**소프트웨어 공학**

**Term Project**



**20164091 송희령**

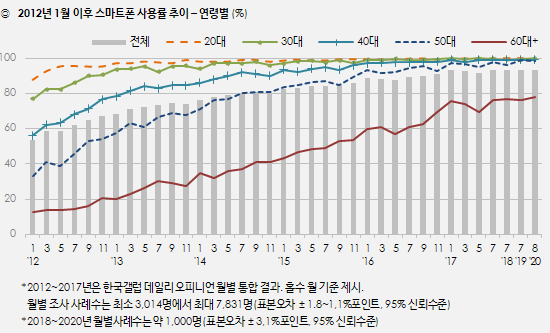
**20164108 임재석**

**20164109 유승우**

**20164113 최상록**

# **Ⅰ. 프로젝트 개요**

## **1. 개발동기와 목표**



* 현재 연령별 스마트폰 사용률이다. 점점 증가하는 추세이고 현재 **50대까지 90%이상 스마트폰**을 사용 중이다. 그 만큼 우리 일상에서 스마트폰은 없어서는 안될 존재이다.
* 이에 따라, 우리 팀은 안드로이드, 즉 앱 개발을 통해 사용자에게 유익한 정보를 주는 것을 이번 프로젝트의 목표로 하였다.

## **2. 프로젝트 주제 선정**

* 프로젝트 주제를 정하기 위해, 회의 전 팀원들은 안드로이드 앱 개발이라는 주제로 각자 1개 이상의 토픽을 준비하도록 하였다.
* 각 팀원들은 **건강, 교통정보, 자기개발, 음식, 1인가구**, 5가지 토픽을 제시하였고 의견을 좁히기   
  위해 **“실생활에서 가장 필요한 것”**을 기준으로 아이디어를 선정하였다.
* 회의 결과, 교통정보 토픽이 선정되었고 세부주제는 고속도로의 교통정보를 하는 것으로 결론을   
  내렸다.

## **3. 프로젝트 산출물**

|  |  |  |
| --- | --- | --- |
| **단 계** | **작 업** | **산출물** |
| **프로젝트 계획** | 프로젝트 일정을 위한 시간 조율 프로젝트 계획서 작성 | 프로젝트 계획서 WBS |
| **요구 분석/설계** | 요구사항 분석/정의/관리 | 요구명세서 |
| **개발** | Android Studio를 통한 App개발 | 기능정의서  코드정의서 |
| **테스트** | 단위테스트  통합테스트 및 미비점 보완  사용자 인수 테스트 및 미비점 보완 | 단위 테스트 시나리오 및 결과서  통합 테스트 시나리오 및 결과서  인수 테스트 결과서 |

## **4. 대상**

* Android 사용자

# **Ⅱ. 팀 소개**

## **1. 프로젝트 팀명 결정 방법**

**브레인 스토밍 기법**을 이용하여 5분 동안 팀원 전부가 아이디어를 제시

* **No.5(N: 너만 O: 오면 5: 5인큐)**
  + 게임 중 League Of Legend에서 최대 인원 수인 5명을 맞추면 협력 플레이가 가능함.
  + 5명이 아니여도 4명에서 문제를 해결 할 수 있고 하나로 뭉친다는 것을 목표로 함.
* **Experiencer**
  + Experience + er을 합한 말로 경험을 하는 사람이라는 뜻을 가짐.
  + 경험을 하며 지식을 쌓아가는 것을 목표로 함.
* **Find**
  + Find의 뜻인 찾다 라는 의미를 통해 지식을 찾아가는 목표를 가짐.
* **S\_Login**
  + Software + Login을 합한 말로 소프트웨어에 접속하는 뜻을 가짐.
  + 작업 전 시작하는 로그인처럼 팀원 모두 소프트웨어에 공학에 처음이므로 기본을 가지고 임하도록 하는 목표를 가짐.
* **Assembly**
  + Assembly의 뜻 중 하나인 조립을 통해 4명이 조립되어 하나의 팀이 된다는 뜻을 가짐.
  + 컴퓨터공학과에서 배우는 공통분모로 의미가 있다 생각함.

**※ 팀원들과 회의 결과 No.5가 선정되었음.**

# **Ⅲ. 조직 구성**

## **1. 조직 구성**

소프트웨어 제작이라는 공통적인 목표를 개인이 모여 해결하기 위해서는 조직을 구성해 각 개인의 역

활을 잘 정의하고 조직원들 간의 상호작용이 조화롭게 이루어 져야 한다. 소프트웨어 생산 같은 경우 다른 제품과 달라 조직 구성방법이 많은 영향을 끼치는데, 특히 의사 결정권 같은 경우 프로젝트 팀 개발 구성에 있어 가장 중요한 요소이다.   
프로젝트 팀 개발 구성 중 계층적 팀 구성은 4명이라는 소수에 어울리지 않았고, 에고레스 팀 구성은 오히려 각 개인의 생산성을 떨어트리고 민주적인 해결방안을 찾기 힘들 것으로 판단하여 **책임 프로그래머 팀 구성**을 선택하였다.

하지만 인원이 소수이고 개인 역량도 비슷하다고 판단하여 리더가 임무에 대한 방향성을 제시하되, 각각 임무를 프로그래머들 간 검토하고 책임을 지는 에고레스 팀의 구성의 장점을 따와 구성하였다.

## **2. 인력 배치**

책임 프로그래머: **최상록**

프로젝트에 대한 방향성 제시, 요구 분석과 설계, 중요 부분의 프로그래밍 및 기술적 판단

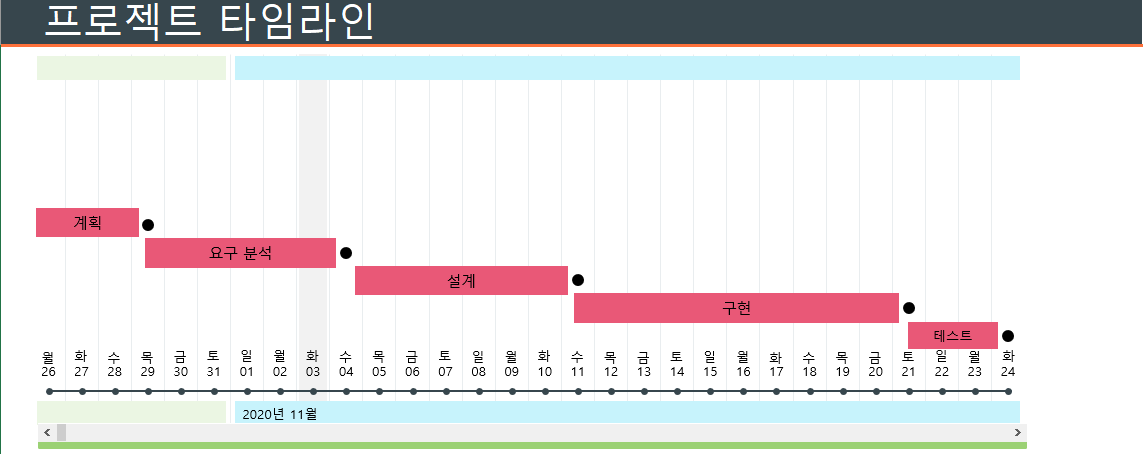
프로그래머 사서: **송희령**

보조 프로그래머: **유승우**

프로그래머: **임재석**

프로그램 리스트, 설계 문서, 테스트 계획 관리와 기술적인 문제에 대한 자문, 분석 설계 담당

# **Ⅳ. 프로젝트 회의**



|  |  |  |
| --- | --- | --- |
| **분 류** | **회의 날짜** | **기 간** |
| **프로젝트 계획** | 10월 26일 | 10월 26일 ~ 10월 29일 |
| **요구 분석** | 10월 29일 | 10월 29일 ~ 11월 4일 |
| **설계** | 11월 4일 | 11월 4일 ~ 11월 11일 |
| **개발** | 11월 11일 | 11월 11일 ~ 11월 21일 |
| **테스트** | 11월 21일 | 11월 21일 ~ |

**Ⅴ. 자원 예측**

## **1. 인력**

비용 예측 방법 중 하나인 COCOMO 방법의 유기적 모드(Organic Mode)

* 예상 총 코드 길이: 16000줄
* 프로그램 길이 KDSI: 12000/1000 = 12
* 개발자 인원/월(PM): 2.4 X = 32.6(명/월)
* 개발 기간(TDEV): 2.5 X = 9.4(개월)
* 투입 인원: 32.6/9.4(PM/TDEV) = 3.4(명)

# **Ⅵ. WBS(Work BreakDown Structure)**

WBS는 프로젝트 착수, 실행, 통제, 종료의 기본단위로 원가와 일정 계획을 수립하고 통제하는 기본 단위 이다.

고속도로 교통 정보 App

요구 분석

계획서 검토

목표 및 제약정의

팀원들과 면담

요구분석서 작성

설 계

분석서 검토

프로그램 구조 설계

UI 수정 및 설계

상세 설계

코 딩

테스트

단위 테스트 실행

통합 테스트 실행

미비점 보완

사용자 인수 테스트

계획서 검토

# **Ⅶ. 기술 관리 방법**

## **1. 위험 관리**

* 외부 UI의 부족: 디자인의 부족으로 미숙함이 보일 가능성이 있음.
* 기술적 취약: 안드로이드 개발이 처음이라 코딩 능력 부족으로 프로그램의 전반적이 완성도 저하 가능성이 있음.
* 프로젝트 개발 지연: 시간적 문제로 지연될 가능성이 있음
* 의견 충돌: 팀원들간 개발 중 의견 충돌이 발생할 가능성이 있음.

