Laporan Praktikum JavaScript – Aplikasi To-Do List

Oleh:
Nama : Zidan wali arubusman
NPM : 4523210001
Mata Kuliah : Praktikum Desain Web
Tanggal : 1 - 10 - 2024

Dosen:
Adi Wahyu Pribadi , S.Si., M.Kom
S1-Teknik Informatika
Universitas Pancasila
2024/2025

A.) Pendahuluan

- Tujuan singkat Dari Praktikum : Praktikum ini bertujuan untuk melatih peserta dalam penggunaan JavaScript DOM dalam mengambil, memanipulasi, dan memperbarui elemen-elemen HTML pada halaman web. Selain itu, peserta akan belajar mengimplementasikan *JavaScript Class* dan *Object* untuk membangun aplikasi yang dinamis dan interaktif.

- Deskripsi Singkat Dari Aplikasi To do List : Aplikasi To do List yang akan dibuat kali ini aplikasi manajement yang berfungsi untuk menambahkan tugas , menghapus tugas , dan memfilter tugas selesai dan belum selesai.

- Penjelasan Teori Dasar :
  - Manipulasi DOM (Document Object Model)
    Aplikasi ini memanfaatkan DOM untuk mengakses dan mengubah elemen-elemen HTML seperti input dan daftar tugas secara dinamis. JavaScript digunakan untuk memodifikasi struktur halaman, misalnya menampilkan tugas yang ditambahkan oleh pengguna dalam elemen daftar (ul dan li).

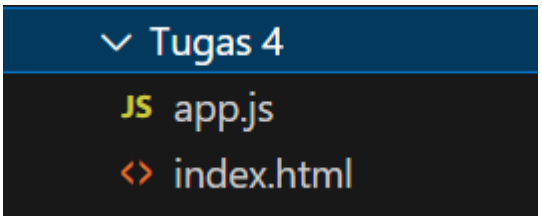
  - Class dan Object
    Setiap tugas yang dibuat di dalam aplikasi diwakili oleh objek yang berasal dari class Task. Objek ini memiliki properti name untuk menyimpan nama tugas dan completed untuk mencatat apakah tugas sudah selesai. Metode toggleCompleted berfungsi untuk mengubah status penyelesaian tugas.

  - Fungsi JavaScript
    Berbagai fungsi digunakan untuk menangani logika aplikasi, seperti menambahkan tugas baru dengan fungsi addTask, menghapus tugas menggunakan deleteTask, dan mengubah status tugas melalui toggleTask. Fungsi-fungsi ini memastikan interaksi pengguna dapat dijalankan dengan lancar.

  - Penanganan Event (Event Handling)
    Setiap interaksi pengguna, seperti menekan tombol "Add Task" untuk menambah tugas atau mencentang kotak untuk menandai tugas sebagai selesai, diatur dengan event handling. Fungsi addEventListener digunakan untuk merespons tindakan-tindakan.

B.) Langkah Pengerjaan

1) **Pertama Buat Folder baru dan isi folder baru tersebut itu dengan file html yang sama dengan dibawah ini :**




```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <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>
```

2) **Membuat file script.js untuk membuat Class dan Object untuk Tugas**

```
// app.js
class Task {
    constructor(name) {
        this.name = name;
        this.completed = false;
    }

    toggleCompleted() {
        this.completed = !this.completed;
    }
}
```

- class untuk merepresentasikan tugas yang memiliki properti name dan completed
- Task memiliki atribut name dan completed
- toggleCompleted digunakan untuk menandai tugas sebagai selesai atau belum

3) Membuat Fungsi untuk Menambah Tugas Dan Menampilkan Tugas di DOM

Source code :

```
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);
    });
}
```

4) Menandai Tugas Selesai dan Menghapus Tugas

```
function toggleTask(index) {
    tasks[index].toggleCompleted();
    displayTasks();
}

function deleteTask(index) {
    tasks.splice(index, 1);
    displayTasks();
}
```

5) Membuat Filter Tugas Berdasarkan Status

```
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);
    });
}
```

