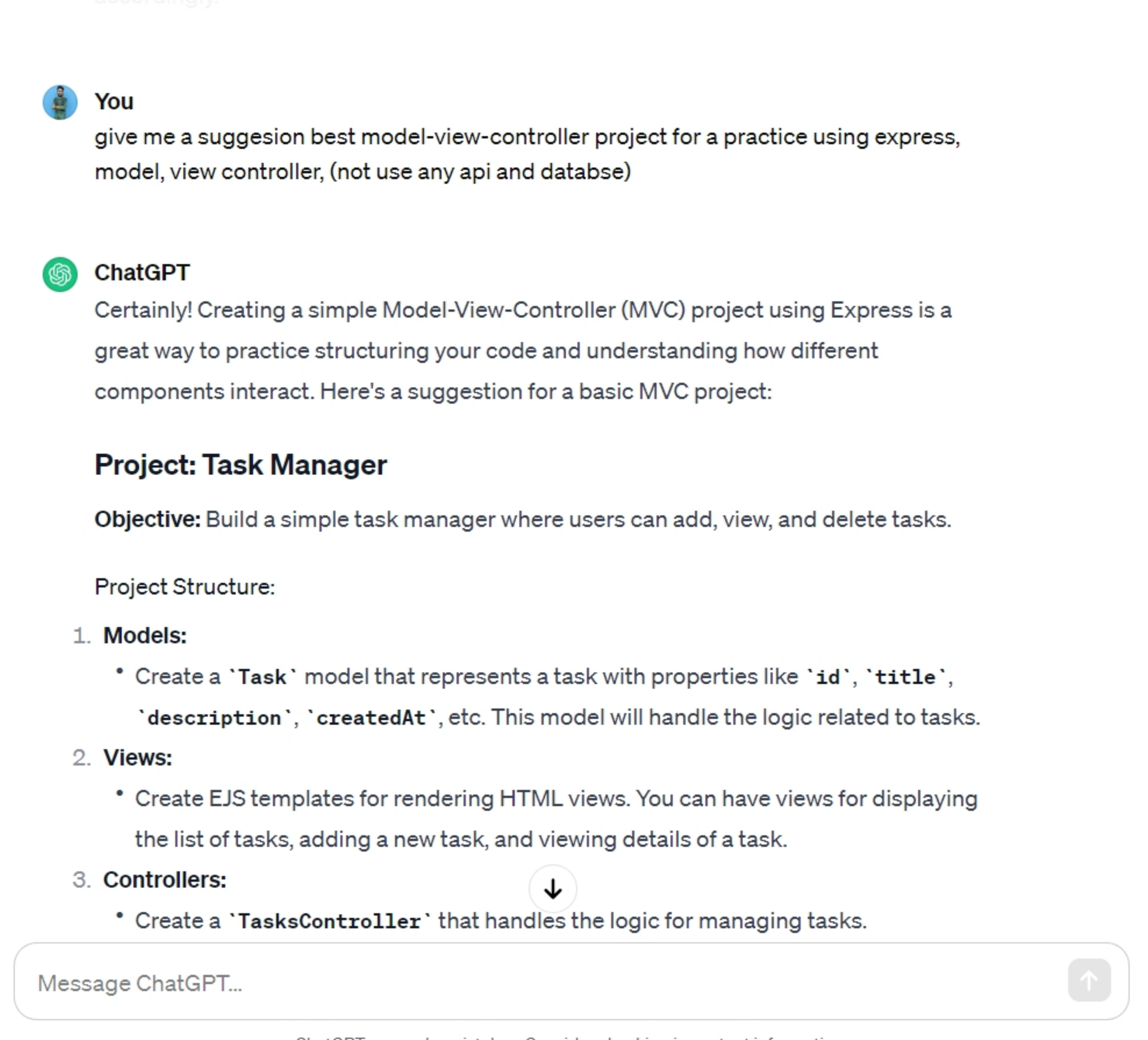


accordingly.

You

give me a suggesion best model-view-controller project for a practice using express, model, view controller, (not use any api and databse)

ChatGPT

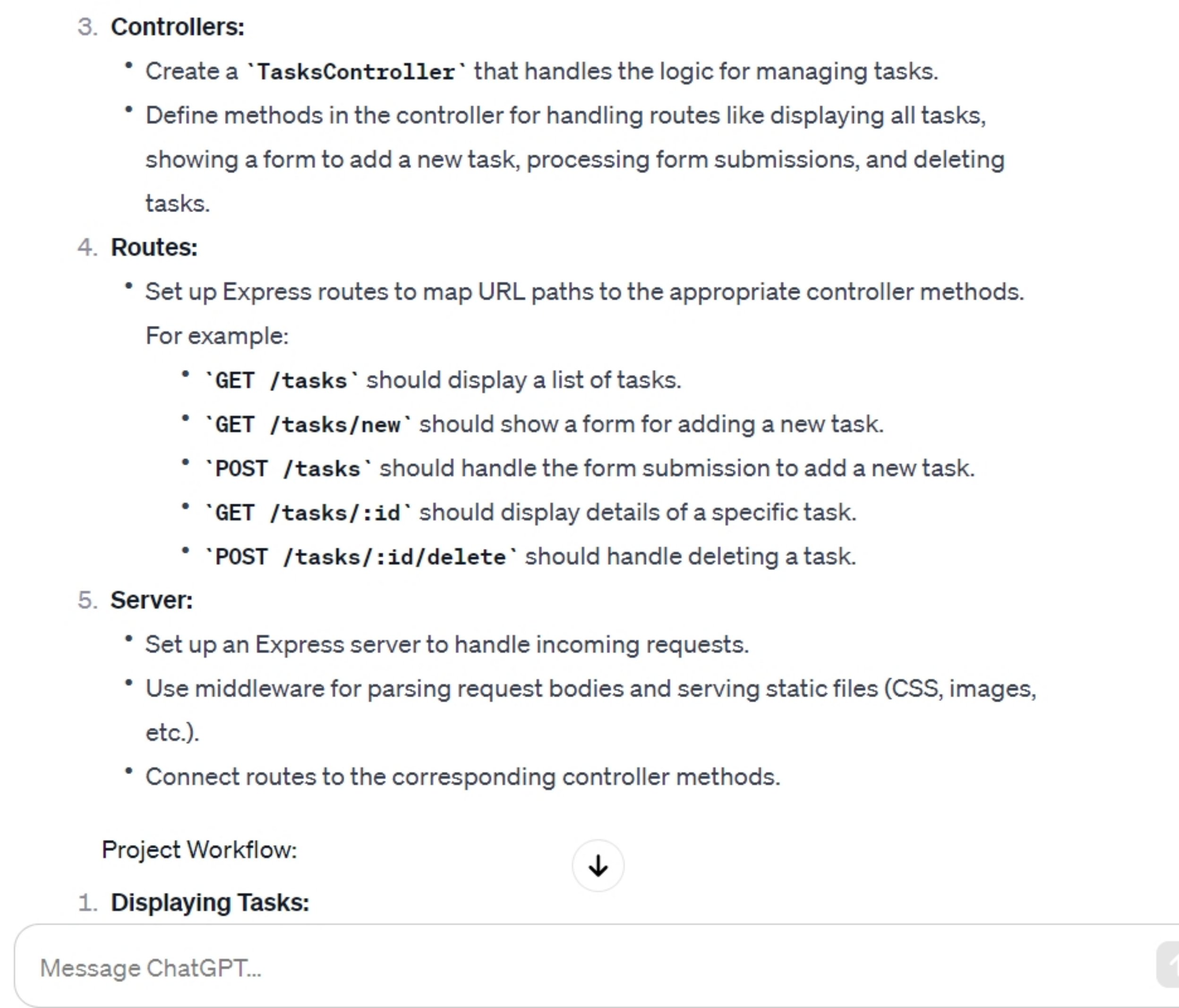
Certainly! Creating a simple Model-View-Controller (MVC) project using Express is a great way to practice structuring your code and understanding how different components interact. Here's a suggestion for a basic MVC project:

Project: Task Manager

Objective: Build a simple task manager where users can add, view, and delete tasks.

Project Structure:

1. **Models:**
   - Create a `Task` model that represents a task with properties like `id`, `title`, `description`, `createdAt`, etc. This model will handle the logic related to tasks.
2. **Views:**
   - Create EJS templates for rendering HTML views. You can have views for displaying the list of tasks, adding a new task, and viewing details of a task.
3. **Controllers:**
   - Create a `TasksController` that handles the logic for managing tasks.



3. **Controllers:**
   - Create a `TasksController` that handles the logic for managing tasks.
   - Define methods in the controller for handling routes like displaying all tasks, showing a form to add a new task, processing form submissions, and deleting tasks.
4. **Routes:**
   - Set up Express routes to map URL paths to the appropriate controller methods. For example:
     - `GET /tasks` should display a list of tasks.
     - `GET /tasks/new` should show a form for adding a new task.
     - `POST /tasks` should handle the form submission to add a new task.
     - `GET /tasks/:id` should display details of a specific task.
     - `POST /tasks/:id/delete` should handle deleting a task.
5. **Server:**
   - Set up an Express server to handle incoming requests.
   - Use middleware for parsing request bodies and serving static files (CSS, images, etc.).
   - Connect routes to the corresponding controller methods.

