



국민대학교
소프트웨어융합대학
소프트웨어학부

캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	지하철 커뮤니티 – 출그니
팀 명	28조
문서 제목	중간보고서

Version	1.4
Date	2023-04-02

팀원	신 명철 (조장)
	송 민준
	서 민석
지도교수	김 장호 교수

 국민대학교 소프트웨어학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 캡스톤 디자인 수강 학생 중 프로젝트 “지하철 커뮤니티 - 출그니”를 수행하는 팀 “28조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “28조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	중간보고서-지하철 커뮤니티 - 출그니.doc
원안작성자	신명철
수정작성자	서민석, 송민준

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-03-25	신명철	1.0	최초 작성	문서 최초 작성
2023-03-28	신명철	1.1	수행 내용 작성	수행 내용 작성
2023-03-29	송민준	1.2	수행 내용 작성	수행 내용 작성
2023-04-01	서민석	1.3	수행 내용 작성	수행 내용 작성
2023-04-02	신명철	1.4	최종 수정	보고서 제출 전 최종 검토 및 수정

 국민대학교 소프트웨어학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

목 차

1	프로젝트 목표	4
2	수행 내용 및 중간결과	5
2.1	계획서 상의 연구내용	5
2.2	수행내용	6
3	수정된 연구내용 및 추진 방향	16
3.1	수정사항	16
4	향후 추진계획	17
4.1	향후 계획의 세부 내용	17
5	고충 및 건의사항	18

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

1 프로젝트 목표

1. 프로젝트 제작 배경

기존에 존재하는 지하철 애플리케이션은 "지하철 도착 정보", "노선도"에 초점을 둔 경우가 많다. 또한 지하철 실시간 정보를 얻기 위해서는 "서울시 교통 공사" 트위터를 통해 얻거나, 뉴스를 통해 얻어야 한다.

이러한 이유로 지하철의 지연이 발생했을 경우에 사전에 정보를 인지하지 못한다. 따라서 이러한 지하철 실시간 정보와 "지하철 노선도", "도착 정보"을 하나의 어플리케이션으로 해결할 수 있도록 "출그니"를 제작하게 되었다.

정보를 일방적으로 제공하는 것에서 벗어나 유저들이 직접 지하철 실시간 정보를 공유하며 더욱 생생하고 빠른 정보를 제공할 수 있도록 커뮤니티 기능을 중심으로 한 어플리케이션을 제작하고자 한다.

2. 목표

본 프로젝트의 목표는, 사용자가 보다 편리하게 정보를 얻을 수 있기 위해 직관적인 UI와 기존 지하철 애플리케이션이 제공하는 필수적인 기능들을 제공하며 커뮤니티 기능이 존재하는 누구나 쉽고 빠르게 지하철 정보를 공유할 수 있는 어플을 제작하고자 한다.


프로젝트를 통해 DB 관리, 개발, 기획, UI 디자인 등 다양한 경험을 쌓고 팀원과의 소통능력과 개발 능력을 향상시키는 것이 목표이다.



