<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1"/>

<title>Bus Transport Management System</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');

\* {

box-sizing: border-box;

}

body {

font-family: 'Roboto', sans-serif;

margin: 0;

padding: 0;

background: #f5f7fa;

color: #333;

min-height: 100vh;

display: flex;

flex-direction: column;

}

header {

background: #007bff;

color: #fff;

padding: 1rem 1.5rem;

text-align: center;

font-size: 1.5rem;

font-weight: 700;

letter-spacing: 1px;

}

nav {

display: flex;

justify-content: center;

background: #ffffff;

box-shadow: 0 3px 8px rgb(0 0 0 / 0.1);

}

nav button {

flex: 1;

padding: 1rem;

background: none;

border: none;

border-bottom: 3px solid transparent;

font-weight: 600;

font-size: 1rem;

cursor: pointer;

transition: border-color 0.3s ease;

color: #555;

}

nav button.active {

border-bottom: 3px solid #007bff;

color: #007bff;

}

main {

flex: 1;

padding: 1rem 1.5rem;

max-width: 900px;

margin: 1rem auto 2rem;

background: #fff;

border-radius: 8px;

box-shadow: 0 5px 15px rgb(0 0 0 / 0.1);

}

h2 {

margin-top: 0;

margin-bottom: 1rem;

font-weight: 700;

color: #007bff;

letter-spacing: 0.5px;

}

form {

margin-bottom: 1.5rem;

}

label {

display: block;

margin-bottom: 0.4rem;

font-weight: 600;

color: #444;

}

input[type="text"],

input[type="number"],

input[type="time"],

select {

width: 100%;

padding: 0.5rem 0.75rem;

margin-bottom: 1rem;

border: 1.5px solid #ccc;

border-radius: 4px;

font-size: 1rem;

transition: border-color 0.3s ease;

}

input[type="text"]:focus,

select:focus,

input[type="number"]:focus,

input[type="time"]:focus {

border-color: #007bff;

outline: none;

}

button.primary {

background-color: #007bff;

border: none;

border-radius: 4px;

color: white;

font-weight: 600;

padding: 0.6rem 1.25rem;

font-size: 1rem;

cursor: pointer;

transition: background-color 0.3s ease;

}

button.primary:hover {

background-color: #0056b3;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 0.5rem;

}

th, td {

text-align: left;

padding: 0.75rem 0.5rem;

border-bottom: 1px solid #dee2e6;

font-size: 0.95rem;

}

th {

color: #007bff;

font-weight: 700;

}

tr:hover {

background-color: #f1f8ff;

}

.actions button {

background: none;

border: none;

cursor: pointer;

font-size: 1.1rem;

margin-right: 0.5rem;

color: #007bff;

}

.actions button.delete {

color: #dc3545;

}

@media (max-width: 600px) {

nav button {

font-size: 0.9rem;

padding: 0.8rem;

}

th, td {

font-size: 0.85rem;

padding: 0.5rem 0.3rem;

}

main {

margin: 1rem 0.5rem 2rem;

padding: 1rem;

}

}

</style>

</head>

<body>

<header>Bus Transport Management System</header>

<nav>

<button id="tab-buses" class="active" aria-controls="section-buses" aria-selected="true" role="tab">Buses</button>

<button id="tab-routes" aria-controls="section-routes" aria-selected="false" role="tab">Routes</button>

<button id="tab-schedules" aria-controls="section-schedules" aria-selected="false" role="tab">Schedules</button>

</nav>

<main>

<section id="section-buses" role="tabpanel" aria-labelledby="tab-buses">

<h2>Manage Buses</h2>

<form id="bus-form" aria-label="Add or edit bus">

<input type="hidden" id="bus-id" />

<label for="bus-number">Bus Number</label>

<input type="text" id="bus-number" placeholder="e.g. BH1234" required />

<label for="bus-capacity">Capacity</label>

<input type="number" id="bus-capacity" min="1" max="100" placeholder="e.g. 40" required />

