Full Task Manager App (MongoLess)

### Backend: Go (Gin framework) with in-memory storage

### Frontend: React (Vite) with Bootstrap

# Step 1: Set Up the Go Backend

### 1.1 Create a New Go Project

|  |
| --- |
| mkdir task-manager-api  cd task-manager-api  go mod init taskmanager |

### 1.2 Install Dependencies

|  |
| --- |
| go get -u github.com/gin-gonic/gin  go get -u github.com/gin-contrib/cors |

**go get -u github.com/gin-gonic/gin**

* This command installs or updates (-u flag) the **Gin** web framework, which is a high-performance, lightweight HTTP web framework for Go.
* It is commonly used to build RESTful APIs with minimal boilerplate code.

**go get -u github.com/gin-contrib/cors**

* This installs or updates the **CORS (Cross-Origin Resource Sharing) middleware** for Gin, which allows handling requests from different origins.
* It is useful when building APIs that are consumed by frontend applications hosted on a different domain.

### 1.3 Implement the Go Backend

Create **main.go**:

|  |
| --- |
| package main  import (      "net/http"      "strconv"      "github.com/gin-contrib/cors"      "github.com/gin-gonic/gin"  )  *// Task struct*  type Task struct {      ID     int    `json:"id"`      Title  string `json:"title"`      Status string `json:"status"`  }  *// In-memory storage*  var tasks = []Task{}  var nextID = 1  func main() {      r := gin.Default()  *// Enable CORS*      r.Use(cors.New(cors.Config{          AllowOrigins:     []string{"http://localhost:5173"},          AllowMethods:     []string{"GET", "POST", "PUT", "DELETE", "OPTIONS"},          AllowHeaders:     []string{"Origin", "Content-Type", "Authorization"},      }))  *// Routes*      r.GET("/tasks", getTasks)      *// Get all tasks*      r.POST("/tasks", createTask)   *// Create a task*      r.GET("/tasks/:id", getTaskByID) *// Get task by ID*      r.PUT("/tasks/:id", updateTask) *// Update task by ID*      r.DELETE("/tasks/:id", deleteTask) *// Delete task by ID*  *// Start the server*      r.Run(":8080")  }  *// Get all tasks*  func getTasks(c \*gin.Context) {      c.JSON(http.StatusOK, tasks)  }  *// Create a new task*  func createTask(c \*gin.Context) {      var newTask Task      if err := c.ShouldBindJSON(&newTask); err != nil {          c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid task data"})          return      }      if newTask.Title == "" {          c.JSON(http.StatusBadRequest, gin.H{"error": "Task title cannot be empty"})          return      }      newTask.ID = nextID      nextID++      tasks = append(tasks, newTask)      c.JSON(http.StatusCreated, newTask)  }  *// Get a task by ID*  func getTaskByID(c \*gin.Context) {      id, err := strconv.Atoi(c.Param("id")) *// Convert ID from URL param*      if err != nil {          c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid task ID"})          return      }      for \_, task := range tasks {          if task.ID == id {              c.JSON(http.StatusOK, task)              return          }      }      c.JSON(http.StatusNotFound, gin.H{"error": "Task not found"})  }  *// Update a task by ID*  func updateTask(c \*gin.Context) {      id, err := strconv.Atoi(c.Param("id")) *// Convert ID from URL param*      if err != nil {          c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid task ID"})          return      }      var updatedTask Task      if err := c.ShouldBindJSON(&updatedTask); err != nil {          c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid task data"})          return      }      for i, t := range tasks {          if t.ID == id {              tasks[i].Title = updatedTask.Title              tasks[i].Status = updatedTask.Status              c.JSON(http.StatusOK, tasks[i])              return          }      }      c.JSON(http.StatusNotFound, gin.H{"error": "Task not found"})  }  *// Delete a task by ID*  func deleteTask(c \*gin.Context) {      id, err := strconv.Atoi(c.Param("id")) *// Convert ID from URL param*      if err != nil {          c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid task ID"})          return      }      for i, t := range tasks {          if t.ID == id {              tasks = append(tasks[:i], tasks[i+1:]...) *// Remove task from slice*              c.JSON(http.StatusOK, gin.H{"message": "Task deleted"})              return          }      }      c.JSON(http.StatusNotFound, gin.H{"error": "Task not found"})  } |

Run the backend:

|  |
| --- |
| go run main.go |

# Step 2: Set Up the React Frontend

## Create a Vite React App

|  |
| --- |
| npm create vite@latest task-manager-client --template react  cd task-manager-client  npm install  npm install axios react-router-dom bootstrap |

# Step 3: Implement Task Manager Frontend

Modify **main.jsx**:

|  |
| --- |
| import { StrictMode } from "react";  import { createRoot } from "react-dom/client";  import "bootstrap/dist/css/bootstrap.min.css";  import "bootstrap/dist/js/bootstrap.bundle.min";  import App from "./App.jsx";  createRoot(document.getElementById("root")).render(    <StrictMode>      <App />    </StrictMode>  ); |