2 수행 내용 및 중간결과

2.1 계획서 상의 연구내용

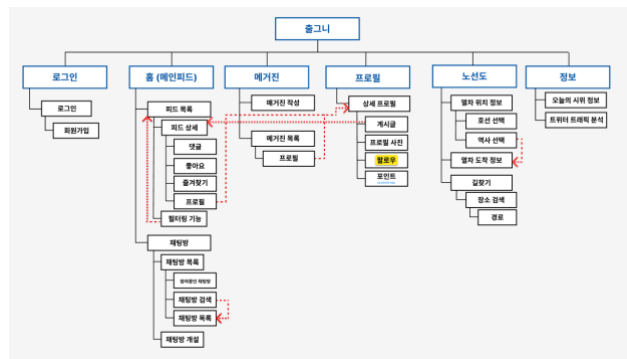
주차	수행 내용	기간
-	프로젝트 주제 선정 주요 기능 작성 구조도 작성 와이어 프레임 구현	2023.2
3월 1주차	DB 스키마, 시위 정보 API 작성, 지하철 실시간 위치 및 도착 정보 기능 구현, AWS Service 이해 및 이해	Mar.1.2023~ Mar.3.2023
3월 2주차	DB ERD 작성, DB Relation Schema 작성, Django rest framework 개발환경 세팅, AWS Lambda 에서 Twitter API 이용,	Mar.3.2023~ Mar.10.2023
3월 3주차	키워드 검색 기능 구현, DB 서버 AWS EC2 적재, DB 커스텀 유저 모델과 로그인, 회원가입 구현, 시위정보를 AWS 통하여 크롤링 구현	Mar.10.2023~ Mar.17.2023
3월 4주차	길찾기 기능 구현, Django 인증 방식 JWT로 변경.	Mar.17.2023~ Mar.24.2023
3월 5주차	중간 발표 준비	Mar.24.2023~ Mar.31.2023
4월 1주차	프로젝트 SW 구조도 작성, Django database - 팔로우 기능 구현, 중간 발표, 영상 제작	Mar.31.2023 ~ Apr.7.2023
4월 2주차	로그인, 회원가입 기능 구현	Apr.7.2023~ Apr.14.2023
4월 3주차	프로필 기능 구현	Apr.14.2023~ Apr.21.2023
4월 4주차	메인 피드 기능 구현	Apr.21.2023 Apr.28.2023
5월 1주차	메인 피드 기능 구현 완료	Apr.28.2023~ May.5.2023

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

5월 2주차	매거진 기능 구현	May.5.2023~ May.12.2023
5월 3주차	채팅방 기능 구현	May.12.2023~ May.19.2023
5월 4주차	채팅방 기능 구현 완료, AWS 상에서 Django 배포 환경 (Nginx, uwsgi) 설정 및 배포	May.19.2023~ May.26.2023
5월 5주차	앱 테스트 및 수정	May.26.2023~ May.31.2023

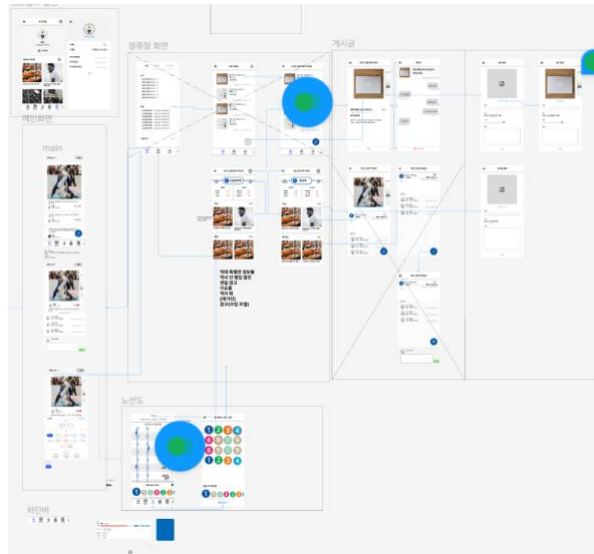
2.2 수행내용

● 구조도 작성



[그림 1] - 출그니 구조도

● 와이어 프레임 작성



[그림 2] - 와이어 프레임

● 길찾기 기능 구현

■ 구현 요약

사용자가 출발지와 도착지를 설정하여, 출발지-도착지 사이에 대중교통을 이용하여 갈 수 있는 경로를 출력한다.

■ 구현 방법


'SK 길찾기 API' 를 사용하기 위해 필요한 요청 **Parameter**에는 출발지와 도착지의 WGS84 좌표가 필요하다. WGS84를 얻기 위해서 카카오 장소 검색 기능을 사용했다.

장소 검색 결과를 바탕으로 얻은 위도와 경도를 'SK API'를 이용하여 길찾기를 구현했다.

