// script.js

const signupForm = document.getElementById('signup-form');

const loginForm = document.getElementById('login-form');

const addExpenseForm = document.getElementById('add-expense-form');

const authSection = document.getElementById('auth');

const mainAppSection = document.getElementById('main-app');

const expensesTable = document.getElementById('expenses-table');

const expenseChartCanvas = document.getElementById('expense-chart');

let expenses = [];

let currentUser = null;

// Authentication logic

signupForm.addEventListener('submit', (e) => {

e.preventDefault();

const username = document.getElementById('signup-username').value;

const password = document.getElementById('signup-password').value;

localStorage.setItem(username, password);

alert('Sign Up Successful!');

signupForm.reset();

});

loginForm.addEventListener('submit', (e) => {

e.preventDefault();

const username = document.getElementById('login-username').value;

const password = document.getElementById('login-password').value;

const storedPassword = localStorage.getItem(username);

if (storedPassword === password) {

alert('Login Successful!');

currentUser = username;

authSection.classList.add('hidden');

mainAppSection.classList.remove('hidden');

} else {

alert('Invalid credentials.');

}

loginForm.reset();

});

// Expense management

addExpenseForm.addEventListener('submit', (e) => {

e.preventDefault();

const category = document.getElementById('expense-category').value;

const amount = parseFloat(document.getElementById('expense-amount').value);

const comments = document.getElementById('expense-comments').value;

const createdAt = new Date().toLocaleString();

const expense = {

category,

amount,

comments,

createdAt,

updatedAt: createdAt,

};

expenses.push(expense);

renderExpenses();

renderChart();

addExpenseForm.reset();

});

function renderExpenses() {

expensesTable.innerHTML = '';

expenses.forEach((expense, index) => {

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

row.innerHTML = `

<td>${expense.category}</td>

<td>${expense.amount}</td>

<td>${expense.createdAt}</td>

<td>${expense.updatedAt}</td>

<td>${expense.comments}</td>

<td>

<button onclick="editExpense(${index})">Edit</button>

<button onclick="deleteExpense(${index})">Delete</button>

</td>

`;

expensesTable.appendChild(row);

});

}

function editExpense(index) {

const expense = expenses[index];

const newCategory = prompt('Edit Category', expense.category);

const newAmount = parseFloat(prompt('Edit Amount', expense.amount));

const newComments = prompt('Edit Comments', expense.comments);

if (newCategory && !isNaN(newAmount)) {

expenses[index] = {

...expense,

category: newCategory,

amount: newAmount,

comments: newComments || expense.comments,

updatedAt: new Date().toLocaleString(),

};

renderExpenses();

renderChart();

}

}

function deleteExpense(index) {

expenses.splice(index, 1);

renderExpenses();

renderChart();

}

function renderChart() {

const ctx = expenseChartCanvas.getContext('2d');

const categoryTotals = expenses.reduce((totals, expense) => {

totals[expense.category] = (totals[expense.category] || 0) + expense.amount;

return totals;

}, {});

const data = {

labels: Object.keys(categoryTotals),

datasets: [{

data: Object.values(categoryTotals),

backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0'],

}],

};

new Chart(ctx, {

type: 'pie',

data,

});

}