frontend/pages/manage\_contract.html "<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Manage Contract</title>

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

<link rel="stylesheet" href="../css/manage\_contract.css">

</head>

<body>

<div class="navbar" style="display: flex; align-items: center; justify-content: space-between; flex-wrap: nowrap; padding: 0 1rem; min-height: 70px; background: #0758aa;">

<ul class="nav-links" id="nav-links" style="display: flex; align-items: center; gap: 1.5rem; margin: 0; padding: 0; list-style: none;">

<li><a href="dashboard.html" style="color: #fff; text-decoration: none; font-weight: 500;">Dashboard</a></li>

<li><a href="manage\_contract.html" style="color: #fff; text-decoration: none; font-weight: 500;">Manage Contract</a></li>

<li id="users-nav" style="display: none;"><a href="users.php" style="color: #fff; text-decoration: none; font-weight: 500;">Users</a></li>

</ul>

<div class="user-info" id="user-info" style="display: flex; align-items: center; gap: 1rem; margin-left: 1.5rem;">

<span id="username-display">Loading...</span>

<span id="role-badge" class="user-badge">user</span>

<button class="logout-btn" onclick="logout()">Logout</button>

</div>

<div class="nwr\_logo" style="margin-left: 1.5rem;">

<img src="../images/nwr\_logo.png" alt="NWR Logo" style="height:60px;">

</div>

</div>

<style>@media (max-width: 900px) {.navbar {flex-direction: row;flex-wrap: nowrap;padding: 0 0.5rem;}.nav-links {gap: 0.7rem !important;}.user-info {gap: 0.5rem !important;}.nwr\_logo img {height: 40px !important;}}@media (max-width: 600px) {.navbar {flex-direction: row;flex-wrap: nowrap;padding: 0 0.2rem;}.nav-links {gap: 0.3rem !important;}.user-info {gap: 0.3rem !important;font-size: 0.85rem;}.nwr\_logo img {height: 30px !important;}}.navbar .nav-links li a:hover {text-decoration: underline;}</style>

<div id="access-denied" class="access-denied" style="display: none;">

<h2>Access Denied</h2>

<p>You don't have permission to manage contracts. Only managers and administrators can create, edit, or delete contracts.</p>

<a href="dashboard.html" class="btn btn-primary">Return to Dashboard</a>

</div>

<div class="container" id="main-container" style="display: none;">

<h1>Manage Contracts</h1>

<!-- Message display area -->

<div id="message" class="message"></div>

<!-- Tabs -->

<div class="tabs">

<button class="tab active" data-tab="create">Create New Contract</button>

<button class="tab" data-tab="edit">Edit Contract</button>

<button class="tab" data-tab="delete">Delete Contract</button>

</div>

<!-- Create Contract Tab -->

<div id="create-tab" class="tab-content active">

<h2>Add a New Contract</h2>

<form id="create-form" enctype="multipart/form-data">

<div class="form-row">

<div class="form-group">

<label for="parties">Parties \*</label>

<input type="text" id="parties" name="parties" placeholder="Enter contract parties" required>

</div>

<div class="form-group">

<label for="typeOfContract">Type of Contract \*</label>

<select id="typeOfContract" name="typeOfContract" required>

<option value="">Select contract type</option>

<option value="Service Agreement">Service Agreement</option>

<option value="Supply Agreement">Supply Agreement</option>

<option value="Employment Contract">Employment Contract</option>

<option value="Lease Agreement">Lease Agreement</option>

<option value="Partnership Agreement">Partnership Agreement</option>

<option value="Other">Other</option>

</select>

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="duration">Duration \*</label>

<input type="text" id="duration" name="duration" placeholder="e.g., 12 months, 2 years"

required>

</div>

<div class="form-group">

<label for="contractValue">Contract Value \*</label>

<input type="number" id="contractValue" name="contractValue" step="0.01" placeholder="0.00"

required>

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="description">Description</label>

<textarea id="description" name="description" rows="3"

placeholder="Enter contract description" style="width: 100%; padding: 8px; border: none; border-radius: 0px; background-color: #0758aa;

color: #ecf0f1;"></textarea>

</div>

<div class="form-group">

<label for="contractFile">Upload Contract Document</label>

<input type="file" id="contractFile" name="contractFile" accept=".pdf,.doc,.docx">

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="expiryDate">Expiry Date \*</label>

<input type="date" id="expiryDate" name="expiryDate" required>

</div>

<div class="form-group">

<label for="reviewByDate">Review By Date</label>

<input type="date" id="reviewByDate" name="reviewByDate">

</div>

</div>

<div class="button-group">

<button type="submit" class="btn btn-primary">Create Contract</button>

<button type="reset" class="btn btn-warning">Reset Form</button>

</div>

</form>

</div>

<!-- Edit Contract Tab -->

<div id="edit-tab" class="tab-content">

<h2>Edit Contract</h2>

<div class="form-row">

<div class="form-group" style="flex: 1;">

<label>Select Contract to Edit</label>

<div id="contract-list" class="contract-list">

<div class="loading">Loading contracts...</div>

</div>

</div>

<div class="form-group" style="flex: 2;">

<label>Edit Contract Details</label>

<form id="edit-form" enctype="multipart/form-data" style="display: none;">

<input type="hidden" id="edit-contractid" name="contractid">

<div class="form-row">

<div class="form-group">

<label for="edit-parties">Parties \*</label>

<input type="text" id="edit-parties" name="parties" required>

</div>

<div class="form-group">

<label for="edit-typeOfContract">Type of Contract \*</label>

<select id="edit-typeOfContract" name="typeOfContract" required>

<option value="">Select contract type</option>

<option value="Service Agreement">Service Agreement</option>

<option value="Supply Agreement">Supply Agreement</option>

<option value="Employment Contract">Employment Contract</option>

<option value="Lease Agreement">Lease Agreement</option>

<option value="Partnership Agreement">Partnership Agreement</option>

<option value="Other">Other</option>

</select>

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="edit-duration">Duration \*</label>

