‘알단지’ 프로젝트

최종보고서

팀 명 3KM

팀 원 32141306 김학재

32140878 김영훈

32177200 김진용

32141665 민준홍

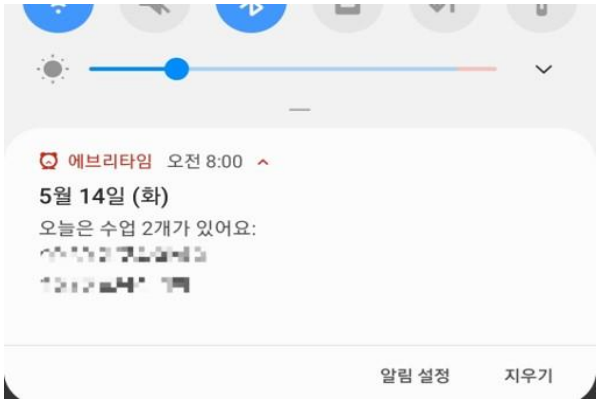
목차

1. 설계 소개

본 프로젝트의 제목은 '알단지'로 알람, 단국대, 지도를 결합하여 만든 단어이다. '알단지'는 단국대 학생들을 위한 통합 어플리케이션으로 안드로이드가 제공하는 알람 기능과, 차세대 기술로 각광받는 AR 기술을 적용한 새로운 캠퍼스 어플리케이션의 개발을 목표로 한다.

이를 위해 현재 단국대 학생들이 주로 사용하는 어플인 '에브리타임'과 단국대 공식 어플을 조사했으며 그 결과 우리가 보완하고자 한 기능은 다음과 같다.

첫째, 알람 기능의 빈약이다. '에브리타임' 어플의 경우 매일 오전 당일 시간표를 토대로 하루의 강의 알람을 일괄적으로 제공하지만 그 외에 상세한 알람은 제공하지 않는다.

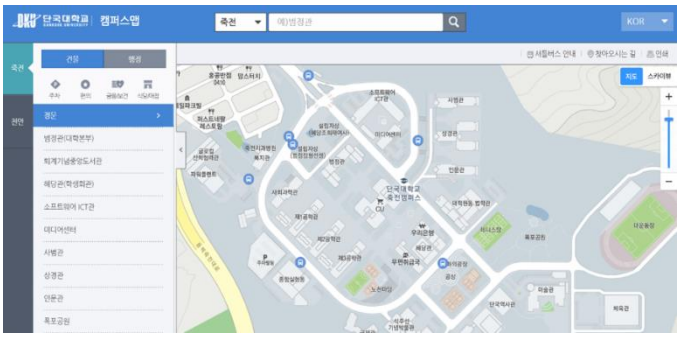


[그림 1] '에브리타임' 어플이 제공하는 알람

이를 해결하기 위해 본 프로젝트에서는 실제 강의 목록을 토대로 원하는 시간대에 알람을 쉽게 설정하고 이를 푸쉬 알람을 통해 수신하는 것을 목표로 한다.

둘째, 캠퍼스 맵 기능의 부재이다. 단국대 공식 어플의 경우 캠퍼스맵 메뉴가 존재하지 않으며, 단국대 홈페이지는 매번 로그인을 통해 접속을 해야 하는 등 그 유용성에 있어 한계가 존재한다.

[그림 2] 단국대 공식 어플



[그림 3] 단국대 홈페이지

이를 해결하기 위해 본 프로젝트에서는 교내 모든 건물 및 보행 가능 경로를 지도 상에 표시하고 목적지를 선택 시 목적지까지의 경로를 2D 및 AR 네비게이션을 통해 확인할 수 있게 함을 목표로 한다.

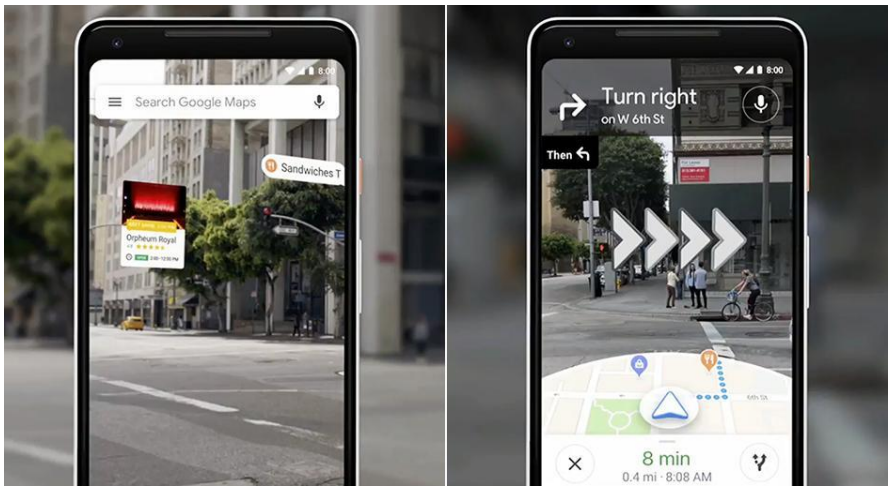
위 기능의 구현에 있어 필요한 알람과 AR 및 Mapbox의 주요 기능 구현에 대한 설명은 후술하도록 한다.

2. 배경 지식

2.1. 지도 기반 길 찾기

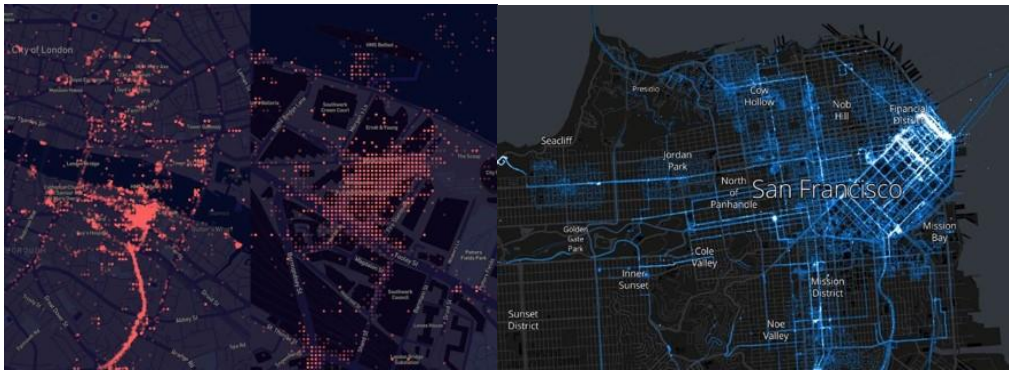
지도를 기반으로 한 길 찾기는 GPS를 사용해 내 위치와 목적지 위치를 토대로 경로를 사용자에게 보여주는 서비스이다. 'Google Map', '카카오 맵', '네이버 지도' 등이 현재 시행 중인 서비스의 대표적 예시이며 다양한 웹 사이트나 애플리케이션에서 이를 사용할 수 있다.

지도 서비스에 AR 기술의 적용도 상용화되어 가고 있다. 2019년 8월 구글은 AR을 활용한 'Google Maps AR'을 선보였다. 목적지를 입력 후 'Start AR' 버튼을 클릭하면 AR 네비게이션 기능이 실행된다. 경로에 대한 안내 사항을 실시간으로 카메라를 통해 확인할 수 있으며, 주변 정보 등의 사항도 확인할 수 있다. 국내에서는 보안 사항 등의 이유로 사용해 볼 수는 없지만 이를 통해 향후 지도가 새로운 중심 플랫폼이 될 것이라고 보는 의견도 적지 않다.



