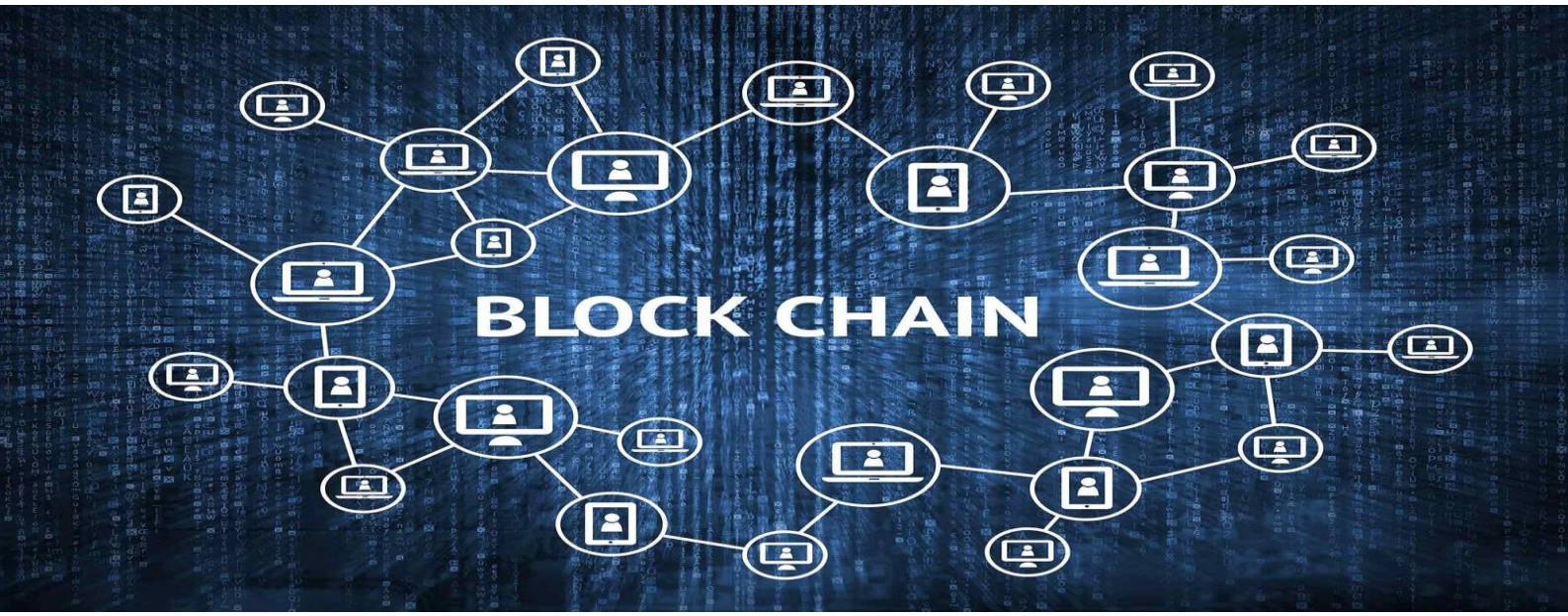
BlockChain with NFT



R E P O R T

과목명 캡스톤디자인-sw
담당교수 박용범 교수님
팀 팀장 소프트웨어학과 32181141 김창오
팀원 소프트웨어학과 32144794 최원제
팀원 소프트웨어학과 32180217 구선우
팀원 소프트웨어학과 32180086 강은솔

차 례

1. Smart Contract

(1) Smart Contract 이해

Smart Contract(이하 스마트 컨트랙트)는 계약 당사자 간 거래 내용을 코드로 기록해 블록체인에 업로드 하면, 조건이 충족됐을 때 계약을 자동으로 이행해 주는 시스템이다. 이는 블록체인 네트워크의 모든 노드에서 동일한 방식으로 실행되고, 이는 신뢰성과 안전성을 보장한다. 배포된 스마트 컨트랙트는 수정이 불가하며 블록체인의 불변성에 의해 변경될 수 없다.