<input type="text" id="edit-duration" name="duration" required>

</div>

<div class="form-group">

<label for="edit-contractValue">Contract Value \*</label>

<input type="number" id="edit-contractValue" name="contractValue" step="0.01" required>

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="edit-description">Description</label>

<textarea id="edit-description" name="description" rows="3"></textarea>

</div>

<div class="form-group">

<label for="edit-contractFile">Upload Contract Document</label>

<input type="file" id="edit-contractFile" name="contractFile" accept=".pdf,.doc,.docx">

</div>

<div id="edit-view-file" style="margin-top:10px; display:none;">

<div class="button-group">

<a id="view-contract-file" href="#" target="\_blank" class="btn btn-primary">

View Contract File

</a>

<a id="download-contract-file" href="#" download class="btn btn-primary">

Download Contract File

</a>

</div>

</div>

</div>

<div class="form-row">

<div class="form-group">

<label for="edit-expiryDate">Expiry Date \*</label>

<input type="date" id="edit-expiryDate" name="expiryDate" required>

</div>

<div class="form-group">

<label for="edit-reviewByDate">Review By Date</label>

<input type="date" id="edit-reviewByDate" name="reviewByDate">

</div>

</div>

<div class="button-group">

<button type="submit" class="btn btn-primary">Update Contract</button>

<button type="button" class="btn btn-warning" onclick="clearEditForm()">Cancel</button>

</div>

</form>

<div id="edit-placeholder" class="loading" style="text-align: center; padding: 50px; color: #999;">

Select a contract from the list to edit

</div>

</div>

</div>

</div>

<!-- Delete Contract Tab -->

<div id="delete-tab" class="tab-content">

<h2>Delete Contract</h2>

<div class="warning"

style="background: #fff3cd; border: 1px solid #ffeaa7; color: #856404; padding: 15px; border-radius: 5px; margin-bottom: 20px;">

<strong>Warning:</strong> Contract deletion is permanent and cannot be undone!

</div>

<div class="form-group">

<label>Select Contract to Delete</label>

<div id="delete-contract-list" class="contract-list">

<div class="loading">Loading contracts...</div>

</div>

</div>

<div id="delete-confirmation"

style="display: none; margin-top: 20px; padding: 20px; background: #f8f9fa; border-radius: 5px; color: #333;">

<h3>Confirm Deletion</h3>

<p><strong>Contract ID:</strong> <span id="delete-contract-id"></span></p>

<p><strong>Parties:</strong> <span id="delete-contract-parties"></span></p>

<p><strong>Type:</strong> <span id="delete-contract-type"></span></p>

<p><strong>Value:</strong> <span id="delete-contract-value"></span></p>

<div class="button-group">

<button type="button" class="btn btn-danger" onclick="confirmDelete()">Delete Contract</button>

<button type="button" class="btn btn-warning" onclick="cancelDelete()">Cancel</button>

</div>

</div>

</div>

</div>

<script src="../js/manage\_contract.js"></script>

</body>

</html>"

frontend/js/manage\_contract.js "// ===============================

// Global Variables

// ===============================

let contracts = [];

let selectedContract = null;

let contractToDelete = null;

let currentUser = null;

// API base URL (backend entrypoint)

const API\_BASE = "/nwrcontractregistry/backend/index.php";

// ===============================

// Authentication & Authorization

// ===============================

async function checkAuth() {

try {

const response = await fetch(

"/nwrcontractregistry/backend/auth\_handler.php?action=check"

);

const result = await response.json();

// Redirect to login if not logged in

if (!result.loggedIn) {

window.location.href = "../index.php";

return false;

}

currentUser = result.user;

// Restrict access → only managers & admins

if (currentUser.role !== "manager" && currentUser.role !== "admin") {

const deniedEl = document.getElementById("access-denied");

const mainEl = document.getElementById("main-container");

if (deniedEl) deniedEl.style.display = "block";

if (mainEl) mainEl.style.display = "none";

return false;

}

// Update UI with user info

updateUserInfo();

// Show main content if available

const mainEl = document.getElementById("main-container");

if (mainEl) mainEl.style.display = "block";

return true;

} catch (error) {

console.error("Auth check failed:", error);

window.location.href = "../index.php";

return false;

}

}

// ===============================

// User Info Display

// ===============================

function updateUserInfo() {

if (!currentUser) return;

const usernameEl = document.getElementById("username-display");

const roleBadge = document.getElementById("role-badge");

if (usernameEl) usernameEl.textContent = currentUser.username;

if (roleBadge) {

roleBadge.textContent = currentUser.role;

roleBadge.className = `user-badge ${currentUser.role}`;

}

// Show "Users" nav link only for admins

if (currentUser.role === "admin") {

const usersNav = document.getElementById("users-nav");

if (usersNav) usersNav.style.display = "block";

}

}

// ===============================

// Logout

// ===============================

async function logout() {

try {

await fetch("/nwrcontractregistry/backend/auth\_handler.php?action=logout", {

method: "POST",

});

window.location.href = "../index.php";

} catch (error) {

console.error("Logout error:", error);

window.location.href = "../index.php";

}

}

// ===============================

// Initialize Page (after DOM load)

// ===============================

document.addEventListener("DOMContentLoaded", async function () {

const hasAccess = await checkAuth();

if (hasAccess) {

initializeTabs();

loadContracts();

setupFormHandlers();

}

});