■ 통신 결과

```
documents = (ArrayList<Place>) size = 10
> { } 0 = (Place@14659) Place(place_name=김유역 4호선, x=127.02506019737066, y=37.603394782316634)
> { } 1 = (Place@14660) Place(place_name=김유역교차로(동대문로)역, x=127.02282275125626, y=37.60064991814712)
> { } 2 = (Place@14661) Place(place_name=김유역교차로(동대문로)역 (2024년04월예정), x=127.02349751525739, y=37.60375099493983)
> { } 3 = (Place@14662) Place(place_name=KB국민은행 김유역지점, x=127.022381769854, y=37.6028583304156)
> { } 4 = (Place@14663) Place(place_name=김유역교차로, x=127.025163527962, y=37.604216463414)
> { } 5 = (Place@14664) Place(place_name=동대문로 김유역점, x=127.02532132867646, y=37.60201801326226)
> { } 6 = (Place@14665) Place(place_name=NH농협은행 김유역지점, x=127.02243773272622, y=37.60443234976018)
> { } 7 = (Place@14666) Place(place_name=불우교차로 김유역점, x=127.023509699321, y=37.6028833388538)
```

[그림 3] - '김유역' 키워드에 대한 통신 결과.

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

```

{
  "response": {
    "statusCode": 200,
    "message": "success",
    "data": {
      "documents": [
        {
          "place_name": "국민대학교",
          "x": 126.996969239236,
          "y": 37.6107638961532
        },
        {
          "place_name": "국민대학교 후문",
          "x": 126.99966022671174,
          "y": 37.61118199354
        },
        {
          "place_name": "국민대학교 북역",
          "x": 126.996887632826,
          "y": 37.6122343081876
        },
        {
          "place_name": "국민대학교 주자점",
          "x": 126.996709891445,
          "y": 37.6105134142104
        },
        {
          "place_name": "국민대학교앞교차로",
          "x": 126.9943326111,
          "y": 37.6107926106404
        }
      ]
    }
  }
}

```

[그림 4] - '국민대학교' 키워드에 대한 통신 결과

```

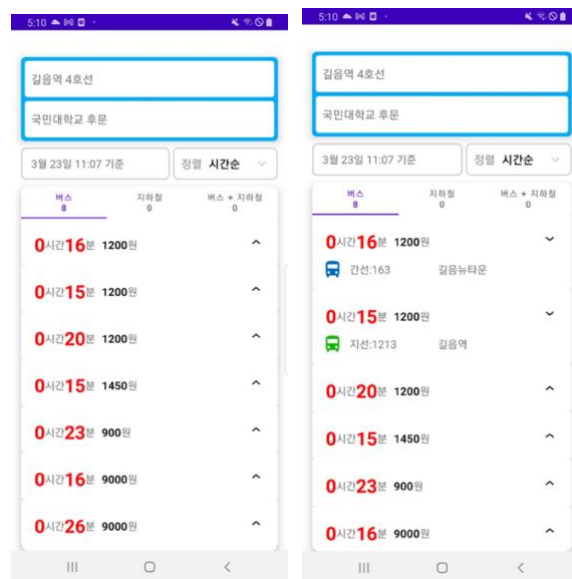
{
  "body": {
    "objects": [
      {
        "metaData": {
          "requestParameters": {
            "requestParameters": {
              "busCount": 8,
              "subwayCount": 0,
              "subwayBusCount": 0,
              "plan": {
                "itineraries": [
                  {
                    "fare": "fare:regular=regularTotalFare=1200, currency=won",
                    "totalTime": 1200,
                    "legs": [
                      {
                        "mode": "WALK",
                        "routeColor": null,
                        "sectionTime": 193.0,
                        "start": "국민대학교",
                        "end": "국민대학교 후문"
                      }
                    ]
                  }
                ]
              }
            }
          }
        }
      }
    ]
  }
}

```

[그림 5] - '국민대학교'와 '길음역 4호선'과의 길찾기 통신 결과

■ UI 구현

'REST API' 통신의 결과 데이터를 바탕으로, Recycler View를 이용하여, UI에 길찾기 통신 결과를 나타내는 작업을 수행했다.



