



포팅매뉴얼

환경 설정

1. 개발 환경

- A. Frontend
- B. Backend
- C. BlockChain
- D. CI/CD
- E. 협업 툴

2. 설정 파일 및 환경 변수

3. 외부 서비스

- MetaMask 확장 프로그램

빌드 및 배포

1. 준비하기

Clone repository

Certifications

Prepare docker

Jenkins

MySQL - docker

- MySQL docker-compose.yml
- MySQL docker build

Redis - docker

- Redis docker-compose.yml
- Redis docker build

AWS CloudFront

- **CloudFront 배포 생성**
- AWS S3
- **S3 Bucket 생성**
- 버킷 정책

Nginx

- Frontend

Backend

Ganache

2. 빌드 및 배포하기

A. Frontend

B. Backend

배포 시 특이사항

— 최종 소스 코드 빌드 및 배포 파이프라인

— 최종 docker-compose.yml 파일

C. Blockchain

서비스 이용

1. 사전 준비

메타마스크 로그인 방법

2. 시연 시나리오



S:tickey 프로젝트의 Gitlab 소스를 클론한 이후
빌드 및 배포할 수 있도록 정리한 문서.

환경 설정

1. 개발 환경

A. Frontend

- **React** : ^18.2.0
- **Node.js** : ^20.11.0
- **npm** : ^10.2.4
- **vite** : ^5.1.6
- **typescript** : ^5.2.2
- **상태관리**
 - **tanstack/react-query** : ^5.18.0
 - **zustand** : ^4.5.2
- **router - react-router-dom** : ^6.22.3
- **api통신 - axios** : ^1.6.7
- **css - tailwindcss** : ^3.4.1

B. Backend

- **Server** : 20.04.6 LTS
- **JDK** : OpenJDK 21
- **Spring Boot** : 3.2.3
- **Spring Boot JPA** : 3.2.1
- **QueryDSL** : 5.0.0
- **Spring Security6, JPA Hibernate, Lombok, Swagger, Spring Websocket, JWT** 0.11.5
- **DB**
 - **MySQL** : 8.0.36
 - **Redis** : 7.2
 - **AWS S3 Bucket**
 - **AWS CloudFront**

C. BlockChain

- **Solidity** : v0.5.16
- **Truffle** : v5.11.5
- **Ganache** : v7.9.1
- **solc** : 0.8.19

D. CI/CD

- **AWS EC2**
- **Docker** : 25.0.4
- **Nginx** : nginx/1.24.0
- **Jenkins** : 2.449

E. 협업 툴

- **GitLab** : 형상관리

- **Jira** : 이슈관리
- **Mattermost, notion** : 소통
- **Figma** : 디자인

2. 설정 파일 및 환경 변수

▼ .env

위치 `jenkins credentials`

Dockerfile 실행에 필요한 환경변수.

```
MYSQL_HOST=j10d211.p.ssafy.io
MYSQL_ROOT_PASSWORD=ZD... # mysql root password
MYSQL_USER=d211user
MYSQL_PASSWORD=ZD... # mysql root password
MYSQL_DATABASE=STICKEY
MYSQL_PORT=3306
MYSQL_EXTERNAL_PORT=33066

REDIS_PASSWORD=D2... # redis password
```

▼ .env.local

위치 : `./Stickey-frontend/.env.local`

— server

```
VITE_CONTRACT_ADDRESS=0x150EA69.... # 블록체인 컨트랙트 주소
VITE_BASE_URL=https://j10d211.p.ssafy.io/api

# websocket URL
VITE_WEBSOCKET_URL=wss://j10d211.p.ssafy.io/api/reserve

# sse URL
VITE_SSE_URL=https://j10d211.p.ssafy.io/api/notify/subscribe
```

— local

```
VITE_CONTRACT_ADDRESS=0x150EA69.... # 블록체인 컨트랙트 주소
VITE_BASE_URL=http://j10d211.p.ssafy.io:9090/api

# webSocket URL
VITE_WEBSOCKET_URL=ws://j10d211.p.ssafy.io:9091/api/reserve

# sse URL
VITE_SSE_URL=http://j10d211.p.ssafy.io:9090/api/notify/subscribe
```

▼ application.yml

위치 `./stickey-main/src/main/resources/application.yml`

```
server:
  port: 9090
  servlet:
    context-path: /api

spring:
  profiles:
    active: local # 기본설정
  group:
    local:
      - db-local
    server:
      - db-server
  include:
    - key
    - db

  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
  hibernate:
```

```

        ddl-auto: update
    properties:
        hibernate:
            format_sql: true
            jdbc:
                time_zone: Asia/Seoul
        show-sql: true
        defer-datasource-initialization: true
        open-in-view: false

servlet:
    multipart:
        max-file-size: 20MB
        max-request-size: 40MB

mail: # smtp
    host: smtp.naver.com # smtp 서버 주소
    port: 465 # smtp 서버 포트
    username: ${mail.username}
    password: ${mail.password}
    properties:
        debug: true
        mail:
            smtp:
                auth: true
                starttls:
                    enable: true
                ssl:
                    trust: smtp.naver.com
                    enable: true

springdoc: #swagger
    packages-to-scan: com.olbl.stickeymain
    default-consumes-media-type: application/json;charset=
UTF-8
    default-produces-media-type: application/json;charset=
UTF-8
    swagger-ui:

```

```

    path: /stickeymain.html
    # Swagger UI 경로 => localhost:9090/stickey-main.html
    tags-sorter: alpha
    # alpha: 알파벳 순 태그 정렬, method: HTTP Method 순 정렬
    operations-sorter: alpha
    # alpha: 알파벳 순 태그 정렬, method: HTTP Method 순 정렬
  api-docs:
    path: /api-docs/json
    groups:
      enabled: true
  cache:
    disabled: true

```

▼ application-db.yml

위치 `./stickey-main/src/main/resources/application-db.yml`

```

---
spring:
  config:
    activate:
      on-profile: db-local
  datasource:
    driver-class-name: ${local.db.driver}
    url: ${local.db.url}
    username: ${local.db.username}
    password: ${local.db.password}
  data:
    redis:
      host: localhost
      port: ${server.redis.port}
      username:
      password:
---
spring:
  config:
    activate:
      on-profile: db-server

```

```

datasource:
  driver-class-name: com.mysql.cj.jdbc.Driver
  url: ${server.db.url}
  username: ${server.db.username}
  password: ${server.db.password}
data:
  redis:
    host: ${server.redis.host}
    port: ${server.redis.port}
    username: ${server.redis.username}
    password: ${server.redis.password}

```

▼ application-key.yml

위치 `./stickey-main/src/main/resources/application-key.yml`

```

cloud: #aws
  aws:
    s3:
      bucket: stickey
      credentials:
        access-key: # s3 access-key
        secret-key: # s3 secret-key
      region:
        static: ap-northeast-2
    stack:
      auto: false # ec2의 spring cloud 자동 구성 해제
  cloudfront:
    domain: {domain}.cloudfront.net # cloudfront 고유
도메인

mail:
  username: stickey
  password: {password} # e-mail 인증 받을 서비스 이메일 비밀
번호

jwt:
  secret: stickey2024olbld211securitykeyfor... # jwt tok
en secret

```



```
local:
  db:
    driver: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/Stickey?allowPublic
    KeyRetrieval=true&useSSL=false&serverTimezone=Asia/Seoul
    &characterEncoding=UTF-8
    username: root
    password: '{password}'

server:
  db:
    driver: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://j10d211.p.ssafy.io:33066/STICKEY?u
    seSSL=false&serverTimezone=Asia/Seoul&characterEncoding=
    UTF-8
    username: d211user
    password: '{password}'
  redis:
    host: j10d211.p.ssafy.io
    port: 6379
    username: default
    password: '{password}'
```

3. 외부 서비스

— MetaMask 확장 프로그램

확장 프로그램

전체 액세스

이 사이트의 정보를 확인하고 변경할 수 있는 확장 프로그램입니다



MetaMask



필요한 액세스 권한 없음

이 사이트의 정보를 확인하고 변경할 필요가 없는 확장 프로그램입니다.



백준허브(BaekjoonHub)



확장 프로그램 관리

- 확장 프로그램 설치 및 로그인
- Stickey 네트워크 설정
- 지갑 설정

⇒ 자세한 설정 방법은 '서비스 이용 > 사전 준비' 참고.

네트워크 선택



You can drag networks to reorder them.



