Github link:
https://github.com/UnicomTIC-Jerobert/unicomtic_2024_leave_management.git

Leave Management System

Phase - 1
* let's keep simple logics in order to understand complex technical stuffs
* we going to use (fetch api) for connecting frontend app build with [HTML , CSS , JavaScript]
* we going to use (async , await) to keep a readable code
* we going to use json-server as data provider for Rest API

Task :

Employees are working in an organization called UnicomTIC , they are entitled to apply for leaves. Once the leave request is applied by an employee , manager level staff can view the leave request and they can approve it , once it done final approval will be made by the director/CEO of the organization.

To keep things simple we will stick with this. In future we will add leave Types, leave allocations and more complex logic

In more detail the responsibility of users are as follows

Admin:
    * can create,edit,delete,view,search all employees

Employee
    * view his/her employee information
    * Apply his/her Leave [Goes to Manager and then CEO]
    * View his/her Status of Leave Requested
    * View his/her Leave History

Manager
    * View All Employees
    * Apply his/her Leave [Goes to CEO]
    * View his/her Status of Leave Requested
    * View All leaves Applied by other employee
    * Accept or Reject Leave

ceo
    * View All Employees
    * Apply his/her Leave [Goes to CEO]
    * View his/her Status of Leave Requested
    * View All leaves Applied by other employee
    * Accept or Reject Leave

For this system you required to the following activities step by step

1. Let's perform CRUD operations with search for Employee once it done credential also have to created
    Employee : {EMP ID , NIC number , first Name , Last Name , DOB , DOJ }
    users : {EMP ID , username , password , role}
2. Going to create Leave Application forms and View the Appiled leaves
3. Let's perform login activities and check whether it navigates to approriate Page , with appropriate Menus
    [Going to use Single Page App Mechanism , instead of creating 4 pages]
4. Further refinement on Leave processing Flow

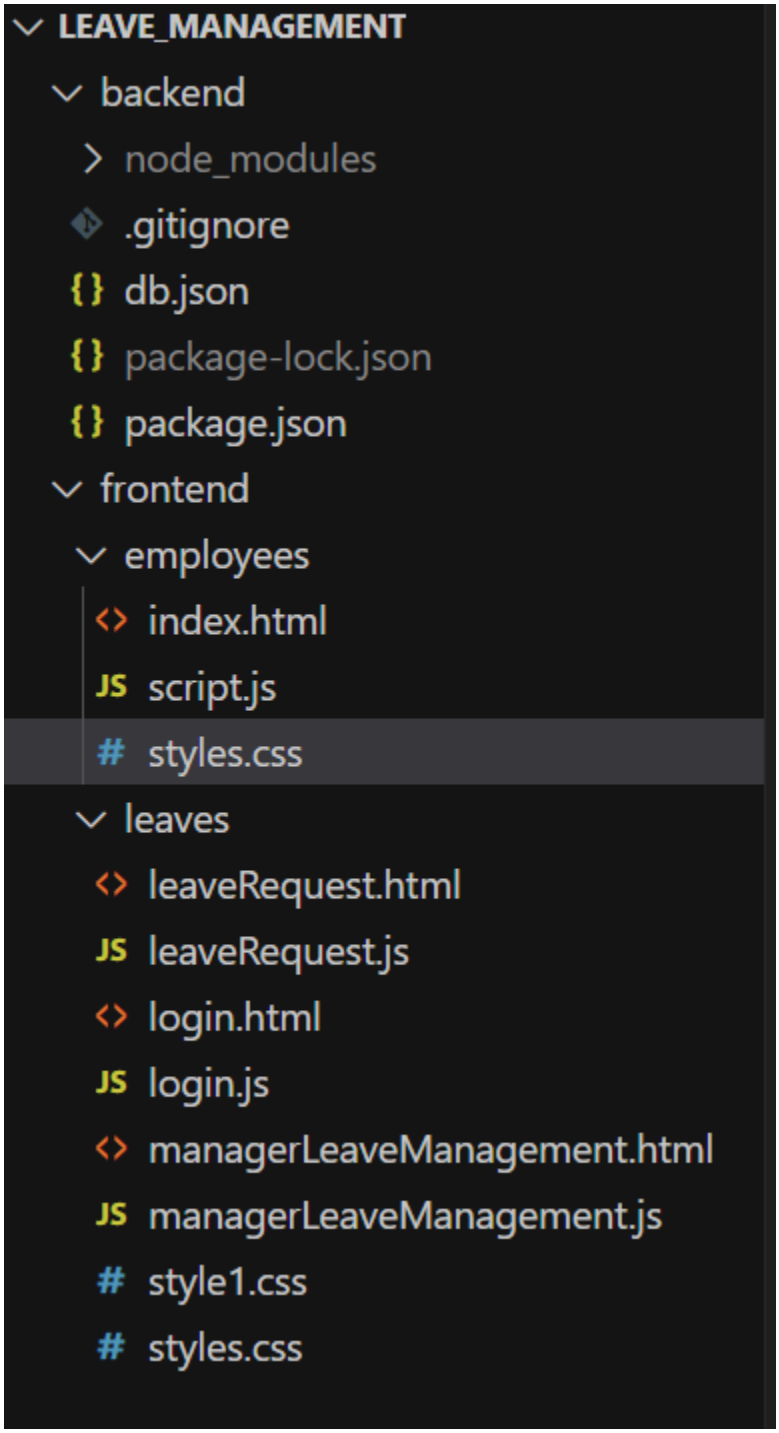
Steps:

