**“**Computer Networks**”**

**Instructor:** Ali Hassan

**Submitted:** 08 November, 2024

# **Submitted By**

|  |  |
| --- | --- |
| Zain Ali | SP23-BSE-165 |

# **Introduction:**

I used the mockAPI for creating a dummy API, then added the CSS, HTML and SCRIPT files to create the front end. The API is able to perform the Update, Read, Delete and Create operations. The information is updated at the server as well.

The Postman screenshots are also pasted below.  
**API LINK:** https://672ddbc5fd8979715643feaf.mockapi.io/api/dummy\_Api/Cafe

# **Question:**

Applying CRUD operations Unsing Dummy API.

# **Answer:**

**HTML FILE:**<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Employee Management with Dummy API</title>

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

</head>

<body>

    <div class="container">

        <h1>Employee Management</h1>

        <form id="employeeForm">

            <input type="text" id="empName" placeholder="Employee Name" required>

            <input type="number" id="employeeId" placeholder="Employee ID" required>

            <input type="datetime-local" id="joinDate" placeholder="Joining Date" required>

            <button type="submit">Add Employee</button>

        </form>

        <div class="employee-list">

            <h2>Employee List</h2>

            <ul id="employeeList"></ul>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**CSS File:**

\* {

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

body {

    font-family: Arial, sans-serif;

    background-color: #f8f9fa;

    display: flex;

    align-items: center;

    justify-content: center;

    min-height: 100vh;

    padding: 20px;

}

.container {

    width: 100%;

    max-width: 400px;

    background-color: #ffffff;

    border-radius: 8px;

    padding: 20px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    text-align: center;

}

h1, h2 {

    color: #333;

    font-size: 1.2em;

    margin-bottom: 15px;

}

form {

    display: flex;

    flex-direction: column;

    gap: 8px;

    margin-bottom: 20px;

}

input {

    padding: 10px;

    font-size: 14px;

    border: 1px solid #ddd;

    border-radius: 4px;

    transition: border-color 0.3s;

}

input:focus {

    border-color: #4CAF50;

    outline: none;

}

button {

    padding: 10px;

    font-size: 14px;

    font-weight: bold;

    color: #fff;

    background-color: #4CAF50;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    transition: background-color 0.3s;

}

button:hover {

    background-color: #45a049;

}

.employee-list {

    margin-top: 20px;

    text-align: left;

}

ul {

    list-style-type: none;

    padding: 0;

}

.user-item {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px;

    border: 1px solid #ddd;

    border-radius: 4px;

    margin-bottom: 8px;

    font-size: 14px;

    transition: background-color 0.3s;

}

.user-item:hover {

    background-color: #f1f1f1;

}

.user-details {

    display: flex;

    gap: 10px;

    flex: 1;

}

.user-actions button {

    margin-left: 5px;

    padding: 6px 10px;

    font-size: 12px;

    border-radius: 4px;

    cursor: pointer;

    transition: background-color 0.3s;

}

.user-actions button.edit {

    background-color: #0275d8;

    color: #fff;

}

.user-actions button.delete {

    background-color: #d9534f;

    color: #fff;

}

.user-actions button:hover {

    opacity: 0.85;

}

**JavaScript:**const employeeForm = document.getElementById("employeeForm");

const employeeList = document.getElementById("employeeList");

const API\_URL = "https://672ddbc5fd8979715643feaf.mockapi.io/api/dummy\_Api/Cafe";

async function fetchEmployees() {

    try {

        const response = await fetch(API\_URL);

        const employees = await response.json();

        renderEmployees(employees);

    } catch (error) {

        console.error("Error fetching employees:", error);

    }

}

function renderEmployees(employees) {

    employeeList.innerHTML = "";

    employees.forEach(employee => {

        const joinDate = new Date(employee.join\_Date \* 1000).toLocaleDateString();

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

        employeeItem.classList.add("user-item");

        employeeItem.innerHTML = `

            <div class="user-details">

                <span>${employee.EmpName}</span>

                <span>ID: ${employee.EmployeeId}</span>

                <span>Joined: ${joinDate}</span>

            </div>

            <div class="user-actions">

                <button onclick="editEmployee(${employee.id})" class="edit">Edit</button>

                <button onclick="deleteEmployee(${employee.id})" class="delete">Delete</button>

            </div>

        `;

        employeeList.appendChild(employeeItem);

    });

}

employeeForm.addEventListener("submit", async (e) => {

    e.preventDefault();

    const empName = document.getElementById("empName").value;

    const employeeId = document.getElementById("employeeId").value;

    const joinDate = new Date(document.getElementById("joinDate").value).getTime() / 1000;

    try {

        const response = await fetch(API\_URL, {

            method: "POST",

            headers: { "Content-Type": "application/json" },

            body: JSON.stringify({ EmpName: empName, EmployeeId: employeeId, join\_Date: joinDate }),