[그림 4] Google Maps AR

2.2. Mapbox

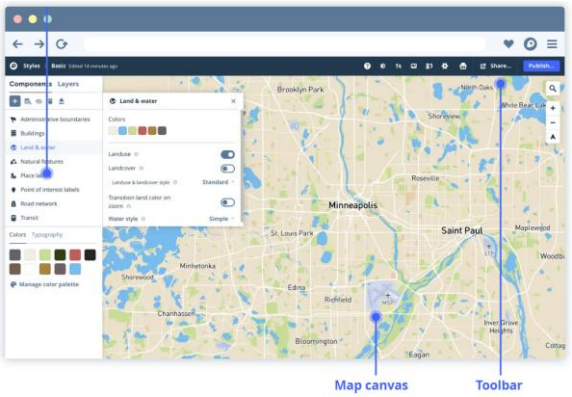


[그림 5] Mapbox의 활용 예시

Mapbox는 웹사이트 및 어플리케이션을 위한 지도 및 위치 제공 클라우드 플랫폼이다. 지도를 사용하는 어플리케이션, 웹 등 서비스를 위한 다양한 SDK 및 API를 제공하며, 현재 페이스북, 스냅챗, 삼성 등 다양한 서비스들이 Mapbox를 활용하고 있다.

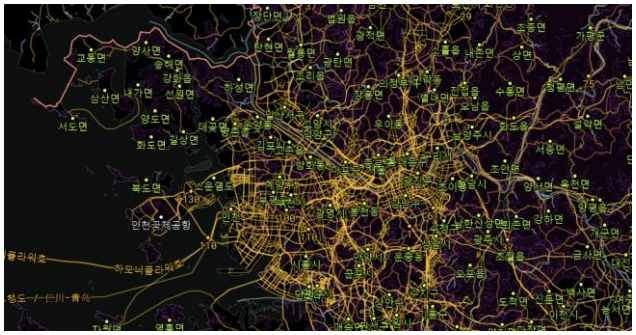
Mapbox는 지도 생성에 있어 다양한 데이터 소스를 활용한다. OpenStreetMap, NASA, DigitalGlobe 등의 자료들을 활용하며 OpenStreetMap의 경우 사용자가 원하는 지역의 지도를 직접 수정하고 편집함으로써 모든 사용자들이 보다 정밀한 지도를 사용할 수 있다.

또한 Mapbox는 Mapbox Studio라는 제작 도구를 지원하며 이를 통해 사용자는 원하는 대로 지도를 수정하거나 스타일을 조절할 수 있다. Mapbox Studio에서는 Style, Tileset, Dataset의 편집이 가능하며 이들에 대한 설명은 다음과 같다.



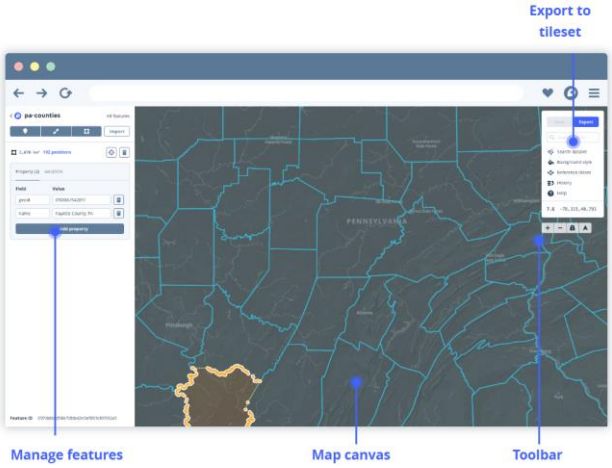
[그림 6] Mapbox - Style

Style은 지도가 페이지에 렌더링되는 방법을 지정한다. 메뉴를 이용해 빌딩, 지형, 도로 등의 스타일을 지정하고 이를 웹 페이지, 어플리케이션 등에서 활용할 수 있다.



[그림 7] Mapbox - Tileset

Tileset은 Mapbox에서 만들어지는 지도의 주요 데이터로, 동일한 크기로 쪼개어진 벡터 데이터들의 집합이다. Mapbox가 지원하는 다양한 라이브러리 및 SDK의 핵심 요소로 완성된 지도를 모바일, 웹 브라우저 환경에서 볼 수 있도록 해준다.



[그림 8] Mapbox - Dataset

Dataset은 GeoJSON 특징들의 집합이다. GeoJSON이란 위치정보를 갖는 점을 기반으로 체계적으로 지형을 표현하기 위해 설계된 표준 형식을 뜻한다. 제작된 Dataset을 Tileset으로 export해 활용할 수 있다.

2.3. AR

AR(Augmented Reality, 증강현실)은 VR(Virtual Reality, 가상현실)의 한 분야로 실제로 존재하는 환경에 가상의 사물이나 정보를 합성하여 마치 원래의 환경에 존재하는 사물처럼 보이도록 하는 컴퓨터 그래픽 기법이다. 기존의 가상 현실은 가상의 공간, 사물만을 대상으로 했다면 증강현실은 현실 세계를 기반으로 가상의 사물을 합성해 현실 세계만으로 얻기 어려운 부가적인 정보들을 보강할 수 있다.

AR 기술은 VR 기술과 달리 실제 환경을 볼 수 있어 보다 나은 현실감을 제공한다. 현재 원격 의료 진단, 방송, 건축설계 등에 활용되고 있으며 스마트폰의 보급과 함께 본격적인 상업화 단계에 들어서는 추세이다.

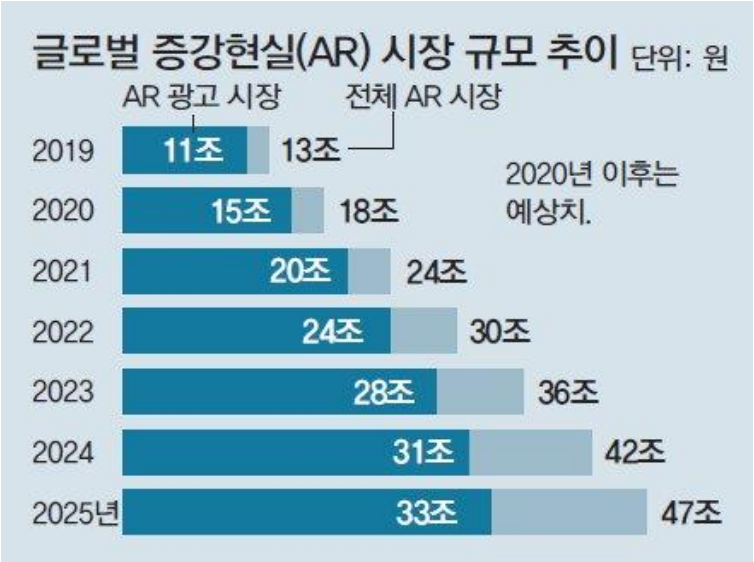
[그림 9] AR 기술 – 건축

[그림 10] AR 기술 - 의료

[그림 11] AR 기술 – 포켓몬 고

[그림 12] AR 기술 – 이케아 플레이스

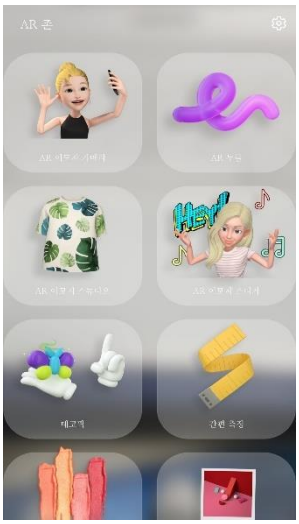
AR 기술은 시장성 측면에서도 향후 전망이 매우 고무적이다. 영국 시장조사기관 Ovum에 따르면 모바일 AR 앱 시장 규모는 올해 약 18조 원에서 2025년 약 47조 원으로 2.6배에 이를 것이라고 밝혔으며, 국내 시장의 경우에도 연평균 42.9%의 성장률을 보이고 있다.



