Coin Toss Game PRD

Overview

A decentralized coin toss game built on Base chain using USDT as the betting currency. Players can bet on heads or tails, with the winner taking the pot minus a small platform fee.

Technical Stack

- Frontend: Next.js, React, TypeScript
- Blockchain: Base (Ethereum L2)
- Smart Contract: Solidity
- Wallet Integration: Web3Modal
- Token: USDT on Base
- Development Tools: Hardhat, ethers.js
- Testing: Base Sepolia Testnet

Project Structure

```
coin-toss-game/
├── src/
│   ├── components/
│   │   └── CoinTossGame.tsx
│   ├── abis/
│   │   └── CoinTossGame_ABI.json
│   ├── types/
│   │   └── index.ts
│   └── utils/
│       └── index.ts
├── contracts/
│   └── CoinTossGame.sol
├── scripts/
│   ├── deploy.ts
│   └── verify.ts
├── test/
│   └── CoinTossGame.test.ts
├── public/
│   └── assets/
│       ├── coin-heads.png
│       └── coin-tails.png
├── hardhat.config.ts
├── package.json
└── .env
```

Complete Smart Contract Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
```

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

contract CoinTossGame is Ownable, ReentrancyGuard {
    // Constants
    uint256 public constant BET_AMOUNT = 100000; // 0.1 USDT (6 decimals)
    uint256 public constant PLATFORM_FEE_PERCENT = 1; // 1% platform fee

    // State variables
    address public player1;
    address public player2;
    bool public player1Choice;
    bool public player2Choice;
    address public winner;
    uint256 public tossNumber;
    IERC20 public usdtToken;

    // Events
    event BetPlaced(address indexed player, bool isHeads, uint256 amount, uint256 tossNumber);
    event WinnerDetermined(address indexed winner, uint256 tossNumber);
    event WinningsSent(address indexed winner, uint256 amount);

    constructor(address _usdtToken) Ownable(msg.sender) {
        usdtToken = IERC20(_usdtToken);
        tossNumber = 1;
    }

    function placeBet(bool isHeads) external nonReentrant {
        require(usdtToken.transferFrom(msg.sender, address(this), BET_AMOUNT), "USDT transfer failed");

        if (player1 == address(0)) {
            player1 = msg.sender;
            player1Choice = isHeads;
        } else if (player2 == address(0)) {
            require(msg.sender != player1, "Cannot bet against yourself");
            player2 = msg.sender;
            player2Choice = isHeads;
            _determineWinner();
        } else {
            revert("Game in progress");
        }

        emit BetPlaced(msg.sender, isHeads, BET_AMOUNT, tossNumber);
    }

    function _determineWinner() internal {
        uint256 randomNumber = uint256(keccak256(abi.encodePacked(
            block.timestamp,
            block.prevrandao,
            blockhash(block.number - 1)
```

```
        ))) % 2;

        bool isHeads = randomNumber == 0;
        winner = (isHeads == player1Choice) ? player1 : player2;

        emit WinnerDetermined(winner, tossNumber);
        _sendWinnings();
    }

    function _sendWinnings() internal {
        uint256 totalAmount = BET_AMOUNT * 2;
        uint256 platformFee = (totalAmount * PLATFORM_FEE_PERCENT) / 100;
        uint256 winnings = totalAmount - platformFee;

        require(usdtToken.transfer(winner, winnings), "Winnings transfer failed");
        require(usdtToken.transfer(owner(), platformFee), "Fee transfer failed");

        emit WinningsSent(winner, winnings);
        _resetGame();
    }

    function _resetGame() internal {
        player1 = address(0);
        player2 = address(0);
        player1Choice = false;
        player2Choice = false;
        winner = address(0);
        tossNumber++;
    }

    function resetGame() external onlyOwner {
        _resetGame();
    }

    function getState() external view returns (
        address _player1,
        address _player2,
        bool _player1Choice,
        bool _player2Choice,
        address _winner,
        uint256 _tossNumber
    ) {
        return (
            player1,
            player2,
            player1Choice,
            player2Choice,
            winner,
            tossNumber
        );
    }
}
```

