TASK TRACKER APPLICATION

INTRODUCTION:

The \*Task Tracker\* application is a simple web-based tool that allows users to manage their daily tasks. It is built using vanilla JavaScript, HTML, and CSS, and provides functionalities to add,view,edit, and delete tasks.

Table Of Contents

1. [Project Setup](#project-setup)

2. [User Interface](#user-interface)

3. [Functionalities](#functionalities)

- [Add Task](#add-task)

- [Display Tasks](#display-tasks)

- [Edit Task](#edit-task)

- [Delete Task](#delete-task)

4. [How It Works](#how-it-works)

5. [Testing the Application](#testing-the-application)

6. [Conclusion](#conclusion)

**Project Setup:**

1. Directory Structure

The project should be organized with the following structure:

* **Task tracker**

1.index.html

2.styles.css

3.script.js

2. File Descriptions

 index.html: The main HTML file that structures the application’s interface.

styles.css: The CSS file that styles the application.

 script.js: JavaScript file that handles the application's logic and functionalities.

3. User Interface

 1. Overview

The user interface is designed to be simple and user-friendly, containing the following elements:

* An input field for adding new tasks.
* A button to add the entered task to the list.
* A dynamically generated list that displays all the tasks.
* Buttons for each task to edit or delete them.

 2. HTML Structure

The index.html file contains the basic structure of the application:

***html***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Task Tracker</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<h1>Task Tracker</h1>

<div class="input-group">

<input type="text" id="task-input" placeholder="Add a new task">

<button id="add-task-btn">Add Task</button>

</div>

<ul id="task-list"></ul>

</div>

<script src="script.js"></script>

</body>

</html>

3. Styling

The styles.css file provides styling for the application, making the interface clean and responsive:

***css***

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}.container {

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: 300px;

}

h1 {

text-align: center;

margin-bottom: 20px;

}

.input-group {

display: flex;

justify-content: space-between;

}

input[type="text"] {

flex: 1;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

margin-left: 10px;

padding: 10px;

background-color: #28a745;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

ul {

list-style-type: none;

padding: 0;

}

li {

background-color: #f8f9fa;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

margin-top: 10px;

display: flex;

justify-content: space-between;

}

li button {

margin-left: 10px;

background-color: #dc3545;

}

li button.edit-btn {

background-color: #007bff;

}

li button:hover {

opacity: 0.8;

}

**Functionalities:**

* **Add Task**
* **Input**: Users can type a task description in the input field.
* **Action**: Clicking the "Add Task" button will add the task to the list below.
* **JavaScript Implementation**:

javascript

addTaskBtn.addEventListener("click", () => {

const task = taskInput.value.trim();

if (task) {

addTask(task);

taskInput.value = '';

}

});

* **Display Tasks**
* **Description:** All tasks are displayed in a list format, with each task having options to edit or delete it.
* **Rendering**: The tasks are rendered dynamically whenever a task is added, edited, or deleted.

                                           javascript

function renderTasks() {

taskList.innerHTML = '';

tasks.forEach((task, index) => {

const li = document.createElement("li");

li.innerHTML = `

<span>${task}</span>

<div>

<button class="edit-btn" data-index="${index}">Edit</button>

<button class="delete-btn" data-index="${index}">Delete</button>

</div>

`;

taskList.appendChild(li);

});

}

* **Edit Task**
* **Action:** Users can click the "Edit" button next to a task to modify its description.
* **Prompt:** A prompt box appears with the current task description, allowing the user to edit and save changes.

Javascript

                             taskList.addEventListener("click", (e) => {

if (e.target.classList.contains("edit-btn")) {

const index = e.target.dataset.index;

const newTask = prompt("Edit Task:", tasks[index]);

if (newTask) {

editTask(index, newTask);

}

}

});

* **Delete Task**

                                           javascript

taskList.addEventListener("click", (e) => {

if (e.target.classList.contains("delete-btn")) {

const index = e.target.dataset.index;

deleteTask(index);

}

});

**How It Works:**

* **Task Management:**

The application manages tasks using an array in JavaScript. Each task is stored as a string in the array, and the array is updated with each CRUD operation (Create, Read, Update, Delete). The list is dynamically re-rendered on the page each time an operation is performed.