<label for="bus-driver">Driver Name</label>

<input type="text" id="bus-driver" placeholder="Driver Name" required />

<button type="submit" class="primary">Add Bus</button>

</form>

<table aria-label="List of buses" id="bus-table">

<thead>

<tr>

<th>Bus Number</th>

<th>Capacity</th>

<th>Driver Name</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<!-- Buses will be listed here -->

</tbody>

</table>

</section>

<section id="section-routes" role="tabpanel" aria-labelledby="tab-routes" hidden>

<h2>Manage Routes</h2>

<form id="route-form" aria-label="Add or edit route">

<input type="hidden" id="route-id" />

<label for="route-name">Route Name</label>

<input type="text" id="route-name" placeholder="e.g. City Center to Airport" required />

<label for="route-start">Start Point</label>

<input type="text" id="route-start" placeholder="Starting location" required />

<label for="route-end">End Point</label>

<input type="text" id="route-end" placeholder="Ending location" required />

<button type="submit" class="primary">Add Route</button>

</form>

<table aria-label="List of routes" id="route-table">

<thead>

<tr>

<th>Route Name</th>

<th>Start Point</th>

<th>End Point</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<!-- Routes will be listed here -->

</tbody>

</table>

</section>

<section id="section-schedules" role="tabpanel" aria-labelledby="tab-schedules" hidden>

<h2>Manage Schedules</h2>

<form id="schedule-form" aria-label="Add or edit schedule">

<input type="hidden" id="schedule-id" />

<label for="schedule-bus">Select Bus</label>

<select id="schedule-bus" required>

<option value="" disabled selected>Select a bus</option>

</select>

<label for="schedule-route">Select Route</label>

<select id="schedule-route" required>

<option value="" disabled selected>Select a route</option>

</select>

<label for="schedule-time">Departure Time</label>

<input type="time" id="schedule-time" required />

<button type="submit" class="primary">Add Schedule</button>

</form>

<table aria-label="List of schedules" id="schedule-table">

<thead>

<tr>

<th>Bus Number</th>

<th>Route Name</th>

<th>Departure Time</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<!-- Schedules will be listed here -->

</tbody>

</table>

</section>

</main>

<script>

// Tabs switch logic

const tabs = document.querySelectorAll('nav button');

const sections = document.querySelectorAll('main section');

tabs.forEach(tab => {

tab.addEventListener('click', () => {

// Deactivate all tabs and hide sections

tabs.forEach(t => {

t.classList.remove('active');

t.setAttribute('aria-selected', 'false');

});

sections.forEach(s => s.hidden = true);

// Activate clicked tab and show section

tab.classList.add('active');

tab.setAttribute('aria-selected', 'true');

const sectionId = tab.getAttribute('aria-controls');

document.getElementById(sectionId).hidden = false;

});

});

// Local Storage Keys

const BUS\_STORAGE = 'tms\_buses';

const ROUTE\_STORAGE = 'tms\_routes';

const SCHEDULE\_STORAGE = 'tms\_schedules';

// Data arrays

let buses = [];

let routes = [];

let schedules = [];

// Elements Buses

const busForm = document.getElementById('bus-form');

const busIdInput = document.getElementById('bus-id');

const busNumberInput = document.getElementById('bus-number');

const busCapacityInput = document.getElementById('bus-capacity');

const busDriverInput = document.getElementById('bus-driver');

const busTableBody = document.querySelector('#bus-table tbody');

// Elements Routes

const routeForm = document.getElementById('route-form');

const routeIdInput = document.getElementById('route-id');

const routeNameInput = document.getElementById('route-name');

const routeStartInput = document.getElementById('route-start');

const routeEndInput = document.getElementById('route-end');

const routeTableBody = document.querySelector('#route-table tbody');

// Elements Schedules

const scheduleForm = document.getElementById('schedule-form');

const scheduleIdInput = document.getElementById('schedule-id');

const scheduleBusSelect = document.getElementById('schedule-bus');

const scheduleRouteSelect = document.getElementById('schedule-route');

const scheduleTimeInput = document.getElementById('schedule-time');

const scheduleTableBody = document.querySelector('#schedule-table tbody');

// Utility Functions

function saveToStorage(key, data) {

localStorage.setItem(key, JSON.stringify(data));

