<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>InvestPro - Complete Earning Platform</title>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600&display=swap" rel="stylesheet" />

<!-- Flutterwave Checkout Script (still used for one deposit option) -->

<script src="https://checkout.flutterwave.com/v3.js"></script>

<style>

/\* Colour Scheme: Green (Primary), Yellow (Secondary), Red (Error) \*/

:root {

--primary-color: #28a745;

--secondary-color: #ffc107;

--error-color: #dc3545;

--background-color: #ffffff;

--text-color: #333333;

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

--transition-speed: 0.3s;

}

/\* Global Styles \*/

body {

margin: 0;

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

background: var(--background-color);

color: var(--text-color);

line-height: 1.6;

scroll-behavior: smooth;

}

a { text-decoration: none; color: var(--primary-color); }

@keyframes fadeIn {

from { opacity: 0; transform: scale(0.95); }

to { opacity: 1; transform: scale(1); }

}

/\* Modal, Link Bank, and Transfer modals \*/

.modal, .link-bank-modal, .transfer-modal {

position: fixed;

top: 0; left: 0;

width: 100%; height: 100%;

background: rgba(0,0,0,0.4);

display: none;

justify-content: center;

align-items: center;

z-index: 2100;

}

.modal .modal-content,

.link-bank-modal .modal-content,

.transfer-modal .modal-content {

background: var(--background-color);

color: var(--text-color);

padding: 20px;

border-radius: 8px;

max-width: 400px;

width: 90%;

text-align: center;

box-shadow: var(--shadow);

animation: fadeIn var(--transition-speed) ease;

}

.modal-content p { margin-bottom: 20px; }

.modal-content input {

width: 80%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 1rem;

background: var(--background-color);

color: var(--text-color);

}

.modal-content button {

padding: 10px 20px;

border: none;

border-radius: 5px;

background: var(--primary-color);

color: var(--background-color);

cursor: pointer;

transition: background var(--transition-speed);

margin: 5px;

}

.modal-content button:hover { background: #218838; }

/\* Landing Page \*/

#landingSection {

background: linear-gradient(90deg, var(--primary-color), var(--secondary-color));

min-height: 100vh;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

text-align: center;

padding: 2rem;

color: var(--background-color);

}

#landingSection h1 { font-size: 3rem; margin-bottom: 1rem; }

#landingSection p { font-size: 1.3rem; margin-bottom: 2rem; max-width: 600px; }

.cta-button {

background: var(--secondary-color);

color: var(--background-color);

padding: 1rem 2rem;

border: none;

border-radius: 5px;

font-size: 1.2rem;

cursor: pointer;

transition: background var(--transition-speed);

}

.cta-button:hover { background: #e0a800; }

/\* Auth Section \*/

#authSection {

display: none;

min-height: 100vh;

background: var(--background-color);

flex-direction: column;

justify-content: center;

align-items: center;

padding: 2rem;

}

.auth-container {

background: var(--background-color);

padding: 30px;

border-radius: 10px;

box-shadow: var(--shadow);

width: 300px;

text-align: center;

margin-bottom: 1rem;

color: var(--text-color);

border: 1px solid #ccc;

}

.auth-container h2 { margin-bottom: 20px; color: var(--primary-color); }

.auth-container input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

background: var(--background-color);

color: var(--text-color);

}

.auth-container button {

width: 100%;

padding: 10px;

background: var(--secondary-color);

border: none;

border-radius: 5px;

color: var(--background-color);

font-weight: bold;

cursor: pointer;

transition: background var(--transition-speed);

}

.auth-container button:hover { background: #e0a800; }

.error { color: var(--error-color); margin-top: 10px; display: none; }

.toggle-link, .forgot-link { margin-top: 15px; font-size: 0.9rem; }

.toggle-link a, .forgot-link a { color: var(--primary-color); cursor: pointer; }

.loading { margin-top: 10px; display: none; }

/\* Dashboard Section \*/

#dashboardSection { display: none; min-height: 100vh; }

.sidebar {

position: fixed;

top: 0; left: 0;

width: 250px;

height: 100vh;

background: var(--primary-color);

color: var(--background-color);

padding: 20px;

overflow-y: auto;

transition: width var(--transition-speed);

}

.sidebar h2 { text-align: center; font-size: 24px; margin-bottom: 20px; }

.sidebar ul { list-style: none; padding: 0; }

.sidebar ul li {

padding: 10px;

margin: 10px 0;

background: rgba(255,255,255,0.2);

text-align: center;

cursor: pointer;

border-radius: 5px;

transition: background var(--transition-speed);

}

.sidebar ul li:hover { background: rgba(255,255,255,0.3); }

.main-content {

margin-left: 270px;

padding: 20px;

transition: margin-left var(--transition-speed);

}

.section {

display: none;

background: var(--background-color);

padding: 20px;

border-radius: 10px;

box-shadow: var(--shadow);

margin-bottom: 20px;

color: var(--text-color);

border: 1px solid #ccc;

animation: fadeIn var(--transition-speed) ease;

}

.dashboard-header { text-align: center; margin-bottom: 20px; }

.dashboard-header h1 { font-size: 28px; margin-bottom: 5px; }

.dashboard-header p { font-size: 16px; color: var(--secondary-color); }

.stats { display: flex; justify-content: space-around; flex-wrap: wrap; margin-bottom: 20px; }

.stat-card {

flex: 1 1 200px;

background: var(--background-color);

padding: 20px;

text-align: center;

border-radius: 10px;

box-shadow: var(--shadow);

margin: 10px;

color: var(--text-color);

border: 1px solid #ccc;

}

.stat-card h3 { font-size: 20px; color: var(--primary-color); margin-bottom: 10px; }

.stat-card p { font-size: 24px; font-weight: 600; }

.cta-buttons { display: flex; justify-content: center; gap: 10px; margin: 20px 0; flex-wrap: wrap; }

.cta-buttons button {

padding: 10px 20px;

background: var(--primary-color);

color: var(--background-color);

border: none;

border-radius: 5px;

cursor: pointer;

transition: background var(--transition-speed);

font-size: 1rem;

}

.cta-buttons button:hover { background: #218838; }

/\* Bonus withdrawal buttons (daily interest & referral bonus) \*/

.cta-buttons .bonus-btn {

background: var(--secondary-color);