// ===============================

// Tabs Handling

// ===============================

function initializeTabs() {

const tabs = document.querySelectorAll(".tab");

const tabContents = document.querySelectorAll(".tab-content");

tabs.forEach((tab) => {

tab.addEventListener("click", function () {

const targetTab = this.dataset.tab;

// Reset all tabs & contents

tabs.forEach((t) => t.classList.remove("active"));

tabContents.forEach((tc) => tc.classList.remove("active"));

// Activate clicked tab & corresponding content

this.classList.add("active");

const targetContent = document.getElementById(targetTab + "-tab");

if (targetContent) targetContent.classList.add("active");

// Refresh contracts when switching to edit/delete

if (targetTab === "edit" || targetTab === "delete") {

loadContracts();

}

});

});

}

// ===============================

// Load & Update Contracts

// ===============================

async function loadContracts() {

try {

const response = await fetch(`${API\_BASE}?action=list`);

if (!response.ok) {

if (response.status === 401) {

window.location.href = "../index.php";

return;

}

throw new Error("Failed to fetch contracts");

}

contracts = await response.json();

console.log("Loaded contracts:", contracts.length);

updateContractLists();

} catch (error) {

console.error("Error loading contracts:", error);

showMessage("Error loading contracts: " + error.message, "error");

}

}

function updateContractLists() {

updateEditContractList();

updateDeleteContractList();

}

function updateEditContractList() {

const container = document.getElementById("contract-list");

if (!container) return;

if (contracts.length === 0) {

container.innerHTML = '<div class="loading">No contracts found</div>';

return;

}

container.innerHTML = contracts

.map(

(contract) => `

<div class="contract-item" onclick="selectContractForEdit(${

contract.contractid

}, event)">

<h4>ID: ${contract.contractid} - ${contract.parties}</h4>

<p><strong>Type:</strong> ${contract.typeOfContract || "N/A"}</p>

<p><strong>Value:</strong> ${formatCurrency(

contract.contractValue

)} | <strong>Expires:</strong> ${contract.expiryDate || "N/A"}</p>

</div>

`

)

.join("");

}

function updateDeleteContractList() {

const container = document.getElementById("delete-contract-list");

if (!container) return;

if (contracts.length === 0) {

container.innerHTML = '<div class="loading">No contracts found</div>';

return;

}

container.innerHTML = contracts

.map(

(contract) => `

<div class="contract-item" onclick="selectContractForDelete(${

contract.contractid

}, event)">

<h4>ID: ${contract.contractid} - ${contract.parties}</h4>

<p><strong>Type:</strong> ${contract.typeOfContract || "N/A"}</p>

<p><strong>Value:</strong> ${formatCurrency(

contract.contractValue

)} | <strong>Expires:</strong> ${contract.expiryDate || "N/A"}</p>

</div>

`

)

.join("");

}

// ===============================

// Forms

// ===============================

function setupFormHandlers() {

const createForm = document.getElementById("create-form");

const editForm = document.getElementById("edit-form");

if (createForm) createForm.addEventListener("submit", handleCreateContract);

if (editForm) editForm.addEventListener("submit", handleUpdateContract);

}

// Handle contract creation

async function handleCreateContract(e) {

e.preventDefault();

const formData = new FormData(e.target);

try {

const response = await fetch(`${API\_BASE}?action=create`, {

method: "POST",

body: formData,

});

if (!response.ok) {

if (response.status === 401) {

window.location.href = "../index.php";

return;

}

if (response.status === 403) {

showMessage(

"Access denied. You do not have permission to create contracts.",

"error"

);

return;

}

throw new Error("Failed to create contract: " + response.statusText);

}

await response.json();

showMessage("Contract created successfully!", "success");

e.target.reset();

loadContracts();

} catch (error) {

console.error("Error creating contract:", error);

showMessage("Error creating contract: " + error.message, "error");

}

}

// ===============================

// Edit Contracts

// ===============================

function selectContractForEdit(contractId, event) {

const contract = contracts.find((c) => c.contractid == contractId);

if (!contract) return;

selectedContract = contract;

// Highlight selected

document.querySelectorAll("#contract-list .contract-item").forEach((item) => {

item.classList.remove("selected");

});

if (event?.target.closest(".contract-item")) {

event.target.closest(".contract-item").classList.add("selected");

}

// Show form & populate fields

const editForm = document.getElementById("edit-form");

const placeholder = document.getElementById("edit-placeholder");

if (editForm && placeholder) {

editForm.style.display = "block";

placeholder.style.display = "none";

}

populateEditForm(contract);

}

function populateEditForm(contract) {

const mapping = {

"edit-contractid": contract.contractid,

"edit-parties": contract.parties || "",

"edit-typeOfContract": contract.typeOfContract || "",

"edit-duration": contract.duration || "",

"edit-contractValue": contract.contractValue || "",

"edit-description": contract.description || "",

"edit-expiryDate": contract.expiryDate || "",

"edit-reviewByDate": contract.reviewByDate || "",

};

Object.entries(mapping).forEach(([id, value]) => {

const el = document.getElementById(id);

if (el) el.value = value;

});

// Handle file links

const viewContainer = document.getElementById("edit-view-file");

const viewLink = document.getElementById("view-contract-file");

const downloadLink = document.getElementById("download-contract-file");

if (contract.filepath && viewContainer && viewLink && downloadLink) {

const fileUrl = `/nwrcontractregistry/backend/${contract.filepath}`;

viewLink.href = fileUrl;

downloadLink.href = fileUrl;

viewContainer.style.display = "block";

} else if (viewContainer && viewLink && downloadLink) {

viewContainer.style.display = "none";

viewLink.href = "#";

downloadLink.href = "#";

}

}

function clearEditForm() {

const editForm = document.getElementById("edit-form");

const placeholder = document.getElementById("edit-placeholder");

if (editForm && placeholder) {

editForm.style.display = "none";

placeholder.style.display = "block";

editForm.reset();