## **2. 위험 해결 방안**

* 포토샵을 이용해 원하는 사진을 편집하여 UI를 좀 더 깔금하게 만든다.
* 안드로이드 개발 스터디를 통하여 정보를 얻고 코딩의 완성도를 높인다.
* 시간적 문제가 일어나기 전에 미리 사전공지하여 다른 팀원들 에게 피해가 최소화 하도록 한다.
* 의견을 모두 들어본 후 최선의 방법을 선택하여 합의점을 빠르게 찾는다.

# **Ⅷ. 개발방법론**

## **1. 개발방법론 선정**

계획, 요구분석, 설계, 개발, 테스트를 각 단계별로 완벽하게 완료해야 다음단계로 넘어 갈 수 있기 때문에 이에 맞는 **폭포수 모델**을 선택하였다.

# **Ⅸ. 개발환경**

|  |  |  |
| --- | --- | --- |
|  | **개발 환경** | **서비스 운영 환경** |
| **안드로이드** | Android Studio 4.1  내부 에뮬레이터 | Android Icecream Sandwich 4.0이상 |
| **저장** | Github, SourceTree | - |
| **UI 이미지** | Adobe PhotoShop CC 2018 |

# **Ⅹ. 성능 시험 방법**

## **1. 성능 시험 방법**

Android Studio가 제공하는 에뮬레이터와 갤럭시 노트2를 이용하여 핵심 기능들을 시험한다. 오류가 발생 할 시 디버깅을 실시한다.

## **2. 설치 및 인수 방법**

안드로이드 마켓을 이용하여 사용자들에게 제공한다.

## **3. 유지 보수 방법**

사용자들에게 오류와 요구사항을 듣고 회의 후에 추가할 수 있도록 한다.

# **ⅩⅤ. 문서화**

문서화는 팀장이 각 팀원들에게 지시하고 팀원들은 팀장에게 정보와 구조를 얻고 작성한다.   
현재의 진행도와 목표에 따른 미래의 준비해야 할 것을 기준으로 문서화를 하며 프로젝트에 반영한다

# **ⅩⅤⅠ. 참고문헌 및 부록**

## **1. 참고 문헌**

소프트웨어 공학 최은만 저 – 정익사. 2007

## **2. 연구보고서**

2012~2018 스마트폰 사용. 한국갤럽 데일리 오피니언.

**소프트웨어 공학**

**Term Project**

**(요구분석서)**



**20164091 송희령**

**20164108 임재석**

**20164109 유승우**

**20164113 최상록**

# **Ⅰ. 개요**

## **1. 시스템 개요**

오늘날 고속도로 이용은 국내 다른 지역을 가는데 매우 필수적인 요소가 되었다. 이에 따라 고속도로에는 운전자들의 편의를 위해 다양한 편의시설이 존재한다. 먼 거리를 오고 가면서 고속도로 이용자들의 쉴 장소와 식사, 간식 등을 제공해주는 **휴게소와 주유소**가 존재하는데 고속도로 App을 통해 고속도로 교통상황 뿐만 아니라 편의시설들의 위치, 정보를 제공함으로써 운전자들에게 큰 도움을 줄 것이다.

## **2. 목표**

* 고속도로 정보 제공을 목표로 함
  + 고속도로 교통상황 정보 제공
  + 휴게소의 위치, 정보와 휴게소 내부의 주유소 정보 제공

# **Ⅱ. 기능적 목표**

## **1. 자료 흐름도**

사용자

고속도로 App

사용자

휴게소

교통상황

### **1-1. 교통 상황**

### 

교통정보 처리

교통량 데이터 API

Map

### **1-2. 휴게소**

편의 시설 현황

휴게시설

푸드 메뉴 현황

주유소 정보

위치 현황

교통정보 처리

휴게소 현황 API

휴게소

## **2. 소단위 명세서**

### 

프로세스 번호: 1

프로세스 이름: 교통정보 처리  
설명:  
{

교통량 API로부터 실시간 전국 교통량, 톨게이트 교통량  
 톨게이트간 통행시간 정보를 추출;  
 Display(현재 지도);  
 Display List에 보이기;

}

프로세스 번호: 2

프로세스 이름: 휴게소 자료 구분  
설명:  
{

휴게소 API로부터 휴게시설 현황, 휴게소 위치, 편의시설 현황,  
 푸드 메뉴 현황 조회 서비스, 주유소 서비스로부터 정보 추출  
 Switch(메뉴 클릭)  
 {

Case: 사용자가 클릭한 기능 Display()

}

}

# **Ⅲ. 기타 요구 및 제약 사항**

## **1. 성능 요구**

* 최소한의 자원소모
* 빠른 반응 속도와 처리 시간
* 최신의 정보 제공 보장
* 사용성이 용이한 직관적 인터페이스
* 접근성 상향

## **2. 성능 요구**

* 모바일 환경에서 실시간 인터넷 접속이 가능한 환경
  + 안드로이드 Icecream Sandwich 4.0이상의 버전이 포팅된 모바일 기기
* 개발 도구
  + Android Studio 4.1, 내부 에뮬레이터

## **3. 예외 조건 및 이의 처리**

* 입력 오류
  + API 라이브러리 대조 중 불일치 혹은 오류 발생 시 초기 검색화면으로 전환 후 초기화
* 네트워크 오류
  + 주기적 송수신 확인으로 연결확인.
  + 사용 중 네트워크 장애 발생 시 검색 조건 유지 및 다시 검색 실시
* GPS상의 오류
  + 제공하는 도로 관할 외 구역 사용시 사용 제한

## **4. 사용자 인터페이스**

* 메뉴 및 구성 요소들은 직관적인 구성으로 사용자가 쉽게 받아들일 수 있도록 한다.
* 고속도로 사고 상황과 사고 구간은 동시에 화면처리 가능하도록 한다.
* 구간 내 휴게소 안내 시 휴게소 관련 검색 블로그 화면과 유가 정보가 동시에 나올 수 있도록 한다.

## **5. 자원, 인력에 대한 제약조건**

* 자원 제약조건
  + 앱 사용자는 반드시 네트워크에 연결되어야 하며 위치 서비스를 실행 중이어야 한다.
  + 공공 API를 이용하는 것인 만큼 해당 API의 최신화된 정보 관리를 상시 진행해야 한다.
* 인력 제약조건
  + 개발자는 전반적인 고속도로와 휴게소에 대한 정보를 숙지해야 한다.
  + 접근하기 쉽도록 전반적인 인터페이스에 대한 이해가 필요하다.

# **Ⅳ. 기타 요구 및 제약 사항**

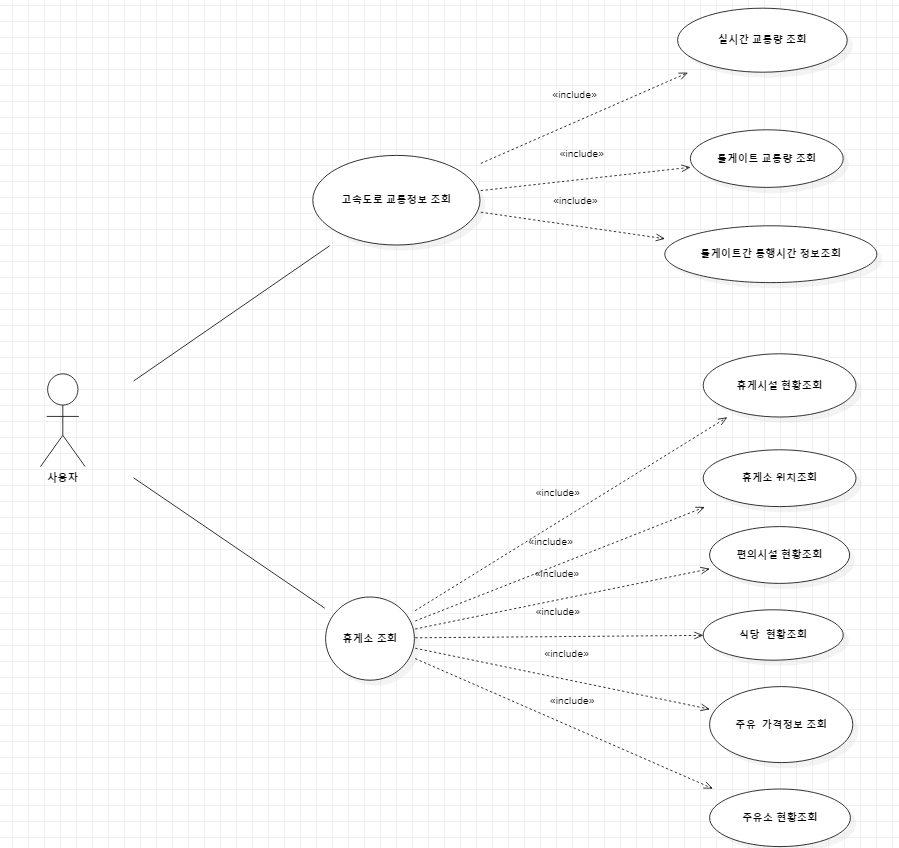
## **1. 객체 시험**

* API를 통한 데이터 로드 및 데이터 처리 확인.
* 노선에 따른 최신화 정보 사항 확인.
* 휴게소별 안내 시 검색 엔진 및 유가 정보 확인란 최신화 확인.