[그림 13] AR 시장 규모 추이(자료 : 영국 시장조사기관 Ovum)

국내외 다양한 기업 및 산업들도 AR 서비스를 선보이는 추세이다. 애플은 AR 관련 스타트업을 인수해 AR 기능을 지원하는 LiDAR 센서를 탑재한 아이패드 프로를 출시한 바 있으며, 삼성은 올해 4월 'AR 존' 서비스를 선보인 바 있다.

[그림 14] 애플 – 아이패드 프로의 AR 기능



[그림 15] 삼성 – AR 존

3. 개발 과정

3.1. 개발 환경

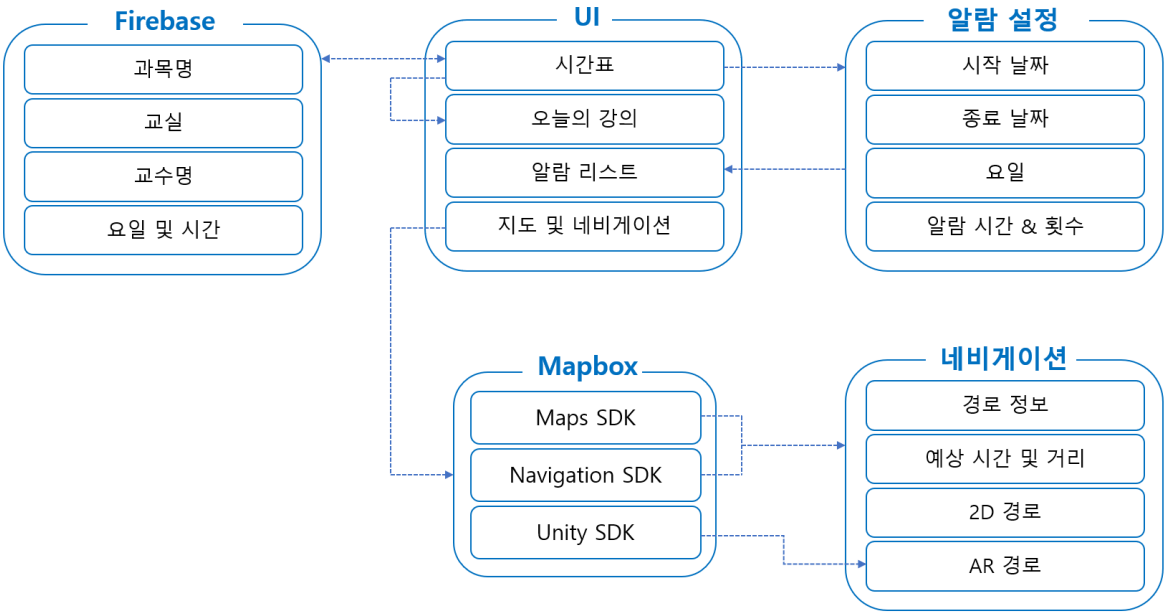
개발에 사용한 도구 및 기술들의 버전 및 설명을 표로 나타냈으며 자세한 설명은 후술한다.

이름	버전	설명
안드로이드 스튜디오	4.1.0	어플리케이션의 기능 및 UI 설계
파이어베이스		종합시간표 저장을 위한 클라우드
SQLite	2.1.0	강의 알람 설정을 위한 안드로이드 내장 데이터베이스
JOSM	17013	OpenStreetMap 수정에 필요한 프로그램
QGIS	3.10.6	GeoReferencing 진행에 필요한 프로그램
mapbox Maps SDK	9.3.0	mapbox에서 제공하는 지도 SDK
mapbox Navigation SDK	1.1.0	mapbox에서 제공하는 네비게이션 SDK
mapbox Unity SDK	2.1.1	mapbox에서 제공하는 Unity SDK
Unity	2019.3.14	AR 설계를 위한 제작 도구
Direction API	5	길 찾기 API

[표 1] 개발 환경

개발 과정에서 협업을 위해 Github repository를 활용하였다. 각 기능들이 개별적으로 구현되어 최종 UI에 합치는 과정을 진행하였으며 상세 내용은 다음 링크에서 확인할 수 있다. (https://github.com/TwinkleRing/Capstone-Project)

3.2. FlowChart



[그림 16] 프로젝트 '알단지' FlowChart

본 프로젝트의 FlowChart를 간단하게 그림으로 나타냈으며 각 기능들에 대한 자세한 개발 과정은 후술한다.

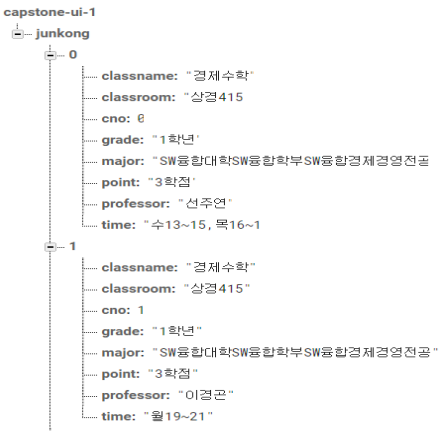
3.3. 세부 기능

3.3.1. 시간표 검색 기능

시간표 기능에서 쓰이는 과목 정보는 단국대학교 종합 시간표를 기반으로 파이썬을 활용해 크롤링 후, JSON파일로 변환하여 Google Firebase DB에 저장하였다. 저장된 데이터는 키 값, 데이터의 쌍으로 구성되어 있다.

cno	classname	grade	major	point	professor	time	classroom
0	경제수학	1학년	SW융합대학SW	3학점	선주연	수13~15,목	상경415
1	경제수학	1학년	SW융합대학SW	3학점	이경곤	월19~21	상경415
2	기술과경영	3학년	SW융합대학SW	2학점	박문수	수13~16	소프트307
3	모바일시스템개론	2학년	SW융합대학SW	2학점	한기철	목9~12	소프트307
4	소프트웨어개론	1학년	SW융합대학SW	2학점	이상범	화15~18	소프트305
5	소프트웨어개론	1학년	SW융합대학SW	2학점	정혜진	화15~18	2공105
6	소프트웨어개론	1학년	SW융합대학SW	2학점	정혜진	수9~12	소프트307
7	선형대수	2학년	SW융합대학SW	3학점	황두성	월13~15,목	소프트307
8	선형대수	2학년	SW융합대학SW	3학점	윤석현	화11~13수	소프트310
9	확률및통계학	2학년	SW융합대학SW	3학점	정영기	수13~18	국제102
10	확률및통계학	2학년	SW융합대학SW	3학점	전경배	금13~18	소프트310
11	확률및통계학	2학년	SW융합대학SW	3학점	전경배	월1~6	소프트310
12	SW융합코딩1	1학년	SW융합대학SW	3학점	정혜진	월1~6	소프트301
13	SW융합코딩1	1학년	SW융합대학SW	3학점	김대원	월1~6	2공522
14	SW융합코딩1	1학년	SW융합대학SW	3학점	백수진	월15~19	소프트310
15	SW융합코딩1	1학년	SW융합대학SW	3학점	정우진	월15~19	국제205_P
16	SW융합코딩1	1학년	SW융합대학SW	3학점	박태근	화1~6	소프트303

