블록체인 기반 안전 거래 플랫폼

팀 명 : 개발의 민족

지 도 교 수 : 이병천 교수님

팀 장 : 이규민

팀 원 : 김준현

성우상

이승훈

전준영

최병준

2023. 11.

중부대학교 정보보호학과

목 차

1. 서론

1.1 연구 배경

기존의 중고거래, 거래 플랫폼은 비대면 거래의 선호도가 증가하면서 사기의 위험과 부정적인 방법으로 활용하는 사례들이 있다. 이런 사례들을 방지하고자 블록체인 도입한 중개자 없는 안전 거래 플랫폼을 구현하고자 해당 주제로 선정했다. 중고거래 플랫폼, 콘서트 티켓팅 등에 블록체인 생태계를 도입하면 거래 프로세스를 자동화로 인한 능률이 올라가며 오류나 사기를 방지할 수 있어서 이전의 악용, 부정적 사례를 방지할 수 있다.

1.2 연구 목적 및 필요성

블록체인이 기반이 되는 플랫폼은 부동산 거래 시스템, 의료 정보, 디지털 인증, 등 많은 응용 사례들이 있다. 그 중, 전통적인 중고거래에서 생길 수 있는 한계점을 블록체인을 도입하는 방안으로 극복하고자 했다. 해당 기술을 조사하면서 블록체인 생태계 자체에 대한 이해를 높이고 플랫폼에 적용하면서 블록체인과 web3 기술의 상호 작용과 적용 방법 연구하였다.

기존의 중고거래와 다르게 블록체인을 도입하면서 거래를 진행할 때 임의로 데이터를 변경할 수 없는 블록체인 생태계에서 거래할 수 있게 하여 신뢰성 있고 투명한 거래를 가능하게 한다. 그러나 블록체인의 블록을 추가하는데 시간이 많이 소요되기 때문에 중고 거래 플랫폼의 모든 데이터를 블록체인에 저장한다면, 사용자가 증가하면서 블록에 담을 수 있는 데이터의 한도를 초과할 우려가 있다. 이와 같은 이유로 블록체인에 저장되는 데이터는 거래 내역과 같이 수정이 필요 없고 데이터 특성상 오랜 기간 동안 보관이 필요한 데이터를 저장하고 변동성이 큰 사용자 간 채팅과 같은 데이터는 데이터베이스 서버를 통해 저장하도록 고안하였다.

거래 과정에서 해당 내역은 블록체인에 저장되고 구매자가 물품을 확인하면 구매할 때 사용한 토큰을 판매자에게 입금되면서 판매자는 판매 수익을 토큰으로 받을 수 있다. 이러한 방식으로 거래하면서 기존 중고거래에서의 입금을 유도하고 제품을 보내지 않는 방식의 사기에 대한 위험도는 줄어들고 거래 내역이 저장된다는 점에서 투명한 거래를 할 수 있게 된다.

2. 관련 연구

2.1 Node JS

Node JS는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼으로, Chrome V8 JavaScript 엔진으로 빌드된 JacaScript 런타임 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 논블로킹(Non-blocking) I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다. 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능케 한다. 블록 체인에 저장하는 거래 내역과 같은 데이터 이외의 채팅, 부가적인 사용자 정보 등을 Node JS의 프레임워크인 Express를 사용하여 만든 서버를 통해 저장하고 사용한다.

2.2 React

리액트(React, React.js 또는 ReactJS)는 자바스크립트 라이브러리의 하나로서 사용자 인터페이스를 만드는 데 사용된다. 페이스북과 개별 개발자 및 기업들 공동체에 의해 유지보수된다. React는 싱글 페이지 애플리케이션이나 모바일 애플리케이션 개발에 사용될 수 있다. 대규모 또는 복잡한 React 애플리케이션 개발에는 보통 라우팅, API 통신 등의 기능이 요구되는데 React에는 기본적으로 제공되지 않기 때문에 추가 라이브러리를 사용해야 한다.

2.3 JavaScript

자바스크립트(영어: JavaScript)는 객체 기반의 스크립트 프로그래밍 언어이다. 이 언어는 웹 브라우저 내에서 주로 사용되며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능이 있다. 또한 Node.js와 같은 런타임 환경과 같이 서버 프로그래밍에도 사용되고 있다. 자바스크립트는 본래 넷스케이프 커뮤니케이션즈 코퍼레이션의 브렌던 아이크(Brendan Eich)가 처음에는 모카(Mocha)라는 이름으로, 나중에는 라이브스크립트(LiveScript)라는 이름으로 개발하였으며, 최종적으로 자바스크립트가 되었다. 자바스크립트가 썬 마이크로시스템즈의 자바와 구문이 유사한 점도 있지만, 이는 사실 두 언어 모두 C 언어의 기본 구문에 바탕을 뒀기 때문이고, 자바와 자바스크립트는 직접적인 연관성은 약하다. 이름과 구문 외에는 자바보다 셀프나 스킴과 유사성이 많다. 자바스크립트는 ECMA스크립트(ECMAScript)의 표준 사양을 가장 잘 구현한 언어로 인정받고 있으며 ECMAScript 5(ES5)까지는 대부분의 브라우저에서 기본적으로 지원되었으나 ECMAScript 6 이후부터는 브라우저 호환성을 위해 트랜스파일러로 컴파일된다.

2.4 MySQL

MySQL은 세계에서 가장 많이 쓰이는 오픈 소스의 관계형 데이터베이스 관리 시스템이다. 다중 스레드, 다중 사용자, 구조질의어 형식의 데이터베이스 관리 시스템으로 오라클이 관리 및 지원하고 있으며, Qt처럼 이중 라이선스가 적용된다.

2.5 AWS

Amazon Web Service의 약자이며 클라우드 컴퓨팅과 관련하여 다른 개발자가 본인의 애플리케이션에 사용할 수 있는 기능을 제공하는 플랫폼을 PasS로서 제공한다. 사용자가 원하는 대로 확장이 가능하며 원하는 수의 가상 서버를 구축하고 보안 및 네트워킹을 구성하며 Storage 또한 관리할 수 있다. 확장이나 축소가 가능하기 때문에 트래픽을 예측할 필요성이 줄어든다는 장점이 있다. 해당 프로젝트에서 Node Js 기반의 express 웹 애플리케이션 서버를 구동시킬 Amazon Elastic Compute Cloud(EC2= 서비스와 HTTPS 통신을 하며 외부에서 들어오는 트래픽을 분산해 주는 애플리케이션 로드 밸런서(Application Load Balencer, ALB), DNS를 제공해 주는 Route53, 이미지 파일을 저장할 정적 스토리지를 제공해 주는 Amazon Simple Storage Service(S3) 등을 제공받았다.

2.6 블록체인

블록체인(Blockchain)은 관리 대상 데이터를 '블록'이라고 하는 소규모 데이터들이 P2P 방식을 기반으로 생성된 체인 형태의 연결고리 기반 분산 데이터 저장 환경에 저장하여 누구라도 임의로 수정할 수 없고 누구나 변경의 결과를 열람할 수 있는 분산 컴퓨팅 기술 기반의 원장 관리 기술이다. 이는 근본적으로 분산 데이터 저장 기술의 한 형태로, 지속적으로 변경되는 데이터를 모든 참여 노드에 기록한 변경 리스트로서 분산 노드의 운영자에 의한 임의 조작이 불가능하도록 고안되었다. 블록체인 기술은 비트코인을 비롯한 대부분의 암호화폐 거래에 사용된다. 암호화폐의 거래 과정은 탈중앙화된 전자 장부에 쓰이기 때문에 블록체인 소프트웨어를 실행하는 많은 사용자들의 각 컴퓨터에서 서버가 운영되어, 중앙에 존재하는 은행 없이 개인 간의 자유로운 거래가 가능하다.

2.6.1 이더리움

이더리움(Ethereum은 블록체인 기술을 기반으로 한 스마트 계약을 실행할 수 있는 분산 컴퓨팅 플랫폼이다.

이더리움은 비탈릭 부테린이 개발한 블록체인 기반 플랫폼으로 기존의 1세대 블록체인인 비트코인의 거래 내역만 저장할 수 있다는 한계를 극복하고 거래 내역 이외에도 스마트 계약 등 추가 정보를 기록할 수 있다. 이러한 핵심 기능을 통해 스마트 계약에 대한 데이터를 블록체인에 저장하고 중앙 권한 없이 계약 시스템에 접근하고, 관련 DApp을 관리할 수 있게 블록체인 기술을 기반으로 분산 컴퓨팅 네트워크를 제공한다.
이와 같은 점에 착안하여 여러 사용자의 컴퓨팅 자원을 이용해 이더리움 가상 머신(Ethereum Vitural Machine, EVM)을 만들고 가상 머신을 이용해 SNS, 투표, 거래, 금융 관련 등 다양한 데이터를 저장하는 시스템을 개발할 수 있게 되었다.

가상 머신은 이더리움에 미리 배포해놓은 스마트 계약과 상호 작용하면서 분산 애플리케이션 (Decentralized App, DApp)에 필요한 데이터를 기록하고 갱신할 수 있다. 데이터를 기록하거나 갱신하는데 이더리움에서 제공하는 이더(Ether)라는 통화를 '가스'라는 대가로 지불한다. 이더는 가상화폐의 일종으로 가스비로 사용되고 암호화폐 거래소에서 활발하게 거래되고 있다.

2.6.2 솔리디티

솔리디티는 이더리움 등의 블록체인 플랫폼에서 스마트 계약의 실질적 구현에 이용되는 계약 지향 프로그래밍 언어이다. 솔리디티는 이더리움과 같은 블록체인 플랫폼상에 스마트 계약의 비즈니스 로직을 담아 프로그래밍하는 것을 목표로 개발되었고, 이더리움 플랫폼 위에서 작동하는 분산 앱(DApp)을 개발하기 위한 핵심 프로그래밍 언어이다.

이더리움 플랫폼을 기반으로 한 분산 앱의 과정을 예로 들면, 개발자는 솔리디티를 통해서 블록체인에 올라갈 스마트 계약을 작성하고 이것을 solc로 컴파일해서 생성된 이더리움 바이트코드를 블록체인에 등록한다. 사용자는 웹 UI를 통해 블록체인에 올라간 스마트 계약과 상호 작용하면서 분산 앱 서비스를 이용할 수 있다.

2.6.3 Web3.js

Web3.js 모든 자료와 정보가 분산화, 분권화된 차세대 네트워크 구조로서, 서버가 없는 인터넷 분산형 웹을 말한다. 자바스크립트 기반으로 분산 앱이나 서비스를 구현할 때 매우 유용하다.

Web3.js는 내부적으로 HTTP나 IPC를 통해 JSON-RPC API를 호출하게 되어있다. 이를 통해 개발자들은 JSON-RPC로 통신하는 불편함을 덜고 Web3.js를 이용하여 쉽게 자바스크립트 인터페이스로 이더리움 노드와 상호작용할 수 있게 해준다.