}

selectedContract = null;

document.querySelectorAll("#contract-list .contract-item").forEach((item) => {

item.classList.remove("selected");

});

}

async function handleUpdateContract(e) {

e.preventDefault();

if (!selectedContract) {

showMessage("No contract selected for editing", "error");

return;

}

const formData = new FormData(e.target);

const contractId = formData.get("contractid");

formData.delete("contractid");

try {

const response = await fetch(`${API\_BASE}?action=update&id=${contractId}`, {

method: "POST",

body: formData,

});

if (!response.ok) {

if (response.status === 401) {

window.location.href = "../index.php";

return;

}

if (response.status === 403) {

showMessage(

"Access denied. You do not have permission to edit contracts.",

"error"

);

return;

}

throw new Error("Failed to update contract");

}

await response.json();

showMessage("Contract updated successfully!", "success");

clearEditForm();

loadContracts();

} catch (error) {

console.error("Error updating contract:", error);

showMessage("Error updating contract: " + error.message, "error");

}

}

// ===============================

// Delete Contracts

// ===============================

function selectContractForDelete(contractId, event) {

const contract = contracts.find((c) => c.contractid == contractId);

if (!contract) return;

contractToDelete = contract;

// Highlight selected

document

.querySelectorAll("#delete-contract-list .contract-item")

.forEach((item) => {

item.classList.remove("selected");

});

if (event?.target.closest(".contract-item")) {

event.target.closest(".contract-item").classList.add("selected");

}

// Show confirmation

const confirmation = document.getElementById("delete-confirmation");

if (confirmation) confirmation.style.display = "block";

const mapping = {

"delete-contract-id": contract.contractid,

"delete-contract-parties": contract.parties || "N/A",

"delete-contract-type": contract.typeOfContract || "N/A",

"delete-contract-value": formatCurrency(contract.contractValue),

};

Object.entries(mapping).forEach(([id, value]) => {

const el = document.getElementById(id);

if (el) el.textContent = value;

});

}

async function confirmDelete() {

if (!contractToDelete) return;

const contractId = contractToDelete.contractid;

console.log("Deleting contract:", contractId);

try {

const response = await fetch(`${API\_BASE}?action=delete&id=${contractId}`, {

method: "DELETE",

});

if (!response.ok) {

if (response.status === 401) {

window.location.href = "../index.php";

return;

}

if (response.status === 403) {

showMessage(

"Access denied. You do not have permission to delete contracts.",

"error"

);

return;

}

throw new Error("Failed to delete contract");

}

await response.json();

showMessage("Contract deleted successfully!", "success");

cancelDelete();

loadContracts();

} catch (error) {

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

showMessage("Error deleting contract: " + error.message, "error");

}

}

function cancelDelete() {

contractToDelete = null;

const confirmation = document.getElementById("delete-confirmation");

if (confirmation) confirmation.style.display = "none";

document

.querySelectorAll("#delete-contract-list .contract-item")

.forEach((item) => {

item.classList.remove("selected");

});

}

// ===============================

// Helpers

// ===============================

function showMessage(message, type) {

const messageEl = document.getElementById("message");

if (!messageEl) return;

messageEl.textContent = message;

messageEl.className = `message ${type}`;

messageEl.style.display = "block";

// Auto-hide after 5 seconds

setTimeout(() => {

messageEl.style.display = "none";

}, 5000);

}

function formatCurrency(value) {

if (!value || isNaN(value)) return "N/A";

const numValue = parseFloat(value);

return new Intl.NumberFormat("en-US", {

style: "currency",

currency: "USD",

minimumFractionDigits: 2,

}).format(numValue);

}

"

frontend/js/app.js "async function loadContracts() {

console.log("Loading contracts...");

const backendUrl = "/nwrcontractregistry/backend/index.php?action=list";

try {

const response = await fetch(backendUrl);

if (!response.ok) {

throw new Error(`HTTP ${response.status}: ${response.statusText}`);

}

let contracts = await response.json();

const tbody = document.querySelector("#contracts-table tbody");

tbody.innerHTML = "";

if (Array.isArray(contracts) && contracts.length > 0) {

// Sort by expiry date (earliest first, null/invalid at the end)

contracts.sort((a, b) => {

const dateA = a.expiryDate ? new Date(a.expiryDate) : null;

const dateB = b.expiryDate ? new Date(b.expiryDate) : null;

if (!dateA && !dateB) return 0;

if (!dateA) return 1; // push invalid/empty dates to bottom

if (!dateB) return -1;

return dateA - dateB; // earliest first

});

contracts.forEach((contract) => {

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

let formattedValue = "N/A";

if (contract.contractValue && !isNaN(contract.contractValue)) {

formattedValue =

"N$ " +

new Intl.NumberFormat("en-US", {

minimumFractionDigits: 2,

maximumFractionDigits: 2,

}).format(parseFloat(contract.contractValue));

}

tr.innerHTML = `

<td>${contract.contractid || "N/A"}</td>

<td>${contract.parties || "N/A"}</td>

<td>${contract.typeOfContract || "N/A"}</td>

<td>${contract.duration || "N/A"}</td>

<td>${contract.description || "N/A"}</td>

<td>${contract.expiryDate || "N/A"}</td>

<td>${contract.reviewByDate || "N/A"}</td>

<td>${formattedValue}</td>

`;

// Highlight expired contracts

if (contract.expiryDate) {

const today = new Date();

today.setHours(0, 0, 0, 0);

const expiryDate = new Date(contract.expiryDate);

if (expiryDate <= today) {

tr.style.backgroundColor = "#ffcdd2";

tr.style.color = "#c62828";

}

}

tbody.appendChild(tr);

});