Ethereum Mainnet



Linea Mainnet

테스트 네트워크 보기



Linea Goerli



Linea Sepolia



stickey

+ 네트워크 추가

빌드 및 배포

1. 준비하기

Clone repository

```
git clone https://lab.ssafy.com/s10-blockchain-contract-sub
2/S10P22D211.git
cd ./S10P22D211
pwd
# /home/opendocs/jenkins
```

Certifications

```
sudo apt-get install letsencrypt

sudo letsencrypt certonly --standalone -d [www제외한 도메인 이름]
# sudo letsencrypt certonly --standalone -d j10d211.p.ssafy.io
```

Prepare docker

```
sudo apt update
sudo apt upgrade

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

Jenkins

```
docker pull jenkins/jenkins:jdk21

sudo mkdir /home/opendocs/jenkins

sudo docker run
--name jenkins
-d
-p 9000:8080
-v /home/opendocs/jenkins:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock
```

```
-u root
jenkins/jenkins:jdk21
```

MySQL - docker

```
docker pull mysql
```

— MySQL docker-compose.yml

Jenkins credentials에 '설정 파일 및 환경 변수' 항목을 참고하여 .env 파일 등록을 먼저 수행한다.

```
mysql:
  image: mysql:8.0.36
  container_name: stickey_mysql
  restart: unless-stopped
  networks:
    - stickey_network
  ports:
    - "${MYSQL_EXTERNAL_PORT}:${MYSQL_PORT}"
  volumes:
    - "mysql_data:/var/lib/mysql"
  environment:
    MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
    MYSQL_USER: "${MYSQL_USER}"
    MYSQL_PASSWORD: "${MYSQL_PASSWORD}"
    MYSQL_DATABASE: "${MYSQL_DATABASE}"
  command: --character-set-server=utf8mb4 --collation-se
rver=utf8mb4_bin
```

— MySQL docker build

```
# /home/opendocs/jenkins/workspace/Backend/deploy
docker compose up -d mysql
```

Redis - docker

```
docker pull redis
```

— Redis docker-compose.yml

```
redis:
  image: redis:7.2
  container_name: stickey_redis
  ports:
    - "6379:6379"
  networks:
    - stickey_network
  volumes:
    - /home/ubuntu/workspace/redis/redis.conf:/etc/redis/redis.conf
    - /home/ubuntu/workspace/redis/backup-redis:/data
  deploy:
    resources:
      limits:
        memory: 512M
    command: redis-server --requirepass ${REDIS_PASSWORD} -maxmemory 512M
```

— Redis docker build

```
# /home/opendocs/jenkins/workspace/Backend/deploy
docker compose up -d redis
```

AWS CloudFront

— CloudFront 배포 생성

STEP1)

CloudFront - 원본 액세스 - 제어 설정 (OAC) 생성 - 원본 유형을 S3로 설정하고 생성

STEP2)

CloudFront - 배포 생성

- Origin domain : AWS S3 버킷 선택
- Origin Access(원본 액세스 제어) : STEP1에서 생성한 OAC 선택
- 뷰어 프로토콜 정책 : HTTP를 HTTPS로 리디렉션

STEP3)

AWS S3 버킷 정책에 CloudFront의 ARN 추가

— AWS S3

IAM 계정 생성

STEP1) 사용자 이름 입력

STEP2) 권한 정책으로 AmazonS3FullAccess 선택하여 S3에 대한 모든 권한 소유

STEP3) 사용자 상세 - 보안 자격 증명 - 액세스 키 발급

※ 액세스 키는 다시 조회할 수 없으므로 반드시 저장하여 관리한다

— S3 Bucket 생성

Step1) AWS 리전 및 버킷 이름을 입력 (리전에 관계없이 고유한 이름 사용)

Step2)

객체 소유권 설정 : 내 AWS 계정만 버킷에 접근하거나 사용할 수 있도록 설정한다

Step3) 퍼블릭 액세스 차단 설정

- 퍼블릭 액세스는 ACL, 버킷 정책, 액세스 지점 정책을 통해 버킷 및 객체에 부여된다
- 액세스 차단을 활성화 하면 자원을 안전하게 보호할 수 있지만, 외부 접근이 불가능하다
- 버킷에 저장된 파일에 접근하지 못하는 현상을 방지하고자, 모두 해제하고 진행한다

Step4) 기본 암호화 : Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화

Step5) 프로젝트 등록 : 백엔드 프로젝트 환경변수 파일에 해당하는 값을 추가한다

— 버킷 정책

버킷 상세 - 권한 - 버킷 정책

```
{
  "Version": "2012-10-17",
  "Id": "Policy1705987522711",
  "Statement": [
    {
      "Sid": "Stmt1705987520867",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "{S3 Bucket ARN}/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "{CloudFront ARN}"
        }
      }
    }
  ]
}
```

Nginx

— Frontend

Stickey-frontend 내부 Dockerfile

위치 : `./Stickey-frontend/Dockerfile`

```

FROM node:lts-alpine as build-stage
# homepage -> app
WORKDIR /app

USER root

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:stable-alpine as production-stage
COPY --from=build-stage ./app/dist /usr/share/nginx/html/
CMD ["nginx", "-g", "daemon off;"]

```

- Jenkins에서 node plugin 설치 및 20.11.0 버전 설정

Backend

stickey-backend 내부 Dockerfile

위치 : `./stickey-backend/Dockerfile`

```

FROM openjdk:21

WORKDIR /app

COPY build/libs/*.jar back_application.jar

EXPOSE 9090

CMD ["java", "-Duser.timezone=Asia/Seoul", "-jar", "-Dspring.profiles.active=server", "back_application.jar"]

```


Ganache

```
# ganache 이미지 태그 지정하여 가져오기
docker build -t ganache .
```

```
# ganache 도커 컴포즈 실행
docker compose up -d ganache
```

stickey-blockchain 내부 Dockerfile

```
# node:alpine will be our base image to create this image
FROM node:alpine
# Set the /app directory as working directory
USER root
WORKDIR /app
# Install ganache-cli globally
EXPOSE 8545
RUN npm install -g ganache-cli
# 가나슈 네트워크 설정
CMD ["ganache-cli", "-h", "0.0.0.0", "-a", "30"]
```

2. 빌드 및 배포하기

Jenkins credentials에 '설정 파일 및 환경 변수' 항목을 참고하여 프로젝트 환경 설정파일들의 등록을 먼저 수행한다.

A. Frontend

Jenkins Pipeline

```
pipeline {
    agent any

    stages {
```

```

        stage('Clone') {
            steps {
                echo "Clone stage"
                git branch: 'FE',
                credentialsId: 'stickey',
                url: 'https://lab.ssafy.com/s10-blockchain-
contract-sub2/S10P22D211.git'
            }
        }
        stage('Deploy') {
            steps {
                withCredentials([file(credentialsId: 'env.l
ocal', variable: 'env')]) {
                    script {
                        sh 'pwd'
                        sh 'cp $env Stickey-frontend/.env.l
ocal'
                    }
                }

                dir("deploy"){
                    sh 'docker compose build --no-cache fro
ntend'

                    sh 'docker compose up -d frontend'
                }
            }
        }
    }
}

```

B. Backend

Jenkins Pipeline

```

pipeline {
    agent any

```

```

stages {
    stage('Clone') {
        steps {
            echo "Clone stage"
            git branch: 'BE',
            credentialsId: 'stickey',
            url: 'https://lab.ssafy.com/s10-blockchain-
contract-sub2/S10P22D211.git'
        }
    }
    stage('Build Backend') {
        steps{
            sh 'echo build start'
            sh 'echo get credentail files...'
            withCredentials([file(credentialsId: 'appli
cation-key', variable: 'appKey'),
                file(credentialsId: 'env', variable: 'e
nv')])) {
                script {
                    sh 'cp $appKey stickey-main/src/mai
n/resources/application-key.yml'
                    sh 'cp $appKey stickey-waiting/src/
main/resources/application-key.yml'
                    sh 'cp $env deploy/.env'
                    sh 'chown jenkins:jenkins stickey-m
ain/src/main/resources/application-key.yml'
                    sh 'chown jenkins:jenkins stickey-w
aiting/src/main/resources/application-key.yml'
                    sh 'chmod 644 stickey-main/src/mai
n/resources/application-key.yml'
                    sh 'chmod 644 stickey-waiting/src/m
ain/resources/application-key.yml'
                }
            }
            sh '''
                chmod +x ./stickey-main/gradlew
                chmod +x ./stickey-waiting/gradlew
            '''
        }
    }
}

```

```

        ./stickey-main/gradlew -p ./stickey-main clean bootJar
        ./stickey-waiting/gradlew -p ./stickey-waiting clean bootJar
        echo "Create deploy-api, deploy-waiting image"
        docker build -t deploy-api ./stickey-main
        docker build -t deploy-waiting ./stickey-waiting
    ''
}
}
stage('Deploy') {
    steps{
        dir('deploy') {
            sh 'echo "Deploy API server"'
            sh 'docker compose up -d redis'
            sh 'docker compose up -d waiting'
            sh 'docker compose up -d api'
        }
    }
}
}
}

```

배포 시 특이사항

- 메타모스트 연동 파이프 라인

```

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.B

```

```

        BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
    )
  }
}
failure {
  script {
    def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
    def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
    mattermostSend (color: 'danger',
    message: "빌드 실패: ${env.JOB_NAME} #${env.B
UILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
    )
  }
}
}
}
}

```

— 최종 소스 코드 빌드 및 배포 파이프라인