}

function loadFromStorage(key) {

const data = localStorage.getItem(key);

return data ? JSON.parse(data) : [];

}

function generateId() {

return '\_' + Math.random().toString(36).substr(2, 9);

}

// Render functions

function renderBuses() {

busTableBody.innerHTML = '';

buses.forEach(bus => {

const tr = document.createElement('tr');

tr.innerHTML = `

<td>${bus.number}</td>

<td>${bus.capacity}</td>

<td>${bus.driver}</td>

<td class="actions">

<button title="Edit Bus" onclick="editBus('${bus.id}')">&#9998;</button>

<button title="Delete Bus" class="delete" onclick="deleteBus('${bus.id}')">&#128465;</button>

</td>

`;

busTableBody.appendChild(tr);

});

populateBusOptions();

}

function renderRoutes() {

routeTableBody.innerHTML = '';

routes.forEach(route => {

const tr = document.createElement('tr');

tr.innerHTML = `

<td>${route.name}</td>

<td>${route.start}</td>

<td>${route.end}</td>

<td class="actions">

<button title="Edit Route" onclick="editRoute('${route.id}')">&#9998;</button>

<button title="Delete Route" class="delete" onclick="deleteRoute('${route.id}')">&#128465;</button>

</td>

`;

routeTableBody.appendChild(tr);

});

populateRouteOptions();

}

function renderSchedules() {

scheduleTableBody.innerHTML = '';

schedules.forEach(schedule => {

const bus = buses.find(b => b.id === schedule.busId);

const route = routes.find(r => r.id === schedule.routeId);

const busNumber = bus ? bus.number : '[Bus Removed]';

const routeName = route ? route.name : '[Route Removed]';

const tr = document.createElement('tr');

tr.innerHTML = `

<td>${busNumber}</td>

<td>${routeName}</td>

<td>${schedule.time}</td>

<td class="actions">

<button title="Edit Schedule" onclick="editSchedule('${schedule.id}')">&#9998;</button>

<button title="Delete Schedule" class="delete" onclick="deleteSchedule('${schedule.id}')">&#128465;</button>

</td>

`;

scheduleTableBody.appendChild(tr);

});

}

function populateBusOptions() {

// clear options except placeholder

const selectedBus = scheduleBusSelect.value;

scheduleBusSelect.innerHTML = '<option value="" disabled>Select a bus</option>';

buses.forEach(bus => {

const option = document.createElement('option');

option.value = bus.id;

option.textContent = bus.number + ' (' + bus.driver + ')';

scheduleBusSelect.appendChild(option);

});

scheduleBusSelect.value = selectedBus || '';

}

function populateRouteOptions() {

const selectedRoute = scheduleRouteSelect.value;

scheduleRouteSelect.innerHTML = '<option value="" disabled>Select a route</option>';

routes.forEach(route => {

const option = document.createElement('option');

option.value = route.id;

option.textContent = route.name + ' (' + route.start + ' → ' + route.end + ')';

scheduleRouteSelect.appendChild(option);

});

scheduleRouteSelect.value = selectedRoute || '';

}

// Bus Form Handlers

busForm.addEventListener('submit', event => {

event.preventDefault();

const id = busIdInput.value;

const number = busNumberInput.value.trim();

const capacity = parseInt(busCapacityInput.value, 10);

const driver = busDriverInput.value.trim();

if (!number || !driver || isNaN(capacity) || capacity < 1) {

alert('Please fill all bus fields correctly.');

return;

}

if (id) {

// Edit

const busIndex = buses.findIndex(b => b.id === id);

if (busIndex !== -1) {

buses[busIndex] = { id, number, capacity, driver };

}

busForm.querySelector('button.primary').textContent = 'Add Bus';

} else {

// Add new

buses.push({ id: generateId(), number, capacity, driver });

}

busIdInput.value = '';

busNumberInput.value = '';

busCapacityInput.value = '';

busDriverInput.value = '';

saveToStorage(BUS\_STORAGE, buses);

renderBuses();

});

window.editBus = function(id) {

const bus = buses.find(b => b.id === id);

if (!bus) return;

busIdInput.value = bus.id;

busNumberInput.value = bus.number;

busCapacityInput.value = bus.capacity;

busDriverInput.value = bus.driver;

busForm.querySelector('button.primary').textContent = 'Update Bus';

document.getElementById('tab-buses').click();