[그림 17] 크롤링한 엑셀 파일



[그림 18] JSON 형식으로 FIrebaseDB에 저장

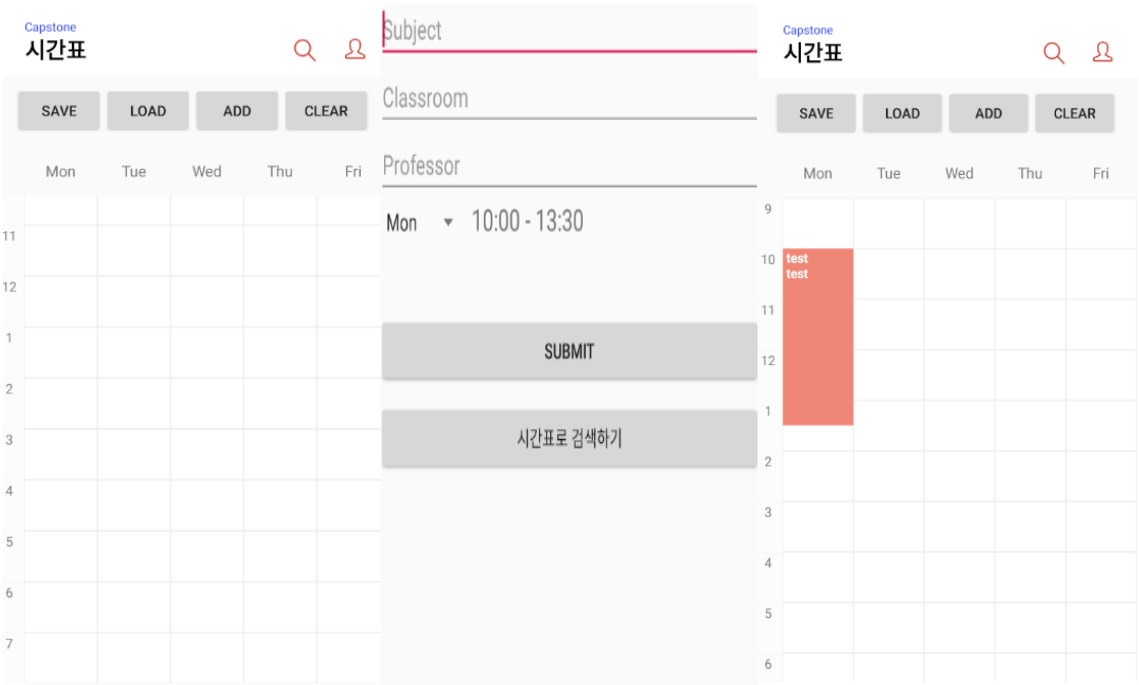
안드로이드에서는 데이터를 검색 시, 쿼리형식으로 Firebase에 저장되어 있는 데이터를 받아 RecyclerView로 띄워지도록 구현하였다.

```
query = databaseReference.orderByChild("classname").startAt(s).endAt(s + "\uf8ff");
query.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        arrayList.clear();
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            arrayList.add(0, snapshot.getValue(junkonglist.class));
        }
```

[그림 19] 시간표 DB 검색 코드

[그림 20] 시간표 검색 예시

databaseReference.orderByChild를 사용하여 Firebase 내에 있는 classname(과목명) 데이터와 같은 값을 가져오도록 하였다.



[그림 21] 시간표 검색 UI

시간표 검색 기능의 어플 내 UI는 위와 같다. TimetableView를 기반으로 레이아웃을 구성하였으며, ADD 버튼을 누를 시, 과목정보, 수업명, 교수, 요일, 시간을 입력할 수 있는 화면이 나오도록 구현하였다.

안드로이드 내부에서는 startActivityForResult를 통해서 다음 액티비티에 요구하는 기능에 대한 REQUEST 값을 전달하고, 작업이 끝나면 REQUEST에 대한 응답을 받는 작업을 통해 액티비티와 프래그먼트 간의 통신 기능을 수행한다. 통신 기능에는 putExtra와 getExtra를 사용하였다. 보내는 쪽에서 putExtra를 통해 값을 넘겨주면 받는 쪽에서 getExtra를 통해 값을 받는다. 이를 통해 검색된 과목의 정보를 넘겨받게 된다.

```
private void inputDataProcessing(){
    schedule.setClassTitle(subjectEdit.getText().toString());
    schedule.setClassPlace(classroomEdit.getText().toString());
    schedule.setProfessorName(professorEdit.getText().toString());

    schedule2.setClassTitle(subjectEdit.getText().toString());
    schedule2.setClassPlace(classroomEdit.getText().toString());
    schedule2.setProfessorName(professorEdit.getText().toString());
}
```

[그림 22] inputDataProcessing 코드

inputDataProcessing 함수를 통해서 각 TextView에 있는 정보를 schedule에 추가하여 Time TableView 에 추가하고, 이를 통해 시간표에 과목명과 강의실이 시간에 맞게 나타나게 된다.

3.3.2. 알람 기능

알람 기능은 시간표에 등록된 강의마다 개별로 원하는 시간에 설정하여 사용자의 디바이스로 푸쉬 알람을 전송하도록 구현하였다.

구현에는 먼저 강의 알람을 설정하는데 사용하는 데이터를 저장하기 위한 데이터베이스로 안드로이드 내장 데이터베이스인 SQLite를 사용하였다.

```
public class MyDBHelper extends SQLiteOpenHelper {
    public MyDBHelper(Context context) { super(context, name: "capstoneDB.db", factory: null, version: 1); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE class (classId INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, className TEXT NOT NULL, " +
                "startDate TEXT NOT NULL,  endDate TEXT NOT NULL,  timesPerDay INTEGER NOT NULL," +
                "mon INTEGER, tue INTEGER, wed INTEGER, thu INTEGER," +
                "fri INTEGER, sat INTEGER, sun INTEGER);");

        db.execSQL("CREATE TABLE time (timeId INTEGER PRIMARY KEY AUTOINCREMENT, oneTime TEXT NOT NULL, " +
                "twoTime TEXT, threeTime TEXT, " +
                "fourTime TEXT, fiveTime TEXT, " +
                "FOREIGN KEY(timeId) REFERENCES class(classId));");
    }
```

[그림 23] 알람을 저장하기 위한 데이터베이스 - SQLite

테이블 속성에는 강의 아이디(기본 키)와 강의 명, 시작 날짜, 종료 날짜, 반복 횟수, 요일로 구성되어 있다.

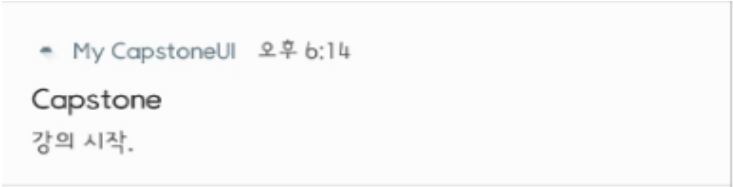
전송할 푸쉬 알람을 설정하기 위해 Android 지원 라이브러리인 NotificationCompat API의 NotificationCompat.Builder 객체를 사용하였고, 이 객체를 통해 알람에 들어간 콘텐츠와 생성한 알람이 저장될 채널을 설정하여 전체 푸쉬 알람을 생성하였다.

```
NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

NotificationCompat.Builder builder = new NotificationCompat.Builder(context, CHANNEL_ID);
builder.setSmallIcon(R.drawable.ic_launcher_foreground).setContentTitle("Capstone").setContentText("강의 시작.")
        .setWhen(Calendar.getInstance().getTimeInMillis())
        .setContentIntent(pendingIntent).setAutoCancel(true);

notificationManager.notify( id: 0, builder.build());
```

[그림 24] 푸쉬 알람 생성을 위한 NotificationCompat API의 사용



[그림 25] 생성한 푸쉬 알람 예시

알람 서비스는 어플이 실행 중이지 않더라도 백그라운드 상태에서 현재 디바이스 시간과 사용자가 설정한 시간을 반복적으로 비교하는 작업이 필요하다. 이 작업을 구현하기 위해 안드로이드 애플리케이션 구성 요소인 Service를 사용하였다. Service는 일반적으로 화면 없이 동작하는 프로세스로 백그라운드에서 오래 실행되는 작업을 구현할 때 사용된다.