```

pipeline {
  agent any

  stages {
    stage('Clone') {
      steps {
        echo "Clone develop stage"
        git branch: 'develop',
        credentialsId: 'stickey',
        url: 'https://lab.ssafy.com/s10-blockchain-
contract-sub2/S10P22D211.git'
      }
    }
    stage('Build Backend') {

```

```

        steps{
            sh 'echo build start'
            sh 'echo get credentail files...'
            withCredentials([file(credentialsId: 'application-key', variable: 'appKey'),
                           file(credentialsId: 'env', variable: 'env')])) {
                script {
                    sh 'cp $appKey stickey-main/src/main/resources/application-key.yml'
                    sh 'cp $appKey stickey-waiting/src/main/resources/application-key.yml'
                    sh 'cp $env deploy/.env'
                    sh 'chown jenkins:jenkins stickey-main/src/main/resources/application-key.yml'
                    sh 'chown jenkins:jenkins stickey-waiting/src/main/resources/application-key.yml'
                    sh 'chmod 644 stickey-main/src/main/resources/application-key.yml'
                    sh 'chmod 644 stickey-waiting/src/main/resources/application-key.yml'
                }
            }
            sh '''
                chmod +x ./stickey-main/gradlew
                chmod +x ./stickey-waiting/gradlew
                ./stickey-main/gradlew -p ./stickey-main clean bootJar
                ./stickey-waiting/gradlew -p ./stickey-waiting clean bootJar
                echo "Create deploy-api, deploy-waiting image"
                docker build -t deploy-api ./stickey-main
                docker build -t deploy-waiting ./stickey-waiting
            '''
        }
    }
}

```

```

    }
    stage('Deploy') {
        steps{
            withCredentials([file(credentialsId: 'env.local', variable: 'env')]) {
                script {
                    sh 'pwd'
                    sh 'cp $env Stickey-frontend/.env.local'

                }
            }

            dir('deploy') {
                sh 'echo "Deploy API server"'
                sh 'docker compose up -d redis'
                sh 'docker compose up -d waiting'
                sh 'docker compose up -d api'

                sh 'echo "Deploy FE"'
                sh 'docker compose build --no-cache frontend'

                sh 'docker compose up -d frontend'
            }
        }
    }
}

post {
    success {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
            mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",

```

```

        )
    }
}
failure {
    script {
        def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
        mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.B
UILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
        )
    }
}
}
}
}

```

— 최종 docker-compose.yml 파일

```

version: '3.8'

services:
  redis:
    image: redis:7.2
    container_name: stickey_redis
    ports:
      - "6379:6379"
    networks:
      - stickey_network
    volumes:
      - /home/ubuntu/workspace/redis/redis.conf:/etc/redis/
redis.conf
      - /home/ubuntu/workspace/redis/backup-redis:/data
    deploy:
      resources:

```



```

    limits:
      memory: 512M
    command: redis-server --requirepass ${REDIS_PASSWORD} -
-maxmemory 512M

api:
  build:
    context: "../stickey-main/"
    dockerfile: Dockerfile
  container_name: stickey_main
  ports:
    - "9090:9090"
  depends_on:
    - mysql
    - redis
  networks:
    - stickey_network

waiting:
  build:
    context: "../stickey-waiting/"
    dockerfile: Dockerfile
  container_name: stickey_waiting
  ports:
    - "9091:9091"
  depends_on:
    - redis
  networks:
    - stickey_network

frontend:
  build:
    context: "../Stickey-frontend/"
    dockerfile: Dockerfile
  container_name: Stickey_frontend
  ports:
    - "80:80"
    - "443:443"

```

```

    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
      - /var/lib/letsencrypt:/var/lib/letsencrypt
      - /home/ubuntu/nginx/sites/conf.d:/etc/nginx/conf.d
      - /home/ubuntu/nginx/sites/sites-enabled:/etc/nginx/s
ites-enabled
    networks:
      - stickey_network

mysql:
  image: mysql:8.0.36
  container_name: stickey_mysql
  restart: unless-stopped
  networks:
    - stickey_network
  ports:
    - "${MYSQL_EXTERNAL_PORT}:${MYSQL_PORT}"
  volumes:
    - "mysql_data:/var/lib/mysql"
  environment:
    MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
    MYSQL_USER: "${MYSQL_USER}"
    MYSQL_PASSWORD: "${MYSQL_PASSWORD}"
    MYSQL_DATABASE: "${MYSQL_DATABASE}"
  command: --character-set-server=utf8mb4 --collation-se
rver=utf8mb4_bin

volumes:
  mysql_data:
  redis-data:

networks:
  stickey_network:
    name: stickey_network
    driver: bridge
    external: true

```

C. Blockchain

Dockerfile이 수행하여 빌드 및 배포된다. (직접도 가능)

```
# openzeppelin 라이브러리 다운로드 필요
npm install

# 배포시 생성된 ApplicationHandler의 CA로 env 설정
truffle migrate --reset
```

결과

1_initial_migration.js

=====

Deploying 'Reword'

```
> transaction hash: 0x04d6e4752e8d191f8ea104202360eeb187bbc471d2a5c20fb3d61d57d8e217cb
> Blocks: 0        Seconds: 0
> contract address: 0xA4f79ebc31EE0f652ea1DF9EDFD0D61bD0C64a74
> block number:    71
> block timestamp: 1712188840
> account:         0x28355495464B95Bf0f48Da220F8002FCc66F7c26
> balance:         99.493444863040573806
> gas used:        1031625 (0xfbd9c9)
> gas price:       2.553110532 gwei
> value sent:      0 ETH
> total cost:      0.0026338526525745 ETH
```

Deploying 'ApplicationHandler'

```
> Blocks: 0        Seconds: 0
> contract address: 0x527B80bD4532A1e722F445013201B838A3f96eB0
> block number:    72
> block timestamp: 1712188841
> account:         0x28355495464B95Bf0f48Da220F8002FCc66F7c26
> balance:         99.48001993203940277
> gas used:        5267758 (0x50612e)
> gas price:       2.548509442 gwei
> value sent:      0 ETH
> total cost:      0.013424931001171036 ETH
```

> Saving artifacts