Modify **App.jsx**:

|  |
| --- |
| import { BrowserRouter, Route, Routes } from "react-router-dom";  import TaskList from "./components/TaskList";  import TaskCreate from "./components/TaskCreate";  import TaskView from "./components/TaskView";  import TaskEdit from "./components/TaskEdit"; *// Added TaskEdit*  function App() {    return (      <BrowserRouter>        <div *className*="container mt-4">          <Routes>            <Route *path*="/" *element*={<TaskList />} />            <Route *path*="/tasks/list" *element*={<TaskList />} />            <Route *path*="/tasks/create" *element*={<TaskCreate />} />            <Route *path*="/tasks/view/:id" *element*={<TaskView />} />            <Route *path*="/tasks/edit/:id" *element*={<TaskEdit />} />{" "}            {*/\* New Edit Route \*/*}          </Routes>        </div>      </BrowserRouter>    );  }  export default App; |

Modify **PageHeader.jsx**:

|  |
| --- |
| import { Link } from "react-router-dom";  const PageHeader = () => (    <nav *className*="navbar navbar-expand-lg navbar-dark bg-dark">      <div *className*="container-fluid">        <Link *to*="/" *className*="navbar-brand">          Task Manager        </Link>        <button  *type*="button"  *className*="navbar-toggler"  *data-bs-toggle*="collapse"  *data-bs-target*="#navbarmenu"        >          <span *className*="navbar-toggler-icon"> </span>        </button>        <div *id*="navbarmenu" *className*="collapse navbar-collapse">          <ul *className*="navbar-nav me-auto mb-2 mb-lg-0">            <li *className*="nav-item">              <Link *to*="/" *className*="nav-link active">                Tasks              </Link>            </li>            <li *className*="nav-item">              <Link *to*="/tasks/create" *className*="nav-link">                Add Task              </Link>            </li>          </ul>        </div>      </div>    </nav>  );  export default PageHeader; |

## Overview of CRUD Operations

**Create Task** → TaskCreate.jsx (Modify to use API)  
**Read Tasks** → TaskList.jsx (Modify to fetch from API, add Delete button)  
**View Task** → TaskView.jsx (Modify to fetch from API)  
**Update Task** → **New: TaskEdit.jsx**  
**Delete Task** → **Handled in TaskList.jsx**

### 1. Modify TaskCreate.jsx (Add API Integration)

**Changes:**

**Added API integration (axios.post)** → Sends task data to backend.  
**Used useNavigate()** → Redirects user to the task list after adding a task.

|  |
| --- |
| import { useState } from "react";  import { useNavigate } from "react-router-dom";  import axios from "axios";  const TaskCreate = () => {    const [task, setTask] = useState({ title: "", status: "Pending" });    const navigate = useNavigate();    const handleSubmit = async (e) => {      e.preventDefault(); *// Prevents the form from refreshing the page*      try {        const response = await axios.post("http://localhost:8080/tasks", task, {          headers: { "Content-Type": "application/json" }, *// Ensure JSON format*        });        console.log("Task Created:", response.data);        alert("Task Created Successfully!");        navigate("/tasks/list"); *// Redirects after successful creation*      } catch (error) {        console.error(          "Error creating task:",          error.response ? error.response.data : error.message        );        alert("Error creating task");      }    };    return (      <div *className*="container mt-4">        <h3>Add Task</h3>        <form *onSubmit*={handleSubmit}>          <div *className*="form-group mb-3">            <label *className*="form-label">Task Title:</label>            <input  *type*="text"  *className*="form-control"  *value*={task.title}  *onChange*={(e) => setTask({ ...task, title: e.target.value })}  *placeholder*="Enter task title"  *required*            />          </div>          <div *className*="form-group mb-3">            <label *className*="form-label">Task Status:</label>            <select  *className*="form-control"  *value*={task.status}  *onChange*={(e) => setTask({ ...task, status: e.target.value })}            >              <option *value*="Pending">Pending</option>              <option *value*="Completed">Completed</option>            </select>          </div>          <button *type*="submit" *className*="btn btn-primary">            Create Task          </button>        </form>      </div>    );  };  export default TaskCreate; |

### 2. Modify TaskList.jsx (Fetch Data, Add Delete Function)

**Changes:**

Fetched tasks from backend (axios.get).  
Added Delete button with axios.delete.  
Used .map() to dynamically display tasks.