}

/\* Transaction History \*/

.transaction-history { margin-top: 20px; }

.transaction-history h3 { color: var(--primary-color); margin-bottom: 10px; text-align: center; }

.transaction-table {

width: 100%;

border-collapse: collapse;

}

.transaction-table th, .transaction-table td {

padding: 12px;

border: 1px solid #ccc;

text-align: left;

}

.transaction-table th { background: #f8f9fa; color: var(--text-color); }

/\* Skeleton Loader for Transactions \*/

.skeleton {

background-color: #e0e0e0;

border-radius: 4px;

display: inline-block;

width: 100%;

height: 1em;

animation: shimmer 1.5s infinite;

}

@keyframes shimmer {

0% { background-position: -1000px 0; }

100% { background-position: 1000px 0; }

}

.plans-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 20px;

margin-top: 20px;

}

.plan-card {

background: var(--background-color);

padding: 20px;

border-radius: 10px;

box-shadow: var(--shadow);

text-align: center;

transition: transform var(--transition-speed);

color: var(--text-color);

border: 1px solid #ccc;

}

.plan-card:hover { transform: scale(1.02); }

.plan-card h2 { color: var(--primary-color); margin-bottom: 10px; font-size: 1.5rem; }

.plan-card p { margin: 5px 0; color: #666666; font-size: 1rem; }

.invest-btn {

margin-top: 10px;

padding: 10px 20px;

background: var(--secondary-color);

color: var(--background-color);

border: none;

border-radius: 5px;

cursor: pointer;

transition: transform var(--transition-speed);

font-size: 1rem;

}

.invest-btn:hover { transform: scale(1.05); }

/\* Profile Section \*/

#profile {

display: none;

background: var(--background-color);

padding: 20px;

border-radius: 10px;

box-shadow: var(--shadow);

margin-bottom: 20px;

color: var(--text-color);

border: 1px solid #ccc;

}

#profile h2 { color: var(--primary-color); margin-bottom: 1rem; }

#profile form {

display: flex;

flex-direction: column;

align-items: center;

max-width: 400px;

margin: auto;

}

#profile form input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

background: var(--background-color);

color: var(--text-color);

}

#profile form button {

width: 100%;

padding: 10px;

background: var(--secondary-color);

color: var(--background-color);

border: none;

border-radius: 5px;

cursor: pointer;

transition: background var(--transition-speed);

font-size: 1rem;

}

#profile form button:hover { background: #e0a800; }

/\* Referral Section \*/

#referral {

display: none;

background: var(--background-color);

padding: 20px;

border-radius: 10px;

box-shadow: var(--shadow);

margin-bottom: 20px;

color: var(--text-color);

text-align: center;

border: 1px solid #ccc;

}

#referral h2 { color: var(--primary-color); margin-bottom: 1rem; }

#referral input {

width: 80%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

text-align: center;

background: var(--background-color);

color: var(--text-color);

}

#referral button {

padding: 10px 20px;

background: var(--secondary-color);

color: var(--background-color);

border: none;

border-radius: 5px;

cursor: pointer;

transition: background var(--transition-speed);

font-size: 1rem;

}

#referral button:hover { background: #e0a800; }

/\* New: Peer-to-Peer Transfer Modal \*/

.transfer-modal {

position: fixed;

top: 0; left: 0;

width: 100%; height: 100%;

background: rgba(0,0,0,0.4);

display: none;

justify-content: center;

align-items: center;

z-index: 2100;

}

.transfer-modal .modal-content {

max-width: 400px;

width: 90%;

}

/\* New: Link Bank Account Modal \*/

.link-bank-modal {

position: fixed;

top: 0; left: 0;

width: 100%; height: 100%;

background: rgba(0,0,0,0.4);

display: none;

justify-content: center;

align-items: center;

z-index: 2100;

}

.link-bank-modal .modal-content {

max-width: 400px;

width: 90%;

}

/\* Footer \*/

footer {

background: #000000;

color: var(--background-color);

text-align: center;

padding: 1rem 0;

margin-top: 2rem;

}

/\* Notification \*/

#notification {

position: fixed;

top: 20px;

right: 20px;

padding: 10px 20px;

border-radius: 5px;

color: var(--background-color);

display: none;

z-index: 3000;

}

/\* Responsive \*/

@media (max-width: 768px) {

.sidebar { width: 60px; padding: 10px; }

.sidebar h2 { display: none; }

.sidebar ul li { padding: 8px; font-size: 14px; }

.main-content { margin-left: 80px; }

.stats { flex-direction: column; align-items: center; }

}

</style>

</head>

<body>

<!-- Landing Section -->

<div id="landingSection">

<h1>InvestPro</h1>

<p>Invest in tailored plans and earn daily withdrawals. Our secure and transparent platform empowers you to grow your wealth effortlessly.</p>

<button class="cta-button" id="getStartedBtn" aria-label="Get Started">Get Started</button>

</div>

<!-- Auth Section: Login & Sign Up -->

<div id="authSection">

<!-- Login Container -->

<div class="auth-container" id="loginContainer" aria-label="Login Form">

<h2>Login</h2>

<input type="email" id="loginEmail" placeholder="Email" required aria-label="Email">

<input type="password" id="loginPassword" placeholder="Password" required aria-label="Password">

<button id="loginButton" aria-label="Login Button">Login</button>

<div class="loading" id="loginLoading">Loading...</div>

<div class="error" id="loginError"></div>

<div class="toggle-link">Don't have an account? <a id="showSignUp">Sign Up</a></div>

<div class="forgot-link">Forgot Password? <a id="forgotPassword">Reset</a></div>

</div>

<!-- Sign Up Container -->

<div class="auth-container" id="signupContainer" style="display:none;" aria-label="Sign Up Form">

<h2>Sign Up</h2>

<input type="text" id="signupName" placeholder="Full Name" required aria-label="Full Name">

<input type="email" id="signupEmail" placeholder="Email" required aria-label="Email">

<input type="password" id="signupPassword" placeholder="Password" required aria-label="Password">

<input type="password" id="signupConfirmPassword" placeholder="Confirm Password" required aria-label="Confirm Password">