2.6.4 Third Web

Third Web은 Web3 개발자들을 위한 모든 블록체인에서 호환되는 Web3 SDK이다. 스마트 계약을 더욱 간편하게 배포할 수 있고 배포한 계약의 유지 보수, 관리를 쉽게 만들어준다. 또한 인증 프로토콜, 분산 스토리지, 스마트 월렛, 온 체인 데이터 관리 등 여러 가지 도구들을 제공하는 플랫폼이다.

2.6.5 메타마스크

메타마스크는 이더리움 플랫폼 상에서 사용되는 개인 지갑으로 편리하고 안전하게 관리할 수 있는 구글 확장 프로그램이었으나 현재는 모바일 앱으로도 사용할 수 있고 네트워크를 등록한 이후에 다른 블록체인 네트워크 상에서도 이용할 수 있다. 이더리움 노드에 직접적으로 접근하지 않아도 메타마스크를 통해 브라우저의 분산 앱(DApp)에 접근해 상호작용이 가능하다. 메타마스크의 핵심 기능으로 키 관리(Key management), 이더 송금, 토큰 확인 및 관리 등이 있다. 또한 사용자로 하여금 블록체인 트랜잭션을 서명할 수 있는 사용자 인터페이스를 제공한다.

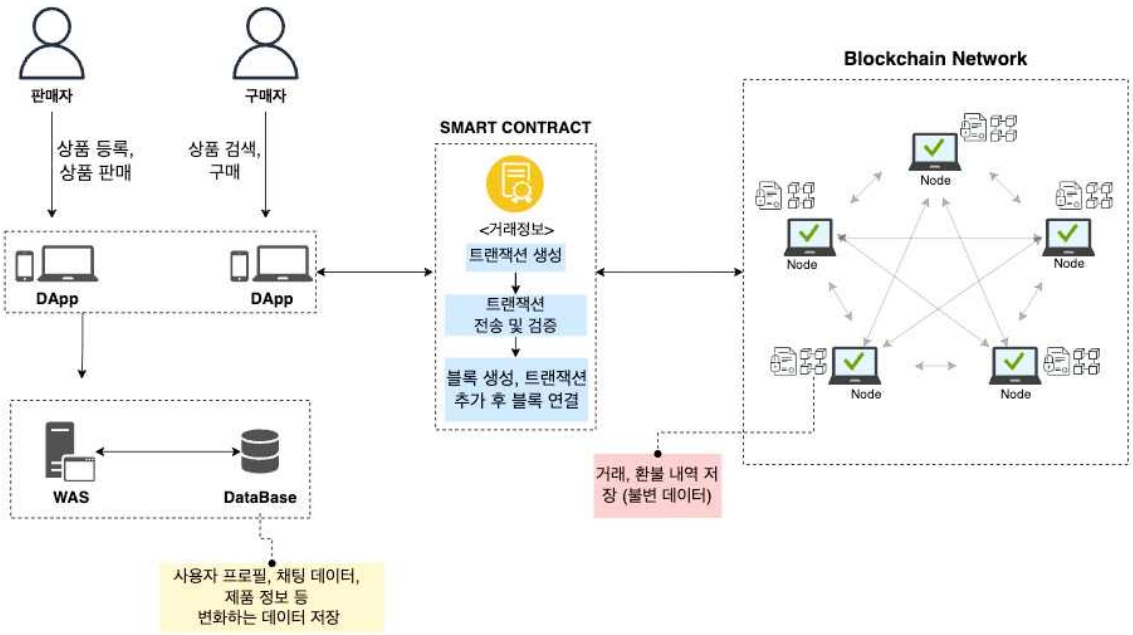
사용자의 키로는 퍼블릭 키(Public key), 프라이빗 키(Private key), 복구용 니모닉 코드(Mnemonic)가 있다. 퍼블릭 키는 가상화폐를 입금할 개인 지갑 주소가 되고 프라이빗 키는 계좌에서 자금을 송금하거나 서명할 때 사용하는 키이다. 프라이빗 키에서 퍼블릭 키를 추출할 수 있지만, 퍼블릭 키에서 프라이빗 키를 추출할 수는 없다. 실질적으로 키 관리를 하는 메타마스크에서는 프라이빗 키를 보관하다가 바로 사용하는 것은 위험하기 때문에 프라이빗 키를 꺼낼 때 정해둔 패스워드를 통해 보안에 신경 썼다. 또한 패스워드를 분실할 시 지갑 생성 시 만들어진 니모닉 코드를 통해 지갑 복구가 가능하다.

3. 본론

3.1 서비스 구성

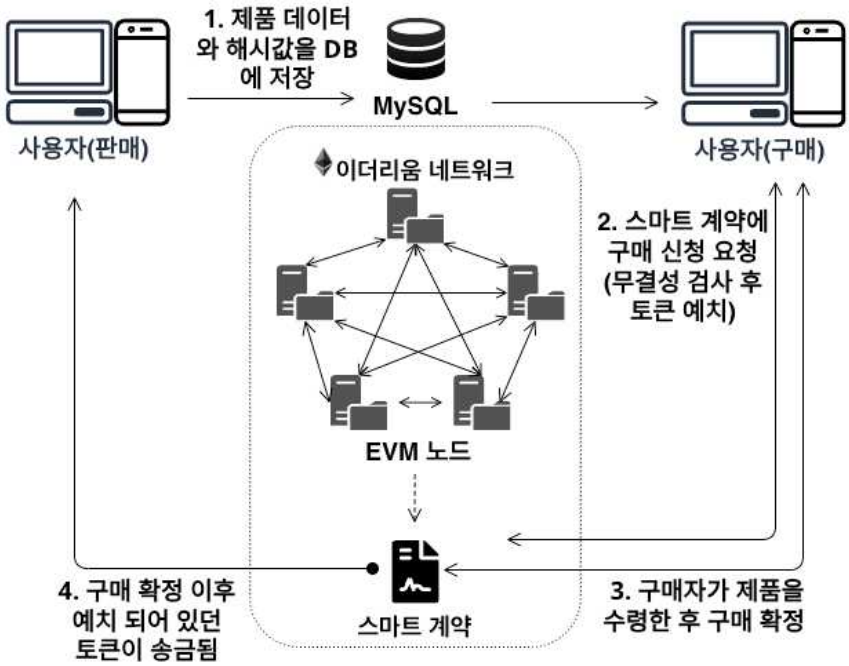
그림 1은 웹 애플리케이션의 서비스 구상도이다. 전통적인 DB와 블록체인에는 각각 다른 데이터가 저장된다. 거래 내역과 같은 필수적으로 유지해야 하는 데이터들은 블록체인에 기록되고 채팅 데이터, 사용자 프로필 등 가변적인 데이터는 Mysql DB에 저장된다. 그렇기 때문에 기능마다 요청이 다르다. 상품 등록, 구매 등의 기능은 스마트 계약의 함수들을 실행하고 이외의 기능은 API 요청을 보낸다.

구매자가 구매할 때 사용한 토큰은 바로 판매자에게 송금되지 않고 스마트 계약에 예치된다. 이후 구매자가 물건을 확인하고 클라이언트에서 물건 확인 버튼을 누르면 그때 스마트 계약에 예치되어 있던 토큰이 판매자에게 릴리스 된다.



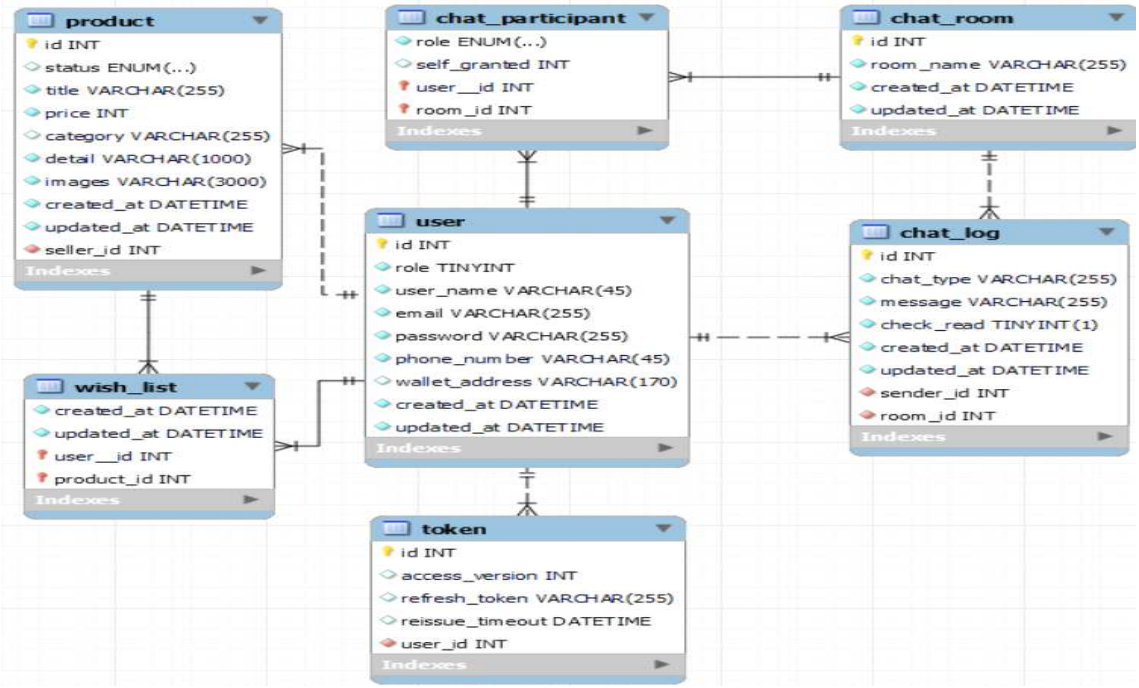
[그림 1. 서비스 구상도]

그림 2는 플랫폼의 에스크로 기능에 대한 구상도이다.



[그림 2. 에스크로 기능 구상도]

그림 3은 플랫폼의 가변 데이터를 저장하는 DB에 대한 ERD(Entity Relationship Diagram)이다.