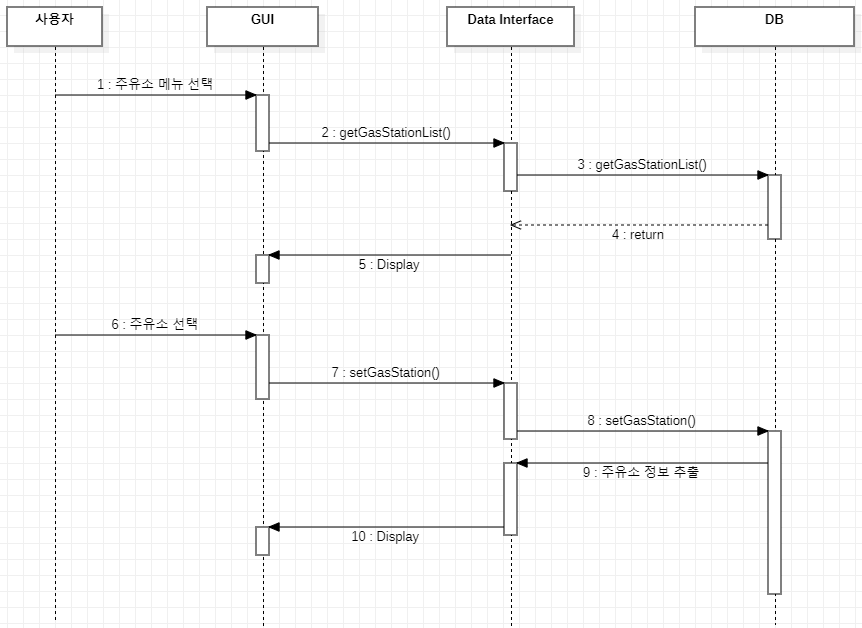
## **2. 최종 시험**

* 인터페이스 내의 모든 버튼 작동 확인
* 사용자 요구에 맞도록 제작 되었는지 확인
* 각 페이지에 맞는 정보와 이벤트 작동 확인
* 실제 외부 네트워크 상 오류 발생 시 처리 확인

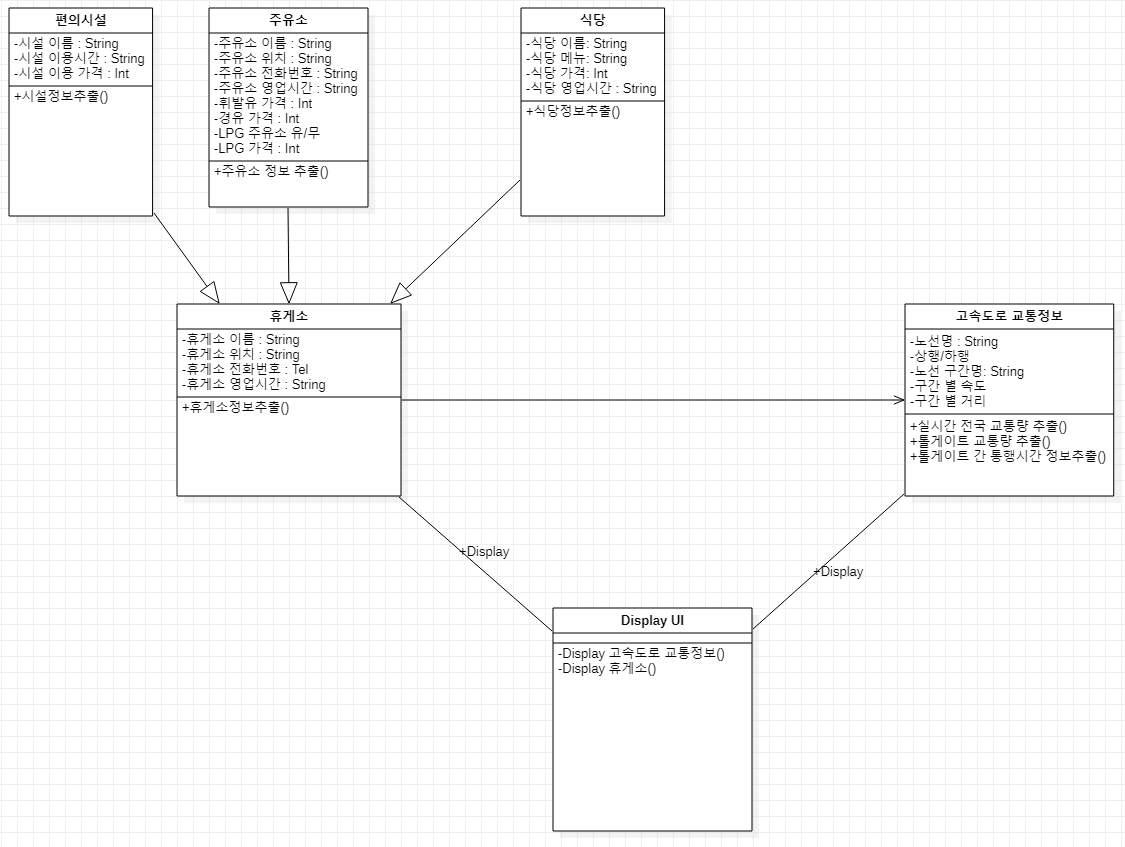
**1. 유스케이스 다이어그램**



**2. 시퀀스 다이어그램**



**3. 클래스 다이어그램**



**소프트웨어 공학**

**Term Project**

**(최종 보고서)**



**20164091 송희령**

**20164108 임재석**

**20164109 유승우**

**20164113 최상록**

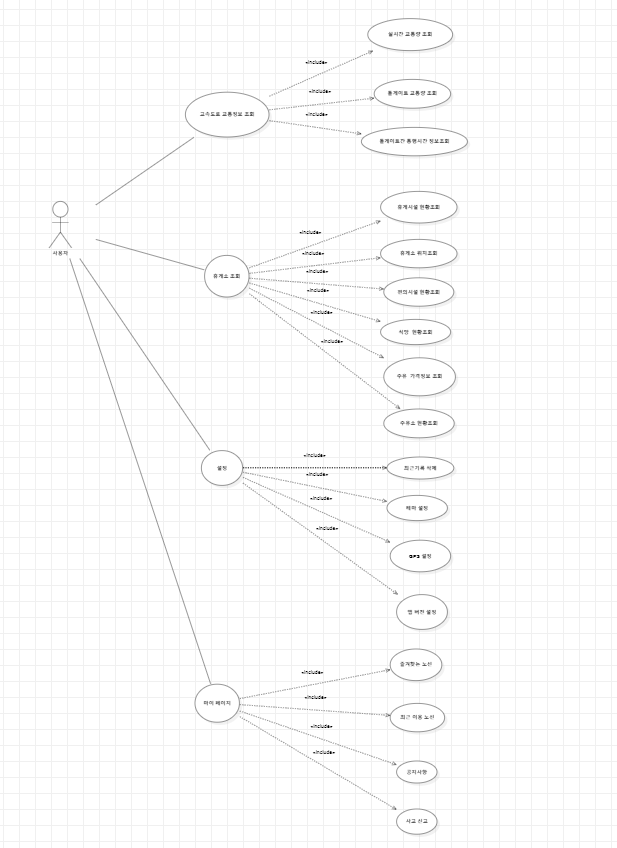
# **Ⅰ. 개요**

중간 프로젝트 발표 이후 교수님께서 피드백 해주신 내용을 기반으로 1차 프로젝트 회의를 진행하여 유즈케이스 다이어그램, 시퀀스 다이어그램, 클래스 다이어그램의 부족한 부분들을 보충하여 이를 기반으로 앱 개발을 하였음.

