<!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">

  <!-- Animate.css for animations -->

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">

  <!-- Flutterwave Checkout -->

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

  <!-- Chart.js for Analytics -->

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <style>

    /\* Color Palette and Variables \*/

    :root {

      --primary-color: #28a745;      /\* Green \*/

      --secondary-color: #ffc107;    /\* Yellow \*/

      --error-color: #dc3545;        /\* Red \*/

      --background-color: #ffffff;

      --text-color: #333333;

      --sidebar-width: 250px;

      --min-amount: 3500;            /\* Minimum deposit/withdrawal \*/

      --transition-speed: 0.3s;

    }

    /\* Dark Mode Variables \*/

    .dark-mode {

      --background-color: #121212;

      --text-color: #dddddd;

      --primary-color: #66bb6a;

      --secondary-color: #ffca28;

    }

    \* { box-sizing: border-box; }

    body {

      margin: 0;

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

      background: var(--background-color);

      color: var(--text-color);

      line-height: 1.6;

      overflow-x: hidden;

      transition: background 0.3s, color 0.3s;

    }

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

    @keyframes fadeIn {

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

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

    }

    /\* Generic Modal Styles \*/

    .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: 2200;

    }

    .modal .modal-content, .link-bank-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: 0 4px 8px rgba(0,0,0,0.1);

      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;

    }

    .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 Section \*/

    #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: 0 4px 8px rgba(0,0,0,0.1);

      width: 300px;

      text-align: center;

      margin-bottom: 1rem;

      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;

    }

    .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: var(--sidebar-width);

      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: var(--sidebar-width);

      padding: 20px;

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

    }

    .dashboard-header {

      display: flex;

      justify-content: space-between;

      align-items: center;

      margin-bottom: 20px;

    }

    .dashboard-header h1 { font-size: 28px; margin: 0; }

    /\* Dark mode toggle button \*/

    #toggleDarkMode {

      background: var(--secondary-color);

      color: var(--background-color);

      border: none;

      padding: 5px 10px;

      cursor: pointer;

      border-radius: 5px;

      font-size: 0.9rem;

    }

    #toggleDarkMode:hover { background: #e0a800; }

    .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: 0 4px 8px rgba(0,0,0,0.1);

      margin: 10px;

      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;

      flex-wrap: wrap;

      justify-content: center;

      gap: 20px;

      margin: 20px 0;

    }

    .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;

      margin: 5px;

    }

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

    .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; }

    .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: 0 4px 8px rgba(0,0,0,0.1);

      text-align: center;

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

      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); }

    /\* Page Sections \*/

    .section { display: none; }

    /\* By default show the overview section \*/

    #overview { display: block; }

    /\* New sections for Analytics and News \*/

    #analytics, #news { display: none; }

    #profile, #referral { text-align: center; }

    #profile form, #referral { text-align: center; }

    #profile form input {

      width: 100%;

      padding: 10px;

      margin: 10px 0;

      border: 1px solid #ccc;

      border-radius: 5px;

    }

    #profile form button, #referral button {

      width: 100%;

      padding: 10px;

      background: var(--secondary-color);

      border: none;

      border-radius: 5px;

      cursor: pointer;

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

      font-size: 1rem;

    }

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

    footer {

      background: #000;

      color: var(--background-color);

      text-align: center;

      padding: 1rem 0;

      margin-top: 2rem;

    }

    #notification {

      position: fixed;

      top: 20px;

      right: 20px;

      padding: 10px 20px;

      border-radius: 5px;

      background: var(--primary-color);

      color: var(--background-color);

      display: none;

      z-index: 3000;

    }

    /\* Responsive Adjustments \*/

    @media (max-width: 768px) {

      .sidebar { width: 60px; }

      .main-content { margin-left: 60px; }

      .sidebar h2 { display: none; }

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

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

    }

  </style>

</head>

<body>

  <!-- Landing Section -->

  <div id="landingSection">

    <h1 class="animate\_\_animated animate\_\_fadeInDown">InvestPro</h1>

    <p class="animate\_\_animated animate\_\_fadeInUp">

      Invest in tailored plans and grow your wealth with daily withdrawals. Experience a professional platform built for performance.