Service를 바탕으로 사용자가 알람을 설정하면 설정한 알람의 정보와 현재 요일 및 시간을 비교하는 작업을 반복하고 설정한 시간이 되면 앱 내부에서 알람이 울리고 BroadcastReceiver가 이벤트를 수신한다.

```
public class MyService extends Service {

 AlarmManager alarmManager = (AlarmManager) getApplicationContext().getSystemService(Context.ALARM_SERVICE);
 alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, calendarTime, sender[count]);
```

[그림 26] 내부 알람이 실행되면 이벤트를 Broadcasting해주는 메서드

안드로이드에서 알람이 울리는 등 이벤트가 발생할 시에 방송 신호가 발신되는데 이러한 방송 신호를 받으면 푸쉬 알람이 디바이스로 전송된다.

```
<receiver android:name=".Alarm.AlarmReceiver"/>
```

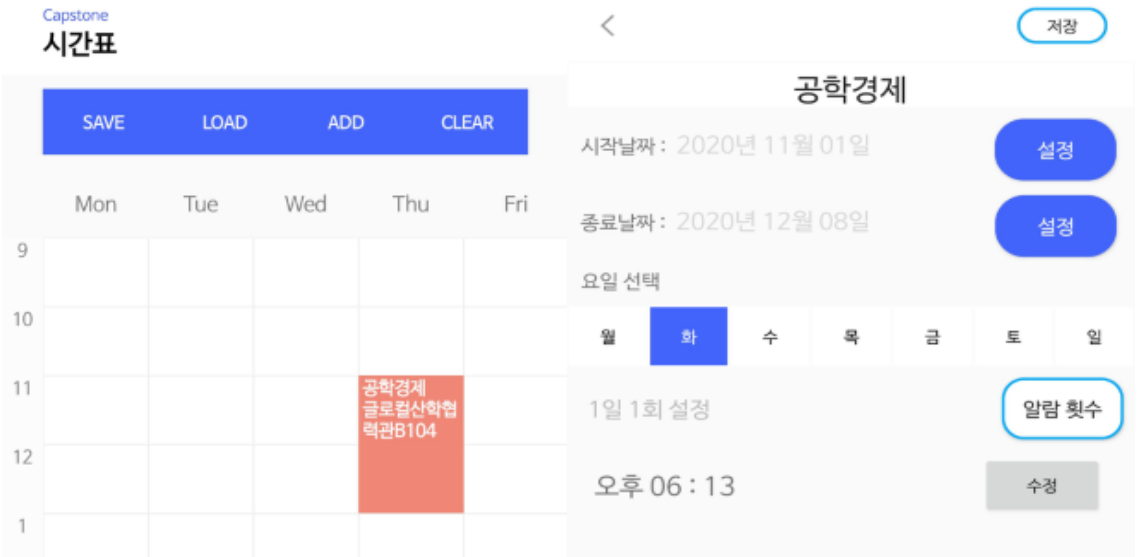
[그림 27] BroadcastReceiver를 상속받은 클래스를 AndroidManifest.xml에 등록

```
public class AlarmReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        notificationManager.notify( id: 0, builder.build());
```

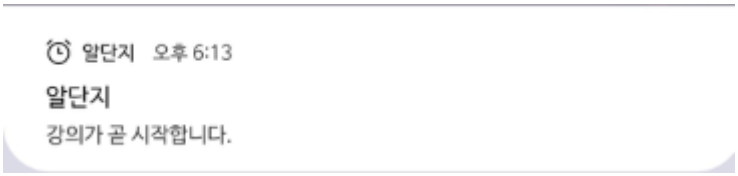
[그림 28] AlarmReceiver – 방송 신호 수신 및 알람 전송

BroadcastReceiver를 상속받은 AlarmReceiver의 onReceive 메서드가 방송 신호를 받으면 notify를 통해 설정해 놓은 푸쉬 알람이 디바이스로 전송된다.

[그림 29] 전송된 푸쉬 알람 예시



[그림 30] 알람 설정 UI



[그림 31] 실제 푸쉬 알람

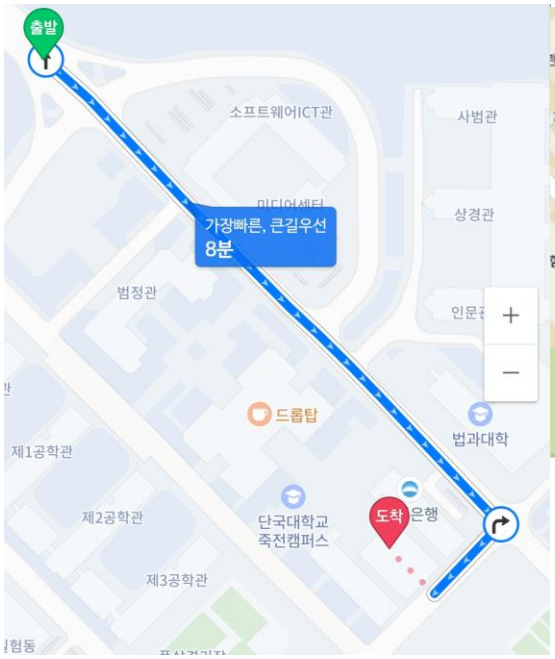
구현된 알람 기능의 어플 내 UI는 위와 같다.

시간표 화면에서 개별 강의를 선택하면 알람 설정 화면으로 이동한다.

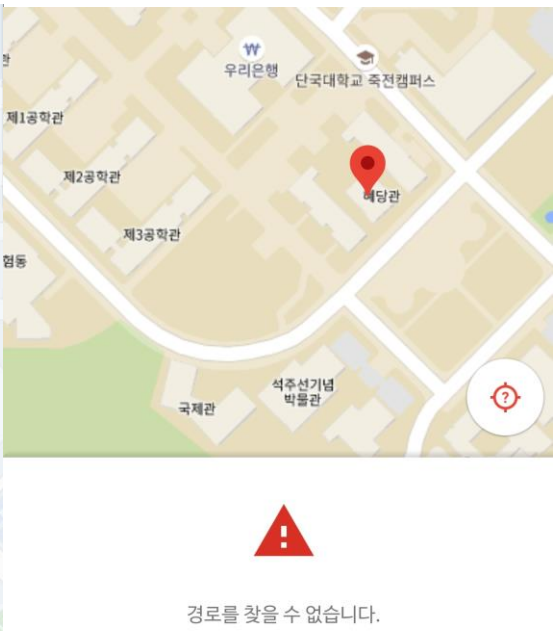
시작 날짜와, 종료 날짜, 요일과 알람 시간, 횟수를 설정 후 저장 버튼을 누르면 알람 서비스가 시작되며 이를 디바이스에서 확인할 수 있다.

3.3.3. 지도 및 네비게이션 기능

지도 및 네비게이션 기능 구현에 있어서는 Mapbox를 활용하였다. 보편적으로 사용하는 네이버, 구글의 길 찾기 기능을 사용하지 않은 이유는 다음과 같다.



