

공익활동의 기념비적 NFT 발행 서비스



박예서

정태영

황진하

지도교수 권동현

목 차

1. 서론.....	1
1.1. 과제 배경.....	1
1.2. 과제 목표.....	2
2. 설계.....	2
2.1. 개발 환경.....	2
2.2. 시스템 구성도.....	3
2.3. 페이지 구성도.....	3
3. 수행 내용.....	4
3.1. Front-end.....	4
3.2. Back-end.....	8
4. 결과물.....	15
4.1. 회원가입.....	15
4.2. 공지사항.....	19
4.3. 마이페이지.....	21
5. 기타.....	26
5.1. 개발 일정.....	26
5.2. 구성원별 역할.....	27

1. 서론

1.1. 과제 배경

자원봉사활동은 사회 또는 공공의 이익을 위한 일을 자기 의지로 행하는 것을 말한다. 자원봉사활동은 봉사자에게는 보람이나 경험 등 정신적 보상을 얻고, 피봉사자는 도움을 받을 수 있는, Win-Win 구조인 활동이다. 그러나 우리나라에서는 이런 자원봉사활동의 참여도와 관심이 떨어지고 있는 추세이다.

구분	실인원			연인원	1인당 참여횟수 (건)	비고
	인원	증가인원	증가율(%)			
2011년	1,743,394	-308,018	▼17.7	13,767,850	7.90	
2012년	2,163,174	419,780	24.1	18,652,383	8.62	
2013년	2,642,529	479,355	22.2	21,394,555	8.09	
2014년	3,174,876	532,347	20.1	22,648,601	7.13	
2015년	3,746,577	571,701	18.0	24,822,008	6.63	
2016년	4,590,079	843,502	22.5	30,030,295	6.54	
2017년	4,876,669	286,590	6.2	31,382,487	6.44	
2018년	4,290,985	-585,684	▼12.0	30,619,226	7.14	
2019년	4,191,548	-99,437	▼2.3	29,129,700	6.95	
2020년	2,233,767	-1,957,781	▼46.7	13,994,115	6.26	

[그림 1] 연도별 자원봉사자 활동 현황

2021년 자원봉사센터 현황 [그림 1]의 표를 보면 우리나라의 자원봉사자 실인원은 2018년부터 꾸준히 떨어지고 있다. 특히 2020년은 코로나19 바이러스로 인해 급격히 감소한 모습을 보인다.

또한 우리나라 총 인구수 대비 자원봉사자 인원의 비율은 10%도 되지 않는 모습도 볼 수 있는데, 그만큼 많은 사람들이 자원봉사활동에 대한 관심과 실천이 부족함을 알 수 있다.

이런 문제를 해결하기 위해, 사람들에게 동기 부여가 될 수 있도록 자원봉사활동을 몇 가지 카테고리로 나눠 랭킹과 진행도를 눈으로 보여주고, 업적을 달성했을 경우 NFT를 발행해주는 웹사이트를 제작하였다.

1.2. 과제 목표

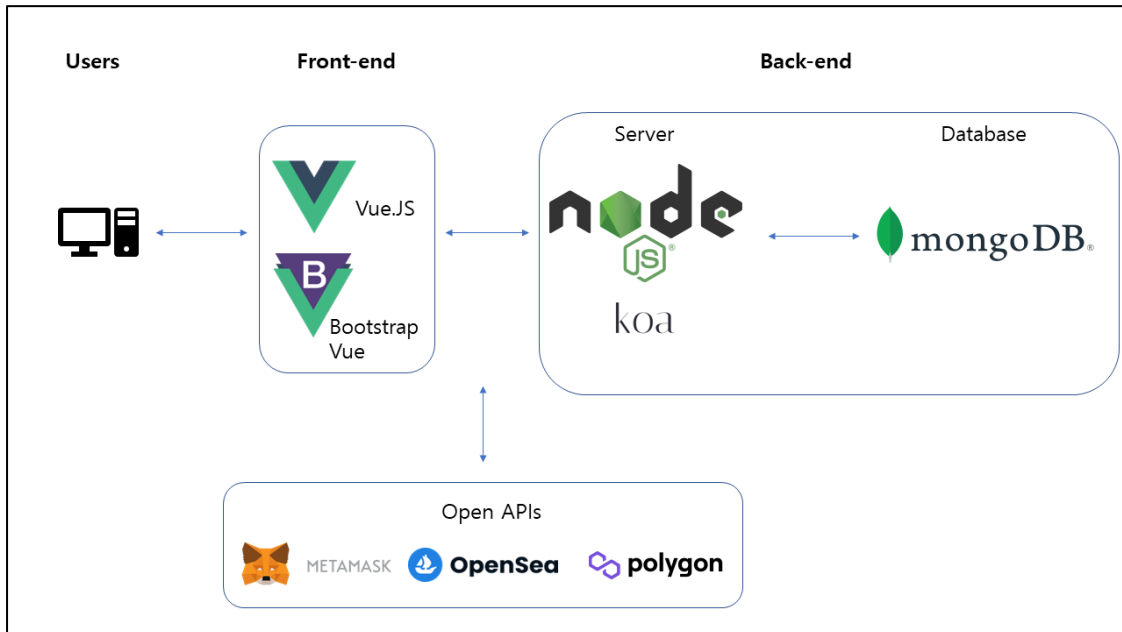
- 웹사이트 형태의 UI 제작
- Front-end와 Back-end 간 데이터 이동과 연동
- 데이터베이스 스키마 설계 및 Back-end와 연동
- 로그인, 로그아웃 토큰 발급 로직 구현
- 이미지 업로드, 이미지 합성 등의 로직 구현
- 페이지 별 라우터 정의 및 연결
- NFT 관련 API 연동
- 웹사이트 부가 기능 개발

2. 설계

2.1. 개발 환경

구분	사용도구
사용 언어	HTML
	CSS
	JavaScript
개발 도구	Visual Studio Code
	Brackets
런타임 환경	Node.js v16.15.1
데이터베이스	MongoDB 5.0.8
프레임워크	Koa v2.13.4
	Vue.js v2.7.8
	BootstrapVue v2.22.0
패키지 관리자	Node Package Manager (npm) v8.11.0
	Yarn v1.22.19
Open API	Metamask API
	Opensea API
	Polygon API