    </p>

    <button class="cta-button animate\_\_animated animate\_\_pulse animate\_\_infinite" id="getStartedBtn">Get Started</button>

  </div>

  <!-- Auth Section -->

  <div id="authSection">

    <div class="auth-container" id="loginContainer">

      <h2>Login</h2>

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

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

      <button id="loginButton">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>

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

      <h2>Sign Up</h2>

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

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

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

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

      <button id="signupButton">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>

  <!-- OTP Modal (Simulated Email OTP) -->

  <div class="modal" id="otpModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Email OTP Verification</h3>

      <p>An OTP has been sent to your email. (For demo, enter 123456.)</p>

      <input type="text" id="otpInput" placeholder="Enter OTP">

      <div>

        <button id="otpSubmitBtn">Verify</button>

        <button id="otpCancelBtn">Cancel</button>

      </div>

    </div>

  </div>

  <!-- Dashboard Section -->

  <div id="dashboardSection">

    <!-- Sidebar Navigation -->

    <div class="sidebar" id="sidebar">

      <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('analytics')">Analytics</li>

        <li onclick="showDashboardSection('news')">News</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" id="mainContent">

      <div class="dashboard-header">

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

        <button id="toggleDarkMode">Dark Mode</button>

      </div>

      <!-- Page Sections -->

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

        <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>

      <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">Deposit (Flutterwave)</button>

          <button id="moniepointDepositBtn">Deposit (opay)</button>

          <button id="ufitPayDepositBtn">Deposit (Platform Bank)</button>

          <button id="peerDepositBtn">P2P Deposit</button>

          <button id="transferBtn">Transfer Funds</button>

          <button id="withdrawBtn" disabled>Withdraw Funds</button>

          <button id="linkBankBtn">Link Bank Account</button>

        </div>

        <div class="transaction-history">

          <h3>Transaction History</h3>

          <table class="transaction-table">

            <thead>

              <tr>

                <th>Date</th>

                <th>Description</th>

                <th>Amount (₦)</th>

                <th>Status</th>

              </tr>

            </thead>

            <tbody id="transactionTableBody">

              <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>

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

        <h1>Investment Plans</h1>

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

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

        </div>

      </div>

      <!-- New Analytics Section -->

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

        <h1>Analytics</h1>

        <canvas id="investmentChart" width="400" height="200"></canvas>

      </div>

      <!-- New Financial News Section -->

      <div id="news" class="section">

        <h1>Financial News</h1>

        <div id="newsContent">

          <!-- Placeholder news articles -->

          <article>

            <h3>Market Update: Stocks on the Rise</h3>

            <p>The market is experiencing positive momentum in the tech sector...</p>

          </article>

          <article>

            <h3>Investment Trends for 2025</h3>

            <p>Experts suggest diversifying portfolios amid changing global trends...</p>

          </article>

        </div>

      </div>

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

        <h2>Your Profile</h2>

        <form id="profileForm">

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

          <input type="text" id="profileName" required>

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

          <input type="email" id="profileEmail" required>

          <button type="submit">Update Profile</button>

        </form>

      </div>

      <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>

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

        <h3>Your Referral History</h3>

        <div id="referralHistory">

          <p>No referrals yet.</p>

        </div>

      </div>

    </div>

  </div>

  <!-- Peer-to-Peer Deposit Modal -->

  <div class="modal" id="peerDepositModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Peer-to-Peer Deposit</h3>

      <p>Please transfer an amount (minimum ₦3,500) from any bank to our account:</p>

      <p>

        <strong>Bank:</strong> opay Bank,<br>

        <strong>Account Number:</strong> 9135476650,<br>

        <strong>Account Name:</strong> Linda james.