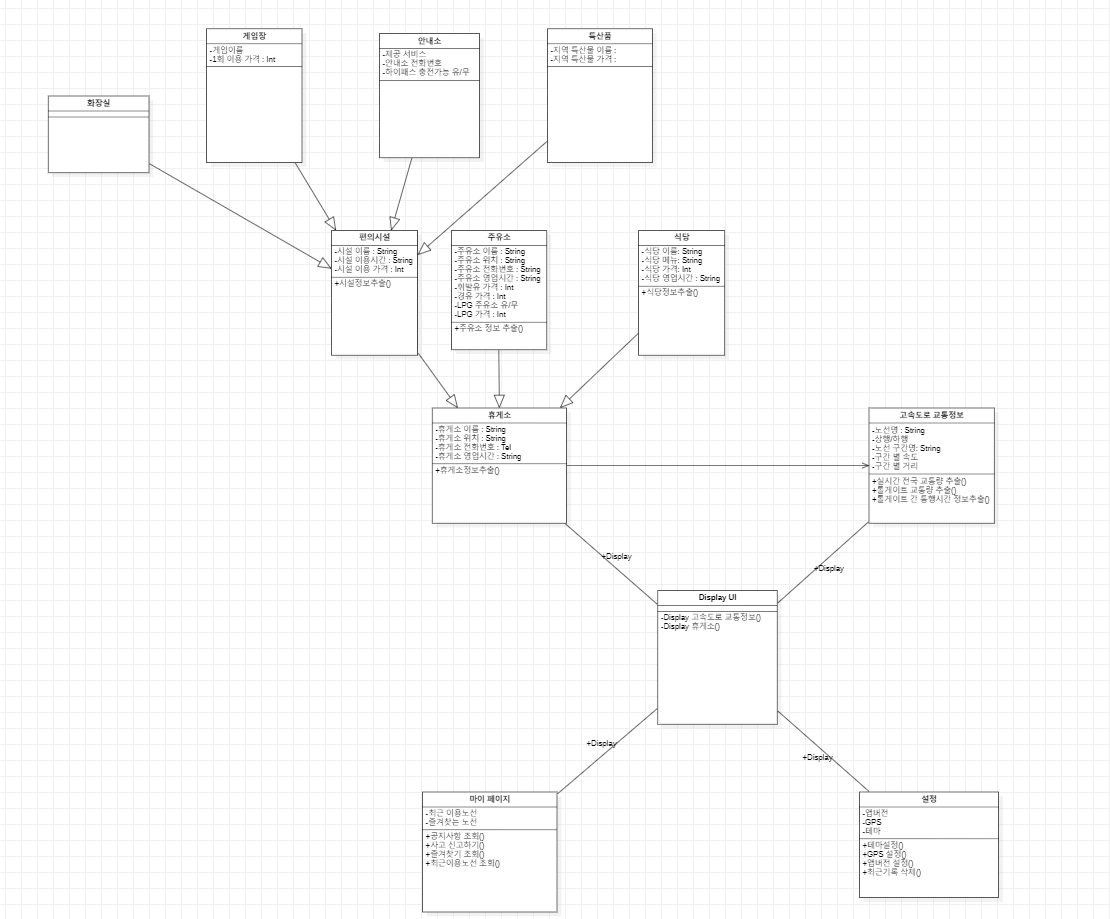
## **1. 1차 프로젝트 회의**

1차 프로젝트 회의는 유즈케이스, 시퀀스, 클래스 다이어그램의 부족한 부분을 보충하였으며, 다이어그램의 구성이 단출하다 느껴 기능적으로 조금 더 추가할 부분이 있는지에 관해 회의를 진행하였으며, 앱의 마이페이지와 설정관련 항목을 추가적으로 구성하여 조금 더 짜임새 있는 앱을 구성하려고 하였으며, 다이어그램을 추가적으로 구성한 뒤 다이어그램에 맞는 앱 구성에 대한 의견을 주고 받았으며, 도출한 결과를 단톡방에 정리하였음.