2.2. 시스템 구성도

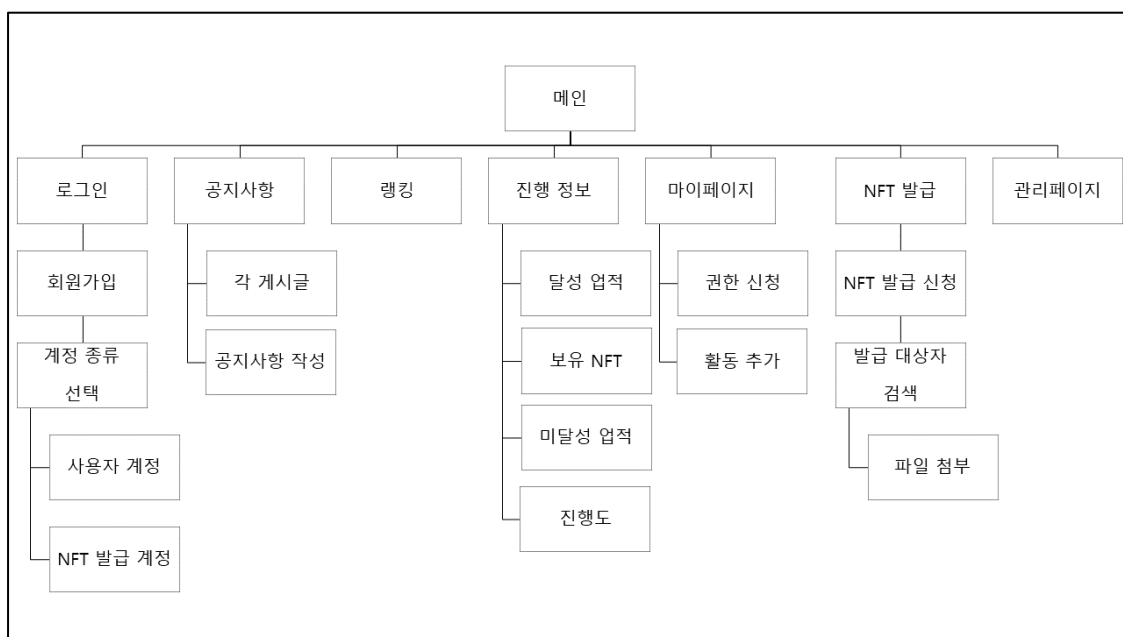


Front-end에서는 SPA이면서 양방향 데이터 바인딩이 가능한 VueJS를 선택했다.

Back-end에서는 NodeJS 기반에서 Express보다 가볍고 async/await를 편하게 사용 가능한 Koa 프레임워크를 선택했다.

데이터베이스로는 데이터 스키마 추가가 편리한, NOSQL에 속하는 MongoDB를 택했다.

2.3. 페이지 구성도



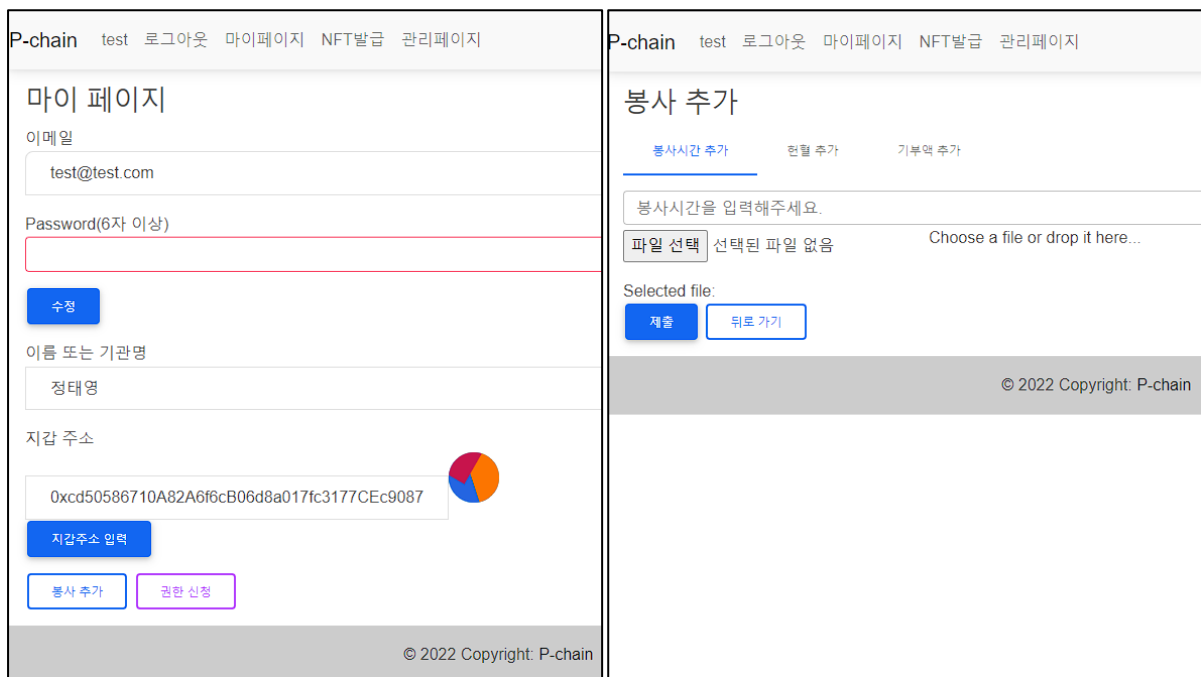
3. 수행 내용

3.1. Front-end

3.1.1. 페이지 구성

```
1 <template>
2   <div id="app">
3     <nav-bar></nav-bar>
4     <router-view></router-view>
5     <main-footer></main-footer>
6   </div>
7 </template>
```

Vue의 특징인 SPA의 장점이 드러나는 코드로, 최상위 컴포넌트인 App.vue의 구조가 Menu Bar - Page - Footer로 되어 있다. 페이지 이동 시에 Menu Bar와 Footer는 바뀌지 않고 router-view 컴포넌트만 재구성되어 불필요한 로딩을 막는다.



[그림 2] 마이페이지, 회원가입 페이지 비교

[그림 2]는 제작한 웹페이지의 마이 페이지와 봉사 추가 페이지이다. 페이지 전환 시에 Menu Bar와 Footer가 바뀌지 않는 모습이다.






3.1.2. 동적 라우팅

SPA의 단점은 최초 로딩 속도 문제가 있다. 웹사이트에 처음 접근했을 때 모든 리소스를 다운받기 때문에 그 순간 페이지 로딩이 느리다는 문제이다. 이를 해결하기 위한 방법으로 동적 라우팅과 chunk를 사용했다.