      </p>

      <input type="number" id="peerDepositAmountInput" placeholder="Enter amount in ₦">

      <input type="text" id="peerDepositRefInput" placeholder="Enter bank transfer reference">

      <div>

        <button id="peerDepositSubmitBtn">Submit</button>

        <button id="peerDepositCancelBtn">Cancel</button>

      </div>

      <p style="font-size: 0.9rem;">\* Your deposit will be automatically confirmed.</p>

    </div>

  </div>

  <!-- Moniepoint Deposit Modal -->

  <div class="modal" id="moniepointDepositModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Deposit via opay</h3>

      <p>Please transfer an amount (minimum ₦3,500) from any bank to our opay account:</p>

      <p>

        <strong>Bank:</strong> opay Bank,<br>

        <strong>Account Number:</strong> 9135476650,<br>

        <strong>Account Name:</strong> linda james.

      </p>

      <input type="number" id="moniepointDepositAmountInput" placeholder="Enter amount in ₦">

      <input type="text" id="moniepointDepositRefInput" placeholder="Enter bank transfer reference">

      <div>

        <button id="moniepointDepositSubmitBtn">Submit</button>

        <button id="moniepointDepositCancelBtn">Cancel</button>

      </div>

      <p style="font-size: 0.9rem;">\* Your deposit will be processed shortly.</p>

    </div>

  </div>

  <!-- UfitPay Deposit Modal -->

  <div class="modal" id="ufitPayDepositModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Deposit via Platform Bank</h3>

      <input type="number" id="ufitPayDepositAmountInput" placeholder="Enter amount in ₦">

      <input type="text" id="ufitPayDepositRefInput" placeholder="Enter bank transfer reference">

      <div>

        <button id="ufitPayDepositSubmitBtn">Submit</button>

        <button id="ufitPayDepositCancelBtn">Cancel</button>

      </div>

      <p style="font-size: 0.9rem;">\* Funds will appear as pending until confirmed.</p>

    </div>

  </div>

  <!-- Transfer Funds Modal -->

  <div class="modal" id="transferModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Transfer Funds to a Peer</h3>

      <input type="email" id="transferRecipientInput" placeholder="Enter recipient's email">

      <input type="number" id="transferAmountInput" placeholder="Enter amount in ₦">

      <div>

        <button id="transferSubmitBtn">Submit</button>

        <button id="transferCancelBtn">Cancel</button>

      </div>

    </div>

  </div>

  <!-- Link Bank Account Modal -->

  <div class="link-bank-modal" id="linkBankModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Link Your Bank Account</h3>

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

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

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

      <div>

        <button id="linkBankSubmitBtn">Link Account</button>

        <button id="linkBankCancelBtn">Cancel</button>

      </div>

    </div>

  </div>

  <!-- Withdraw Modal -->

  <div class="modal" id="withdrawModal">

    <div class="modal-content animate\_\_animated animate\_\_zoomIn">

      <h3>Withdraw Funds</h3>

      <input type="number" id="withdrawAmountInput" placeholder="Enter amount in ₦">

      <div>

        <button id="withdrawSubmitBtn">Submit</button>

        <button id="withdrawCancelBtn">Cancel</button>

      </div>

      <p style="font-size: 0.9rem;">\* Minimum withdrawal is ₦3,500.</p>

    </div>

  </div>

  <!-- Notification -->

  <div id="notification"></div>

  <!-- 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 configuration values

      const firebaseConfig = {

        apiKey: "AIzaSyBpqEgGiRoZO\_Hb4IXt\_YuTV9WQLmwHBZM",

        authDomain: "investoship.firebaseapp.com",

        projectId: "investoship",

        storageBucket: "investoship.firebasestorge.app",

        messagingSenderId: "264495950069",

        appId: "1:264495950069:web:dbda7567da9cb3f0b4bdbc"