        });

        await response.json();

        fetchEmployees();

        employeeForm.reset();

    } catch (error) {

        console.error("Error adding employee:", error);

    }

});

async function editEmployee(id) {

    const empName = prompt("Enter new Employee Name:");

    const employeeId = prompt("Enter new Employee ID:");

    const joinDate = new Date(prompt("Enter new Joining Date (YYYY-MM-DD):")).getTime() / 1000;

    if (empName && employeeId && joinDate) {

        try {

            await fetch(`${API\_URL}/${id}`, {

                method: "PUT",

                headers: { "Content-Type": "application/json" },

                body: JSON.stringify({ EmpName: empName, EmployeeId: employeeId, join\_Date: joinDate }),

            });

            fetchEmployees();

        } catch (error) {

            console.error("Error editing employee:", error);

        }

    }

}

async function deleteEmployee(id) {

    try {

        await fetch(`${API\_URL}/${id}`, { method: "DELETE" });

        fetchEmployees();

    } catch (error) {

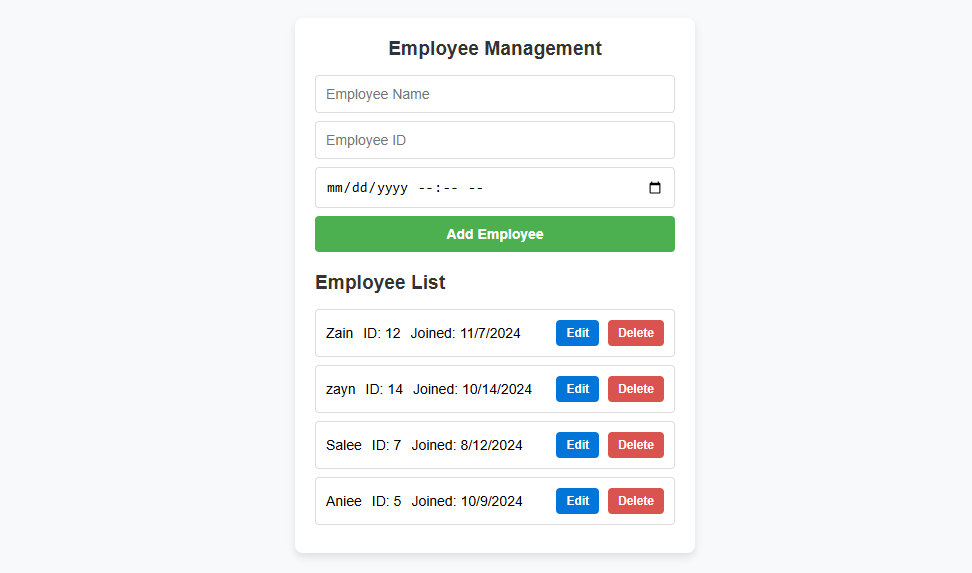
        console.error("Error deleting employee:", error);

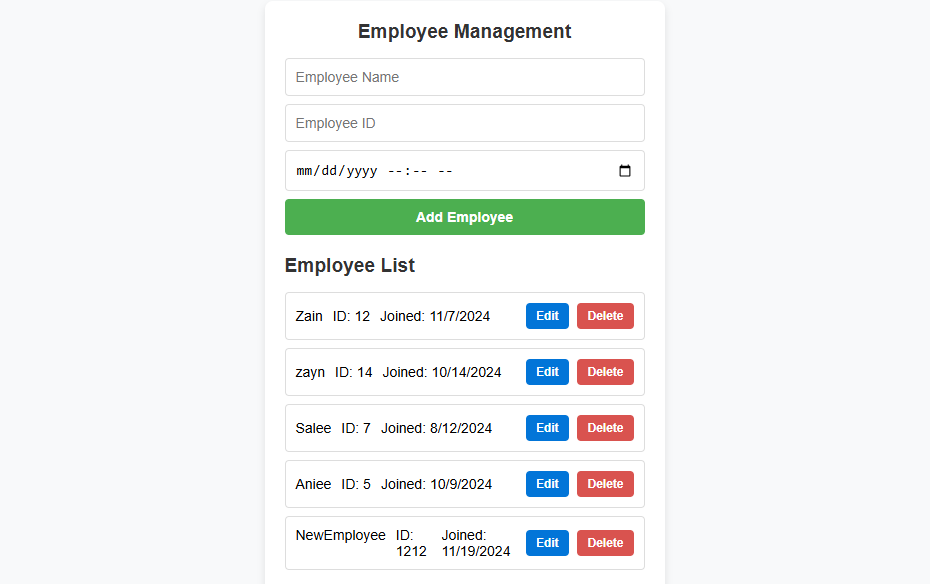
    }

}

fetchEmployees();