[그림 6] - 경로 검색 결과 (상세보기)


● 실시간 열차 위치 정보

■ 구현 요약

사용자가 원하는 노선(1호선 ~ 9호선)을 선택하면, 해당 노선의 실시간 열차 위치 정보를 제공한다.

■ 구현 방법

'서울시 실시간 열차 위치 정보 API'를 'Retrofit2'를 이용한 통신을 한다. 통신 결

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

과로 얻은 데이터를 이용하여 지하철 노선도를 업데이트 한다.

■ 통신 결과

```

body = [Response@14303] Response(response=[ResponseList@14308] ResponseList@14308 size = 3
> { } 0 = (ResponseList@14311) ResponseList@14311 size = 3
> { } 1 = (ResponseList@14312) ResponseList@14312 size = 3
> { } 2 = (ResponseList@14313) ResponseList@14313 size = 3

```

[그림 7] 실시간 열차 위치 정보 통신 결과 (2호선)

■ UI 구현

◆ 노선 선택 기능

사용자가 노선을 선택할 수 있도록 선택 바를 구현하였다. 사용자가 노선을 선택할 경우 노선을 대표하는 색상으로 화면 UI의 색상이 바뀌도록 구현하였다.

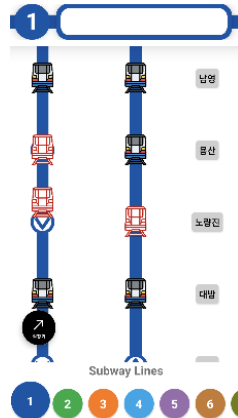


[그림 8] 노선 선택 결과로 색상 변화

◆ 열차 위치 제공

API 통신결과를 바탕으로 기존 노선도에 열차 아이콘을 업데이트하여 열차 위치를 제공하였다.

또한 급행 노선의 경우 아이콘 색상을 분리하였다.



[그림 9] 1호선 실시간 열차 위치 결과

● 실시간 지하철 도착 정보

■ 구현 요약

사용자가 지하철 노선도를 선택하거나, 상단의 검색바를 통해 '역사'를 선택하면, 지하철 도착 정보가 제공됩니다.

■ 구현 방법

'서울시 지하철 도착 정보 API'를 'Retrofit2'를 이용한 통신을 한다. 통신 결과로 얻은 데이터를 'Recycler View'를 이용하여 화면을 업데이트 한다.

■ 통신 결과

```

R body = {Seoul(25034) SeoulErrorBody=ErrorBody(status=200, code=INFO-000, message=정상 처리되었습니다), realtimeArrivalList=[RealtimeArrivalList(subwayId=1001, subwayHeading=null, arvlMsg2=전역 도착, arvlMsg3=동작전, recptnDt=2023-03-28 16:50:42, trainLineNm=소요산행, updnLine=9),
  > { } 0 = (RealtimeArrivalList(25043) RealtimeArrivalList(subwayId=1001, subwayHeading=null, arvlMsg2=전역 도착, arvlMsg3=동작전, recptnDt=2023-03-28 16:52:50, trainLineNm=소요산행, updnLine=9),
  > { } 1 = (RealtimeArrivalList(25044) RealtimeArrivalList(subwayId=1001, subwayHeading=null, arvlMsg2=137번째 전역 (대행), arvlMsg3=대행, recptnDt=2023-03-28 16:52:50, trainLineNm=소요산행, updnLine=9),
  > { } 2 = (RealtimeArrivalList(25045) RealtimeArrivalList(subwayId=1001, subwayHeading=null, arvlMsg2=10번째 전역 (의정부), arvlMsg3=의정부, recptnDt=2023-03-28 16:50:42, trainLineNm=소요산행, updnLine=9),
  > { } 3 = (RealtimeArrivalList(25046) RealtimeArrivalList(subwayId=1001, subwayHeading=null, arvlMsg2=142번째 전역 (가산디지털단지), arvlMsg3=가산디지털단지, recptnDt=2023-03-28 16:52:50, trainLineNm=소요산행, updnLine=9)
  ]
  }
  
```

[그림 10] - 실시간 지하철 도착 정보 API 통신 결과 ('소요산행' 기준)

■ UI 구현

'서울역'과 같이 여러 호선이 지나가는 '환승역'의 경우와 '환승역'이 아닌 경우를 분리하여 구현하였다. 사용자가 검색을 통하여 접근하게 되는 경우 호선에 상관 없이 해당역에 도착하는 모든 열차 도착 정보를 제공한다. 지하철 노선도를 클릭하여 접근하게 되는 경우 클릭 시 선택된 호선의 도착 정보만을 사용자에게 제공한다.