<button id="signupButton" aria-label="Sign Up Button">Sign Up</button>

<div class="loading" id="signupLoading">Loading...</div>

<div class="error" id="signupError"></div>

<div class="toggle-link">Already have an account? <a id="showLogin">Login</a></div>

</div>

</div>

<!-- Onboarding Modal (shown once on first login) -->

<div class="modal" id="onboardingModal" role="dialog" aria-modal="true" aria-labelledby="onboardingTitle" style="display:none;">

<div class="modal-content">

<h2 id="onboardingTitle">Welcome to InvestPro!</h2>

<p>Get started by depositing funds, transferring to peers, and exploring our investment plans.</p>

<button id="onboardingCloseBtn" aria-label="Close Onboarding">Got it!</button>

</div>

</div>

<!-- Dashboard Section -->

<div id="dashboardSection">

<!-- Sidebar Navigation -->

<div class="sidebar" role="navigation">

<h2>User Panel</h2>

<ul>

<li onclick="showDashboardSection('overview')">Overview</li>

<li onclick="showDashboardSection('wallet')">Wallet</li>

<li onclick="showDashboardSection('investment')">Investments</li>

<li onclick="showDashboardSection('profile')">Profile</li>

<li onclick="showDashboardSection('referral')">Referral</li>

<li onclick="logout()">Logout</li>

</ul>

</div>

<!-- Main Content -->

<div class="main-content">

<!-- Overview Section -->

<div id="overview" class="section">

<div class="dashboard-header">

<h1>Welcome, <span id="userName">User</span>!</h1>

<p>Your earnings and stats overview</p>

</div>

<div class="stats">

<div class="stat-card">

<h3>Balance</h3>

<p id="balanceOverview">₦0.00</p>

</div>

<div class="stat-card">

<h3>Daily Earnings</h3>

<p id="dailyEarningsOverview">₦0.00</p>

</div>

<div class="stat-card">

<h3>Referral Bonus</h3>

<p id="referralBonusOverview">₦0.00</p>

</div>

</div>

</div>

<!-- Wallet Section -->

<div id="wallet" class="section">

<h1>User Wallet</h1>

<div class="balance-section">

<h2>₦<span id="walletBalance">0.00</span></h2>

<p>Available Balance</p>

</div>

<div class="cta-buttons">

<button id="depositBtn" aria-label="Deposit Funds via Flutterwave">Deposit Funds (Flutterwave)</button>

<button id="ufitpayDepositBtn" aria-label="Deposit Funds via UfitPay">Deposit with UfitPay</button>

<button id="peerDepositBtn" aria-label="Peer-to-Peer Deposit">Peer-to-Peer Deposit</button>

<button id="transferBtn" aria-label="Transfer Funds">Transfer Funds</button>

<button id="withdrawBtn" disabled aria-label="Withdraw Funds">Withdraw Funds</button>

<button id="withdrawDailyBtn" class="bonus-btn" aria-label="Withdraw Daily Interest">Withdraw Daily Interest</button>

<button id="withdrawReferralBtn" class="bonus-btn" aria-label="Withdraw Referral Bonus">Withdraw Referral Bonus</button>

<button id="linkBankBtn" aria-label="Link Bank Account">Link Bank Account</button>

</div>

<div class="transaction-history">

<h3>Transaction History</h3>

<table class="transaction-table" aria-label="Transaction History Table">

<thead>

<tr>

<th>Date</th>

<th>Description</th>

<th>Amount (₦)</th>

<th>Status</th>

</tr>

</thead>

<tbody id="transactionTableBody">

<!-- Skeleton Loader Rows -->

<tr>

<td><div class="skeleton" style="width: 80%;"></div></td>

<td><div class="skeleton" style="width: 90%;"></div></td>

<td><div class="skeleton" style="width: 60%;"></div></td>

<td><div class="skeleton" style="width: 70%;"></div></td>

</tr>

<tr>

<td><div class="skeleton" style="width: 80%;"></div></td>

<td><div class="skeleton" style="width: 90%;"></div></td>

<td><div class="skeleton" style="width: 60%;"></div></td>

<td><div class="skeleton" style="width: 70%;"></div></td>

</tr>

</tbody>

</table>

</div>

</div>

<!-- Investment Section -->

<div id="investment" class="section">

<h1>Investment Plans</h1>

<div class="plans-grid" id="plansGrid">

<!-- Investment plan cards will be rendered here -->

</div>

</div>

<!-- Profile Section -->

<div id="profile" class="section">

<h2>Your Profile</h2>

<form id="profileForm" aria-label="Profile Form">

<label for="profileName">Full Name:</label>

<input type="text" id="profileName" required aria-label="Full Name">

<label for="profileEmail">Email:</label>

<input type="email" id="profileEmail" required aria-label="Email">

<button type="submit" aria-label="Update Profile">Update Profile</button>

</form>

</div>

<!-- Referral Section -->

<div id="referral" class="section">

<h2>Your Referral Program</h2>

<p>Share your referral link to earn bonuses:</p>

<input type="text" id="referralLink" readonly aria-label="Referral Link">

<button onclick="copyReferralLink()" aria-label="Copy Referral Link">Copy Referral Link</button>

<h3>Your Referral History</h3>

<div id="referralHistory">

<p>No referrals yet.</p>

</div>

</div>

</div>

</div>

<!-- Generic Confirmation Modal -->

<div class="modal" id="modal" role="dialog" aria-modal="true" aria-labelledby="modalTitle">

<div class="modal-content">

<p id="modalMessage">Modal Message</p>

<button id="modalConfirm" aria-label="Confirm Modal">OK</button>

</div>

</div>

<!-- Deposit Modal (Flutterwave) -->

<div class="modal" id="depositModal" role="dialog" aria-modal="true" aria-labelledby="depositModalTitle">

<div class="modal-content">

<h3 id="depositModalTitle">Deposit Funds (Flutterwave)</h3>

<input type="number" id="depositAmountInput" placeholder="Enter amount in ₦" aria-label="Deposit Amount">