- Create Folder called Leave_management_system
- Under that Folder create Two folders call Backend & FrontEnd

Command

Npm init -y
Npm install json-server

Overall Folder Structure



Files

Package.json

```
{
  "name": "leave_management",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "json-server --watch db.json --port 3000"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "json-server": "^1.0.0-beta.2"
  }
}
```

Db.json

```
{
  "employees": [
    {
      "id": "EMP_Dy4kP",
      "employeeId": "EMP_Dy4kP",
      "nic": "92312212V",
      "firstName": "Jerobert",
      "lastName": "zachariah",
      "dob": "2015-01-06",
      "doj": "2024-09-03"
    },
    {
      "id": "EMP_4SK8Q",
      "employeeId": "EMP_4SK8Q",
      "nic": "723451278V",
      "firstName": "Nelson",
      "lastName": "Mandella",
      "dob": "1972-12-05",
      "doj": "2024-09-01"
    }
  ],
  "users": [
```

```
    {
      "id": "EMP_Dy4kP",
      "employeeId": "EMP_Dy4kP",
      "username": "EMP_Dy4kP",
      "password": "pwd123",
      "role": "Employee"
    },
    {
      "id": "EMP_4SK8Q",
      "employeeId": "EMP_4SK8Q",
      "username": "EMP_4SK8Q",
      "password": "pwd123",
      "role": "Employee"
    }
  ],
  "leaveRequests": [
    {
      "id": "e491",
      "reason": "trip",
      "dateFrom": "2024-09-12",
      "numOfDays": "2",
      "applicationDate": "2024-09-06",
      "status": "Approved",
      "processedDate": "2024-09-05"
    },
    {
      "id": "a86b",
      "reason": "party",
      "dateFrom": "2024-09-17",
      "numOfDays": "1",
      "applicationDate": "2024-09-06",
      "status": "Rejected",
      "processedDate": "2024-09-05",
      "rejectionReason": "cant allow"
    }
  ]
}
```

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Management</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="container">
        <div class="flex-container">
            <h1>Employee Management</h1>
            <!-- Add Employee Button -->
            <button id="createEmployeeBtn">Add Employee</button>
        </div>
        <!-- Search Input -->
        <input type="text" id="searchInput" placeholder="Search by NIC, First Name, Last
Name..." style="margin-top: 10px;">
        <!-- Employee Table -->
        <table id="employeeTable">
            <thead>
                <tr>
                    <th>EMP ID</th>
                    <th>NIC</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Date of Birth</th>
                    <th>Date of Joining</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody id="employeeTableBody"></tbody>
        </table>
        <br>
        <!-- Toast Notification -->
        <div id="toast"></div>
        <!-- Employee Modal -->
        <div id="employeeModal" class="modal">
            <div class="modal-content">
                <span class="close">&times;</span>
                <h2 id="modalTitle">Add Employee</h2>
                <form id="employeeForm">
                    <input type="hidden" id="empId" name="empId">
```