[그림 3. 데이터베이스 ERD]

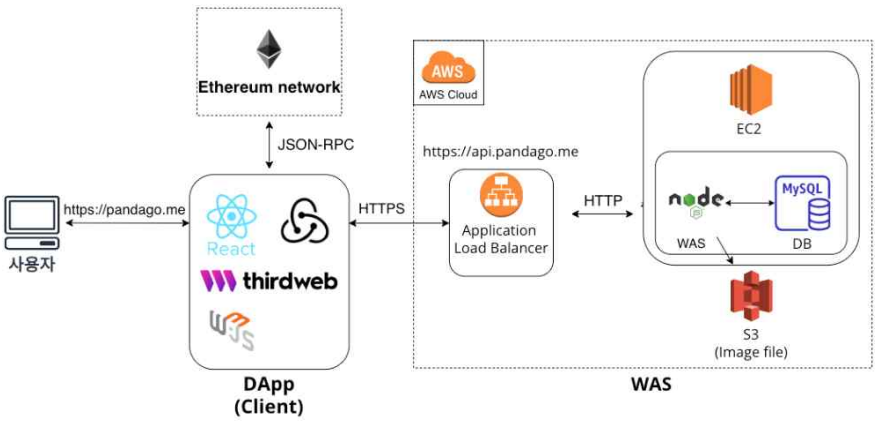
3.2 개발 환경

분류	내용
프로그래밍 언어	javascript, 솔리디티
프레임워크/런타임	NodeJs, React, Redux, express, socket.io, Thirdweb, web3.js
데이터베이스	MySql
인프라	AWS, Vercel, 이더리움 네트워크
도구	Git, vscode, 가나슈, 메타마스크

[표 1. 개발 환경]

3.3 아키텍처 구성

그림 4는 전체적인 서비스 구상도에서 DApp과 WAS에 대한 자세한 아키텍처 구상도이다. DApp에서 온 요청 중 블록체인에 데이터를 기록하거나 데이터를 가져오는 요청은 그림 1에서의 과정처럼 스마트 계약에 Third Web과 Web3.js 를 통해 요청을 보내고 가변 데이터에 대한 API 요청은 WAS로 보내게 된다. 클라이언트에서 온 API 요청은 express 서버를 배포한 AWS의 ALB(Application Load Balancer)로 보내지게 된다. 이때 요청은 HTTPS 통신으로 보내지고 ALB는 들어온 트래픽을 EC2 인스턴스로 보낸다. express 서버는 Mysql 서버와 데이터를 주고, 받고 AWS S3에 이미지를 저장한다. 부가적으로 채팅 관련 통신은 소켓을 통해 통신한다.



[그림 4. 아키텍처 구상도]

3.4 프로그램 및 스마트 계약 구성

프로그램의 구성과 서비스 사용 시나리오, 그 과정에서의 스마트 계약 구성은 다음과 같다. (홈페이지의 이미지와 자세한 사항은 4장 서비스 안내와 웹페이지 서비스 이용 가이드 참조)

1. 사용자는 회원가입을 진행하고 로그인한다.

사용자 로그인 후 인증에는 access token과 refresh token을 도입했다. 로그인 시 access token과 refresh token을 JWT로 발급해 각각 로컬 스토리지와 쿠키에 저장한다. refresh token은 DB token테이블에 저장해둔 후 access token을 검증해 기간이 만료되었다면 DB에 있는 사용자의 refresh token을 체크해 refresh token이 만료되지 않았다면 access token을 재발급해준다. 만약 두 가지 토큰을 모두 탈취되었을 경우 보안에 취약해지기 때문에 access token 재발급 가능 시간에 제한을 두었다.

2. 웹서비스에 메타마스크 지갑을 연동한다.

로그인 후 우측 상단의 '내 지갑' 버튼을 클릭한 후 '지갑 연결' 버튼을 클릭해 Sepolia 네트워크로 메타마스크 지갑을 웹사이트에 연결한다.
(자세한 안내는 4장 서비스 안내와 웹페이지 서비스 이용 가이드 참조)

3. 제품 판매를 원하는 사용자는 좌측 상단(모바일은 우측 상단 메뉴버튼)의 판매하기 버튼을 누르고 제품의 상세 정보를 입력한 후 등록하기 버튼을 누른다.

스마트 계약을 구현할 때 구매자가 제품을 구매하면 곧바로 토큰이 판매자에게 송금되지 않고 계약 자체에 예치되었다가 구매자가 제품을 확인하면 판매자에게로 토큰이 송금되는 에스크로 기능을 필수적으로 넣어 사기의 위험을 낮췄다. 그림 5는 구매 진행 중인 제품의 에스크로 데이터와 제품 데이터를 구조체로 선언한 부분과 제품 등록 함수 부분이다.

```
struct Product {
  uint32 productId;
  uint256 price;
  uint32 sellerId;
  address sellerAddress;
  State status;
}
mapping(uint256 => Product) products;

struct EscrowData {
  uint32 productId;
  uint32 buyerId;
  address buyerAddress;
  uint32 sellerId;
  address sellerAddress;
  uint256 amount;
  bool isApprove;
}
mapping (uint256 => EscrowData) escrows;
```

[그림 5. 진행중인 에스크로 데이터와 제품 데이터 구조체]

```
//제품 등록
function addProduct(uint32 _sellerId, uint32 _productId, uint256 _price)
public virtual override returns (bool) {
    require(_price>=0, "price of the product is less than zero");
    increaseTotalProduct();

    products[_productId] = Product(_productId, _price * 10**decimals(),
    _sellerId, msg.sender, State.SALE);

    require(products[_productId].sellerAddress != address(0), "Faild to
    register product");
    emit ProductRegister(msg.sender, _productId, _price * 10**decimals(),
    block.timestamp);
    return true;
}
```

[그림 6. 제품 등록 함수]

4. 내 지갑 버튼을 클릭하고 토큰 발급하기 버튼을 눌러 토큰을 발급한다.

스마트 계약의 토큰 발급 함수를 실행해 토큰을 사용자에게 발급해준다. 현재 프로젝트에서는 결제 모듈이 구현되어 있지 않기 때문에 인증과 결제 없이 토큰이 발급되지만 추후 결제 모듈을 추가된 후에는 결제 후에 토큰을 발급받을 수 있다.

5. 제품 구매를 원하는 사용자는 상단 검색창이나 구매하기 버튼으로 제품을 검색한다.

제품은 카테고리와 검색어 등으로 페이지네이션 되고 원하는 제품을 클릭하면 상세 페이지로 이동한다.

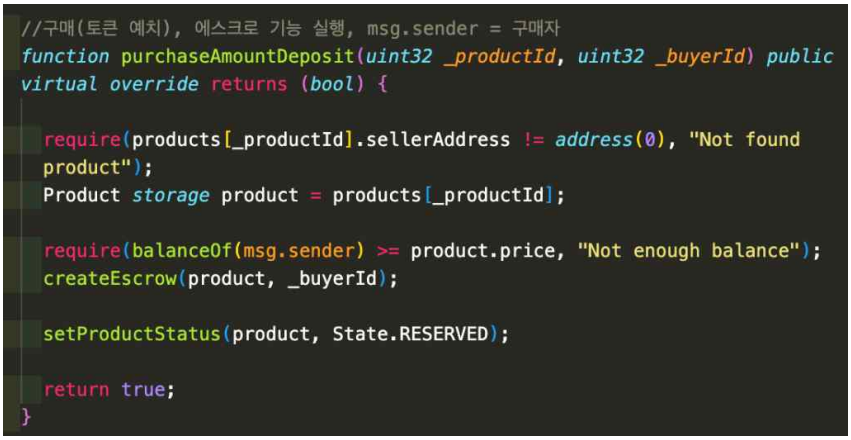
6. 제품 상세 페이지에서 구매 버튼을 누르면 메타마스크 지갑 서명 이후 구매가 진행되고 제품은 구매 진행 중 상태로 바뀐다.

제품 구매 신청과 함께 계약의 에스크로 구조체에 데이터가 추가되고 토큰이 계약 자체에 예치된다. 그림 7은 에스크로 데이터를 생성하고 토큰을 계약에 예치하는 로직이다. address(this)는 현재 스마트 계약의 주소를 뜻한다.

```
function createEscrow(Product storage _product, uint32 _buyerId) internal returns(bool) {
    escrows[_product.productId] = EscrowData(_product.productId, _buyerId, msg.sender, _product.
    sellerId, _product.sellerAddress, _product.price, false);
    EscrowData memory escrow = escrows[_product.productId];
    require(escrow.buyerAddress != address(0), "Faild to create escrow");
    emit EscrowCreate(msg.sender, _product.sellerAddress, _product.productId, _product.price,
    block.timestamp);

    transfer(address(this), _product.price);
    emit EscrowDeposit(escrow.productId , _product.price, block.timestamp);
    return true;
}
```

[그림 7. 에스크로 데이터 생성 및 토큰 예치 함수]



```
//구매(토큰 예치), 에스크로 기능 실행, msg.sender = 구매자
function purchaseAmountDeposit(uint32 _productId, uint32 _buyerId) public
virtual override returns (bool) {

  require(products[_productId].sellerAddress != address(0), "Not found
  product");
  Product storage product = products[_productId];

  require(balanceOf(msg.sender) >= product.price, "Not enough balance");
  createEscrow(product, _buyerId);

  setProductStatus(product, State.RESERVED);

  return true;
}
```

[그림 8. 제품 구매(토큰 예치) 함수]

7. 제품 상세 페이지에서 구매 이외의 채팅방 생성, 찜 목록 추가 등을 할 수 있다.

구매하기 위에 있는 찜하기, 판다톡 버튼을 누르면 각각 찜 목록 추가, 채팅방 생성을 할 수 있다. 이미 해당 제품의 판매자와 채팅을 진행하고 있는 경우 기존 채팅방으로 이동한다. 또한 이미 찜 목록 되어있다면 버튼의 색이 다르게 표시되고 해당 제품을 몇 명의 사용자가 찜했는지 표시된다.

8. 구매자는 제품을 수령한 이후 내 정보 페이지의 구매 진행 중 상품에서 구매 확정을 해야한다.

구매를 확정하면 컨트랙트 상에 예치되어 있던 토큰이 판매자에게 릴리즈된다. 그림 9는 구매 확정 함수이다.

```
//구매완료, 제품 확인
//msg.sender = 구매자
function purchaseConfirmation(uint32 _productId) external virtual
override returns (bool) {
  require(escrows[_productId].buyerAddress != address(0), "Not found
  escrow");
  EscrowData storage escrow = escrows[_productId];

  require(balanceOf(address(this)) >= escrow.amount, "Not enough balance
  of contract");
  confirmProductAndReleaseApprove(escrow);
  releaseToSeller(escrow.isApprove, escrow.sellerAddress ,escrow.amount);
  emit CompleteTransaction(escrow.productId, block.timestamp);
  delete escrows[_productId];
  delete products[_productId];
  decreaseTotalProduct();
  return true;
}
```

[그림 9. 구매 확정 함수]

9. 내정보 수정하기

그 이외의 내 정보에서 프로필 수정하기, 판매 중인 제품. 구매 진행 중(토큰 예치), 구매 완료된 제품, 찜 목록 등을 확인할 수 있다. 판매 중인 제품, 구매 진행 중인 제품, 구매가 완료된 제품은 스마트 계약에 미리 만들어둔 인덱스 값을 통해 블록체인에서 조회할 수 있다.

