

Day 05 Assignment

Name – Soham Sarkar

College Name – MCKV INSTITUTE OF ENGINEERING



June 17, 2025

**Create a simple To-Do List web application using Node.js, Express, and EJS, by applying the same structure and concepts used in the BMI calculator project.**

**Requirements:**

1. Use **Node.js + Express + EJS** as in the BMI calculator.
2. Implement **a form** to enter tasks.
3. Display all added tasks dynamically using **EJS rendering**.
4. Include a **POST route** to add new tasks.
5. Add functionality to **delete a task** using a separate POST route.
6. Use **body-parser** to handle form data.
7. Use a **simple array** to store tasks in memory.
8. Style the page using basic CSS (optional).

**Console:**

npm init -y

npm I express ejs nodemon body-parser

**Folder Structure:**

cd public

code style.css

cd ..

cd views

code index.ejs

cd ..

code app.js

**For Using app.js file with nodemon:**

We have to add this modified script in package.json file

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "app":"nodemon app.js"

  },

**Main Server File(app.js):**

* //Loads the Express module, which is a framework to build web servers easily.
* const express = require("express");
* //Loads the body-parser middleware, used to extract data from form submissions (req.body).
* const bodyParser = require("body-parser");
* //Creates an instance of the Express application. This app will be used to define routes and configure the server.
* const app = express();
* //Creates an empty array to store the tasks temporarily (in memory). This means data will be lost when the server restarts.
* let tasks = [];
* Tells Express to use EJS as the template engine so it can render dynamic HTML files from the views/ folder.
* app.set("view engine", "ejs");
* //Configures body-parser to parse URL-encoded form data (like form submissions), so you can access data using req.body.
* app.use(bodyParser.urlencoded({ extended: true }));
* //Tells Express to serve static files (like CSS or images) from the public/ directory.
* app.use(express.static("public"));
* // Defines a GET route for the homepage (/):
* //It renders the index.ejs file inside views/
* //It passes the current tasks array to the template for display.
* app.get("/", (req, res) => {
* res.render("index", { tasks });
* });
* //Defines a POST route to handle task submission from the form:
* //Gets the task input using req.body.task
* //Trims and checks if it's not empty, then adds it to the tasks array
* //Redirects back to / to refresh the task list.
* app.post("/add", (req, res) => {
* const newTask = req.body.task;
* if (newTask.trim()) tasks.push(newTask);
* res.redirect("/");
* });
* //Defines a POST route to delete a task:
* //Gets the index of the task to delete from a hidden input field
* //If it's a valid number, removes that task from the array
* //Redirects to / to update the UI.
* app.post("/delete", (req, res) => {
* const index = parseInt(req.body.index);
* if (!isNaN(index)) tasks.splice(index, 1);
* res.redirect("/");
* });
* //Starts the server on port 3000. When running, it prints a message to the console.
* app.listen(3000, () => {
* console.log("Server started on http://localhost:3000");
* });

**views/Index.ejs File:**

* <!DOCTYPE html>
* <html lang="en">
* <head>
* <meta charset="UTF-8">
* <title>To-Do List</title>
* <link rel="stylesheet" href="/style.css">
* <!-- Links the external CSS file for styling (served from /public folder in Express) -->
* </head>
* <body>
* <div class="container">
* <!-- Main container div for layout and styling -->
* <h1>My To-Do List</h1>
* <!-- Page heading -->
* <!-- Form to add a new task -->
* <form action="/add" method="POST">
* <!-- Sends a POST request to /add when submitted -->
* <input type="text" name="task" placeholder="Enter a new task" required>
* <!-- Text input for task name; 'name="task"' used to access this in req.body.task -->
* <button type="submit">Add</button>
* <!-- Button to submit the task -->
* </form>
* <!-- Unordered list to display all tasks -->
* <ul>
* <% tasks.forEach((task, index) => { %>
  + <!-- Loop through each task in the tasks array -->
  + <li>
  + <%= task %>
  + <!-- Output the task text using EJS (HTML-escaped) -->
  + <!-- Inline form to delete the specific task -->
  + <form action="/delete" method="POST" style="display:inline;">
    - <!-- Sends a POST request to /delete when the delete button is clicked -->
    - <!-- 'display:inline' keeps the form/button next to the task text -->
    - <input type="hidden" name="index" value="<%= index %>">
    - <!-- Hidden input to send the index of the task to delete -->
    - <button type="submit">❌</button>
    - <!-- Button to delete the task -->
  + </form>
  + </li>
* <% }); %>
* <!-- End of EJS forEach loop -->
* </ul>
* </div>
* </body>
* </html>

**Public/style.css File:**

* body {
* font-family: Arial, sans-serif;
* background: #f4f4f4;
* padding: 40px;
* }
* .container {
* background: white;
* max-width: 500px;
* margin: auto;
* padding: 20px;
* border-radius: 8px;
* box-shadow: 0 0 10px #ccc;
* }
* h1 {
* text-align: center;
* }
* form input[type="text"] {
* width: 70%;
* padding: 8px;
* }
* form button {
* padding: 8px 12px;
* margin-left: 10px;
* }
* ul {
* list-style: none;
* padding: 0;
* }
* li {
* margin-top: 10px;
* padding: 6px;
* background: #e2e2e2;
* border-radius: 4px;
* display: flex;
* justify-content: space-between;
* align-items: center;
* }

**Run The app.js File:**

node app.js

or

npm run app.js

(this runs using nodemon and changes saved in the server automatically no need to restart server)

**View The Result:**

Visit <http://localhost:3000>

**OutPut Screenshot**