```
                <div>
                    <label for="nic">NIC Number:</label>
                    <input type="text" id="nic" name="nic" required>
                </div>
                <div>
                    <label for="firstName">First Name:</label>
                    <input type="text" id="firstName" name="firstName" required>
                </div>
                <div>
                    <label for="lastName">Last Name:</label>
                    <input type="text" id="lastName" name="lastName" required>
                </div>
                <div>
                    <label for="dob">Date of Birth:</label>
                    <input type="date" id="dob" name="dob" required>
                </div>
                <div>
                    <label for="doj">Date of Joining:</label>
                    <input type="date" id="doj" name="doj" required>
                </div>
                <button type="submit" id="saveButton">Save</button>
            </form>
        </div>
    </div>
    <!-- Credentials Modal -->
    <div id="credentialsModal" class="modal">
        <div class="modal-content">
            <span class="close">&times;</span>
            <h2>Edit Credentials</h2>
            <form id="credentialsForm">
                <input type="hidden" id="credentialsEmpId" name="empId">
                <div>
                    <label for="username">Username:</label>
                    <input type="text" id="username" name="username" disabled required>
                </div>
                <div>
                    <label for="password">Password:</label>
                    <input type="text" id="password" name="password" required>
                </div>
                <div>
                    <label for="userrole">Role:</label>
                    <select id="userrole" name="userrole">
                        <option value="Employee">Employee</option>
                        <option value="Manager">Manager</option>
                        <option value="Admin">Admin</option>
                        <option value="Director">Director</option>
                    </select>
```

```
                    </div>
                    <button type="submit">Save Credentials</button>
                </form>
            </div>
        </div>
    </div>
    <script src="./script.js"></script>
</body>

</html>
```

script.js

```
document.addEventListener("DOMContentLoaded", async () => {

    const apiUrl = "http://localhost:3000/employees";
    const userApiUrl = "http://localhost:3000/users";

    let employees = [];
    let users = [];
    let editingEmployeeId = null;
    const employeeTableBody = document.getElementById("employeeTableBody");
    const employeeModal = document.getElementById("employeeModal");
    const credentialsModal = document.getElementById("credentialsModal");
    const employeeForm = document.forms['employeeForm'];
    const credentialsForm = document.forms['credentialsForm'];
    const toast = document.getElementById("toast");
    // Fetch Employees
    const fetchEmployees = async () => {
        const response = await fetch(apiUrl);
        employees = await response.json();
        renderEmployees();
    };
    // Fetch Users
    const fetchUsers = async () => {
        const response = await fetch(userApiUrl);
        users = await response.json();
    };
    // Render Employees
    const renderEmployees = () => {
        employeeTableBody.innerHTML = "";
        employees.forEach((employee) => {
            const row = document.createElement("tr");
            row.innerHTML = `
    <td>${employee.employeeId}</td>
    <td>${employee.nic}</td>
    <td>${employee.firstName}</td>
```

```
    <td>${employee.lastName}</td>
    <td>${employee.dob}</td>
    <td>${employee.doj}</td>
    <td>
    <button onclick="editEmployee('${employee.employeeId}')">Edit</button>
    <button onclick="deleteEmployee('${employee.employeeId}')">Delete</button>
    <button onclick="editCredentials('${employee.employeeId}')">Edit
    Credentials</button>
    </td>
    `;
            employeeTableBody.appendChild(row);
        });
    };
    // Open Employee Modal for Create or Edit
    document.getElementById("createEmployeeBtn").onclick = () => {
        openEmployeeModal();
    };
    const openEmployeeModal = (employee = null) => {
        employeeModal.style.display = "block";
        const modalTitle = document.getElementById("modalTitle");
        const saveButton = document.getElementById("saveButton");
        if (employee) {
            modalTitle.innerText = "Edit Employee";
            saveButton.innerText = "Update";
            employeeForm.nic.value = employee.nic;
            employeeForm.firstName.value = employee.firstName;
            employeeForm.lastName.value = employee.lastName;
            employeeForm.dob.value = employee.dob;
            employeeForm.doj.value = employee.doj;
            editingEmployeeId = employee.employeeId;
        } else {
            employeeForm.reset();
            modalTitle.innerText = "Add Employee";
            saveButton.innerText = "Save";
            editingEmployeeId = null;
        }
    };
    // Close Modal
    document.querySelectorAll(".close").forEach(btn => {
        btn.onclick = () => {
            employeeModal.style.display = "none";
            credentialsModal.style.display = "none";
        };
    });
    // Function to generate unique EMP ID
    const generateEmpId = () => {
        const characters =
```