[그림 3] 일반 라우팅(위 그림)과 동적 라우팅(아래 그림) 코드

[그림 3]은 일반 라우팅과 동적 라우팅 코드의 차이점을 보여준다. 일반 라우팅을 했을 경우 웹사이트에 처음 접근하면 해당 리소스를 모두 다운받지만, 동적 라우팅은 해당 페이지로 이동하거나 같은 chunkName으로 묶인 페이지로 이동할 경우 동적으로 리소스를 다운받는다.

 build.js	304	script	(색인)	204 B	24밀리...
 client	200	script	(색인)	(디스...	4밀리...
 axios.min.js	200	script	axios.min.js	(디스...	1밀리...
 0.build.js	304	script	build.js:762	203 B	315...
 6.build.js	304	script	build.js:762	203 B	3밀리...

[그림 4] chunk 파일 실행 예시

[그림 4]는 Google Chrome 개발자 모드의 네트워크 탭에서 웹사이트 페이지 이동 시 build.js 이외에 0.build.js, 6.build.js 등 chunk 파일을 추가적으로 다운받았음을 보여준다.

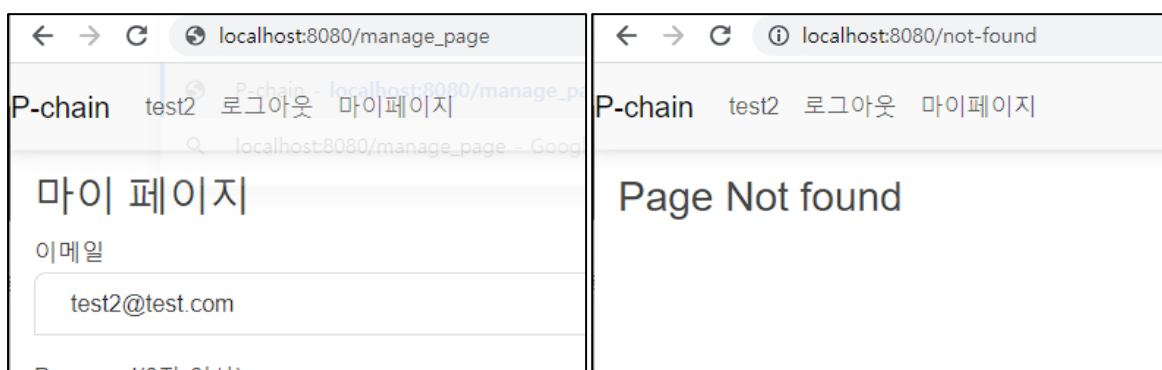
3.1.3. 네비게이션 가드

웹사이트의 각 페이지들은 관리자 페이지와 같이 접근 권한이 필요한 페이지가 존재한다. 일반적으로는 계정 권한에 따라 화면에 페이지 링크를 숨기거나 보이게 한다. 그러나 페이지 주소를 알고 있는 경우 주소창에 해당 페이지 주소를 입력해 접근하는 문제가 발생할 수 있다. 이를 막기 위한 방법이 네비게이션 가드이다.

```
1 router.beforeEach((to, from, next) => {
2   // authenticationState는 유저가 로그인되어있는지 아닌지 값을 가져와 판별해준다.
3   const authenticationState = store.state.token // false: not login, true: login
4   const authenticationAccess = store.state.access // 0: user/client, 1: issuer, 2: admin
5   // authorization에서는 라우터에서 메타 속성을 정의해준 값이 담겨진다.
6   // [], ["client"], ["issuer"], ["user"], ["admin"]
7   const { authorization } = to.meta
8
9   if (authorization !== "all") {
10    if (authorization === "admin" && authenticationAccess >= 2) next()
11    else if (authorization === "issuer" && authenticationAccess >= 1) next()
12    else if (authorization === "user" && authenticationAccess >= 0 && authenticationState) next()
13    else if (authorization === "client" && !authenticationState) next()
14    else { next('/not-found'); return; }
15  }
16  else next()
17 });
```

[그림 5] 네비게이션 가드 적용 코드

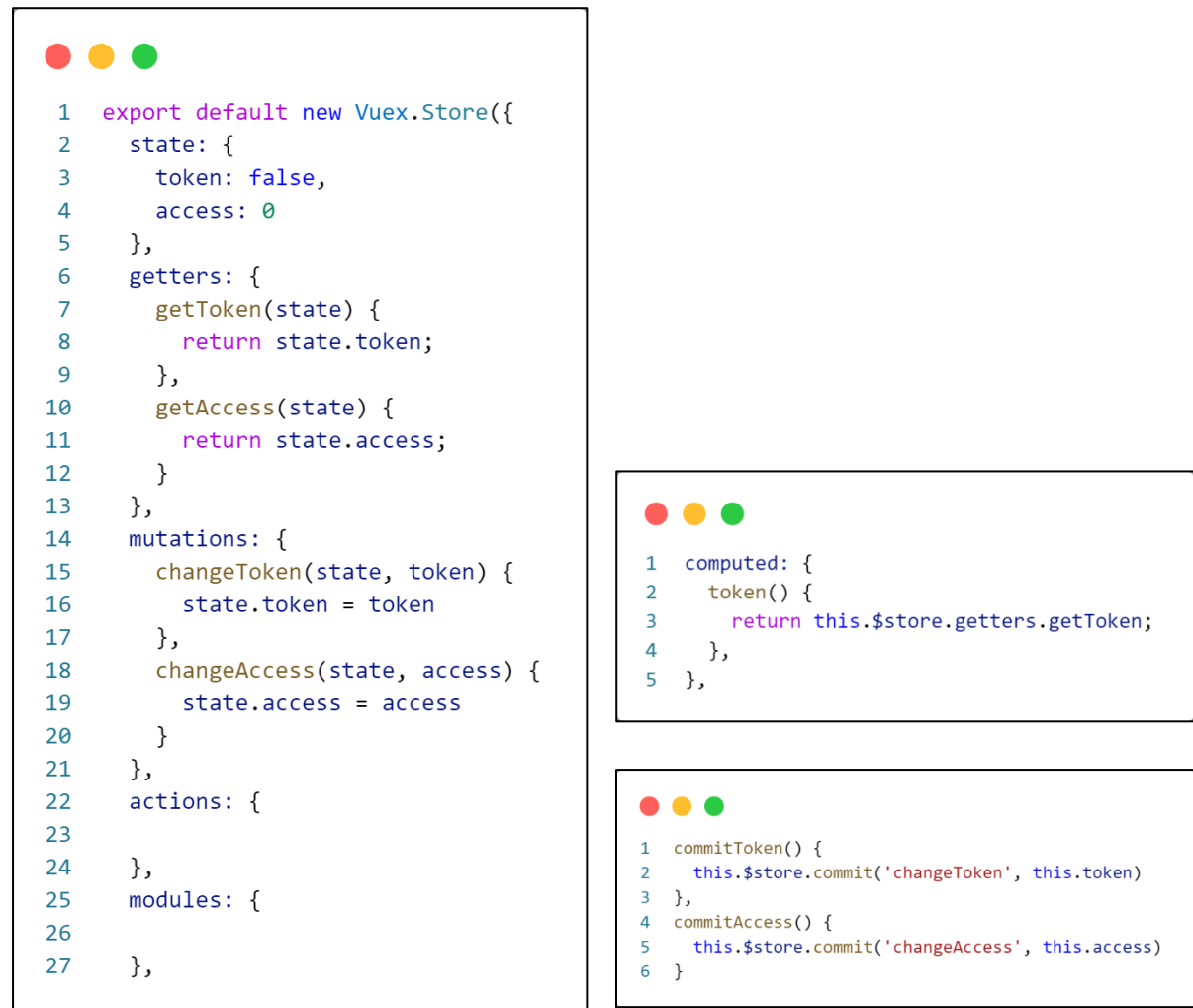
[그림 5]는 제작한 웹사이트에 적용되고 있는 네비게이션 가드 코드이다. 라우터에서는 페이지가 로딩되기 전 beforeEach를 호출한다. 이 함수 내에서, 먼저 로그인 여부와 계정 권한을 vuex에서 가져온다. 그 후 [그림 3]에서 정의한 meta 속성의 authorization 값을 가져온 뒤 조건문을 거쳐 next() 혹은 next('/not-found')가 호출된다. Next()가 호출되면 페이지 접근이 승인된 것이고, next('/not-found')가 호출되면 승인되지 않고 not-found 페이지를 호출한다. [그림 6]은 웹페이지에서의 네비게이션 가드 적용 예를 보여준다.