<div>

<button id="depositSubmitBtn" aria-label="Submit Deposit">Submit</button>

<button id="depositCancelBtn" aria-label="Cancel Deposit">Cancel</button>

</div>

</div>

</div>

<!-- Deposit Modal (UfitPay) -->

<div class="modal" id="ufitpayDepositModal" role="dialog" aria-modal="true" aria-labelledby="ufitpayModalTitle">

<div class="modal-content">

<h3 id="ufitpayModalTitle">Deposit Funds (UfitPay)</h3>

<input type="number" id="ufitpayDepositAmountInput" placeholder="Enter amount in ₦" aria-label="UfitPay Deposit Amount">

<div>

<button id="ufitpayDepositSubmitBtn" aria-label="Submit UfitPay Deposit">Submit</button>

<button id="ufitpayDepositCancelBtn" aria-label="Cancel UfitPay Deposit">Cancel</button>

</div>

</div>

</div>

<!-- Peer-to-Peer Deposit Modal -->

<div class="modal" id="peerDepositModal" role="dialog" aria-modal="true" aria-labelledby="peerDepositModalTitle">

<div class="modal-content">

<h3 id="peerDepositModalTitle">Peer-to-Peer Deposit</h3>

<p>Please transfer the desired amount from any bank to our account:</p>

<p><strong>Our Bank Details:</strong> Bank: XYZ Bank, Account Number: 1234567890, Account Name: InvestPro Ltd.</p>

<input type="number" id="peerDepositAmountInput" placeholder="Enter amount in ₦" aria-label="Peer Deposit Amount">

<input type="text" id="peerDepositRefInput" placeholder="Enter bank transfer reference" aria-label="Bank Transfer Reference">

<div>

<button id="peerDepositSubmitBtn" aria-label="Submit Peer Deposit">Submit</button>

<button id="peerDepositCancelBtn" aria-label="Cancel Peer Deposit">Cancel</button>

</div>

<p style="font-size: 0.9rem;">\* Your deposit will be reviewed and approved manually.</p>

</div>

</div>

<!-- Peer-to-Peer Transfer Modal -->

<div class="transfer-modal" id="transferModal" role="dialog" aria-modal="true" aria-labelledby="transferModalTitle">

<div class="modal-content">

<h3 id="transferModalTitle">Transfer Funds to a Peer</h3>

<input type="email" id="transferRecipientInput" placeholder="Enter recipient's email" aria-label="Recipient Email">

<input type="number" id="transferAmountInput" placeholder="Enter amount in ₦" aria-label="Transfer Amount">

<div>

<button id="transferSubmitBtn" aria-label="Submit Transfer">Submit</button>

<button id="transferCancelBtn" aria-label="Cancel Transfer">Cancel</button>

</div>

</div>

</div>

<!-- Link Bank Account Modal -->

<div class="link-bank-modal" id="linkBankModal" role="dialog" aria-modal="true" aria-labelledby="linkBankModalTitle">

<div class="modal-content">

<h3 id="linkBankModalTitle">Link Your Bank Account</h3>

<input type="text" id="bankNameInput" placeholder="Bank Name" aria-label="Bank Name">

<input type="text" id="accountNumberInput" placeholder="Account Number" aria-label="Account Number">

<input type="text" id="accountHolderInput" placeholder="Account Holder Name" aria-label="Account Holder Name">

<div>

<button id="linkBankSubmitBtn" aria-label="Submit Bank Account">Link Account</button>

<button id="linkBankCancelBtn" aria-label="Cancel Bank Account Linking">Cancel</button>

</div>

</div>

</div>

<!-- Withdraw Modal -->

<div class="modal" id="withdrawModal" role="dialog" aria-modal="true" aria-labelledby="withdrawModalTitle">

<div class="modal-content">

<h3 id="withdrawModalTitle">Withdraw Funds</h3>

<input type="number" id="withdrawAmountInput" placeholder="Enter amount in ₦" aria-label="Withdraw Amount">

<div>

<button id="withdrawSubmitBtn" aria-label="Submit Withdrawal">Submit</button>

<button id="withdrawCancelBtn" aria-label="Cancel Withdrawal">Cancel</button>

</div>

</div>

</div>

<!-- Notification -->

<div id="notification" role="alert"></div>

<!-- Footer -->

<footer>

<p>&copy; 2025 YourWallet. All Rights Reserved.</p>

</footer>

<!-- Firebase SDKs -->

<script src="https://www.gstatic.com/firebasejs/9.21.0/firebase-app-compat.js"></script>

<script src="https://www.gstatic.com/firebasejs/9.21.0/firebase-auth-compat.js"></script>

<script src="https://www.gstatic.com/firebasejs/9.21.0/firebase-firestore-compat.js"></script>

<script>

(function() {

// Initialize Firebase – REPLACE with your actual config values

const firebaseConfig = {

apiKey: "AIzaSyBpqEgGiRoZO\_Hb4IXt\_YuTV9WQLmwHBZM",

authDomain: "your-project-id.firebaseapp.com",

projectId: "your-project-id",

storageBucket: "your-project-id.appspot.com",

messagingSenderId: "your-messaging-sender-id",

appId: "your-app-id"

};

firebase.initializeApp(firebaseConfig);

const auth = firebase.auth();

const db = firebase.firestore();

const maxWalletBalance = 1000000;

/\* Utility Functions \*/

function showNotification(message, type) {

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

notification.innerText = message;

notification.style.backgroundColor = type === "error" ? "var(--error-color)" : type === "info" ? "gray" : "var(--primary-color)";

notification.style.display = "block";

setTimeout(() => { notification.style.display = "none"; }, 3000);

}

function openModal(message, callback) {

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

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

modalMessage.innerText = message;

modal.style.display = "flex";

document.getElementById("modalConfirm").focus();

document.getElementById("modalConfirm").onclick = () => {

modal.style.display = "none";

if (callback) callback();

};

}

function clearAuthMessages() {

document.getElementById("loginError").style.display = "none";

document.getElementById("signupError").style.display = "none";

}

/\* Onboarding Modal – Show once on first login \*/

function showOnboardingIfNeeded() {

if (!localStorage.getItem("onboardingCompleted")) {

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

onboardingModal.style.display = "flex";

document.getElementById("onboardingCloseBtn").onclick = () => {

onboardingModal.style.display = "none";

localStorage.setItem("onboardingCompleted", "true");

