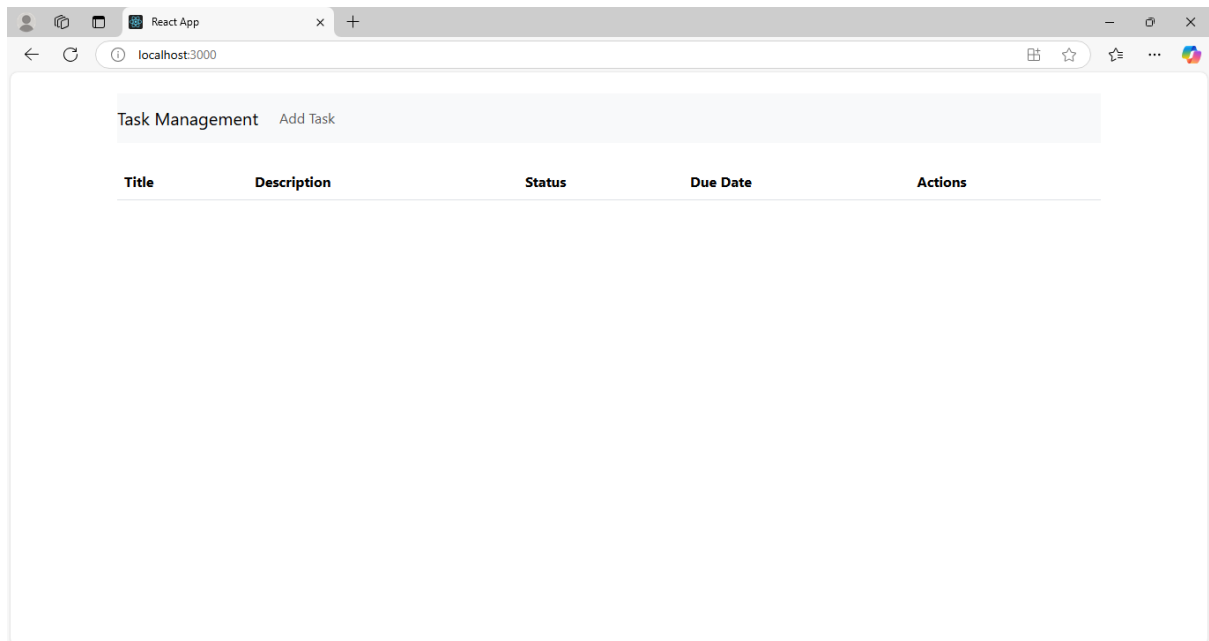
Example: <http://10.11.xxx.xxx>

NOTE:

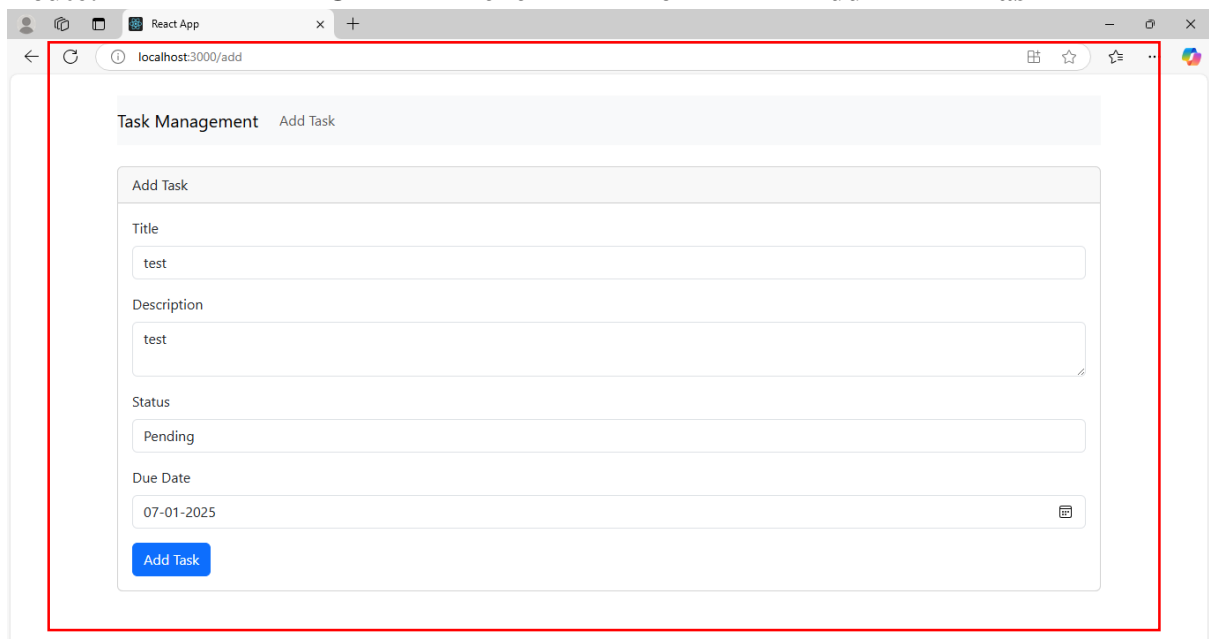
- While developing, Use **3000** port number for the frontend and **5000** port number for the backend.
 - Do not forget to use your **local ip address in axios calls**.
-

Sample Screenshots for reference

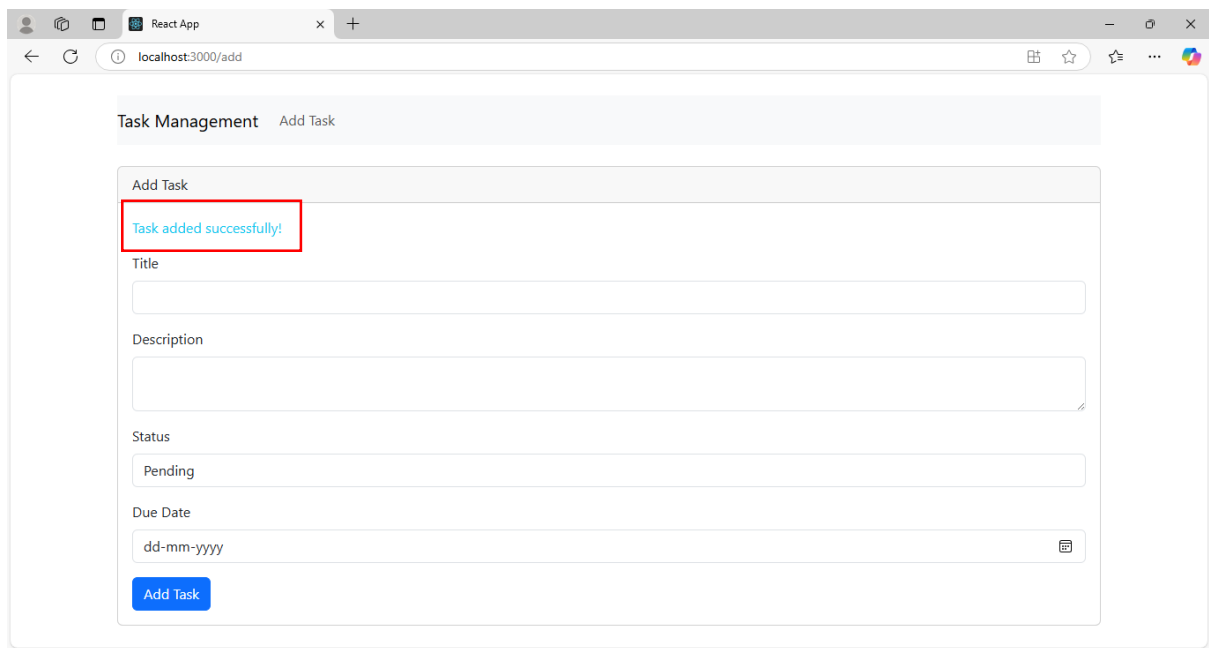
Home Component



Route: On click on Add Task link

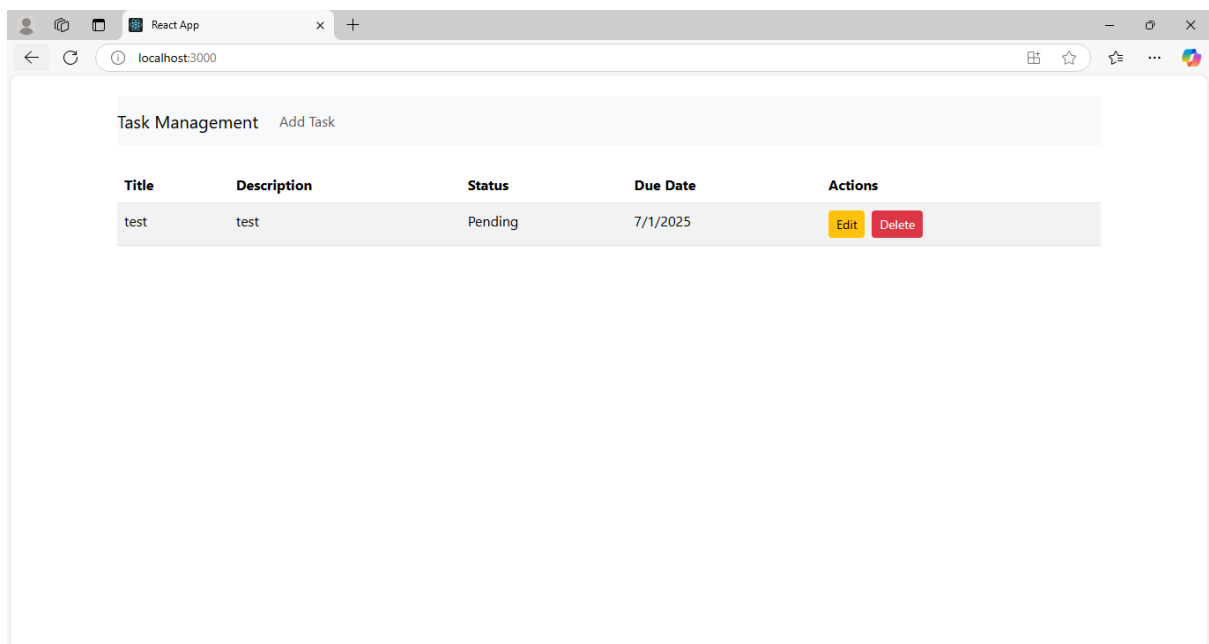


After adding task, Show up message after “Task added successfully” message.



The screenshot shows a web browser window with a single tab titled 'React App'. The address bar displays 'localhost:3000/add'. The page content is titled 'Task Management' with a sub-header 'Add Task'. Below this, there is a form titled 'Add Task'. Inside the form, a blue message box with the text 'Task added successfully!' is highlighted with a red border. Below the message, there are input fields for 'Title', 'Description', 'Status' (with a dropdown menu showing 'Pending'), and 'Due Date' (with a date picker showing 'dd-mm-yyyy'). At the bottom of the form is a blue button labeled 'Add Task'.