[그림 6] 네비게이션 가드 적용 예

3.1.4. Vuex

Vuex는 웹페이지와 관련된 데이터를 관리해주는 라이브러리이다. 기본적으로 Vue에서는 컴포넌트 간 데이터 공유 시 prop이나 EventBus를 사용한다. 그러나 prop은 부모-자식 컴포넌트 간 데이터 이동 밖에 하지 못해, 많은 컴포넌트가 있을 경우 코드가 난해해지는 문제가 생긴다. EventBus의 경우에는 컴포넌트 간 데이터 이동은 자유롭지만 데이터 흐름이 복잡해져 코드의 유지보수가 어렵다는 문제가 있다.



[그림 7] Vuex 사용 예제

[그림 7]의 왼쪽 그림은 Vuex 인스턴스를 생성하고 로그인과 관련된 정보를 state 속성에 저장한 모습이다. State 내부에는 로그인과 관련된 데이터인 token과 access가 정의되어 있다. 이 state 객체는 단일 상태 트리이며, 데이터를 읽는 건 가능하지만, 변경하기 위해서는 getters나 mutations에서 method를 이용한 접근을 해야 한다. [그림 7]의 오른쪽 그림은 각각 getters와 mutations에 정의된 method를 통해 접근한 모습이다.

3.2. Back-end

3.2.1. Rest api

백엔드 서버는 node.js의 koa를 이용하여 구성하였다. Rest api를 구현하였고, 프론트엔드와의 통신시, 포트가 달라서 발생하는 CORS문제를 해결하기 위해서, proxy서버를 사용하였다.

```
proxy: {  
  "/api": {  
    target: 'http://localhost:4000'  
  },  
},
```

[그림 8] proxy 등록부분

3.2.2. 파일 업로드

사용자의 증빙서류 업로드를 위해서, koa의 multer모듈을 사용해서 파일 업로드를 구현하였다.

```
router.post('/api/upload', upload.single('file'), (ctx, next)=>{  
  console.log("ok");  
  console.log(ctx.request.file);  
  const { fieldname, originalname, encoding, mimetype, destination, filename, path, size } = ctx.request.file  
  const { name } = ctx.request.body;  
  
  console.log("body 데이터 : ", name);  
  console.log("폼에 정의된 필드명 : ", fieldname);  
  console.log("사용자가 업로드한 파일 명 : ", originalname);  
  console.log("파일의 인코딩 타입 : ", encoding);  
  console.log("파일의 Mime 타입 : ", mimetype);  
  console.log("파일이 저장된 폴더 : ", destination);  
  console.log("destination에 저장된 파일 명 : ", filename);  
  console.log("업로드된 파일의 전체 경로 ", path);  
  console.log("파일의 바이트(byte 사이즈)", size);  
  
  ctx.body = {ok: true, data: "Single Upload Ok"}  
})
```

[그림 9] 파일 업로드 구현부분

3.2.3. DB연동

몽고DB를 사용해서 구현하기위해, 프론트엔드 -> api호출 -> 백엔드 라우터 -> 백엔드 핸들러(함수) -> 몽고DB메소드 -> 몽고DB 변수의 단계를 사용하였다. 예를 들어 봉사시간 랭킹 구현을 보면,

```

created(){
  axios.get("/api/ranking/vol").then(response =>{
    console.log(response.data);
    var arr = response.data;
    for(var i in arr){
      arr[i].index = Number(i)+1;
    }
    this.items = arr;
  })
}

```

[그림 10] 봉사시간 랭킹 프론트

프론트엔드에서, 봉사시간 랭킹을 불러오는 api를 호출하면,

```

const ranking = new Router();
const rank = require('./ranking')

const handler = (ctx, next) => {
  ctx.body = `${ctx.request.method}`
};

ranking.get('/achieve', rank.achieveRank);
ranking.get('/vol', rank.volRankInfo);

```

[그림 11] 봉사시간 랭킹 백엔드 라우터

백엔드 라우터에서, 해당 api 함수(rank.volRankInfo)를 호출해주고,

```

exports.volRankInfo = async (ctx) => {
  // 봉사시간 랭킹 불러오기
  var data = await APL.printVolRank();
  ctx.response.body = data;
};

```

[그림 12] 봉사시간 랭킹 백엔드 핸들러

핸들러 함수에서, 몽고DB메소드를 호출해준다.

```

AchieveProgressLev.statics.printVolRank = function(){
  // 전체 DB 봉사시간랭킹순으로 불러오기
  return this.find({"volTime": {$gt: 0}}, {"nickname":true, "volTime":true}).sort({volTime:-1}).exec();
};

```

[그림 13] 봉사시간랭킹 DB메소드

호출된 DB 메소드에서, DB변수 수정, 문서 불러오기, sort등의 작업을 수행해준다.