};

}

}

/\* Landing & Auth Events \*/

document.getElementById("getStartedBtn").addEventListener("click", () => {

document.getElementById("landingSection").style.display = "none";

document.getElementById("authSection").style.display = "flex";

});

document.getElementById("showSignUp").addEventListener("click", () => {

document.getElementById("loginContainer").style.display = "none";

document.getElementById("signupContainer").style.display = "block";

clearAuthMessages();

});

document.getElementById("showLogin").addEventListener("click", () => {

document.getElementById("signupContainer").style.display = "none";

document.getElementById("loginContainer").style.display = "block";

clearAuthMessages();

});

document.getElementById("forgotPassword").addEventListener("click", async () => {

const email = prompt("Enter your registered email for password reset:");

if (email) {

try {

await auth.sendPasswordResetEmail(email);

alert("Password reset email sent. Check your inbox.");

} catch (error) {

alert("Error: " + error.message);

}

}

});

/\* Authentication Actions \*/

document.getElementById("loginButton").addEventListener("click", async () => {

const email = document.getElementById("loginEmail").value;

const password = document.getElementById("loginPassword").value;

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

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

loginError.style.display = "none";

loginLoading.style.display = "block";

try {

await auth.signInWithEmailAndPassword(email, password);

loginLoading.style.display = "none";

showDashboard();

} catch (error) {

loginLoading.style.display = "none";

loginError.style.display = "block";

loginError.innerText = error.message;

}

});

document.getElementById("signupButton").addEventListener("click", async () => {

const name = document.getElementById("signupName").value;

const email = document.getElementById("signupEmail").value;

const password = document.getElementById("signupPassword").value;

const confirmPassword = document.getElementById("signupConfirmPassword").value;

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

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

signupError.style.display = "none";

if (password !== confirmPassword) {

signupError.style.display = "block";

signupError.innerText = "Passwords do not match.";

return;

}

signupLoading.style.display = "block";

try {

const userCredential = await auth.createUserWithEmailAndPassword(email, password);

const user = userCredential.user;

await user.updateProfile({ displayName: name });

await db.collection("users").doc(user.uid).set({

name: name,

email: email,

walletBalance: 0,

dailyEarnings: 0,

referralBonus: 0,

referralCount: 0

});

signupLoading.style.display = "none";

showDashboard();

} catch (error) {

signupLoading.style.display = "none";

signupError.style.display = "block";

signupError.innerText = error.message;

}

});

/\* Dashboard Functions \*/

async function loadOverview() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("userName").innerText = data.name;

document.getElementById("balanceOverview").innerText = `₦${data.walletBalance.toFixed(2)}`;

document.getElementById("dailyEarningsOverview").innerText = `₦${data.dailyEarnings.toFixed(2)}`;

document.getElementById("referralBonusOverview").innerText = `₦${data.referralBonus.toFixed(2)}`;

}

} catch (error) {

showNotification("Error loading overview: " + error.message, "error");

}

}

async function loadWallet() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

let balance = 0;

if (doc.exists) {

balance = doc.data().walletBalance || 0;

}

document.getElementById("walletBalance").innerText = balance.toFixed(2);

document.getElementById("withdrawBtn").disabled = balance <= 0;

} catch (error) {

showNotification("Error loading wallet: " + error.message, "error");

}

}

function listenTransactionHistory() {

const user = auth.currentUser;

if (!user) return;

db.collection("transactions")

.where("userId", "==", user.uid)

.orderBy("timestamp", "desc")

.onSnapshot(snapshot => {

const tableBody = document.getElementById("transactionTableBody");

tableBody.innerHTML = "";

snapshot.forEach(doc => {

const tx = doc.data();

const dateStr = tx.timestamp && tx.timestamp.toDate ? tx.timestamp.toDate().toLocaleString() : "";

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

row.innerHTML = `

<td>${dateStr}</td>

<td>${tx.description}</td>

<td>₦${Math.abs(tx.amount).toFixed(2)}</td>

<td>${tx.status}</td>

`;

tableBody.appendChild(row);

});

}, error => {

showNotification("Error loading transactions: " + error.message, "error");

});

}

/\* Generate Investment Plans \*/

function generatePlans() {

const plans = [];

for (let price = 3500; price <= 500000; price += 1500) {

let dailyReturn, duration;

if (price <= 10000) {

dailyReturn = 5;

duration = 30;

} else if (price <= 25000) {

dailyReturn = 7;

duration = 35;

} else if (price <= 50000) {

dailyReturn = 9;

duration = 40;

} else if (price <= 100000) {

dailyReturn = 12;

duration = 45;

} else if (price <= 200000) {

dailyReturn = 14;

duration = 50;

} else {

dailyReturn = 15;

duration = 60;

}

plans.push({

name: `Plan for ₦${price.toLocaleString()}`,

price: price,

dailyReturn: dailyReturn,

duration: duration

});

}

return plans;

}

function renderInvestmentPlans() {

const grid = document.getElementById("plansGrid");

grid.innerHTML = "";

const plans = generatePlans();

plans.forEach(plan => {

const card = document.createElement("div");

card.className = "plan-card";

card.innerHTML = `

<h2>${plan.name}</h2>

<p>Fixed Price: ₦${plan.price.toLocaleString()}</p>

<p>Daily Return: ${plan.dailyReturn}%</p>

<p>Duration: ${plan.duration} days</p>

<p>Total Profit: ${plan.dailyReturn \* plan.duration}%</p>

<button class="invest-btn" onclick="investPlan(${plan.price}, ${plan.dailyReturn}, ${plan.duration}, '${plan.name}')">

Invest Now

</button>

`;

grid.appendChild(card);

});

}

async function investPlan(price, dailyReturn, duration, planName) {

const user = auth.currentUser;

if (!user) {

alert("Please log in to invest.");

window.location.href = "auth.html";

return;

}

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance < price) {

throw new Error("Insufficient funds to invest.");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-price) });

const investRef = db.collection("investments").doc();

transaction.set(investRef, {

userId: user.uid,

planName: planName,

amount: price,

dailyReturn: dailyReturn,

duration: duration,

date: new Date().toLocaleDateString(),

status: "Active",

timestamp: firebase.firestore.Timestamp.now()