```
            'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
        let empId;
        do {
            empId = 'EMP_' + Array.from({ length: 5 }, () =>
                characters.charAt(Math.floor(Math.random() * characters.length))).join('');
        } while (employees.some(emp => emp.empId === empId));
        return empId;
    };
    // Save Employee
    employeeForm.onsubmit = async (e) => {
        e.preventDefault();
        const empId = editingEmployeeId || generateEmpId();
        const nic = employeeForm.nic.value;
        const firstName = employeeForm.firstName.value;
        const lastName = employeeForm.lastName.value;
        const dob = employeeForm.dob.value;
        const doj = employeeForm.doj.value;

        const employeeData = {
            id: empId,
            employeeId: empId,
            nic: nic,
            firstName: firstName,
            lastName: lastName,
            dob: dob,
            doj: doj
        };

        if (editingEmployeeId) {
            // Update employee
            await fetch(`${apiUrl}/${editingEmployeeId}`, {
                method: "PUT",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(employeeData)
            });
        } else {
            // Create new employee
            await fetch(apiUrl, {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(employeeData)
            });
            // Automatically create user credentials
```

```
            const userData = {
                id: empId,
                employeeId: empId,
                username: empId,
                password: "pwd123",
                role: "Employee" // Default role as Employee; can be modified as needed
            };
            await fetch(userApiUrl, {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(userData)
            });
        }
        employeeModal.style.display = "none";
        await fetchEmployees();
        await fetchUsers(); // To refresh user data if needed
    };
    // Delete Employee
    window.deleteEmployee = async (empId) => {
        if (confirm("Are you sure you want to delete this employee?")) {
            await fetch(`${apiUrl}/${empId}`, {
                method: "DELETE"
            });
            await fetch(`${userApiUrl}/${empId}`, {
                method: "DELETE"
            });
            showToast("Deleted Successfully");
            await fetchEmployees();
        }
    };
    // Edit Employee
    window.editEmployee = (empId) => {
        const employee = employees.find(emp => emp.employeeId === empId);
        openEmployeeModal(employee);
    };
    // Open Credentials Modal
    window.editCredentials = (empId) => {
        const user = users.find(usr => usr.employeeId === empId);
        if (user) {
            credentialsModal.style.display = "block";
            credentialsForm.username.value = user.username;
            credentialsForm.password.value = user.password;
            credentialsForm.userrole.value = user.role;
            credentialsForm.credentialsEmpId.value = user.employeeId;
        }
```

```
};
// Save Credentials
credentialsForm.onsubmit = async (e) => {
    e.preventDefault();
    const userData = {
        username: credentialsForm.username.value,
        password: credentialsForm.password.value,
        role: credentialsForm.userrole.value,
        employeeId: credentialsForm.credentialsEmpId.value
    };
    const user = users.find(usr => usr.employeeId === userData.employeeId);
    if (user) {
        await fetch(`${userApiUrl}/${user.id}`, {
            method: "PUT",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(userData)
        });
    } else {
        await fetch(userApiUrl, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(userData)
        });
    }
    credentialsModal.style.display = "none";
    await fetchUsers();
};
// Show Toast
const showToast = (message) => {
    toast.innerText = message;
    toast.classList.add("show");
    setTimeout(() => {
        toast.classList.remove("show");
    }, 3000);
};
// Search Employees
document.getElementById("searchInput").oninput = (e) => {
    const query = e.target.value.toLowerCase();
    const filteredEmployees = employees.filter(emp => {
        return emp.nic.toLowerCase().includes(query) ||
            emp.firstName.toLowerCase().includes(query) ||
            emp.lastName.toLowerCase().includes(query);
    });
```

```
        renderFilteredEmployees(filteredEmployees);
    };
    const renderFilteredEmployees = (filteredEmployees) => {
        employeeTableBody.innerHTML = "";
        filteredEmployees.forEach((employee) => {
            const row = document.createElement("tr");
            row.innerHTML = `
    <td>${employee.employeeId}</td>
    <td>${employee.nic}</td>
    <td>${employee.firstName}</td>
    <td>${employee.lastName}</td>
    <td>${employee.dob}</td>
    <td>${employee.doj}</td>
    <td>
    <button onclick="editEmployee('${employee.employeeId}')">Edit</button>
    <button onclick="deleteEmployee('${employee.employeeId}')">Delete</button>
    <button onclick="editCredentials('${employee.employeeId}')">Edit
    Credentials</button>
    </td>
    `;
            employeeTableBody.appendChild(row);
        });
    };
    // Initialize
    fetchEmployees();
    fetchUsers();
});
```

styles.css

```
* {
    margin: 0;
    padding: 0;
}

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
}

.container {
    max-width: 1000px;
    margin: 0 auto;
```

```
    background: #fff;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.flex-container {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
}

h1 {
    text-align: center;
}

#searchInput {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    box-sizing: border-box;
}

