**LAPORAN PRAKTIKUM JAVASCRIPT**

“Tugas 04 –Aplikasi To-Do-List*”*

****

Oleh:

**Nama : Irmawati Pakpahan**

**NPM : 4523210054**

**Kelas : A**

**Tanggal : 1 Oktober 2024**

Dosen:

**Adi Wahyu Pribadi , S.Si., M.Kom**

**S1-Teknik Informatika**

**Fakultas Teknik Universitas Pancasila**

**2023/2024**

**BAB I**

**PENDAHULUAN**

* 1. **Penjelasan Tujuan Praktikum**
* Menggunakan JavaScript DOM untuk mengambil, memanipulasi, dan memperbarui elemen HTML.
* Mengimplementasikan JavaScript Class dan Object dalam aplikasi yang dinamis.
* Memanfaatkan JavaScript Function untuk mengatur logika aplikasi.
* Menerapkan event handling dalam interaksi pengguna dengan aplikasi.
  1. **Deskripsi Aplikasi To-Do-List**

Aplikasi manajemen daftar tugas yang interaktif menggunakan JavaScript. Aplikasi ini akan memungkinkan pengguna untuk menambahkan, menandai tugas sebagai selesai, dan menghapus tugas dari daftar. Fitur-fitur ini akan menguji pemahaman mahasiswa tentang manipulasi DOM, event handling, penggunaan class dan object, serta fungsi.

* 1. **Penjelasan Teori Dasar Praktikum**

JavaScript adalah bahasa pemrograman yang sangat berperan dalam perkembangan desain web modern. Sejak pertama kali diperkenalkan, JavaScript telah menjadi salah satu komponen utama untuk menciptakan situs web yang dinamis dan interaktif.