```
> Total cost:      0.016058783653745536 ETH
```

Summary

=====

```
> Total deployments: 2
> Final cost:        0.016058783653745536 ETH
```

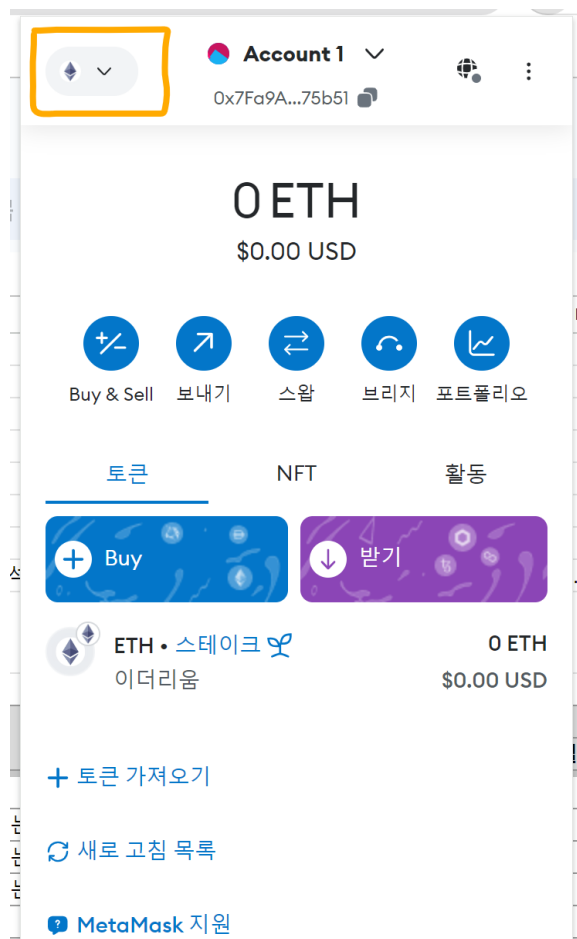
서비스 이용

1. 사전 준비

- 메타마스크 어플 혹은 크롬 확장 프로그램 설치
- 메타마스크 로그인 및 지갑 설정
- Stickey 네트워크 설정 및 접속
- 지갑 가져오기

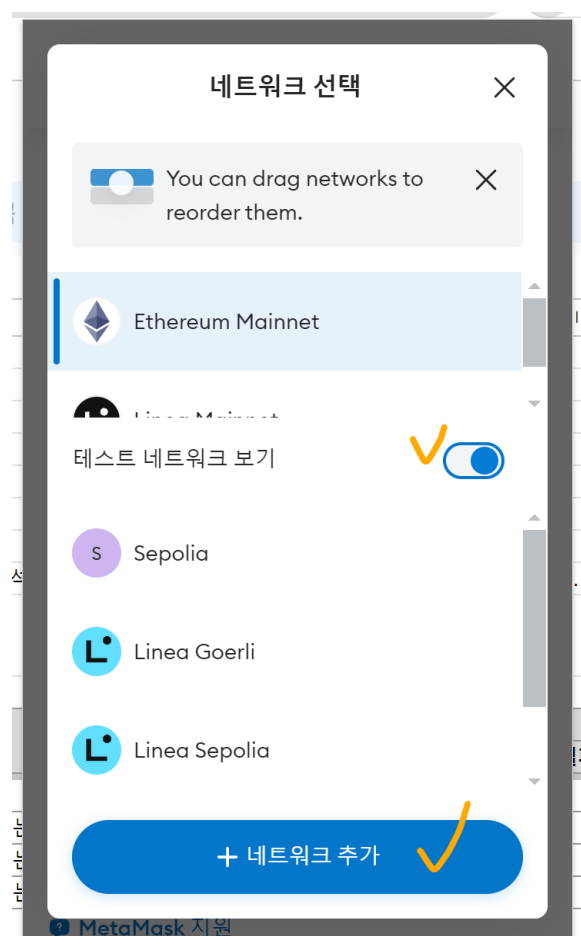
메타마스크 로그인 방법

1. 왼쪽 상단 토글 선택



왼쪽 상단 토글 선택

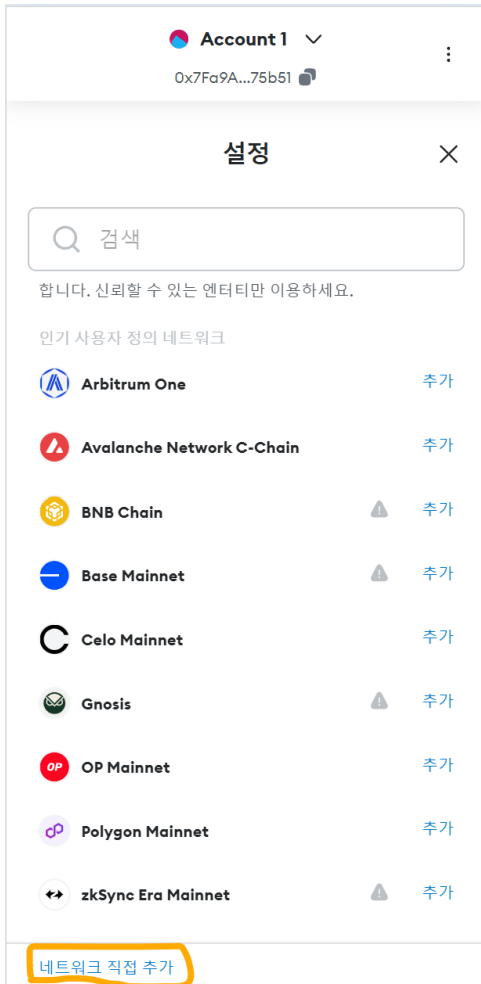
2. 테스트 네트워크 활성화 및 네트워크 추가 버튼 클릭



테스트 네트워크 보기 활성화

3. 네트워크 직접 추가 선택

4. 네트워크 정보 설정



네트워크 직접 추가

네트워크 > 네트워크 추가 > 네트워크 직접 추가

i 악성 네트워크 공급업체는 블록체인 상태를 거짓으로 보고하고 네트워크 활동을 기록할 수 있습니다. 신뢰하는 맞춤 네트워크만 추가하세요.

네트워크 이름

새 RPC URL

이 URL은 현재 HTTPS 가나슈 네트워크에서 사용됩니다.

체인 ID

통화 기호

Suggested ticker symbol: ETH

블록 탐색기 URL (옵션)

취소
저장

네트워크 > 네트워크 추가 > 네트워크 직접 추가

i 악성 네트워크 공급업체는 블록체인 상태를 거짓으로 보고하고 네트워크 활동을 기록할 수 있습니다. 신뢰하는 맞춤 네트워크만 추가하세요.

네트워크 이름

새 RPC URL

이 URL은 현재 HTTPS 가나슈 네트워크에서 사용됩니다.

체인 ID

통화 기호

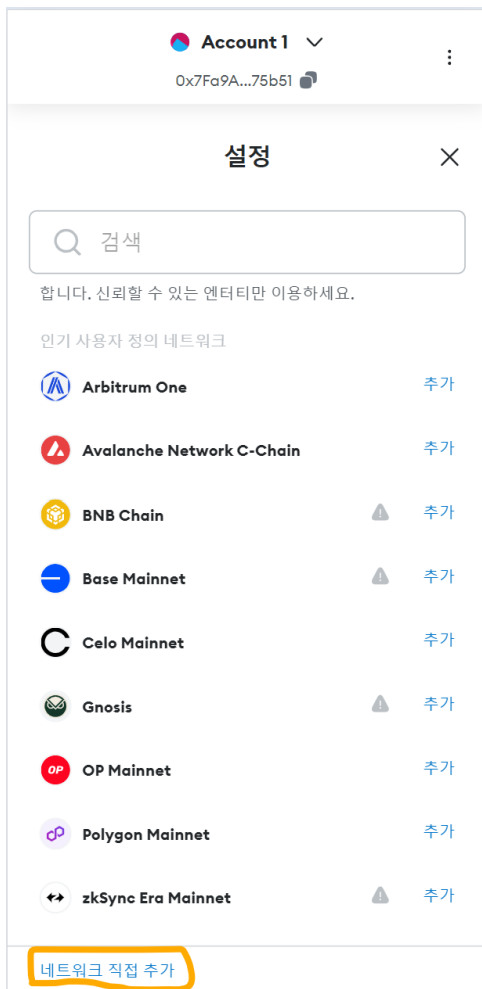
Suggested ticker symbol: ETH

블록 탐색기 URL (옵션)

취소
저장

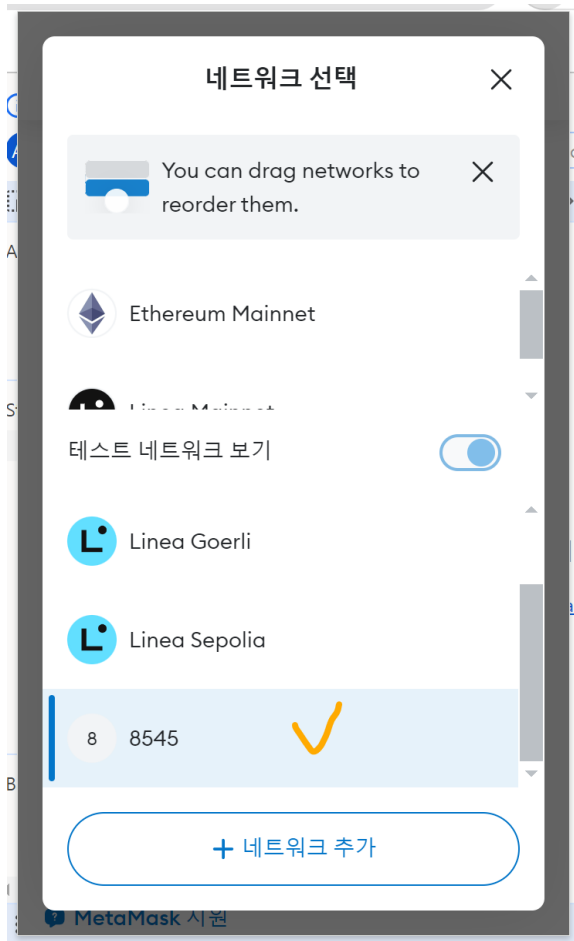
https://j10d211.p.ssafy.io/block