Smart Contract Deployment

1. Environment Setup

```
# Create project directory
mkdir coin-toss-game
cd coin-toss-game

# Initialize npm project
npm init -y

# Install dependencies
npm install --save-dev hardhat @nomicfoundation/hardhat-toolbox
npm install @openzeppelin/contracts ethers@6.0.0
```

2. Hardhat Configuration

```
// hardhat.config.ts
import { HardhatUserConfig } from "hardhat/config";
import "@nomicfoundation/hardhat-toolbox";
import * as dotenv from "dotenv";

dotenv.config();

const config: HardhatUserConfig = {
  solidity: "0.8.20",
  networks: {
    baseSepolia: {
      url: process.env.BASE_SEPOLIA_RPC_URL || "",
      accounts: process.env.PRIVATE_KEY ? [process.env.PRIVATE_KEY] : [],
    },
  },
  etherscan: {
    apiKey: {
      baseSepolia: process.env.BASESCAN_API_KEY || "",
    },
  },
};

export default config;
```

3. Deployment Script

```
// scripts/deploy.ts
import { ethers } from "hardhat";

async function main() {
  const USDT_ADDRESS = "0xEF37f57D8a64Fd6EdF2184Ad4b2c4Cd718ec4538"; // Base Sepolia USDT

  const CoinTossGame = await ethers.getContractFactory("CoinTossGame");
  const game = await CoinTossGame.deploy(USDT_ADDRESS);

  await game.waitForDeployment();

  console.log("CoinTossGame deployed to:", await game.getAddress());
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

4. Environment Variables

```
# .env
PRIVATE_KEY=your_private_key_here
BASE_SEPOLIA_RPC_URL=https://base-sepolia.publicnode.com
BASESCAN_API_KEY=your_basescan_api_key
```

5. Deployment Commands

```
# Compile contracts
npx hardhat compile

# Deploy to Base Sepolia
npx hardhat run scripts/deploy.ts --network baseSepolia

# Verify contract
npx hardhat verify --network baseSepolia <contract_address>
"0xEF37f57D8a64Fd6EdF2184Ad4b2c4Cd718ec4538"
```

Frontend Setup

1. Project Initialization

```
# Create Next.js project
npx create-next-app@latest coin-toss-game --typescript --tailwind --eslint

# Install dependencies
npm install ethers@6.0.0 framer-motion react-hot-toast @web3modal/ethers
```

2. Environment Variables

```
# .env.local
NEXT_PUBLIC_CONTRACT_ADDRESS=your_contract_address
NEXT_PUBLIC_USDT_ADDRESS=0xEF37f57D8a64Fd6EdF2184Ad4b2c4Cd718ec4538
NEXT_PUBLIC_WALLET_CONNECT_PROJECT_ID=your_wallet_connect_project_id
```

3. Web3Modal Configuration

```
// src/utils/web3modal.ts
import { createWeb3Modal, defaultWagmiConfig } from '@web3modal/ethers/react'
import { mainnet, baseSepolia } from 'viem/chains'

const projectId = process.env.NEXT_PUBLIC_WALLET_CONNECT_PROJECT_ID || ''

const metadata = {
  name: 'Coin Toss Game',
  description: 'Play a simple coin toss game with USDT on Base',
  url: 'https://your-domain.com',
  icons: ['https://your-domain.com/icon.png']
}

const chains = [mainnet, baseSepolia]
const wagmiConfig = defaultWagmiConfig({ chains, projectId, metadata })