- Smart Contract 등장 배경

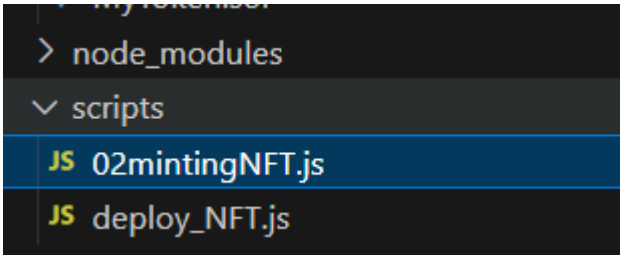
기존 계약 방식에는 많은 문제가 있었다. 예를 들어 온라인으로 주식을 거래하거나 해외에 송금을 할 때, 언제나 계약 중간에 '중개자'가 존재한다. 이들은 매수자 및 매매자의 거래에 신뢰성을 더해주는 동시에 수수료가 발생하게 한다. 이러한 과정은 여러 단계를 거쳐 진행되며, 필연적으로 단계가 많아질수록 비용이 증가하며, 이는 비효율적인 손실을 가져오게 된다. 이를 해결하기 위한 방안 중 하나가 바로 블록체인을 이용한 스마트 컨트랙트이다.

2. Hardhat 활용 CP.2

(1) Deploy your SC on a testnet

지난 번 컴파일 한 스마트 컨트랙트를 deploy 하는 과정이다.

1. Scripts 폴더 생성 후 deploy_NFT.js 파일 작성



2. 오픈소스로 배포된 코드를 deploy 내에 삽입

```
const main = async () => {
    // const nft_token_contract = await hre.ethers.getContractFactory('MyToken');
    const nft_token_contract = await ethers.getContractFactory('MyToken');

    const contract_deployed_ = await nft_token_contract.deploy("METAMASK ACCOUNT ADDRESS");

    console.log('Contract deployed to:', await contract_deployed_.getAddress());

};
const runMain = async () => {
    try {
        await main()
        process.exit(0);
    } catch (error) {
        console.log(error);
        process.exit(1);
    }
};
runMain();
```

3. **METAMASK ACCOUNT ADDRESS 부분에 본인의 metamask 지갑 링크를 삽입해준다.**

```
const main = async () => {
    // const nft_token_contract = await hre.ethers.getContractFactory('MyToken');
    const nft_token_contract = await ethers.getContractFactory('MyToken');

    const contract_deployed_ = await nft_token_contract.deploy("0x0E0A7449717AC4B1B282aD0f3E2E10c2aCb18297");

    console.log('Contract deployed to:', await contract_deployed_.getAddress());

};
const runMain = async () => {
    try {
        await main()
        process.exit(0);
    } catch (error) {
        console.log(error);
        process.exit(1);
    }
};
runMain();
```

4. **이전 디렉토리로 되돌아 간다.**

(2) Add the networks

Sepolia와 Besu 중 사용하는 네트워크를 추가해 줄 것이다.

1. 지난 실습 때 만들었던 hardhat.config.js 파일에 오픈소스 네트워크 코드를 삽입한다.

```
require("@nomicfoundation/hardhat-toolbox");
require('@openzeppelin/hardhat-upgrades');
/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.20",
  networks: {
        // sepolia: {
        //   url: `https://sepolia.infura.io/v3/INFURA_API_KEY`,
        //   accounts: [SEPOLIA_PRIVATE_KEY]
        // },
        NAME_OF_BESU_NETWORK: {
              url: `JSONRPC-ADDRESS_FOR_BESU_NODE`,
              chainId: 1337,
              accounts: ["PRIVATE_KEY_FROM_METAMASK_ACCOUNT"]
            }
      }
};
```

2. Besu는 사용하지 않으니 주석처리 된 Sepolia 부분을 활성화 시켜준다

```
require("@nomicfoundation/hardhat-toolbox");
require('@openzeppelin/hardhat-upgrades');


/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.20",
  networks: {
        sepolia: {
          url: `https://sepolia.infura.io/v3/63cb9afb610146c4a497719c317d2ef7`,
          accounts: ["597f[redacted]"]
        },

    }
};
```