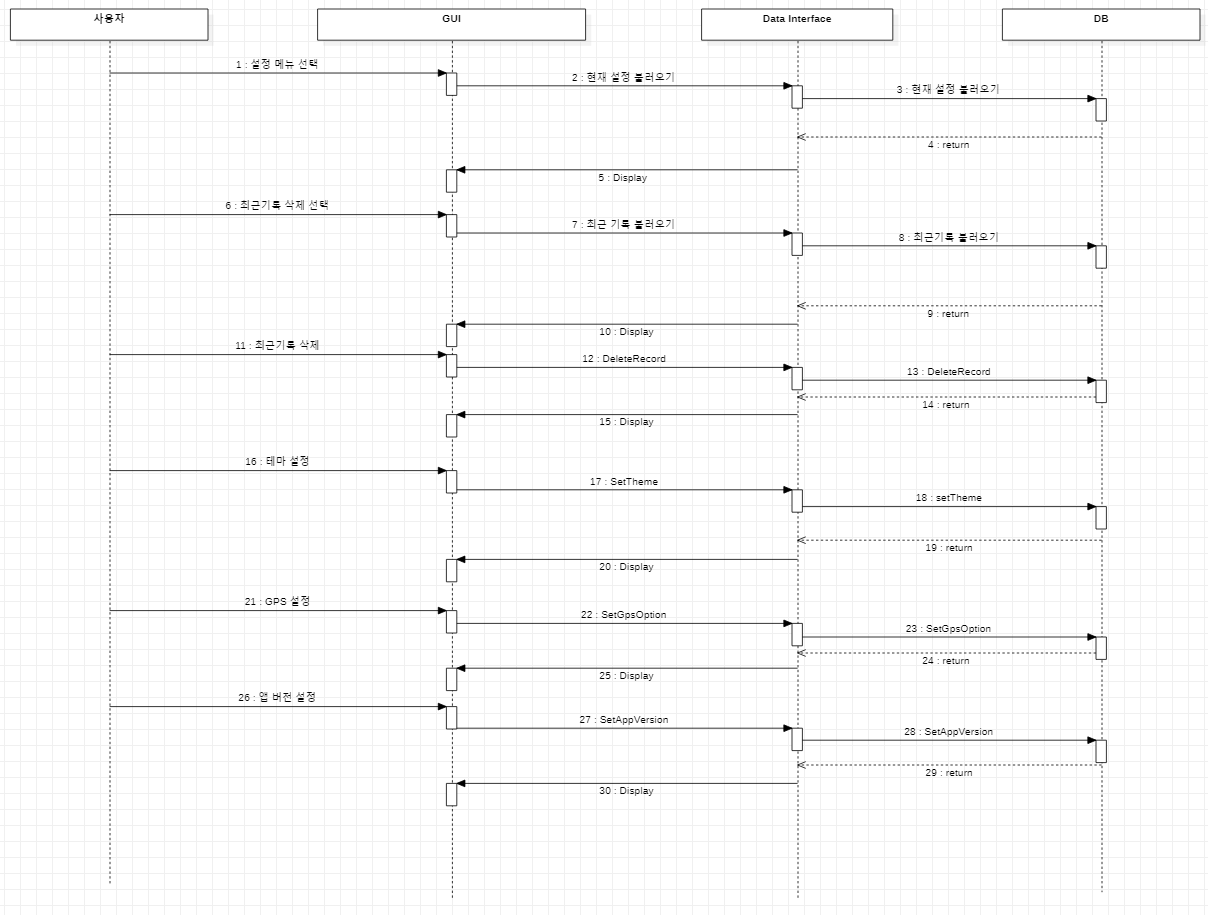
### **1-1. 유즈케이스 다이어그램**



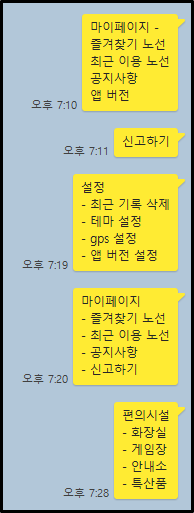
### **1-2. 클래스 다이어그램**



### **1-3. 시퀀스 다이어그램**



### **1-4. 페이지 구성**

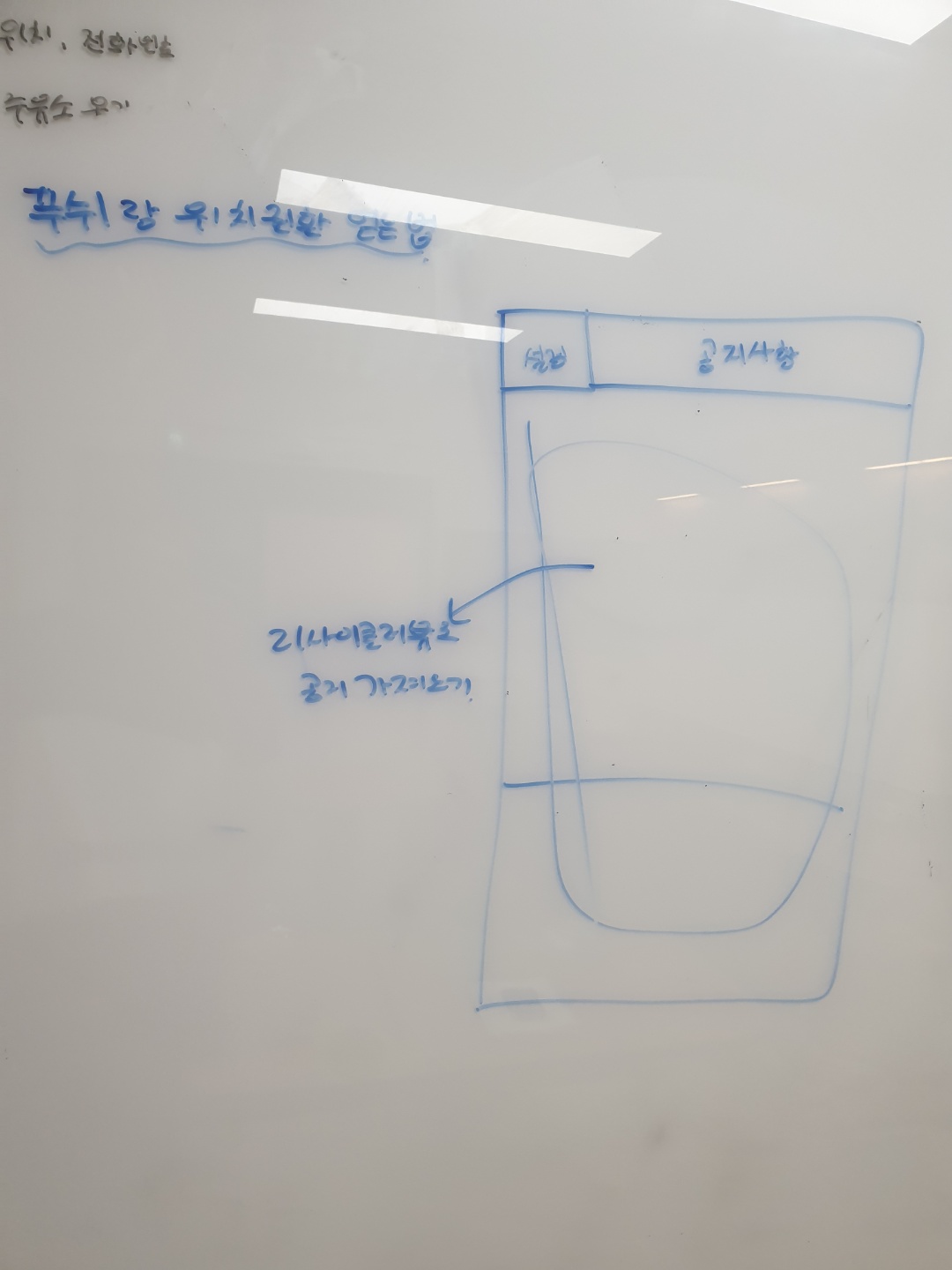


## **2. 2차 프로젝트 회의**

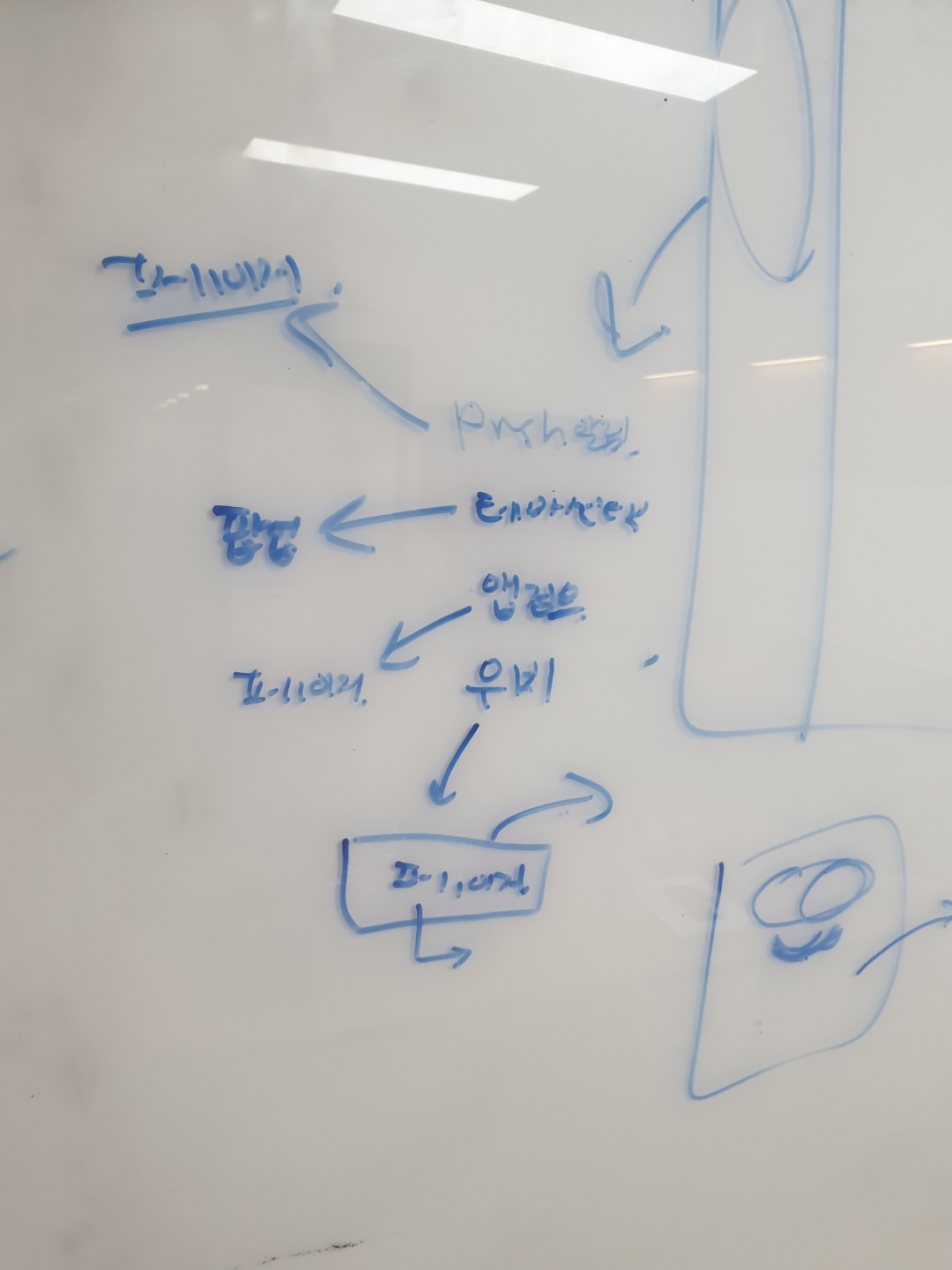
2차 프로젝트 회의는 액티비티 레이아웃 구성을 위하여 프로젝트 회의를 진행하였음. 레이아웃 구성에 관한 의견을 표현하는데 관해 비대면으로 진행할 경우 의견을 표출하는데 어려움이 있을 것 같아 대면으로 회의를 진행하였음. 회의는 전반적인 사용자의 경험요소들에 대해 직관적으로 구성하는 것을 목표로 하였으며, 각 항목 별 큰 틀을 먼저 잡은 후, 최대한 깔끔하고 사용자의 접근이 쉽도록 세부 요소들을 구성함.