createWeb3Modal({ wagmiConfig, projectId, chains })
```

4. Running the Frontend

```
# Development
npm run dev

# Production build
npm run build
npm start
```

Testing the Game

1. Test USDT Setup

```
# Get test USDT from Base Sepolia faucet
# Visit: https://sepoliafaucet.com/
# Request test USDT for your wallet address
```

2. Game Testing Steps

1. Connect wallet using the "Connect Wallet" button
2. Switch to Base Sepolia network in your wallet
3. Approve USDT spending for the game contract
4. Place a bet on either heads or tails
5. Wait for another player to join
6. Watch the coin flip and see if you won

3. Common Issues and Solutions

1. **USDT Approval Failed**
    - Check if you have enough USDT
    - Ensure you're on Base Sepolia network
    - Try increasing gas limit
2. **Transaction Stuck**
    - Check network congestion
    - Try increasing gas price
    - Reset wallet connection
3. **Game Not Progressing**
    - Check if another player has joined
    - Verify contract state using getState()
    - Try resetting the game (owner only)

Security Considerations

1. **Smart Contract Security**
    - Uses OpenZeppelin's ReentrancyGuard
    - Implements proper access control
    - Handles failed transfers
    - Uses secure random number generation
2. **Frontend Security**
    - Validates all user inputs
    - Handles failed transactions gracefully
    - Implements proper error handling
    - Uses secure wallet connection
3. **Best Practices**
    - Always test on testnet first
    - Use proper error messages
    - Implement proper logging
    - Follow security guidelines

Maintenance and Updates

1. Regular Maintenance

```
# Update dependencies
npm update

# Check for security vulnerabilities
npm audit

# Run tests
npm test
```

2. Deployment Updates

```
# Deploy new version
npx hardhat run scripts/deploy.ts --network baseSepolia

# Verify new contract
npx hardhat verify --network baseSepolia <new_contract_address>
"0xEF37f57D8a64Fd6EdF2184Ad4b2c4Cd718ec4538"
```

3. Monitoring

- Monitor contract events
- Track gas usage
- Monitor error rates
- Track user engagement

Support and Resources

1. Documentation

- Base Documentation (https://docs.base.org)
- USDT Documentation (https://tether.to)
- Web3Modal Documentation (https://docs.walletconnect.com/web3modal)
- Hardhat Documentation (https://hardhat.org/docs)

2. Community Support

- Base Discord
- Ethereum Stack Exchange
- GitHub Issues

3. Development Tools

- Base Block Explorer (https://sepolia.basescan.org)
- USDT Faucet (https://sepoliafaucet.com)
- Hardhat Network (https://hardhat.org/hardhat-network)

Frontend Implementation

1. Core Components

CoinTossGame.tsx

```tsx
// src/components/CoinTossGame.tsx
'use client';

import React, { useState, useEffect, useMemo } from 'react';
import { motion } from 'framer-motion';
import Image from 'next/image';
import { ethers } from 'ethers';
import CoinTossGameABI from '@/abis/CoinTossGame_ABI.json';
import { useWeb3Modal, useWeb3ModalAccount, useDisconnect } from '@web3modal/ethers/react';
import toast, { Toaster } from 'react-hot-toast';

// Constants
const CONTRACT_ADDRESS = process.env.NEXT_PUBLIC_CONTRACT_ADDRESS || '';
const USDT_ADDRESS = process.env.NEXT_PUBLIC_USDT_ADDRESS || '';
const BET_AMOUNT = ethers.parseUnits("0.1", 6); // 0.1 USDT

// Utility functions
const shortenAddress = (addr: string) => addr.slice(0, 6) + '...' + addr.slice(-4);

export default function CoinTossGame() {
  // State management
  const [tossNumber, setTossNumber] = useState(1);
  const [player1, setPlayer1] = useState<any>(null);
  const [player2, setPlayer2] = useState<any>(null);
  const [winner, setWinner] = useState<string | null>(null);
  const [sideToShow, setSideToShow] = useState<'heads' | 'tails'>('heads');
  const [statusMessage, setStatusMessage] = useState('Choose your side');
  const [loading, setLoading] = useState(false);

  // Web3 hooks
  const { open } = useWeb3Modal();
  const { address, isConnected } = useWeb3ModalAccount();
  const { disconnect } = useDisconnect();

  // Core functions
  const handleBet = async (side: 'heads' | 'tails') => {
    try {
      setLoading(true);
      const provider = getBrowserProvider();
      if (!provider) throw new Error('No provider available');
      const signer = await provider.getSigner();
      const userAddress = await signer.getAddress();