3.2.4. KOA.JS

Koa.js는 가장 인기있는 웹 프레임워크인 express.js 개발팀이 만든 새로운 웹 프레임워크이다. Express.js보다 훨씬 가볍고, Node.js v7의 async/await기능을 사용하기 쉽다는 장점이 있다. 관련 부분만 코드를 가져오면 다음과 같다.

```
const Koa = require('koa');
const Router = require('koa-router');

const app = new Koa();
const router = new Router();

...
const port = process.env.PORT || 4000; // 4000 if not defined

/* link part */
const auth_account = require('./auth_account');
const auth_apply = require('./auth_apply');
~
const meta = require('./meta');
const DB_test = require('./DB_Test');
...
app.proxy = true

app.use(bodyParser()); // have to be upward of router
app.use(jwtMiddleware); // apply middleware

...
router.use('/api/auth_account', auth_account.routes());
router.use('/api/auth_apply', auth_apply.routes());
~
router.use('/api/meta', meta.routes());
router.use('/DB_test', DB_test.routes());

app.use(router.routes()).use(router.allowedMethods());
app.listen(port, () => {
  console.log('test server is listening to port ' + port);
})
```

Koa.js는 미들웨어의 배열로 구성되어 있다. DnI 코드에서 사용한 app.use라는 함수가 미들웨어를 어플리케이션에 등록해 주는 것이다. Link part에서 모듈들이 저장된 path를 가져와, 손쉽게 router.use()를 하여 라우터를 추가할 수 있다. Router에 get이나, post 등의 메소드를 이용해 index.js파일에서 간단하게 라우팅 구성도 할 수 있다.

3.2.5. JWT 토큰

유저가 회원 인증을 하게 될 때 토큰을 발급해줌으로서 유저가 자기 자신임을 인증 할 수 있게 해주는 것이다. 발급을 하게 되면, 토큰의 유효기간, 정보등을 가지며, 해싱 알고리즘을 통해 인증이 되어있어서 서버에서 검증을 통하여 처음 서버가 발급해주었던 정보가 변조되지 않았음을 보장해 줄 수 있는 것이다. 토큰은 웹 스토리지(local storage 혹은 session storage)에 넣는 방법도 있고, httpOnly 속성이 활성화된 쿠키에 정보 전달의 의미로 넣는 방식도 있다. (httpOnly는 네트워크 통신 상에서만 쿠키가 붙게해주는 옵션)

발급을 하기 위해 JWT 비밀 키를 설정해 주어야 한다.

```
const jwtSecret = process.env.JWT_SECRET;
const jwt = require('jsonwebtoken');

function generateToken(payload) {
  return new Promise(
    (resolve, reject) => {
      jwt.sign(
        payload,
        jwtSecret,
        {
          expiresIn: '7d'
        }, (error, token) => {
          if(error) reject(error);
          resolve(token);
        }
      );
    }
  );
};
```

위는 7일동안 유효하도록 토큰을 생성해주는 방법이다. JWT를 사용하기 위한 라이브러리 또한 명시되어 있다.

```
Account.methods.generateToken = function() {
  const payload = {
    _id: this._id,
    profile: this.profile,
    isIssuer: this.isIssuer,
    isManager: this.isManager,
    nickname: this.nickname,
    email: this.email,
    wallet: this.walletAddress
  };
};
```

```
return generateToken(payload, 'account');
};
```

Generate token을 다른 모듈 등에서 불러올 수 있게 Account라는 계정 관련 파일에 선언해 준 것이다.

위의 Account.generateToken()을 이용하여

```
let token = null;
try {
  token = await account.generateToken();
} catch (e) {
  ctx.throw(500, e);
}

ctx.cookies.set('access_token', token, { httpOnly: true,
maxAge: 1000 * 60 * 60 * 24 * 7 });
```

위와 같은 코드를 회원가입, 로그인 함수에 더해주어 cookie를 발급 하여 로그인을 유지 할 수 있고, 이 cookie를 null로 바꿔 줌으로써 로그아웃을 할 수 있다.

3.2.6. Metamask API

거래 플랫폼으로서의 간단한 기능으로 이더리움을 전송하는 코드가 필요했다. Src/lib/metamask.js가 그에 관한 파일이고, 프로그램 서버를 작동할 시에는 post메소드를 통해 /meta/getTC 라는 url를 이용하여 사용할 수 있다.

이 거래를 도와줄 인터페이스로 Infura를 사용하였다.

```
var Web3 = require("web3");
var Tx = require("ethereumjs-tx")
var infura_token = process.env.INFURA_TOKEN;
var private_key = process.env.MM_PRIVATE_KEY;
var node_host = `https://ropsten.infura.io/v3/${infura_token}`;
var web3 = new Web3(node_host);
```

Process.env는 프로젝트 전역에 대한 변수이다. 보안을 위해 위와 dotenv 모듈을 사용하여 처리했다. 이더리움을 다루기 위한 Web3 모듈, 이더리움 코인 트랜잭션을 만들기 위한 ethereumjs-tx 모듈 등을 사용했다.

```
exports.getTC = async (ctx) => {
  var bodyString = '';
  web3.eth.getTransactionCount(send_account, (err, txCount) => { // (1)
    const txObject = {
      nonce: web3.utils.toHex(txCount),
      gasLimit: web3.utils.toHex(1000000),
```

```

        gasPrice: web3.utils.toHex(web3.utils.toWei("10", "gwei")),
        to: receive_account,
        value: "0x2C68AF0BB140000"
    };
    const tx = new Tx(txObject);
    tx.sign(privateKeyBuffer);

    const serializedTx = tx.serialize();
    const raw = "0x" + serializedTx.toString("hex");
    web3.eth
        .sendSignedTransaction(raw) // (2)
        .once("transactionHash", hash => {
            console.info("transactionHash",
"https://ropsten.etherscan.io/tx/" + hash);
            // tx 가 pending 되는 즉시 etherscan 에서 tx 진행상태를 보여주는
링크를 제공합니다.
        })
        .once("receipt", receipt => {
            console.info("receipt", receipt); // 터미널에 receipt 출력
        })
        .on("error", console.error);
    });
    ctx.body = bodyString;
};

```

getTransactionCount 메소드를 호출하여 현시점 transaction count를 파악한 뒤 해당값을 콜백함수의 파라미터로 넣어주었다. Transaction 오브젝트의 value 항목은 얼마만큼의 이더리움을 전송 할 것인지를 0x헥사코드로 적어야 한다. 0.2 이더리움을 보내기 위해 0x2C68AF0BB140000를 입력하였고, 해당 오브젝트를 사용하여 transaction을 만들어냈다.