});

});

showNotification(`Investment in ${planName} successful!`, "success");

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

}

/\* New: Withdraw Daily Interest Function \*/

async function withdrawDailyInterest() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentInterest = userDoc.data().dailyEarnings || 0;

if (currentInterest < 2000) {

throw new Error("You must have at least ₦2,000 in daily interest to withdraw.");

}

// Reset daily interest and update wallet balance

const walletRef = db.collection("users").doc(user.uid);

transaction.update(userRef, { dailyEarnings: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentInterest) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Daily Interest",

amount: currentInterest,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Daily interest withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing daily interest: " + error.message, "error");

}

}

/\* New: Withdraw Referral Bonus Function \*/

async function withdrawReferralBonus() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentBonus = userDoc.data().referralBonus || 0;

if (currentBonus < 2000) {

throw new Error("You must have at least ₦2,000 in referral bonus to withdraw.");

}

// Reset referral bonus and update wallet balance

const walletRef = db.collection("users").doc(user.uid);

transaction.update(userRef, { referralBonus: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentBonus) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Referral Bonus",

amount: currentBonus,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Referral bonus withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing referral bonus: " + error.message, "error");

}

}

/\* Peer-to-Peer Deposit Functionality \*/

async function submitPeerDeposit() {

const depositAmount = parseFloat(document.getElementById("peerDepositAmountInput").value);

const bankRef = document.getElementById("peerDepositRefInput").value.trim();

const user = auth.currentUser;

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

if (!bankRef) {

alert("Please enter your bank transfer reference.");

return;

}

try {

await db.collection("transactions").add({

userId: user.uid,

description: "Peer-to-Peer Deposit (Pending Verification)",

amount: depositAmount,

status: "Pending",

tx\_ref: bankRef,

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

showNotification("Your deposit is pending verification.", "info");

document.getElementById("peerDepositModal").style.display = "none";

} catch (error) {

showNotification("Error recording deposit: " + error.message, "error");

}

}

/\* Link Bank Account Functionality \*/

async function linkBankAccount() {

const bankName = document.getElementById("bankNameInput").value.trim();

const accountNumber = document.getElementById("accountNumberInput").value.trim();

const accountHolder = document.getElementById("accountHolderInput").value.trim();

const user = auth.currentUser;

if (!bankName || !accountNumber || !accountHolder) {

alert("Please fill in all bank account details.");

return;

}

try {

await db.collection("users").doc(user.uid).update({

bankAccount: {

bankName: bankName,

accountNumber: accountNumber,

accountHolder: accountHolder

}

});

showNotification("Bank account linked successfully!", "success");

document.getElementById("linkBankModal").style.display = "none";

} catch (error) {

showNotification("Error linking bank account: " + error.message, "error");

}

}

/\* Transfer Funds Functionality \*/

async function transferFunds() {

const recipientEmail = document.getElementById("transferRecipientInput").value.trim();

const transferAmount = parseFloat(document.getElementById("transferAmountInput").value);

const sender = auth.currentUser;

if (!recipientEmail) {

alert("Please enter the recipient's email.");

return;

}

if (isNaN(transferAmount) || transferAmount <= 0) {

alert("Please enter a valid transfer amount.");

return;

}

const senderWalletRef = db.collection("users").doc(sender.uid);

const senderDoc = await senderWalletRef.get();

let senderBalance = senderDoc.exists ? senderDoc.data().walletBalance || 0 : 0;

if (senderBalance < transferAmount) {

alert("Insufficient funds for transfer.");

return;

}

const recipientQuery = await db.collection("users").where("email", "==", recipientEmail).get();

if (recipientQuery.empty) {

alert("Recipient not found.");

return;

}

const recipientDoc = recipientQuery.docs[0];

const recipientId = recipientDoc.id;

const recipientWalletRef = db.collection("users").doc(recipientId);

try {

await db.runTransaction(async (transaction) => {

const senderSnapshot = await transaction.get(senderWalletRef);

const recipientSnapshot = await transaction.get(recipientWalletRef);

let senderBalance = senderSnapshot.exists ? senderSnapshot.data().walletBalance || 0 : 0;

if (senderBalance < transferAmount) {

throw new Error("Insufficient funds for transfer.");

}

transaction.update(senderWalletRef, { walletBalance: firebase.firestore.FieldValue.increment(-transferAmount) });

transaction.update(recipientWalletRef, { walletBalance: firebase.firestore.FieldValue.increment(transferAmount) });

const senderTxnRef = db.collection("transactions").doc();

transaction.set(senderTxnRef, {

userId: sender.uid,

description: "Transfer to " + recipientEmail,

amount: -transferAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

const recipientTxnRef = db.collection("transactions").doc();

transaction.set(recipientTxnRef, {

userId: recipientId,

description: "Transfer from " + sender.email,

amount: transferAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

showNotification("Transfer successful!", "success");

document.getElementById("transferModal").style.display = "none";

} catch (error) {

showNotification("Transfer error: " + error.message, "error");

}

}

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

// Minimum withdrawal from wallet is still ₦2000

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentInterest = userDoc.data().dailyEarnings || 0;

if (currentInterest < 2000) {

throw new Error("You must have at least ₦2,000 in daily interest to withdraw.");

}

// Reset daily interest and update wallet balance

transaction.update(userRef, { dailyEarnings: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentInterest) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Daily Interest",

amount: currentInterest,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Daily interest withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing daily interest: " + error.message, "error");

}

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentBonus = userDoc.data().referralBonus || 0;

if (currentBonus < 2000) {

throw new Error("You must have at least ₦2,000 in referral bonus to withdraw.");

}

// Reset referral bonus and update wallet balance

transaction.update(userRef, { referralBonus: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentBonus) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Referral Bonus",

amount: currentBonus,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Referral bonus withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing referral bonus: " + error.message, "error");

}

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

await withdrawDailyInterest();

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

await withdrawReferralBonus();

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

await withdrawDailyInterest();

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

await withdrawReferralBonus();

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

await withdrawDailyInterest();

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

await withdrawReferralBonus();

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

await withdrawDailyInterest();

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

await withdrawReferralBonus();