4. 서비스 안내

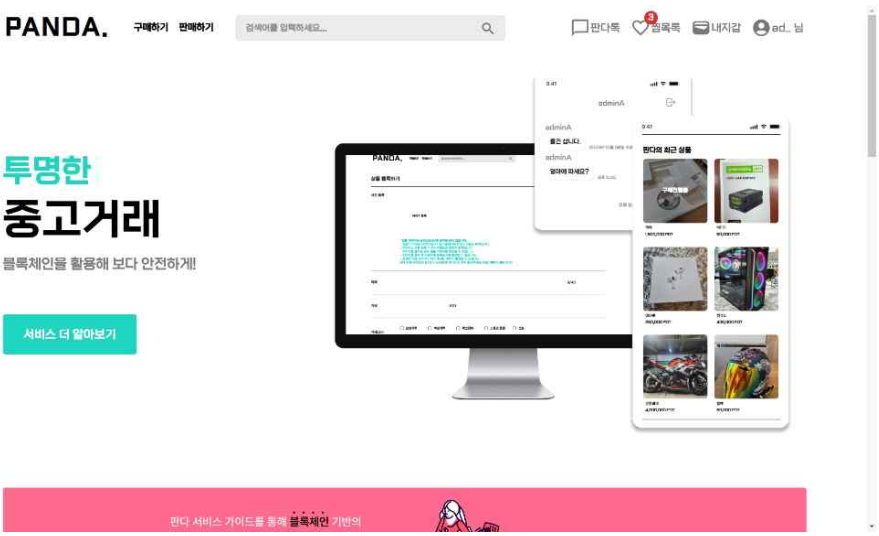
4.1 웹서비스 구성

- **웹서비스 주소**: https://pandago.me

- **서비스 메뉴**

1. 로그인, 회원가입
2. 내 지갑 - 지갑 연결, 토큰 발급 등을 할 수 있다.
2. 구매하기 - 원하는 제품을 검색해 구매할 수 있다.
3. 판매하기 - 판매할 제품을 등록할 수 있다.
4. 판다톡(채팅방) - 판매자 또는 구매자와 채팅할 수 있다.
5. 찜 목록 - 찜한 제품을 확인할 수 있다.
6. 내 정보 - 대시보드, 판매 중, 구매 중, 구매완료 제품을 확인할 수 있다.

그림 10은 웹서비스의 초기(메인 페이지)화면이다.

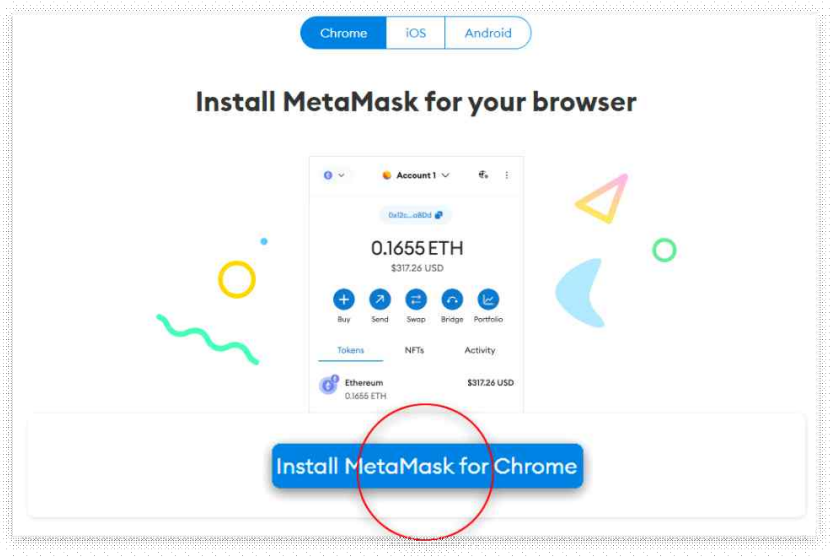


[그림 10. 웹서비스 초기 화면 (메인 페이지)]

서비스 이용에는 두 가지 필수적인 요소가 필요하다. 메타마스크 지갑과 테스트 네트워크 코인(가스비)가 필요하다. (자세한 사항은 웹페이지 메인페이지의 서비스 이용 가이드와 4.2 메타마스크 연결의 내용을 참고)

4.2 메타마스크 연결

해당 서비스를 이용할 때 필수적인 요소로 메타 마스크 지갑이 필요하다. 크롬 확장 프로그램(모바일인 경우 앱)인 메타마스크가 설치되어 있지 않다면, 메타마스크 다운로드 페이지 상단의 다운로드를 클릭하고 그림 11과 같이 해당하는 환경(브라우저/모바일)에서 설치 버튼을 클릭한다.



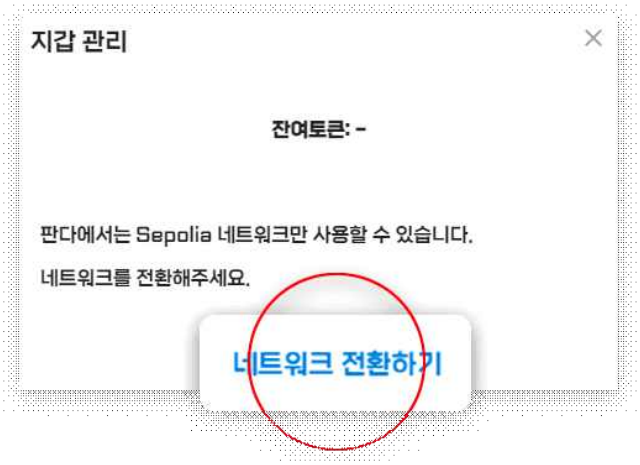
[그림 11. 메타마스크 다운로드 화면]

이후, 웹사이트 우측 상단에 있는 지갑 버튼을 눌러 메타마스크 지갑을 연결한다.

[그림 12. 웹사이트 헤더의 내 지갑 메뉴]

[그림 13. 내 지갑 팝업창]

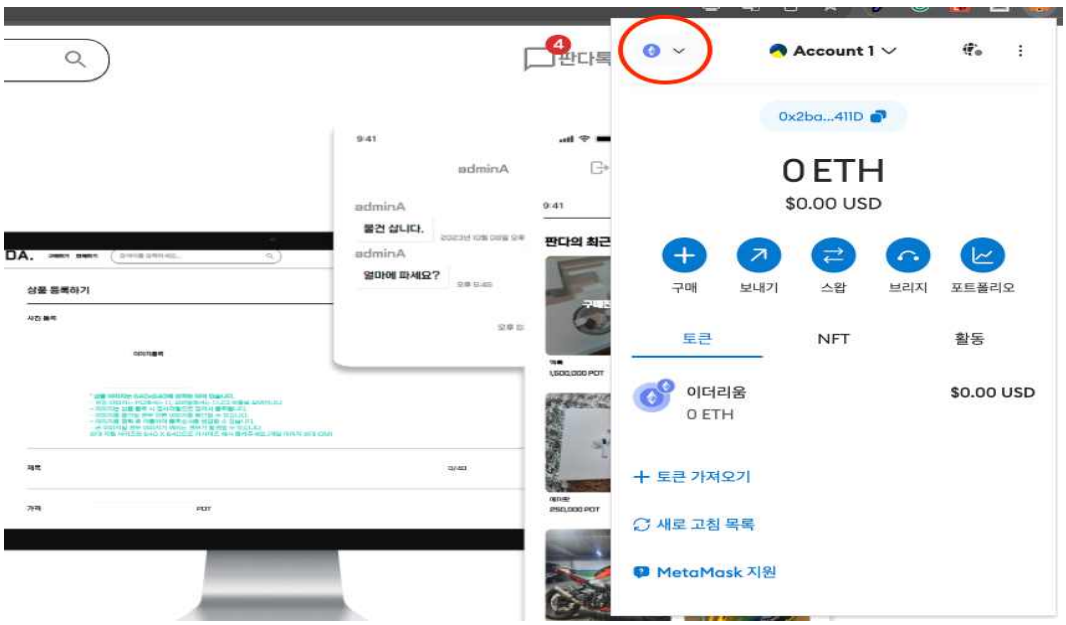
메타마스크의 블록체인 네트워크가 세폴리아(Sepolia)로 지정되어 있지 않을 시 지갑이 연결되어도 잔액이 표시되지 않는다. 그런 경우 그림 14와 같이 화면에 표시되는 네트워크 전환하기 버튼을 통해 네트워크를 전환한다.



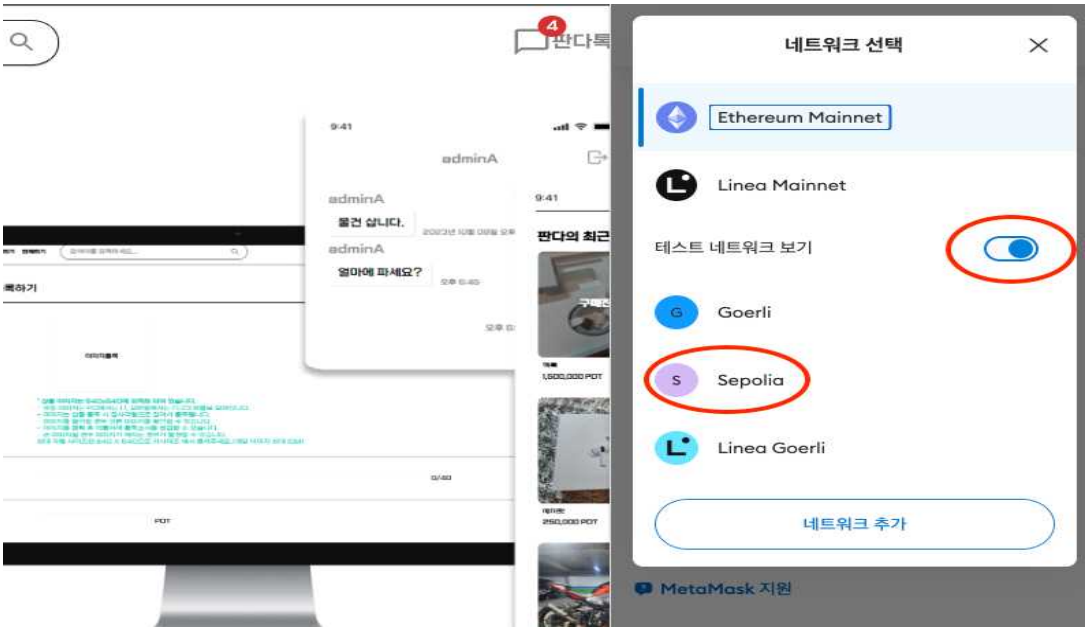
[그림 14. 네트워크 전환하기 팝업창]

웹사이트에서 지갑을 연결했을 때 네트워크가 맞지 않으면 전환할 수도 있고 보유 토큰도 표시되지만, 메타 마스크 확장 프로그램 또는 앱에서 네트워크를 전환하거나 보유 토큰을 확인하려면 다음과 같은 과정을 거쳐야 한다.

1. 메타마스크 네트워크를 그림 15, 16과 같이 테스트 네트워크 보기를 클릭하고 Sepolia 네트워크로 변경한다. 테스트 네트워크 보기가 활성화 되어있지 않으면 Sepolia 네트워크가 표시되지 않는다.