[그림 32] 네이버 지도 경로



[그림 33] 구글 지도 경로

네이버의 경우 국내 지도에 대해 자세하고 정확한 정보를 제공하지만 큰 길을 우선적으로 고려하므로 본 프로젝트에서 목표로 하는 캠퍼스 내 경로의 경우 정확도가 떨어질 수 있다. [그림 32]에서 혜당관 앞 구역은 보행이 가능하지만 나타나지 않는 것을 볼 수 있다.

구글은 보안 상의 이유로 국내 지도에 대해 자동차, 도보 경로 정보를 제공하지 않는다. 따라서, 기능 구현에 있어 Mapbox를 활용하기로 결정하였다.

정확한 길 안내를 위해서는 정확한 지도의 제작 과정이 필수적이다. Mapbox의 지도는 최신의 국내 지리 정보를 반영하지 못한 상태여서 지도 제작에 GeoReferencing 과정을 진행하였다.

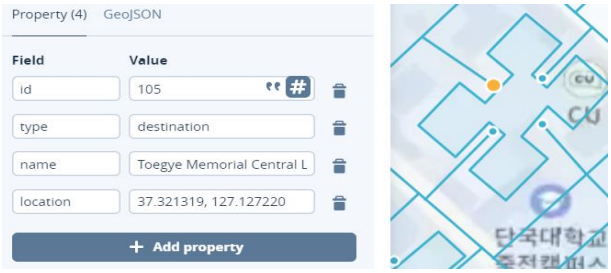
GeoReferencing이란 지리적 이미지를 실제 세계의 좌표와 연동하는 작업으로 위성 이미지나 최신 지도 이미지를 통해 원하는 지도의 제작이 가능하다. 따라서, 국내 지도 중 자세하고 정확한 데이터를 제공하는 네이버 맵의 위성 지도 및 일반 지도를 기반으로 어플리케이션에 사용될 지도를 제작하였다.

[그림 34] GeoReferencing 진행 과정

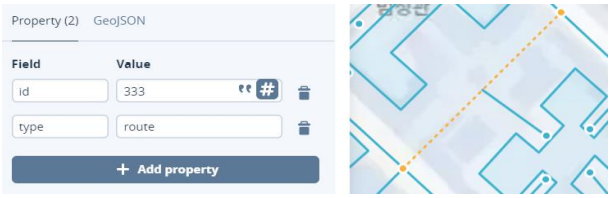
OpenStreetMap에서 단국대 지역의 이미지를 불러온 뒤 우리가 사용할 위성 이미지를 불러와 4개 이상의 좌표를 지정해주면 우측 그림과 같이 GeoReferencing 과정이 완료된다. 생성된 지도 이미지는 TIFF 형식으로 저장된다. TIFF란 Tag Image File Format의 약자로, 무손실 압축과 태그를 지원하는 최초의 이미지 포맷이다. 전문 사진, 래스터 그래픽 이미지 등을 저장하는데 사용된다.

TIFF 이미지는 Mapbox에서 제공하는 지도 제작 도구인 Mapbox Studio에서 사용이 가능하다. Mapbox Studio에서 해당 이미지를 불러온 뒤 교내 모든 건물, 입구, 보행 가능 경로들을 조원들이 직접 교내를 조사해서 지도에 다음과 같이 지정하였다.

[그림 35] Mapbox Studio – 교내 건물



[그림 36] Mapbox Studio – 교내 건물 입구



[그림 37] Mapbox Studio – 교내 경로

이렇게 생성된 지도 데이터를 style에서 불러와 색깔 및 텍스트 조정을 거쳐 가독성을 높이는 과정을 진행하였다. 공대, 인문대, 기숙사, 예체대 총 4개의 주요 구역으로 구분했으며 각 구역마다 belong 태그를 지정해줌으로써 각각 다른 색깔로 구분이 가능하도록 하였다.

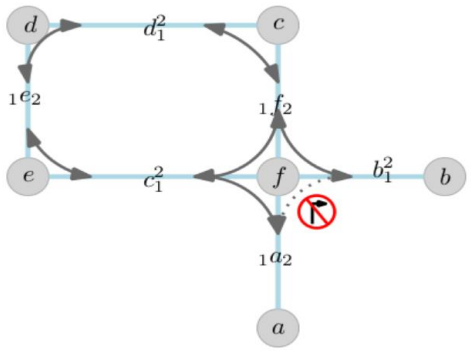
[그림 38] styling을 위한 교내 건물의 belong 태그 지정

[그림 39] 구현된 학교 지도

이렇게 생성된 지도는 Maps SDK를 사용해 어플리케이션 내에서 활용이 가능하다. Maps SDK는 실시간 위치를 수신하는 LocationEngine을 제공하며 callback 함수를 통해 지정한 시간마다 위치 정보를 지속적으로 업데이트한다. callback의 결과로 위도, 경도 정보를 받아오며 이 정보는 별도의 변수로 저장되어 위치 정보가 바뀔 때마다 변경되게 된다.

네비게이션 기능 구현에는 Navigation SDK와 Directions API를 활용하였다. Navigation SDK는 mapbox가 제공하는 네비게이션 기능을 사용할 수 있도록 해주며, 이 SDK는 Directions API에 요청을 보내 길 안내 및 경로 정보를 받아오게 된다.

네비게이션 즉, 길 찾기 기능의 동작 원리는 다음과 같다. 길 찾기에는 최단 경로 알고리즘 중 하나인 에이스타 알고리즘이 사용된다. 에이스타 알고리즘은 최소 비용으로 목표 지점까지 도착하는 경로를 찾는 알고리즘이다. 현재 상태의 비용을 g(x), 현재 상태에서 다음 상태로 이동할 때의 휴리스틱 함수를 h(x)라고 할 때, 둘을 더한 f(x) = g(x) + h(x)가 최소가 되는 지점을 우선적으로 탐색한다.



[그림 40] Direction API의 동작 원리

Mapbox는 상기한 알고리즘 외에도 다양한 요소들을 고려해 경로를 생성한다. [그림 40]처럼 교통 정보를 반영하거나 도로 속도 제한 정보 등을 실시간으로 반영하여 네비게이션 기능을 제공한다.

```
private void getRoute_navi_walking (Point origin, Point destinaton) {
    // https://docs.mapbox.com/android/navigation/overview/map-matching/
    NavigationRoute.builder(getActivity()).accessToken(Mapbox.getAccessToken())
            .profile(DirectionsCriteria.PROFILE_WALKING)//도보 길찾기
            .origin(origin)//출발지
            .destination(destinaton).//도착지
            build().
            getRoute(new Callback<DirectionsResponse>() {
                @Override
                public void onResponse(Call<DirectionsResponse> call, Response<DirectionsResponse> response) {
                    if (response.body() == null) {
                        return;
                    } else if (response.body().routes().size() ==0) {
                        return;
                    }
                    currentRoute = response.body().routes().get(0);
                    if (navigationMapRoute != null) {
                        navigationMapRoute.removeRoute();
                    } else {
                        navigationMapRoute = new NavigationMapRoute(null, mapView, mapboxMap, R.style.NavigationMapRoute);
                    }
                    navigationMapRoute.addRoute(currentRoute);
                }
                @Override
                public void onFailure(Call<DirectionsResponse> call, Throwable t) {
                }
            });
}
```