#employeeTable {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

#employeeTable th,
#employeeTable td {
    border: 1px solid #ddd;
    padding: 8px;
}

#employeeTable th {
    background-color: #f2f2f2;
    text-align: left;
}

#employeeTable tr:nth-child(even) {
    background-color: #f9f9f9;
}

button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
```

```
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

.modal {
    display: none;
    position: fixed;
    z-index: 1;
    padding-top: 100px;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.4);
}

.modal-content {
    background-color: white;
    margin: auto;
    padding: 20px;
    border: 1px solid #888;
    width: 50%;
    box-sizing: border-box;
}

.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}

.toast {
    visibility: hidden;
    min-width: 250px;
    margin-left: -125px;
```

```
    background-color: #333;
    color: #fff;
    text-align: center;
    border-radius: 2px;
    padding: 16px;
    position: fixed;
    z-index: 1;
    left: 50%;
    bottom: 30px;
    font-size: 17px;
}

.toast.show {
    visibility: visible;
    animation: fadein 0.5s, fadeout 0.5s 2.5s;
}

@keyframes fadein {
    from {
        bottom: 0;
        opacity: 0;
    }

    to {
        bottom: 30px;
        opacity: 1;
    }
}

@keyframes fadeout {
    from {
        bottom: 30px;
        opacity: 1;
    }

    to {
        bottom: 0;
        opacity: 0;
    }
}
```

Leaves

leaveRequest.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Leave Request</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="header">
            <h2 id="welcomeMessage"></h2>
            <button id="logoutButton" class="btn-logout">Logout</button>
        </div>
        <h2>Submit Leave Request</h2>
        <form id="leaveRequestForm" class="form">
            <label for="reason">Reason:</label>
            <input type="text" id="reason" name="reason" required><br><br>

            <label for="dateFrom">From:</label>
            <input type="date" id="dateFrom" name="dateFrom" required><br><br>

            <label for="numOfDays">Number of Days:</label>
            <input type="number" id="numOfDays" name="numOfDays" required><br><br>

            <label for="applicationDate">Application Date:</label>
            <input type="date" id="applicationDate" name="applicationDate"
required><br><br>

            <button type="submit" class="btn">Submit</button>
        </form>

        <h3>Your Leave Requests</h3>
        <ul id="leaveRequests" class="leave-list"></ul>
    </div>

    <script src="leaveRequest.js"></script>
</body>
</html>
```

leaveRequest.js

```
document.addEventListener('DOMContentLoaded', async function() {
    const loggedInUser = JSON.parse(localStorage.getItem('loggedInUser'));

    if (!loggedInUser) {
        alert('You must log in first!');
        window.location.href = 'login.html';
        return;
    }

    // Display welcome message
    const welcomeMessage = document.getElementById('welcomeMessage');
    welcomeMessage.textContent = `Welcome, ${loggedInUser.firstName}
${loggedInUser.lastName}`;

    // Logout button functionality
    const logoutButton = document.getElementById('logoutButton');
    logoutButton.addEventListener('click', function() {
        localStorage.removeItem('loggedInUser');
        alert('You have been logged out.');
        window.location.href = 'login.html';
    });

    const leaveRequestForm = document.getElementById('leaveRequestForm');
    const leaveRequestsList = document.getElementById('leaveRequests');

    leaveRequestForm.addEventListener('submit', async function(event) {
        event.preventDefault();

        const reason = document.getElementById('reason').value;
        const dateFrom = document.getElementById('dateFrom').value;
        const numOfDays = document.getElementById('numOfDays').value;
        const applicationDate = document.getElementById('applicationDate').value;

        // Validate that application date is less than dateFrom
        if (new Date(applicationDate) >= new Date(dateFrom)) {
            alert('Application date must be earlier than the leave start date.');
            return;
        }

        const newLeaveRequest = {
            employeeId: loggedInUser.empId,
            reason,
            dateFrom,
            numOfDays,
            applicationDate, // Use the selected application date
```

```
            status: 'Pending' // Initial status
        };

        try {
            const response = await fetch('http://localhost:3000/leaveRequests', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(newLeaveRequest)
            });

            if (response.ok) {
                alert('Leave request submitted successfully!');
                leaveRequestForm.reset();
                loadLeaveRequests(loggedInUser.empId);
            } else {
                alert('Failed to submit leave request.');
            }
        } catch (error) {
            console.error('Error:', error);
            alert('An error occurred while submitting the leave request.');
        }
    });

    async function loadLeaveRequests(employeeId) {
        try {
            const response = await
fetch(`http://localhost:3000/leaveRequests?employeeId=${employeeId}`);
            const leaveRequests = await response.json();

            leaveRequestsList.innerHTML = '';
            leaveRequests.forEach(request => {
                const li = document.createElement('li');
                li.classList.add('leave-item'); // Add a class for styling
                li.innerHTML = `
                    <div>Reason: ${request.reason}</div>
                    <div>From: ${request.dateFrom}</div>
                    <div>Days: ${request.numOfDays}</div>
                    <div>Applied on: ${request.applicationDate}</div>
                    <div>Status: <span class="status
${request.status.toLowerCase()}">${request.status}</span></div>
                `;
                leaveRequestsList.appendChild(li);
            });
        } catch (error) {
            console.error('Error:', error);
```

```
        }
    }

    loadLeaveRequests(loggedInUser.empId);
});
```