1337

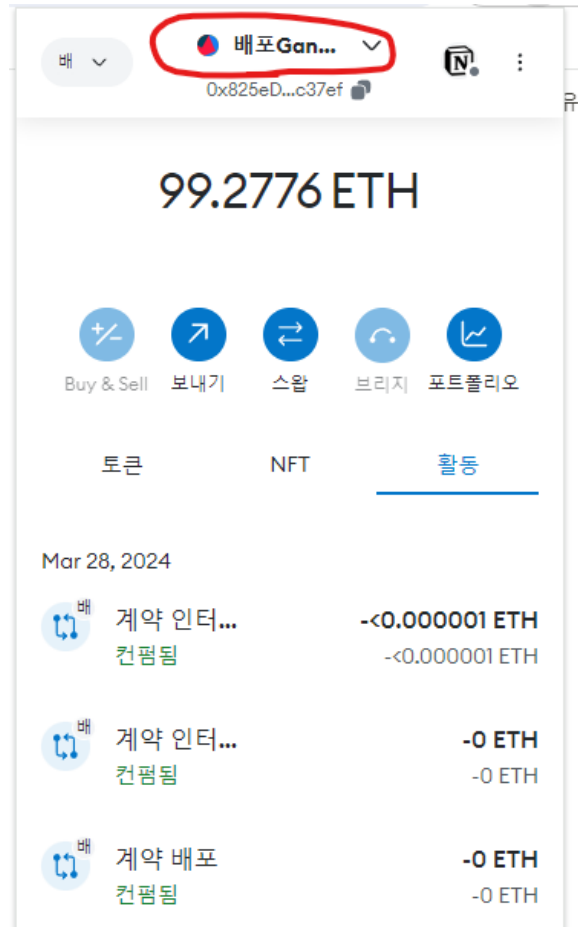


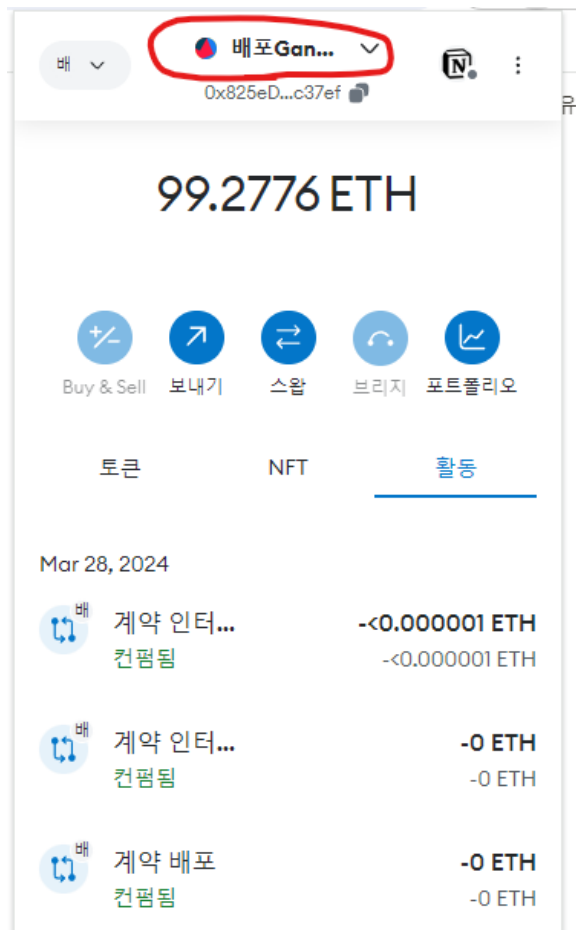
5. 추가된 네트워크 선택

6. 위쪽의 지갑 선택



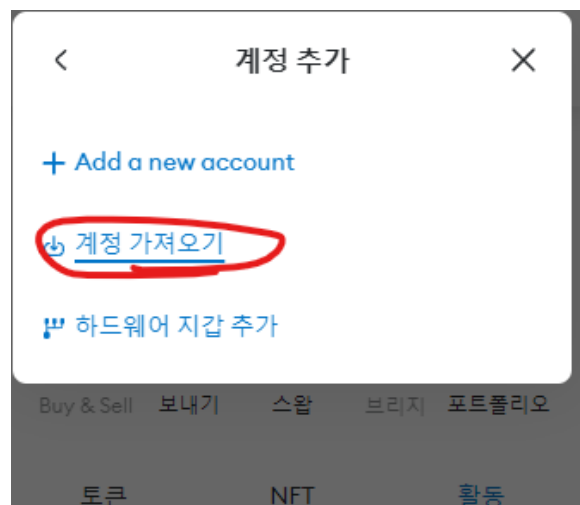
추가된 네트워크 선택

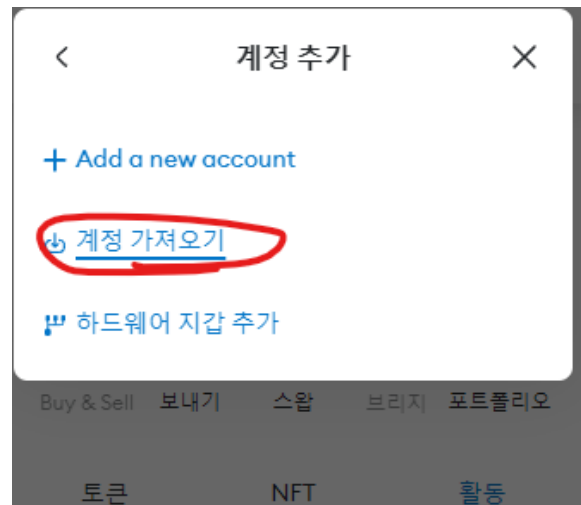
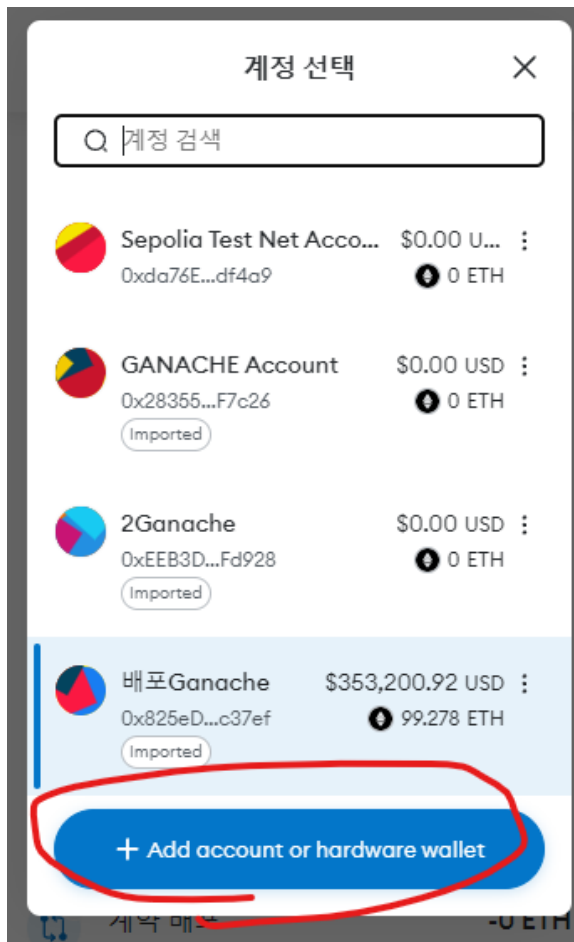


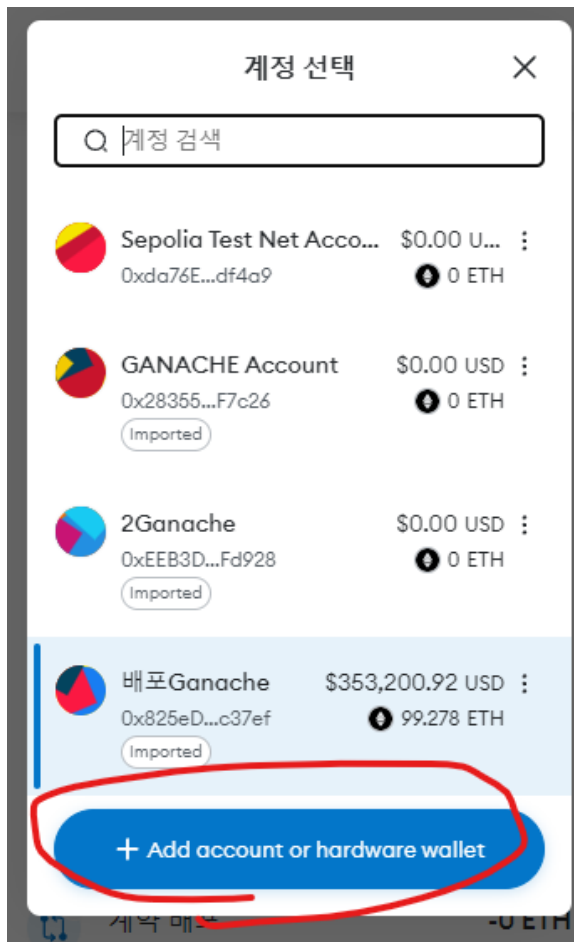


7. 아래의 지갑 추가 선택

8. 계정 가져오기 선택







9. 오른쪽의 비공개 키 중 하나 입력

Private Keys

=====

- (1) 0xf0049991425577e13f2a02
- (2) 0xf5ffb3c534cd51f7c6a30e
- (3) 0x6b42a2ad3d93a7733ad8f7
- (4) 0x5366da7822dece8417e072
- (5) 0x9438f6f97b5f9d25309c9b
- (6) 0x048525cf73fe39026eacab
- (7) 0xab5ce5771511acfbcec5dc
- (8) 0x9069997fc1f385939fce49
- (9) 0x03cc4d68ceb7a17764d2a2
- (10) 0x785236097d900ebe181eb
- (11) 0xd12a7bf08a8eae88cd03a
- (12) 0x2fe344c5730bf66bc1041
- (13) 0x856d5d9f837ce4b13b1b2
- (14) 0xcdc91dcd12cee1378ee0c
- (15) 0x82e759ee442850ed2c9c8

<

계정 가져오기

×

가져온 계정은 MetaMask 계정 비밀번호구문과 연결하지 못합니다. 가져온 계정에 대해 자세히 알아보기 [여기](#)

유형 선택

비공개 키 ▾

여기에 비공개 키 문자열을 붙여넣으세요.

취소

가져오기

배

계약 인터...

컨펌됨

-<0.000001 ETH

-<0.000001 ETH

배

계약 인터...

컨펌됨

-0 ETH

-0 ETH