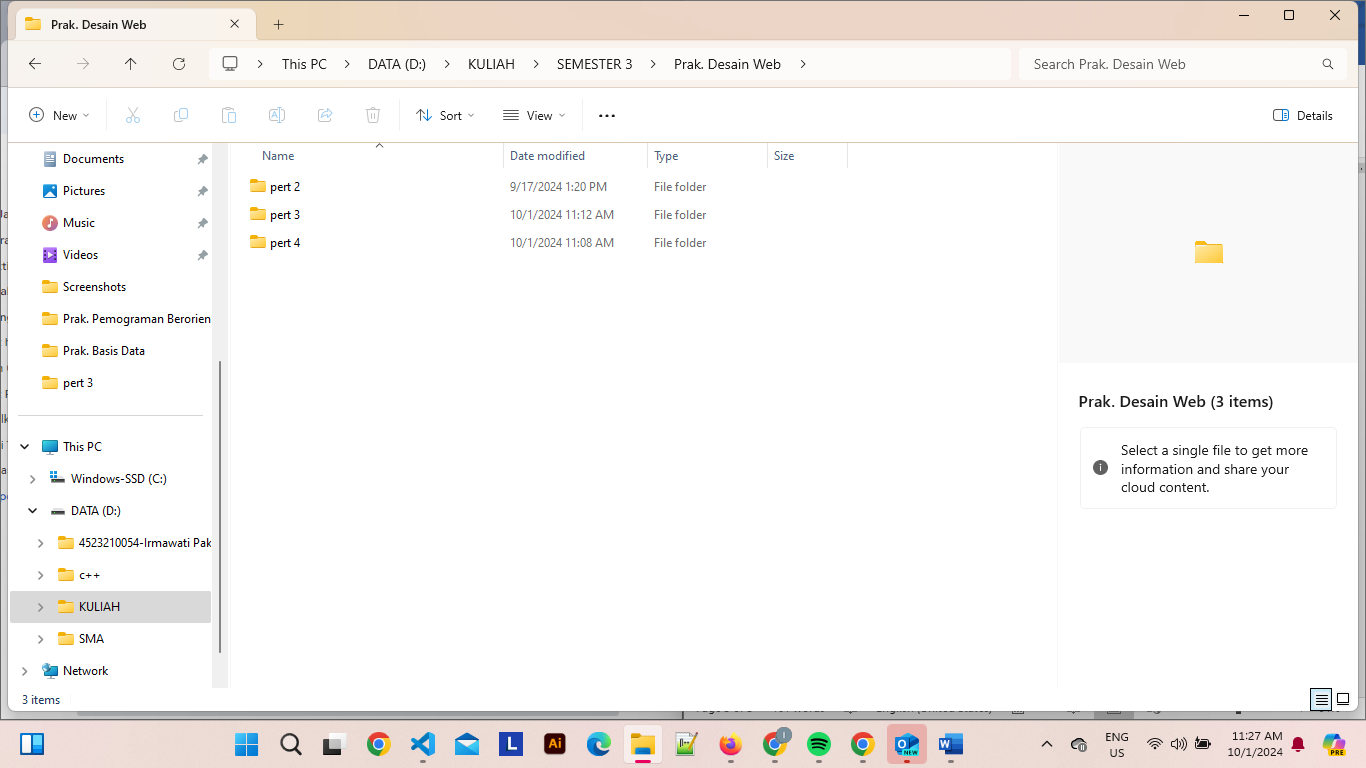
**BAB II**

**LANGKAH PENGERJAAN**

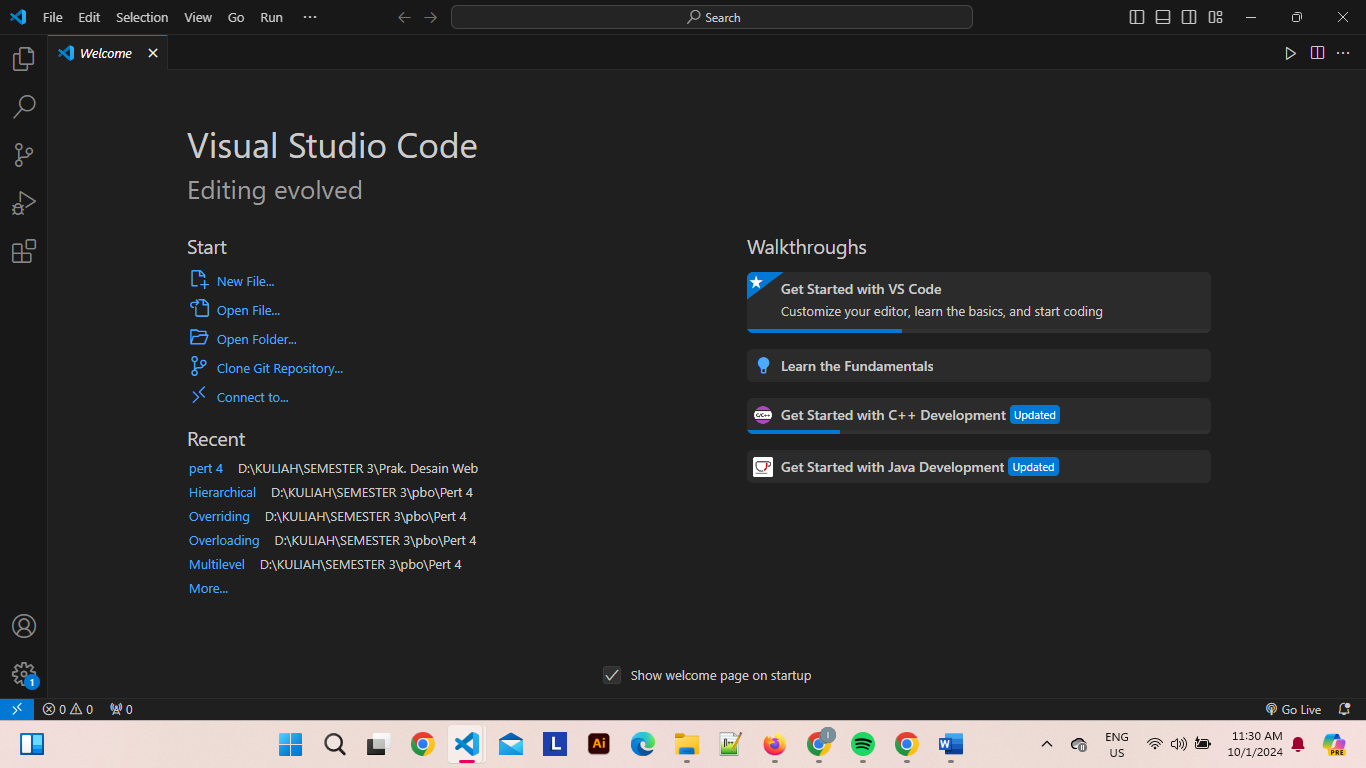
**2.1 Tahapan Pengerjaan**

**2.1.1 Membuat Folder File di File Explorer**

Dengan membuat Folder dapat memudahkan kita untuk mengelola data dengan rapih

