```
document.addEventListener("DOMContentLoaded", () => {
  const tabs = document.querySelectorAll(".tab-link");
  const panes = document.querySelectorAll(".tab-pane");
  const dropdown = document.querySelector(".dropdown-content");
  const dropbtn = document.querySelector(".dropbtn");

  document.querySelectorAll('.flash-message').forEach(flash => {
    setTimeout(() => {
      flash.classList.add('fade-out');
      setTimeout(() => flash.remove(), 500);
    }, 3000);
  });

  if (dropbtn && dropdown) {
    dropbtn.addEventListener('click', e => {
      e.stopPropagation();
      dropdown.classList.toggle('show');
    });
    document.addEventListener('click', () => {
      dropdown.classList.remove('show');
    });
    document.addEventListener('keydown', (e) => {
      if (e.key === 'Escape' && dropdown.classList.contains('show')) {
        dropdown.classList.remove('show');
        dropbtn.focus();
      }
    });
  }

  function clearActive() {
    tabs.forEach(t => t.classList.remove('active'));
    panes.forEach(p => p.classList.remove('active'));
  }

  async function showTab(tabID) {
    clearActive();
    const tab = document.querySelector(`.tab-link[data-tab="${tabID}"]`);
    const pane = document.getElementById(tabID);
    if (tab && pane) {
      tab.classList.add('active');
      tab.focus();
      pane.classList.add('active');
      switch (tabID) {
        case 'balances':
```

```
        await fetchBalances();
        break;
      case 'transactions':
        await loadBlockchainVisual();
        break;
      case 'pending':
        await fetchPendingTransactions();
        break;
      case 'dashboard':
        await fetchDashboardStats();
        break;
    }
  }
}

tabs.forEach(tab => {
  tab.addEventListener('click', () => showTab(tab.dataset.tab));
  tab.addEventListener('keydown', e => {
    let index = Array.from(tabs).indexOf(tab);
    if (e.key === 'ArrowRight') {
      e.preventDefault();
      tabs[(index + 1) % tabs.length].focus();
    } else if (e.key === 'ArrowLeft') {
      e.preventDefault();
      tabs[(index - 1 + tabs.length) % tabs.length].focus();
    } else if (e.key === 'Enter' || e.key === ' ') {
      e.preventDefault();
      showTab(tab.dataset.tab);
    }
  });
});

const initialTab = document.querySelector(".tab-link.active");
if (initialTab) showTab(initialTab.dataset.tab);

async function fetchDashboardStats() {
  try {
    const res = await fetch('/dashboard_stats');
    if (!res.ok) throw new Error('Failed to fetch dashboard stats');
    const stats = await res.json();

    document.getElementById('totalWallets').textContent = stats.total_wallets ?? '-';
    document.getElementById('totalCoins').textContent = stats.total_coins?.toFixed(2) + ' SIM'
?? '-';
```

```
    document.getElementById('totalTx').textContent = stats.total_transactions ?? '-';
    document.getElementById('totalBlocks').textContent = stats.total_blocks ?? '-';
    document.getElementById('lastMined').textContent = stats.last_mined ?? '-';
  } catch (err) {
    console.error(err);
    document.getElementById('dashboard').innerHTML = '<p>Error loading dashboard.</p>';
  }
}

const qrCanvas = document.getElementById('transactionQR');
const qrSection = document.getElementById('qrCodeSection');
const downloadBtn = document.getElementById('downloadQRBtn');

function generateTransactionQR(code) {
  if (!qrCanvas || !code) return;
  const url = `${location.origin}/transactions/${encodeURIComponent(code)}`;
  const qr = new QRious({
    element: qrCanvas,
    value: url,
    size: 200
  });
  qrSection.style.display = 'block';
  downloadBtn.onclick = () => downloadQRCode(qrCanvas, `tx_qr_${code}.png`);
}

function downloadQRCode(canvas, filename) {
  const link = document.createElement('a');
  link.download = filename;
  link.href = canvas.toDataURL('image/png');
  link.click();
}

window.generateTransactionQR = generateTransactionQR; // expose globally if needed
});
```