### **2-1. 액티비티 레이아웃**

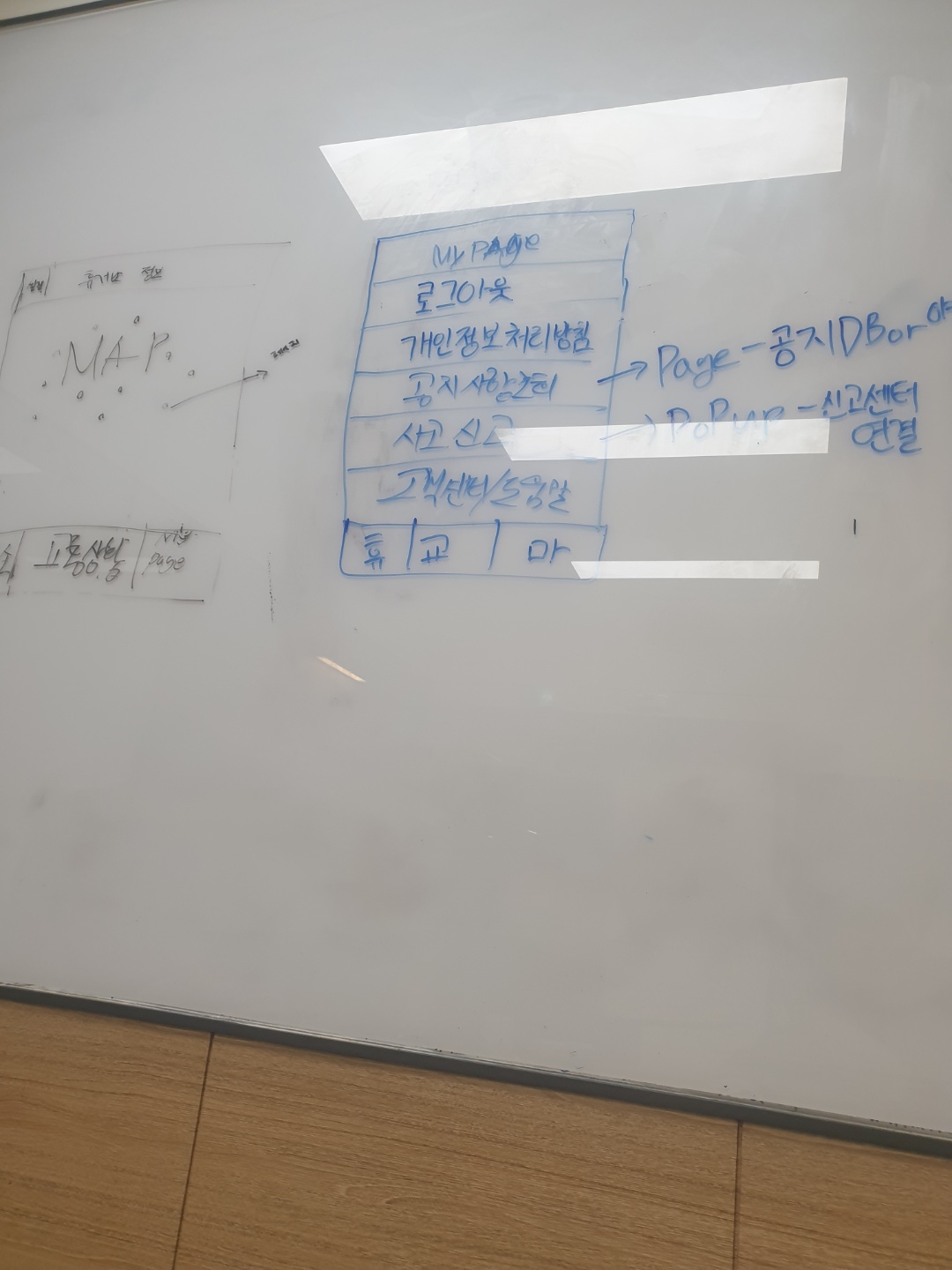
**2-1-1. 공지사항 레이아웃**



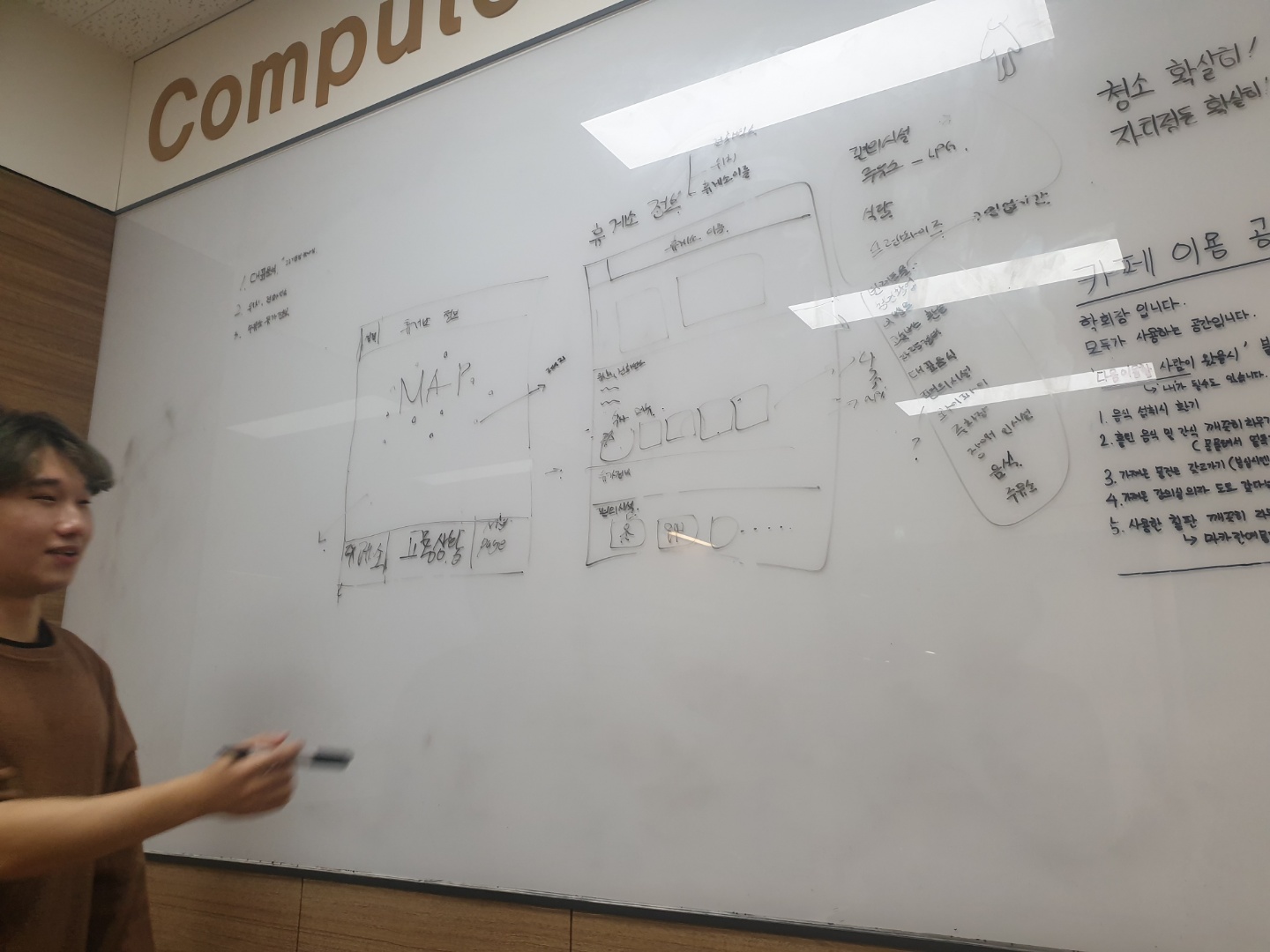
**2-1-2. 설정 레이아웃**



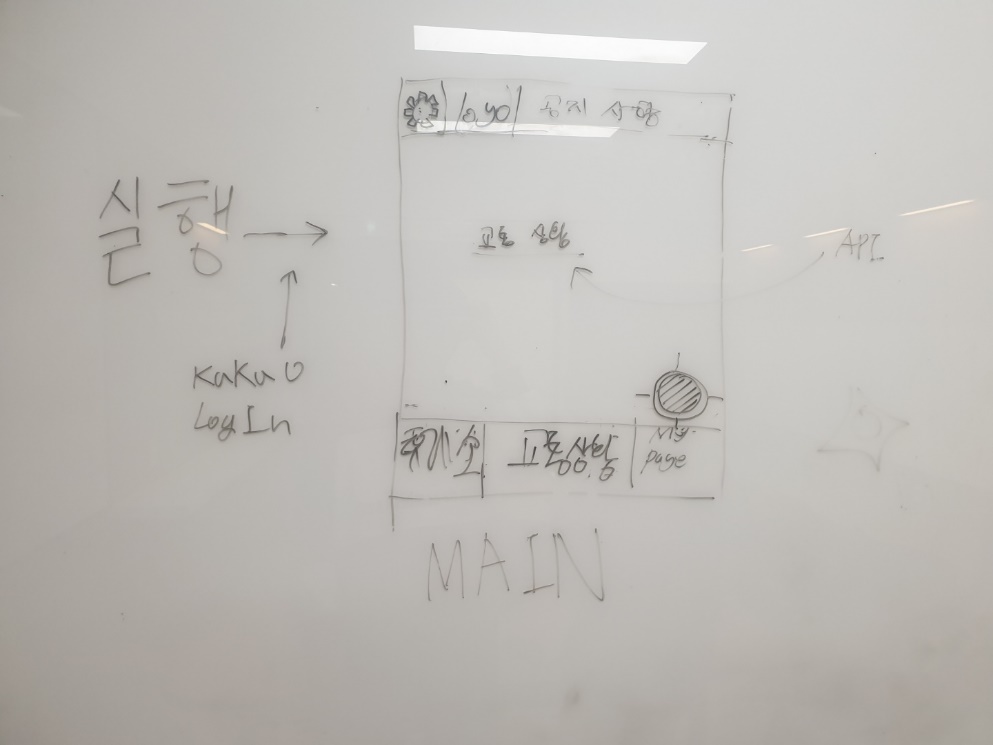
**2-1-3. 마이페이지 레이아웃**



**2-1-4. 휴게소 레이아웃**



**2-1-5. 교통상황 레이아웃**

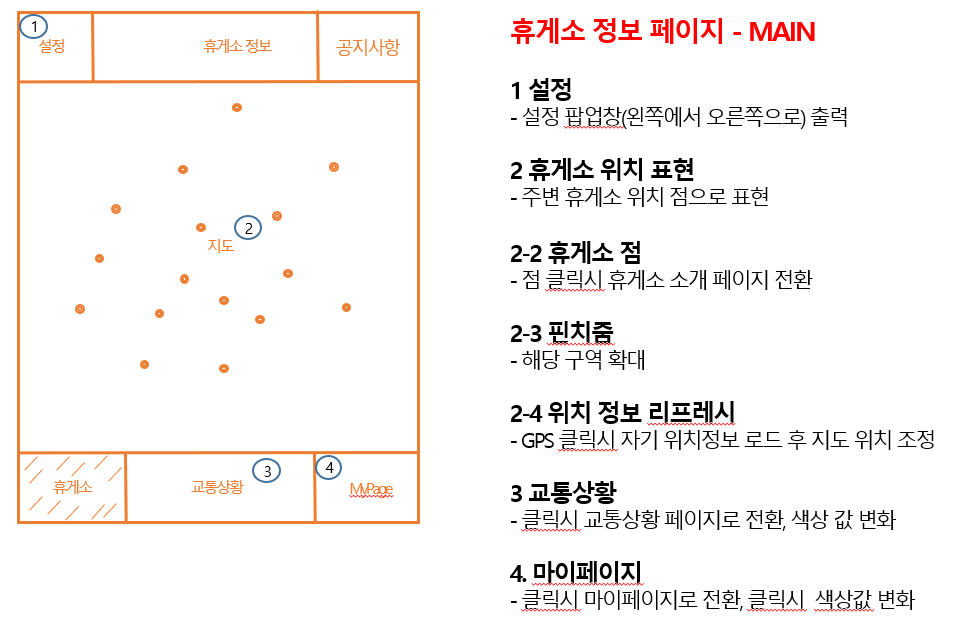


## **3. 3차 프로젝트 회의**

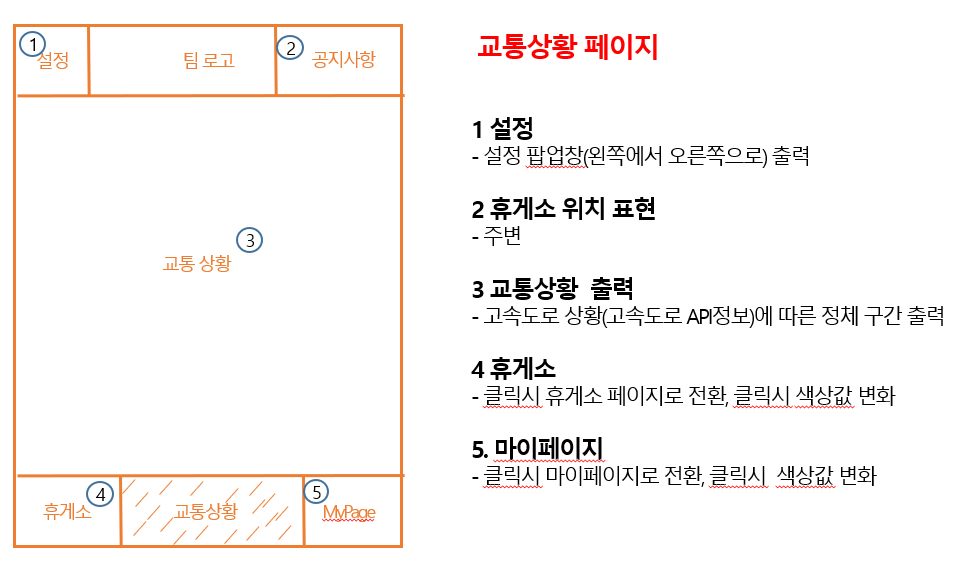
2차 프로젝트 회의에서 구성했던 각 레이아웃에 대한 이벤트를 논의하기 위해 회의를 진행하였음.