} else {

tbody.innerHTML =

'<tr><td colspan="8" style="text-align: center;">No contracts found</td></tr>';

}

} catch (err) {

console.error("Error loading contracts:", err);

const tbody = document.querySelector("#contracts-table tbody");

if (tbody) {

tbody.innerHTML = `<tr><td colspan="8" style="text-align: center; color: red;">Error: ${err.message}</td></tr>`;

}

}

}

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

loadContracts();

// CSV Export

document

.getElementById("download-csv")

?.addEventListener("click", function () {

const rows = [

[

"Contract ID",

"Parties",

"Type of Contract",

"Duration",

"Description",

"Expiry Date",

"Review By",

"Contract Value",

],

];

document.querySelectorAll("#contracts-table tbody tr").forEach((tr) => {

const cols = tr.querySelectorAll("td");

const row = Array.from(cols).map((td) => td.innerText.trim());

rows.push(row);

});

const csvContent =

"data:text/csv;charset=utf-8," +

rows

.map((r) => r.map((v) => `"${v.replace(/"/g, '""')}"`).join(","))

.join("\n");

const downloadLink = document.createElement("a");

downloadLink.href = encodeURI(csvContent);

downloadLink.download = "contracts.csv";

document.body.appendChild(downloadLink);

downloadLink.click();

document.body.removeChild(downloadLink);

});

// Search filter

const searchInput = document.querySelector(".search-bar");

const tableBody = document.querySelector("#contracts-table tbody");

function filterContracts() {

const searchText = searchInput?.value.toLowerCase() || "";

Array.from(tableBody.querySelectorAll("tr")).forEach((row) => {

const party = row.children[1].textContent.toLowerCase();

const type = row.children[2].textContent.toLowerCase();

let matchesSearch =

!searchText || party.includes(searchText) || type.includes(searchText);

row.style.display = matchesSearch ? "" : "none";

});

}

searchInput?.addEventListener("input", filterContracts);

});

"

frontend/css/manage\_contract.css ".user-info {

display: flex;

align-items: center;

gap: 1rem;

margin-left: auto;

padding-right: 1rem;

}

.user-badge {

background: #007bff;

color: white;

padding: 4px 8px;

border-radius: 12px;

font-size: 0.8rem;

font-weight: 500;

text-transform: uppercase;

}

.user-badge.admin {

background: #dc3545;

}

.user-badge.manager {

background: #28a745;

}

.user-badge.user {

background: #6c757d;

}

.logout-btn {

background: #dc3545;

color: white;

border: none;

padding: 8px 16px;

border-radius: 4px;

cursor: pointer;

font-size: 0.9rem;

}

.logout-btn:hover {

background: #c82333;

}

.access-denied {

text-align: center;

padding: 3rem;

background: #f8d7da;

border: 1px solid #f5c6cb;

color: #721c24;

border-radius: 8px;

}

.tabs .tab {

color: #fff;

background: none;

border: none;

padding: 10px 24px;

margin-right: 4px;

border-radius: 4px 4px 0 0;

cursor: pointer;

font-size: 1rem;

transition: background 0.2s, color 0.2s;

outline: none;

}

.tabs .tab:not(.active) {

background: none;

color: #fff;

opacity: 0.8;

}

.tabs .tab.active {

background: #fff;

color: #007bff;

opacity: 1;

font-weight: bold;

border-bottom: 2px solid #007bff;

}"

frontend/css/style.css "/\* ---------- General Styles ---------- \*/

body {

background-color: #2c3e50;

color: #ecf0f1;

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

/\* ---------- Navbar ---------- \*/

.navbar {

width: 100%;

margin: 0 auto;

background-color: #34495e;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

display: flex;

justify-content: space-between;

align-items: center;

padding: 10px 10px;

border-radius: 10px;

}

/\* Links container \*/

.navbar .nav-links {

display: flex;

list-style: none;

gap: 5px; /\* space between links \*/

margin: 0;

padding: 0;

}

/\* Individual links \*/

.navbar .nav-links li a {

color: #ecf0f1;

text-decoration: none; /\* remove underline \*/

padding: 10px 15px;

border-radius: 5px;

transition: background-color 0.3s;

}

.navbar .nav-links li a:hover {

background-color: #2980b9;

color: #ecf0f1;

}

.navbar .nav-links li a.active {

background-color: #007bff;

color: #fff;

}

/\* Logo styling \*/

.navbar .nwr\_logo img {

display: block;

margin-right: 25px;

height: 60px;

}

/\* Responsive for small screens \*/

@media (max-width: 768px) {

.navbar {

flex-direction: column;

align-items: flex-start;

}

.navbar .nav-links {

flex-direction: column;

width: 100%;

gap: 0;

margin-bottom: 10px;

}

.navbar .nav-links li a {

width: 100%;

text-align: left;

}

.navbar .nwr\_logo {

align-self: center;

margin-top: 10px;

}

}

/\* ---------- Containers ---------- \*/

.container {

padding: 20px;

text-align: center;

}

/\* ---------- Headings ---------- \*/

h1, h2 {

color: #abc6d1;

}

/\* ---------- Forms ---------- \*/

.form-group {

margin: 10px 0;

text-align: left;

width: 70%;

margin-left: auto;

margin-right: auto;

}

.form-group label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

.form-group input {

width: 100%;

padding: 8px;

border: none;

border-radius: 5px;

background-color: #0758aa;

color: #ecf0f1;

}

.form-group input:focus {

outline: none;

background-color: #0750a0;

}

/\* ---------- Login Form ---------- \*/

.login-form {

background-color: #34495e;

padding: 30px 40px;

border-radius: 10px;

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);

width: 100%;

max-width: 400px;

margin: 20px auto;

text-align: left;

color: #ecf0f1;