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* New: Withdraw Daily Interest Function \*/

async function withdrawDailyInterest() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentInterest = userDoc.data().dailyEarnings || 0;

if (currentInterest < 2000) {

throw new Error("You must have at least ₦2,000 in daily interest to withdraw.");

}

transaction.update(userRef, { dailyEarnings: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentInterest) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Daily Interest",

amount: currentInterest,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Daily interest withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing daily interest: " + error.message, "error");

}

}

/\* New: Withdraw Referral Bonus Function \*/

async function withdrawReferralBonus() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentBonus = userDoc.data().referralBonus || 0;

if (currentBonus < 2000) {

throw new Error("You must have at least ₦2,000 in referral bonus to withdraw.");

}

transaction.update(userRef, { referralBonus: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentBonus) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Referral Bonus",

amount: currentBonus,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Referral bonus withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing referral bonus: " + error.message, "error");

}

}

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

/\* Dashboard Navigation \*/

window.showDashboardSection = function(sectionId) {

document.getElementById("overview").style.display = "none";

document.getElementById("wallet").style.display = "none";

document.getElementById("investment").style.display = "none";

if (document.getElementById("profile")) document.getElementById("profile").style.display = "none";

if (document.getElementById("referral")) document.getElementById("referral").style.display = "none";

document.getElementById(sectionId).style.display = "block";

if (sectionId === "referral") {

const user = auth.currentUser;

if (user) {

document.getElementById("referralLink").value = "https://yourwallet.com/ref/" + user.uid;

loadReferralHistory();

}

}

if (sectionId === "profile") loadProfile();

};

window.showDashboard = function() {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

};

window.logout = async function() {

try {

await auth.signOut();

showNotification("Logged out successfully", "info");

setTimeout(() => { window.location.href = "auth.html"; }, 1000);

} catch (error) {

showNotification("Error during logout", "error");

}

};

/\* Auth State Listener \*/

auth.onAuthStateChanged(user => {

if (user) {

document.getElementById("authSection").style.display = "none";

document.getElementById("landingSection").style.display = "none";

document.getElementById("dashboardSection").style.display = "block";

loadOverview();

loadWallet();

renderInvestmentPlans();

showDashboardSection("overview");

listenTransactionHistory();

showOnboardingIfNeeded();

} else {

document.getElementById("landingSection").style.display = "block";

document.getElementById("dashboardSection").style.display = "none";

document.getElementById("authSection").style.display = "none";

}

});

/\* Deposit Modal Actions (Flutterwave) \*/

document.getElementById("depositBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "flex";

});

document.getElementById("depositCancelBtn").addEventListener("click", () => {

document.getElementById("depositModal").style.display = "none";

});

document.getElementById("depositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("depositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

const user = auth.currentUser;

if (!user) return;

try {

FlutterwaveCheckout({

public\_key: "FLWPUBK\_TEST-86327f2265746dea33850da34d15e513-X",

tx\_ref: "TX-" + Date.now(),

amount: depositAmount,

currency: "NGN",

payment\_options: "card, banktransfer, ussd",

customer: {

email: user.email,

phonenumber: "08012345678",

name: user.displayName || "Customer"

},

callback: async function(response) {

if (response.status === "successful" || response.status === "success") {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

let currentBalance = userDoc.exists ? userDoc.data().walletBalance || 0 : 0;

if (currentBalance + depositAmount > maxWalletBalance) {

throw new Error(`Deposit rejected. Your wallet cannot exceed ₦${maxWalletBalance.toLocaleString()}.`);

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(depositAmount) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Deposit (Flutterwave)",

amount: depositAmount,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp()

});

});

loadWallet();

loadOverview();

showNotification(`Deposit of ₦${depositAmount} successful!`, "success");

} else {

showNotification("Deposit failed. Please try again.", "error");

}

},

onclose: function() {

console.log("Flutterwave Checkout closed.");

}

});

document.getElementById("depositModal").style.display = "none";

depositInputEl.value = "";

} catch (error) {

showNotification("Error during deposit: " + error.message, "error");

}

});

/\* Deposit Modal Actions (UfitPay) \*/

document.getElementById("ufitpayDepositBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "flex";

});

document.getElementById("ufitpayDepositCancelBtn").addEventListener("click", () => {

document.getElementById("ufitpayDepositModal").style.display = "none";

});

document.getElementById("ufitpayDepositSubmitBtn").addEventListener("click", async () => {

const depositInputEl = document.getElementById("ufitpayDepositAmountInput");

let depositAmount = parseFloat(depositInputEl.value);

if (isNaN(depositAmount) || depositAmount <= 0) {

alert("Please enter a valid deposit amount.");

return;

}

await depositWithUfitPay(depositAmount);

document.getElementById("ufitpayDepositModal").style.display = "none";

depositInputEl.value = "";

});

/\* Peer-to-Peer Deposit Modal Actions \*/

document.getElementById("peerDepositBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "flex";

});

document.getElementById("peerDepositCancelBtn").addEventListener("click", () => {

document.getElementById("peerDepositModal").style.display = "none";

});

document.getElementById("peerDepositSubmitBtn").addEventListener("click", async () => {

await submitPeerDeposit();

});

/\* Transfer Funds Modal Actions \*/

document.getElementById("transferBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "flex";

});

document.getElementById("transferCancelBtn").addEventListener("click", () => {

document.getElementById("transferModal").style.display = "none";

});

document.getElementById("transferSubmitBtn").addEventListener("click", async () => {

await transferFunds();

});

/\* Link Bank Account Modal Actions \*/

document.getElementById("linkBankBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "flex";

});

document.getElementById("linkBankCancelBtn").addEventListener("click", () => {

document.getElementById("linkBankModal").style.display = "none";

});

document.getElementById("linkBankSubmitBtn").addEventListener("click", async () => {

await linkBankAccount();

});

/\* Withdraw Modal Actions \*/

document.getElementById("withdrawBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "flex";

});

document.getElementById("withdrawCancelBtn").addEventListener("click", () => {

document.getElementById("withdrawModal").style.display = "none";

});

document.getElementById("withdrawSubmitBtn").addEventListener("click", () => {

const withdrawInputEl = document.getElementById("withdrawAmountInput");

let withdrawAmount = parseFloat(withdrawInputEl.value);

if (isNaN(withdrawAmount) || withdrawAmount < 2000) {

alert("Minimum withdrawal amount from your wallet is ₦2,000.");

return;