      // Check USDT allowance
      const usdtContract = new ethers.Contract(USDT_ADDRESS, USDT_ABI, signer);
      const allowance = await usdtContract.allowance(userAddress, CONTRACT_ADDRESS);

      if (allowance < BET_AMOUNT) {
        setStatusMessage('Approving USDT...');
        const approveTx = await usdtContract.approve(CONTRACT_ADDRESS, BET_AMOUNT);
        await handleTxToast(approveTx, 'Approving USDT');
```

```
      await approveTx.wait();
    }

    // Place bet
    setStatusMessage(`Placing bet on ${side.toUpperCase()}...`);
    const contract = await getContractWithSigner();
    if (!contract) throw new Error('No contract available');

    const tx = await contract.placeBet(side === 'heads');
    await handleTxToast(tx, 'Placing Bet');
    await tx.wait();

    await fetchGameState();
    setStatusMessage("Waiting for Player 2...");
  } catch (err: any) {
    console.error(err);
    toast.error(err.reason || err.message || 'Failed to place bet');
  } finally {
    setLoading(false);
  }
};

const fetchGameState = async () => {
  if (!isConnected) return;

  try {
    const provider = getBrowserProvider();
    const contract = new ethers.Contract(CONTRACT_ADDRESS, CoinTossGameABI, provider);
    const [p1, p2, p1Choice, p2Choice, winAddr, toss] = await contract.getState();

    setPlayer1(p1 !== ethers.ZeroAddress ? { address: p1, choice: p1Choice ? 'heads' : 'tails' } : null);
    setPlayer2(p2 !== ethers.ZeroAddress ? { address: p2, choice: p2Choice ? 'tails' : 'heads' } : null);
    setWinner(winAddr === ethers.ZeroAddress ? null : winAddr);
    setTossNumber(Number(toss));
  } catch (err) {
    console.error('Failed to fetch game state:', err);
  }
};

// UI rendering
return (
  <div className="min-h-screen flex flex-col items-center justify-center p-6 bg-white text-black font-
mono">
    <Toaster />

    {/* Wallet Connection */}
    <div className="mb-4">
      {isConnected ? (
        <div className="flex gap-3 items-center">
          <span className="text-sm text-gray-600">Connected: {shortenAddress(address!)}</span>
          <button onClick={disconnect} className="text-xs px-3 py-1 bg-red-600 text-white rounded">
            Disconnect
```

```
      </button>
    </div>
  ) : (
    <button onClick={() => open()} className="px-4 py-2 bg-black text-white rounded">
      Connect Wallet
    </button>
  )}
</div>

{/* Game Status */}
<div className="flex items-center justify-between gap-2 w-full max-w-md mb-4">
  <h2 className="text-xl font-bold">Toss #{tossNumber}</h2>
</div>

{/* Coin Display */}
<motion.div
  className="w-36 h-36 border-4 border-black rounded-full flex items-center justify-center bg-gray-100"
  animate={{
    rotateX: statusMessage === "Flipping the coin..." ? [0, 360, 720, 1080] : 0,
    rotateY: statusMessage === "Flipping the coin..." ? [0, 360, 720, 1080] : 0
  }}
  transition={{
    duration: 2,
    repeat: statusMessage === "Flipping the coin..." ? Infinity : 0,
    ease: "linear"
  }}
>
  <Image
    src={`/assets/coin-${sideToShow}.png`}
    alt="coin"
    width={500}
    height={500}
    className="w-full h-full object-contain rounded-full"
  />
</motion.div>

{/* Status Message */}
<p className="mt-4 text-lg font-medium min-h-[24px]">{statusMessage}</p>

{/* Bet Buttons */}
<div className="flex gap-6 mt-6 w-full max-w-sm">
  {(['heads', 'tails'] as const).map((side) => {
    const player = player1?.choice === side ? player1 : player2?.choice === side ? player2 : null;
    return (
      <button
        key={side}
        className={`flex-1 px-4 py-3 text-white text-lg rounded-lg font-bold ${
          side === 'heads' ? 'bg-green-500' : 'bg-blue-500'
        } ${isButtonDisabled(side) ? 'opacity-40 cursor-not-allowed' : 'hover:scale-105'}`}
        onClick={() => handleBet(side)}
        disabled={isButtonDisabled(side)}
      >
```