Login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="container">
        <h2>Login</h2>
        <form id="loginForm" class="form">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required><br><br>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required><br><br>

            <button type="submit" class="btn">Login</button>
        </form>
    </div>

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

</html>
```

login.js

```
document.getElementById('loginForm').addEventListener('submit', async function (event) {
    event.preventDefault();

    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
```

```
    try {
        // Fetch user data with embedded employee data

        // http://localhost:3000/users?_embed=employee&username=EMP_L6Zae&password=pwd123

        const response = await
fetch(`http://localhost:3000/users?username=${username}&password=${password}&_embed=employe
e`);
        //console.log(response);

        const users = await response.json();
        //console.log(users);

        if (users.length > 0) {
            const user = users[0];
            //console.log(user);
            // Find the associated employee data
            const employee = user.employee;

            //console.log(employee);

            if (employee) {
                // Store the logged-in user and employee details
                const loggedInUser = {
                    ...user,
                    firstName: employee.firstName,
                    lastName: employee.lastName,
                };

                console.log(loggedInUser);

                localStorage.setItem('loggedInUser', JSON.stringify(loggedInUser));

                alert('Login successful!');
                window.location.href = 'leaveRequest.html';
            } else {
                alert('Employee data not found.');
            }
        } else {
            alert('Invalid username or password.');
        }
    } catch (error) {
        console.error('Error:', error);
        alert('An error occurred while logging in.');
    }
});
```

managerLeaveManagement.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Leave Management</title>
    <link rel="stylesheet" href="style1.css">
</head>

<body>
    <div class="container">
        <div class="header">
            <h2 id="welcomeMessage"></h2>
            <button id="logoutButton" class="btn-logout">Logout</button>
        </div>
        <h2>Manage Leave Requests</h2>

        <label for="statusFilter">Filter by Status:</label>
        <select id="statusFilter">
            <option value="all">All</option>
            <option value="pending">Pending</option>
            <option value="approved">Approved</option>
            <option value="rejected">Rejected</option>
        </select>

        <ul id="leaveRequests" class="leave-list"></ul>
    </div>

    <!-- Modal for Rejecting Leave -->
    <div id="rejectModal" class="modal">
        <div class="modal-content">
            <span class="close">&times;</span>
            <h3>Reject Leave Request</h3>
            <form id="rejectForm">
                <label for="rejectionReason">Reason for Rejection:</label>
                <textarea id="rejectionReason" name="rejectionReason"
required></textarea><br><br>
                <button type="submit" class="btn">Submit</button>
            </form>
        </div>
    </div>

    <script src="managerLeaveManagement.js"></script>
</body>
```

```
</html>
```