Project Workflow:

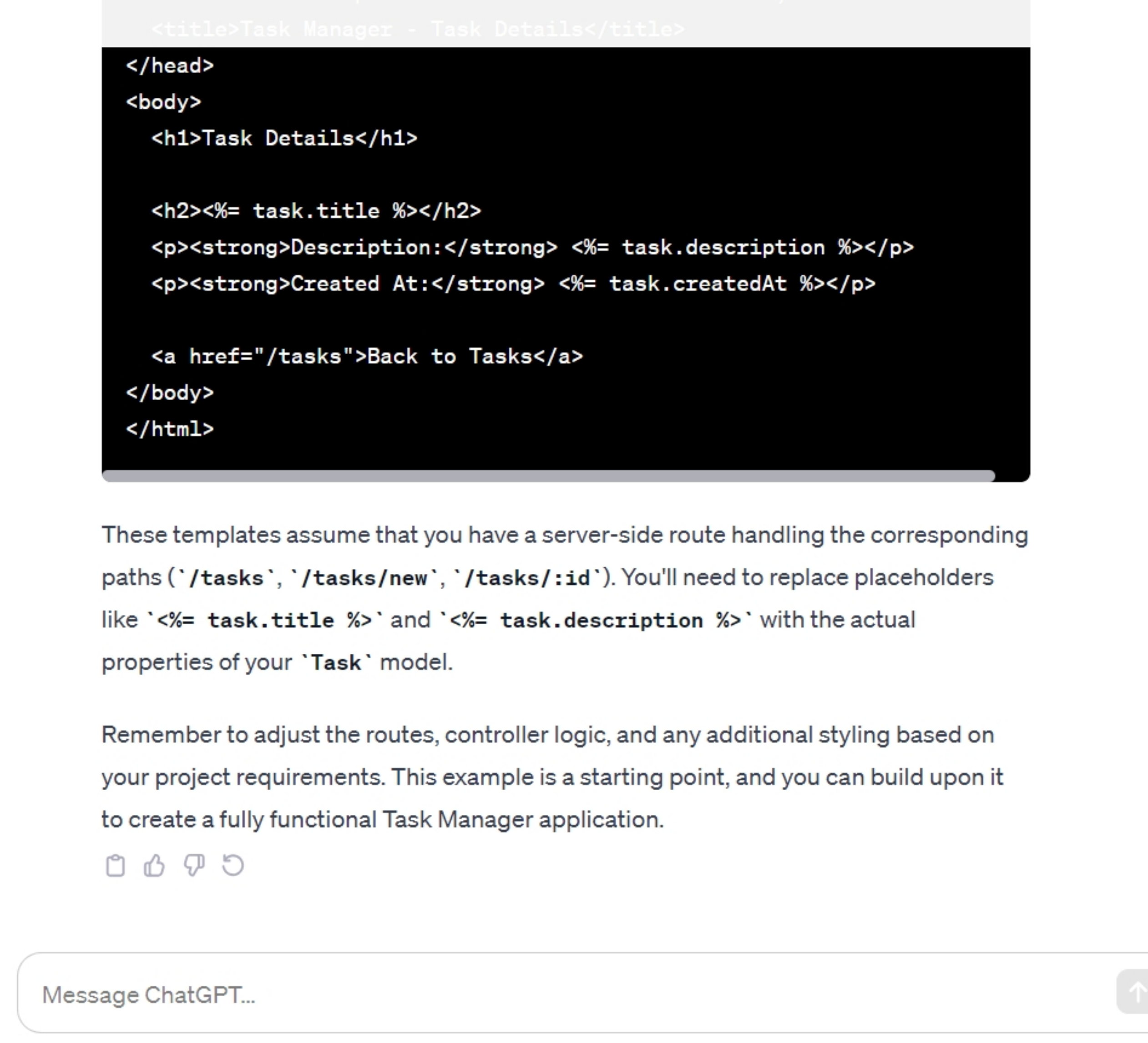
1. **Displaying Tasks:**

Project Workflow:

1. **Displaying Tasks:**
    - Implement the route and controller method to display a list of tasks on the homepage.
2. **Adding Tasks:**
    - Create a route to display a form for adding a new task.
    - Implement the corresponding controller method to handle the form submission and add a new task.
3. **Viewing Task Details:**
    - Implement a route and controller method to display details of a specific task when its link is clicked.
4. **Deleting Tasks:**
    - Add functionality to delete tasks. This can be achieved by having a delete button next to each task in the list.

Note:

- For simplicity, you can store tasks in an array in memory instead of using a database.
- Focus on the structure and separation of concerns (MVC), and you can later enhance the project by adding more features or int ting a database.



```
  <title>Task Manager - Task Details</title>
</head>
<body>
  <h1>Task Details</h1>

  <h2><%= task.title %></h2>
  <p><strong>Description:</strong> <%= task.description %></p>
  <p><strong>Created At:</strong> <%= task.createdAt %></p>

  <a href="/tasks">Back to Tasks</a>
</body>
</html>
```

These templates assume that you have a server-side route handling the corresponding paths (`/tasks`, `/tasks/new`, `/tasks/:id`). You'll need to replace placeholders like `<%= task.title %>` and `<%= task.description %>` with the actual properties of your `Task` model.

Remember to adjust the routes, controller logic, and any additional styling based on your project requirements. This example is a starting point, and you can build upon it to create a fully functional Task Manager application.