3.2.7. OpenSea API

Nft를 웹상에서 볼 수 있는 기능을 제공하기 위해 OpenSea API를 사용해야 했다. OpenSea의 assets를 볼 수 있도록 제공하고 있는 방법으로 get request를 "<https://testnets-api.opensea.io/api/v1/쿼리문>" 이라는 url로 만들어 줄 수 있었다. (testnet api을 사용하여 위와 같이 쉽게 되었지만, testnet용이 아닌 때도 헤더 "X-API-KEY: [API 키]" 리퀘스트 GET -i --url "<https://api.opensea.io/api/v1/assets/쿼리문>" 을 통해 간단히 가능하다.)

위의 쿼리문에 owner를 해당 회원의 address로 설정하고, 필요에 따라 offset과 limit을 조절하여 페이지별로 보이게 할 수 있다. 만들어진 option을 이용하여 request를 보내면, body에 String으로 assets를 받을 수 있다. JSON.parse()메소드를 이용하여 json으로 파싱해주어 인덱스를 String으로 쉽게 접근할 수 있다. 이후 for문을 통하여 해당 회원의 nft

를 순회하며, if문을 통하여 발급자를 creator를 확인하여 본 웹사이트에서 발급한 것인지 확인할 수 있다.

```
request.get(options, (error, response, body) => {
  console.log("error : ", error);
  var request_body = body;
  var wallet = JSON.parse(request_body);
  wallet = wallet['assets'];

  let nft_list = [];

  for (nft in wallet){
    if (wallet[nft]["creator"] != null){
      // if (wallet[nft]["creator"]["address"] ==
      // "0xbc19ffef966bff35cb0fee54741fef4f1a33662a"){
      //   nft_list.push(JSON.parse('{ "img" : "' +
      //     wallet[nft][image_url] + '" }'));
      //   console.log("creator : ",
      //     wallet[nft]["creator"]["address"]);
      // }

      if (wallet[nft]["creator"]["address"] == admin_account){
        nft_list.push(JSON.parse('{ "img" : "' +
          wallet[nft][image_url] + '" }'));
        console.log("creator : ",
          wallet[nft]["creator"]["address"]);
      }
      // nft_list.push(JSON.parse('{ "img" : "' +
      //   wallet[nft][image_url] + '" }'));
    }
  }
  console.log(nft_list);
});
```


4. 결과물

4.1. 회원가입

P-chain 로그인

공지사항

테스트1

2022-9-30

테스트2

2022-9-30

테스트3

2022-9-30

테스트4

2022-9-30

테스트5

2022-9-30

테스트6

2022-9-30

테스트7

2022-9-30

테스트8

2022-9-30

테스트9

2022-9-30

테스트10

2022-9-30

랭킹

업적 랭킹

봉사시간 랭킹

NFT 보유 랭킹

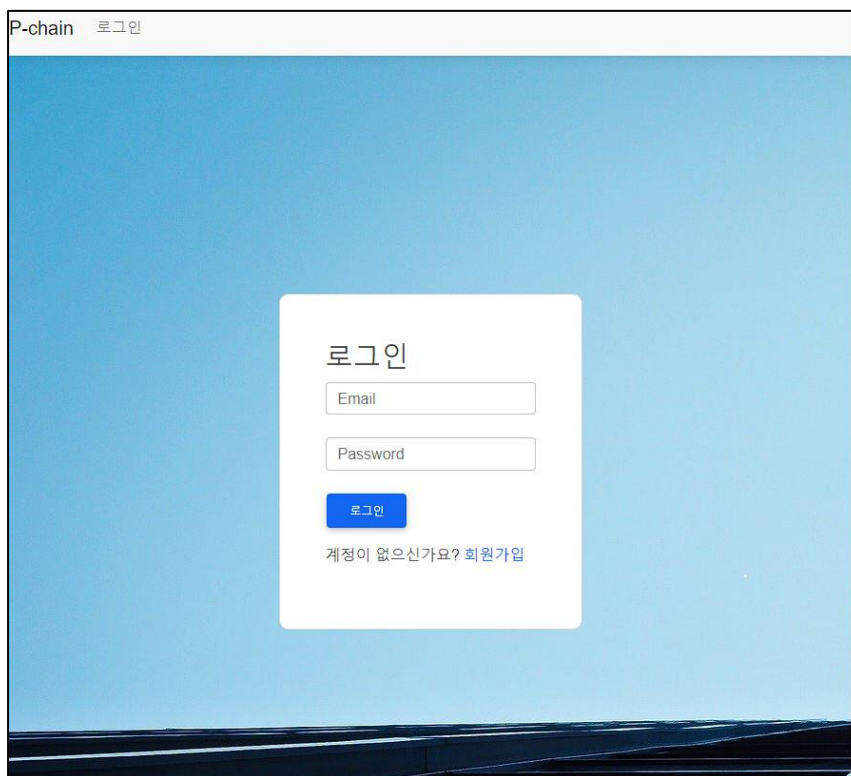
웹사이트에 접속하면 메뉴 바가 있고 공지사항, 랭킹, 달성 업적 항목을 확인할 수 있다.

달성 업적

로그인이 필요합니다

© 2022 Copyright: P-chain

달성 업적의 경우 로그인이 필요하다는 메시지가 출력된다.



상단 메뉴바의 로그인을 클릭하면 로그인 페이지로 이동한다. 계정이 있는 경우 로그인, 계정이 없는 경우 회원가입을 할 수 있다.



회원가입 페이지로 이동하면 사용자 계정과 발급 계정 중 하나를 선택할 수 있다. 사용자 계정은 일반 사용자를 위한 계정이고, 발급 계정은 봉사기관을 위한 계정이다.

P-chain 로그인

사용자 계정 회원가입

이메일

이메일을 입력해주세요. ex)example@example.com

이메일을 입력해주세요.

비밀번호

비밀번호를 입력해주세요.

6자 이상 비밀번호를 입력해주세요.

이름

이름을 입력해주세요.

이름을 입력해주세요

닉네임

닉네임을 입력해주세요.

4자 이상 15자 이하 닉네임을 입력해주세요.

제출 초기화

© 2022 Copyright: P-chain

사용자 계정 회원가입 페이지에서는 이메일, 비밀번호, 이름, 닉네임을 입력하는 양식이 있다.

P-chain 로그인

발급 계정 회원가입

대표 이메일

대표 이메일을 입력해주세요. ex)example@example.com

이메일을 입력해주세요.

비밀번호

비밀번호를 입력해주세요.