managerLeaveManagement.js

```
document.addEventListener('DOMContentLoaded', async function () {
    const loggedInUser = JSON.parse(localStorage.getItem('loggedInUser'));

    if (!loggedInUser) {
        alert('You must log in first!');
        window.location.href = 'login.html';
        return;
    }

    // Display welcome message
    const welcomeMessage = document.getElementById('welcomeMessage');
    welcomeMessage.textContent = `Welcome, ${loggedInUser.firstName}
${loggedInUser.lastName}`;

    // Logout button functionality
    const logoutButton = document.getElementById('logoutButton');
    logoutButton.addEventListener('click', function () {
        localStorage.removeItem('loggedInUser');
        alert('You have been logged out.');
        window.location.href = 'login.html';
    });

    const statusFilter = document.getElementById('statusFilter');
    const leaveRequestsList = document.getElementById('leaveRequests');
    let leaveRequests = [];

    statusFilter.addEventListener('change', function () {
        renderLeaveRequests();
    });

    async function loadLeaveRequests() {
        try {
            const response = await fetch('http://localhost:3000/leaveRequests');
            leaveRequests = await response.json();
            renderLeaveRequests();
        } catch (error) {
            console.error('Error:', error);
        }
    }

    function renderLeaveRequests() {
        const status = statusFilter.value;
```

```
        leaveRequestsList.innerHTML = '';

        const filteredRequests = leaveRequests.filter(request => {
            return status === 'all' || request.status.toLowerCase() === status;
        });

        filteredRequests.forEach(request => {
            const li = document.createElement('li');
            li.classList.add('leave-item');
            li.innerHTML = `
                <div>Reason: ${request.reason}</div>
                <div>From: ${request.dateFrom}</div>
                <div>Days: ${request.numOfDays}</div>
                <div>Applied on: ${request.applicationDate}</div>
                <div>Status: <span class="status
${request.status.toLowerCase()}">${request.status}</span></div>
                ${request.status === 'Pending' ? `
                <button class="btn-approve" data-id="${request.id}">Approve</button>
                <button class="btn-reject" data-id="${request.id}">Reject</button>` : ''}
            `;
            leaveRequestsList.appendChild(li);
        });

        document.querySelectorAll('.btn-approve').forEach(button => {
            button.addEventListener('click', handleApprove);
        });

        document.querySelectorAll('.btn-reject').forEach(button => {
            button.addEventListener('click', handleReject);
        });
    }

    async function handleApprove(event) {
        const requestId = event.target.dataset.id;
        const request = leaveRequests.find(req => req.id == requestId);
        if (request) {
            request.status = 'Approved';
            request.processedDate = new Date().toISOString().split('T')[0]; // Adding
processed date

            try {
                await fetch(`http://localhost:3000/leaveRequests/${requestId}`, {
                    method: 'PATCH',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify(request)
```

```
            });
            loadLeaveRequests();
        } catch (error) {
            console.error('Error:', error);
        }
    }
}

const rejectModal = document.getElementById('rejectModal');
const rejectForm = document.getElementById('rejectForm');
let requestToReject = null;

function handleReject(event) {
    requestToReject = leaveRequests.find(req => req.id == event.target.dataset.id);
    if (requestToReject) {
        rejectModal.style.display = 'block';
    }
}

rejectForm.addEventListener('submit', async function (event) {
    event.preventDefault();
    const rejectionReason = document.getElementById('rejectionReason').value;

    if (requestToReject) {
        requestToReject.status = 'Rejected';
        requestToReject.processedDate = new Date().toISOString().split('T')[0];
        requestToReject.rejectionReason = rejectionReason;

        try {
            await fetch(`http://localhost:3000/leaveRequests/${requestToReject.id}`, {
                method: 'PATCH',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(requestToReject)
            });
            rejectModal.style.display = 'none';
            loadLeaveRequests();
        } catch (error) {
            console.error('Error:', error);
        }
    }
});

// Close modal
document.querySelector('.close').addEventListener('click', function () {
    rejectModal.style.display = 'none';
```

```
    });

    // Load initial leave requests
    loadLeaveRequests();
});
```

Style1.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 800px;
    margin: auto;
    padding: 20px;
    background: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.header h2 {
    margin: 0;
}

.btn-logout {
    background-color: #d9534f;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 4px;
    cursor: pointer;
}

h2 {
```

```
    margin-bottom: 20px;
    color: #333;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

input,
textarea,
select {
    width: 100%;
    padding: 8px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
}



.btn {
    background-color: #5cb85c;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 4px;
    cursor: pointer;
}

.btn-approve {
    background-color: #5cb85c;
    color: white;
    border: none;
    padding: 5px 10px;
    border-radius: 4px;
    cursor: pointer;
}

.btn-reject {
    background-color: #d9534f;
    color: white;
    border: none;
    padding: 5px 10px;
    border-radius: 4px;
    cursor: pointer;
```

```
}

.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgba(0, 0, 0, 0.4);
    padding-top: 60px;
}

.modal-content {
    background-color: #fff;
    margin: 5% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
    border-radius: 8px;
}