      };

      firebase.initializeApp(firebaseConfig);

      const auth = firebase.auth();

      const db = firebase.firestore();

      const MIN\_AMOUNT = 3500;

      const urlParams = new URLSearchParams(window.location.search);

      const REFERRAL\_ID = urlParams.get("ref");

      /\* Utility Function \*/

      function showNotification(message, type) {

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

        note.innerText = message;

        note.style.backgroundColor = type === "error" ? "var(--error-color)" :

                                    type === "info" ? "gray" : "var(--primary-color)";

        note.style.display = "block";

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

      }

      /\* 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";

      });

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

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

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

      });

      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 \*/

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

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

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

        try {

          await auth.signInWithEmailAndPassword(email, password);

          // Simulate OTP send (in production, integrate proper OTP sending)

          sessionStorage.setItem("otp", "123456");

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

        } catch (error) {

          showNotification(error.message, "error");

        }

      });

      /\* Sign-up with duplicate check and email verification \*/

      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;

        if (password !== confirmPassword) {

          alert("Passwords do not match.");

          return;

        }

        try {

          const existingMethods = await auth.fetchSignInMethodsForEmail(email);

          if (existingMethods.length > 0) {

            showNotification("Email already in use", "error");

            return;

          }

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

          const user = userCredential.user;

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

          await user.sendEmailVerification();

          const userData = {

            name: name,

            email: email,

            balance: 0,

            dailyEarnings: 0,

            referralBonus: 0,

            referralCount: 0

          };

          if (REFERRAL\_ID) {

            userData.referrer = REFERRAL\_ID;

          }

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

          if (REFERRAL\_ID) {

            await db.collection("users").doc(REFERRAL\_ID).update({

              referralCount: firebase.firestore.FieldValue.increment(1)

            });

          }

          sessionStorage.setItem("otp", "123456");

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

        } catch (error) {

          showNotification(error.message, "error");

        }

      });

      /\* OTP Modal \*/

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

        const otp = document.getElementById("otpInput").value.trim();

        if (otp === sessionStorage.getItem("otp")) {

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

          showDashboard();

        } else {

          alert("Invalid OTP. Please try again.");

        }

      });

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

        auth.signOut();

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

      });

      /\* Dashboard Section Navigation \*/

      window.showDashboardSection = function(section) {

        const sections = document.querySelectorAll('#mainContent .section');

        sections.forEach(sec => sec.style.display = 'none');

        const targetSection = document.getElementById(section);

        if (targetSection) {

          targetSection.style.display = 'block';

          if (section === 'analytics') {

            renderChart();

          }

          if (section === 'referral') {

            loadReferralHistory();

          }

        }

      };

      /\* 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.balance||0).toFixed(2)}`;

            document.getElementById("dailyEarningsOverview").innerText = `₦${(data.dailyEarnings||0).toFixed(2)}`;

            document.getElementById("referralBonusOverview").innerText = `₦${(data.referralBonus||0).toFixed(2)}`;

          }

        } catch (error) {

          showNotification();

        }

      }

      async function loadWallet() {

        const user = auth.currentUser;

        if (!user) return;

        try {

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

          const balance = (doc.exists && doc.data().balance) || 0;

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

          document.getElementById("withdrawBtn").disabled = balance < MIN\_AMOUNT;

        } catch (error) {

          showNotification();

        }

      }

      function listenTransactionHistory() {

        const user = auth.currentUser;

        if (!user) return;

        db.collection("transactions")

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

          .orderBy("timestamp", "desc")

          .onSnapshot(snapshot => {

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

            tbody.innerHTML = "";

            snapshot.forEach(doc => {

              const tx = doc.data();

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

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

              tr.innerHTML = `

                <td>${dateStr}</td>

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

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

                <td>${tx.status}</td>

              `;

              tbody.appendChild(tr);

            });

          }, error => {

            showNotification();

          });

