// Wipe Web App Frontend (Hyper Mode Fully Enabled + Backend + Season Panel + Flush Alert)

import { useEffect, useState } from "react";

import { Connection, PublicKey, clusterApiUrl } from "@solana/web3.js";

import {

AnchorProvider,

Program,

web3,

} from "@project-serum/anchor";

import idl from "../idl/wipe.json";

import { WalletAdapterNetwork } from "@solana/wallet-adapter-base";

import {

ConnectionProvider,

WalletProvider,

} from "@solana/wallet-adapter-react";

import {

WalletModalProvider,

WalletMultiButton,

} from "@solana/wallet-adapter-react-ui";

import {

PhantomWalletAdapter,

SolflareWalletAdapter,

BackpackWalletAdapter,

} from "@solana/wallet-adapter-wallets";

import axios from "axios";

require("@solana/wallet-adapter-react-ui/styles.css");

const network = WalletAdapterNetwork.Devnet;

const endpoint = clusterApiUrl(network);

const wallets = [

new PhantomWalletAdapter(),

new SolflareWalletAdapter(),

new BackpackWalletAdapter(),

];

const programID = new PublicKey(idl.metadata.address);

const connection = new Connection(endpoint);

function App() {

const [walletAddress, setWalletAddress] = useState(null);

const [program, setProgram] = useState(null);

const [xp, setXp] = useState(0);

const [wipeBalance, setWipeBalance] = useState(0);

const [nftBoost, setNftBoost] = useState(0);

const [seasonId, setSeasonId] = useState(0);

const [flushAlert, setFlushAlert] = useState(false);

const [leaderboard, setLeaderboard] = useState([]);

const [battleHistory, setBattleHistory] = useState([]);

const provider = new AnchorProvider(connection, window.solana, {

preflightCommitment: "processed",

});

useEffect(() => {

if (window.solana?.isPhantom) {

window.solana.connect({ onlyIfTrusted: true }).then(({ publicKey }) => {

setWalletAddress(publicKey);

});

}

}, []);

useEffect(() => {

if (walletAddress) {

const anchorProgram = new Program(idl, programID, provider);

setProgram(anchorProgram);

fetchUserData(anchorProgram);

fetchGlobalState(anchorProgram);

fetchLeaderboard(anchorProgram);

fetchBattleHistory();

}

}, [walletAddress]);

const fetchUserData = async (program) => {

try {

const [stakerPDA] = await PublicKey.findProgramAddress(

[Buffer.from("staker"), walletAddress.toBuffer()],

programID

);

const user = await program.account.staker.fetch(stakerPDA);

setXp(user.xp);

setWipeBalance(user.amountStaked);

setNftBoost(user.nftBoost);

} catch (e) {

console.error("Error loading user data:", e);

}

};

const fetchGlobalState = async (program) => {

try {

const stateAccount = await program.account.globalState.all();

if (stateAccount.length > 0) {

setSeasonId(stateAccount[0].account.seasonId);

const now = Math.floor(Date.now() / 1000);

if (stateAccount[0].account.flushLockedUntil > now) {

setFlushAlert(true);

}

}

} catch (e) {

console.error("Global state error:", e);

}

};

const fetchLeaderboard = async (program) => {

try {

const allStakers = await program.account.staker.all();

const sorted = allStakers

.sort((a, b) => b.account.pvpRating - a.account.pvpRating)

.slice(0, 10);

setLeaderboard(sorted);

} catch (e) {

console.error("Leaderboard fetch failed:", e);

}

};

const fetchBattleHistory = async () => {

try {

const response = await axios.get("https://wipe-api.vercel.app/battle-history/" + walletAddress);

setBattleHistory(response.data);

} catch (e) {

console.error("Battle history fetch failed:", e);

}

};

const handlePvPBattle = async () => {

try {

const tx = await program.methods

.pvpBattle(new PublicKey(walletAddress))

.accounts({ staker: walletAddress })

.rpc();

alert("Battle initiated! TX: " + tx);

fetchUserData(program);

fetchLeaderboard(program);

fetchBattleHistory();

} catch (e) {

console.error("PvP Battle Error:", e);

}

};

return (

<ConnectionProvider endpoint={endpoint}>

<WalletProvider wallets={wallets} autoConnect>

<WalletModalProvider>

<div className="min-h-screen bg-gray-900 text-white flex flex-col items-center justify-center p-4">

<WalletMultiButton />

<h1 className="text-3xl font-bold mt-6">WIPE Dashboard 🧻</h1>

<div className="bg-gray-800 p-6 rounded-xl mt-4">

<p><strong>XP:</strong> {xp}</p>

<p><strong>Staked WIPE:</strong> {wipeBalance}</p>

<p><strong>NFT Boost:</strong> x{nftBoost}</p>

<p><strong>Season:</strong> #{seasonId}</p>

{flushAlert && (

<p className="text-red-500 font-bold">🚨 Emergency Flush Active!</p>

)}

<button

onClick={handlePvPBattle}

className="mt-4 bg-yellow-500 hover:bg-yellow-600 px-4 py-2 rounded shadow"

>

PvP Battle 💥

</button>

</div>

<div className="bg-gray-700 p-6 rounded-xl mt-6 w-full max-w-lg">

<h2 className="text-xl font-bold mb-2">🏆 PvP Leaderboard</h2>

<ul>

{leaderboard.map((entry, i) => (

<li key={entry.publicKey.toBase58()}>

#{i + 1} – {entry.publicKey.toBase58().slice(0, 6)}... – {entry.account.pvpRating} RP

</li>

))}

</ul>

</div>

<div className="bg-gray-700 p-6 rounded-xl mt-6 w-full max-w-lg">

<h2 className="text-xl font-bold mb-2">📜 Battle History</h2>

<ul>

{battleHistory.map((entry, i) => (

<li key={i}>

{entry.timestamp} — {entry.result}

</li>

))}

</ul>

</div>

</div>

</WalletModalProvider>

</WalletProvider>

</ConnectionProvider>

);

}

export default App;