.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
```

Styles.css

```css
/* Global Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #ffffff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 400px;
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

h2 {
    text-align: center;
    color: #333;
    margin-bottom: 20px;
}

.form label {
    display: block;
    color: #555;
    font-weight: bold;
    margin-bottom: 5px;
}

.form input[type="text"],
.form input[type="password"],
.form input[type="date"],
```

```css
.form input[type="number"] {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
}

.form input[type="text"]:focus,
.form input[type="password"]:focus,
.form input[type="date"]:focus,
.form input[type="number"]:focus {
    border-color: #0066cc;
    outline: none;
}

.btn, .btn-logout {
    background-color: #0066cc;
    color: white;
    padding: 10px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

.btn-logout {
    padding: 5px 10px;
    font-size: 14px;
}

.btn:hover, .btn-logout:hover {
    background-color: #004999;
}

.leave-list {
    list-style-type: none;
    padding: 0;
}

.leave-list li {
    background-color: #e9ecef;
    margin: 5px 0;
    padding: 10px;
    border-radius: 4px;
    font-size: 14px;
```

```
}


.leave-item {
    padding: 10px;
    border: 1px solid #ddd;
    margin-bottom: 10px;
    background-color: #f9f9f9;
    border-radius: 4px;
}

.status {
    font-weight: bold;
}

.status.pending {
    color: #f0ad4e;
}

.status.approved {
    color: #5cb85c;
}

.status.rejected {
    color: #d9534f;
}
```

Extra Source Code :

app.js

```
const http = require("http");
const fs = require("fs").promises;
const path = require("path");

async function renderPage(layoutFile, contentFile) {
  try {
    let layout = await fs.readFile(layoutFile, "utf8");
    let content = await fs.readFile(contentFile, "utf8");

    // Extract <style> block
    const styleStart = content.indexOf("<style>");
    const styleEnd = content.indexOf("</style>");

    let style = "";
    if (styleStart !== -1 && styleEnd !== -1) {
      style = content.substring(styleStart, styleEnd + "</style>".length);
      content =
        content.substring(0, styleStart) +
        content.substring(styleEnd + "</style>".length);
    }

    // Extract <script> block
    const scriptStart = content.indexOf("<script>");
    const scriptEnd = content.indexOf("</script>");

    let script = "";
    if (scriptStart !== -1 && scriptEnd !== -1) {
      script = content.substring(scriptStart, scriptEnd + "</script>".length);
      content =
        content.substring(0, scriptStart) +
        content.substring(scriptEnd + "</script>".length);
    }

    // Replace placeholders with actual content
    const renderedPage = layout
      .replace("{{content}}", content)
      .replace("{{style}}", style)
      .replace("{{script}}", script);

    return renderedPage;
  } catch (err) {
    throw err;
```

```
  }
}

const server = http.createServer(async (req, res) => {
  // Handle static file requests
  if (req.url.startsWith("/static/")) {
    const filePath = path.join(__dirname, req.url);
    try {
      const data = await fs.readFile(filePath);
      res.writeHead(200, { "Content-Type": getContentType(filePath) });
      res.end(data);
      return;
    } catch (err) {
      res.writeHead(404, { "Content-Type": "text/html" });
      return res.end("<h1>404 Not Found</h1>");
    }
  }

  // Handle HTML page requests
  let filePath = "./views/Home/index.html";
  let layout_use = "./layouts/layout_user.html";

  if (req.url === "/") {
    filePath = "./views/Home/index.html";
  } else if (req.url === "/about") {
    filePath = "./views/about.html";
  } else {
    res.writeHead(404, { "Content-Type": "text/html" });
    return res.end("<h1>404 Not Found</h1>");
  }

  try {
    let renderedPage = await renderPage(layout_use, filePath);
    res.writeHead(200, { "Content-Type": "text/html" });
    res.end(renderedPage);
  } catch (err) {
    console.log(err);
    res.writeHead(500, { "Content-Type": "text/html" });
    res.end("<h1>500 Internal Server Error</h1>");
  }
});

// Function to determine content type
function getContentType(filePath) {
  const extname = path.extname(filePath);
  switch (extname) {
    case ".css":
```

```
      return "text/css";
    case ".js":
      return "application/javascript";
    case ".png":
      return "image/png";
    case ".jpg":
      return "image/jpeg";
    case ".gif":
      return "image/gif";
    case ".svg":
      return "image/svg+xml";
    case ".html":
      return "text/html";
    default:
      return "application/octet-stream";
  }
}

// Set the server to listen on port 3000
const PORT = process.env.PORT || 4000;
server.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```