각 페이지마다 구성했던 요소들에 관해 기술적으로 가능한 지에 관하여도 논의하였으며, 요소들의 증감에 대한 토의 또한 같이 진행하였음. 초기 프로젝트 구상 시 카카오 계정을 연동하여 로그인 이벤트를 구현하려고 하였으나 로그인을 통한 앱에 추가적인 요인은 없을 것이라 판단하여 로그인에 관한 이벤트를 제거하기로 하였음.

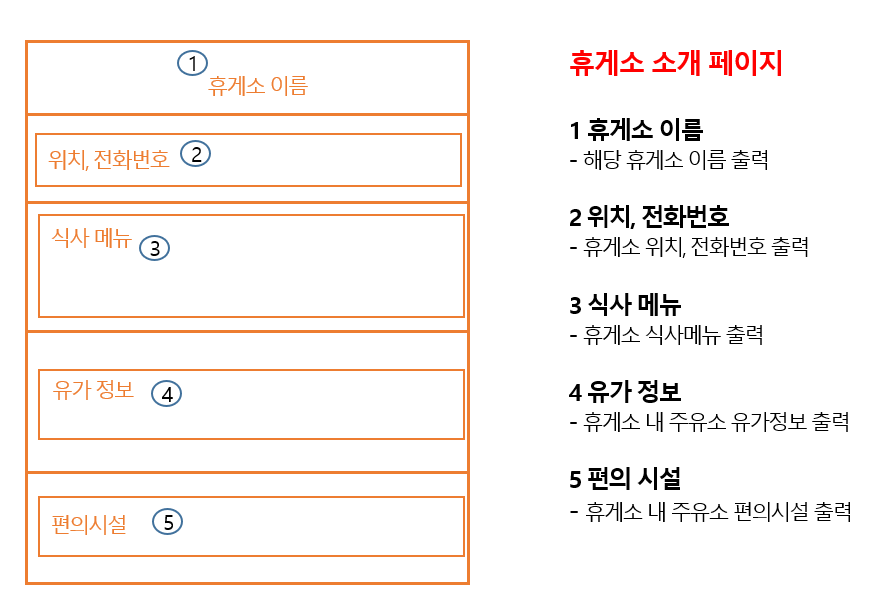
### **3-1. 휴게소 정보 페이지 이벤트 처리**



### **3-2. 교통상황 페이지 이벤트 처리**



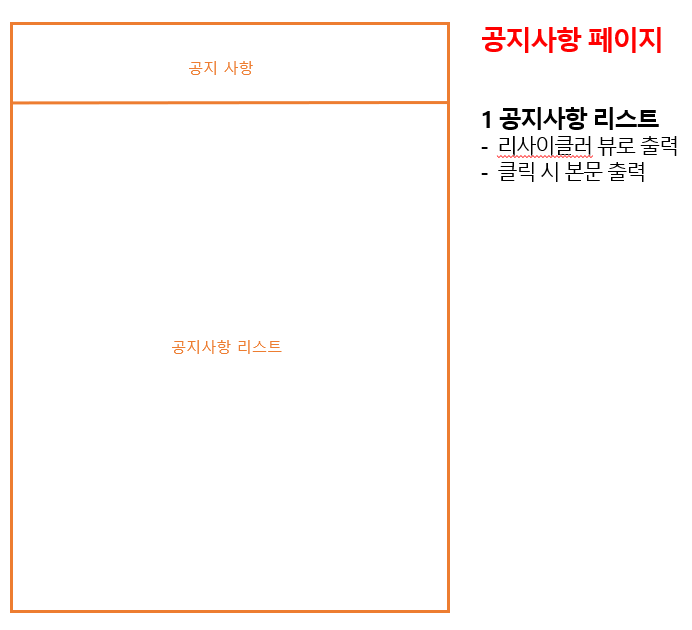
### **3-3. 휴게소 소개 페이지 이벤트 처리**



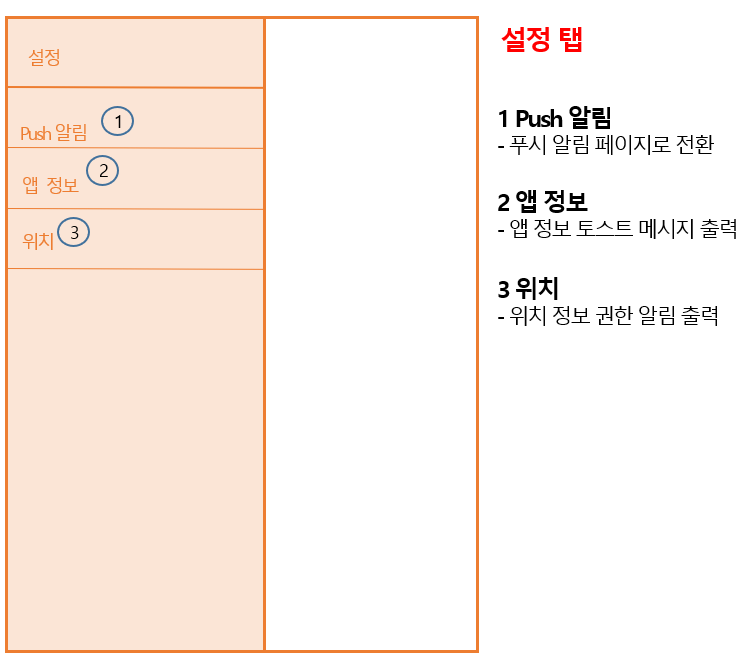
### **3-4. 마이페이지 이벤트 처리**



### **3-5. 공지사항 페이지 이벤트 처리**



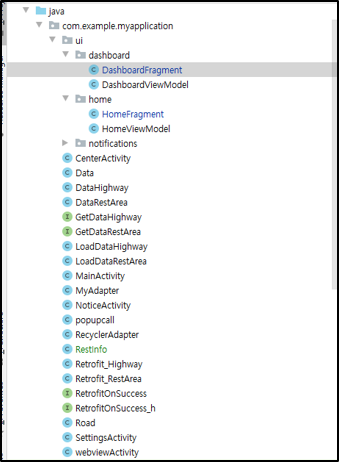
### **3-6. 설정 이벤트 처리**



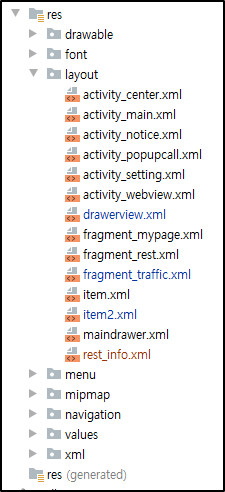
# **Ⅱ. 구현**

## **1. 클래스와 리소스 파일**

### **1-1. 클래스**

****

### **1-2. 리소스 파일**

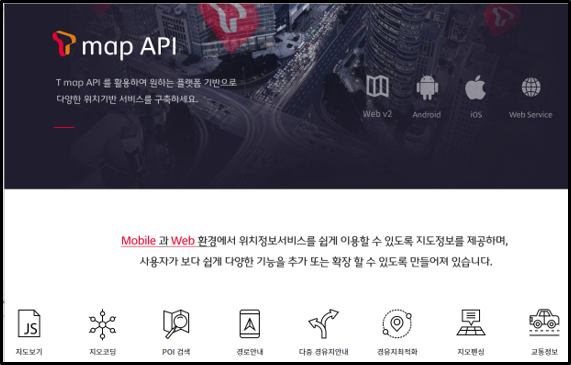
****

## **2. 데이터 소스**

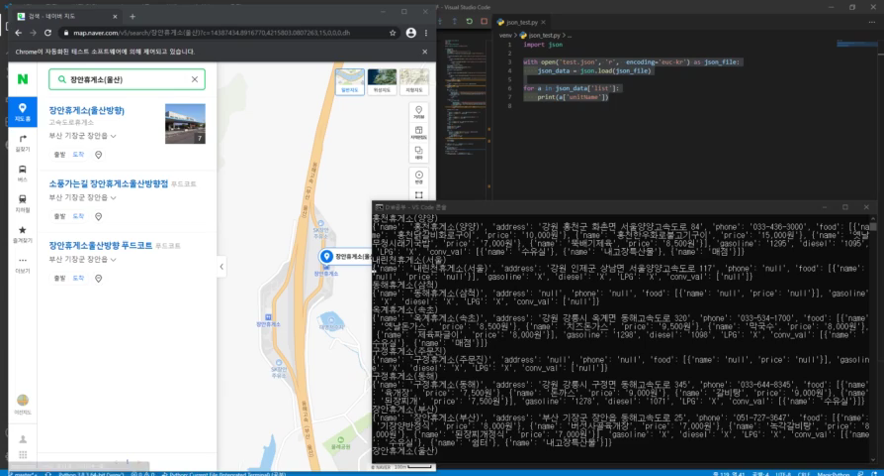
### **2-1. 도로공사 API**

****

### **2-2. T MAP API**

****

### **2-3. 휴게소 정보 크롤링 (네이버 지도)**

****

## **3. 코드**

### **3-1. 휴게소 정보**

**//레트로핏2를 GET방식으로 데이터를 가져옴  
//해당 API는 KEY값과 출력 형식을 요구함  
public interface GetDataRestArea {  
 @GET("locationinfoRest?")  
 Call<LoadDataRestArea> getInstanceRestArea(@Query("key") String key, @Query("type") String type,@Query("numOfRows")String count, @Query("pageNo")String page);  
}**