      }

      function generatePlans() {

        const plans = [];

        for (let price = 3500; price <= 500000; price += 2500) {

          let dailyReturn, duration;

          if (price <= 10000) { dailyReturn = 12; duration = 30; }

          else if (price <= 25000) { dailyReturn = 17; duration = 30; }

          else if (price <= 50000) { dailyReturn = 20; duration = 45; }

          else if (price <= 100000) { dailyReturn = 23; duration = 50; }

          else if (price <= 200000) { dailyReturn = 27; duration =55 ; }

          else { dailyReturn = 45; duration = 60; }

          const dailyInterest = price \* (dailyReturn / 100);

          const totalProfit = dailyInterest \* duration;

          plans.push({ name: `Plan for ₦${price.toLocaleString()}`, price, dailyInterest, duration, totalProfit });

        }

        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 Interest: ₦${plan.dailyInterest.toFixed(2)}</p>

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

            <p>Total Profit: ₦${plan.totalProfit.toFixed(2)}</p>

            <button class="invest-btn" onclick="openPeerDepositForPlan(${plan.price})">Invest Now</button>

          `;

          grid.appendChild(card);

        });

      }

      window.openPeerDepositForPlan = function(price) {

        if (price < MIN\_AMOUNT) {

          alert("Minimum investment is ₦3,500.");

          return;

        }

        document.getElementById("peerDepositAmountInput").value = price;

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

      };

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

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

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

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

        if (isNaN(amount) || amount < MIN\_AMOUNT) {

          alert("Please enter a valid amount (minimum ₦3,500).");

          return;

        }

        if (!reference) {

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

          return;

        }

        const user = auth.currentUser;

        try {

          // Add transaction as completed and update wallet

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

            userId: user.uid,

            description: "P2P Deposit",

            amount: amount,

            status: "pending",

            reference: reference,

            timestamp: firebase.firestore.Timestamp.now()

          });

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

            balance: firebase.firestore.FieldValue.increment(amount)

          }, { merge: true });

          showNotification(`P2P deposit of ₦${amount} successful!`, "success");

          document.getElementById("peerDepositAmountInput").value = "";

          document.getElementById("peerDepositRefInput").value = "";

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

          // Apply referral bonus

          await applyReferralBonus(user.uid, amount);

        } catch (error) {

          showNotification("Error processing P2P deposit: " + error.message, "error");

        }

      });

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

        document.getElementById("peerDepositAmountInput").value = "";

        document.getElementById("peerDepositRefInput").value = "";

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

      });

      /\* Moniepoint Deposit Modal Actions \*/

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

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

      });

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

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

      });

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

        const amount = parseFloat(document.getElementById("moniepointDepositAmountInput").value);

        const reference = document.getElementById("moniepointDepositRefInput").value.trim();

        if (isNaN(amount) || amount < MIN\_AMOUNT) {

          alert("Please enter a valid deposit amount (minimum ₦3,500).");

          return;

        }

        if (!reference) {

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

          return;

        }

        const user = auth.currentUser;

        try {

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

            userId: user.uid,

            description: "Deposit (Moniepoint)",

            amount: amount,

            status: "Completed",

            reference: reference,

            timestamp: firebase.firestore.Timestamp.now()

          });

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

            balance: firebase.firestore.FieldValue.increment(amount)

          }, { merge: true });

          showNotification(`Moniepoint deposit of ₦${amount} successful!`, "success");

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

          document.getElementById("moniepointDepositAmountInput").value = "";

          document.getElementById("moniepointDepositRefInput").value = "";

          // Apply referral bonus

          await applyReferralBonus(user.uid, amount);

        } catch (error) {

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

        }

      });

      /\* UfitPay Deposit Modal Actions \*/

      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 amount = parseFloat(document.getElementById("ufitPayDepositAmountInput").value);

        const reference = document.getElementById("ufitPayDepositRefInput").value.trim();

        if (isNaN(amount) || amount < MIN\_AMOUNT) {

          alert("Please enter a valid deposit amount (minimum ₦3,500).");

          return;