}

.login-form button {

width: 100%;

padding: 12px;

background-color: #3498db;

color: #ecf0f1;

border-radius: 5px;

border: none;

cursor: pointer;

font-size: 16px;

transition: all 0.3s;

}

.login-form button:hover {

background-color: #2980b9;

}

/\* ---------- Search Bar ---------- \*/

.search-bar {

width: 70%;

padding: 10px;

margin: 10px 0;

border: none;

border-radius: 5px;

background-color: #e0e0e0;

color: #25343b;

}

/\* ---------- Notes ---------- \*/

.note {

font-size: 0.9em;

color: #bdc3c7;

}

/\* ---------- Tables ---------- \*/

table {

width: 90%;

margin: 20px auto;

border-collapse: collapse;

}

th, td {

padding: 10px;

border: 1px solid #7f8c8d;

}

th {

background-color: #34495e;

}

/\* ---------- Buttons ---------- \*/

button, .btn {

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 14px;

transition: all 0.3s;

}

button:hover, .btn:hover {

background-color: #2980b9;

opacity: 0.9;

transform: translateY(-1px);

}

.btn-primary {

background-color: #007bff;

color: white;

}

.btn-success {

background-color: #28a745;

color: white;

}

.btn-warning {

background-color: #ffc107;

color: #212529;

}

.btn-danger {

background-color: #dc3545;

color: white;

}

.btn:disabled {

opacity: 0.6;

cursor: not-allowed;

transform: none;

}

/\* ---------- Tabs ---------- \*/

.tabs {

display: flex;

margin-bottom: 20px;

border-bottom: 2px solid #ddd;

}

.tab {

padding: 12px 24px;

cursor: pointer;

border: none;

background: none;

font-size: 16px;

border-bottom: 3px solid transparent;

transition: all 0.3s;

}

.tab.active {

border-bottom-color: #007bff;

color: #007bff;

font-weight: bold;

}

.tab:hover {

background-color: #2980b9;

}

.tab-content {

display: none;

}

.tab-content.active {

display: block;

}

/\* ---------- Form Rows ---------- \*/

.form-row {

display: flex;

gap: 20px;

margin-bottom: 15px;

}

.form-row .form-group {

flex: 1;

margin-bottom: 0;

}

/\* ---------- Contract List ---------- \*/

.contract-list {

max-height: 400px;

overflow-y: auto;

border: 1px solid #ddd;

border-radius: 5px;

background-color: #f9f9f9;

}

.contract-item {

padding: 15px;

border-bottom: 1px solid #eee;

cursor: pointer;

transition: background-color 0.2s;

}

.contract-item:hover {

background-color: white;

}

.contract-item.selected {

background-color: #e3fefd;

border-left: 4px solid #007bff;

}

.contract-item h4 {

margin: 0 0 5px 0;

color: #333;

}

.contract-item p {

margin: 0;

color: #666;

font-size: 14px;

}

/\* ---------- Button Groups ---------- \*/

.button-group {

display: flex;

gap: 10px;

margin-top: 20px;

}

/\* ---------- Messages ---------- \*/

.message {

padding: 10px;

border-radius: 5px;

margin: 10px 0;

display: none;

}

.message.success {

background-color: #d4edda;

border: 1px solid #c3e6cb;

color: #155724;

}

.message.error {

background-color: #f8d7da;

border: 1px solid #f5c6cb;

color: #721c24;

}

/\* ---------- Loading ---------- \*/

.loading {

text-align: center;

color: #666;

font-style: italic;

}

/\* ---------- Warnings ---------- \*/

.warning {

background-color: #7d8a3e;

padding: 10px;

margin: 10px 0;

border-radius: 5px;

}

/\* ---------- Responsive Design ---------- \*/

@media (max-width: 768px) {

.tabs {

flex-direction: column;

}

.contract-list {

max-height: 300px;

}

.form-row {

flex-direction: column;

gap: 10px;

}

.form-row .form-group {

width: 100%;

}

}"

backend/controllers/ContractController.php "<?php

class ContractController {

private $model;

public function \_\_construct($pdo) {

require\_once \_\_DIR\_\_ . '/../models/Contract.php';

require\_once \_\_DIR\_\_ . '/../config/auth.php';

$this->model = new Contract($pdo);

}

// fecth all

public function list() {

return $this->model->getAll();

}

// Filter by contractid

public function view($id) {

return $this->model->getById($id);

}

// Create contract

public function create($data) {

$user = Auth::getCurrentUser();

if (!$user) {

throw new Exception("User not authenticated");

}

$userid = $user['userid'];

$newId = $this->model->create($data, $userid);

if ($newId) {

return ['success' => true, 'contractid' => $newId];

} else {

return ['success' => false, 'error' => 'Contract creation failed'];

}

}

// Update contract

public function update($id, $data) {

$user = Auth::getCurrentUser();

if (!$user) {

return ['success' => false, 'error' => 'User not logged in'];

}

$userid = $user['userid'];

$success = $this->model->update($id, $data, $userid);

return ['success' => $success];

}

// Delete contract

public function delete($id) {

return $this->model->delete($id);

}

}

"

backend/modeles/Contract.php "<?php

// ============================================

// Contract.php

// Handles contract CRUD + email notifications

// ============================================

use PHPMailer\PHPMailer\PHPMailer;

use PHPMailer\PHPMailer\Exception;

// Include PHPMailer files

require \_\_DIR\_\_ . '/../PHPMailer/PHPMailer.php';

require \_\_DIR\_\_ . '/../PHPMailer/SMTP.php';

require \_\_DIR\_\_ . '/../PHPMailer/Exception.php';

class Contract

{

private $pdo;

private $table = 'contracts';

public function \_\_construct($pdo)

{

$this->pdo = $pdo;

}

// --------------------

// CRUD Methods

// --------------------

public function getAllContracts()