[그림 41] 경로 찾기 코드

구현된 경로 찾기 코드는 위와 같다. 출발지와 도착지 좌표가 정해지면 Directions API에 요청을 보내며 성공 시 두 좌표 간의 경로를 응답으로 받게 된다.


```
{
  - routes: [
      - {
            weight_name: "pedestrian",
            weight: 584.021,
            duration: 559.021,
            distance: 781.029,
          - legs: [
              - {
                    steps: [ ],
                  - admins: [
                      - {
                            iso_3166_1_alpha3: "KOR",
                            iso_3166_1: "KR"
                        }
                    ],
                    duration: 559.021,
                    distance: 781.029,
                    weight: 584.021,
                    summary: "세교1로, 통복시장로54번길"
                }
            ],
            geometry: "e`qqeA{irkqF~CvDeE~H~DdFjEeFfOkVnb@xk@vgA}IAfCuDx@cf@lE_I`ClCb`@k_@tJolfXhk@lZhl@pGxM"
        }
    ],
  - waypoints: [
      - {
            distance: 6.37,
            name: "",
          - location: [
                127.084206,
                37.004307
            ]
        },
      - {
            distance: 11.204,
            name: "통복시장로53번길",
          - location: [
                127.085075,
                36.999718
            ]
        }
    ],
    code: "Ok",
    uuid: "xeQvIMGMXU3C3J_8WB4xY1ftsuL8YPGJnGfTrrL2_erHDMZqLQfnOQ=="
}
```


[그림 42] Directions API의 응답

경로에 대한 정보는 위와 같이 JSON 형태이며, 도로명(name), 거리(distance), 좌표(location) 등의 정보를 확인할 수 있다.

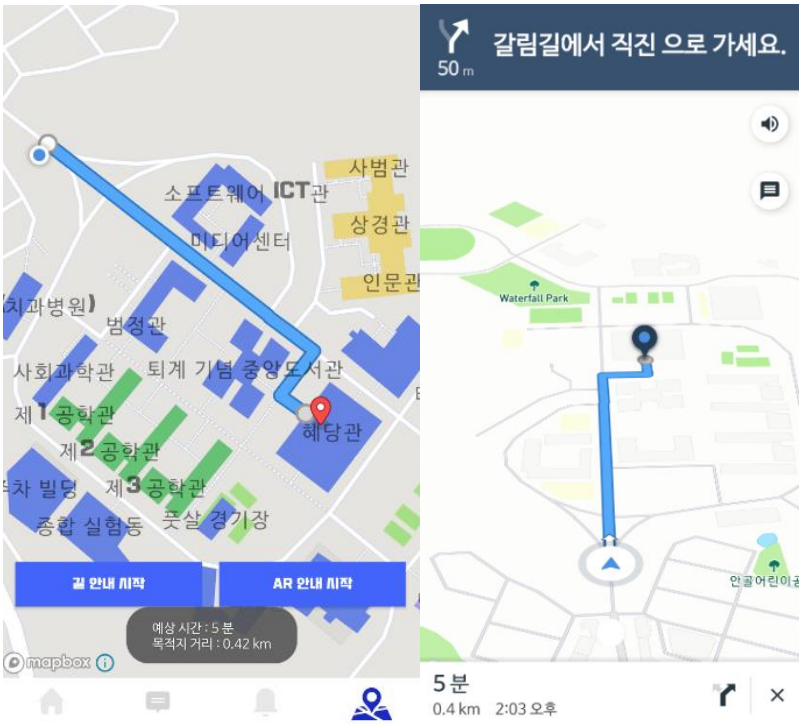
그러나 주어진 지도와 기능들 로만 제작된 네비게이션에는 문제점이 존재하였다. 지도 상에 우리가 임의로 경로를 추가했으나 실제 지도 상에는 존재하지 않는 경로이기 때문에 경로 안내가 원하는 대로 구현되지 않았다. [그림 43]과 같이 사용자는 빨간 경로를 보행 가능하나 어플리케이션의 경로는 파란 경로대로만 안내하는 것을 확인할 수 있다.

[그림 43] 네비게이션 경로 문제 발생

이를 해결하기 위해 Mapbox 측에 직접 문의 메일을 보냈고 그 결과 Mapbox는 OpenStreetMap이라는 지도 데이터를 기반으로 지도를 생성한다는 답변을 받을 수 있었다. OpenStreetMap은 누구나 전 세계 지도의 편집이 가능한 협동 프로젝트로 약 650만 명(2020년 11월 기준)의 유저 수를 보유하고 있다. 단국대 지도의 생성을 위해 OpenStreetMap의 편집 도구인 JOSM을 활용했으며, 교내 모든 경로 및 건물의 입구들을 포함해 새로운 지도를 제작하였다. (https://www.openstreetmap.org/changeset/91498317)

[그림 44] JOSM을 활용한 새로운 교내 지도의 생성

이렇게 새롭게 제작된 지도는 약 일주일의 승인 과정을 거쳐 Mapbox의 지도에 반영되게 된다. 이 지도를 기반으로 어플리케이션 내 지도를 생성하고 Directions API 요청을 보내면 응답으로 정밀해진 경로 및 정보를 받게 되며 이를 어플리케이션 내에서 확인할 수 있다.

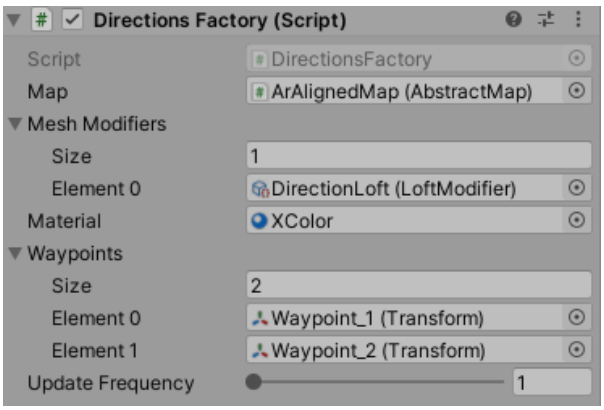


[그림 45] 네비게이션 UI

구현된 네비게이션의 UI는 위와 같다. 목적지를 선택하면 현재 위치로부터 경로를 확인할 수 있으며 목적지까지의 예상 시간, 거리를 확인할 수 있다. '길 안내 시작' 버튼을 누르면 네비게이션이 실행되며 오른쪽 그림과 같이 길 안내 사항과 함께 음성 안내, 사용자의 피드백 기능이 제공된다.

Directions API는 유니티를 통해 제작된 AR 환경 내에서도 동일한 동작을 한다.

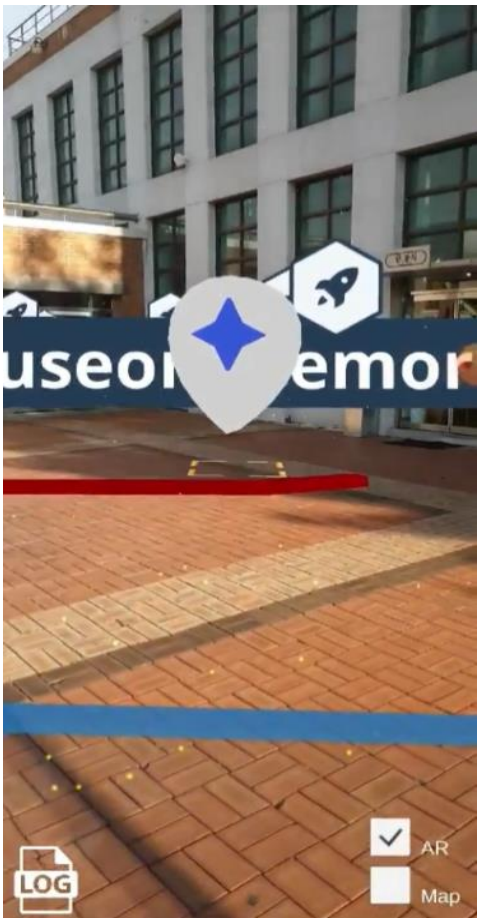
Mapbox Unity SDK는 Directions API를 지원하는 Directions.prefab과 DirectionsFactory를 제공한다. Directions.prefab에서 Waypoint를 지정해주게 되면 DirectionsFactory는 일정 시간마다 Directions API에 요청을 보내며, 이에 대한 응답으로 Waypoint간의 경로를 확인할 수 있다. 본 프로젝트에서는 경로의 종류를 도보(Walking)로 지정했다.



```
void Query()
{
    var count = _waypoints.Length;
    var wp = new Vector2d[count];
    for (int i = 0; i < count; i++)
    {
        wp[i] = _waypoints[i].GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
        Debug.Log("Test : " + wp[i]);
    }
    var _directionResource = new DirectionResource(wp, RoutingProfile.Walking);
    _directionResource.Steps = true;
    _directions.Query(_directionResource, HandleDirectionsResponse);
}
```