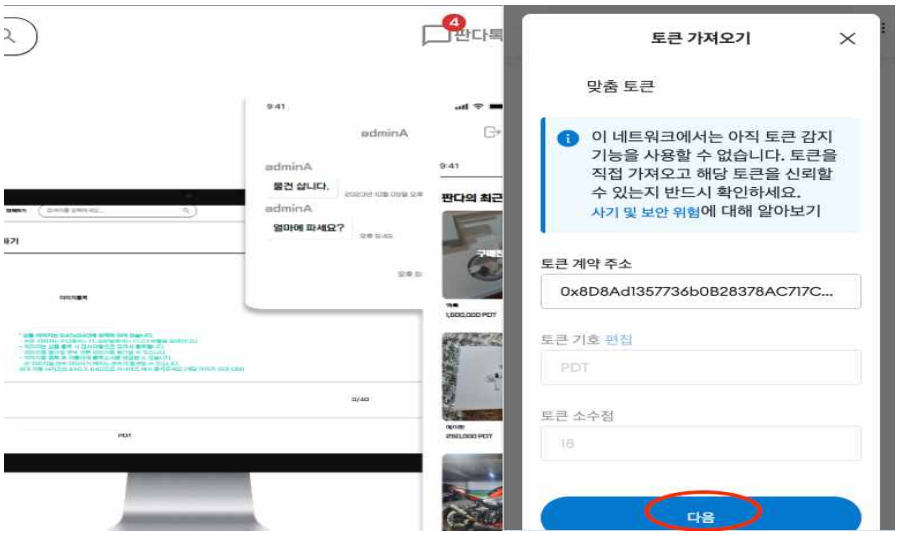


[그림 15. 메타마스크 네트워크 변경]

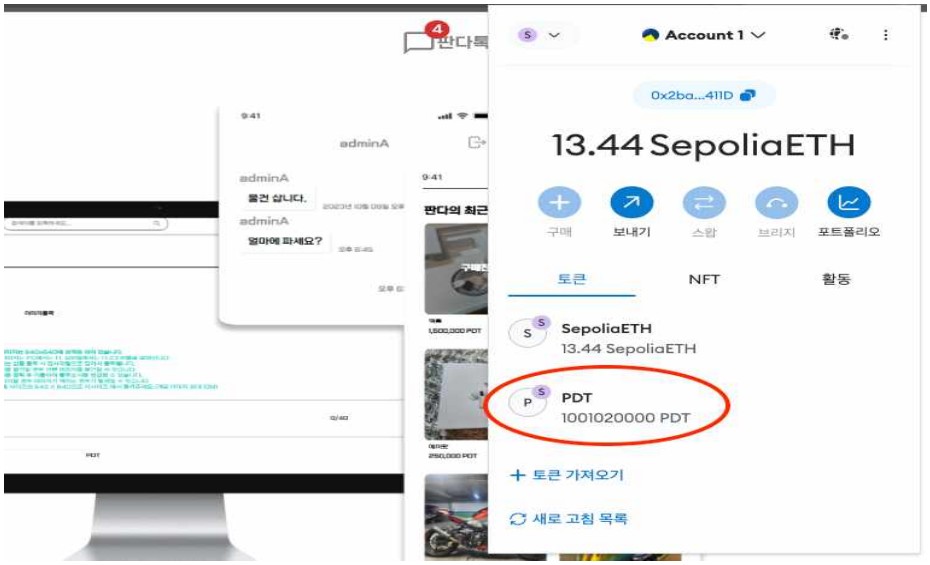


[그림 16. 메타마스크 네트워크 변경]

2. 토큰 가져오기 버튼을 클릭하고 토큰 계약 주소를 붙여넣고 다음 버튼을 누른 후 가져오기 버튼을 누른다. 토큰 기호와 토큰 소수점은 자동으로 PDT(Panda Token), 18로 변경된다. 이후 메타마스크에 보유 토큰 수가 표시된다. (토큰 계약 주소는 웹사이트 메인 페이지 슬라이드의 서비스 이용 가이드 참조)



[그림 17. 계약 주소 입력]

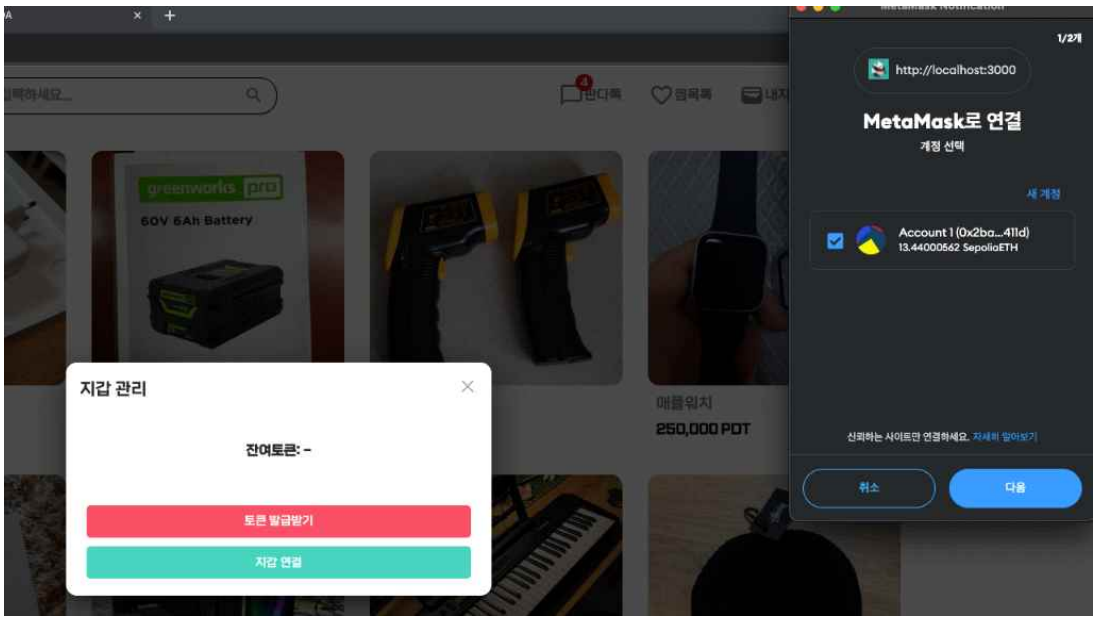


[그림 18. 토큰 가져오기 완료]

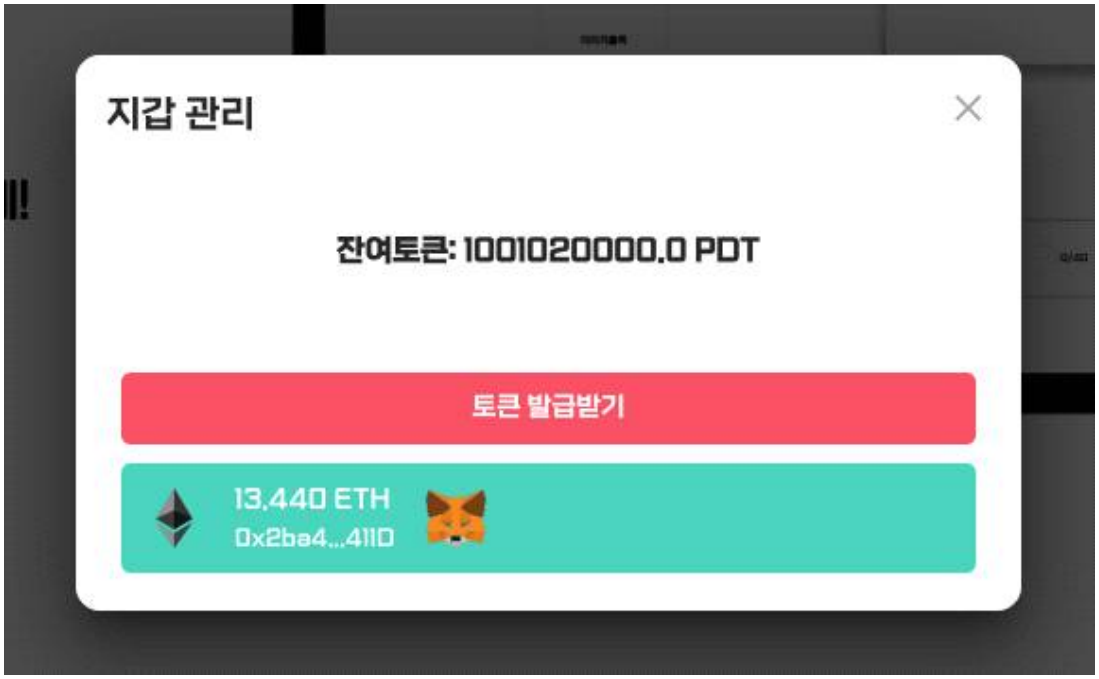
두 번째로 필수적인 요소는 테스트 네트워크의 코인(SepoliaETH)이다. 추후 플랫폼 상의 토큰으로 가스비 대체가 예정되어 있지만, 현재는 가스비를 테스트 네트워크(Sepolia)의 코인으로 지불해야 한다. 해당 테스트 네트워크 코인은 무료 지급 사이트에서 지갑 주소를 입력하면 지급받을 수 있다. (https://sepoliafaucet.com/)

4.3 토큰 발급

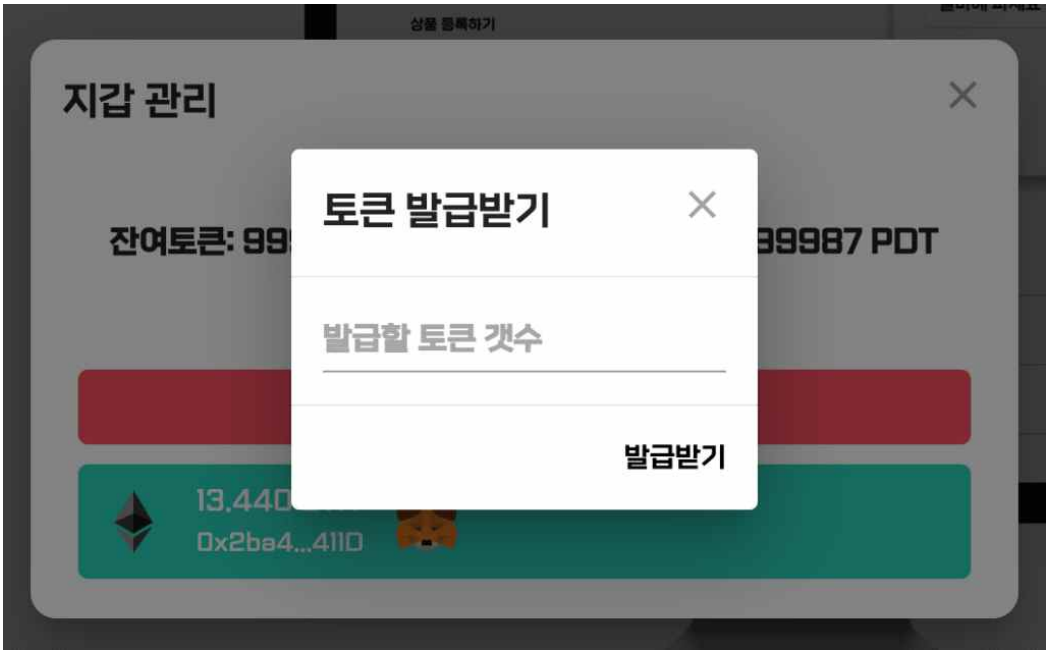
그림 19은 지갑 연결 화면이다. 우측 상단 메뉴의 내 지갑 메뉴에서 지갑을 연결하면 해당 화면이 표시된다. 연결이 완료되면 그림 19와 같이 잔여 토큰이 표시된다. 내 지갑에서 토큰 발급하기를 클릭하고 토큰 수를 입력하고 발급받기 버튼을 누르면 토큰이 발급된다.