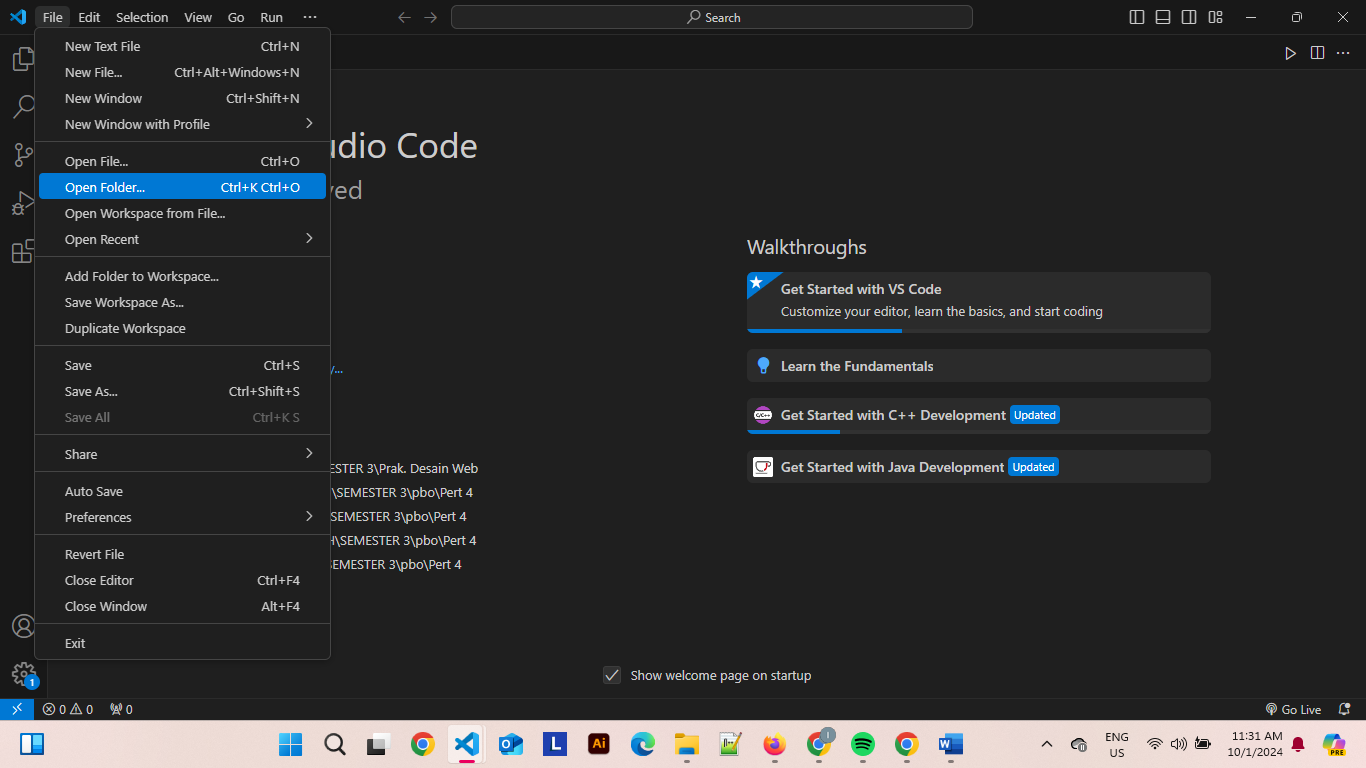
****

**2.1.2 Membuka Visual Code**

****

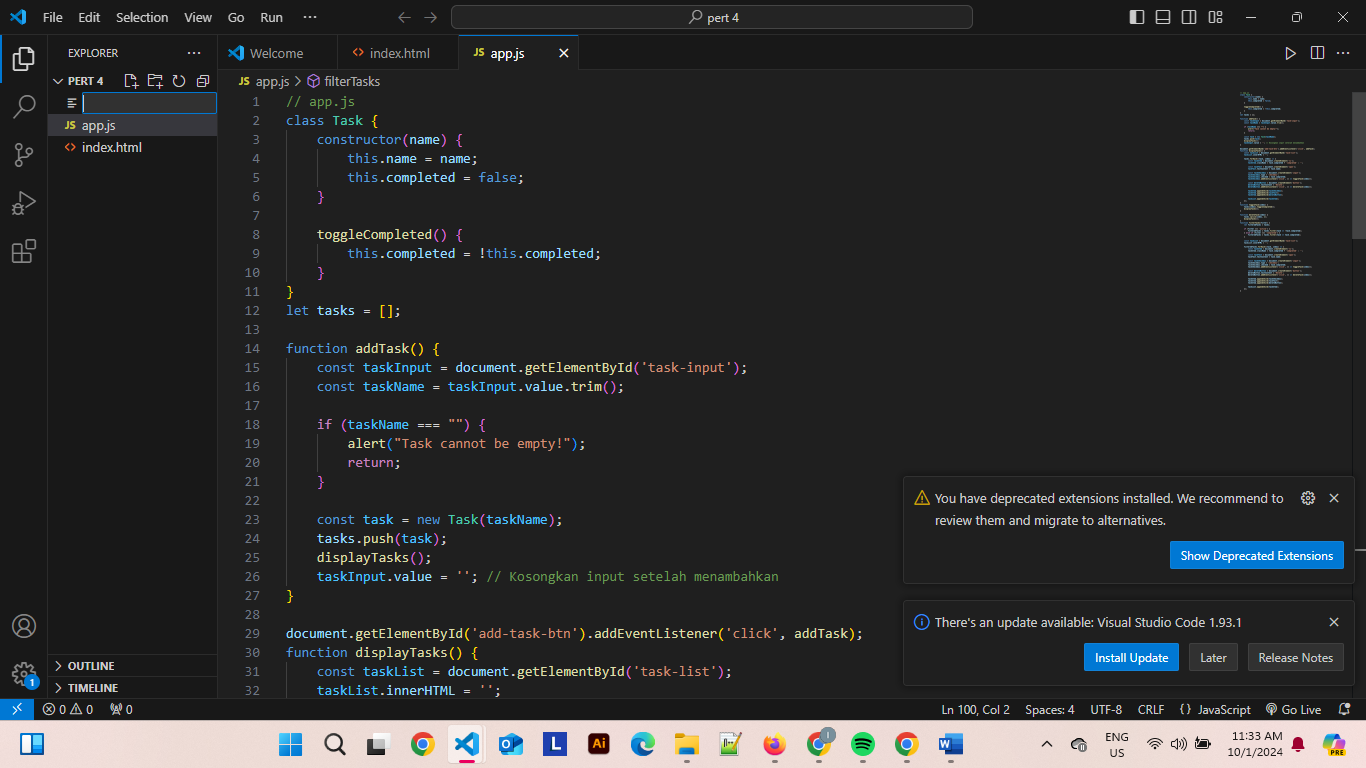
**2.1.3 Buka Folder yang telah disiapkan**

Klik file lalu klik open folder yang tujuannya untuk menaruh file praktikum kedalam folder

****

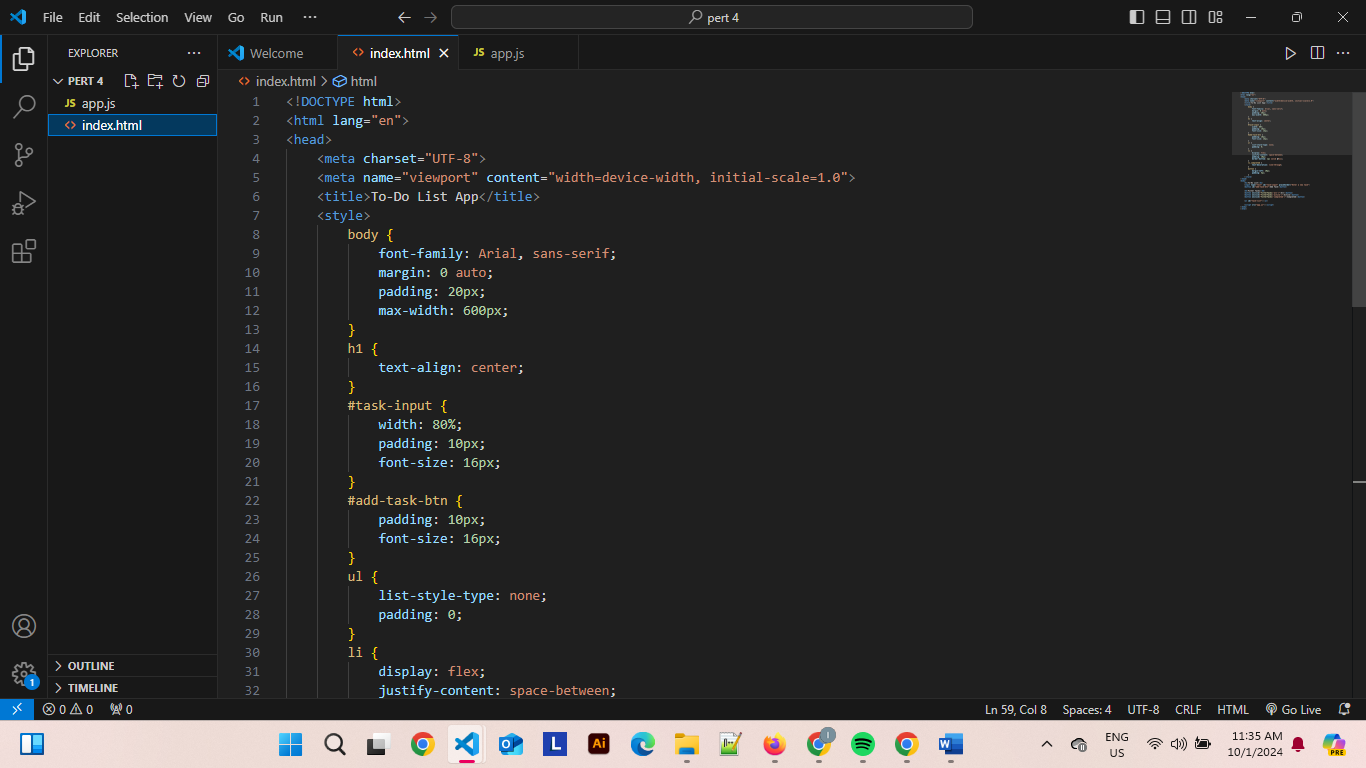
**2.1.4 Buat nama File**

Membuat nama file Index.html dan app.js untuk membuat aplikasi To-Do-List

****

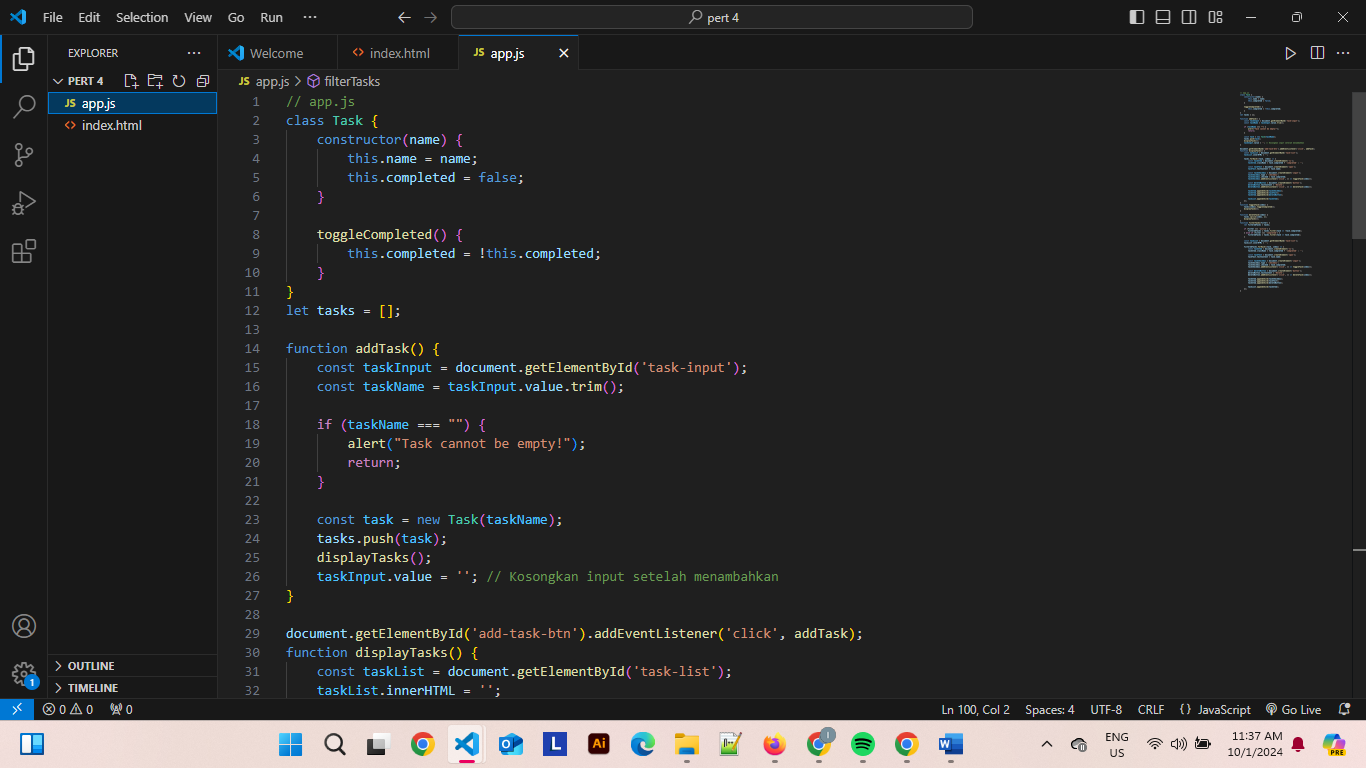
**2.1.5 Masukkan Source Code**

Masukkan Source Code yang ada kedalam index.html

****

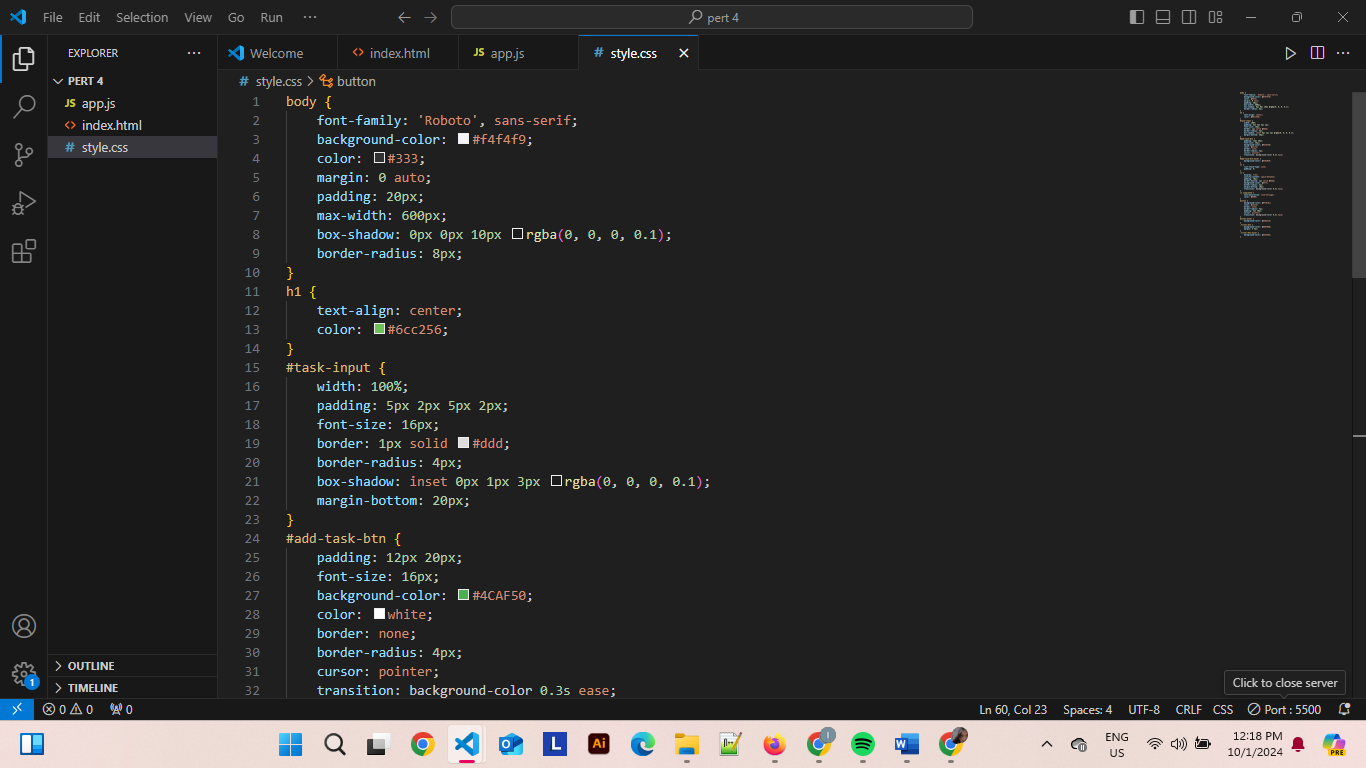
**2.1.6 Masukkan Source Code**

Masukkan Source Code JavaScript kedalam File app.js

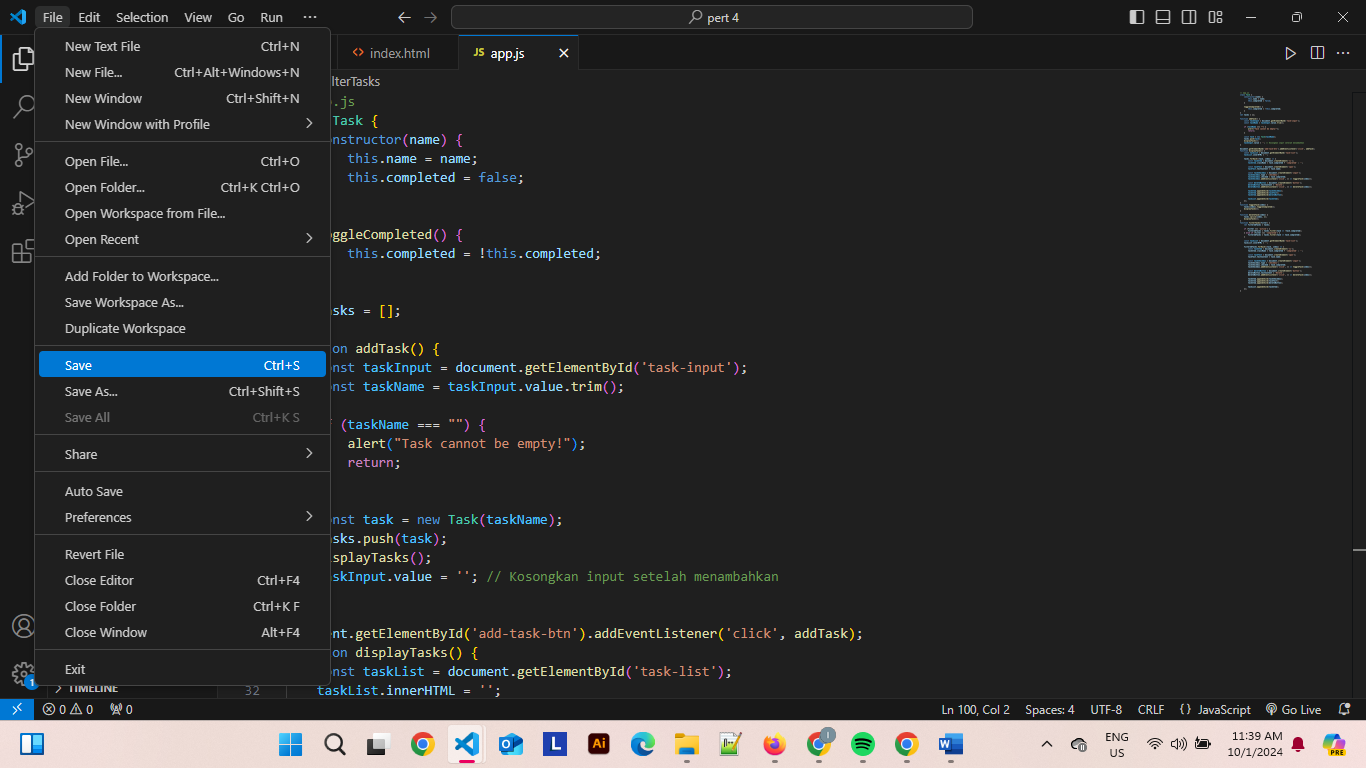
****

**2.1.7 Masukkan Source Code**

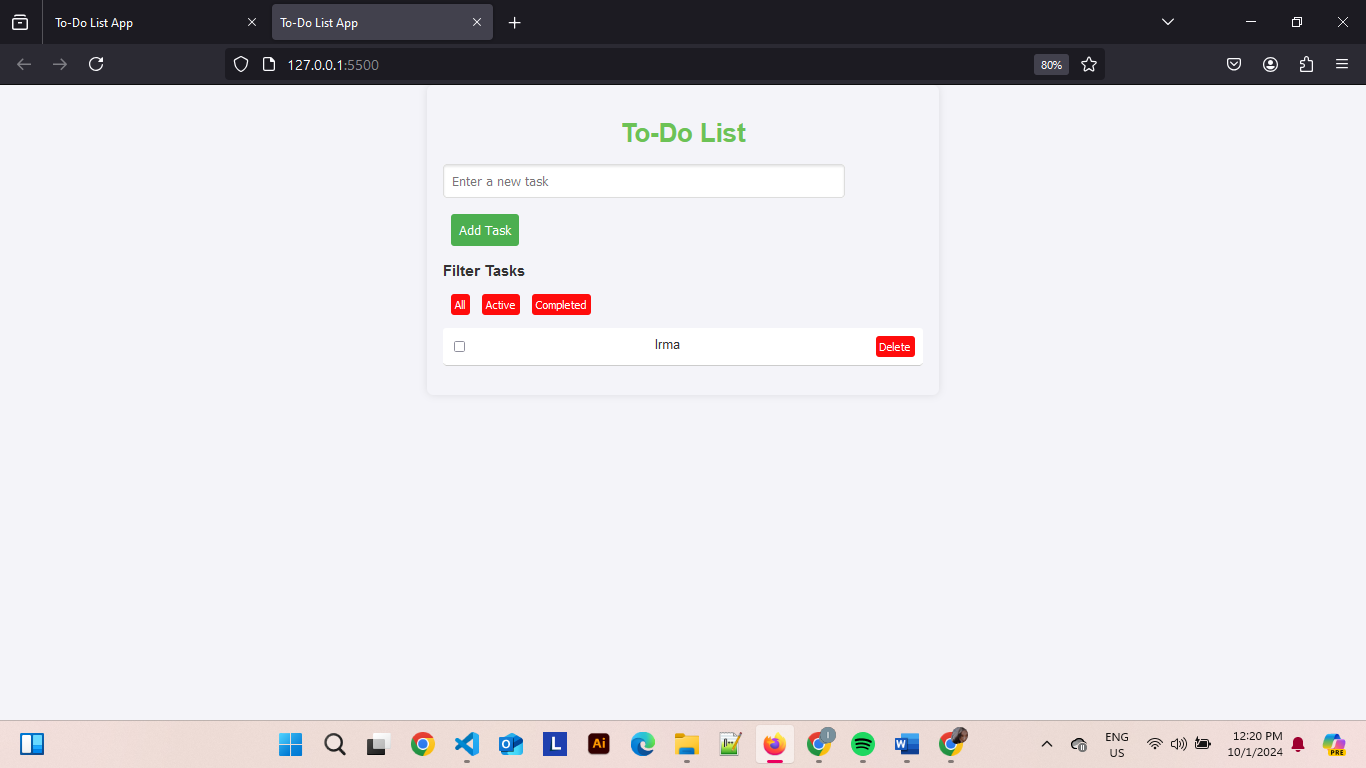
Masukkan Source Code Css kedalam style.css

****

**2.1.7 Save setiap File**

****

**2.1.8 Running**

****

**BAB III**

**STRUKTUR DAN PENJELASAN KODE**

**3.1 Source Code HTML Aplikasi To-Do-List**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>To-Do List App</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0 auto;

            padding: 20px;

            max-width: 600px;

        }

        h1 {

            text-align: center;

        }

        #task-input {

            width: 80%;

            padding: 10px;

            font-size: 16px;

        }

        #add-task-btn {

            padding: 10px;

            font-size: 16px;

        }

        ul {

            list-style-type: none;

            padding: 0;

        }

        li {

            display: flex;

            justify-content: space-between;

            padding: 10px;

            border-bottom: 1px solid #ccc;

        }

        li.completed {

            text-decoration: line-through;

        }

        button {

            margin-left: 10px;

            padding: 5px;

        }

    </style>

</head>

<body>

    <h1>To-Do List</h1>

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

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

    <h3>Filter Tasks</h3>

    <button onclick="filterTasks('all')">All</button>

    <button onclick="filterTasks('active')">Active</button>

    <button onclick="filterTasks('completed')">Completed</button>

    <ul id="task-list"></ul>

    <script src="app.js"></script>

</body>

</html>

**3.2 Source Code JavaScript Aplikasi To-Do-List**

// app.js

class Task {

    constructor(name) {

        this.name = name;

        this.completed = false;

    }

    toggleCompleted() {

        this.completed = !this.completed;

    }

}

let tasks = [];

function addTask() {

    const taskInput = document.getElementById('task-input');

    const taskName = taskInput.value.trim();

    if (taskName === "") {

        alert("Task cannot be empty!");

        return;

    }

    const task = new Task(taskName);

    tasks.push(task);

    displayTasks();

    taskInput.value = ''; // Kosongkan input setelah menambahkan

}

document.getElementById('add-task-btn').addEventListener('click', addTask);

function displayTasks() {

    const taskList = document.getElementById('task-list');

    taskList.innerHTML = '';

    tasks.forEach((task, index) => {

        const taskItem = document.createElement('li');

        taskItem.className = task.completed ? 'completed' : '';

        const taskText = document.createElement('span');

        taskText.textContent = task.name;

        const taskCheckbox = document.createElement('input');

        taskCheckbox.type = 'checkbox';

        taskCheckbox.checked = task.completed;

        taskCheckbox.addEventListener('click', () => toggleTask(index));

        const deleteButton = document.createElement('button');

        deleteButton.textContent = 'Delete';

        deleteButton.addEventListener('click', () => deleteTask(index));

        taskItem.appendChild(taskCheckbox);

        taskItem.appendChild(taskText);

        taskItem.appendChild(deleteButton);

        taskList.appendChild(taskItem);

    });

}

function toggleTask(index) {

    tasks[index].toggleCompleted();

    displayTasks();

}

function deleteTask(index) {

    tasks.splice(index, 1);

    displayTasks();

}

function filterTasks(filter) {

    let filteredTasks = tasks;

    if (filter === 'active') {

        filteredTasks = tasks.filter(task => !task.completed);

    } else if (filter === 'completed') {

        filteredTasks = tasks.filter(task => task.completed);

    }

    const taskList = document.getElementById('task-list');

    taskList.innerHTML = '';

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

        const taskItem = document.createElement('li');

        taskItem.className = task.completed ? 'completed' : '';

        const taskText = document.createElement('span');

        taskText.textContent = task.name;

        const taskCheckbox = document.createElement('input');

        taskCheckbox.type = 'checkbox';

        taskCheckbox.checked = task.completed;

        taskCheckbox.addEventListener('click', () => toggleTask(index));

        const deleteButton = document.createElement('button');

        deleteButton.textContent = 'Delete';

        deleteButton.addEventListener('click', () => deleteTask(index));

        taskItem.appendChild(taskCheckbox);

        taskItem.appendChild(taskText);

        taskItem.appendChild(deleteButton);

        taskList.appendChild(taskItem);

    });

}

**3.2 Fungsi Class Task**

Class Task berfungsi untuk merepresentasikan sebuah tugas yang memiliki nama (name) dan status penyelesaian (completed). Secara default, tugas belum selesai (completed = false), dan method toggleCompleted() digunakan untuk mengubah status penyelesaian tugas antara selesai dan belum selesai.

**3.3 Fungsi AddTask**

Fungsi addTask() digunakan untuk menambahkan tugas baru ke dalam daftar tugas (tasks). Fungsi ini mengambil nilai input dari elemen dengan ID task-input, memvalidasi apakah input tidak kosong, lalu membuat objek baru dari kelas Task. Objek tugas tersebut ditambahkan ke array tasks, ditampilkan menggunakan fungsi displayTasks(), dan input teks dikosongkan setelah tugas ditambahkan.

**3.4 Fungsi toggleTask**

Fungsi toggleTask(index) digunakan untuk mengubah status penyelesaian (completed) dari tugas pada indeks tertentu dalam array tasks. Fungsi ini memanggil method toggleCompleted() pada tugas yang dipilih, lalu memperbarui tampilan tugas dengan memanggil displayTasks().

**3.5 Fungsi deleteTask**

Fungsi deleteTask(index) digunakan untuk menghapus tugas pada indeks tertentu dalam array tasks. Fungsi ini menggunakan metode splice() untuk menghapus tugas dari array, lalu memperbarui tampilan daftar tugas dengan memanggil displayTasks().

**3.6 Fungsi filterTask**

Fungsi filterTasks(filter) digunakan untuk memfilter dan menampilkan tugas berdasarkan status penyelesaiannya. Jika parameter filter adalah 'active', fungsi ini hanya menampilkan tugas yang belum selesai. Jika filter adalah 'completed', hanya tugas yang selesai yang ditampilkan. Jika tidak ada filter, semua tugas ditampilkan. Fungsi ini juga memperbarui tampilan daftar tugas (task-list) sesuai dengan hasil filter tersebut.

**3.7 Cara Penggunaan Manipulasi DOM**

Manipulasi DOM untuk Menampilkan Tugas

displayTasks():

Mengosongkan elemen daftar tugas (task-list), kemudian membuat elemen <li> untuk setiap tugas.

Menambahkan elemen checkbox, teks tugas, dan tombol hapus ke dalam <li>, dan menambahkannya ke dalam daftar.

