```
document.addEventListener("DOMContentLoaded", () => {
  const connectBtn = document.getElementById("connectBtn");
  const buyBtn = document.getElementById("buyBtn");
  const walletInfo = document.getElementById("walletInfo");
  const walletAddressSpan = document.getElementById("walletAddress");
  const cryptoSelect = document.getElementById("cryptoSelect");
  const amountInput = document.getElementById("amountInput");
  const raisedSpan = document.getElementById("raised");
  const soldSpan = document.getElementById("sold");

  // Your wallet addresses (replace these)
  const wallets = {
    ETH: "0xYourEthereumWalletHere",
    SOL: "YourSolanaWalletHere",
    BTC: "YourBitcoinWalletHere",
    USDT: "0xYourPolygonWalletHere", // Same as MATIC
    DOGE: "YourDogecoinWalletHere"
  };

  let provider, signer, walletAddress;

  connectBtn.addEventListener("click", async () => {
    const crypto = cryptoSelect.value;
    if (crypto === "ETH" || crypto === "USDT") {
      if (typeof window.ethereum !== "undefined") {
        provider = new ethers.providers.Web3Provider(window.ethereum);
        await provider.send("eth_requestAccounts", []);
        signer = provider.getSigner();
        walletAddress = await signer.getAddress();
      } else {
        alert("Install MetaMask or Trust Wallet!");
        return;
      }
    } else if (crypto === "SOL" && window.solana) {
      await window.solana.connect();
      walletAddress = window.solana.publicKey.toString();
    } else {
      walletAddress = "Manual_" + crypto; // BTC/DOGE placeholder
    }
    walletAddressSpan.textContent = walletAddress.slice(0, 6) + "..." + walletAddress.slice(-4);
    walletInfo.style.display = "block";
    connectBtn.style.display = "none";
  });
```

```
    buyBtn.addEventListener("click", async () => {
        const crypto = cryptoSelect.value;
        const amount = parseFloat(amountInput.value);
        if (!amount || amount <= 0) { alert("Enter a valid amount!"); return; }

        let txHash;
        if (crypto === "ETH") {
            const tx = { to: wallets.ETH, value: ethers.utils.parseEther(amount.toString()) };
            const txResponse = await signer.sendTransaction(tx);
            txHash = txResponse.hash;
        } else if (crypto === "USDT") {
            const usdtContract = new
ethers.Contract("0xc2132D05D31c914a87C6611C10748AEb04B58e8F", ["function
transfer(address to, uint256 value)"], signer);
            const txResponse = await usdtContract.transfer(wallets.USDT,
ethers.utils.parseUnits(amount.toString(), 6));
            txHash = txResponse.hash;
        } else if (crypto === "SOL") {
            const connection = new
solanaWeb3.Connection(solanaWeb3.clusterApiUrl("mainnet-beta"));
            const transaction = new solanaWeb3.Transaction().add(
                solanaWeb3.SystemProgram.transfer({
                    fromPubkey: window.solana.publicKey,
                    toPubkey: new solanaWeb3.PublicKey(wallets.SOL),
                    lamports: Math.floor(amount * solanaWeb3.LAMPORTS_PER_SOL),
                })
            );
            const signature = await window.solana.signAndSendTransaction(transaction);
            txHash = signature;
        } else {
            alert(`Send ${amount} ${crypto} to: ${wallets[crypto]}\nDM TX hash on X
@YourXHandle!`);
            return;
        }
        alert(`Success! TX: ${txHash}\nDM TX hash + Polygon address on X @YourXHandle!`);
        // Manual update tracker (you edit later)
    });
});
```