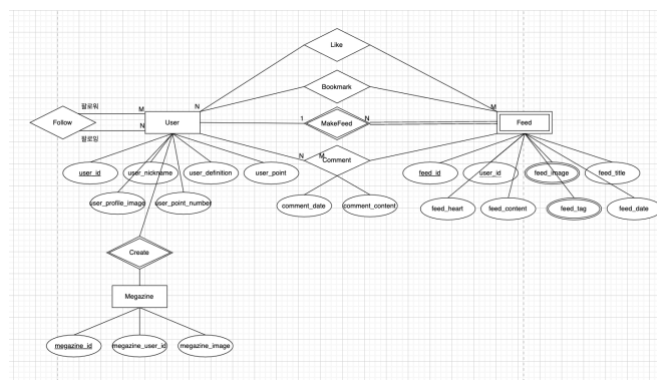
[그림 11] - '좌측' : 노선도 선택, '우측' : 검색 결과 선택

```
body = {Response:24527} ResponseData={DataDate=230313, protest=Protest(place=마곡미스복합단지CP2 일, region=<마곡동>, time=06:20-06:40, people=1,000), DataId=24591} size = 17
Data = [Array@24591] size = 17
> {0} = {Data@24594} DataDate=230313, protest=Protest(place=마곡미스복합단지CP2 일, region=<마곡동>, time=06:20-06:40, people=1,000)
> {1} = {Data@24595} DataDate=230313, protest=Protest(place=사직연교회관앞, region=<여의도동>, time=06:20-06:40, people=1,000)
> {2} = {Data@24596} DataDate=230313, protest=Protest(place=시청 도서관앞, region=<세종대로>, time=19:00-20:00, people=600)
> {3} = {Data@24597} DataDate=230312, protest=Protest(place=교보빌딩 앞, region=<세종대로>, time=15:00-18:00, people=800.0)
> {4} = {Data@24598} DataDate=230312, protest=Protest(place=정동분수대 <=송파동, region=<강동>, time=14:00-16:00, people=100.0)
> {5} = {Data@24599} DataDate=230311, protest=Protest(place=서울역 4출 <= 시청 동편, region=<종로동>, time=15:30-16:30, people=1,000)
> {6} = {Data@24600} DataDate=230311, protest=Protest(place=사직연교회관앞, region=<여의도동>, time=06:20-06:40, people=1,000)
> {7} = {Data@24601} DataDate=230311, protest=Protest(place=강직지역 11출입 3개지점, region=<한강로1가>, time=14:00-17:00, people=10,000)
> {8} = {Data@24602} DataDate=230311, protest=Protest(place=송파동-대림역사실 11출입 3개지점 <= 한강로1가>, region=<북촌동 동>, time=17:00-20:30, people=100,000)
```

[그림 12] - 오늘의 시위정보 API 통신 결과

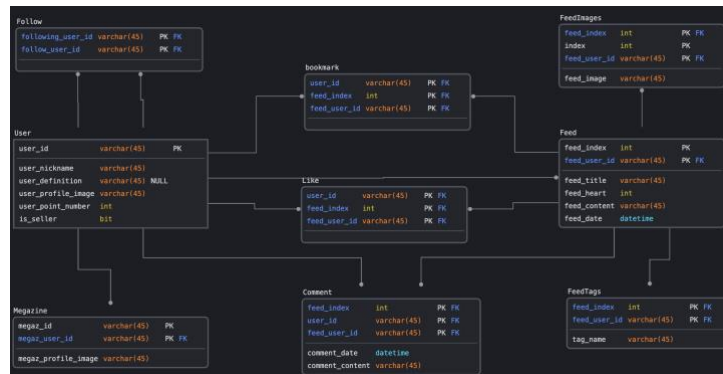
● Database 구축

■ Element Relation Diagram (ERD) 작성



[그림 13] - Element Relation Diagram (ERD)