C.) **Struktur Dan Penjelasan Code**

- Sertakan kode JavaScript lengkap yang digunakan dalam 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);
    });
}
```

- Jelaskan secara singkat setiap bagian dari kode:

- **Class Task dan fungsinya.**
  Class Task adalah sebuah blueprint atau cetak biru untuk membuat objek tugas (task). Setiap objek Task memiliki dua properti: name, yang menyimpan nama tugas, dan completed, yang menyimpan status apakah tugas sudah selesai atau belum (default-nya false). Method toggleCompleted digunakan untuk mengubah status completed dari false menjadi true atau sebaliknya setiap kali dipanggil, memungkinkan pengguna untuk menandai tugas sebagai selesai atau belum selesai.

- **Fungsi addTask, toggleTask, deleteTask, dan filterTasks.**

    - **addTask**: Fungsi ini digunakan untuk menambahkan tugas baru ke dalam daftar tugas (tasks). Ia mengambil nilai dari input teks, memeriksa apakah input kosong atau tidak, dan jika tidak kosong, ia membuat objek Task baru dan menambahkannya ke array tasks. Setelah itu, ia menampilkan kembali daftar tugas dan mengosongkan input.

- **toggleTask**: Fungsi ini digunakan untuk mengubah status completed dari tugas tertentu. Saat dipanggil, ia membalik status completed dari tugas berdasarkan indeksnya di array tasks dan menampilkan ulang daftar tugas.

- **deleteTask**: Fungsi ini digunakan untuk menghapus tugas dari daftar. Ia menghapus tugas dari array tasks berdasarkan indeksnya, kemudian memperbarui tampilan daftar tugas.

- **filterTasks**: Fungsi ini digunakan untuk menyaring dan menampilkan tugas berdasarkan statusnya (semua, aktif, atau selesai). Tugas yang disaring ditampilkan kembali sesuai dengan kriteria filter yang diberikan (aktif: tugas yang belum selesai, selesai: tugas yang sudah selesai).

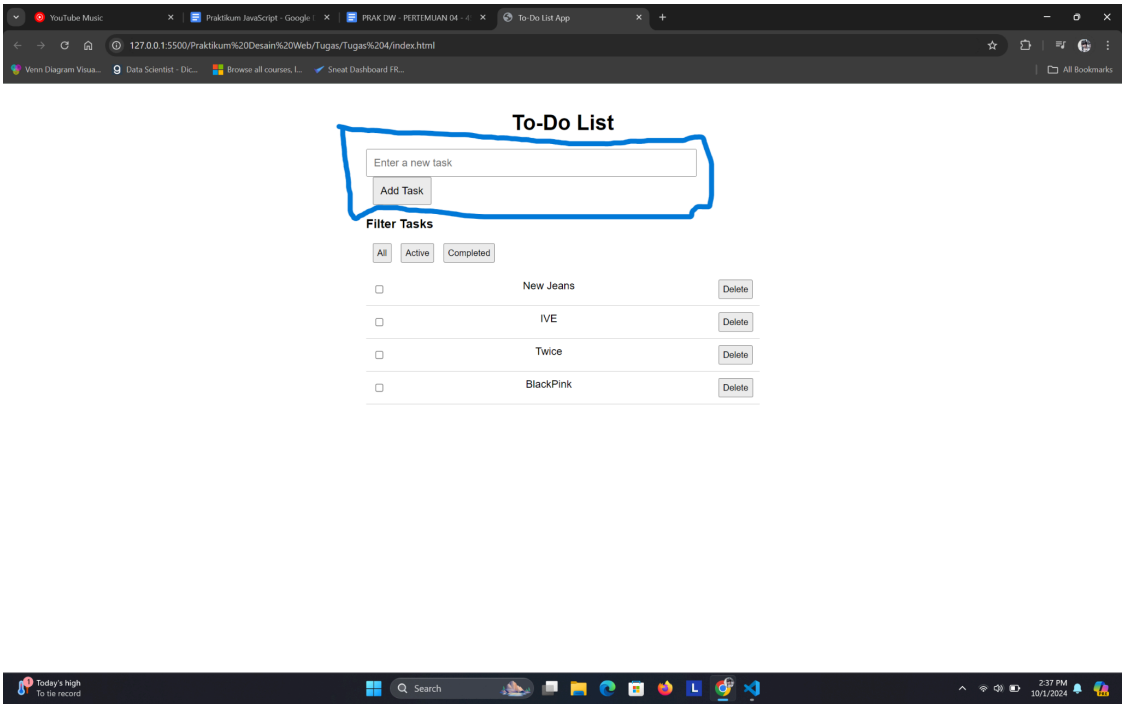
- **Cara penggunaan manipulasi DOM untuk menampilkan tugas.**

Manipulasi DOM dilakukan menggunakan fungsi displayTasks dan di dalam filterTasks. Dalam fungsi ini, daftar tugas diambil dari array tasks dan di-loop menggunakan forEach. Setiap tugas dibuatkan elemen HTML <li> baru. Di dalam elemen tersebut terdapat elemen <input> untuk checkbox (menandakan status selesai atau belum) dan <button> untuk menghapus tugas. Semua elemen ini kemudian di-append ke dalam elemen HTML <ul> dengan id task-list. DOM diperbarui secara dinamis setiap kali ada perubahan dalam daftar tugas, seperti penambahan, penghapusan, atau perubahan status tugas.

- **Penjelasan event handling yang digunakan, seperti pada tombol “Add Task” dan checkbox.**
  - **Tombol "Add Task"**: Event handling diterapkan pada tombol "Add Task" menggunakan metode addEventListener. Ketika tombol ini diklik, fungsi addTask akan dipanggil. Ini memungkinkan pengguna untuk menambahkan tugas baru ke daftar tugas.
  - **Checkbox**: Event handling juga diterapkan pada setiap checkbox yang ada di daftar tugas. Setiap kali checkbox diklik, event ini memicu fungsi toggleTask, yang akan mengubah status tugas apakah sudah selesai atau belum. Hal ini membuat interaksi pengguna lebih dinamis karena daftar tugas langsung diperbarui secara visual.
  - **Tombol "Delete"**: Setiap tugas juga memiliki tombol "Delete" dengan event listener. Saat tombol ini diklik, tugas akan dihapus dari daftar, dan daftar tugas diperbarui secara otomatis.

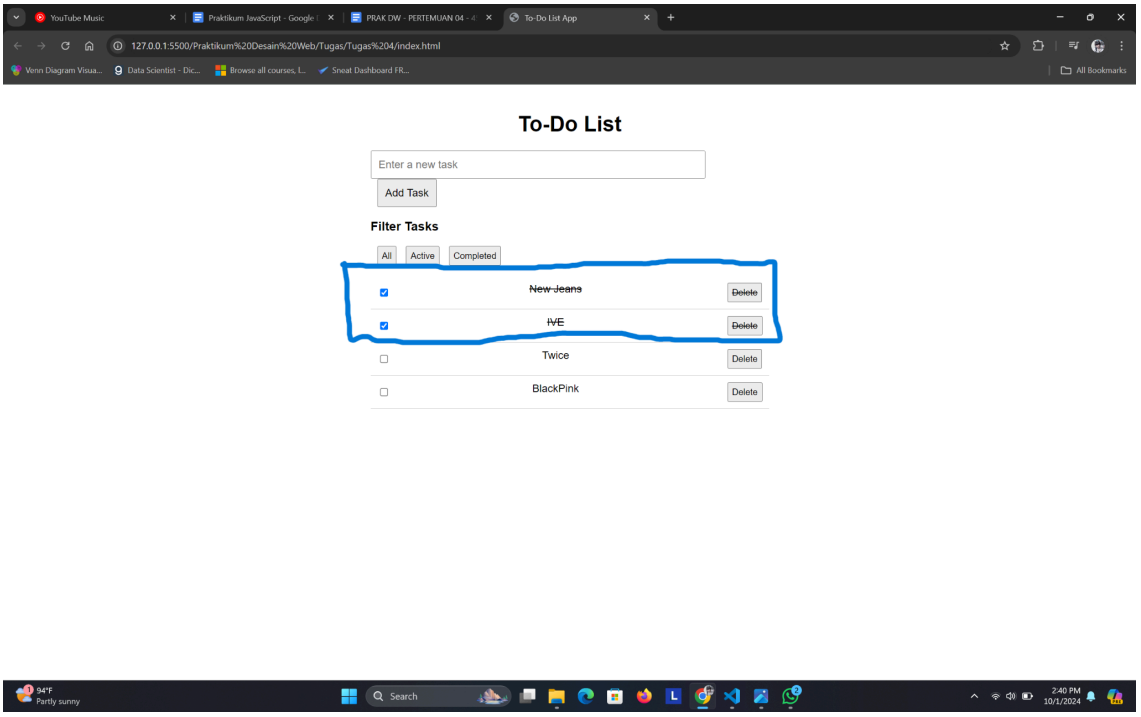
D. Hasil Uji Coba

- **Add Task**



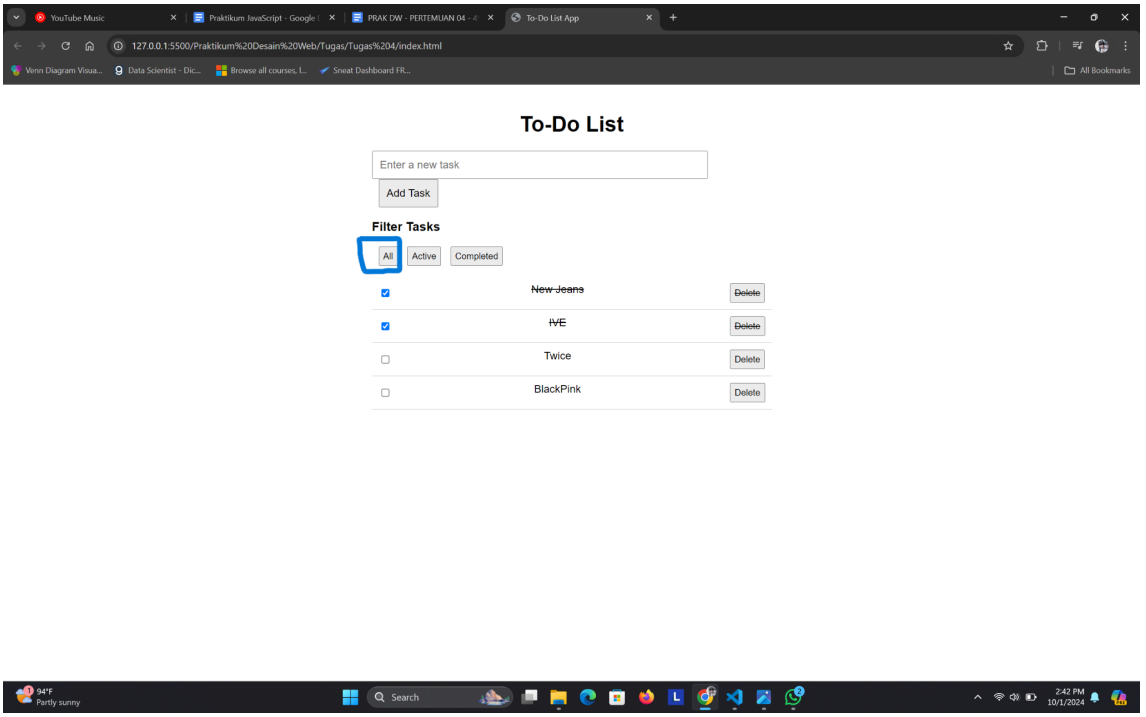
Pengguna dapat menambahkan task melalui kolom kemudian klik tombol **Add Task**

- **Completed Task**



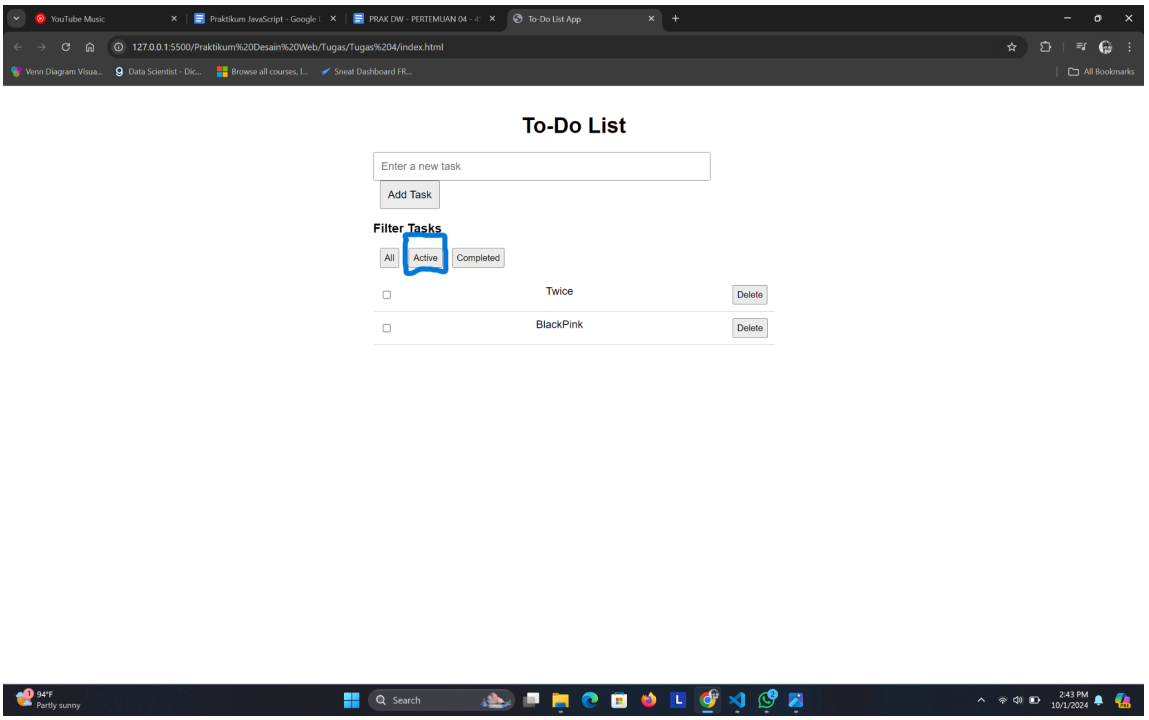
Pengguna dapat mencoret Task jika task selesai dengan cara menceklis checkbox yang ada di samping Task

- **Filter All Task**



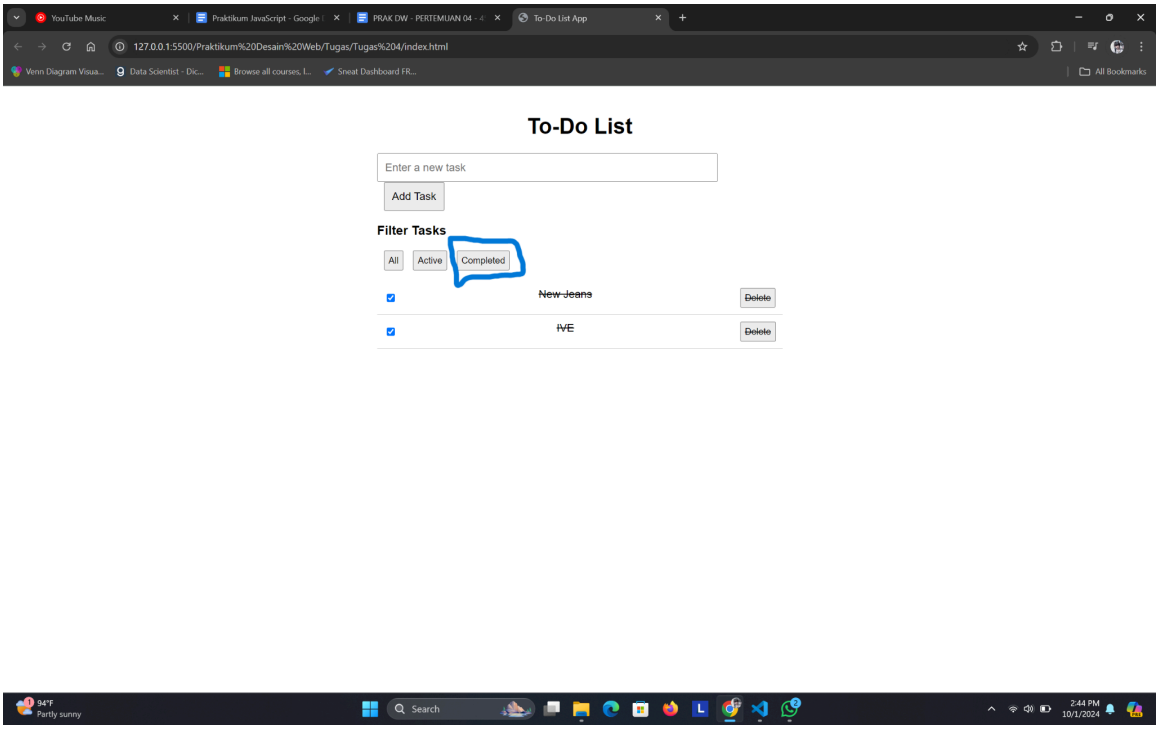
Pengguna dapat melihat semua Task yang ada