};

window.deleteBus = function(id) {

if (!confirm('Are you sure you want to delete this bus? This will also remove related schedules.')) return;

buses = buses.filter(b => b.id !== id);

schedules = schedules.filter(s => s.busId !== id); // remove related schedules

saveToStorage(BUS\_STORAGE, buses);

saveToStorage(SCHEDULE\_STORAGE, schedules);

renderBuses();

renderSchedules();

};

// Route Form Handlers

routeForm.addEventListener('submit', event => {

event.preventDefault();

const id = routeIdInput.value;

const name = routeNameInput.value.trim();

const start = routeStartInput.value.trim();

const end = routeEndInput.value.trim();

if (!name || !start || !end) {

alert('Please fill all route fields.');

return;

}

if (id) {

const routeIndex = routes.findIndex(r => r.id === id);

if (routeIndex !== -1) {

routes[routeIndex] = { id, name, start, end };

}

routeForm.querySelector('button.primary').textContent = 'Add Route';

} else {

routes.push({ id: generateId(), name, start, end });

}

routeIdInput.value = '';

routeNameInput.value = '';

routeStartInput.value = '';

routeEndInput.value = '';

saveToStorage(ROUTE\_STORAGE, routes);

renderRoutes();

});

window.editRoute = function(id) {

const route = routes.find(r => r.id === id);

if (!route) return;

routeIdInput.value = route.id;

routeNameInput.value = route.name;

routeStartInput.value = route.start;

routeEndInput.value = route.end;

routeForm.querySelector('button.primary').textContent = 'Update Route';

document.getElementById('tab-routes').click();

};

window.deleteRoute = function(id) {

if (!confirm('Are you sure you want to delete this route? This will also remove related schedules.')) return;

routes = routes.filter(r => r.id !== id);

schedules = schedules.filter(s => s.routeId !== id);

saveToStorage(ROUTE\_STORAGE, routes);

saveToStorage(SCHEDULE\_STORAGE, schedules);

renderRoutes();

renderSchedules();

};

// Schedule Form Handlers

scheduleForm.addEventListener('submit', event => {

event.preventDefault();

const id = scheduleIdInput.value;

const busId = scheduleBusSelect.value;

const routeId = scheduleRouteSelect.value;

const time = scheduleTimeInput.value;

if (!busId || !routeId || !time) {

alert('Please select bus, route and departure time.');

return;

}

if (id) {

const scheduleIndex = schedules.findIndex(s => s.id === id);

if (scheduleIndex !== -1) {

schedules[scheduleIndex] = { id, busId, routeId, time };

}

scheduleForm.querySelector('button.primary').textContent = 'Add Schedule';

} else {

schedules.push({ id: generateId(), busId, routeId, time });

}

scheduleIdInput.value = '';

scheduleBusSelect.value = '';

scheduleRouteSelect.value = '';

scheduleTimeInput.value = '';

saveToStorage(SCHEDULE\_STORAGE, schedules);

renderSchedules();

});

window.editSchedule = function(id) {

const schedule = schedules.find(s => s.id === id);

if (!schedule) return;

scheduleIdInput.value = schedule.id;

scheduleBusSelect.value = schedule.busId;

scheduleRouteSelect.value = schedule.routeId;

scheduleTimeInput.value = schedule.time;

scheduleForm.querySelector('button.primary').textContent = 'Update Schedule';

document.getElementById('tab-schedules').click();

};

window.deleteSchedule = function(id) {

if (!confirm('Are you sure you want to delete this schedule?')) return;

schedules = schedules.filter(s => s.id !== id);

saveToStorage(SCHEDULE\_STORAGE, schedules);

renderSchedules();

};

// Initialization function

function init() {

buses = loadFromStorage(BUS\_STORAGE);

routes = loadFromStorage(ROUTE\_STORAGE);

schedules = loadFromStorage(SCHEDULE\_STORAGE);

renderBuses();

renderRoutes();

renderSchedules();

}

init();

</script>

</body>

</html>