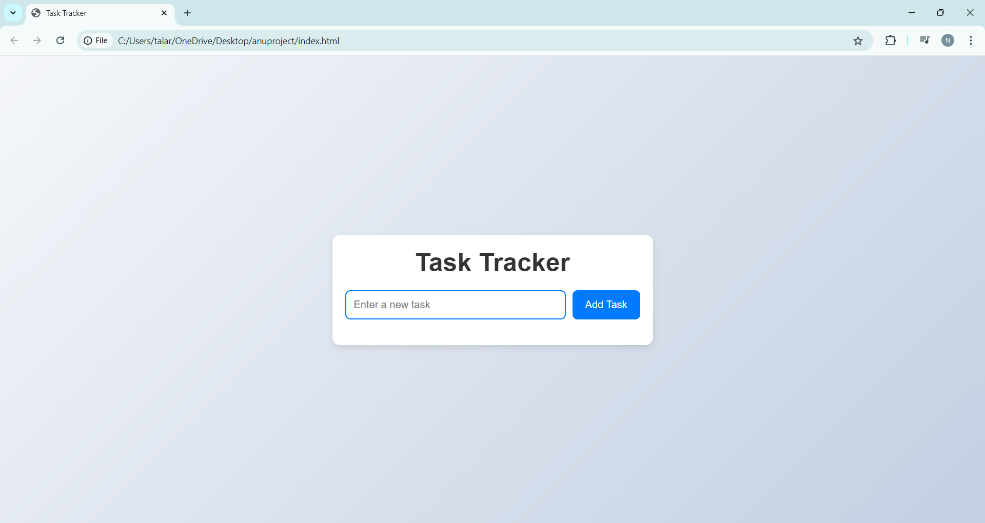
* **Event Listeners:**

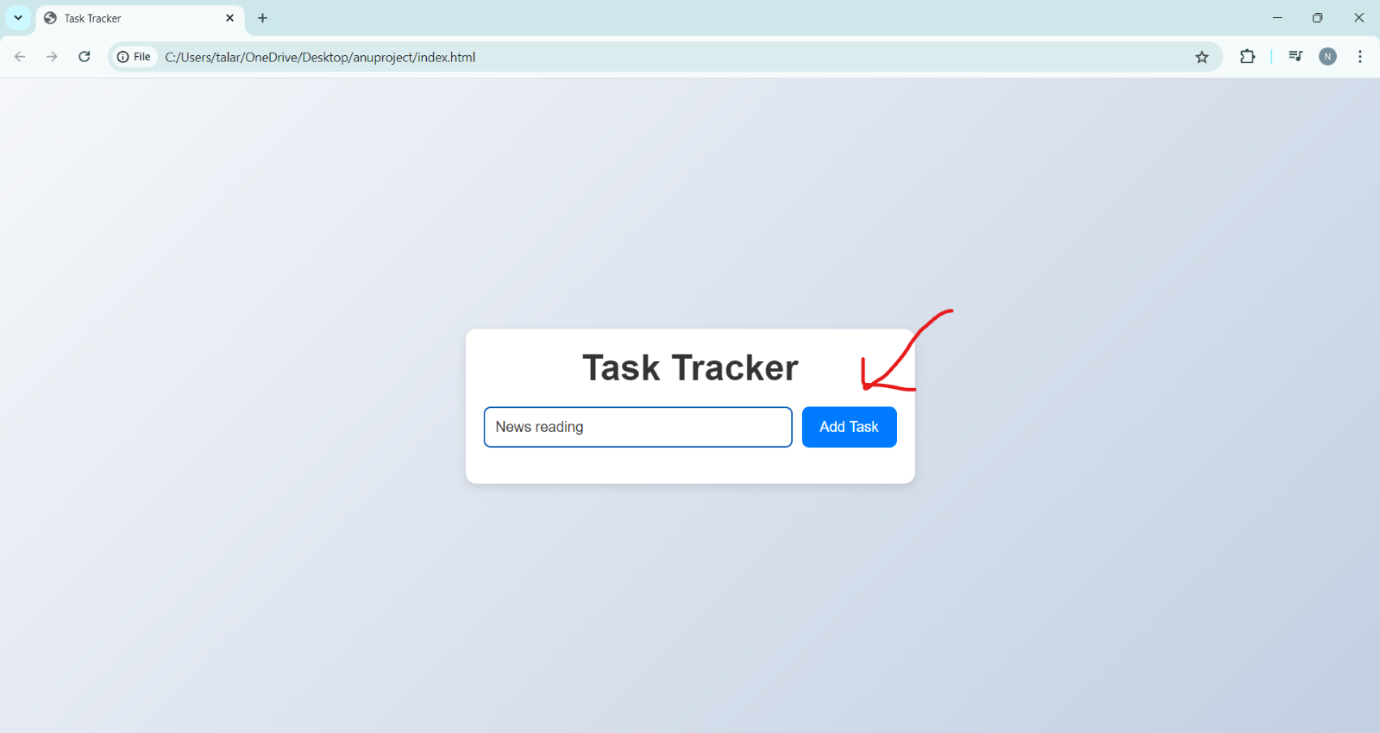
Event listeners are used to handle user interactions such as adding, editing, and deleting tasks. These listeners trigger the corresponding functions that update the task list.