```
(16) 0x69a88d69d729d0539f09e
(17) 0xf6158e6d8b818f756cc36
(18) 0xca8af96d74cd425ff233d
(19) 0x3265ac39ccd1a7220c77d
(20) 0xd4352d17479230cb771da
(21) 0x491d5925e01db1145c320
(22) 0x0b83c6d4ff71e63c88cb4
(23) 0x9bf87a8ac0e0956c24402
(24) 0x8febbc0fdd2034397fdd6
(25) 0x10bb36184350a9c93b214
(26) 0x5b36c16dd5ee8f3414262
(27) 0x61f70035aad4f17875a9
(28) 0x3f06e853f0c38b38b3f39
(29) 0x3e89aad056ca49d40d1cb
```

<

계정 가져오기

×

가져온 계정은 MetaMask 계정 비밀번호구문과 연결하지 못합니다. 가져온 계정에 대해 자세히 알아보기 [여기](#)

유형 선택

비공개 키 ▾

여기에 비공개 키 문자열을 붙여넣으세요.

취소

가져오기

배

계약 인터...

컨펌됨

-<0.000001 ETH

-<0.000001 ETH

배

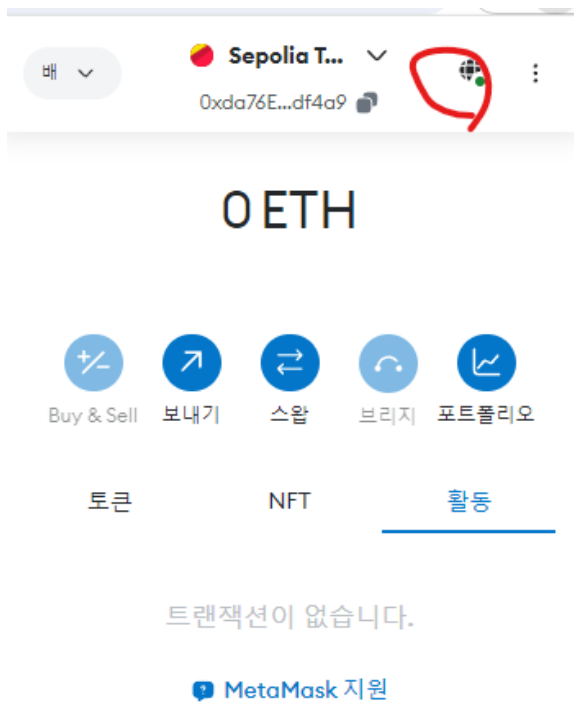
계약 인터...

컨펌됨

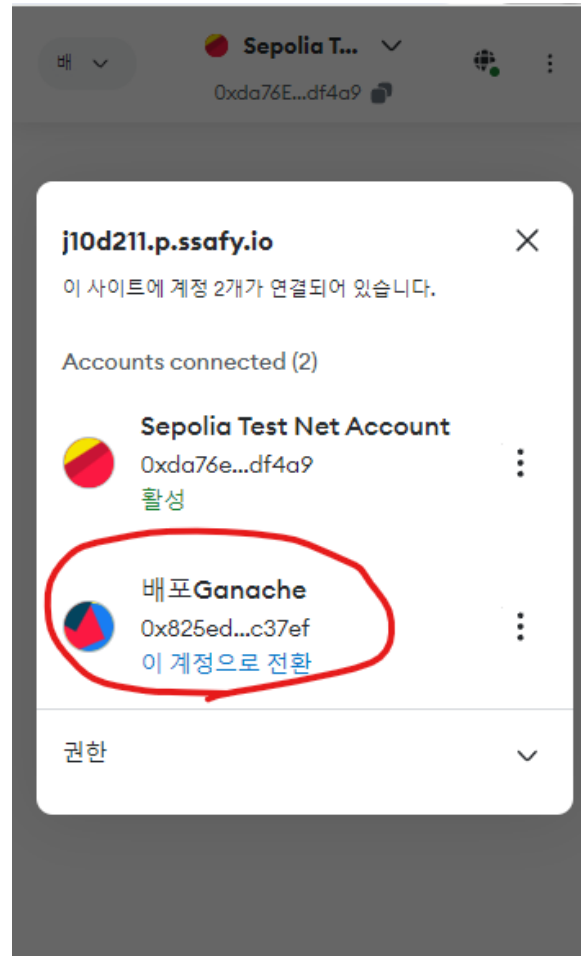
-0 ETH

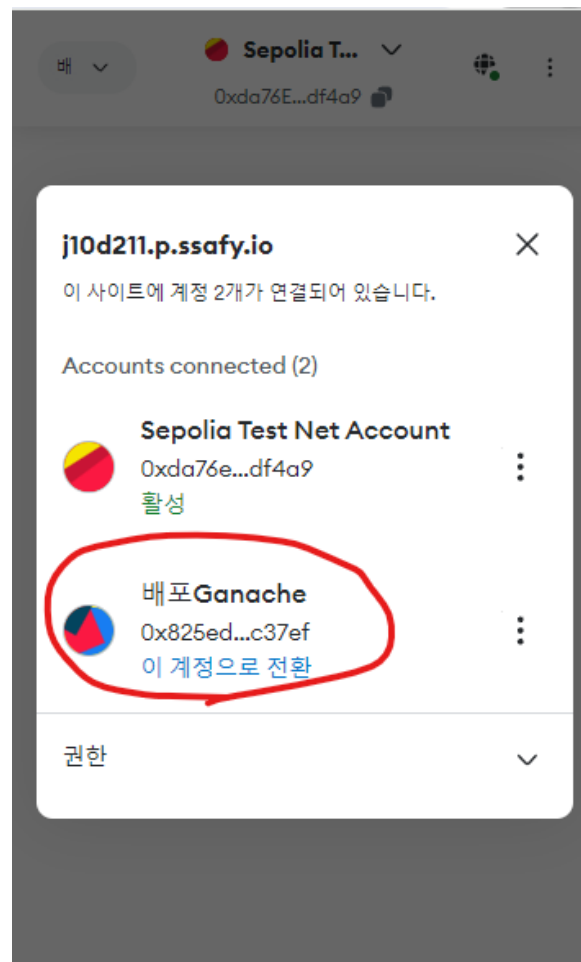
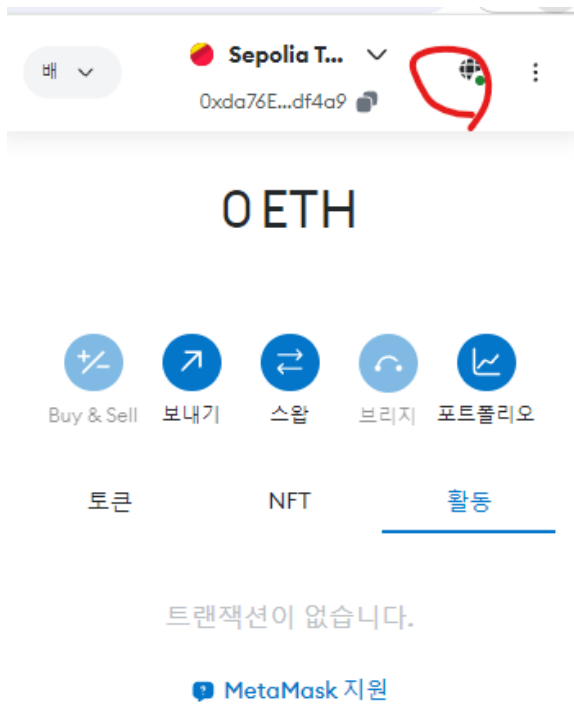
-0 ETH

10. 계정 연결 (만약 다른 지갑과 연결했다면)

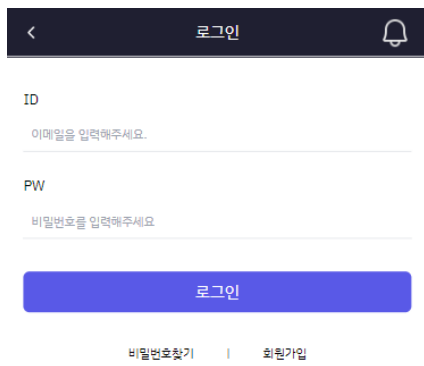


11. 추가했던 지갑으로 연결





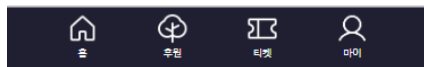
2. 시연 시나리오

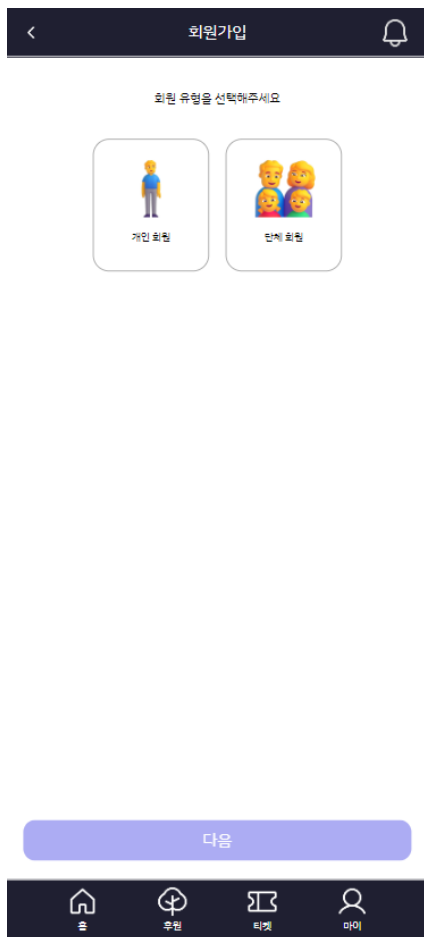