{

// Updated to fetch manager email

$sql = "SELECT c.\*, u.email AS manager\_email, u.username AS manager\_name

FROM {$this->table} c

INNER JOIN users u ON c.manager\_id = u.userid

ORDER BY c.expiryDate ASC";

$stmt = $this->pdo->prepare($sql);

$stmt->execute();

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

public function getAll()

{

$sql = "SELECT contractid, parties, typeOfContract, duration, description, filepath, expiryDate, reviewByDate, contractValue, manager\_id

FROM {$this->table}";

$stmt = $this->pdo->query($sql);

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

public function getById($id)

{

$stmt = $this->pdo->prepare("SELECT \* FROM {$this->table} WHERE contractid = :id");

$stmt->execute(['id' => $id]);

return $stmt->fetch(PDO::FETCH\_ASSOC);

}

public function create($data, $userid)

{

$fields = [];

$placeholders = [];

$values = [];

foreach (['parties','typeOfContract','duration','contractValue','description','expiryDate','reviewByDate'] as $col) {

if (isset($data[$col])) {

$fields[] = $col;

$placeholders[] = '?';

$values[] = $data[$col];

}

}

if (!empty($data['filepath'])) {

$fields[] = "filepath";

$placeholders[] = "?";

$values[] = $data['filepath'];

}

// Add manager\_id

$fields[] = "manager\_id";

$placeholders[] = "?";

$values[] = $userid;

$sql = "INSERT INTO contracts (" . implode(", ", $fields) . ")

VALUES (" . implode(", ", $placeholders) . ")";

$stmt = $this->pdo->prepare($sql);

if ($stmt->execute($values)) {

return $this->pdo->lastInsertId();

}

error\_log("Creating contract for user $userid with data: " . json\_encode($data));

return false;

}

public function update($id, $data, $userid)

{

$fields = [];

$values = [];

foreach (['parties','typeOfContract','duration','contractValue','description','expiryDate','reviewByDate'] as $col) {

if (isset($data[$col])) {

$fields[] = "$col = ?";

$values[] = $data[$col];

}

}

// File upload handling

if (isset($\_FILES['contractFile']) && $\_FILES['contractFile']['error'] === UPLOAD\_ERR\_OK) {

$uploadDir = \_\_DIR\_\_ . '/../uploads/';

if (!is\_dir($uploadDir)) mkdir($uploadDir, 0777, true);

$partiesName = isset($data['parties']) ? preg\_replace('/[\/\\\\:\*?"<>|]/', '\_', $data['parties']) : 'contract';

$extension = pathinfo($\_FILES['contractFile']['name'], PATHINFO\_EXTENSION);

$newFileName = $partiesName . "\_" . time() . "." . $extension;

$fullPath = $uploadDir . $newFileName;

if (move\_uploaded\_file($\_FILES['contractFile']['tmp\_name'], $fullPath)) {

$fields[] = "filepath = ?";

$values[] = 'uploads/' . $newFileName;

}

}

if (empty($fields)) return false;

$values[] = $id;

$sql = "UPDATE contracts SET " . implode(", ", $fields) . " WHERE contractid = ?";

$stmt = $this->pdo->prepare($sql);

return $stmt->execute($values);

}

public function delete($id)

{

$sql = "DELETE FROM {$this->table} WHERE contractid = :id";

$stmt = $this->pdo->prepare($sql);

return $stmt->execute(['id' => $id]);

}

// --------------------

// Email Notification Methods

// --------------------

public function sendExpiryNotifications()

{

$this->checkAndSend(90, 'weekly'); // 3 months

$this->checkAndSend(60, 'twice'); // 2 months

$this->checkAndSend(30, 'daily'); // 1 month

}

private function checkAndSend($days, $frequency)

{

if (!$this->shouldSend($frequency)) return;

$sql = "SELECT c.\*, u.email AS manager\_email

FROM {$this->table} c

INNER JOIN users u ON c.manager\_id = u.userid

WHERE c.expiryDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL :days DAY)";

$stmt = $this->pdo->prepare($sql);

$stmt->execute(['days' => $days]);

$contracts = $stmt->fetchAll(PDO::FETCH\_ASSOC);

foreach ($contracts as $contract) {

$recipientEmail = $contract['manager\_email'];

if (filter\_var($recipientEmail, FILTER\_VALIDATE\_EMAIL)) {

$this->sendEmailNotification(

$recipientEmail,

$contract['typeOfContract'],

$contract['expiryDate']

);

}

}

}

private function shouldSend($frequency)

{

$day = date('N'); // 1=Monday, 7=Sunday

if ($frequency === 'weekly') return $day == 1;

if ($frequency === 'twice') return ($day == 1 || $day == 4);

if ($frequency === 'daily') return true;

return false;

}

// Public so it can be tested directly

public function sendEmailNotification($recipientEmail, $contractType, $expiryDate)

{

$mail = new PHPMailer(true);

try {

// ===== Sender info =====

$senderEmail = 'uraniathomas23@gmail.com'; // Gmail sender

$senderName = 'Contract Registry';

$senderPassword = 'lkmkivxthjizqojc'; // Gmail App Password

$mail->isSMTP();

$mail->Host = 'smtp.gmail.com';

$mail->SMTPAuth = true;

$mail->Username = $senderEmail;

$mail->Password = $senderPassword;

$mail->SMTPSecure = PHPMailer::ENCRYPTION\_STARTTLS;

$mail->Port = 587;

$mail->setFrom($senderEmail, $senderName);

$mail->addAddress($recipientEmail);

$mail->isHTML(false);

$mail->Subject = 'Contract Expiry Notification';

$mail->Body = "Hello,\n\nThe contract of '{$contractType}' is set to expire on {$expiryDate}.\nPlease take necessary action before it expires.\n\nThank you. \n\nContract Registry System";