        }

        if (!reference) {

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

          return;

        }

        await depositWithUfitPay(amount, reference);

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

        document.getElementById("ufitPayDepositAmountInput").value = "";

        document.getElementById("ufitPayDepositRefInput").value = "";

      });

      async function depositWithUfitPay(amount, reference) {

        const user = auth.currentUser;

        if (!user) return;

        const url = 'https://api.ufitpay.com/bank-transfer';

        const API\_KEY = "YOUR\_UFITPAY\_API\_KEY";

        const API\_TOKEN = "YOUR\_UFITPAY\_API\_TOKEN";

        try {

          const response = await fetch(url, {

            method: "POST",

            headers: {

              "Api-Key": API\_KEY,

              "API-Token": API\_TOKEN,

              "Content-Type": "application/json"

            },

            body: JSON.stringify({

              amount,

              reference,

              narration: "Deposit via Platform Bank"

            })

          });

          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") {

            let txnStatus = data.confirmed ? "Completed" : "Pending";

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

              userId: user.uid,

              description: "Deposit (UfitPay)",

              amount,

              status: txnStatus,

              reference,

              timestamp: firebase.firestore.Timestamp.now()

            });

            if (txnStatus === "Completed") {

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

                balance: firebase.firestore.FieldValue.increment(amount)

              }, { merge: true });

              showNotification(`Deposit of ₦${amount} confirmed!`, "success");

              await applyReferralBonus(user.uid, amount);

            } else {

              showNotification(`Deposit of ₦${amount} is pending confirmation.`, "info");

            }

          } else {

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

          }

        } catch (error) {

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

        }

      }

      /\* 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 () => {

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

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

        const sender = auth.currentUser;

        if (!recipientEmail) {

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

          return;

        }

        if (isNaN(transferAmount) || transferAmount < MIN\_AMOUNT) {

          alert("Minimum transfer amount is ₦3,500.");

          return;

        }

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

        const senderWalletDoc = await senderWalletRef.get();

        let senderBalance = senderWalletDoc.exists ? (senderWalletDoc.data().balance || 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("wallets").doc(recipientId);

        try {

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

            const senderDoc = await transl/action.get(senderWalletRef);

            const recipientDocSnap = await transaction.get(recipientWalletRef);

            let currentSenderBalance = senderDoc.exists ? (senderDoc.data().balance || 0) : 0;

            if (currentSenderBalance < transferAmount) {

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

            }

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

            transaction.update(recipientWalletRef, { balance: 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.Timestamp.now()

            });

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

            transaction.set(recipientTxnRef, {

              userId: recipientId,

              description: "Transfer from " + sender.email,

              amount: transferAmount,

              status: "Completed",

              timestamp: firebase.firestore.Timestamp.now()

            });

          });

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

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

        } catch (error) {

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

        }

      });

      /\* 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 () => {

        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, accountNumber, accountHolder }

          });

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

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

        } catch (error) {

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

        }

      });

      /\* 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 withdrawAmount = parseFloat(document.getElementById("withdrawAmountInput").value);

        if (isNaN(withdrawAmount) || withdrawAmount < MIN\_AMOUNT) {

          alert("Minimum withdrawal amount is ₦3,500.");

          return;

        }

        const user = auth.currentUser;

        if (!user) return;

        if (confirm(`Are you sure you want to withdraw ₦${withdrawAmount}?`)) {

          db.runTransaction(async (transaction) => {

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

            const walletDoc = await transaction.get(walletRef);

            let currentBalance = walletDoc.data().balance || 0;

            if (withdrawAmount > currentBalance) {

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

            }

            transaction.update(walletRef, { balance: firebase.firestore.FieldValue.increment(-withdrawAmount) });

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

            transaction.set(txnRef, {

              userId: user.uid,

              description: "Withdrawal",

              amount: -withdrawAmount,

              status: "Pending",

              timestamp: firebase.firestore.Timestamp.now()