[그림 19. 내 지갑 연결 화면]



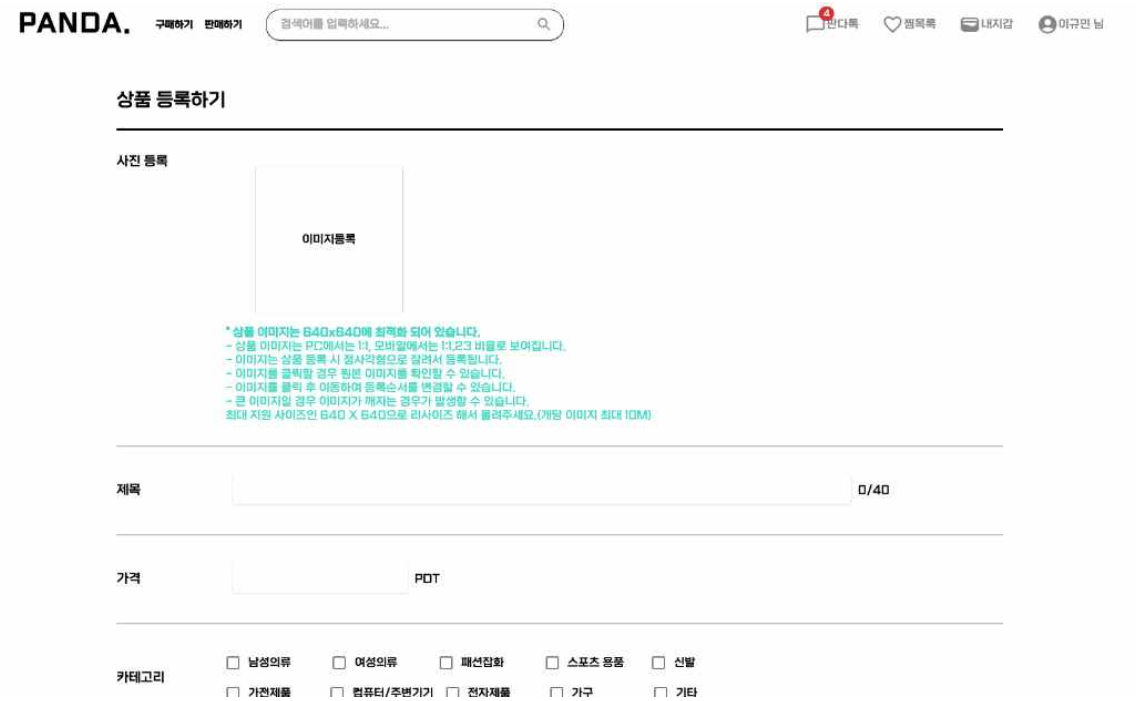
[그림 20. 지갑 연결 완료]



[그림 21. 토큰 발급 팝업창]

4.4 판매 등록

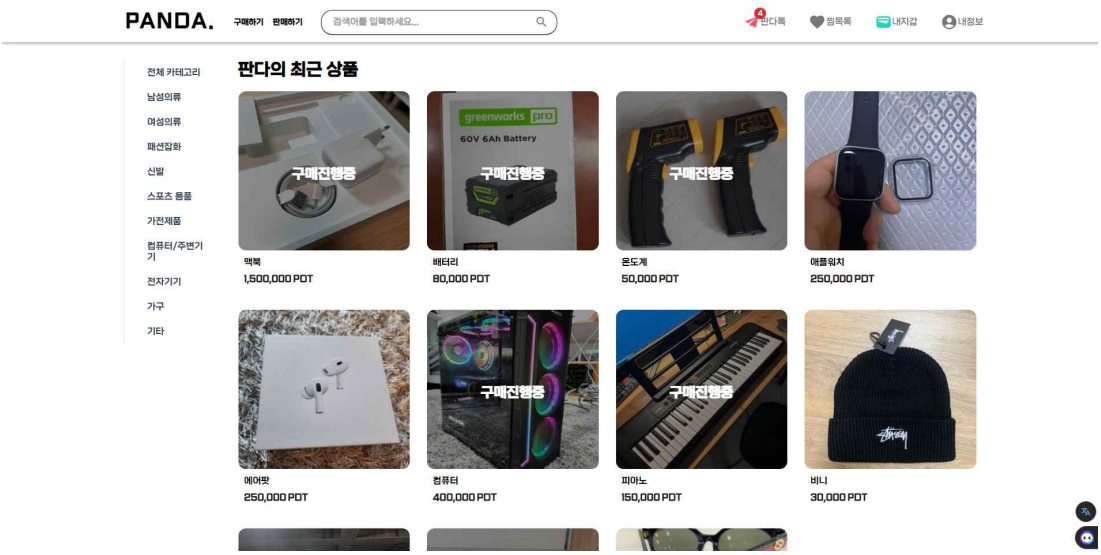
그림 22는 판매할 제품을 등록하는 웹페이지의 UI이다.



[그림 22. 판매 제품 등록]

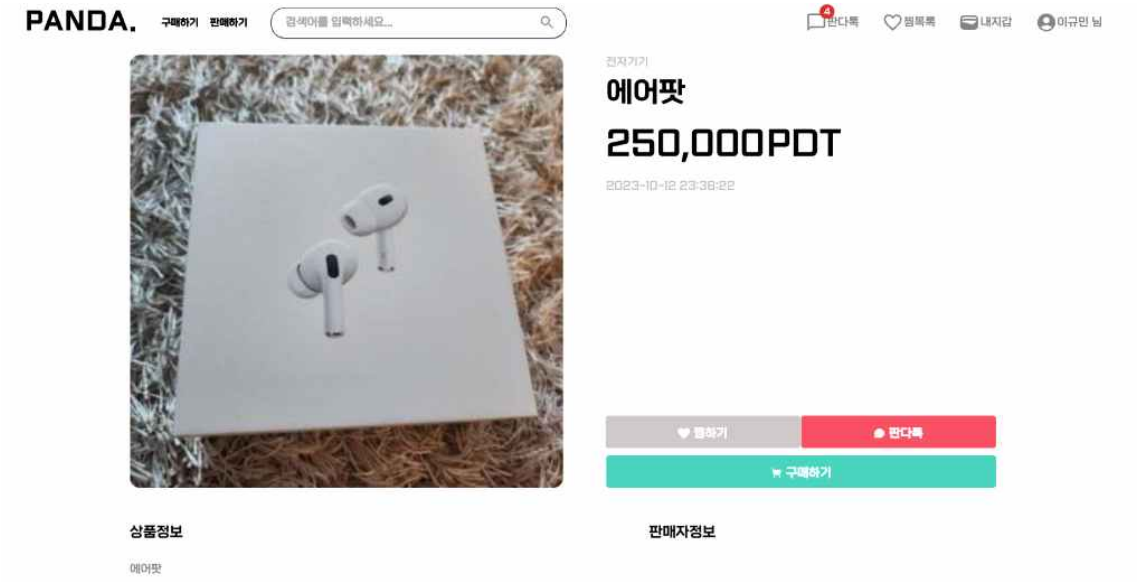
4.5 구매

그림 23은 제품 리스트를 화면에 표시하는 제품 검색 페이지이다. 이 페이지에서 자신이 원하는 카테고리나 키워드를 입력해서 제품을 찾는 것이 가능하다. 초기에 카테고리나 키워드가 입력되지 않은 상태에서는 최근 등록된 상품 리스트가 화면에 표시된다.



[그림 23. 제품 검색 페이지]

그림 24는 제품 상세 페이지 UI이다. 제품 검색 페이지에서 제품을 클릭하면 해당 페이지로 이동하게 되고 여기서 제품을 구매하거나, 판매자와의 채팅, 찜하기 등의 기능을 사용할 수 있다. 구매하기 버튼을 누르면 스마트 계약 함수가 호출되고 스마트 계약과 상호작용이 끝나면 그림 25와 같이 내정보 페이지의 구매 진행중 상품에 표시된다. 구매자가 물건을 확인하고 그림 25에 있는 구매 확정 버튼을 누르면 토큰이 판매자에게 전송된다.



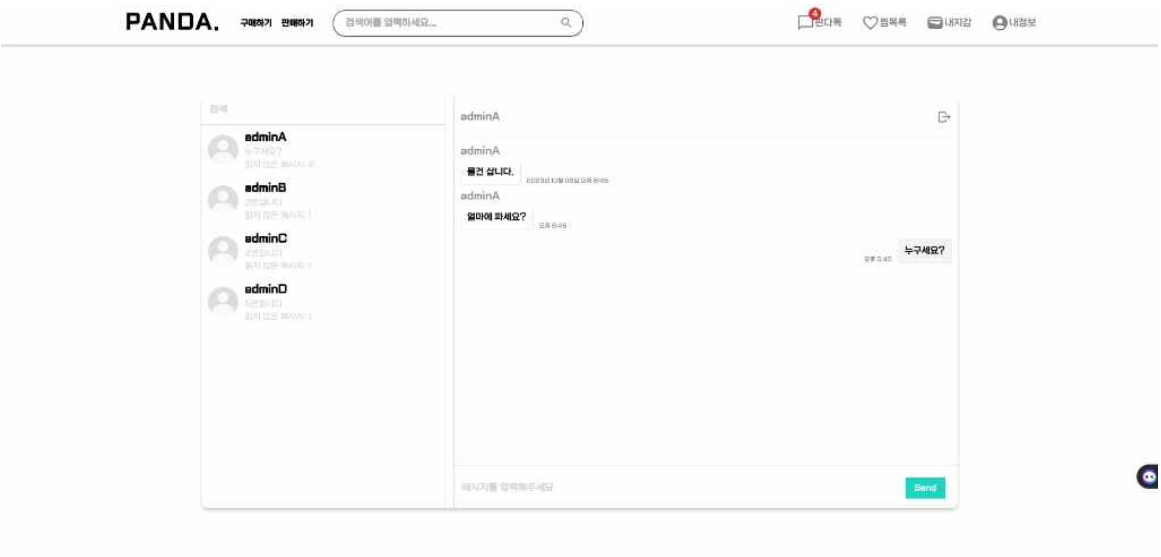
[그림 24. 제품 상세 페이지]

[그림 25] 구매 진행 중인 상품.

4.6 채팅

제품 상세 페이지에서 판다톡 버튼을 누르면 채팅을 시작할 수 있다. 그림 26은 판매자와의 웹 채팅 페이지다. 채팅 기능은 socket.io를 통해 socket 통신으로 구현했다. 채팅방에

서는 채팅 보내기와 나가기가 가능하다.