**3.8 Penjelasan Event Handling**

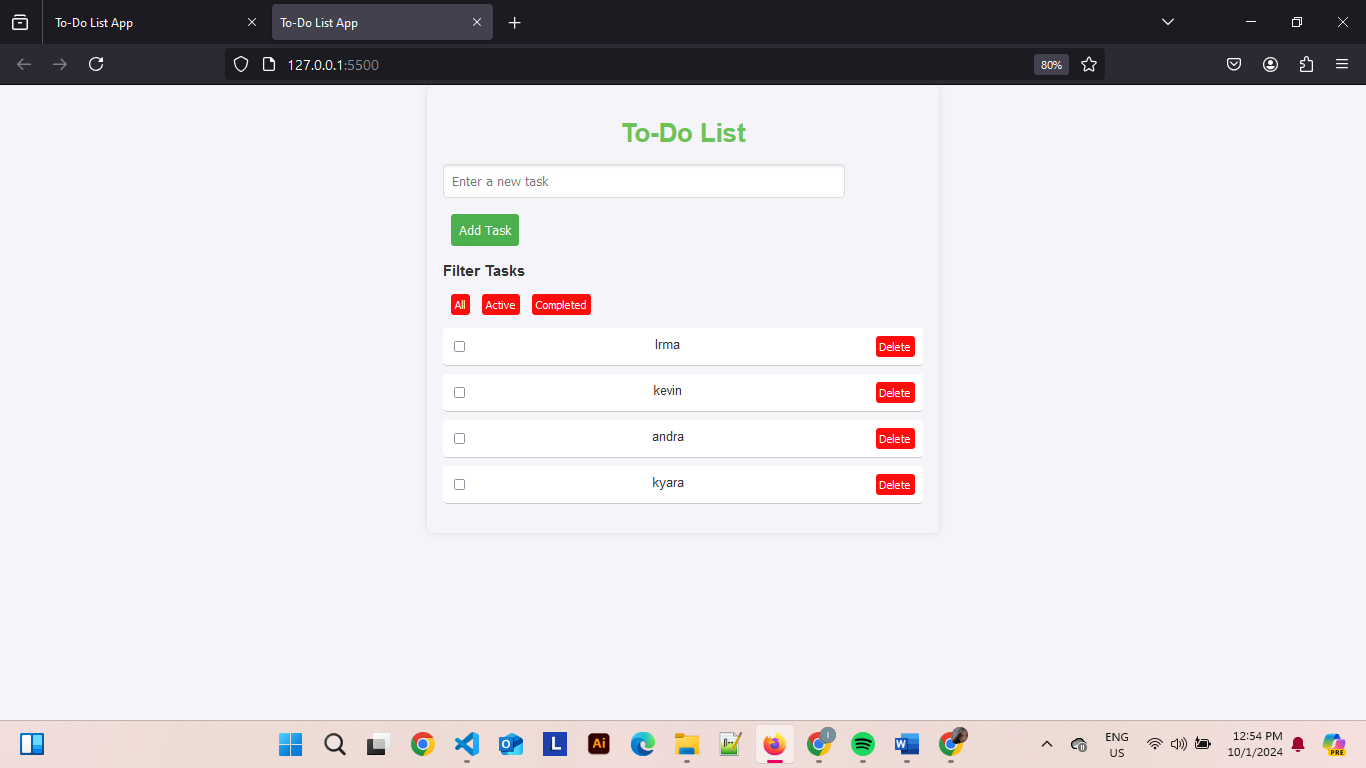
Tombol “Add Task”:

Menggunakan addEventListener untuk menambahkan event click pada tombol. Ketika tombol diklik, fungsi addTask() akan dipanggil.

Checkbox:

Setiap checkbox memiliki event listener yang memanggil toggleTask(index) ketika dicentang atau dicopot. Ini memungkinkan pengguna untuk menandai tugas sebagai selesai atau tidak.

**3.7 Hasil Uji Coba**

****

* **Kolom ”enter a new task”**

memungkinkan pengguna untuk mengetik dan menambahkan tugas baru ke dalam daftar tugas. Ketika pengguna memasukkan nama tugas dan menekan tombol, tugas tersebut akan ditambahkan ke daftar dan ditampilkan pada antarmuka.

* **Fitur Add Task**

Fitur **add task** berfungsi untuk menambahkan tugas baru ke dalam daftar tugas, memungkinkan pengguna mencatat dan mengelola item tugas yang ingin diselesaikan.

* **Fitur All**

Fitur **"all"** berfungsi untuk menampilkan **semua tugas** dalam daftar, baik yang sudah selesai maupun yang belum selesai, tanpa memfilter berdasarkan status penyelesaiannya.

* **Fitur Active**

Fitur **active** berfungsi untuk menampilkan hanya tugas-tugas yang belum selesai (tugas aktif) dalam daftar tugas.

* **Fitur Completed**

Fitur completed berfungsi untuk menandai apakah sebuah tugas sudah selesai atau belum. Jika completed bernilai true, tugas dianggap selesai; jika false, tugas masih aktif atau belum diselesaikan. Fitur ini memungkinkan pengguna untuk memantau status penyelesaian tugas dalam daftar

* **Fitur Delete**

Fitur delete berfungsi untuk menghapus item (seperti tugas) dari daftar, sehingga item tersebut tidak lagi muncul atau disimpan di sistem. Ini berguna untuk mengelola daftar dengan menghilangkan entri yang tidak diperlukan.

**BAB IV**

**PENUTUP**

**KESIMPULAN**

aplikasi ini memungkinkan pengguna untuk mengelola daftar tugas dengan cara yang sederhana dan interaktif. Pengguna dapat menambahkan tugas baru, menandai tugas yang sudah selesai, menghapus tugas, serta memfilter tugas berdasarkan status (aktif atau selesai). Melalui penggunaan fungsi-fungsi seperti addTask(), toggleTask(), deleteTask(), dan filterTasks(), aplikasi ini menawarkan fitur dasar yang diperlukan untuk memantau dan mengorganisir aktivitas harian secara efektif. JavaScript memungkinkan aplikasi ini berjalan secara dinamis di sisi klien, sehingga perubahan pada daftar tugas dapat langsung terlihat tanpa memuat ulang halaman.