■ Relation Schema 작성



[그림 14] - Relation Schema

◆ 설명

django 프로젝트를 설치하면 자동으로 auth 모델이 생성되는데 이는 인증 방식에도 사용이 되기 때문에 우리 프로젝트에서 필수적으로 필요하다. 하지만 기본적으로 auth 모델에는 user_name, user_nickname, password 같은 정보만 포함하기 때문에 우리 프로젝트에 맞게 user_definition과 user_point_number 와 user_profile_image와 is_seller 필드를 추가시키기 위해 Cusom User Model을 생성했다.

◆ 구현 요약

AbstractBaseUser를 상속받아서 다음과 같은 필드를 생성했다. 그리고 프로젝트 설정 파일(settings.py)의 AUTH_USER_MODEL을 우리가 생성한 커스텀 모델로 설정했다.


User		
user_id	varchar(45)	PK
user_nickname	varchar(45)	
user_definition	varchar(45)	NULL
user_profile_image	varchar(45)	
user_point_number	int	
is_seller	bit	

[그림 15] - User 모델

◆ 결과 (MySQL Workbench)

1 • SELECT * FROM subway.authentication_user;

	id	password	last_login	user_email	user_nickname	user_definiti...	user_profile_image	user_point_num...	is...
> 3	pbkdt2_sha256S3900008HICG73vIPzPRRLw...			admin.kookmin.ac.kr	admin	H6	https://capstone2subway.s3.ap-northeast-2.am...	0	1
4	pbkdt2_sha256S3900008HICG73vIPzPRRLw...			minjun7410@kookmin.ac.kr	minjun7410	H6	http...	0	0

 국민대학교 소프트웨어학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

[그림 16] - MySQL Workbench 결과

■ JWT 토큰 인증 방식으로 변경

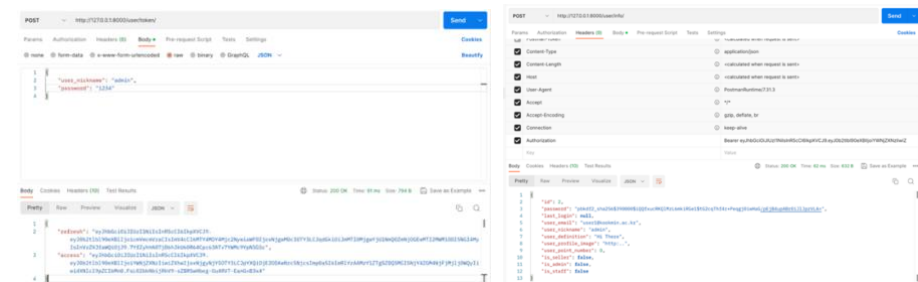
◆ 설명

Django 프로젝트에서 기본적으로 사용하는 Base 인증 방식은 보안에 매우 취약한 인증 방식이다. Token, Session 인증 방식도 마찬가지로 세션이나 쿠키가 노출되면 위험한 인증 방식이 되기 때문에 JWT 인증방식으로 시간이 지나면 파기되면서 서버에 부담이 안가는 인증방식을 채택하였다. 로그인 시, 아이디와 비밀번호를 대조 후 맞으면 jwt 제공을 한다. Jwt를 받은 클라이언트는 헤더에 authorization에 jwt를 포함시켜서 보낸다.

◆ 구현 요약

JWT 인증 방식을 사용하기 위해 jwt 모듈을 설치 후 settings.py에서 SIGNING_KEY를 시크릿 키로 설정. 이는 암호화에 사용한다. 또한 DEFAULT_AUTHENTICATION_CLASSES를 simplejwt로 설정하여 앞으로 요청이 올 때 특별한 경우가 아니라면 jwt로 인증 후 서비스를 이용하게 한다. 그리고 urls.py에 TokenObtainPairView를 token/ 경로에 추가하여 클라이언트가 아이디와 비밀번호를 보냈을 때 응답으로 jwt를 제공하도록 한다.