[그림 26. 웹 채팅 페이지]

4.7 내정보(대시보드)

메인 페이지의 우측 상단 메뉴의 자신의 사용자 아이디 명을 클릭한 후 드롭다운되는 메뉴 중 내 정보를 누르면 판매 중인 상품, 구매 진행 중 상품, 구매 완료된 상품, 찜 목록 등을 확인할 수 있고, 사용자 명과 소개 글을 수정할 수 있다. 그림 27은 내 정보 페이지의 UI이다.

[그림 27. 내 정보 페이지]

5. 결론

5.1 결론 및 기대효과

팬데믹 이후에 늘어난 비대면 중고 거래로 인해 그에 따른 사기, 악용 사례가 꾸준히 늘어왔지만, 그 위험성은 이용자가 감수할 수밖에 없었다. 블록체인 기술에 기반을 둔 중고 거래 플랫폼을 서비스하여 해당 플랫폼에서 거래를 하면 선입금으로 인한 사기의 위험성을 낮출 수 있다. 현재 가상화폐 폭락에 더해 아직까진 느린 성능과 어려운 사용성으로 인해 Web3.0의 발전 속도가 매우 더뎌져 있다. 블록체인 플랫폼을 이용한 분산 앱들을 만들고 블록체인 생태계에 기여하면서 Web3.0 관련 산업이 더욱 빠르게 발전될 수 있다.

5.2 추후 보완사항

블록체인을 도입한 중고 거래를 구현하는 것에 초점이 맞춰져 있어서 아직 토큰을 환전할 때의 기능 모듈이 구현되어 있지 않다. 해당 모듈을 구현할 때 보안 측면에서 굉장히 중요한 부분이기 때문에 추후 충분한 검토가 필요하다.

현재 사용 중인 이더리움 네트워크의 초당 트랜잭션(TPS) 최대 17이다. 이는 이더리움의 사용자가 많다는 점과 네트워크의 수많은 노드들이 모든 트랜잭션을 검증한다는 점에 있다. 다수의 노드가 모든 트랜잭션을 검증하기 때문에 탈중앙성과 보안성은 확보하였으나 이로 인해 폴리곤(Polygon)과 솔라나(Solana)와 같은 초당 트랜잭션이 2,000회 이상 되는 블록체인들에 비해 트랜잭션 처리 속도가 현저히 낮다. 그렇기 때문에 추후 기존 블록체인의 포화된 트랜잭션을 처리하는 레이어 2 솔루션을 도입해서 느린 트랜잭션 처리에 대한 문제를 해결할 것이다.

또한 현재는 웹 서비스만 구현해 사용 환경이 웹에 한정되어 있지만, 메타마스크 앱이 있는 한 웹, 앱과 같은 환경에 구애받지 않고 중고 거래를 이용할 수 있도록 앱으로도 빌드해 배포할 예정이다.

6. 별첨

6.1 팀원 소개

- 이규민 (팀장)
  역할: 백엔드, 스마트 컨트랙트 개발, 인프라 구축
  개인 깃허브 주소: https://github.com/legm0310

- 최병준 (팀원)
  역할: 프론트엔드, 백엔드 개발
  개인 깃허브 주소: https://github.com/qudwns017

- 김준현 (팀원)
  역할: 프론트엔드 개발, UI 디자인
  개인 깃허브 주소: https://github.com/junhyeon0218

- 이승훈 (팀원)
  역할: 프론트엔드 개발, UI 디자인
  개인 깃허브 주소: https://github.com/ysh73900

- 전준영 (팀원)
  역할: 백엔드, 스마트 컨트랙트 개발
  개인 깃허브 주소: https://github.com/patric7732

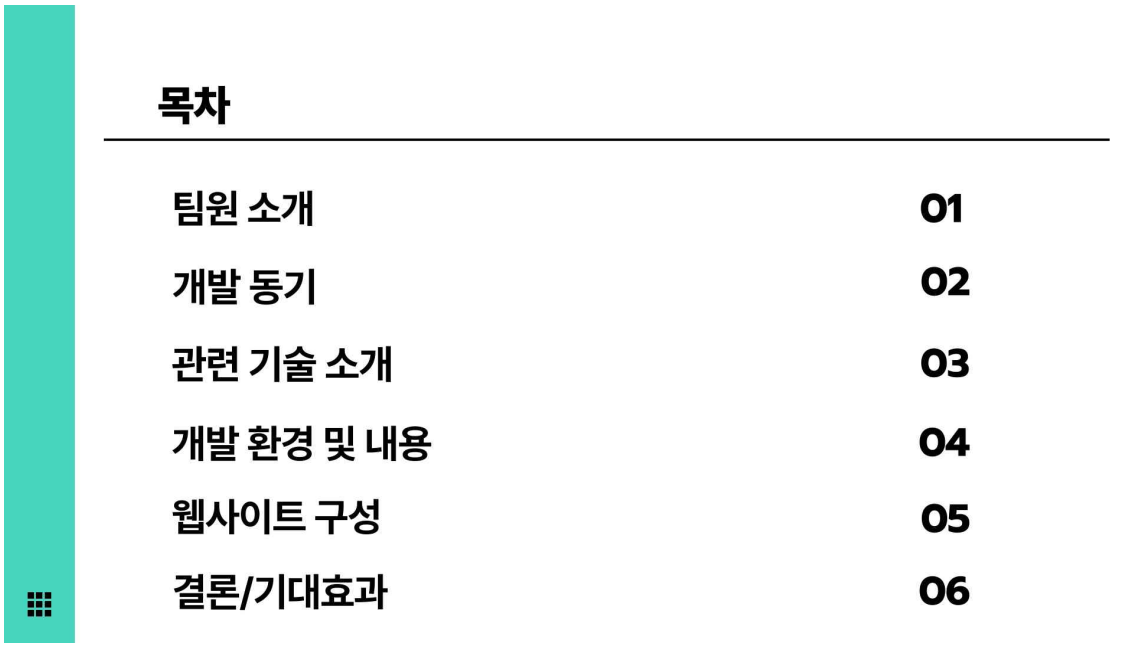
- 성우상 (팀원)
  역할: 프론트엔드 개발, UI 디자인
  개인 깃허브 주소: https://github.com/seongwoosang

6.2 소스 코드

팀프로젝트 - https://github.com/legm0310/SafeTransactionPlatform.git

6.3 발표 자료


블록체인 기반
안전거래 플랫폼
지도교수: 이병천 교수님
팀명: 개발의민족


목차
팀원 소개 01
개발 동기 02
관련 기술 소개 03
개발 환경 및 내용 04
웹사이트 구성 05
결론/기대효과 06

팀원 소개

이규민
팀장
백엔드 개발
스마트 컨트랙트 개발
인프라 구축

김준현
팀원
프론트엔드 개발
UI 디자인

성우상
팀원
프론트엔드 개발
UI 디자인

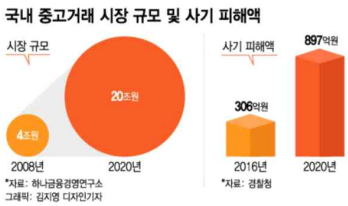
이승훈
팀원
프론트엔드 개발
UI 디자인

전준영
팀원
백엔드
스마트 컨트랙트 개발

최병준
팀원
프론트엔드 개발
백엔드 개발

개발 동기

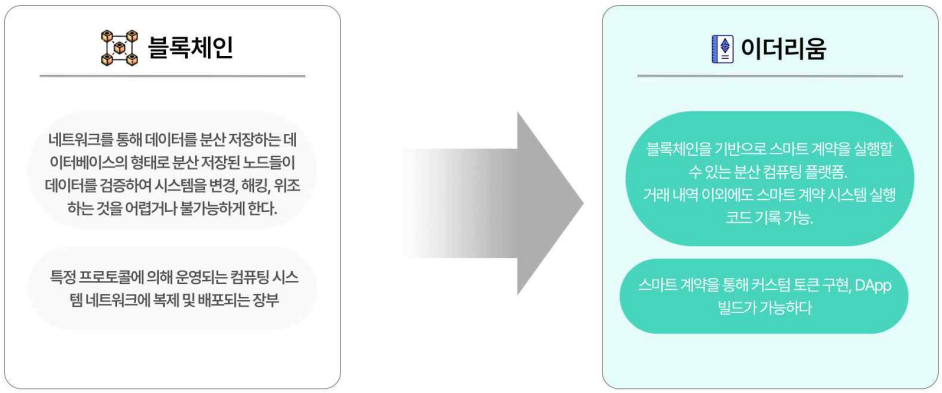
- 팬데믹 이후 비대면 중고거래가 늘어나면서 그에 따라 악용 사례, 현금 입금 유도 후 물건을 보내지 않는 등의 사기가 큰 폭으로 증가
- 기존의 중고거래 플랫폼은 사기 위험을 감수하거나 요금을 지불하는 방법으로 거래