The image shows a mobile app login screen. At the top is a dark blue header bar with a back arrow on the left, the word '로그인' (Login) in the center, and a bell icon on the right. Below the header, the text 'ID' is followed by a light blue input field with the placeholder text '이메일을 입력해주세요.' (Please enter your email). Below that, the text 'PW' is followed by another light blue input field with the placeholder text '비밀번호를 입력해주세요.' (Please enter your password). A large blue button with the text '로그인' (Login) is positioned below the password field. At the bottom of the login section, there are two links: '비밀번호찾기' (Find password) and '회원가입' (Sign up), separated by a vertical line.

로그인 화면

- ID, 비밀번호를 통해 로그인





회원가입 화면

- 개인 회원과 단체 회원 선택
- 개인 회원과 단체 회원 공통 폼 작성
 - 이메일 인증
- 단체 회원 선택시 단체 회원 폼 작성

단체회원가입

담당자이름

담당자명을 입력해주세요

주소

주소 검색

사업자번호

000-0000-0000

사업자등록증

파일 선택

선택된 파일 없음

가입하기



홈



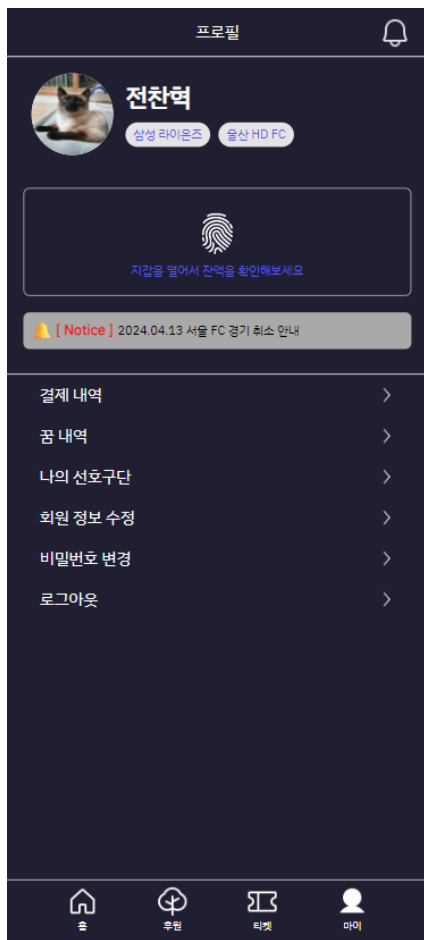
회원



티켓

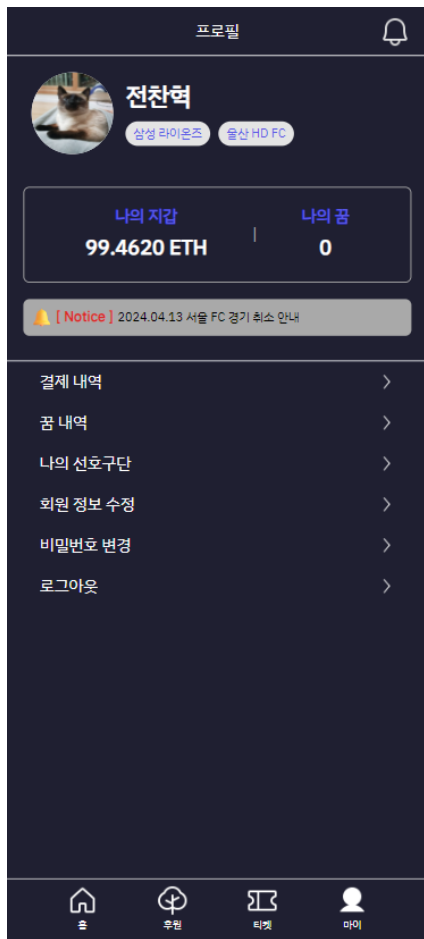


마이



회원정보 화면

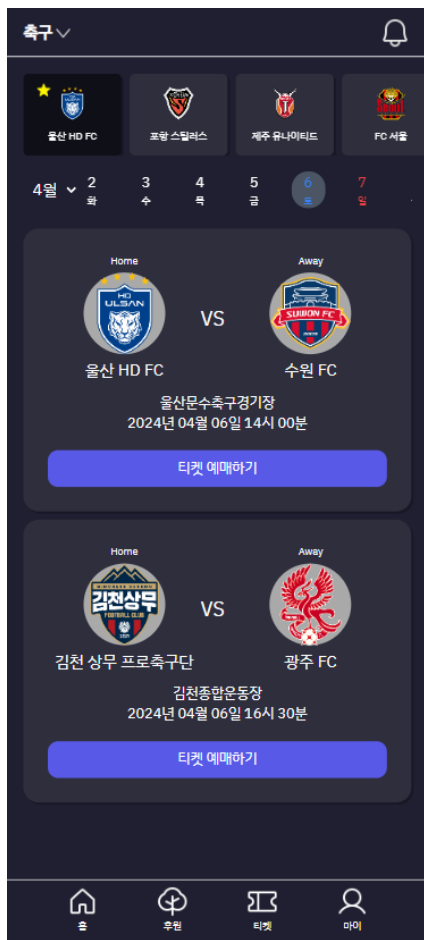
- 등록된 회원 정보 조회
- 지갑 조회