}

const user = auth.currentUser;

if (!user) return;

openModal(`Are you sure you want to withdraw ₦${withdrawAmount}? (20% fee will be applied)`, async () => {

try {

await db.runTransaction(async (transaction) => {

const userRef = db.collection("users").doc(user.uid);

const userDoc = await transaction.get(userRef);

const currentBalance = userDoc.data().walletBalance || 0;

const fee = withdrawAmount \* 0.20;

const totalDeduction = withdrawAmount + fee;

if (totalDeduction > currentBalance) {

throw new Error("Insufficient funds for withdrawal (including fee).");

}

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(-totalDeduction) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal from Wallet",

amount: withdrawAmount,

fee: fee,

netAmount: withdrawAmount - fee,

status: "Pending",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

document.getElementById("withdrawModal").style.display = "none";

withdrawInputEl.value = "";

loadWallet();

loadOverview();

showNotification(`Withdrawal of ₦${withdrawAmount} initiated (₦${(withdrawAmount\*0.20).toFixed(2)} fee applied)!`, "info");

} catch (error) {

showNotification("Error during withdrawal: " + error.message, "error");

}

});

});

/\* New: Withdraw Daily Interest Action \*/

document.getElementById("withdrawDailyBtn").addEventListener("click", async () => {

await withdrawDailyInterest();

});

/\* New: Withdraw Referral Bonus Action \*/

document.getElementById("withdrawReferralBtn").addEventListener("click", async () => {

await withdrawReferralBonus();

});

// Expose investPlan globally for plan cards

window.investPlan = investPlan;

/\* New: Withdraw Daily Interest Function \*/

async function withdrawDailyInterest() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentInterest = userDoc.data().dailyEarnings || 0;

if (currentInterest < 2000) {

throw new Error("You must have at least ₦2,000 in daily interest to withdraw.");

}

transaction.update(userRef, { dailyEarnings: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentInterest) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Daily Interest",

amount: currentInterest,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Daily interest withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing daily interest: " + error.message, "error");

}

}

/\* New: Withdraw Referral Bonus Function \*/

async function withdrawReferralBonus() {

const user = auth.currentUser;

if (!user) return;

try {

const userRef = db.collection("users").doc(user.uid);

await db.runTransaction(async (transaction) => {

const userDoc = await transaction.get(userRef);

const currentBonus = userDoc.data().referralBonus || 0;

if (currentBonus < 2000) {

throw new Error("You must have at least ₦2,000 in referral bonus to withdraw.");

}

transaction.update(userRef, { referralBonus: 0 });

transaction.update(userRef, { walletBalance: firebase.firestore.FieldValue.increment(currentBonus) });

const txnRef = db.collection("transactions").doc();

transaction.set(txnRef, {

userId: user.uid,

description: "Withdrawal of Referral Bonus",

amount: currentBonus,

status: "Completed",

timestamp: firebase.firestore.FieldValue.serverTimestamp(),

});

});

loadWallet();

loadOverview();

showNotification("Referral bonus withdrawn successfully!", "success");

} catch (error) {

showNotification("Error withdrawing referral bonus: " + error.message, "error");

}

}

/\* Deposit using UfitPay API with Enhanced Error Handling \*/

async function depositWithUfitPay(depositAmount) {

const user = auth.currentUser;

if (!user) return;

try {

const response = await fetch("https://api.ufitpay.com/api/transaction/initialize", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Api-Key": "pub-330ea1ef3dfa413bb40931f1b890c732",

"API-Token": "sec-4b4754aac619c33df1cf30df29b35d9c"

},

body: JSON.stringify({

amount: depositAmount,

currency: "NGN",

customer: {

email: user.email,

name: user.displayName || "Customer",

phone: "08012345678"

},

tx\_ref: "UP-" + Date.now(),

narration: "Deposit via UfitPay"

})

});

if (!response.ok) {

const errorText = await response.text();

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

}

const data = await response.json();

if (data && data.status === "success" && data.data && data.data.payment\_url) {

window.location.href = data.data.payment\_url;

} else {

throw new Error(data.message || "UfitPay deposit failed");

}

} catch (error) {

showNotification("UfitPay Deposit Error: " + error.message, "error");

}

}

function copyReferralLink() {

const referralInput = document.getElementById("referralLink");

referralInput.select();

document.execCommand("copy");

showNotification("Referral link copied!", "success");

}

async function loadReferralHistory() {

const user = auth.currentUser;

if (!user) return;

try {

const snapshot = await db.collection("referrals")

.where("referrerId", "==", user.uid)

.orderBy("date", "desc")

.get();

const referralDiv = document.getElementById("referralHistory");

referralDiv.innerHTML = "";

if (snapshot.empty) {

referralDiv.innerHTML = "<p>No referrals yet.</p>";

} else {

snapshot.forEach(doc => {

const refData = doc.data();

const entry = document.createElement("p");

entry.innerText = `${refData.referredEmail} on ${refData.date}`;

referralDiv.appendChild(entry);

});

}

} catch (error) {

showNotification("Error loading referral history: " + error.message, "error");

}

}

async function loadProfile() {

const user = auth.currentUser;

if (!user) return;

try {

const doc = await db.collection("users").doc(user.uid).get();

if (doc.exists) {

const data = doc.data();

document.getElementById("profileName").value = data.name;

document.getElementById("profileEmail").value = data.email;

}

} catch (error) {

showNotification("Error loading profile: " + error.message, "error");

}

}

if (document.getElementById("profileForm")) {

document.getElementById("profileForm").addEventListener("submit", async function(e) {

e.preventDefault();

const newName = document.getElementById("profileName").value;

const newEmail = document.getElementById("profileEmail").value;

const user = auth.currentUser;

if (!user) return;

try {

await user.updateProfile({ displayName: newName });

await db.collection("users").doc(user.uid).update({

name: newName,

email: newEmail

});

showNotification("Profile updated successfully!", "success");

loadOverview();

} catch (error) {

showNotification(`Error: ${error.message}`, "error");

}

});

}

})();

</script>

</body>

</html