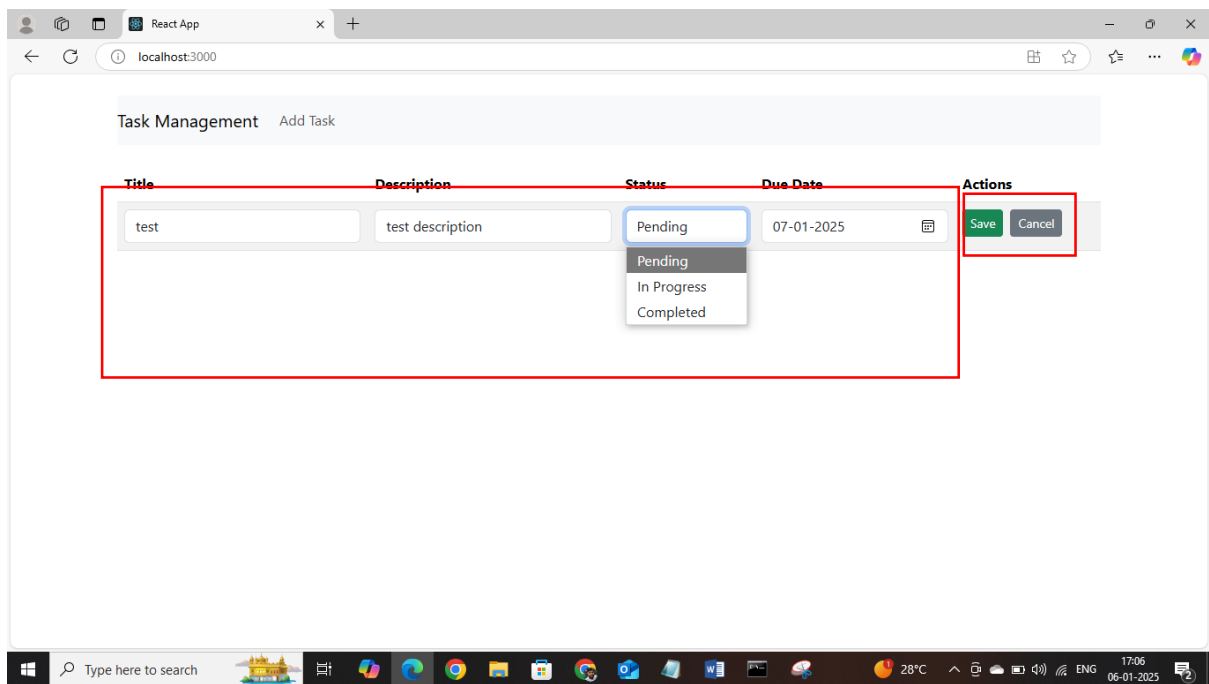
After adding task, show up added task in Task management Route



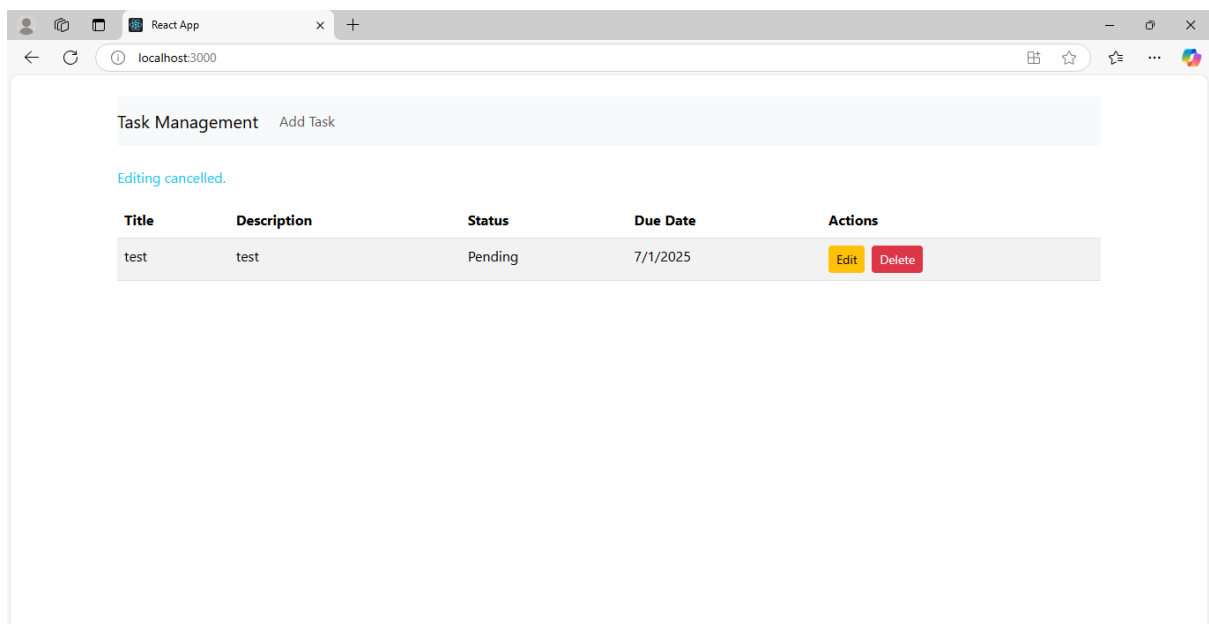
The screenshot shows a web browser window with a single tab titled 'React App'. The address bar displays 'localhost:3000'. The page content is titled 'Task Management' with a sub-header 'Add Task'. Below this, there is a table with the following columns: 'Title', 'Description', 'Status', 'Due Date', and 'Actions'. The table contains one row with the following data: 'test', 'test', 'Pending', '7/1/2025', and 'Edit Delete'.