$mail->send();

return true;

} catch (Exception $e) {

error\_log("Mailer Error for {$recipientEmail}: {$mail->ErrorInfo}");

return false;

}

}

}

"

backend/index.php "<?php

session\_start();

header("Content-Type: application/json");

require\_once \_\_DIR\_\_ . "/config/db.php";

require\_once \_\_DIR\_\_ . "/config/auth.php";

require\_once \_\_DIR\_\_ . "/controllers/ContractController.php";

require\_once \_\_DIR\_\_ . "/controllers/UserController.php";

Auth::init($pdo);

$controller = new ContractController($pdo);

$userController = new UserController($pdo);

// Function to handle file upload

function handleFileUpload($file, $partiesName)

{

$uploadDir = \_\_DIR\_\_ . '/uploads/';

if (!is\_dir($uploadDir)) {

mkdir($uploadDir, 0777, true);

}

// Sanitize parties name

$safeParties = preg\_replace('/[\/\\\\:\*?"<>|]/', '\_', $partiesName ?: 'contract');

// Get original file extension

$extension = pathinfo($file['name'], PATHINFO\_EXTENSION);

// Create new filename

$newFileName = $safeParties . "\_" . time() . "." . $extension;

$targetPath = $uploadDir . $newFileName;

if (move\_uploaded\_file($file['tmp\_name'], $targetPath)) {

return 'uploads/' . $newFileName; // relative path

}

return false; // upload failed

}

// Decide action from query string

$action = $\_GET['action'] ?? 'list';

try {

switch ($action) {

// --------------------Contract Management--------------------

// Contract actions - require authentication

case 'list':

Auth::requireLogin(); // All logged in users can view dashboard

echo json\_encode($controller->list());

break;

case 'view':

Auth::requireLogin();

$id = $\_GET['id'] ?? null;

if ($id) {

echo json\_encode($controller->view($id));

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Contract ID required"]);

}

break;

case 'create':

Auth::requireRole('manager'); // Only managers and admins can create contracts

$data = $\_POST;

if (!empty($\_FILES['contractFile']['name'])) {

$filepath = handleFileUpload($\_FILES['contractFile'], $data['parties'] ?? '');

if ($filepath) {

$data['filepath'] = $filepath;

} else {

http\_response\_code(500);

echo json\_encode(["error" => "File upload failed"]);

exit;

}

}

if ($controller->create($data)) {

echo json\_encode(["message" => "Contract created"]);

} else {

http\_response\_code(500);

echo json\_encode(["error" => "Failed to create contract"]);

}

break;

case 'update':

Auth::requireRole('manager'); // Only managers and admins can edit contracts

$id = $\_GET['id'] ?? null;

if ($id) {

$data = $\_POST;

if (!empty($\_FILES['contractFile']['name'])) {

$filepath = handleFileUpload($\_FILES['contractFile'], $data['parties'] ?? '');

if ($filepath) {

$data['filepath'] = $filepath;

} else {

http\_response\_code(500);

echo json\_encode(["error" => "File upload failed"]);

exit;

}

}

if ($controller->update($id, $data)) {

echo json\_encode(["message" => "Contract updated"]);

} else {

http\_response\_code(500);

echo json\_encode(["error" => "Failed to update contract"]);

}

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Contract ID required"]);

}

break;

case 'delete':

Auth::requireRole('manager'); // Only managers and admins can delete contracts

$id = $\_GET['id'] ?? null;

if ($id && $controller->delete($id)) {

echo json\_encode(["message" => "Contract deleted"]);

} else {

http\_response\_code(500);

echo json\_encode(["error" => "Failed to delete contract"]);

}

break;

// --------------------User Management--------------------

// User management actions - admin only

case 'users':

Auth::requireRole('admin'); // Only admins can manage users

echo json\_encode($userController->getAllUsers());

break;

case 'update\_user\_role':

Auth::requireRole('admin'); // Only admins can change user roles

$userId = $\_POST['user\_id'] ?? null;

$newRole = $\_POST['role'] ?? null;

if (!$userId || !$newRole) {

http\_response\_code(400);

echo json\_encode(["error" => "User ID and role required"]);

break;

}

if (!in\_array($newRole, ['user', 'manager', 'admin'])) {

http\_response\_code(400);

echo json\_encode(["error" => "Invalid role"]);

break;

}

if ($userController->updateUserRole($userId, $newRole)) {

echo json\_encode(["message" => "User role updated successfully"]);

} else {

http\_response\_code(500);

echo json\_encode(["error" => "Failed to update user role"]);

}

break;

case 'delete\_user':

Auth::requireRole('admin'); // Only admins can delete users

$userId = $\_POST['user\_id'] ?? null;

if (!$userId) {

http\_response\_code(400);

echo json\_encode(["error" => "User ID required"]);

break;

}

// Prevent admin from deleting themselves

if ($userId == $\_SESSION['user\_id']) {

http\_response\_code(400);

echo json\_encode(["error" => "Cannot delete your own account"]);

break;

}

if ($userController->deleteUser($userId)) {

echo json\_encode(["message" => "User deleted successfully"]);

} else {

http\_response\_code(500);

echo json\_encode(["error" => "Failed to delete user"]);

}

break;

case 'user\_stats':

Auth::requireRole('admin'); // Only admins can view user stats

echo json\_encode($userController->getUserStats());

break;

default:

http\_response\_code(400);

echo json\_encode(["error" => "Invalid action"]);

}

} catch (Exception $e) {

http\_response\_code(500);

echo json\_encode(["error" => $e->getMessage()]);

}

?>"

the view and download buttons are not vesible in the edit tab, even when there is a file attached to the contract, can you assit in solving this issue.