6자 이상 비밀번호를 입력해주세요.

기관명

기관명을 입력해주세요.

4자 이상 15자 이하 기관명을 입력해주세요.

제출 초기화

© 2022 Copyright: P-chain

반대로 발급 계정 회원가입 페이지에는 이메일, 비밀번호, 기관명을 입력하는 양식이 있다.

P-chain 로그인

사용자 계정 회원가입

이메일

test2@test.com

비밀번호

이름

test2

닉네임

테스트2

재출

초기화

© 2022 Copyright: P-chain

양식에 맞게 입력했을 경우, 입력 창의 테두리가 빨간색에서 초록색으로 바뀌고, 피드백 문구도 사라진다.

P-chain 테스트2 로그아웃 마이페이지

공지사항

테스트1

2022-9-30

테스트2

2022-9-30

테스트3

2022-9-30

테스트4

2022-9-30

테스트5

2022-9-30

테스트6

2022-9-30

테스트7

2022-9-30

테스트8

2022-9-30

테스트9

2022-9-30

테스트10

2022-9-30

랭킹

업적 랭킹

봉사시간 랭킹

NFT 보유 랭킹

계정을 만들고 로그인을 하면 상단 메뉴바가 바뀐 것을 확인할 수 있다. 로그인 메뉴에서 내 닉네임과 로그아웃, 마이페이지 메뉴로 바뀌었다.

4.2. 공지사항

P-chain 테스트2 로그아웃 마이페이지	
공지사항	
테스트1	2022-9-30
테스트2	2022-9-30
테스트3	2022-9-30
테스트4	2022-9-30
테스트5	2022-9-30
테스트6	2022-9-30
테스트7	2022-9-30
테스트8	2022-9-30
테스트9	2022-9-30
테스트10	2022-9-30
First Prev 1 2 Next Last	
© 2022 Copyright: P-chain	

공지사항 버튼을 누르면 공지사항 페이지로 이동한다. 각 페이지마다 공지사항이 10개씩 출력되고, 쪽수를 매기는 pagination nav가 표시된다.

P-chain test1 로그아웃 마이페이지 NFT발급 관리페이지	
공지사항	
테스트1	2022-9-30
테스트2	2022-9-30
테스트3	2022-9-30
테스트4	2022-9-30
테스트5	2022-9-30
테스트6	2022-9-30
테스트7	2022-9-30
테스트8	2022-9-30
테스트9	2022-9-30
테스트10	2022-9-30
First Prev 1 2 Next Last 	
© 2022 Copyright: P-chain	

관리자 계정으로 로그인 할 경우, 우측 하단에 공지사항을 작성할 수 있는 버튼이 추가된다. 추가적으로 상단 메뉴바도 일반 계정에 NFT발급, 관리페이지가 추가되었다.

P-chain
test1
로그아웃
마이페이지
NFT발급
관리페이지

공지사항

테스트13

공지사항 테스트

제출

© 2022 Copyright: P-chain

공지사항 작성 버튼을 누르면, 제목과 내용을 입력할 수 있는 textArea가 추가된다. 제목과 내용에 적당한 문구를 입력해 놓고 제출을 누른다.

P-chain
test1
로그아웃
마이페이지
NFT발급
관리페이지

공지사항

테스트11	2022-9-30
테스트12	2022-9-30
테스트13	2022-9-30

First
Prev
1
2
Next
Last

© 2022 Copyright: P-chain

공지사항 2페이지로 넘어가면, 제목과 작성 날짜가 추가된 것이 보인다.

P-chain
test1
로그아웃
마이페이지
NFT발급
관리페이지

공지사항

테스트13	2022-9-30
-------	-----------

공지사항 테스트

© 2022 Copyright: P-chain

해당 제목을 클릭하면, 작성한 문구들이 게시글 형태로 출력되는 것을 확인할 수 있다.

4.3. 마이페이지

P-chain
테스트2
로그아웃
마이페이지

마이 페이지

이메일

test2@test.com

Password(6자 이상)