* **Testing the Application:**

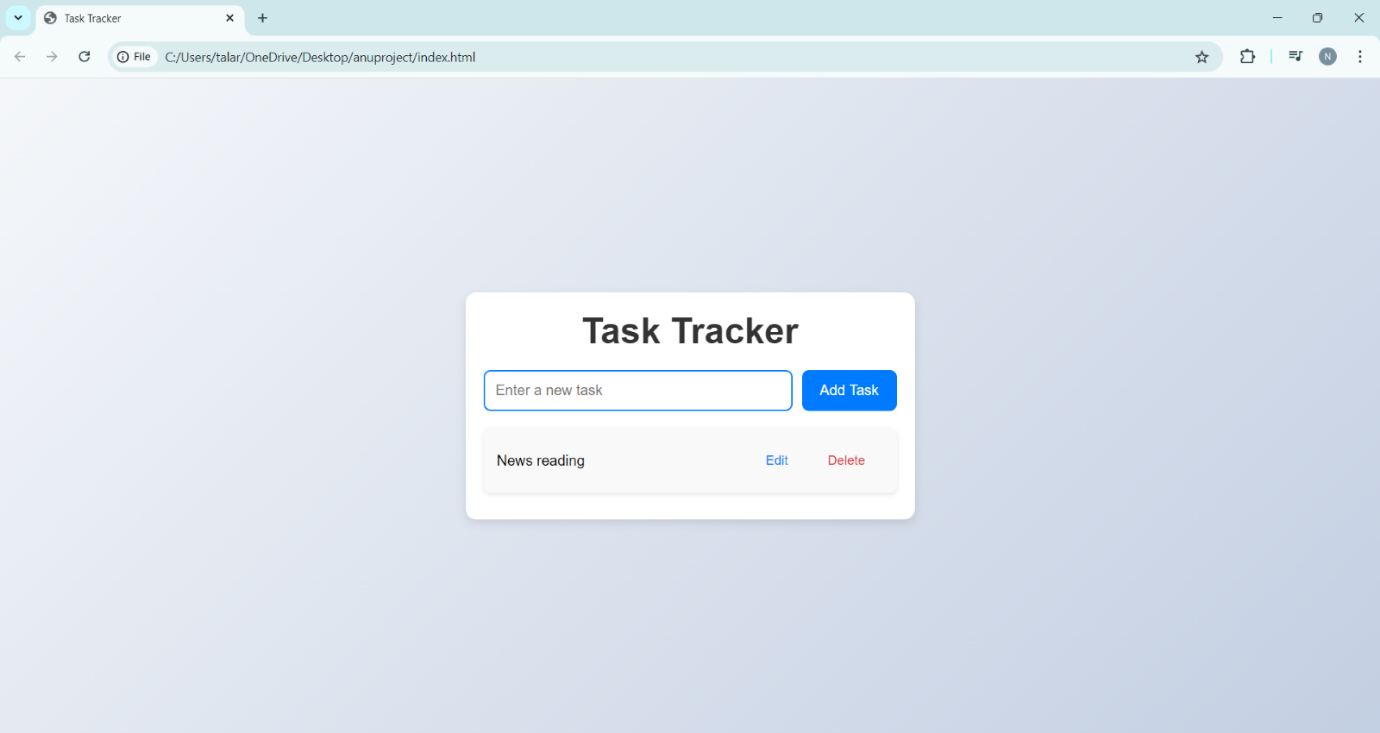
1. **Add a Task:**

- Enter a task in the input field and - click "Add Task".



