**Testing Report for Stabilizer Bot**

**Introduction**

The stabilizer bot is designed to maintain the price stability of a token by pegging it to a stable coin. The bot uses a mint/burn algorithm to maintain the price stability. In this testing report, we will test the functionality of the bot and ensure that it is working as expected.

**Test Environment**

• Remix IDE

• Programming Language: JavaScript

• Development Framework: Hardhat

• Smart Contract Platform: EthereumHardhat Localhost

• Network :- LocalHost

**Test Cases**

**Test Case 1**: Deploying Contract with Initial Supply of 100 Tokens and Transferring 100 USDT should produce price of 1 USDT

* Input: USDT Reserve = 100 USDT, Current token supply = 100 tokens
* Output: Price of the token = 1 USDT
* **RESULT : PASS**

**Test Case 2**: Mint 2 token to stimulate Low Demand Bot should Burn Tokens until price is at threshold

* Input: Price of the token = 1 USDT, Threshold = 0.01, Current token supply = 102 tokens
* Output: Burn 2 tokens and price of stablecoin is 0.998USDT and will transfer required USDT to make it 1USDT again from reserve wallet to contract
* **RESULT : PASS**

**Test Case 3**: Transfer 1 USDT to contract to stimulate High Demand Bot should Burn tokens to stabilize exceeding price

* Input: Price of the token = 1 USDT, Current token supply = 100 tokens 101 USDT in reserves
* Output: Burn 1 tokens and transfer required USDT from the contract to reserved wallet **RESULT : PASS**

**Test Case 4**: Price depreciates due to increase in USDT reserves in smart contract BOT should mint tokens adjustments to stabilize the price by equalizing supply and reserves

* **RESULT : PASS**

**Test Case 5**: Do not mint/burn tokens when the price is within the threshold

* Input: Price of the token = 0.99 USDT, Threshold price = 1 USDT, Current token supply = 99 tokens USDT reserves 99 USDT
* Output: Do not mint or burn tokens
* **RESULT : PASS**

**Conclusion**

The stabilizer bot was successfully tested and all test cases passed. The bot was able to maintain the price stability of the token by pegging it to a stable coin. The mint/burn algorithm was effective in controlling the token supply and maintaining the price stability.

**Scripts:**

**To Run Tests:**

**npx harhat node**

**npx hardhat test test/TestBot.js --network localhost**

const { expect } = require("chai");

const { ethers } = require("hardhat");

const Web3 = require('web3');

const stabilizer=require('./testingBot')

describe("StabilizerBot", function () {

  let owner;

  let add;

  let stablecoinContract;

  let usdtContract;

  before(async () => {

    [owner, add] = await ethers.getSigners();

      const USDT = await ethers.getContractFactory("MockUSDT");

      usdtContract = await USDT.deploy();

      await usdtContract.deployed();

      const Contract = await ethers.getContractFactory("Stablecoin");

      stablecoinContract = await Contract.deploy(100,usdtContract.address);

      await stablecoinContract.deployed();

    await usdtContract.transfer(stablecoinContract.address, 100);

    await usdtContract.approve(

      stablecoinContract.address,

      ethers.constants.MaxUint256

    );

  });

  it("should check the price", async function () {

    const getUSDTBalance=await stablecoinContract.balanceOfUSDT()

    const gettotalSupply=await stablecoinContract.totalSupply()

    // const price = await stablecoinContract

    console.log(getUSDTBalance.toNumber());

    console.log(parseInt(gettotalSupply.toString())/10\*\*18);

    console.log("add\_\_\_\_\_\_\_\_",add.address);

    console.log("usdtContract.address",usdtContract.address);

    console.log("stablecoinContract.address",stablecoinContract.address);

  try{

  let check=  await stabilizer.stabilize(usdtContract.address,stablecoinContract.address)

  console.log(check);

  expect(check).to.equal(1);

}

  catch(e){

    console.log(e);

  }

  });

  it("should not Mint or Burn if price is 1", async function () {

    const TargetPrice = 1;

    try{

      let check=  await stabilizer.stabilize(usdtContract.address,stablecoinContract.address)

      console.log(check);

      expect(check).to.equal(TargetPrice);

    }

      catch(e){

        console.log(e);

      }

  });

  it("should adjust the supply by minting stablecoin", async function () {

  //  ustabilize stablecoin by increasing price by transfering usdt

    await usdtContract.transfer(stablecoinContract.address, 100);

    try{

      let check=  await stabilizer.stabilize(usdtContract.address,stablecoinContract.address)

      console.log(check);

      expect(check).to.equal(1);

    }

      catch(e){

        console.log(e);

      }

  });

  it("should adjust the supply by burning stablecoin", async function () {

    const Mint = await stablecoinContract.mintAdjust("10000000000000000000")

    try{

      let check=  await stabilizer.stabilize(usdtContract.address,stablecoinContract.address)

      console.log(check);

      expect(check).to.equal(1);

    }

      catch(e){

        console.log(e);

      }

  });

});