수정

이름 또는 기관명

test2

지갑 주소

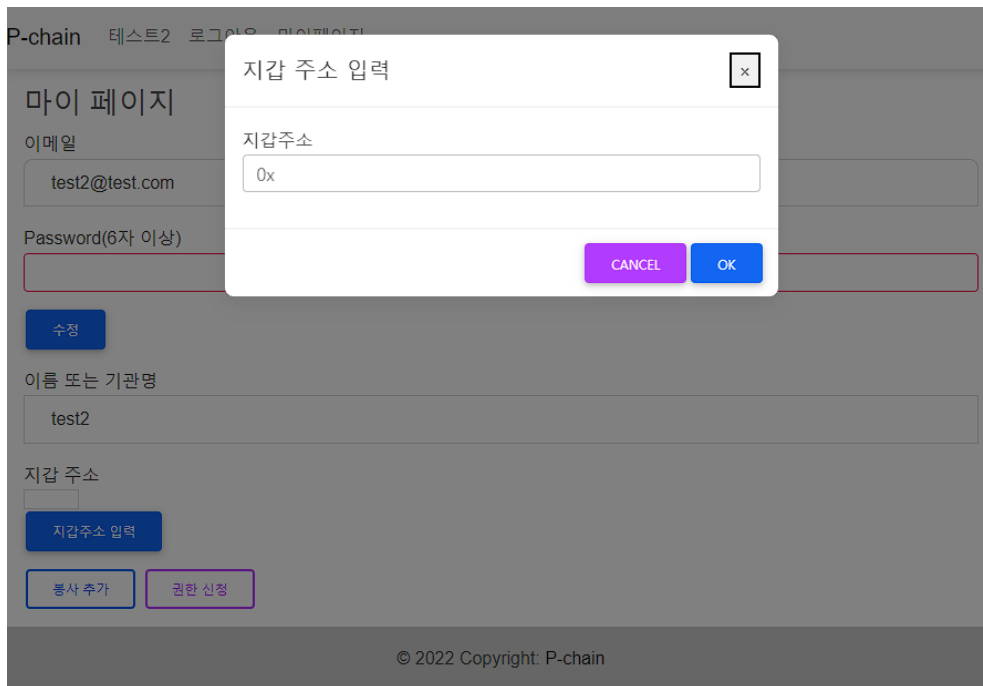
지갑주소 입력

봉사 추가

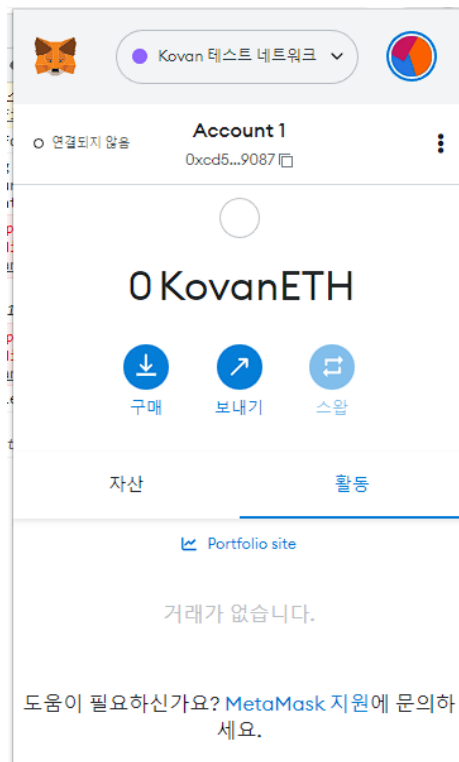
권한 신청

© 2022 Copyright: P-chain

다시 일반 계정으로 돌아와서, 상단 메뉴바의 마이페이지 글자를 누르면 내 정보를 확인할 수 있다. 추가적으로 비밀번호 변경과 지갑주소 입력도 할 수 있다.



지갑주소 입력 버튼을 누르면 주소를 입력하는 modal이 표시된다. 0x로 시작하는 메타마스크 주소를 입력하면 된다.



지갑주소는 메타마스크 확장프로그램에서 확인할 수 있다.

P-chain
테스트2
로그아웃
마이페이지

마이 페이지

이메일

test2@test.com

Password(6자 이상)

수정

이름 또는 기관명

test2

지갑 주소

0xcd50586710A82A6f6cB06d8a017fc3177CEc9087

지갑주소 입력

봉사 추가

권한 신청

Modal에 주소를 입력하고 확인을 누르면, 지갑 주소가 표시된다. 그 옆에는 메타마스크 확장프로그램의 지갑 이미지와 같은 아이콘이 출력되었다.

P-chain
테스트2
로그아웃
마이페이지

봉사 추가

봉사시간 추가

헌혈 추가

기부액 추가

봉사시간을 입력해주세요.

파일 선택

선택된 파일 없음

Choose a file or drop it here...

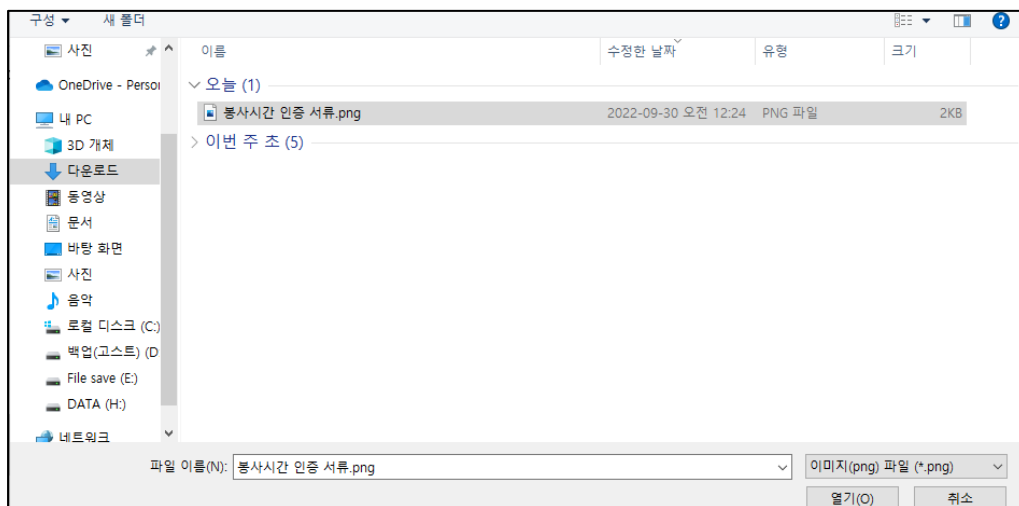
Selected file:

제출

뒤로 가기

© 2022 Copyright: P-chain

마이페이지의 봉사 추가 버튼을 누르면, 봉사시간 입력 창과 파일을 업로드 할 수 있는 버튼이 보인다. 그 외에도 헌혈 추가, 기부 금액 추가와 같은 탭이 존재한다.



파일 선택 버튼을 누르면 내 PC에서 파일을 업로드 할 수 있는 창이 띄워진다. 파일 종류는 이미지 파일로 한정된다.



봉사시간 인증서 파일을 선택하면 해당 파일의 제목이 화면에 표시된다.



봉사 시간을 입력하고, 파일을 업로드한 후 제출 버튼을 누르면 제출 완료라는 창이 출력되고, 이전 페이지로 돌아간다.

P-chain
test1
로그아웃
마이페이지
NFT발급
관리페이지

관리자 승인 페이지

NFT 발급 승인
봉사시간 승인
헌혈 승인
기부액 승인
발급권한 부여

봉사시간 승인 목록

☐ 대상자: test2 / 신청봉사시간: 10 시간
[파일 보기](#)

발급
거절

© 2022 Copyright: P-chain

관리자 계정으로 접속해서 관리 페이지로 이동하면, 봉사시간 승인 목록에 제출한 양식이 추가되었다.

달성 업적

달성 임박

달성업적

현재 업적

봉사시간

10 / 50

기부액

헌혈

일반 계정으로 돌아와서 메인 페이지의 달성 업적 항목을 보면 봉사 시간 10시간이 추가되어 progress bar가 증가한 것을 확인할 수 있다.

5. 기타

5.1. 개발 일정

5월		6월					7월				8월				9월				
3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5
오픈 API 신청																			
		사이트 화면 구상																	
				라우터 구성															
						DB 스키마 정의													
							Back-end 코드 작성												
									Front-end 코드 작성										
									중간 보고서 작성										
											Back-end와 Front-end 연동								
											Open API 적용								
															테스트 및 디버깅				
																	최종 보고서 작성		

5.2. 구성원별 역할

이름	역할
황진하	<ul style="list-style-type: none">● Back-end 코드 작성● 스키마 구성 내용 구현● NFT 관련 Open API 적용
정태영	<ul style="list-style-type: none">● Front-end 코드 작성● 페이지별 라우터 구성● DB 스키마 정의
박예서	<ul style="list-style-type: none">● 웹사이트 화면 구상● Back-end와 Front-end 연동● Back-end와 Front-end 코드 작성
공 통	<ul style="list-style-type: none">● 필요 문서 작성● 보고서 작성● 테스트 및 디버깅