Title	Description	Status	Due Date	Actions
test	test	Pending	7/1/2025	Edit Delete

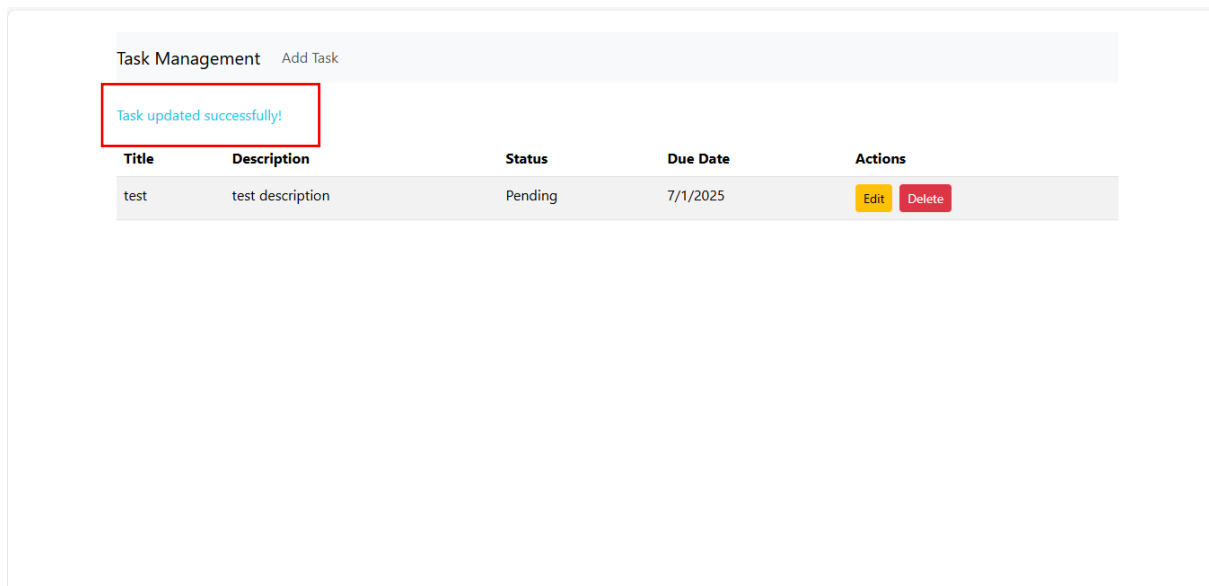
On Edit, inline editing should be supported



On Cancelling editing, show up “Editing cancelled” message



On saving editing, show up “Task updated successfully” message



On deleting a task, show up Task deleted successfully message