|  |
| --- |
| import { useEffect, useState } from "react";  import { Link } from "react-router-dom";  import axios from "axios";  const TaskList = () => {  const [tasks, setTasks] = useState([]);  const [error, setError] = useState("");  useEffect(() => {  axios  .get("http://localhost:8080/tasks")  .then((response) => {  console.log("API Response:", response.data); // Debugging log  setTasks(response.data);  })  .catch(() => setError("Failed to load tasks"));  }, []);  // Delete task function  const deleteTask = async (id) => {  if (window.confirm("Are you sure you want to delete this task?")) {  try {  await axios.delete(http://localhost:8080/tasks/${id}); // DELETE request to backend  setTasks(tasks.filter((task) => task.id !== id)); // Remove task from state  alert("Task deleted successfully");  } catch (error) {  alert("Failed to delete task");  console.error("Error deleting task:", error);  }  }  };  return (  <div className="container mt-4">  <h3>Task List</h3>  {error && <p className="text-danger">{error}</p>}  <table className="table table-striped table-hover">  <thead className="table-dark">  <tr>  <th>ID</th>  <th>Task Title</th>  <th>Status</th>  <th>Actions</th>  </tr>  </thead>  <tbody>  {tasks.map((task) => (  <tr key={task.id}>  <td>{task.id}</td>  <td>{task.title || "No Title"}</td> {/\* Ensure title is displayed \*/}  <td>{task.status}</td>  <td>  <Link  to={/tasks/view/${task.id}}  className="btn btn-primary btn-sm me-2"  >  View  </Link>  <Link  to={/tasks/edit/${task.id}}  className="btn btn-warning btn-sm me-2"  >  Edit  </Link>  <button  className="btn btn-danger btn-sm"  onClick={() => deleteTask(task.id)}  >  Delete  </button>  </td>  </tr>  ))}  </tbody>  </table>  </div>  );  };  export default TaskList; |

### 3. Modify TaskView.jsx (Fetch Single Task)

**Changes:**

Fetched task by ID (axios.get).  
Used useParams() to get task ID from URL.  
Displayed task details dynamically.

|  |
| --- |
| import { useEffect, useState } from "react";  import { useParams, Link } from "react-router-dom";  import axios from "axios";  const TaskView = () => {    const { id } = useParams(); *// Extract ID from URL*    const [task, setTask] = useState(null);    const [error, setError] = useState("");    useEffect(() => {      axios        .get(`http://localhost:8080/tasks/${id}`)        .then((response) => {          console.log("API Response:", response.data); *// Debugging Log*          setTask(response.data);        })        .catch(() => setError("Task not found"));    }, [id]);    if (error) return <p *className*="text-danger">{error}</p>;    if (!task) return <p>Loading task...</p>;    return (      <div *className*="container mt-4">        <h3>View Task</h3>        <div *className*="card p-3">          <p>            <strong>Task ID:</strong> {task.id}          </p>          <p>            <strong>Task Title:</strong> {task.title || "No Title"}          </p>          <p>            <strong>Task Status:</strong> {task.status}          </p>        </div>        <Link *to*="/tasks/list" *className*="btn btn-light mt-3">          Go Back        </Link>      </div>    );  };  export default TaskView; |

### 4. New TaskEdit.jsx (Update Task)

**Features:**

Fetches **task details by ID** and pre-fills form.  
Allows **updating task title and status**.  
Uses **axios.put** to send updated task data to backend.

|  |
| --- |
| import { useEffect, useState } from "react";  import { useParams, useNavigate, Link } from "react-router-dom";  import axios from "axios";  import PageHeader from "./PageHeader";  const TaskEdit = () => {    const { id } = useParams();    const navigate = useNavigate();    const [task, setTask] = useState({ name: "", status: "Pending" });    useEffect(() => {      axios        .get(`http://localhost:8080/tasks/${id}`)        .then((response) => setTask(response.data))        .catch(() => alert("Task not found"));    }, [id]);    const handleSubmit = async (e) => {      e.preventDefault();      try {        await axios.put(`http://localhost:8080/tasks/${id}`, task);        alert("Task Updated Successfully!");        navigate("/tasks/list");      } catch (error) {        alert("Error updating task");      }    };    return (      <div>        <PageHeader />        <div *className*="container">          <h3>Edit Task</h3>          <form *onSubmit*={handleSubmit}>            <input  *type*="text"  *className*="form-control mb-2"  *value*={task.name}  *onChange*={(e) => setTask({ ...task, name: e.target.value })}            />            <select  *className*="form-control mb-2"  *value*={task.status}  *onChange*={(e) => setTask({ ...task, status: e.target.value })}            >              <option *value*="Pending">Pending</option>              <option *value*="Completed">Completed</option>            </select>            <button *className*="btn btn-warning">Update Task</button>          </form>          <Link *to*="/tasks/list" *className*="btn btn-light mt-2">            Go Back          </Link>        </div>      </div>    );  };  export default TaskEdit; |

## Step 4: Run the Frontend

|  |
| --- |
| npm run dev |