◆ 결과




[그림 17] - 토큰 받기

[그림 18] - 유저 정보 요청 결과

■ Amazon S3 이미지 업로드

◆ 설명

유저가 프로필 이미지를 설정하기 위해 이미지 파일을 django 서버에 업데이트 요청했을 때 S3에 올린 다음 올린 URL을 User 테이블의

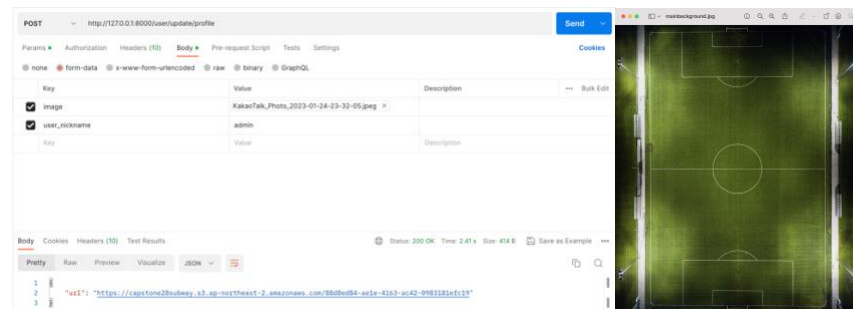
 국민대학교 소프트웨어학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

user_profile_image에 업데이트한다. 나중에 유저가 프로필 정보를 요구할 때 User 테이블에 질의하면 user_profile_image에서 URL을 얻을 수 있으므로 모바일에서는 그 URL을 가지고 이미지에 접근할 수 있다.

◆ 구현 요약

먼저 Django 서버가 S3에 접근하기 위해 S3 버킷을 생성한 후 S3에 이미지를 업로드할 수 있는 권한을 받은 AWS IAM 사용자를 생성한다. 그리고 그 사용자에게 접근하기 위해 AWS_ACCESS_ID와 AWS_SECRET_KEY를 받아 Django 프로젝트의 secrets.json에 적재한다. 그리고 유저로부터 이미지 업로드 요청을 받을 때 boto3 모듈을 통해 S3 client를 얻기 위해 받았던 아이디와 시크릿 키를 사용한다. Client를 얻었다면 업로드할 파일이 S3 버킷 내에서 유일한 uid를 가지도록 uuid 모듈을 통해 uid를 생성하여 그 파일 이름대로 S3에 업로드 시킨다. 마지막으로 파일의 URL을 가지고 DB에 이미지 파일 주소를 업데이트한다.

◆ 결과




[그림 19] - (좌측 : 유저 파일 업로드 결과) / (우측 : URL 결과)

● 트위터 키워드 크롤링

■ 기능 구현

◆ 구현 요약

Twitter에서는 Twitter API를 개발자들에게 제공한다. 이러한 Twitter API를 통하여 최근 Twitter 사용자들이 언급한 지하철 호선도의 정보를 크롤링한다.

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

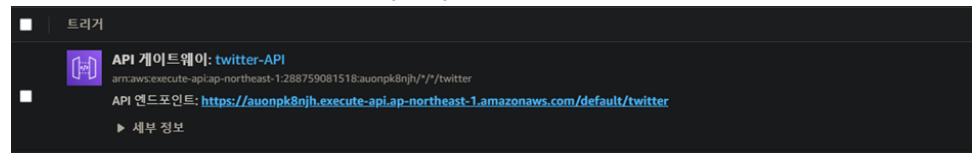
◆ 구현 방법

AWS Service에서 AWS Lambda를 이용하여 Python에서 Twitter가 제공하는 Tweepy 패키지를 이용하여 Twitter에서 실시간 정보를 가져와 정제 한 후 json파일로 저장하여 API Gateway를 통하여 앱에 전달한다.