**CRUD-Operations:**

****

****

**A screenshot of a computer

Description automatically generated**

**UPDATING EMPLOYYE:**

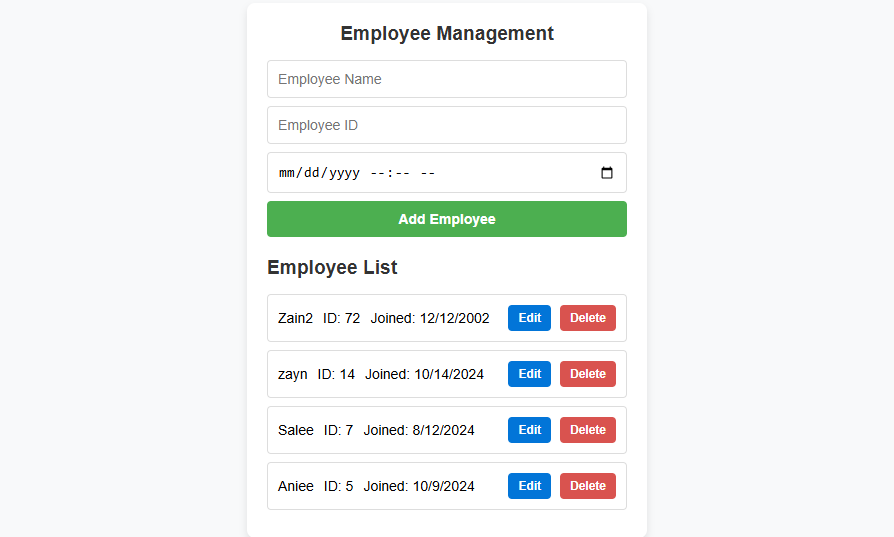
**A screenshot of a computer

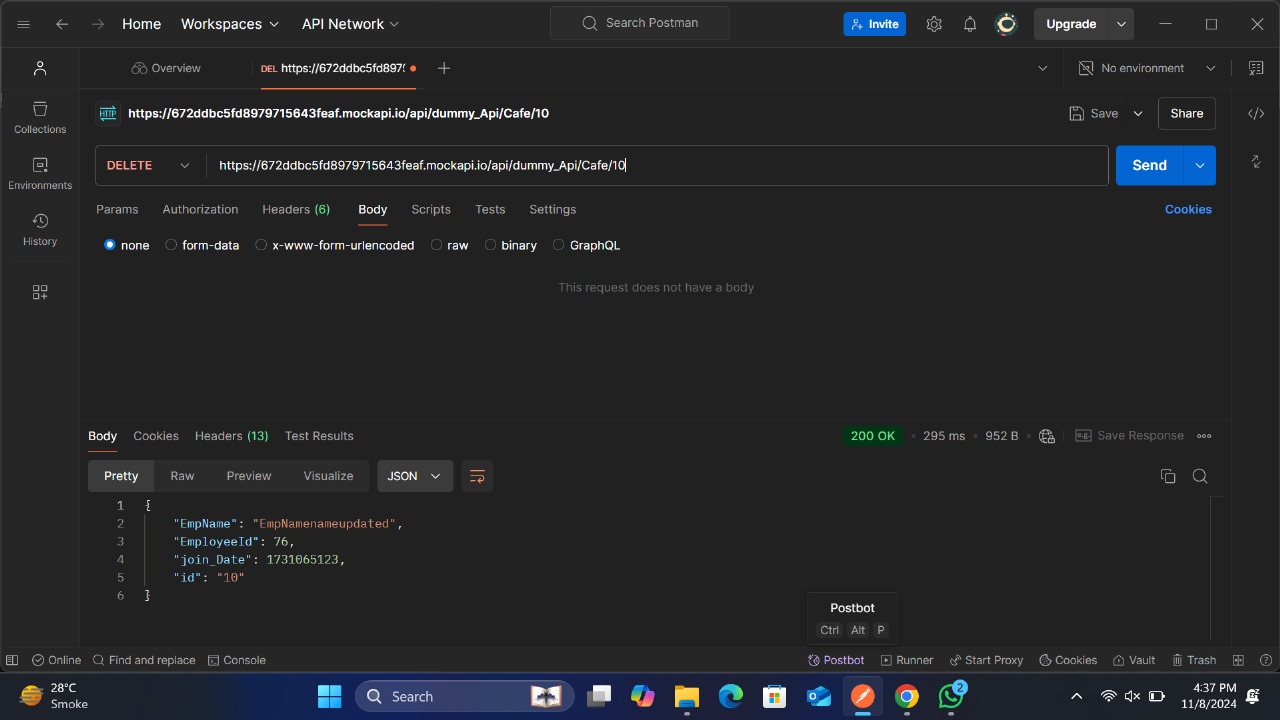
Description automatically generated**

**UPATED FILE:**

**A screenshot of a list

Description automatically generated**

**Deleted EMPLOYEE:  
  
**

**POSTMAN SS:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated