캡스톤디자인 최종보고서

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. 프로젝트 개요

1.1 문제 정의

최근 다양한 행사에서 암표 거래와 같은 부적절한 티켓팅 방식이 만연하고 있다. 이는 멀리서 벌어지는 일이 아니라 우리 가까이에도 존재한다. 최근 대학교 축제에 초청된 연예인 공연을 가까이서 볼 수 있는 입장권이 중고나라, 에브리타임 등에서 암표형식을 거래가 되고 있기 때문이다. 이러한 암표 거래는 정가보다 훨씬 높은 가격에 티켓이 거래되면서 행사에 참여하려는 일반 관객들이 높은 비용을 지불해야 하는 문제를 야기한다.

또한, 티켓의 진위 여부를 확인하기 어렵기 때문에 가짜 티켓으로 인한 피해 사례 역시 증가하고 있다. 특히 종이 형식의 행사 참여 티켓의 경우 위변조가 쉽고, 각 티켓마다 시리얼 넘버가 존재하여도 하나하나 확인이 어려운 점을 이용하여 가짜 티켓을 활용한 악용이 만연한 상황이다.

이와 같은 암표 거래 문제는 행사 주최 측뿐만 아니라 참가자들에게도 큰 피해를 입히고 있다. 행사 주최 측의 경우, 티켓 판매 수익으 감소와 더불어 브랜드 이미지의 실추를 겪게 되며, 참가자들은 높은 비용을 지불하고도 정당한 서비스를 받지 못하는 상황에 처하게 된다.
우리는 이러한 문제를 해결하기 위한 새로운 티켓팅 방식이 필요하다고 판단하였다.

1.2 개발 배경

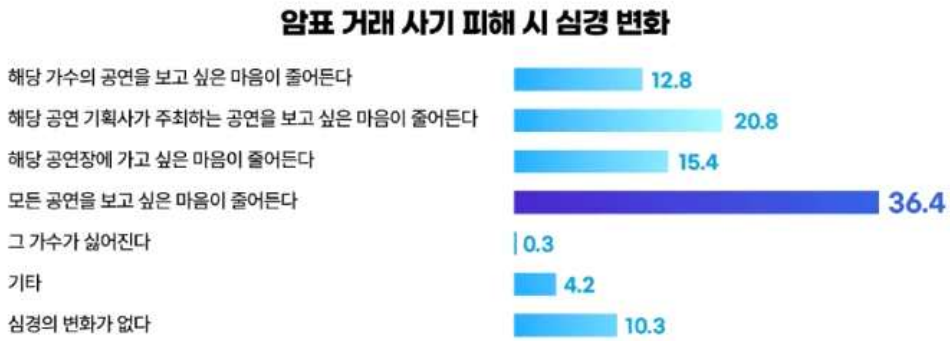
이 프로젝트의 개발 아이디어는 블록체인 기술의 대표적인 응용 분야인 NFT(Non-Fungible Token)의 특성을 활용하는 것이다. NFT는 고유한 디지털 자산을 표현하며, 복제나 위변조가 불가능한 특성을 가지고 있다.
이러한 특성을 티켓팅 시스템에 접목하면 암표 거래와 가짜 티켓 문제를 효과적으로 해결할 수 있다고 판단하였다.

프로젝트를 진행하기에 앞서 시장조사를 해본 결과, 현재 시장에는 다양한 티켓팅 시스템이 존재하지만, 대부분 중앙화된 방식으로 운영되고 있어 투명성과 신뢰성 면에서 한계가 있다는 것을 알게 되었다.
또한 기존의 방식들은 티켓의 진위 여부를 확인하는 과정이 복잡하며, 암표 거래를 완전히 막기에는 역부족한 경우가 많았다. 이러한 상황에서 공연 및 음악에 관심이 있는 이용자들을 종종 티켓 구매를 경험할 때 사기 및 암표로 인한 피해를 보기도 한다.

아래의 자료는 한국음악레이블산업협회에서 2023년 3월 전국 남녀 572명을 대상으로 '공연 예매 및 암표 거래에 대한 이용자 의견'을 조사한 것이다.

이용자 중 23.4%는 공식 예매처 외 티켓 구매 경험이 있다고 응답하였다. 중고나라, 당근과 같은 2차 거래 시장이 얼마나 활성화되어 있는지 알 수 있다. 이러한 2차 거래 시장은 가격을 정하는 기준이 개인에게 달려 있기 때문에 추가 지불이 필수적인게 현실이다.

따라서 다음 자료를 살펴 보면, 공식 예매처 외 티켓을 구매할 때 약 2명중 한 명은 1~5만원에 달하는 금액을 추가로 지불하고 티켓을 구매하는 것이다. 더욱 놀라운 점은 10만원이 넘는 거액을 지불하고 티켓을 구매한 비율이 14.4%나 된다는 것이다.



암표거래가 불평등한 티켓 가격을 형성하면서 이용자들은 다양한 심경 변화를 겪고 이는 음악 및 공연 산업에도 영향을 미친다. '암표 거래 지출 증가시 변화'설문에서 변화 없다고 대답한 19.1%을 제외한 80.9%에서 공연에 직 간접적인 부정적 영향을 미치는 것으로 나타났다.

가장 충격적인 부분은 이렇게 불합리한 소비를 감수하고 암표를 구매하더라도 26.1%의 사람은 제대로된 티켓을 받지 못했다는 것이다. '공식 예매처 외 티켓 구매 시 사기 피해 경험'설문에서는 입금 후 티켓을 주지 않는 단순 사기부터 중복 양도, 공연 취소 등으로 사기를 당한 경험이 있다고 설문하였다.

해당 시장조사를 마치고 우리 팀은 NFT를 활용한 티켓팅 시스템이 기존 시스템이 가지고 있었던 문제를 예상했던 것보다 효과적으로 해결할 것이고, 많은 사회적 비용을 절감할 것이라고 생각하였다.

1.3 개발 시스템 요약

본 프로젝트는 NFT를 활용하여 행사 티켓의 발행 및 진위여부를 관리하는 웹서비스를 개발하라고 한다.
이와 같은 서비스를 제공하기 위해 User, Owner, Admin 3가지 권한을 부여하여 안정적인 서비스를 제공하도록 한다.

User는 일반 사용자로, 행사를 참여하기 위해 행사 목록을 확인하고 원하는 행사의 NFT를 Mint를 요청하여 참여하는 주체이다.

Owner는 행사 주최자로, 웹서비스를 통해 행사를 등록하고, 행사 인원을 모집한다.

Admin는 관리자로, Owner의 행사 등록 요청을 승인하고, Smart Contract 및 웹 안정성을 관리하는 주체이다.

결과적으로 3가지 권한이 유기적으로 작동하여 안정적이고 효율적인 NFT 티켓팅 시스템을 구성하게 된다.

2. 개발관련 기술

위에 언급했듯이 본 프로젝트는 NFT를 기반으로한 티켓팅 웹 서비스이다. 따라서 프로젝트 진행을 위해서는 NFT와 웹 서비스에 대한 이해도가 모두 필요하다. 그렇기 때문에 우리팀은 NFT가 어떠한 방식으로 발행되는지에 대해서와 이를 티케팅 웹서비스에 접목시키기 위해서 어떻게 웹 프로젝트를 구성해야하는가에 대해서 학습하였다.

2.1 NFT관련 기술

2.1.1. Digital Wallet

가상화폐도 화폐의 기능을 하기 때문에 이를 저장하는 수단이 필요한데 이를 Digital wallet(전자 지갑)이라한다. 전자 지갑은 가상화폐를 저장하고 송금하는데 사용하며, 블록체인 기술을 기반으로 하여 개인의 고유한 암호키를 사용하여 자산을 안전하게 보호한다.

본 프로젝트에서는 Metamask라는 웹브라우저 지갑을 사용하기로 했다. 설치 후 간단하게 등록하고 사용할 수 있어 전자 지갑 사용에 익숙하지 않은 사람이 쓰기 적합하기 때문에 Metamask를 사용하기로 결정했다.

등록 절차를 거친 후 로그인하면 좌측과 같은 모습을 볼 수 있다.

1.은 지갑의 주소로 계좌번호와 같은 역할이다.

2.에서 현재 접속한 네트워크가 어디인지 알 수 있다.

3.은 현재 남아 있는 잔액을 표시한다.

2.1.2. Testnet

본 프로젝트는 NFT를 활용하기 위한 블록체인 플랫폼으로 Ethereum을 사용하기로 했다.
그렇기 때문에 NFT를 발행하는 과정에서 비용으로 ETH가 소모되는데, 이를 비용없이 수행하기 위해 Ethereum의 Testnet인 Sepolia network를 사용하기로 결정했다.

Testnet은 블록체인 기술을 개발하고 테스트하는데 활용되는 네트워크이다. Sepolia network에서는 Faucet을 통해 SepoilaETH를 무료로 받아 테스트 환경에서 사용할 수 있어 비용없이 NFT를 발행해볼 수 있다.

2.1.3. IPFS

IPFS는 InterPlanetary File System의 약자로, 분산형 파일 시스템에 데이터를 저장하고 공유하기 위한 프로토콜이며, 중앙화된 서버가 없는 파일 공유 P2P 네트워크이다.

IPFS는 파일을 블록 단위로 분할한 후 각 블록을 해싱한다. Merkle Directed Acyclic Graph(DAG) 구조로 파일을 구성한다. 이 구조는 파일의 각 블록을 그 이전의 블록의 해시 값으로 참조하는 구조이다. 이러한 블록 조합이 끝나면 이를 해싱하여 해시 값을 파일의 루트 해시(CID)로 사용한다. 그 다음 파일의 각 블록을 네트워크에 분산하여 저장한다. 저장한 파일을 찾을 때는 CID를 검색해 블록을 찾는다.

이처럼 IPFS는 해싱을 통해 고유한 식별자를 생성하여 파일의 고유성이 보장되며, 각 파일에 대한 고유한 주소도 만들 수 있다. 이러한 특성이 NFT의 고유성과 잘 맞아 NFT를 생성할 때 파일을 저장하는 형태로 적합하다. 또한 중앙화된 서버에 의존하지 않고 분산된 저장소를 이용하기 때문에 파일의 안정성이 향상되고, 비용도 줄어든다는 장점이 있다.

2.1.4. Pinata

NFT를 발행하기 위해서는 NFT의 메타데이터를 가지고 있는 JSON 파일이 필요하다. 또한 본 프로젝트에서는 이미지를 기반으로 NFT를 발행할 예정이기 때문에 JSON 파일에 이미지의 정보도 포함되어야한다. 이를 위해서 NFT 발행에 사용할 이미지와 발행할 NFT의 메타데이터를 담은 JSON 파일을 IPFS의 형태로 저장할 필요가 있다. 이를 위해 Pinata라는 도구를 사용하기로 했다.

Pinata는 파일을 업로드하면 해당 파일을 IPFS로 저장해주는 웹 서비스이다. 가입 후 파일을 업로드하는 형태로 매우 간단하게 활용할 수 있다. Pinata를 통해 NFT에 활용될 이미지 파일과 JSON 파일을 IPFS 형태로 변환할 수 있다.

2.1.5. Remix IDE

Remix(Ethereum IDE)는 이더리움 스마트 계약을 개발하고 디버깅하는데 사용되는 온라인 도구이다. 웹 브라우저에서 실행되며 Solidity언어로 작성된 스마트 계약을 만들고 테스트할 수 있는 환경을 제공한다. Remix는 EVM(이더리움 가상머신)과 상호작용하여 스마트 계약을 배포, 실행, 디버깅한다. Remix는 아래와 같은 주요 기능들을 가진다.

편집기는 Solidity언어를 사용하여 스마트 계약을 편집할 수 있는 기능을 제공한다.

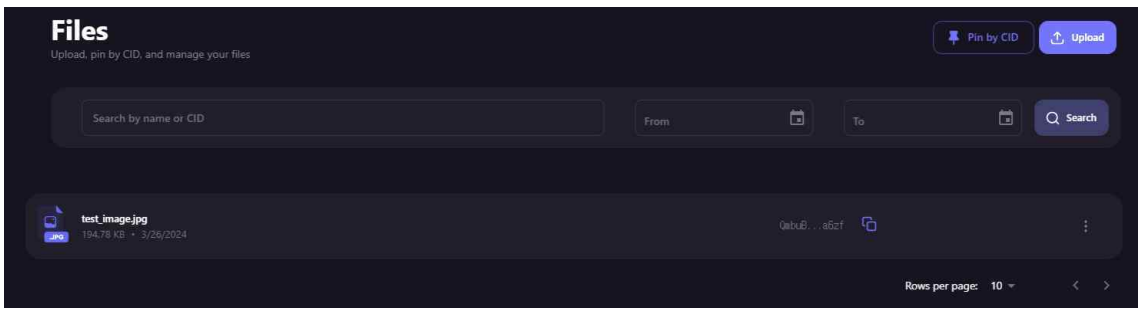
컴파일러는 스마트 계약을 컴파일하고 컴파일된 결과를 확인할 수 있는 기능을 제공한다.

디버거는 스마트 계약을 디버깅하고 실행하는데 사용되는 기능을 제공한다. 사용자는 계약의 각 단계에서 상태를 확인하고 실행 흐름을 분석할 수 있다.

또한 다양한 플러그인을 지원하여 추가 기능을 확장할 수 있다. 예시로 Metamask와의 통합을 통해 스마트 계약을 배포하고 테스트할 수 있다.

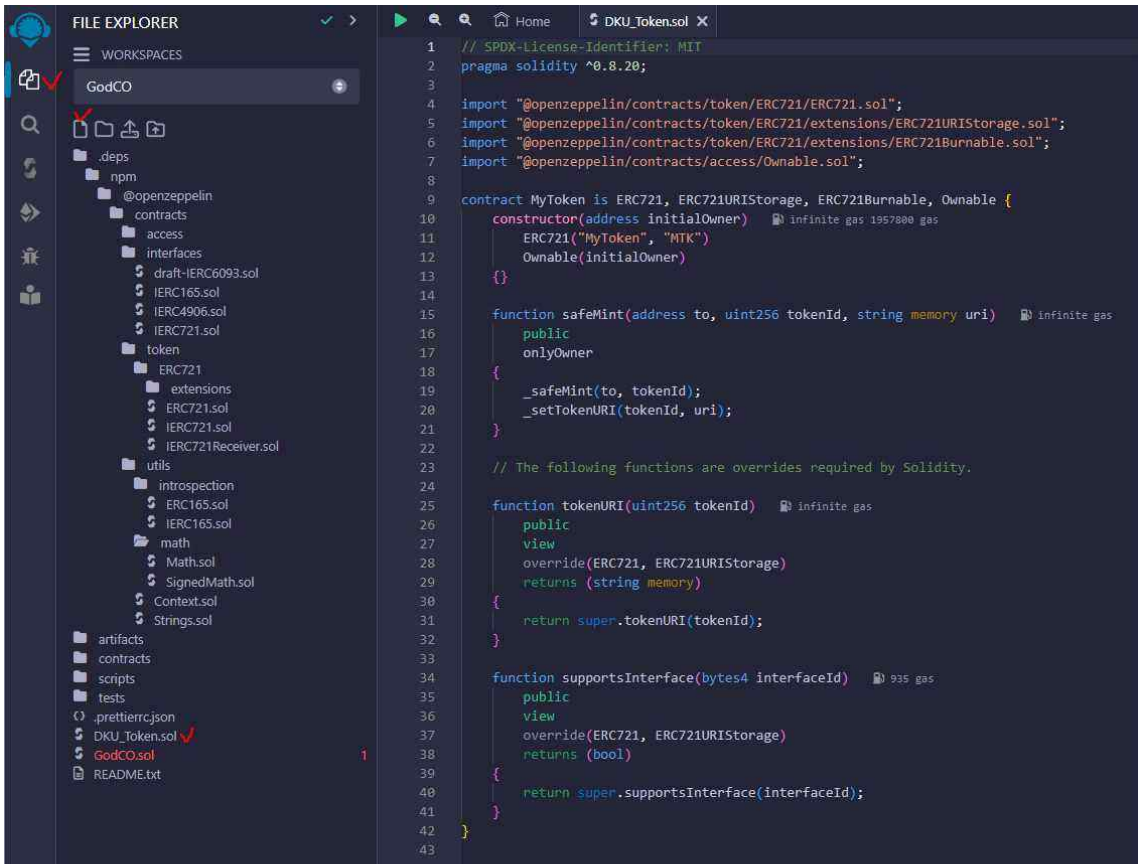
2.1.6. NFT 직접 발행

위에서 학습한 내용들을 활용하여 NFT를 직접 발행해보았다. 먼저 Pinata를 통해 NFT에 사용될 이미지를 업로드했다.

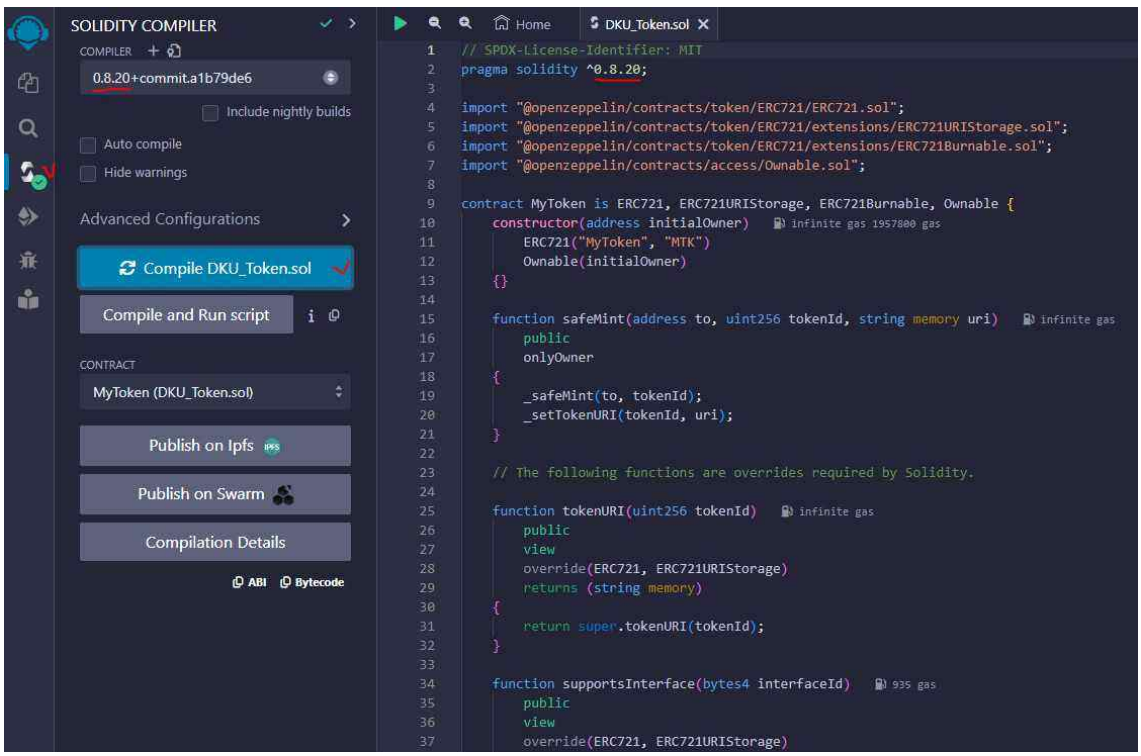


그 다음 이미지의 IPFS 주소를 JSON 파일에 입력하고 JSON 파일도 Pinata로 업로드했다.

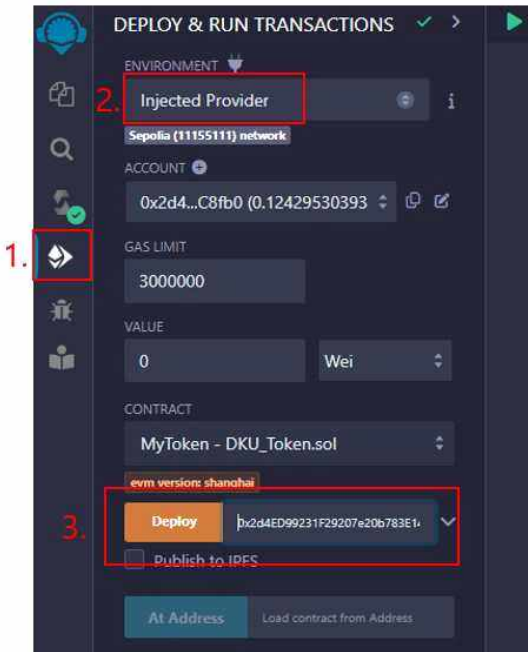
업로드가 끝났다면 Remix를 활용하여 본격적으로 NFT를 발행하는 과정으로 들어간다. 먼저 Solidity 언어로 작성된 소스파일을 Remix에 업로드했다. 소스파일은 학습과정에서 제공받은 소스를 사용하였다.



알맞은 버전의 컴파일러를 사용해서 해당 소스파일을 컴파일했다.



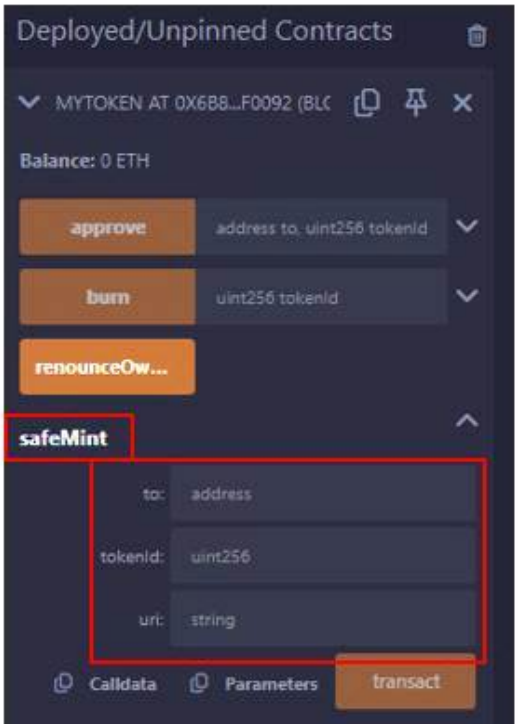
Deploy&run transaction 탭에 가서 NFT를 deploy하고 transaction을 발생시켰다.



환경은 Metamask를 사용했기 때문에 Injected Provider를 선택해주었다.

Deploy에는 본인의 Metamask를 입력해주었다.

transaction까지 완료한 다음, Deployed/Unpinned Contracts 탭을 열어 발행된 NFT를 본인의 Metamask 주소로 Mint해주었다.



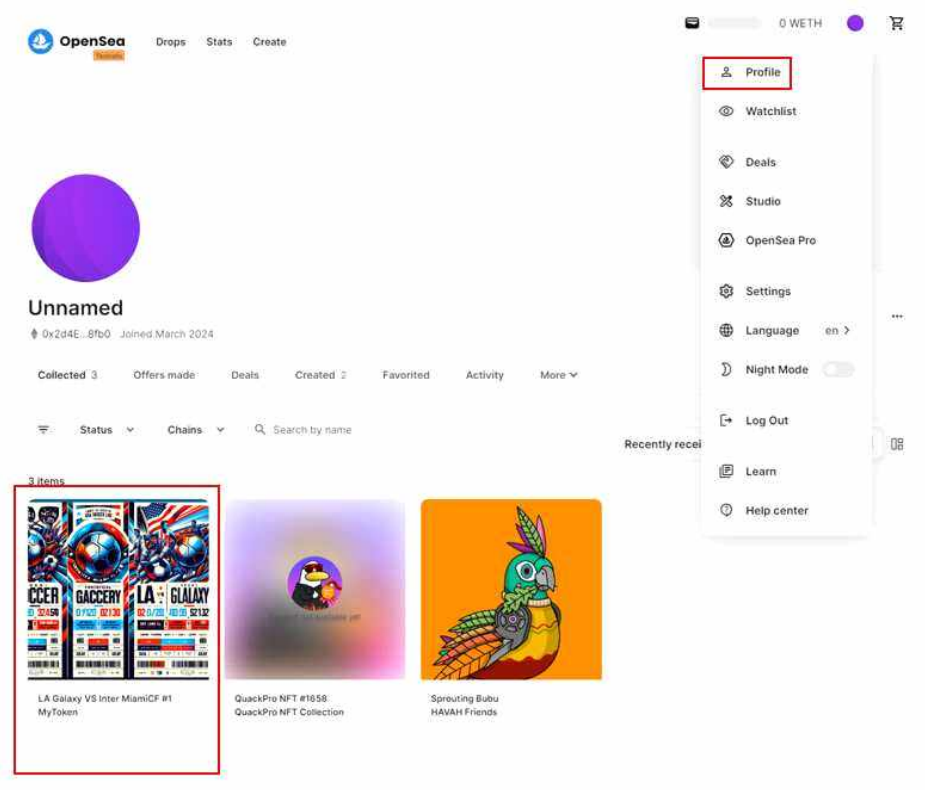
to에는 NFT를 받을 주소인 본인의 Metamask 주소를 입력해주었다.

tokenId는 토큰을 고유하게 식별해줄 숫자이다. 이번에는 1로 설정해주었다.

uri에는 이전에 올려둔 JSON 파일의 주소를 입력해주었다.

위 세가지를 입력한 후 transact 버튼을 눌러 발행한 NFT가 본인의 Metamask에 전송되도록 하였다.

이렇게 발행한 NFT가 잘 전송했는지 확인하기 위해 OpenSea를 활용했다. 아래는 OpenSea의 Testnet에 접속하여 NFT가 본인의 지갑에 잘 들어왔는지 확인한 모습이다.



2.2. Web 개발 관련 기술

2.2.1. Figma

Figma는 웹 기반 UI/UX 디자인 협업 도구이다. 웹 기반 도구이기 때문에 설치 없이 간편하게 이용할 수 있으며, 공유 프로그램 없이 계정 소유자의 링크 공유를 통해 웹 브라우저에서 여러 명이 동시에 실시간으로 작업할 수 있다는 것이 특징이다.

아트 보드에서 사용할 프레임을 만들고 안에 디자인 요소들을 넣는 방식으로 프로토타입을 디자인할 수 있다.

사용 방식이 간단하며 편하고 효율적으로 협업할 수 있는 도구라 이번 프로젝트 프로토타입 제작에 활용하였다.

2.2.2. Thymeleaf

템플릿 엔진의 일종으로 html 태그에 속성을 추가해서 동적인 렌더링을 구현해낼 수 있다. 스프링과 통합되어 있어, 스프링의 여러 기능들을 간편하게 사용할 수 있도록 지원한다.

처음에는 React를 사용하려 했으나, 팀 구성원 중 타임리프를 이용한 프로젝트 개발을 경험해본 인원이 있었고, NFT라는 생소한 것을 활용한 프로젝트를 진행하는 상황이기 때문에 경험이 없는 React를 새로 익히면서 하기 보다는 이미 경험이 있는 타임리프를 사용함으로써 NFT에 대한 학습에 좀 더 비중을 두기 위해 타임리프를 채택하여 프로젝트를 진행하기로 결정하였다.

2.2.3. Spring Boot

스프링 부트는 스프링을 이용하는 프로젝트를 만들 때 간편하게 설정할 수 있게 해주는 프레임워크이다. 스프링은 Java 기반 애플리케이션 개발을 편하게 할 수 있게 해주는 오픈소스 애플리케이션 프레임워크이다. 스프링 부트는 복잡한 초기 설정을 대신 해주어 프로젝트 초기 설정하는 과정을 줄일 수 있으며, 자체적인 웹 서버를 내장하고 있어, 빠르고 간편하게 배포를 진행할 수 있다는 특징을 가지고 있다.

2.2.4. JPA

JPA는 자바에서 ORM 기술 표준으로 사용하는 인터페이스들의 모음으로 자바 애플리케이션에서 관계형 데이터베이스를 사용하는 방식을 정의한 인터페이스이다. ORM은 객체지향언어인 자바와 관계형 데이터베이스 사이에 객체와 관계형이 일치하지 않기 때문에 이를 맵핑 시켜주는 것이다.

JPA는 특정 데이터베이스에 종속되어있지 않아 어느 데이터베이스를 이용해도 좋다는 장점을 가지고 있다. 하지만 복잡한 쿼리를 처리하는 것이 어렵다는 단점이 있다.

JPA는 entity manager factory가 entity manager를 생성하여 관리하고 entity manager가 entity에 접근하여 해당 entity에 대해 데이터베이스의 작업을 제공하는 방식으로 작동한다. entity를 통해 데이터베이스의 table을 구성하고 repository를 통해 데이터베이스에서 데이터를 검색, 저장 등을 할 수 있다. service와 controller를 통해 entity와 repository로 데이터베이스에서 가져온 데이터를 view단에 전달하거나 view단에서 데이터를 받아온다. 이 때, service와 controller로 나누어 service는 데이터베이스 model단에 controller는 view단에 닿아있게 만들어 model단과 view단이 분리되게 만들어준다. controller는 html과 같은 view에게 dto를 이용해 데이터를 전달하거나 전달받는다.

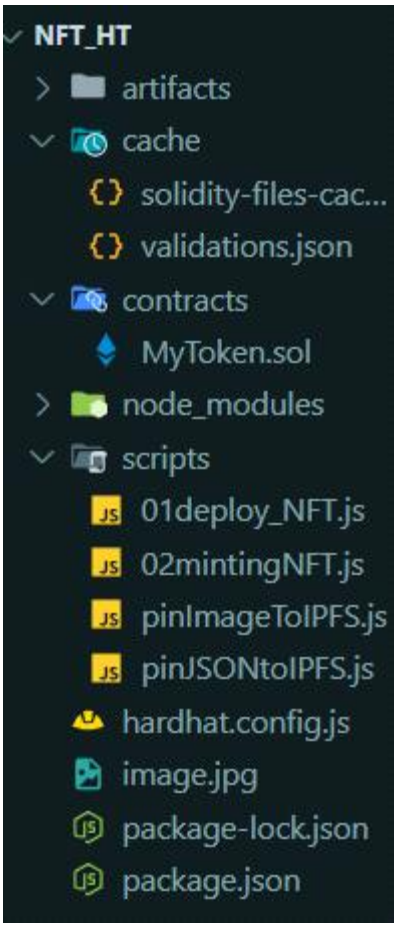
이러한 방식을 MVC 패턴이라 하며 스프링 부트를 이용해 웹 프로젝트를 만들 때에 사용하기 적합한 패턴이다.

3. 프로젝트 설계 및 개발

3.1 프로젝트 타임라인

3.2 NFT 관련 설계 및 개발

개발관련 자료 학습단계에서 학습한 스마트 컨트랙트 환경 구축을 hardhat 모듈을 통해 로컬 단에서 진행하고, 웹 서비스에 필요한 기능들을 구현해보고자 하였다.



해당 프로젝트 내의 hardhat.config.js 파일을 통해 스마트 컨트랙트에 사용될 테스트넷인 SEPOLIA 프라이빗 키값과 같은 정보를 관리하고, Mytoken.sol을 통해 스마트 컨트랙트 환경에서 제공할 함수들을 작성하였다. 그 다음 01deploy_NFT.js를 통해 스마트 컨트랙트를 배포하고, 02mintingNFT.js를 사용해 원하는 NFT를 발행하는 구조이다.

웹 서비스에서 필요한 기능으로는 제한된 개수의 NFT 발행, Pinata API 연동을 통한 IFPS 관련 기능, 코드의 재활용성 향상을 위한 컨트랙트 주소, 지갑 주소, NFT 최대 발행 개수 등의 변수 입력 받기 기능이 있었다.

```
contract MyToken is ERC721, ERC721URIStorage, ERC721Burnable, Ownable {
    uint256 public tokenIdCounter;

    constructor(address initialOwner)
        ERC721("MyToken", "MTK")
        Ownable(initialOwner)
    {
        tokenIdCounter = 1;
    }

    function safeMint(address to, string memory uri, uint256 max_token)
        public
        onlyOwner
    {
        require(tokenIdCounter <= max_token, "Maximum tokens minted");
        uint256 tokenId = tokenIdCounter;
        _safeMint(to, tokenId);
        _setTokenURI(tokenId, uri);
        ++tokenIdCounter;
    }
```

우선 개수 제한 기능의 경우 기존의 MyToekn.sol 코드 내에서 uint256 변수 tokenIdCounter를 선언한 후 생성자를 통해 1로 초기화 시켰다. 이후 safeMint 함수가 불리게 되면 최대 개수 제한 값인 매개변수 max_token과 값을 비교하게 되고 조건문을 충족하여 safeMint가 진행되면 증감함수를 통해 tokenIdCounter의 값을 증가시키는 구조로 설계하였다.

해당 구현에서는 스마트 컨트랙트 환경이 생성되었을 때부터 한 개의tokenIdCounter가 계속해서 증가하는 구조이기 때문에 safeMint 도중에 다른 개수 제한을 가진 safeMint 요청을 하는 것이 어렵다. 따라서 웹 서비스와의 연동 시에는 이미 발급된 NFT의 개수를 새어 이를 최대 개수값과 비교하는 구조를 구현해야 할 것으로 보인다.

두번째 Pianta API 연동을 통한 IFPS 관련 기능으로는 이미지 파일의 IFPS등록, JSON 코드의 IFPS등록을 구현하고자 하였다. 이를 위해 Pinata SDK 모듈을 설치하였고 가장 상단의 사진과 같이 각각 pinImageToIPFS.js, pinJSONtoIPFS.js로 구현하였다.

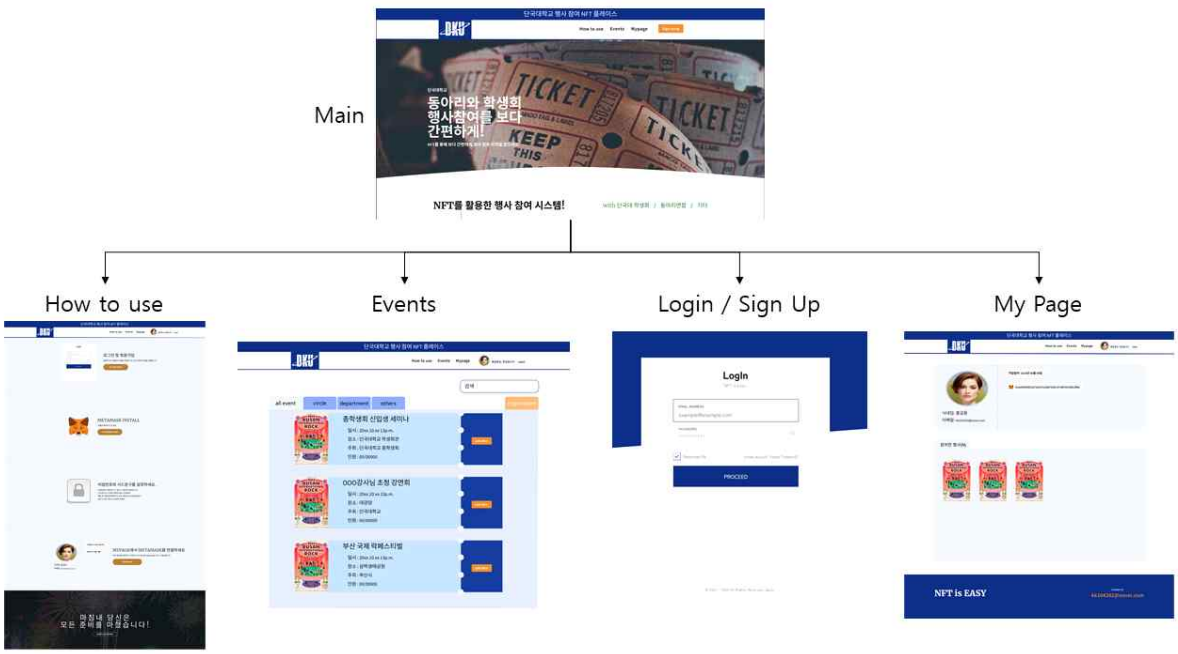
```
const gateway = 'https://'+ readlineSync.question('type GateWay: ') +'/ipfs/';

// Connect to the deployed contract
// const contractAddress = "0x4236178C3CCB8F4c8A020E2564fD4A006159a937"; // Replac
const contractAddress = readlineSync.question('type contAddr: ');
const userWalletAddress = readlineSync.question('type User\'s WalletAddr: ');
const max_token = readlineSync.questionInt('type Max_token: ');
const uri = gateway + readlineSync.question('type CID: ');
console.log('your uri is \''+uri+'\'');

const contract = await SimpleContract.attach(contractAddress);
```

마지막으로 컨트랙트 주소값, 사용자의 가상화폐 지갑 주소, 최대 토큰 값, IFPS의 해쉬 값(CID) 등의 정보를 입력 받을 수 있도록 02mintingNFT.js 코드 내에 readline-sync 모듈을 사용해 코드를 작성하였다. 해당 코드를 통해서 02mintingNFT.js 파일을 실행시킬 때 마다 원하는 데이터들을 입력할 수 있어 코드의 재활용성이 향상되었다.

이후 웹 서비스와의 연동 시에는 직접적인 문자열 입력이 아닌 웹 서비스 시스템에서 전송한 데이터를 변수에 받는 형태로 수정될 예정이다.

3.3 Web 설계 및 개발

3.3.1. Web 서비스 프로토타입



해당 웹 서비스의 프로토타입을 구상하였을 때 Main 페이지의 상단 메뉴에서 사용 안내 페이지(How to use).

로그인 및 회원가입 페이지(Login/Sgin Up), 행사 페이지(Events), 마이 페이지 (My Page)의 4개의 페이지에 접근 할 수 있도록 설계하였다.

3.3.2. 요구사항 정리

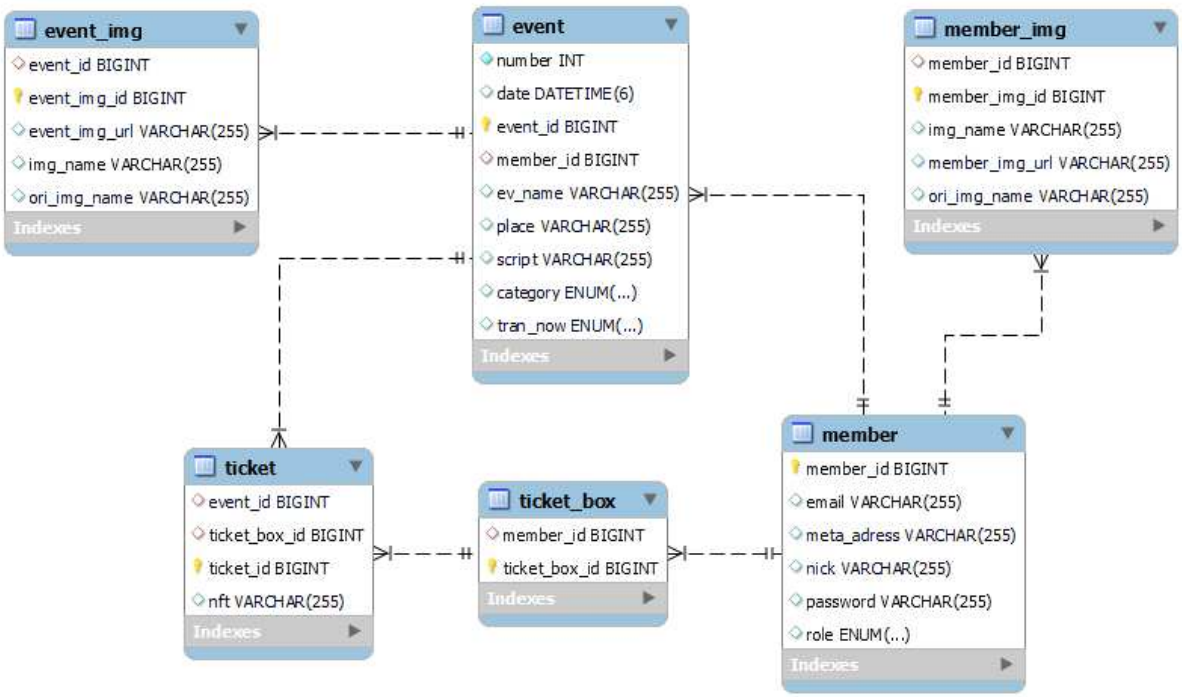
해당 웹 서비스는 사용하는 유저의 Role을 User(일반 사용자), Owner(행사 등록자), Admin(사이트 관리인)으로 나누었으며, 각각의 Role이 요구하는 기능을 아래와 같이 각각 표로 정리하였다.

<User>	1 Depth	2 Depth	3 Depth	Detail
Main	DKU			메인 화면으로 이동 (사이트 안내, 최근 등록된 축제 중 5개 간략 노출)
	로그인			이메일과 비밀번호 입력
		회원가입		일반 사용자, 행사 등록자 Role 선택 가능
	How to use			사이트 사용법 안내
		MetaMask 이동		MetaMask 확장 프로그램 설치를 위한 사이트로 이동
	Events			SafeMint가 가능한 행사 리스트 노출 (한 페이지에 5개씩 노출)
		카테고리 선택		All event / Circle / Department / Others 중에 선택 가능
		검색 기능		원하는 카테고리 내에서 행사명 or 등록자를 검색 가능
		SafeMint		Admin에게 해당 행사에 대한 NFT 티켓 SafeMint 및 전송을 요청
	MyPage			MyPage로 이동
		MetaMask 연결		User의 MetaMask 지갑 주소 저장
			참여한 행사 조회	Admin로부터 발급받은 행사 티켓 목록 확인
		프로필 변경		프로필 사진 변경
	로그아웃			

<Owner>	1 Depth	2 Depth	3 Depth	Detail
Main	Events			
		Regist Event		행사의 이름, 이미지, 카테고리, 인원수, 날짜 등의 정보를 기입하여 등록
	MyPage			
		등록 행사 조회		Regist Event를 통해 등록 요청된 행사가 Admin에 의해 노출되도록 전환되었는지, 몇 명의 User가 SafeMint 하였는지 확인

<Admin>	1 Depth	2 Depth	3 Depth	Detail
Main	MyPage			
		등록 요청된 행사 목록 확인		Owner가 등록한 요청 행사 목록 확인
			행사 등록 요청 수락	요청된 행사를 Events 메뉴에 노출되도록 전환
		SafeMint 요청 확인		User가 신청한 SafeMint 요청 목록 확인
			SafeMint 후 NFT 티켓 발급	User와 행사 정보가 담긴 IPFS로 저장한 뒤 이를 통해 NFT를 발급하고 DB의 Ticket table에 관련 정보 저장

3.3.3. DB 구조 설계



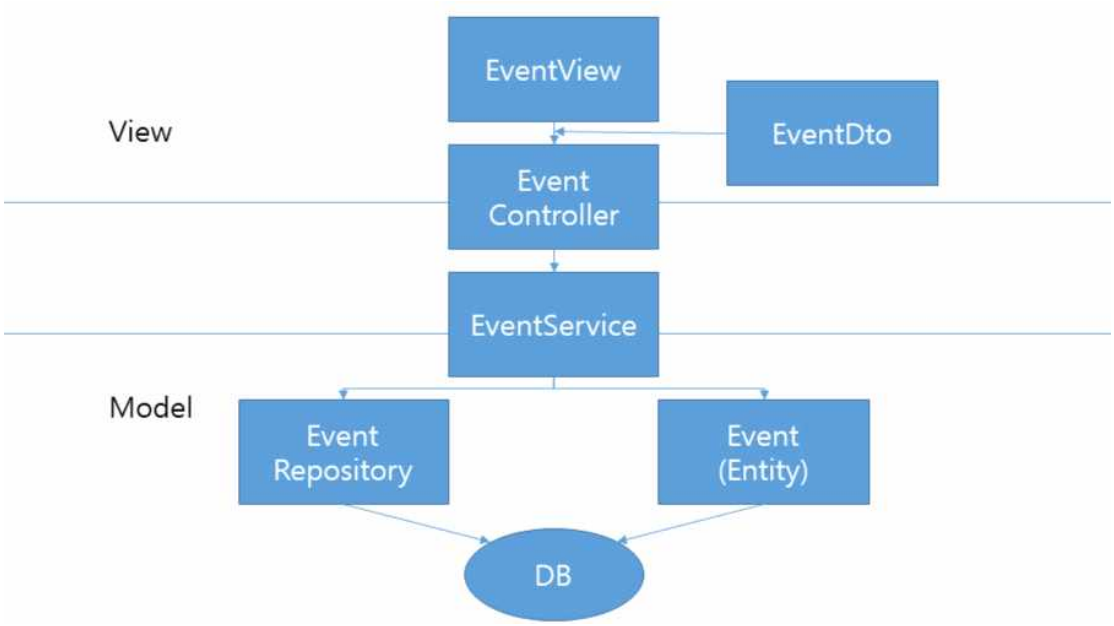
위의 항목에서 서술한 요구사항 정리를 바탕으로 사용될 DB의 구조를 설계하였다.

대표적으로 member, event 테이블이 있으며 이는 각각 회원과 행사 정보를 관리하는데 member에는 회원의 Role 정보가, event 에는 행사의 카테고리와 Admin의 등록 요청 수락 여부가 ENUM의 형태로 저장된다. 또한 Owner 유저는 여러 개의 행사를 등록할 수 있음으로 member와 event는 1대다 관계로 매핑하였다.

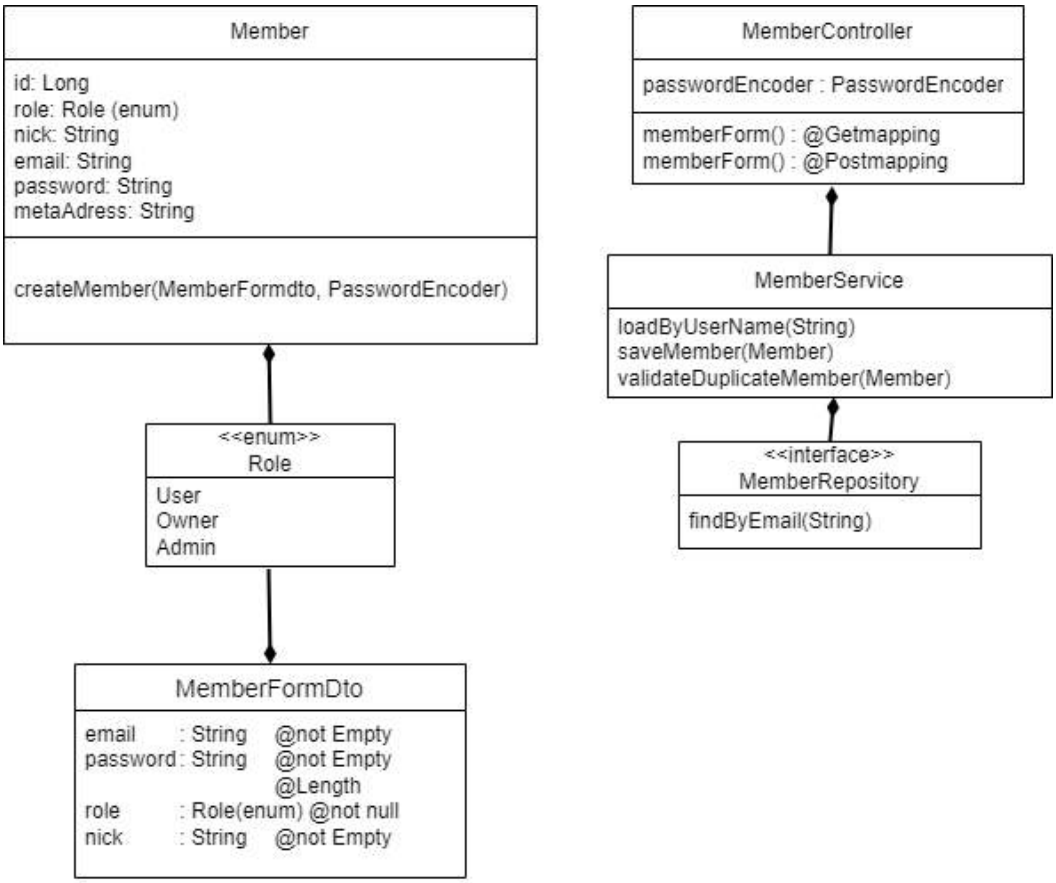
각각의 회원과 행사는 한 개의 이미지를 가질 수 있는데 이는 각각 member_img, event_img 테이블과의 1대1 매핑으로 구현되었다.

또한 한 개의 이벤트로부터 여러 개의 ticket이 발급될 수 있음으로 event와 발급된 티켓의 정보가 담긴 ticket 이벤트가 1대다 관계로 매핑되며, 한 명의 사용자는 하나의 ticket_box 테이블과 1대1 매핑이 되고 이 ticket_box에 유저가 가진 ticket들이 1대다 매핑이 되어 유저가 가진 ticket들을 관리할 수 있게 된다.

3.3.4. 시스템 구조 및 기능 설계



해당 웹 프로젝트에서는 JPA를 활용한 MVC패턴을 사용하였다.
이를 통해 JAVA 클래스 파일의 형태인 entity를 통해 table의 형태를 정의하고 repository를 통해 DB의 데이터 검색 및 저장을 할 수 있도록 구현하였다.



해당 패턴의 기능 구조를 위해 회원가입 기능을 예시로 들자면, Controller 클래스에서 Get Mapping 된 memberForm 메소드가 호출되면 회원가입에 필요한 정보를 기입할 MemberFormDto 객체를 생성해 View가 사용할 수 있도록 한다.

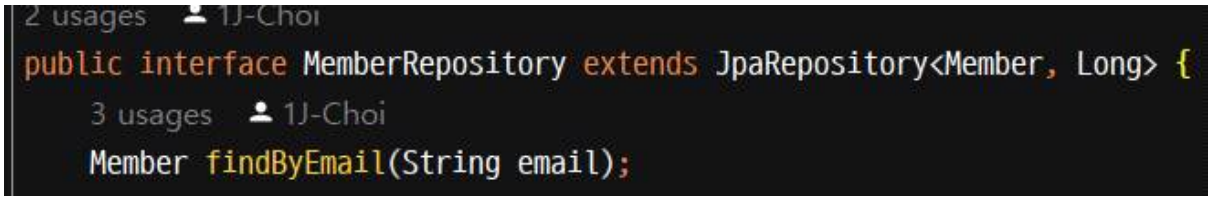
이후 Post Mapping된 memberForm 메소드가 불리면 memberFormDto에 입력된 정보를 바탕으로 주입되어 있는 memberService에서 중복 검사 메소드를 거치게 되며 이에 성공하면 서비스 안에 주입된 memberRepository를 통해 해당 회원의 데이터를 저장하게 된다.

```java
@GetMapping(value = "/new")
public String memberForm(Model model){
    model.addAttribute( attributeName: "memberFormDto", new MemberFormDto());
    return "member/memberForm";
}

1J-Choi
@PostMapping(value = "/new")
public String memberForm(@Valid MemberFormDto memberFormDto, BindingResult bindingResult, Model model) {
    // @Valid를 통해 Dto에서 @를 통해 설정한 조건들 적용
    if (bindingResult.hasErrors()) { // 각 변수명 체크 문제가 생기면 호출 EX) 이름 X 이메일 형식 X
        return "member/memberForm";
    }
    try {
        Member member = Member.createMember(memberFormDto, passwordEncoder);
        memberService.saveMember(member);
    } catch (IllegalStateException e) {
        model.addAttribute( attributeName: "errorMessage", e.getMessage()); // memberForm.html의 경고창으로 메세지가 보내진다
        return "member/memberForm";
    }
    return "redirect:/";
}
```

조금 더 구체적으로 살펴보자면, memberController 내에서는 위 사진과 같이 각각 GET, POST 매핑이 된 메소드들이 존재하며 Post 매핑이 된 메소드에서 MemberService의 saveMember 메소드를 통한 저장이 이루어지는 것을 확인할 수 있다.

```java
2 usages  1J-Choi
public Member saveMember(Member member) {
    validateDuplicateMember(member);
    return memberRepository.save(member);
}

1 usage  1J-Choi
private void validateDuplicateMember(Member member) {
    Member findMember = memberRepository.findByEmail(member.getEmail());
    if (findMember != null) {
        throw new IllegalStateException("이미 가입 된 회원입니다.");
    }
}
```

그 다음으로 MemberService 내에서는 주입된 MemberRepository를 통해 입력된 데이터가 중복되는지를 체크하고 이를 저장하는 과정을 거치게 된다.



마지막 MemberRepository에서는 중복 방지 기능에 사용된 email을 통해 해당 데이터를 가진 Member entity 객체를 찾는 findByEmail 메소드를 확인 할 수 있다.

MemberRepository는 인터페이스의 형태이며 JpaRepository를 상속받고 있는데 이를 통해 사진과 같이 규칙에 따라 메소드 명을 작성하면 따로 오버라이딩 하지 않더라도 기능을 하며 JpaRepository 내에 있는 메소드의 경우 추상 메소드를 작성할 필요 없이 바로 사용할 수 있다.

복잡한 쿼리를 요구하는 행사 검색 기능의 경우 QueryDsl을 통해 구현하였지만 그 외의 이미지 저장, 행사 등록 등의 기능들에는 JPA를 사용한 방식이 적용되었다.

3.4. Smart contract와 Web 연동

3.2과 3.3에서 구현한 스마트 컨트랙트와 웹 서비스를 통합하여 NFT 민팅 기능을 제공하는 웹 서비스를 만드록자 하였다.
이때 스마트 컨트랙트 주소를 받아오는 deploy 과정은 기존의 코드를 사용하여 진행하고 컨트랙트 주소의 값을 받아와 Spring boot 시스템에서 사용하여 minting 하는 과정을 JAVA 코드로서 구현하고자 하였다.

하지만 3.2의 스마트 컨트랙트는 Node.js 프레임워크를, 웹 서비스는 Spring boot 프레임워크를 통해 구현되었기 때문에 하나로 통합시키는 것에는 어려움이 있었다. 때문에 이를 해결하기 위해 크게 두가지의 방법을 시도해 보았는데, 첫번째는 기존의 solidity 파일을 JAVA Class 파일로 변환시켜 Spring boot 환경 내에 넣는 방법, 두번째는 Spring boot 프로젝트와 Node.js 프로젝트 사이에서 통신을 통해 기능을 구현하는 방법이었다.

전자의 solidity 파일을 ABI와 BIN으로 추출한 뒤 Spring boot에서 의존성으로 가져온 모듈인 Web3j를 통해 JAVA 파일로 변환하는 방식이였으나, 추출 까지만 진행이 되고 Web3j를 통한 JAVA 파일로의 전환이 정상적으로 이루어지지 않았다.

이에 전환하고자 하는 Solidity 파일에 문제가 있을 수 있다 판단하여, JAVA로 전환되어 있는 임의의 JAVA 클래스 파일을 프로젝트에 삽입해 기능하는지 확인하려 하였으나, 해당 파일이 상속받는 클래스의 메소드 매개변수가 다르게 적용되어있는 문제로 인해 정상 작동하지 않았다.
이는 작성된 파일이 사용한 Web3j 모듈의 차이로 예상되었다.

때문에 후자의 각각의 프로젝트 사이에서 통신하는 방식을 선택하게 되었다. 이 경우 기존의 스마트 콘트랙트 환경에서 크게 수정하거나 변형 해야하는 부분이 줄어들고 각각의 시스템 사이에서 통신하는 기능만 구현하면 되었기에 효율적인 코드 구현이 가능하다고 판단하였다.

4 프로젝트 테스트 및 결과

4.1 NFT관련 테스트 및 결과

4.1.1 Smart Contract 관련 테스트 및 결과

```
PS C:\Users\GSW\NFT> npx hardhat run scripts/mintingNFT.js --network sepolia
Token Symbol: MTK
Token Name: MyToken
Token Contract Owner: 0x1a095045D0819dB64C8Ef4E8D6a3893e0b03a4b2
Token Minted Succesfully
```

[주어진 Solidity파일을 활용한 Deploy, Mint기능만 가능한 최초 테스트]

```
C:\Users\최원제\NFT_HT>npx hardhat run scripts/02mintingNFT.js --network sepolia
Token Symbol: MTK
Token Name: MyToken
Token Contract Owner: 0xAC09D9C896bb4dFB72954150b2d5bB3e7D6Aef27
Token Minted Succesfully
```

```
C:\Users\최원제\NFT_HT>npx hardhat run scripts/02mintingNFT.js --network sepolia
Token Symbol: MTK
Token Name: MyToken
Token Contract Owner: 0xAC09D9C896bb4dFB72954150b2d5bB3e7D6Aef27
ProviderError: execution reverted: Maximum tokens minted
```

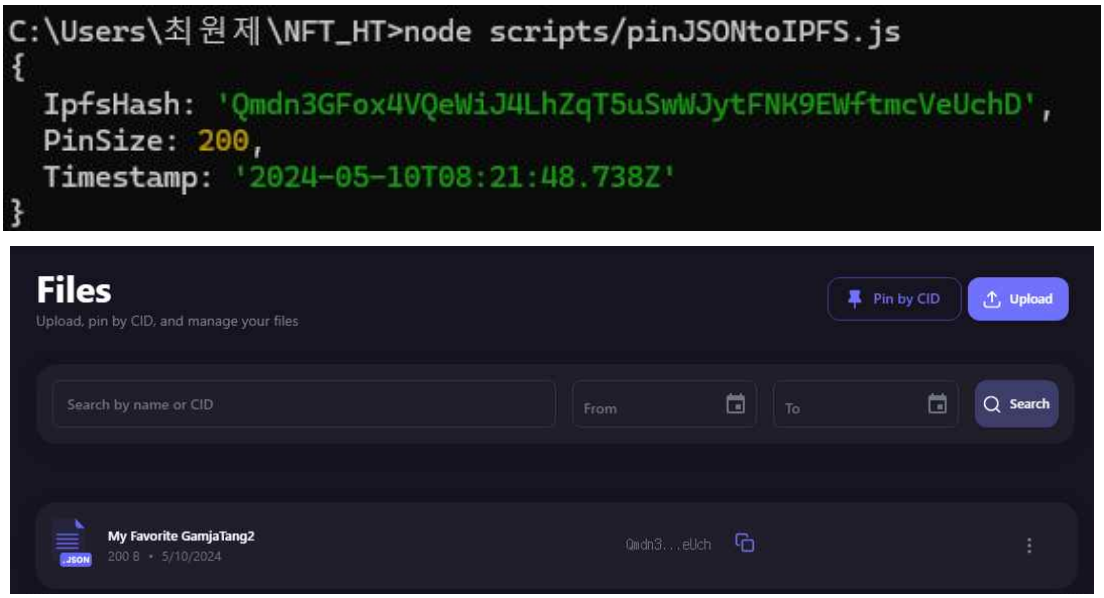
[행사 참여인원 수 만큼의 mint가 발생했을 때 추가mint가 이루어지지 않아야 한다는 과제 해결 이후 테스트]

```
PS C:\Users\GSW\NFT> npx hardhat run scripts/mintingNFT.js --network sepolia
Contract Address : 0xebaaf779953b393F8f497911f154B7a4323610d9
Matamask Address : 0x1a095045D0819dB64C8Ef4E8D6a3893e0b03a4b2
Max Token : 3
Token Symbol: MTK
Token Name: MyToken
Token Contract Owner: 0x1a095045D0819dB64C8Ef4E8D6a3893e0b03a4b2
Token Minted Succesfully
```

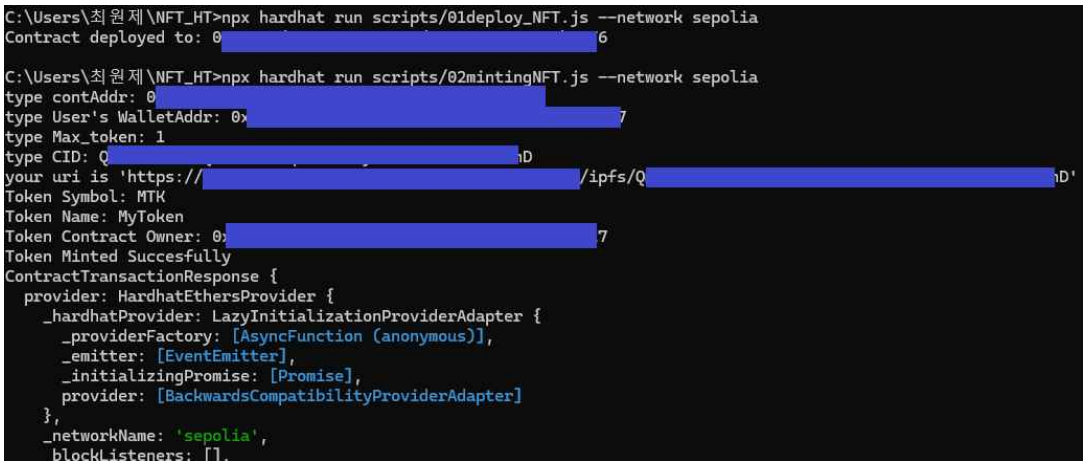
[Wallet Address, Contract Address, Max_token 값을 입력받아 유연성과 호환성을 개선한 이후 테스트]

4.1.2 PINATA(IPFS) 테스트 및 결과

[PINATA API를 활용하여 일반 IMG파일을 IPFS화한 테스트]



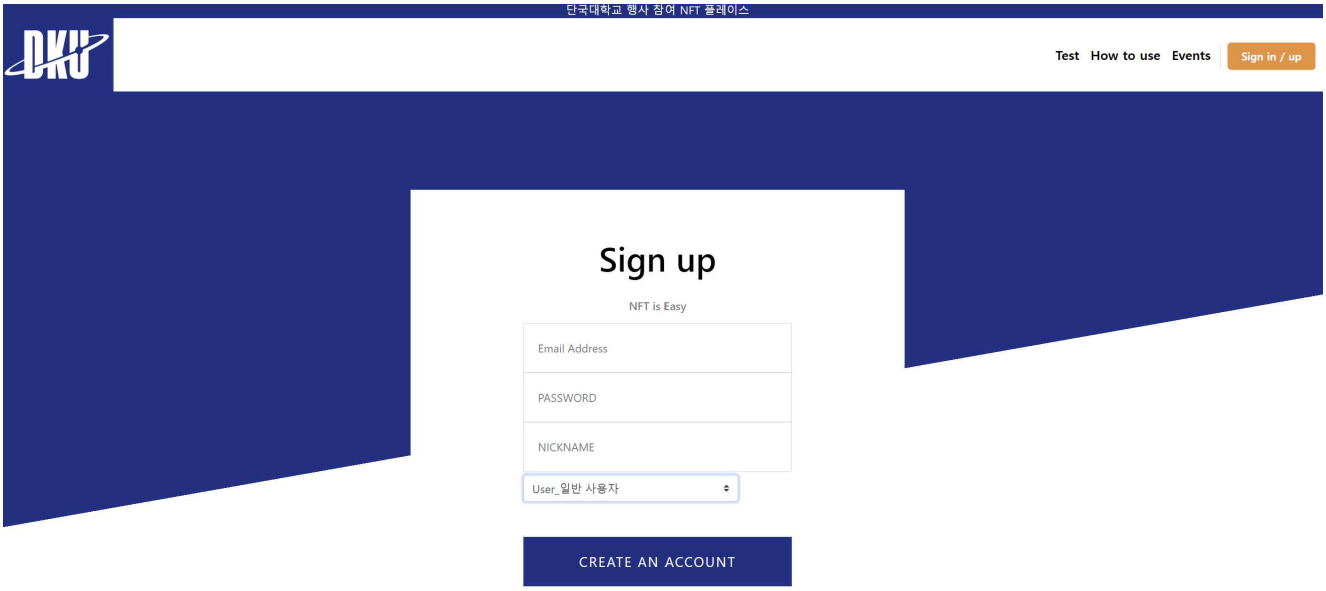
[PINATA API를 활용하여 JSON파일을 IPFS화한 테스트]



[PINATA API를 활용하여 IPFS uri를 불러와 safemint를 진행한 테스트]

4.2 Web 테스트 및 결과

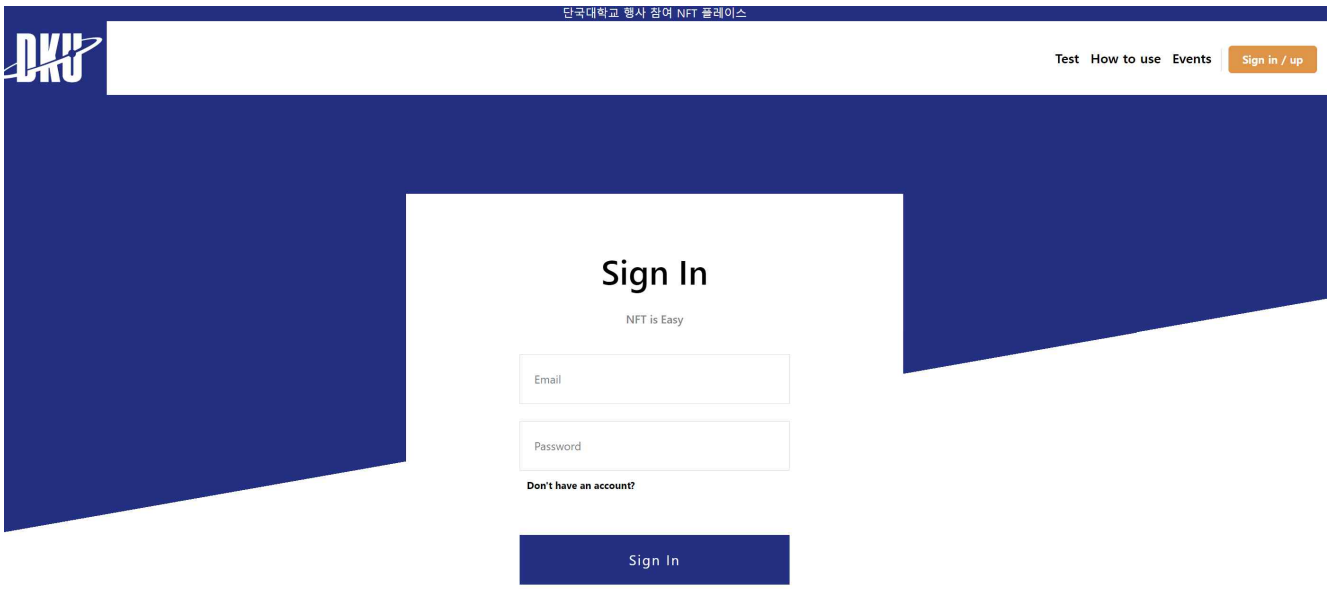
4.2.1 회원가입, 로그인 페이지



[회원가입 페이지]

	member_id	email	meta_adress	nick	password	role
▶	1	test@t.t	NULL	test	$2a$10$tG0xMYchhy0LLNGjSRuKTeY1QHEHRY...	USER
*	NULL	NULL	NULL	NULL	NULL	NULL

[회원가입 인적사항이 Database에 저장된 모습]

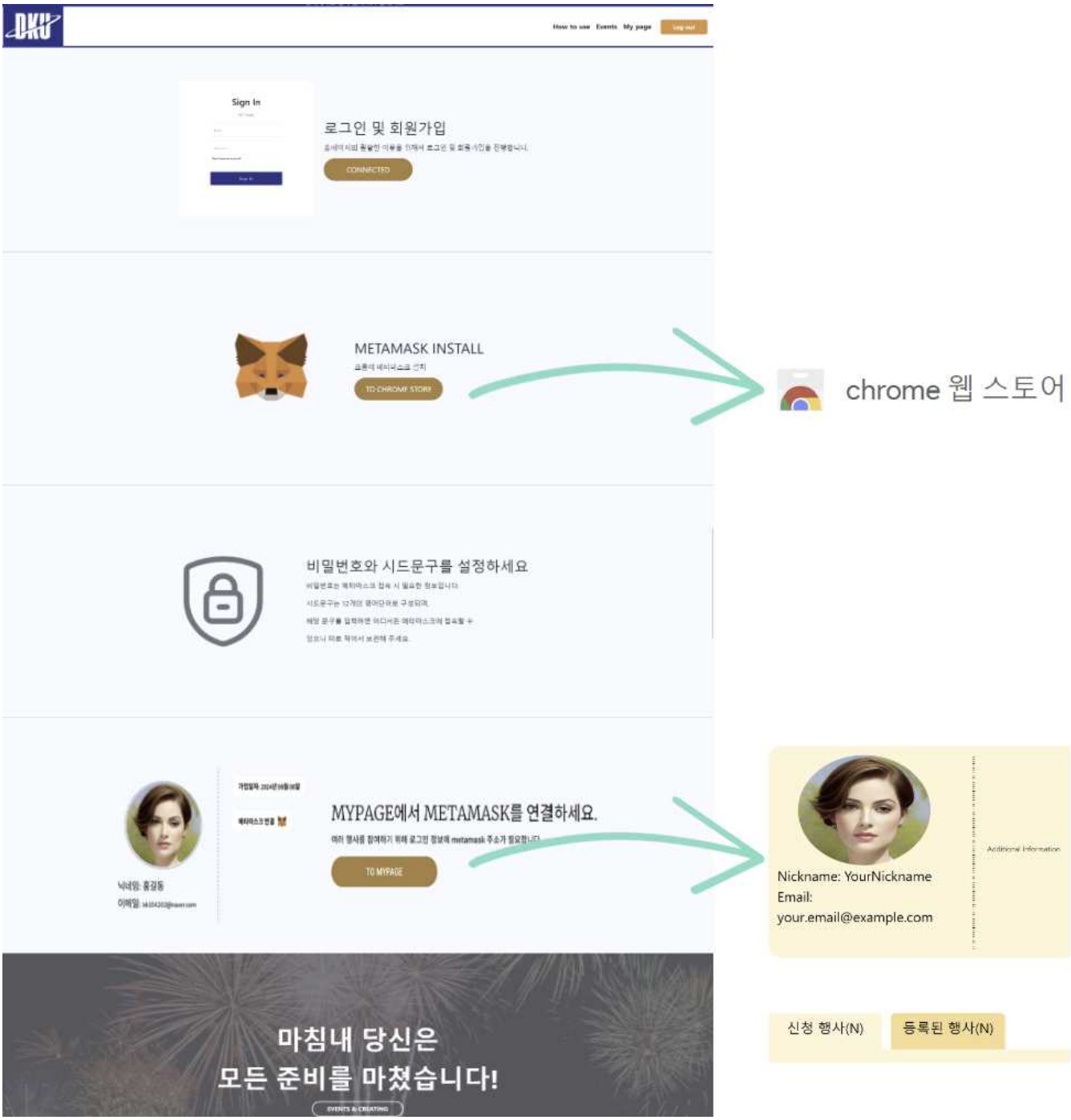


[로그인 페이지]

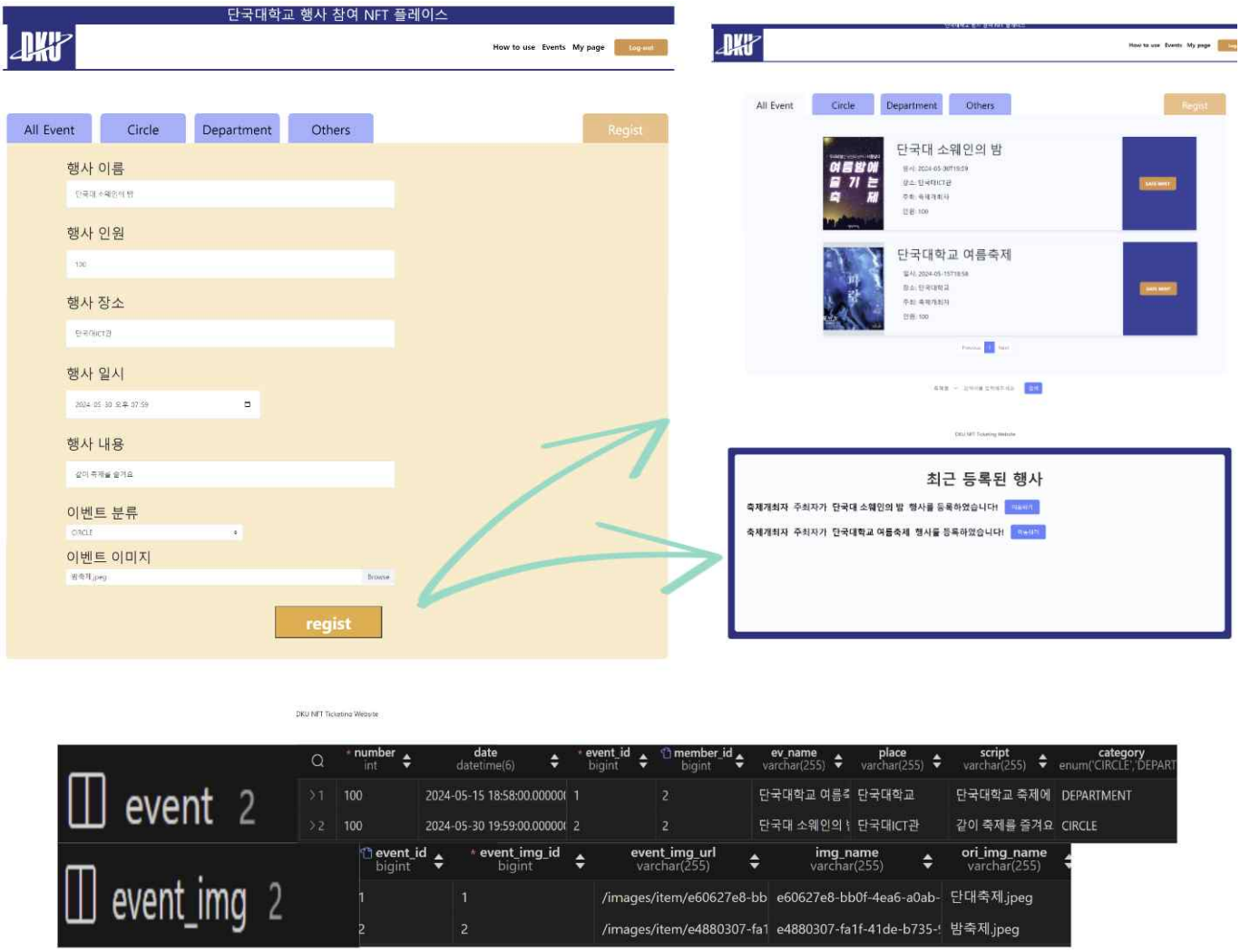
4.2.2 메인페이지


How to use
Events
My page
동아리와 학생회
행사참여를 보다
간편하게!
NFT를 활용한 행사 참여 시스템!
with 단국대 학생회 / 동아리연합 / 기타
TICKET
"NFT Ticket"
새로운 예약의 시작
간편한 3단계!
Step1. 행사 서칭
Events탭에서
원하는 행사를 찾아보세요!
Step2. NFT Mint
원하는 행사의
티켓(NFT)를 Mint하세요!
Step3. 보여주면 끝
지갑 속 NFT를
보여주기만 하면 끝!
최근 등록된 행사
축제개최자 주최자가 단국대 소웨인의 밤 행사를 등록하였습니다!
축제개최자 주최자가 단국대학교 여름축제 행사를 등록하였습니다!

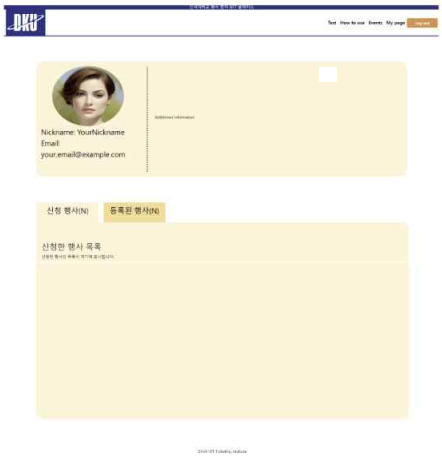
4.2.3 How to use 페이지



4.2.4 행사 리스트 및 등록 페이지

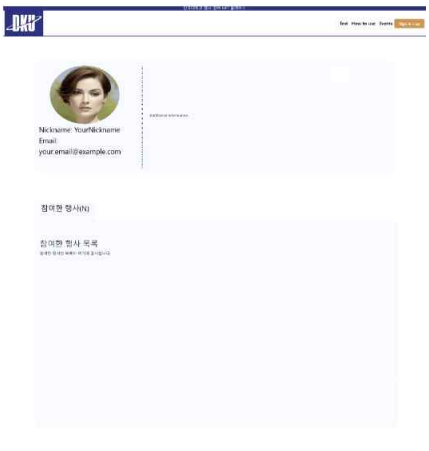


4.2.5 마이 페이지



[Owner]

[Admin]



[User]

5. 프로젝트 소감

이름	내용
김창오	캡스톤 프로젝트를 진행하면서 가장 고민됐던 문제는 어떠한 주제로 프로젝트를 진행할지 결정하는 것이었다. 좋은 기회로 NFT와 Smart Contract에 대해 공부할 수 있었고, 이를 통해 NFT를 활용한 티켓팅 시스템 제공 서비스를 만들자고 의견이 모여졌다. 매년 이맘때쯤 여러 대학교가 축제를 개최한다. 동시에 축제관람에 대한 불공평성과 암표논란이 일어난다. 이번년도 역시 대학교 축제를 관람하기 위해 여러 2차거래 사이트에서 암표거래가 이루어지는 것을 보면서 우리가 선택한 주제가 사회적으로 큰 비용과 피로감을 개선시킬 수 있다고 생각하여 뿌듯했다. 개발 과정에 있어서는 1. NFT와 Smart Contract에 대한 기본적인이해 2. Smart Contract를 개발하기 위한 Remix IDE와 Hardhat의 활용 3. 웹서비스를 제공하기 위한 프론트•백단의 개발이 있었다. 일반적인 캡스톤보다 필요한 개념과 사용해여할 도구가 많아 시간적으로 부족하였지만 3가지 요소 모두 잘 마무리하여 Smart Contract를 웹에 연동하는 과정만 마무리된다면 프로젝트가 유의미한 결과를 보여줄 것이라고 생각한다. 팀원 모두 프로젝트와 블록체인을 활용한 이번 프로젝트 컨셉에 흥미를 느끼고 만족하기에, 이 프로젝트를 학기로 종결짓지 않고 학기 이후에도 기간을 더 잡고 구체화할 예정이다. 사회적으로 문제가 되는 이슈를 개발자로서 풀어나가는 과정이 너무 뿌듯하고 즐거웠다. 특히 팀원과 소통하며 하나씩 알아가고 서로 도와주며 최선을 다하는 모습이 인상깊었다. 앞으로도 사회에 도움이 되는 프로그램과 서비스를 제공하는 개발자가 되고 싶다.
최원제	Spring boot를 사용한 웹 서비스 프로젝트를 진행한 경험은 있었지만 NFT 기술 자체에는 문외한 이였기 때문에 이에 대한 학습을 진행하는데 큰 부담이 있었다. 또한 웹 개발 경험이 없는 팀원들에게 프로젝트 협업 방식과 사용될 기술에 대해 논의하고 안내하는 중에도 부담감이 있었다. 하지만 이를 통해서 NFT에 대한 지식을 보다 직접적으로 얻을 수 있었고, 이를 NFT 기술을 적용 시켜 프로젝트를 진행할 때 능동적으로 활용할 수 있었다. 또한 이미 알고 있는 웹 개발 기술이나 방식에 대해서도 다른 팀원에게 설명하기 위해 용어나 개념, 원리를 확실히 정리하고 문서화 하는 과정을 거치면서, 스스로도 공부가 많이 되었고 팀으로서 협업하는 방식에 대해서도 배운 것 같았다. 학기가 끝난 이후에도 해당 프로젝트를 마무리 짓고 개인적으로 NFT와 관련된 프로젝트를 진행하고자 한다.

구선우	처음으로 진행해본 프로젝트라 과정 하나하나가 새롭고 신선한 느낌이 들었다. 또, 이름은 들어봤지만 잘 모르던 NFT에 대한 것을 공부하면서 이런 기술도 있구나 하는 것을 알게 되어 좋았고, 이를 이용해 프로젝트를 만들어보니까 재미도 있었던 것 같다. 프로젝트를 처음 해보는데다가 생소했던 NFT를 접목시켜 프로젝트를 만들어 가니 어려움과 시행착오들이 꽤 있었지만 이러한 과정으로 한 층 더 성장할 수 있게 되었다고 생각한다. 우선 NFT 쪽으로는 전혀 지식이 없었는데 이번 학기 동안 이에 대해 공부하면서 NFT가 활용될 수 있는 여러 분야에 대해 알게 되었고 그 과정이 매우 흥미로웠다. 앞으로 NFT에 대한 소식을 들으면 이 이해도를 바탕으로 관심있게 볼 수 있을 것 같다. 프로젝트 쪽으로는 이번 프로젝트를 통해 어떤 식으로 프로젝트가 구성되고 흘러가는지 알게 되었다. 예를 들자면, 처음에는 프론트 엔드와 백 엔드가 완전히 분리되어 마지막에 결합하는 식으로 진행되는줄 알았으나 서로 유기적으로 연결되어 서로가 잘 완성되어있어야 프로그램이 돌아간다는 것을 알게 되었다. 아마 시간적인 문제로 이번 학기 안에 프로젝트를 완성시키기는 어려울 것 같지만 방학기간을 이용해 끝까지 완성해보고 싶다. 한 학기 동안에 여러가지로 의미 있는 프로젝트였던 것 같다.
강은솔	NFT라는 기술을 주변에서 들어보기만 했고 직접 사용해 본 경험은 따로 없었기에 걱정되면서도 흥미로운 주제였다. NFT라는 기술이 어떤 방식으로 이루어 지는지, 어떻게 사용이 되고 어떤 이점이 있는지 등 대략적으로만 알고 있던 상태였기에 직접 스마트 컨트랙트를 구현하는 데에 조금 부담감과 어려움이 있었다. 하지만 팀과 협동을 하고 회의를 진행하면서, 또 그리고 교수님과 대학원생님께 궁금한 부분, 막히는 부분에 대한 질문을 이어가며 문제를 해결 해 갔다. 이후 알려주신 것을 기반으로 직접 스마트 컨트랙트를 구현해 보고, 세부적인 기능을 추가하였다. NFT와 관련하여 서비스 할 수 있는 것이 무엇이 있을까 고민하다 티켓팅이라는 키워드를 떠올리게 되었고, 경험이 있으신 팀원분과 팀장님의 주도로 웹 개발을 진행하였다. 스마트 컨트랙트와 웹 개발, 이렇게 투 트랙으로 개발이 진행되었기에 중간중간 어려움이 있었지만, 회의를 진행하며 대부분의 문제를 해결하였다. 학기중에 할 수 있는 개발과 구현을 진행하고, 이후에도 프로젝트 완성을 위해 팀원들과 협업을 진행 해 볼 예정이다. 이런 실무적인 프로젝트는 처음이었기에 어려움이 많았지만 그만큼 해결 방안을 알게 되고, 어떤 방식으로 개발이 진행되는지, 협업을 진행할때 효율적으로 진행하는 방식 등 많은 실무적 경험을 얻을 수 있었다.