[그림 46] DirectionsFactory

이 경로는 지도 상에 생성되게 되며 실제 지도 좌표와 연동이 되어 AR로 확인이 가능하게 된다. 사용자의 일정 거리 이동을 통해 지도의 보정이 진행되면 경로 정보도 업데이트되어 더욱 정확한 AR 경로를 확인할 수 있다.



[그림 47] AR 경로 확인

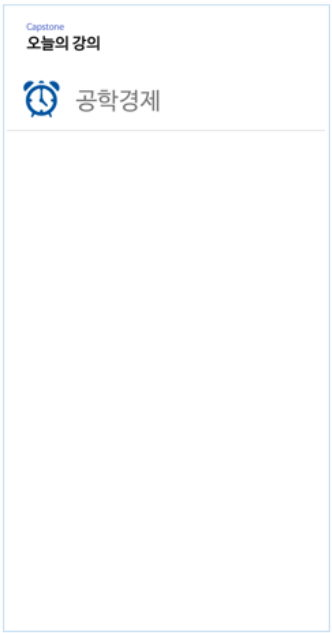
네비게이션 상에서 목적지를 설정한 뒤 'AR 안내 시작' 버튼을 누르면 목적지까지의 경로를 [그림47]과 같이 AR로 확인할 수 있다.

3.4. 종합 UI

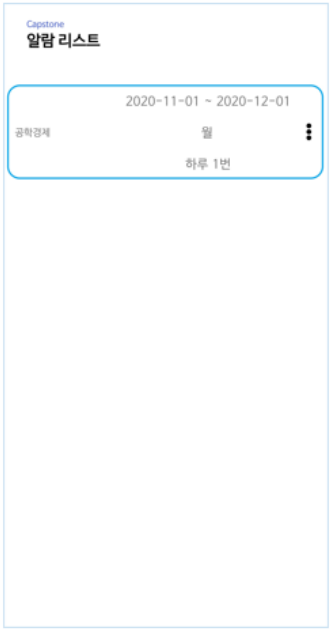
[그림 48] 시간표 화면

종합 UI는 위와 같다. 메인 화면에서 현재 시간표를 확인할 수 있으며 상단에는 SAVE(시간표 저장), LOAD(시간표 불러오기), ADD(과목 추가하기), CLEAR(시간표 지우기) 기능을 수행하는 버튼들이 위치해 있다.

아래쪽 메뉴는 차례대로 오늘의 강의, 알람 리스트, 지도 및 네비게이션 메뉴이며 각 메뉴별 화면은 다음과 같다.



[그림 49] 오늘의 강의



[그림 50] 알람 리스트

[그림 51] 지도 및 네비게이션

'오늘의 강의' 에서는 오늘 예정된 강의 목록을 보여준다.
'알람 리스트' 에서는 현재 설정된 모든 알람 목록을 보여주며, 시간, 알람 횟수 등의 수정이 가능하다.
'지도 및 네비게이션' 은 실행 시 자동으로 내 위치를 받아오며 실패 시 학교 위치를 자동으로 보여준다. 상단에 학교 구역, 단국대 지도, 내 위치 3개의 버튼을 배치해 사용자의 편의성을 높였으며 목적지를 선택 시 길 안내 시작, AR 안내 시작 버튼이 활성화된다.

4. 보완 사항 및 설계 제한 요소

• 보완 사항

- 학교 강의 시간표의 업데이트를 통해 매 학기마다 사용이 가능할 것이다.
- 알람에 좀 더 자세한 정보를 제공할 수 있을 것이다.
- AR의 경우 사용자의 이동을 통한 좌표 조정이 이루어져야 그 정확도가 향상된다. 별도의 과정 없이 AR의 정확도를 높일 수 있는 방법을 탐색해야 할 것이다.
- 학교 근처 상가, 버스정류장 등의 정보를 활용해 더 광범위하고 자세한 정보 제공이 가능할 것이다.

• 설계 제한 요소

- 일주일에 2일 이상 있는 강의의 경우 Schedule에 직접 추가해줘야 한다.
- mapbox가 기본적으로 한글을 제공하지 않아 모든 건물명을 직접 지정해줘야 한다.

5. 결론 및 평가

임의의 시간을 설정하여 확인한 결과 푸쉬 알람과 Firebase의 데이터를 가져와 과목에 대한 시간표 설정 기능도 정상적으로 동작했다.
Mapbox와 연동을 통한 네비게이션 시스템은 단국대학교 캠퍼스 내 임의의 목적지까지의 최단 거리를 안내하는 것을 성공적으로 확인했으며, AR 안내를 통한 목적지 마커도 육안으로 성공적으로 확인할 수 있었다.

우리는 본 프로젝트를 통해 실제 사용중인 서비스의 부족한 점을 분석하고 구현했으며, 구현 과정에 있어 새롭고 다양한 기술 스택의 사용, 오픈 소스에 기여를 할 수 있었다. 또한, 실제 기업과의 소통을 통해 문제점에 대한 해결책을 찾아냈으며 구현 과정에 있어 발생했던 문제점들에 대해 협의를 통해 해결책을 모색할 수 있었다.

결론적으로 초기에 설정했던 목표를 모두 달성할 수 있었으며 추후 시간표의 지속적인 연동과 기능 보완을 통해 단국대 학생들의 더 쾌적한 캠퍼스 라이프를 가능하게 할 수 있을 것이다.

6. 참고문헌, 역할분담

• 참고 문헌

- Mapbox Tutorial (https://docs.mapbox.com/help/tutorials/)
- Mapbox Documentation (https://docs.mapbox.com/)
- Android 개발자 가이드(https://developer.android.com/guide?hl=ko)
- Mapbox와 주고받은 메일

• 역할 분담

이름	역할	설명
김학재	**팀장**	- 팀장으로서 전체 단계의 조율 및 팀원들 역할 분담 - 네비게이션 기능 구현 담당
김영훈	**팀원**	- 적절한 자료 탐색을 바탕으로 프로젝트의 청사진 마련 - 전체적 UI, 알람 기능 구현 담당
김진용	**팀원**	- 기능 구현(AR)에 있어 다양한 방법을 모색, 시도함 - 시연 영상 보고서 등의 제작에 있어 적극적 기여
민준홍	**팀원**	- 기능 구현에 있어 발생한 문제점들을 끝까지 탐구 및 해결함 - 시간표 추가 기능 구현 담당

[표 2] 프로젝트 '알단지' 역할 분담