Hashim Tayyab (19L-1013)

Hassan Hadayat(19L-0940)

**Project Phase 4 Documentation**

In the 4th phase of the project, we worked on the Testing and Test Automation in web3.

For this project we used hardhat, a development environment that facilitates building on Ethereum. It allows to manage and automate the recurring tasks that are inherent to the process of building smart contracts and dApps.

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

const { expect }  = require("chai");

We use hardhat and chai library for assertions and perform testing. The project we chose was an NFT Minting smart contract written in solidity.

We different several different functionalities of the smart contract such as checking the correct name and symbol for NFT, showing the current supply, minting function working properly, no minting once Max Supply has reached and update the NFT Token URI after each NFT is minted.

The functions shown below perform the testing for name and symbol, supply and NFT minting functionality:

  describe("Deployment", function () {

    it("Should set the correct name and symbol", async function () {

      expect(await myToken.name()).to.equal("Super Hat Club");

      expect(await myToken.symbol()).to.equal("SHAC");

    });

    it("Should show current supply 0", async function () {

      expect(await myToken.totalSupply()).to.equal(0);

    });

  });

  describe("Minting", function () {

    it("Should mint a new token when called by owner", async function () {

      await myToken.connect(owner).safeMint(addr1.address, { value: ethers.utils.parseEther("0.000001") });

      expect(await myToken.balanceOf(addr1.address)).to.equal(1);

    });

Below is a function example of only 40 NFTs can be minted:

    it("Should not mint a new token when the maximum supply has been reached", async function () {

        const maxSupply = 40;

        for (let i = 0; i < maxSupply; i++) {

          await myToken.connect(owner).safeMint(addr1.address,

            { value: ethers.utils.parseEther("0.000001") });

        }

        await expect(myToken.connect(owner).safeMint(addr1.address,

            { value: ethers.utils.parseEther("0.000001") }))

            .to.be.revertedWith("Sorry, all NFTs have been minted!");

        });

In the example below URI update is being checked:

    it("Should update the tokenURI when minting a new token", async function () {

        await myToken.connect(owner).safeMint(addr1.address, { value: ethers.utils.parseEther("0.000001") });

        const tokenId = await myToken.tokenOfOwnerByIndex(addr1.address, 0);

        const tokenURI = await myToken.tokenURI(tokenId);

        expect(tokenURI).to.equal("ipfs://QmSSAvQfq5QPMGcwqVu3xGQfLXNaYkTPKjRKcn3PAL3HjG/1.json"); });

The following loom recording shows the hardhat testing on NFT Minting contract using hardhat and chai library:

<https://www.loom.com/share/a10ea4212174482f911704c1a2c87610>