- 악용 사례나 사기를 방지하고자 블록체인을 중고거래 시스템에 도입하고 에스크로 기능이 포함된 스마트 컨트랙트를 구현한 안전 거래 플랫폼을 기획

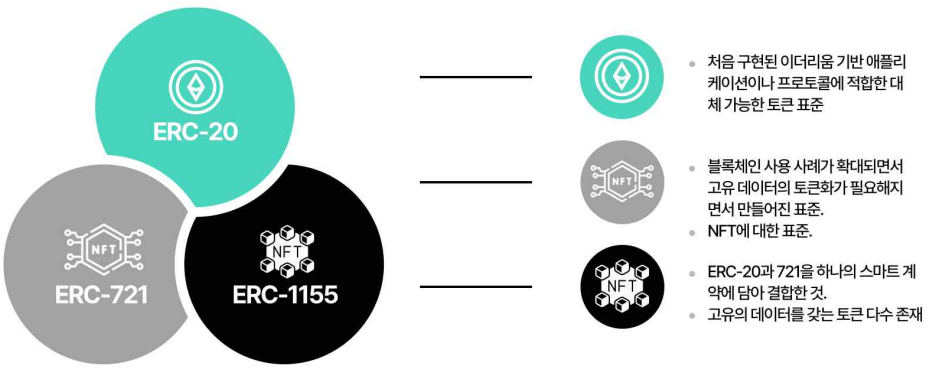
관련 기술 소개

블록체인/이더리움



관련 기술 소개

이더리움 토큰 표준 규격

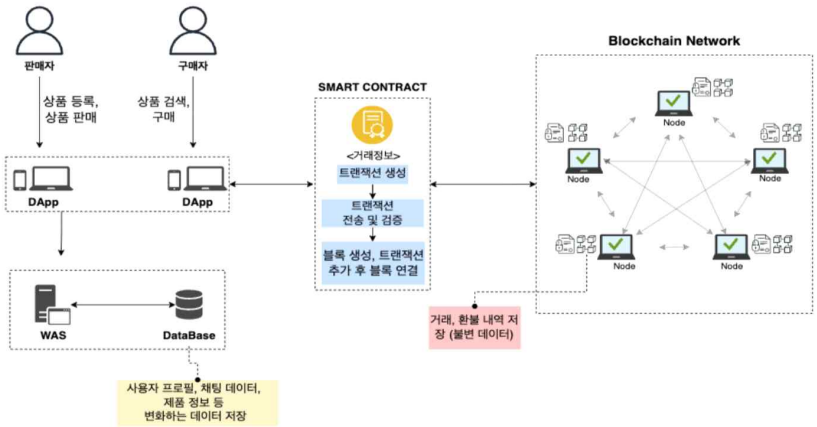


개발 환경 및 내용

개발 환경

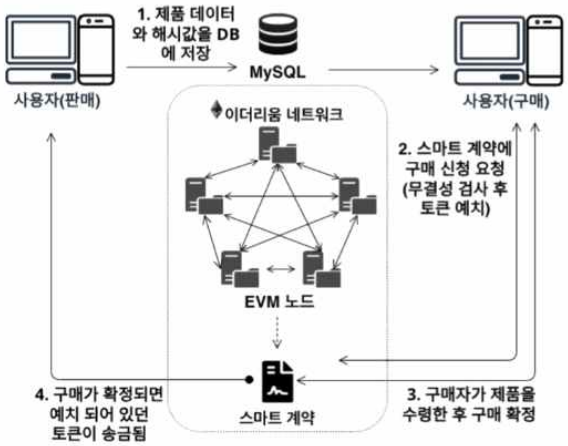
개발 환경 및 내용

개발 내용, 서비스 구상도



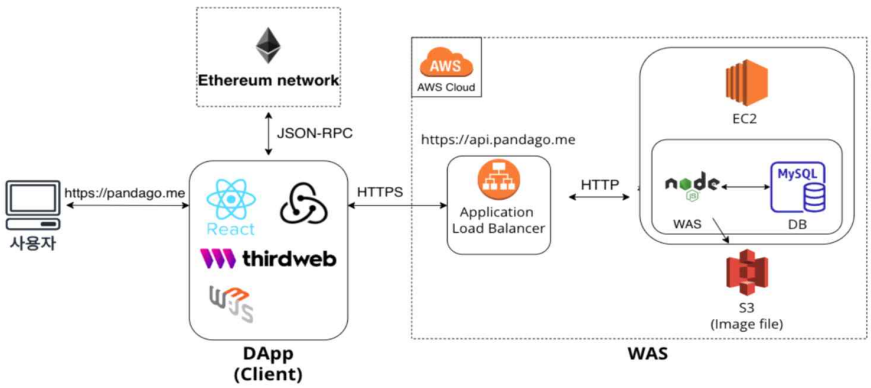
개발 환경 및 내용

개발 내용, 에스크로 기능 구상도



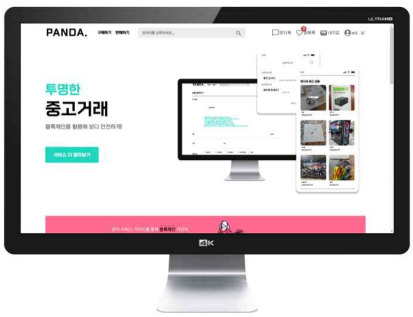
개발 환경 및 내용

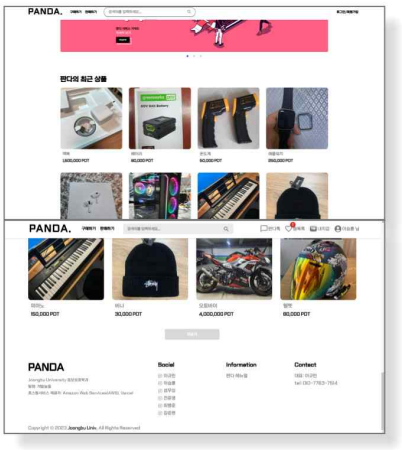
개발 내용, DApp-WAS 아키텍처 구상도



웹사이트 구성

웹 사이트 구성 - 메인 화면

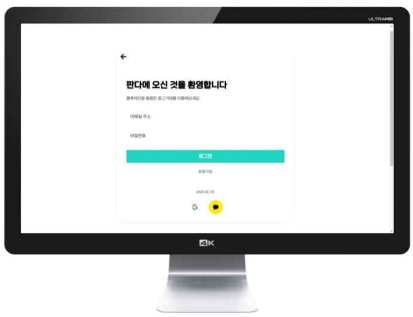


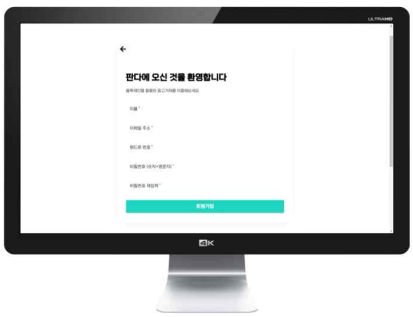


메인 페이지 상단 우측에 메뉴 (지갑관리, 내정보, 채팅, 찜목록 등) 표시
메인 페이지에는 이용 가이드, 슬라이드 배너, 최근 상품 표시

웹사이트 구성

웹 사이트 구성 - 로그인, 회원가입

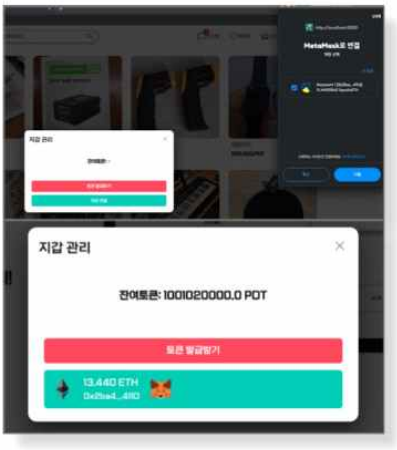




로그인, 회원가입 페이지 UI

웹사이트 구성

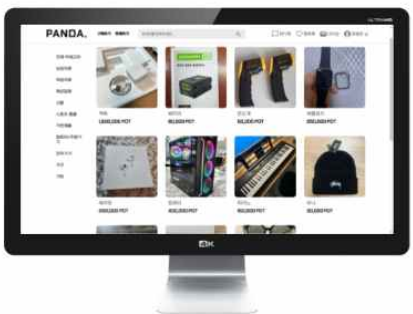
웹 사이트 구성 - 내 지갑 팝업 창



PANDA를 이용하기 위하여 토큰 발급 필요
메타마스크를 이용하여 지갑 연결
Sepolia 네트워크가 아닐 경우 네트워크 전환 버튼 표시

웹사이트 구성

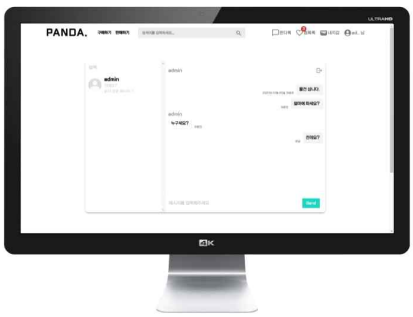
웹 사이트 구성 - 제품 등록, 검색 페이지

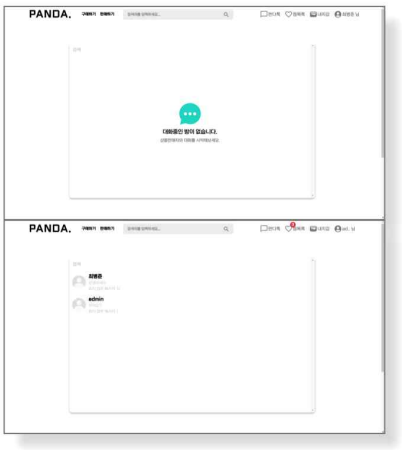


제품 등록을 위한 데이터를 입력하면 DB에 제품 데이터와 해당 데이터에 대한 해시값 저장
useSearchParams 훅을 이용하여 카테고리별 제품 정렬 및 검색 기능 구현

웹사이트 구성

웹 사이트 구성 - 웹 채팅 페이지

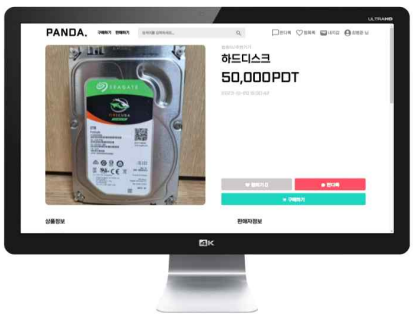


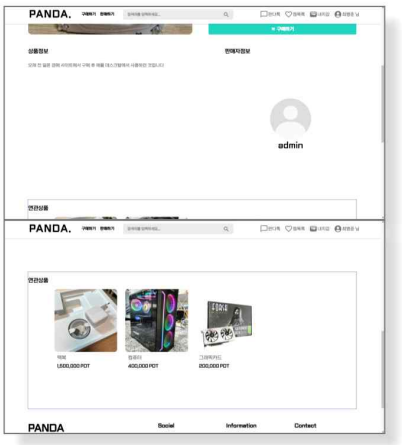


socket.io를 이용한 구매자와 판매자간 채팅 기능 구현

웹사이트 구성

웹 사이트 구성 - 제품 상세 페이지

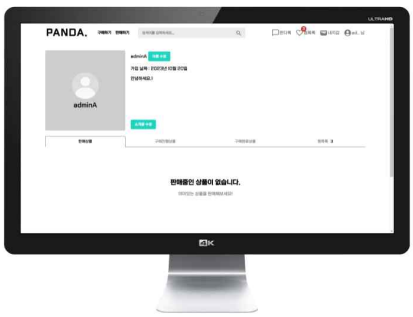


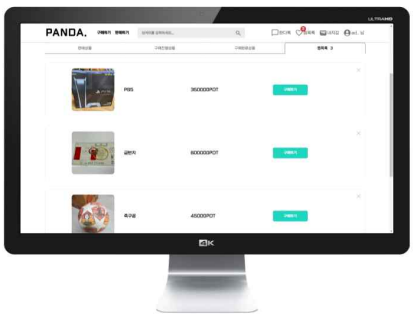


제품 구매, 찜하기, 판매자와의 채팅, 연관 상품 기능 제공
구매하기 버튼 클릭 시 블록체인의 스마트 계약 함수 호출
이후 스마트 계약에 토큰 예치

웹사이트 구성

웹 사이트 구성 - 내 정보 페이지, 찜목록





유저 정보 수정, 판매 중, 구매 진행 중, 구매 완료, 찜한 제품 확인 기능 제공
스마트 계약에 의해 미리 설정해둔 인덱스 값을 통해 블록체인에서 조회

결론/기대효과

결론/기대효과

- 블록체인 기술이 기반된 중고 거래 플랫폼에서 중고 거래를 하면 선입금을 통한 비대면 거래에서의 사기 위험성을 낮춘다.
- 이용자들이 많아지고 다른 DApp이 늘어나면서 블록체인 생태계에 기여하게 되어 Web3.0 산업과 블록체인 생태계가 더욱 발전된다.

감사합니다