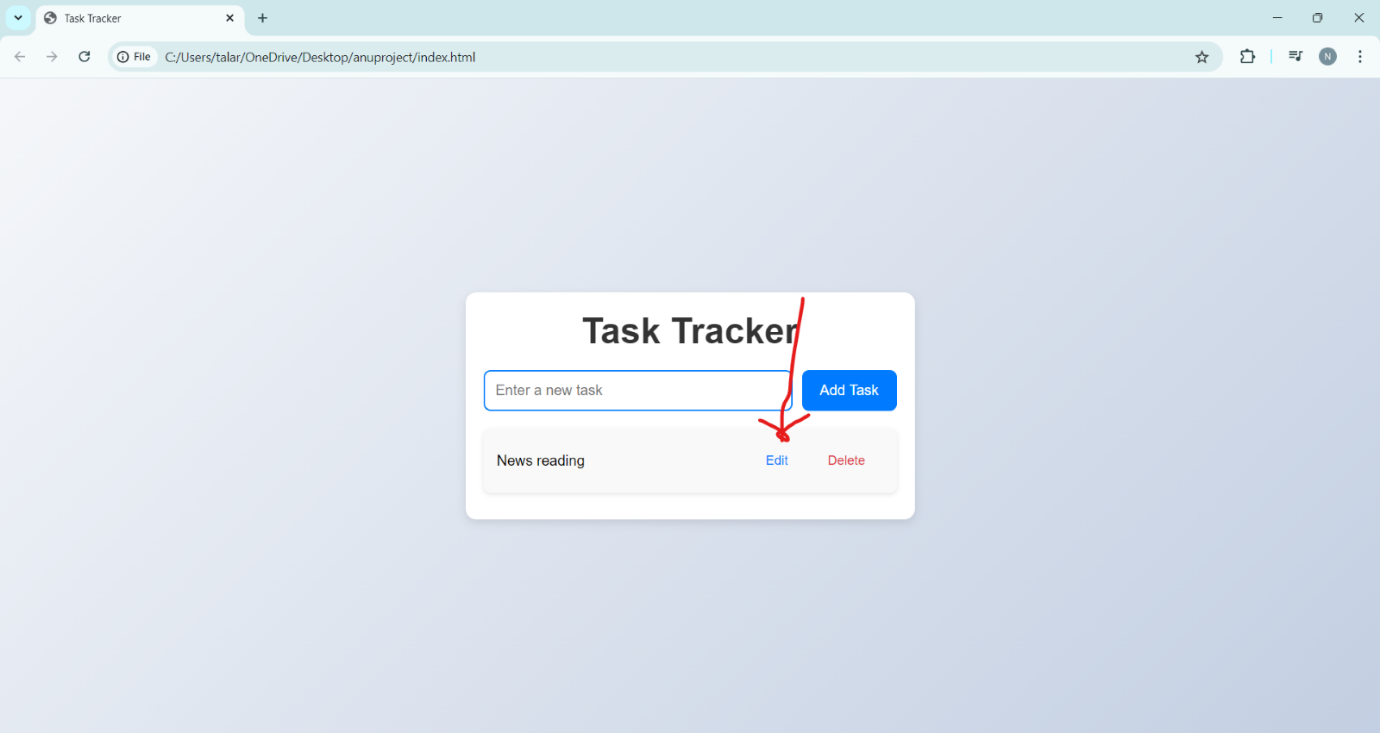


  - The task should appear in the list below.

