요구사항 명세서(기능)

분류	요구사항명	요구사항 내용	중요도
회원 👤	이메일 로그인	- 이메일 인증 코드를 보내고 인증 코드를 입력하면 로그인 되는 형식	상
	metamask 로그인	- metamask 로그인 연동	상
	회원정보 수정	- 닉네임 - 자기소개 - 이메일 주소	상
	회원정보 조회	- 닉네임 - 자기소개 - 이메일 주소 - 가입날짜 - 프로필 이미지 - 배경 이미지 - 지갑잔액 - NFT(아이템) 보유 현황 - 내가 작성한 웹툰 목록 - 거래 내역 - 팔로우한 웹툰/작가 수 | 목록 - 팔로잉 수 | 목록	상
	NFT 전송	- 보유한 NFT 전송	
웹툰 👀	웹툰 등록	- 웹툰명 - 장르 - 태그 - 한줄요약 - 줄거리 - 대표이미지 가로형, 세로형 - 2차 창작 허용 여부(허용 시 삭제 불가)	
	웹툰 수정	- 웹툰명은 변경 불가(웹툰명 변경은 삭제 후 재등록)	
	웹툰 삭제	- ~~2차 창작 허용 시, 회차 NFT 발행 시 삭제 불가~~	
	웹툰 목록 보기	- 정렬기준에 따라 정렬 - 작품별로 보임 - 업데이트 날짜 조회(며칠 전)	
	웹툰 회차 목록 보기	- 웹툰의 상세정보 표기 - 관심등록 - 제목 - 이미지 - 작가	

분류	요구사항명	요구사항 내용	중요도
		- 줄거리 - 회차 목록 - 태그 - 첫화보기 - 공유하기	
	관심 웹툰 등록	- 관심있는 웹툰에 대해 하트(예시) 누르면 등록	
	웹툰 댓글 작성	- 대댓글 기능 구현 - 신고 - 베스트 댓글 - 좋아요 - 싫어요	
	회차 등록	- 회차명 - 회차 번호 (자동) - 대표이미지 - 원고 이미지 목록 - 작가의 말 - 댓글 등록 가능 여부	
	회차 수정		
	회차 삭제	~~NFT가 발행되지 않은 경우에만 삭제 가능~~	
	회차 상세 보기	- 이미지 파일 - 댓글 - 태그 - 버튼들(관심웹툰, 좋아요, 별점주기, 공유하기 등) - nav바(현재 회차 정보, 다음화로 넘기는 버튼 등)	
	회차 별점 평가	- 5점 만점(0.5점 단위)	
	회차 댓글 작성	- 대댓글 작성 - 신고 - 베스트 댓글 - 좋아요 - 싫어요	
스토어 🛍️	NFT 발행	- NFT 발행 시 웹툰, 시리즈, 굿즈, 팬아트 수정 및 삭제 불가 - NFT 종류: 시리즈, 팬아트, 굿즈	
	NFT 판매 등록	- 경매 방식으로 진행 - 시작 입찰가 - 바로 구매가 - 경매 종료 시간 - 재거래 시, 거래 금액 일부는 원작자에게 전송	
	NFT 목록 조회	- 작가, 가격 낮은 순, 가격 높은 순, 최근 순, 웹툰별, 최근 자주 거래 순, 검색어로 검색 및 정렬	
	NFT 상세 조회	- NFT 거래 내역, 가격 추이 - NFT 생성자, 소유자, 판매 여부(판매 중) - 판매 중일 경우 현재 입찰가, 즉시 구매가, 경매 종료 시간	
	NFT 즉시 구매	- 즉시 구매 가격을 내고 NFT 즉시 구매	

분류	요구사항명	요구사항 내용	중요도
	NFT 입찰	- 기본값: 최근 입찰가 +10% 및 사용자 지정 가능 - 최소 입찰 단위는 추후 사용하는 가상화폐에 따라 결정	
굿즈 🪄	굿즈 등록	- jpg, png, gif - 웹툰 원작자만 가능	
	굿즈 수정		
	굿즈 삭제		
팬아트 🫶	팬아트 등록	- jpg, png, gif - 웹툰이 2차 창작을 허용한 경우만 등록 가능 - 팬아트 NFT 발행할 경우 원작자에게 판매금 일부 전송	
	팬아트 수정		
	팬아트 삭제		

▼ 지갑 구현

✅ 1. 우리 사이트에서 지갑 생성하기

1) 비수탁형(Non-Custodial) 방식 (권장)

- 사용자의 개인키를 **우리가 저장하지 않고** 직접 생성 및 보관하도록 유도하는 방식.
- 예: MetaMask, Trust Wallet 같은 월렛과 유사한 방식.

🛠️ 구현 방식

- **브라우저에서 직접 지갑 생성** (예: web3.js, ethers.js 활용)
- **BIP-39 방식으로 시드 문구(Mnemonic) 생성 후 사용자에게 제공**
- **지갑 주소(공개키)는 데이터베이스에 저장 가능**, 하지만 개인키는 절대 저장하지 않음.

```
javascript
복사편집
import { ethers } from "ethers";

// 1. 랜덤 지갑 생성
const wallet = ethers.Wallet.createRandom();

// 2. 시드 문구 및 개인키 가져오기
console.log("Mnemonic:", wallet.mnemonic.phrase);
console.log("Private Key:", wallet.privateKey);
console.log("Address:", wallet.address);
```

📌 **주의:** 개인키를 우리 서버에 저장하면 안 됨. 사용자가 직접 보관하도록 유도해야 함.

2) 수탁형(Custodial) 방식

- 우리가 사용자의 지갑을 관리하는 방식 (거래소 같은 서비스에서 사용).
- 사용자의 개인키를 **우리 서버에서 생성하고 관리**해야 함.
- 보안이 매우 중요하며, 암호화 및 보안 서버가 필요함.

🛠️ 구현 방식

- 백엔드(Spring Boot 등)에서 **새로운 키 쌍 생성**
- 개인키는 **암호화된 형태**로 DB 또는 키 관리 시스템(HSM, AWS KMS 등)에 저장
- 사용자는 로그인 후 지갑을 사용 가능

```
java
복사편집
import org.web3j.crypto.*;

ECKeyPair keyPair = Keys.createEcKeyPair();
String privateKey = keyPair.getPrivateKey().toString(16);
String address = "0x" + Keys.getAddress(keyPair);

System.out.println("Private Key: " + privateKey);
System.out.println("Address: " + address);
```

📌 **주의:**

- 개인키를 안전하게 저장하려면 HSM(하드웨어 보안 모듈)이나 AWS KMS 같은 키 관리 서비스를 사용해야 함.
- 수탁형 방식은 법적 규제(KYC/AML) 및 보안 부담이 큼.

✅ 2. 기존 지갑을 가져오기 (지갑 연결 기능)

사용자가 이미 가지고 있는 지갑을 연결하도록 지원할 수도 있음.

대표적인 방법은 아래와 같습니다.

1) MetaMask, WalletConnect 등 웹 지갑 연결

사용자가 **자신의 개인키를 직접 관리**하면서 우리 사이트와 연동할 수 있음.

🛠️ 구현 방식 (MetaMask 예제)

```
javascript
복사편집
async function connectWallet() {
```

```
  if (window.ethereum) {
    const accounts = await window.ethereum.request({ method: "eth_requestAccounts" });
    console.log("Connected Wallet Address:", accounts[0]);
  } else {
    console.log("MetaMask is not installed.");
  }
}
```

📌 **주의:**

- 사용자는 MetaMask에서 로그인하고, 우리 사이트는 지갑 주소를 받아 인증만 수행.
- 트랜잭션은 사용자가 직접 서명함.

---

2) 사용자가 직접 개인키를 입력해 지갑 복원

- 시드 문구(12~24단어)나 개인키를 입력하면 기존 지갑을 복원 가능.
- 보안 위험이 커서 가급적 권장하지 않음.

```
const wallet = new ethers.Wallet("사용자의 개인키");
console.log("Wallet Address:", wallet.address);
```

📌 **주의:**

- 이 방식은 보안 위험이 크므로 사용자에게 **경고 메시지**를 반드시 띄워야 함.

✅ 3. 우리 사이트에서 제공할 기능 정리

기능	비수탁형 방식 (권장)	수탁형 방식 (고위험)
새로운 지갑 생성	사용자 브라우저에서 생성	우리 서버에서 생성
개인키 저장	사용자가 직접 보관	우리 서버에서 관리 (보안 위험 큼)
기존 지갑 연결	MetaMask, WalletConnect 지원	직접 가져오기 기능 제공 가능
보안성	🔒 안전 (사용자가 관리)	🚨 위험 (우리가 관리)
규제 부담	낮음	높음 (KYC, AML 필요)