            });

          }).then(() => {

            showNotification(`Withdrawal of ₦${withdrawAmount} initiated!`, "info");

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

          }).catch(error => {

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

          });

        }

      });

      window.copyReferralLink = function() {

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

        input.select();

        document.execCommand("copy");

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

      };

      window.loadProfile = async function() {

        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;

            // Generate a unique referral link based on the user's UID

            const referralLink = window.location.origin + "/?ref=" + user.uid;

            document.getElementById("referralLink").value = referralLink;

          }

        } catch (error) {

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

        }

      };

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

        e.preventDefault();

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

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

        const user = auth.currentUser;

        try {

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

          await db.collection("users").doc(user.uid).update({ name, email });

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

          // Update referral link after profile update if needed

          const referralLink = window.location.origin + "/?ref=" + user.uid;

          document.getElementById("referralLink").value = referralLink;

        } catch (error) {

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

        }

      });

      async function applyReferralBonus(userId, amount) {

        const userDoc = await db.collection("users").doc(userId).get();

        if (userDoc.exists && userDoc.data().referrer) {

          const bonus = 0.2 \* amount;

          await db.collection("users").doc(userDoc.data().referrer).update({

            referralBonus: firebase.firestore.FieldValue.increment(bonus)

          });

          showNotification(`Referrer bonus of ₦${bonus.toFixed(2)} awarded!`, "success");

        }

      }

      window.loadReferralHistory = async function() {

        const user = auth.currentUser;

        if (!user) return;

        try {

          const snapshot = await db.collection("users").where("referrer", "==", user.uid).get();

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

          historyDiv.innerHTML = '';

          if (snapshot.empty) {

            historyDiv.innerHTML = '<p>No referrals yet.</p>';

          } else {

            snapshot.forEach(doc => {

              const data = doc.data();

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

              entry.textContent = data.name || doc.id;

              historyDiv.appendChild(entry);

            });

          }

        } catch (error) {

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

        }

      };

      let chartInstance = null;

      function renderChart() {

        const ctx = document.getElementById('investmentChart').getContext('2d');

        const sampleData = {

          labels: ["Jan", "Feb", "Mar", "Apr", "May", "Jun"],

          datasets: [{

            label: "Invested Amount (₦)",

            data: [5000, 10000, 7500, 12000, 9000, 15000],

            backgroundColor: "rgba(40, 167, 69, 0.5)",

            borderColor: "rgba(40, 167, 69, 1)",

            borderWidth: 1

          }]

        };

        if (chartInstance) {

          chartInstance.data = sampleData;

          chartInstance.update();

        } else {

          chartInstance = new Chart(ctx, {

            type: 'bar',

            data: sampleData,

            options: {

              scales: {

                y: { beginAtZero: true }

              }

            }

          });

        }

      }

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

        document.body.classList.toggle("dark-mode");

        const mode = document.body.classList.contains("dark-mode") ? "dark" : "light";

        localStorage.setItem("theme", mode);

      });

      if (localStorage.getItem("theme") === "dark") {

        document.body.classList.add("dark-mode");

      }

      window.addEventListener('online', () => {

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

        showNotification();

      });

      window.addEventListener('offline', () => {

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

        showNotification();

      });

      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();

          listenTransactionHistory();

          loadProfile();

          loadReferralHistory();

        } else {

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

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

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

        }

      });

      window.showDashboard = function() {

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

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

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

        loadOverview();

        loadWallet();

        renderInvestmentPlans();

        listenTransactionHistory();

        loadProfile();

        loadReferralHistory();

        if (!auth.currentUser.emailVerified) {

          showNotification("Please verify your email to access full features.", "info");

        }

      };

      window.logout = async function() {

        try {

          await auth.signOut();

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

          setTimeout(() => { window.location.reload(); }, 1000);

        } catch (error) {

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

        }

      };

    })();

  </script>

</body>

</html>