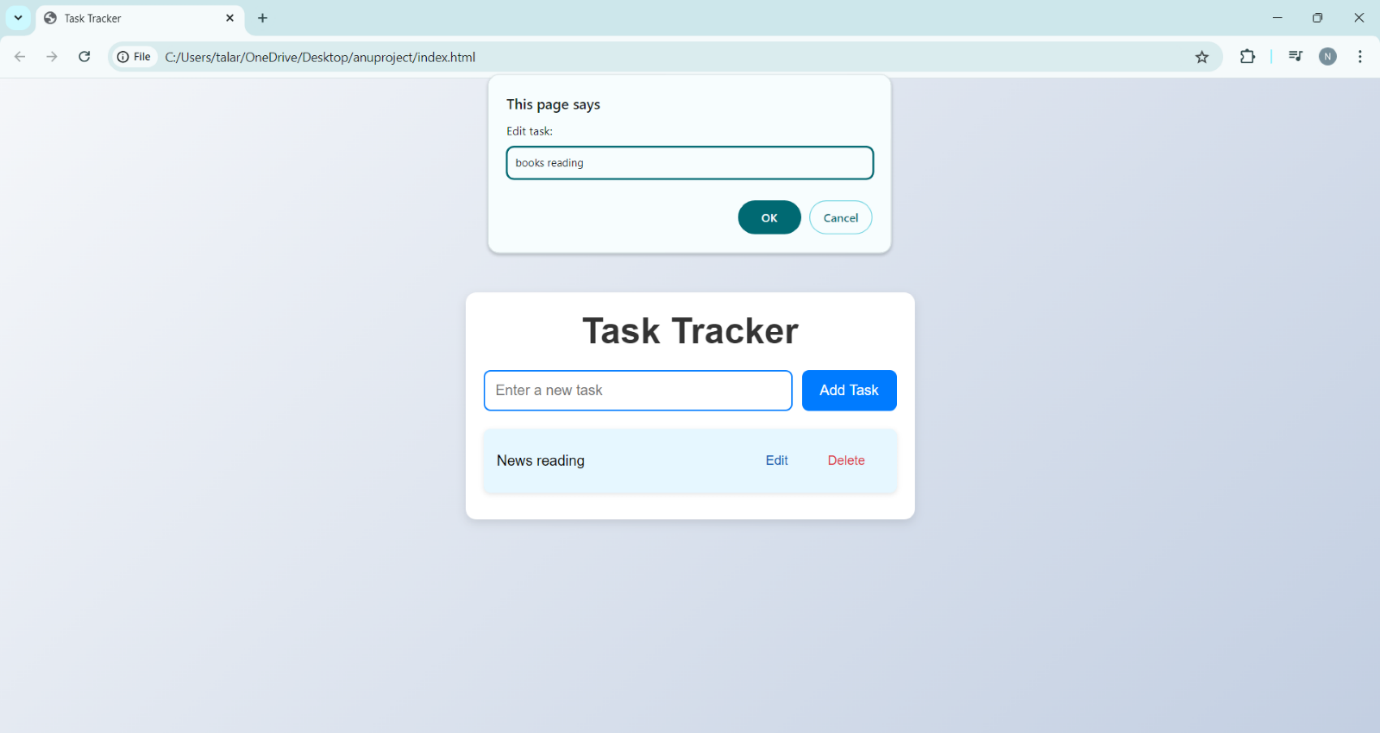


2. **Edit a Task:**

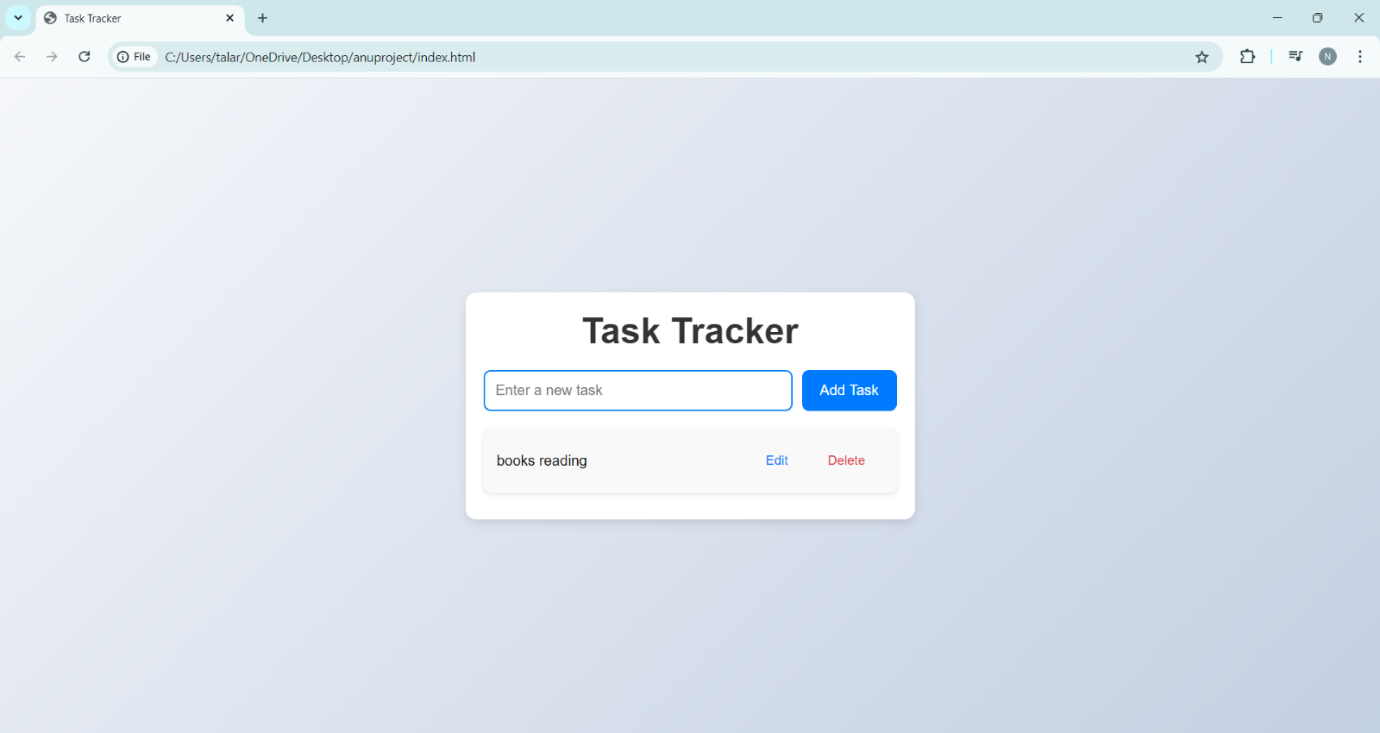
- Click the "Edit" button next to any task.



- Modify the task description in the prompt and save changes.