- **Filter Task Active**



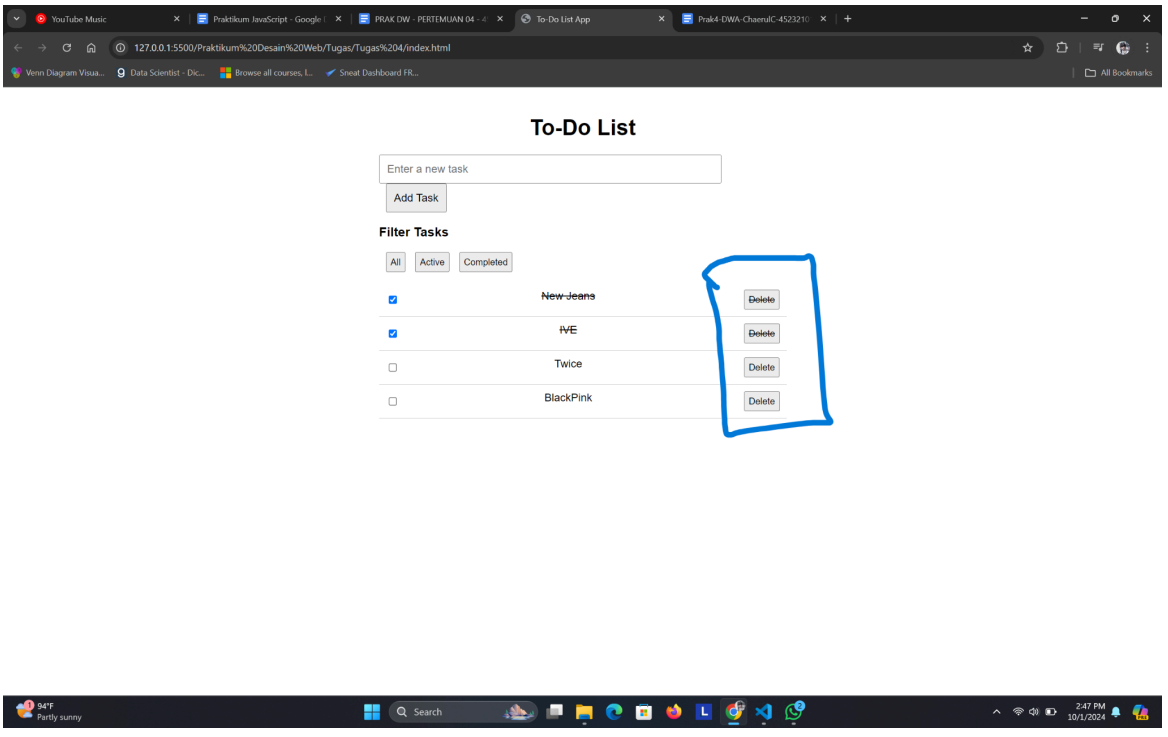
pengguna dapat melihat semua Task yang active

- **Filter Task Completed**



Pengguna dapat melihat semua Task yang completed

- **Delete Task**



Pengguna dapat menghapus Task

E.) Kesimpulan

Berdasarkan hasil praktikum yang telah dilaksanakan, mahasiswa mampu mengembangkan aplikasi To-Do List dengan memanfaatkan JavaScript DOM untuk mengambil, memanipulasi, dan memperbarui elemen-elemen HTML. Selanjutnya, mahasiswa merepresentasikan setiap tugas dengan properti name dan completed. Selain itu, mahasiswa dapat menggunakan JavaScript Function untuk mengatur logika aplikasi serta menerapkan penanganan peristiwa dalam interaksi pengguna dengan aplikasi. Dengan demikian, pengguna dapat menambahkan tugas, yang kemudian akan ditampilkan di DOM, menandai tugas yang telah diselesaikan, serta menghapus tugas yang tidak diperlukan

Link Github : https://github.com/Kaze212/Desain-Web-Pak-Adi-.git