3. url 부분에는 infura의 개인 키 url을, accounts에는 metamask의 개인 키 값을 삽입한다

```
networks: {
    sepolia: {
      url: `https://sepolia.infura.io/v3/63cb9afb610146c4a497719c317d2ef7`,
      accounts: ["597f[redacted]"]
    },
```

4. 모두 완료되었다면 npx에서 sepolia 네트워크를 추가해주는 명령어를 실행해준다

```
C:₩Users₩Weak_Eunsol₩NFT_HT>npx hardhat run scripts/02mintingNFT.js --network sepolia
Token Symbol: MTK
Token Name: MyToken
Token Contract Owner: 0xB856Cdf82f5cd482F621028C4483eD6C0a6a522B
ProviderError: execution reverted
    at HttpProvider.request (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩hardhat₩src₩internal₩core₩providers₩http.ts:90:21)    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at HardhatEthersProvider.estimateGas (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩@nomicfoundation₩hardhat-ethers₩src₩internal₩hardhat-ethers-provider.ts:237:27)
    at C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩@nomicfoundation₩hardhat-ethers₩src₩signers.ts:235:35
    at async Promise.all (index 0)
    at HardhatEthersSigner._sendUncheckedTransaction (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩@nomicfoundation₩hardhat-ethers₩src₩signers.ts:256:7)
    at HardhatEthersSigner.sendTransaction (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩@nomicfoundation₩hardhat-ethers₩src₩signers.ts:125:18)
    at send (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩ethers₩src.ts₩contract₩contract.ts:313:20)
    at Proxy.safeMint (C:₩Users₩Weak_Eunsol₩NFT_HT₩node_modules₩ethers₩src.ts₩contract₩contract.ts:352:16)
    at main (C:₩Users₩Weak_Eunsol₩NFT_HT₩scripts₩02mintingNFT.js:21:26)

C:₩Users₩Weak_Eunsol₩NFT_HT>
```

(3) Minting NFT

지급받은 minting.js 파일을 scripts에 추가해준다

```
const hre = require("hardhat");

async function main() {
  try {
    // Get the ContractFactory of your Smart Contract
    const SimpleContract = await hre.ethers.getContractFactory("MyToken");

    // Connect to the deployed contract
    const contractAddress =  "0x3AD78d89eF2AB4417947e3114Cf44D1f69C1533D"; // Replace with your deployed contract address

    const contract = await SimpleContract.attach(contractAddress);

    // Call some basic functions
    const TokenSymbol = await contract.symbol();
    console.log("Token Symbol:", TokenSymbol);
    const TokenName = await contract.name();
    console.log("Token Name:", TokenName);
    const TokenContractOwner = await contract.owner();
    console.log("Token Contract Owner:", TokenContractOwner);
    //Minting the NFT with account address, index and metadata
    const TokenMinting = await contract.safeMint("0x0E0A7449717AC4B1B282aD0f3E2E10c2aCb18297",1,"https://ipfs.io/ipfs/QmcWY5YFqekeRi6D3QC
    console.log("Token Minted Succesfully");
    console.log(TokenMinting);
    } catch (error) {
    console.error(error);
    process.exit(1);
  }
}

main();
```

이 때, 각 요소들을 조정함으로써 발급하는 NFT의 개수의 한도를 정하는 등, 커스텀이 가능하다

5. 실습 결론 및 소감

이름	내용
김창오	이번 시간은 대학원생 선배님의 마지막 강의로 hardhat을 사용해 deploy하는 방법까지 배웠다. 지난시간에 배웠던 스마트 컨트랙트를 컴파일 하는것에 이어 deploy까지 진행하면서 etherscan에 직접 컨펌된 것까지 확인하였다. 추가로 IPFS를 활용해 NFT를 민팅하는 것까지 배웠다. 3주 동안 메타마스크 활용, REMIX를 활용한 스마트 컨트랙트, NFT민팅에서부터 직접 코드로 로컬환경에서 위와 같은 행동을 할 수 있게 되어 신기하였다. 현재 진행하고 있는 홈페이지에 NFT를 민팅하는 기능이 필요하였는데, 선배님의 도움 덕분에 잘 진행할 수 있다고 생각한다. 추가로 티켓 형식의 NFT를 구상하고 있는 만큼, NFT 수량을 한정 짓는 방법도 생각해봐야 할 것 같다.
최원제	이번 주 실습에서는 로컬 스마트 컨트랙트 환경에서 hardhat.config.js 파일에 INFURA와 SEPOLIA 키 값을 입력하고 deploy와 safeMinting 함수 코드들을 작성하여 해당 로컬 환경에서 NFT를 생성하는 기능을 구현해보았다. 이를 바탕으로 현재 진행하고 있는 NFT 관련 웹 서비스 프로젝트에서도 NFT 발행 개수 제한과 같은 세부적인 기능들을 구현할 수 있도록 그 방법을 의논하는 시간을 가져보았다. 실습 초반에는 NFT에 대한 개념 자체가 생소했기에 이를 진행하는데 있어 걱정이 많았지만 실습이 진행되면서 NFT에 대한 이해와 함께 이를 응용할 수 있는 기반을 다질 수 있어 유용했다. 이후로는 우선적으로 실습 과정에서 다룬 로컬 스마트 컨트랙트 환경 구축을 복습하여 확실한 이해를 가지고 일어날 수 있는 오류 사항들에 대해 대처 방안을 학습할 예정이다. 그 이후 위에서 말한 NFT 발행 개수 제한 외에도 프로젝트에 필요한 세부 기능들을 설계 및 구현하여 더 발전된 스마트 컨트랙트 환경을 구축할 예정이다.
구선우	이번 시간에는 지난 시간에 이어 hardhat을 사용하는 법에 대해 배웠다. 컴파일한 이후 NFT를 deploy하고 그것을 minting하는 과정까지를 학습했다. 간단하게 deploy와 minting을 어떤 방식으로 구현할 수 있는지를 간단한 코드와 함께 보기도 했다. 지난 3주간 NFT를 배우며 deploy와 mint를 직접 해보며 그 과정을 배우긴 했지만 뭔가 이해가 막연했었는데 이번에 코드를 보며 간단하게나마 구현하는 방식을 보니 이해가 좀 더 뚜렷하게 된 것 같다. 그리고 지금까지 배운 것들을 어떻게 응용할 수 있는지를 보며 흥미를 느낄 수 있었고 우리 프로젝트에는 어떻게 적용해볼 수 있을까도 생각해보게 되었다.
강은솔	마지막 강의였던 이번 주는 지난 시간 컴파일 했던 hardhat 프로젝트를 최종적으로 deploy 하는 법과 실습 후 확인, 그리고 스마트 컨트랙트 환경에서 NFT의 민팅 방식, 어떤 함수를 조작하면 민팅 설정을 조정할 수 있는지 등에 대해 배웠다. 처음 강의를 듣기 전, 접해본 적 없는 생소한 공부였기에 기대보단 걱정이 조금 앞섰지만 교수님께서 간단하게 알려주신 사전 설명과 대학원생 선배님의 자료 및 강의가 있었기에 흥미를 느끼며 공부할 수 있었던 것 같다. 특히 '스마트 컨트랙트' 라는 매커니즘 자체가 매우 효율적이고 매력적인 구조로 느껴졌고, 이런 환경을 직접 구현하고 설정 및 서비스 할 수 있다는 점이 가장 흥미로운 점으로 다가왔다. 3주간 배운 것들을 이용하여 현재 진행중인 NFT 티켓에 적용한다면 분명 인상적인 결과물이 나올 것이라는 생각이 들었고, 그만큼 고려해야할 점도 꽤나 생각이 나 문제 해결에 많은 도움이 되었던 것 같다.