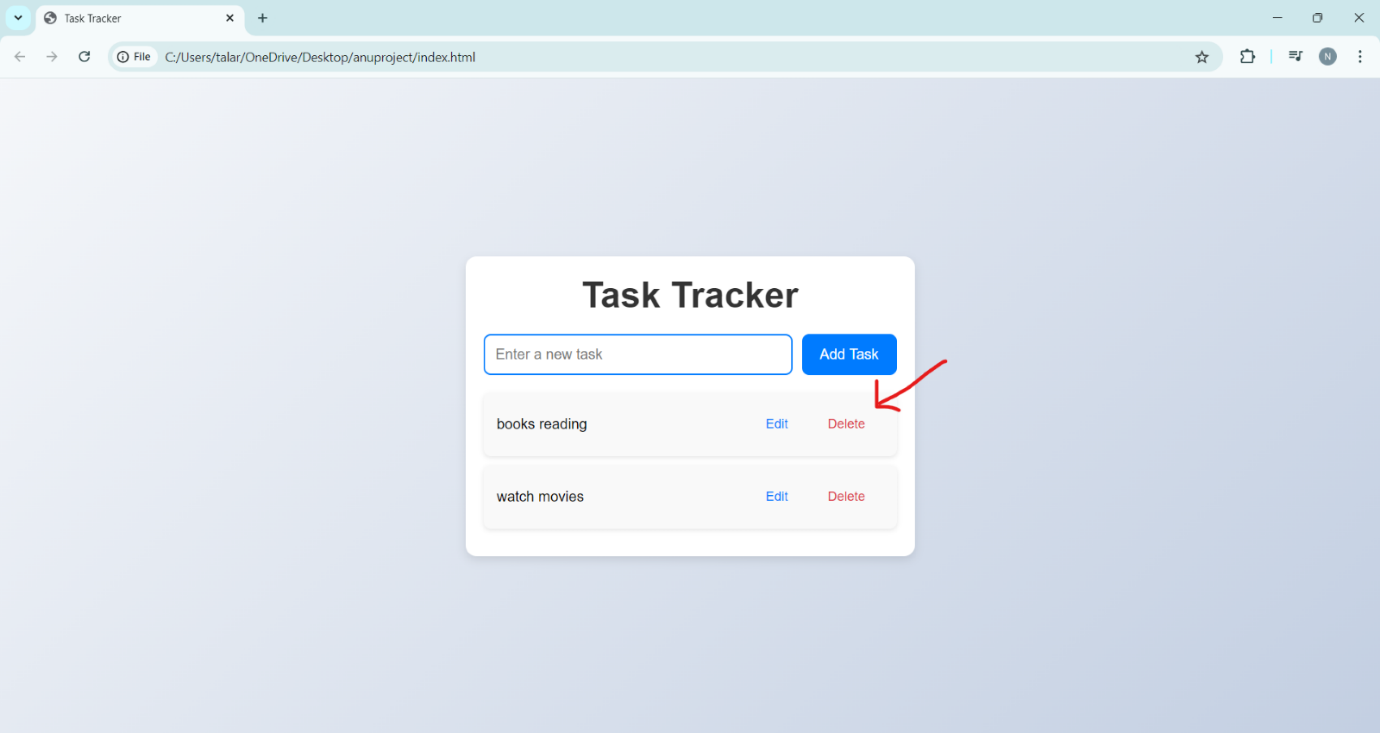


- The updated task should appear in the list.

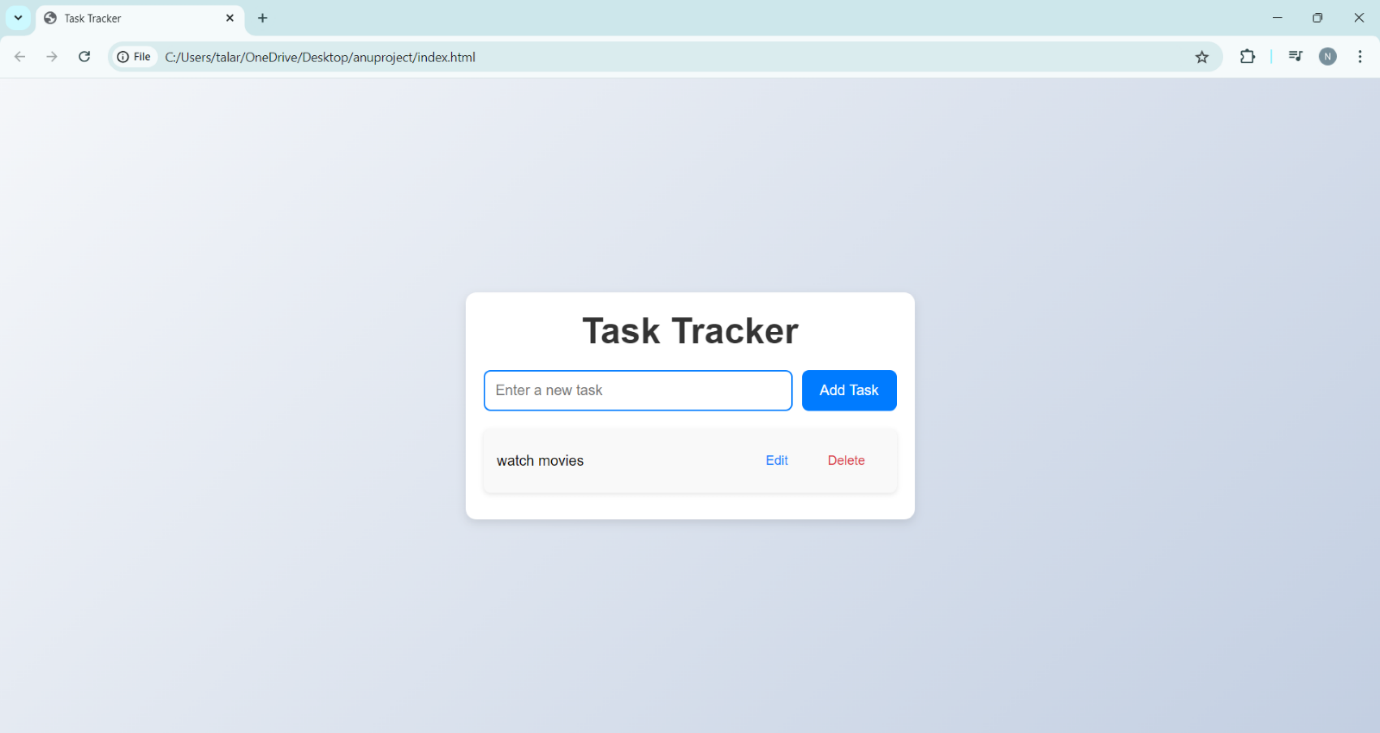


3. **Delete a Task:**

- Click the "Delete" button next to any task.

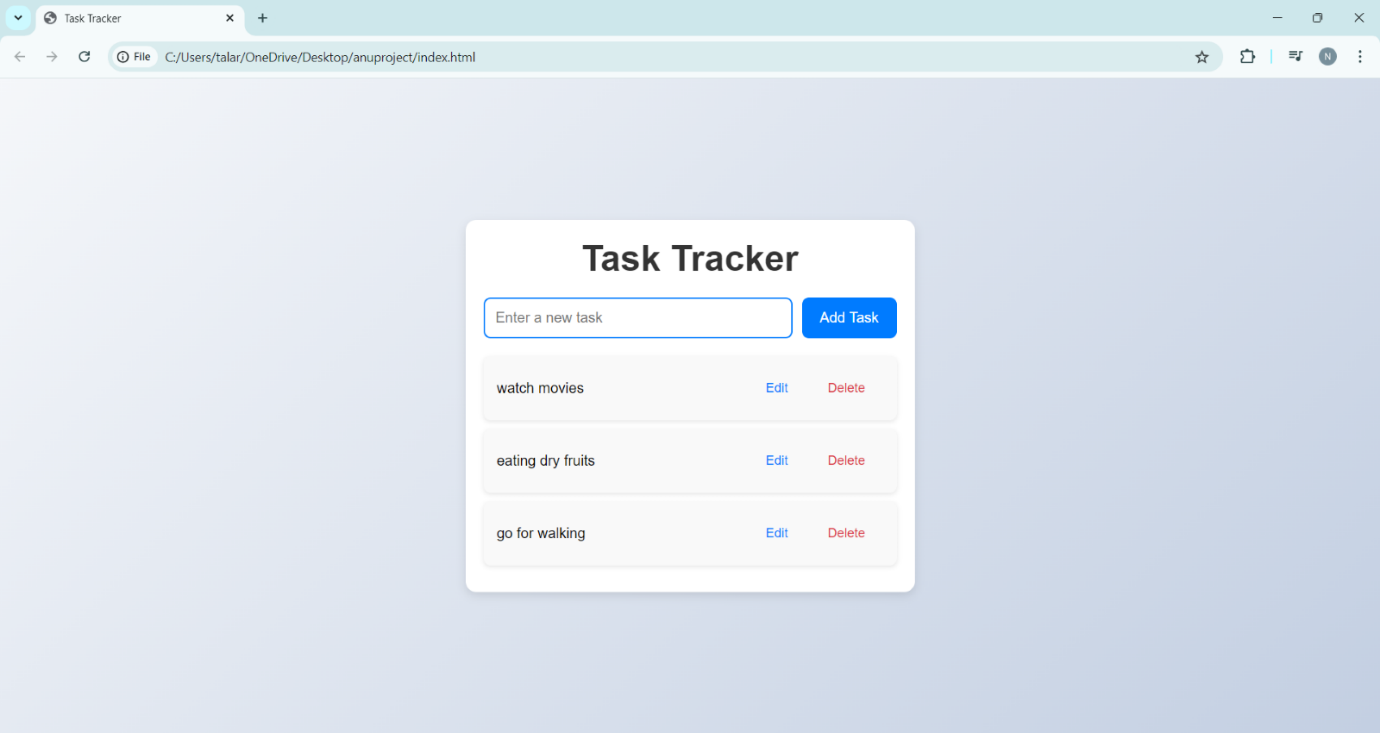


- The task should be removed from the list.



4. **Multiple Tasks:**

- Add multiple tasks and ensure all functionalities work correctly with multiple entries.



**Conclusion:**

The Task Tracker application provides a simple and effective way to manage tasks using basic web technologies. It covers all CRUD operations, allowing users to add, edit, delete, and display tasks efficiently. This application is an excellent starting point for learning how to build dynamic web applications using vanilla JavaScript, HTML, and CSS.

This documentation should provide a clear and concise overview of your Task Tracker application, making it easy for others to understand how it works and how to use it.