 - 서비스와 연동된 메타마스크에서 정보를 읽어 지갑 잔액 조회





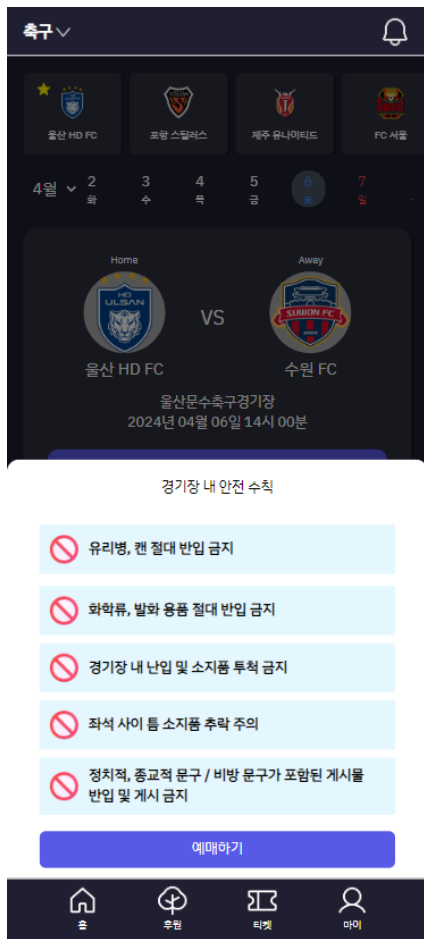
메인 화면

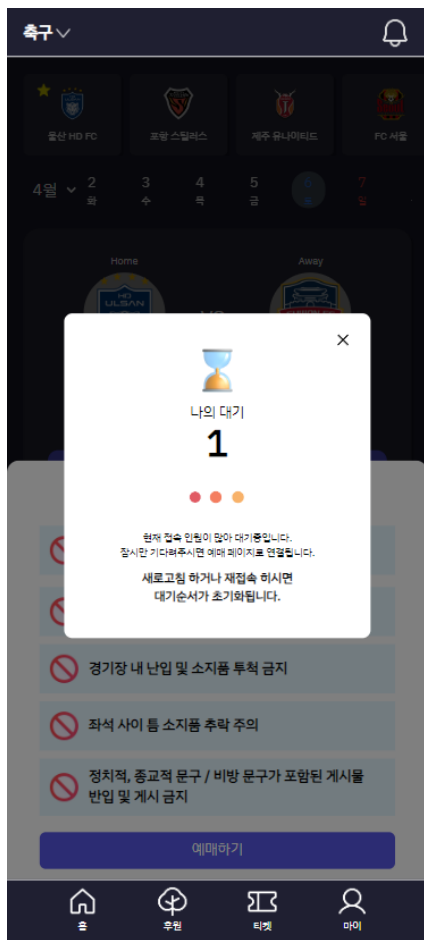
- 종목 선택을 통해 종목 별 경기 조회 가능
- 로그인시 등록된 선호 구단의 경기 일정 조회



경기 조회 화면

- 해당 날에 경기 조회
- 구단과 날짜로 필터링
- 티켓 예매 버튼 클릭 시 안전 수칙 모달에 화면에 표시
- 안전 수칙 화면에서 예매하기 버튼 클릭 시 예매 화면으로 전환





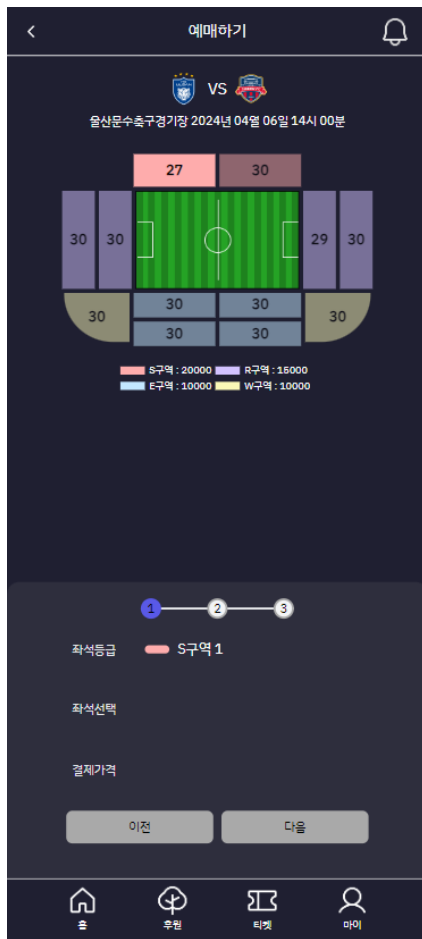
예매 - 대기열 화면

- 현재 경기에 예매 중인 사람 정보를 가져와 대기열 표시
- 본인의 차례가 되면 구역 선택 화면으로 이동

화면 캡처

예매 - 구역 선택 화면

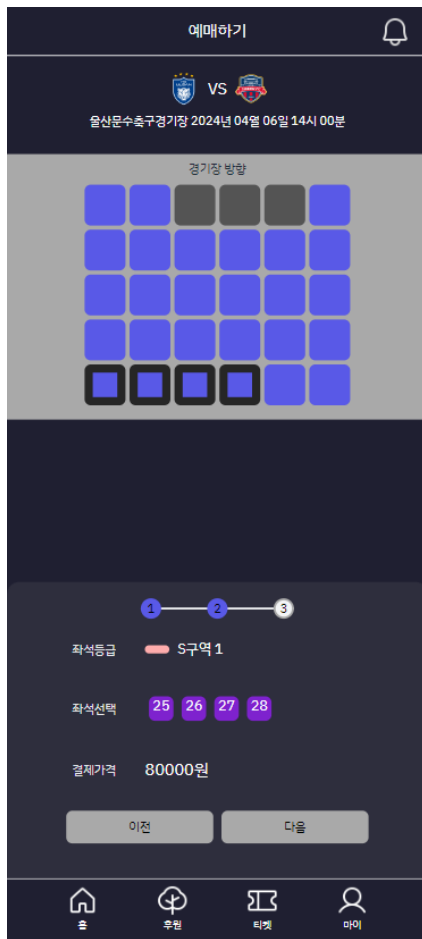
- 해당 경기에 구역 선택
- 경기 정보 화면에 표시
- 구역 별 남은 좌석 조회
 - 남은 좌석이 있다면 선택 가능
 - 선택한 구역 정보 저장
- 이전 버튼 클릭 시 경기 목록으로 이동
- 다음 버튼 클릭 시 좌석 선택 화면으로 이동

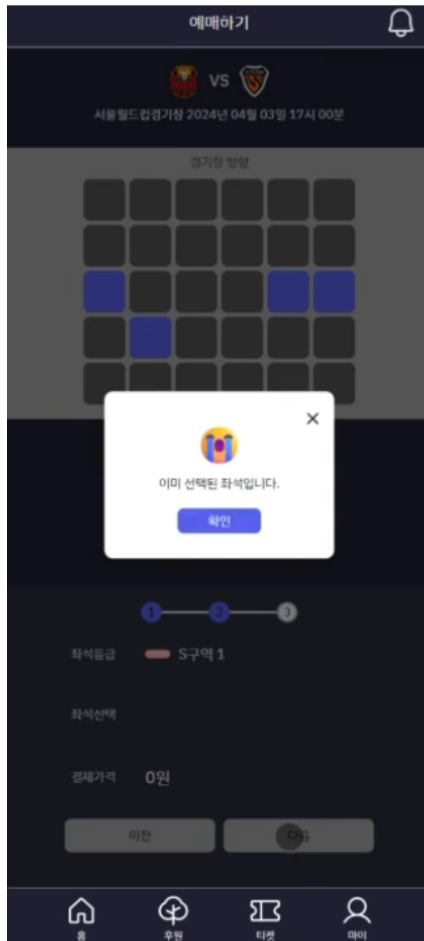


화면 캡처

예매 - 좌석 선택 화면

- 해당 구역의 현재 좌석의 상태를 조회
- 선택한 좌석의 정보를 저장하고 화면에 표시
- 이전 버튼 클릭 시 구역 선택 화면으로 이동
- 다음 버튼 클릭 시 결제 화면으로 이동
 - 이미 선택된 좌석이라면 좌석 정보 재조회

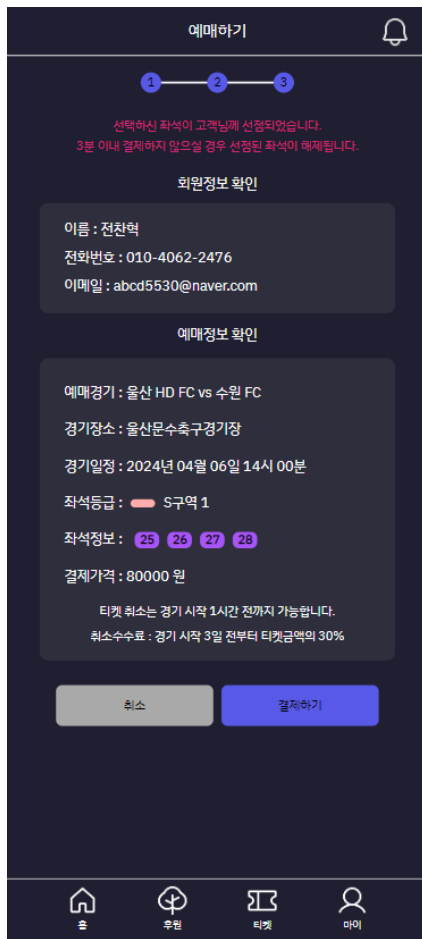


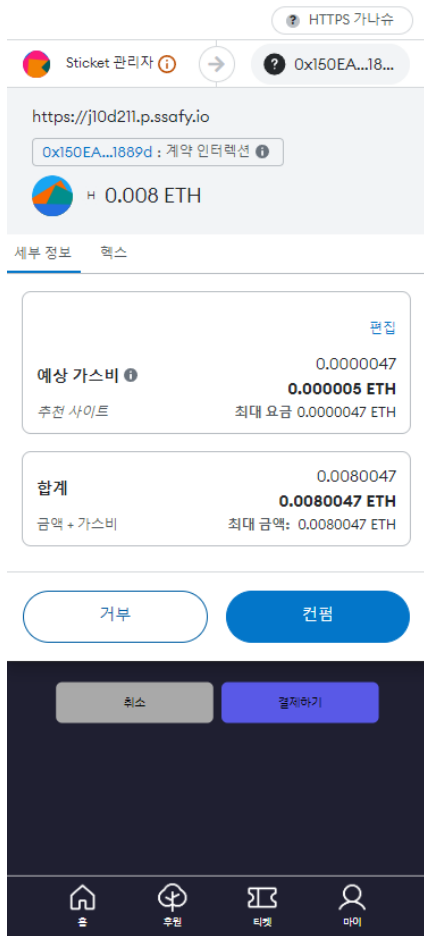


화면 캡처

예매 - 결제 화면

- 현재 회원 정보 표시
- 사용자가 선택한 예매 정보 표시
- 취소 선택 시 구역 선택 화면으로
- 다음 선택시 메타마스크 결제
 - 결제 실패시 구역 선택





화면 캡처

[] 화면

•

화면 캡처

[] 화면

•

화면 캡처

[] 화면

•