◆ 구현 결과



[그림 20] - API 통신 결과



[그림 20] - API GATEWAY

● 경찰청 시위 정보 추출

■ 시위정보 추출 기능 구현

◆ 구현 요약

경찰청 시위정보를 토대로 정보를 가져온 후 AWS Service를 통하여 정보를 앱에 전달한다.

◆ 구현 방법

AWS Service에서 AWS Lambda를 이용하여 Python을 이용해 API Gateway로 json 파일 형태로 결과값을 출력하여 앱은 이 값을 토대로 앱은 사용자에게 정보 제공한다.


◆ 구현 결과



[그림 21] - API GATEWAY



[그림 22] - API 통신 결과

 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

3 수정된 연구내용 및 추진 방향

3.1 수정사항

1. SNS 기능 추가

초기 제안서의 목표는 지하철 시위 정보나 지연정보를 빠르게 얻을 수 있는 앱을 제작하고자 했다.

하지만 타 앱과의 차별성을 생각해보았을 때, 기존의 '서울메트로', '카카오 맵', '카카오 지하철'과 경쟁력이 없었다. 이를 해결하기 위해 팀 회의간 아이디어의 도출 과정에서 '사람들이 지하철의 상황을 공유하면 더 빠르게 알 수 있지 않을까?' 라는 가정 속에 타 앱과의 차별성으로 커뮤니티를 목표로 설정하게 되었다.

재설정된 목표를 기준으로 기존에 설정하였던 기능에서 커뮤니티 기능을 중심으로 재구성하였다. 기존에도 '게시판'이라는 기능이 존재하여 소통할 수 있는 공간이 있었다.

프로젝트의 목표인 직관적인 UI를 위해 각 노선별로 게시판이 존재하는 방식에서 '트위터' 또는 '인스타그램'과 같이 '팔로우한 친구의 게시글' 또는 '본인이 필터링한 태그'를 중심으로 한 게시글을 탐색하는 방식으로 변경하였다.

2. 길찾기 기능 추가

'길찾기 기능'은 타 지하철 관련 앱이 기본적으로 가지고 있는 기능이다. 단순 커뮤니티 기능만을 제공한다면 유저들은 타 앱을 이용하여 '길찾기' 기능을 이용해야 하기 때문에 프로젝트의 목표인 '하나의 애플리케이션'으로 해결하고자 하는 목표를 어기게 되므로 길찾기 기능을 추가하였다.

'길찾기 기능'은 출발점과 도착점을 키워드 검색을 통해 장소를 설정하고, 설정된 장소를 기반으로 최적의 대중교통 경로를 설정하여 제공하는 기능이다.

'길찾기 기능'을 실제로 구현하는 것보다, '카카오', '네이버', 'SK'에서 제공하는 API를 사용하는 것이 더욱 효율적이고 정확한 결과를 도출한다고 판단하여 'SK'에서 제공하는 '대중교통 API'를 사용하여 구현하였다.

 국민대학교 소프트웨어학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

4 향후 추진계획

4.1 향후 계획의 세부 내용

- 커뮤니티 기능 구현

커뮤니티 관련 기능 구현을 위해 관련 DB 모델을 완성하고, Android와 DB API 서버간 데이터 통신을 통해 '커뮤니티' 관련 기능 (로그인, 피드, 매거진)을 제공할 계획이다.

- 포인트 사용처 기능 구상

포인트 기능은 커뮤니티 유입에 있어 중요한 수단으로 이용이 된다. 포인트 기능 구현을 위해서는 사용처와 포인트 획득 방법에 대한 구상이 필수적이다. 이를 위해 관련 기능을 구상하고 구현할 계획이다.

	중간보고서		
	프로젝트 명	지하철 커뮤니티 - 출그니	
	팀 명	28조	
	Confidential Restricted	Version 1.4	2023-APR-02

5 고충 및 건의사항

개발과 보고서 작성 및 발표준비를 병행하다 보니, 시간이 많이 부족한 것 같습니다.