```
            {player ? `Bet by ${shortenAddress(player.address)}` : side.toUpperCase()}
          </button>
        );
      })}
    </div>
  </div>
 );
}
```

2. Utility Functions

Contract Interaction

```
// src/utils/contract.ts
import { ethers } from 'ethers';
import CoinTossGameABI from '@/abis/CoinTossGame_ABI.json';

const CONTRACT_ADDRESS = process.env.NEXT_PUBLIC_CONTRACT_ADDRESS || '';

export async function getContractWithSigner(): Promise<ethers.Contract | null> {
 const provider = getBrowserProvider();
 if (!provider) return null;
 const signer = await provider.getSigner();
 return new ethers.Contract(CONTRACT_ADDRESS, CoinTossGameABI, signer);
}

export function getBrowserProvider(): ethers.BrowserProvider | null {
 if (typeof window === 'undefined') return null;
 const eth = (window as any).ethereum;
 if (!eth || typeof eth.request !== 'function') return null;
 return new ethers.BrowserProvider(eth, {
  name: "Base Sepolia",
  chainId: 84532
 });
}
```

Transaction Handling

```
// src/utils/transactions.ts
import { ethers } from 'ethers';
import toast from 'react-hot-toast';

export async function handleTxToast(tx: ethers.ContractTransactionResponse, actionLabel: string) {
  const explorerLink = `https://sepolia.basescan.org/tx/${tx.hash}`;

  const pendingToast = toast.loading(
    <span>
      ⏳ {actionLabel} (Pending)<br />
      <a href={explorerLink} target="_blank" rel="noopener noreferrer">
        View on BaseScan ↗
      </a>
    </span>
  );

  try {
    await tx.wait();
    toast.success(
      <span>
            {actionLabel}<br />
        <a href={explorerLink} target="_blank" rel="noopener noreferrer">
          View on BaseScan ↗
        </a>
      </span>,
      { id: pendingToast }
    );
  } catch (err) {
    toast.error(`      ${actionLabel} failed`, { id: pendingToast });
  }
}
```

3. Types and Interfaces

```
// src/types/index.ts
export interface Player {
  address: string;
  choice: 'heads' | 'tails';
}

export interface GameState {
  player1: Player | null;
  player2: Player | null;
  winner: string | null;
  tossNumber: number;
  statusMessage: string;
}
```

Game Flow

1. Initialization

1. User connects wallet
2. Frontend fetches initial game state
3. UI displays current game status

2. Betting Phase

1. Player 1 selects heads or tails
2. Frontend checks USDT allowance
3. If needed, approves USDT spending
4. Places bet on contract
5. Waits for Player 2

3. Resolution Phase

1. Player 2 joins with opposite choice
2. Contract determines winner
3. Frontend shows coin flip animation
4. Winner receives winnings
5. New game starts automatically

UI Components

1. Wallet Connection

- Connect/Disconnect buttons
- Display shortened wallet address
- Network status indicator

2. Game Status

- Current toss number
- Game state message
- Player information

3. Coin Display

- Animated coin flip
- Heads/Tails images
- Loading states

4. Bet Buttons

- Heads/Tails options
- Player status indicators
- Disabled states
- Loading states

Dependencies

```
{
  "dependencies": {
    "next": "^14.0.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "ethers": "^6.0.0",
    "framer-motion": "^10.0.0",
    "react-hot-toast": "^2.4.0",
    "@web3modal/ethers": "^3.0.0"
  },
  "devDependencies": {
    "@types/node": "^20.0.0",
    "@types/react": "^18.2.0",
    "typescript": "^5.0.0",
    "tailwindcss": "^3.0.0",
    "autoprefixer": "^10.0.0",
    "postcss": "^8.0.0"
  }
}
```