**//휴게소API를 가져오기위한 레트로핏 객체 생성  
//레트로핏 객체의 남용을 막아줌  
public class Retrofit\_RestArea {  
 private static final String *baseUrl* ="https://data.ex.co.kr/openapi/locationinfo/";  
 private static Retrofit *retrofit* = null;  
  
 private Retrofit\_RestArea(){}  
  
 public static Retrofit getInstance(){  
 if(*retrofit* == null){  
 *retrofit* = new Retrofit.Builder()  
 .baseUrl(*baseUrl*)  
 .addConverterFactory(GsonConverterFactory*.create*())  
 .build();  
 }  
 return *retrofit*;  
 }  
}**

**//휴게소 API를 가져올 시 필요한 클래스 형태  
//Serializable을 이용하여 객체를 직렬화  
public class DataRestArea implements Serializable {  
  
 String unitName;  
 String routeName;  
 String xValue;  
 String yValue;  
 String numOfRows;  
 String pageNo;  
  
 public DataRestArea(String unitName, String routeName, String xValue, String yValue, String numOfRows, String pageNo) {  
 this.unitName = unitName;  
 this.routeName = routeName;  
 this.xValue = xValue;  
 this.yValue = yValue;  
 this.numOfRows = numOfRows;  
 this.pageNo = pageNo;  
 }  
  
 public String getUnitName() {  
 return unitName;  
 }  
  
 public void setUnitName(String unitName) {  
 this.unitName = unitName;  
 }  
  
 public String getRouteName() {  
 return routeName;  
 }  
  
 public void setRouteName(String routeName) {  
 this.routeName = routeName;  
 }  
  
 public String getxValue() {  
 return xValue;  
 }  
  
 public void setxValue(String xValue) {  
 this.xValue = xValue;  
 }  
  
 public String getyValue() {  
 return yValue;  
 }  
  
 public void setyValue(String yValue) {  
 this.yValue = yValue;  
 }  
  
 public String getNumOfRows() {  
 return numOfRows;  
 }  
  
 public void setNumOfRows(String numOfRows) {  
 this.numOfRows = numOfRows;  
 }  
  
 public String getPageNo() {  
 return pageNo;  
 }  
  
 public void setPageNo(String pageNo) {  
 this.pageNo = pageNo;  
 }  
}**

**//휴게소 정보 로드  
//휴게소API를 사용시 최대 가져올 수 있는 정보가 99개, 총 정보는 226개이므로 3번 반복해야함.  
public void Load\_RestArea\_1(){  
 /\*레트로핏을 이용하여 데이터 가져오기\*/  
 GetDataRestArea service1 = Retrofit\_RestArea*.getInstance*().create(GetDataRestArea.class);  
 Call<LoadDataRestArea>call = service1.getInstanceRestArea(key, "json","99","1");  
 //레트로핏 비동기 실행  
 call.enqueue(new Callback<LoadDataRestArea>() {  
 @Override  
 //응답 시  
 public void onResponse(Call<LoadDataRestArea> call, Response<LoadDataRestArea> response) {  
 if(response.isSuccessful()){  
 result2 = (LoadDataRestArea)response.body();  
 if(result2.getList().size() != 0){  
 for (int i = 0; i < result2.getList().size(); i++) {  
 result\_data2.add(result2.getList().get(i));  
 }  
 }  
 //비동기의 문제로 인터페이스를 구현하여 사용  
 onSuccess(result\_data2);  
 }  
 else{  
 Log*.v*("결과는 : ","실패");  
 }  
 }  
  
 @Override  
 public void onFailure(Call<LoadDataRestArea> call, Throwable t) {  
 Log*.d*("결과는 : ","서버에서 부터 실패");  
 }  
 });  
}**

**//json 객체를 이용하여 JSON파싱  
JSONObject root = (JSONObject) new JSONTokener(jsonData).nextValue();  
JSONArray jsonArray = new JSONArray(root.getString("data"));  
for(int i = 0; i<jsonArray.length(); i++){  
 JSONObject data = jsonArray.getJSONObject(i);  
 String name = data.getString("name");  
 String address = data.getString("address");  
 String phone = data.getString("phone");  
 String gasoline = data.getString("gasoline");  
 String diesel = data.getString("diesel");  
 String LPG = data.getString("LPG");  
  
 JSONArray jsonArray1 = data.getJSONArray("food");  
 for(int j = 0; j<jsonArray1.length(); j++){  
 JSONObject data2 = jsonArray1.getJSONObject(j);  
 String temp2 = data2.getString("name");  
 String temp3 = data2.getString("price");  
 }  
  
 JSONArray jsonArray2 = data.getJSONArray("conv\_val");  
 for(int j = 0; j<jsonArray2.length(); j++){  
 JSONObject data3 = jsonArray2.getJSONObject(j);  
 String temp2 = data3.getString("name");  
 }  
  
 //만약 인텐트로 넘어온 이름과 같다면?  
 if(Name.equals(name)){  
 t2.setText(address);  
 t3.setText(phone);  
 t4.setText(gasoline);  
 }  
}**

**//JSON파일 형식**

**{  
 "name": "만남의광장",  
 "address": "서울 서초구 양재대로12길 73-71",  
 "phone": "02-578-3372",  
 "food": [  
 {  
 "name": "말죽거리 소고기국밥",  
 "price": "6,500원"  
 },  
 {  
 "name": "제육돌솥 비빔밥",  
 "price": "7,500원"  
 },  
 {  
 "name": "옛날돈까스",  
 "price": "8,000원"  
 },  
 {  
 "name": "생선까스",  
 "price": "8,000원"  
 }  
 ],  
 "gasoline": "1,289원",  
 "diesel": "1,089원",  
 "LPG": "848원",  
 "conv\_val": [  
 {  
 "name": "샤워실"  
 },  
 {  
 "name": "수유실"  
 },  
 {  
 "name": "쉼터"  
 }  
 ]  
}**

### **3-2. 교통상황**

**//레트로핏2를 GET방식으로 데이터를 가져옴  
//해당 API는 KEY값과 출력 형식을 요구함  
public interface GetDataHighway {  
 @GET("trafficAmountByCongest?")  
 Call<LoadDataHighway>getInstanceHighway(@Query("key") String key,@Query("type") String type);  
}**

**//고속도로 정체 상황 API를 가져오기위한 레트로핏 객체 생성  
//레트로핏 객체의 남용을 막아줌  
public class Retrofit\_Highway {  
  
 private static final String *baseUrl* ="https://data.ex.co.kr/openapi/odtraffic/";  
 private static Retrofit *retrofit* = null;  
  
 private Retrofit\_Highway(){}  
  
 public static Retrofit getInstance(){  
 if(*retrofit* == null){  
 *retrofit* = new Retrofit.Builder()  
 .baseUrl(*baseUrl*)  
 .addConverterFactory(GsonConverterFactory*.create*())  
 .build();  
 }  
 return *retrofit*;  
 }  
}**

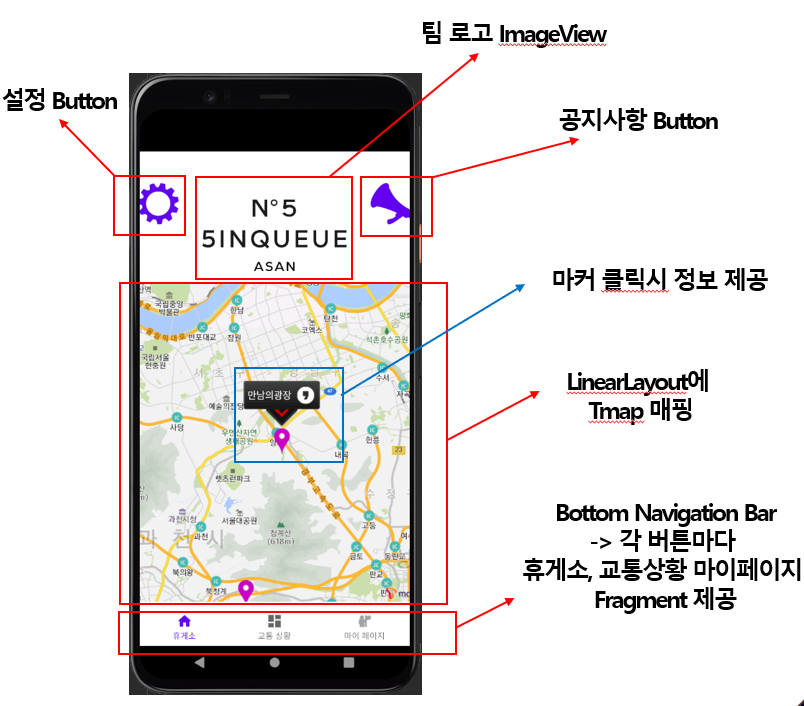
**//고속토로 정체 상황 API를 가져올 시 필요한 클래스 형태  
//Serializable을 이용하여 객체를 직렬화  
public class DataHighway implements Serializable {  
 public DataHighway(String stdHour, String routeNo, String routeName, String updownTypeCode, String vdsId, String trafficAmout, String shareRatio, String conzoneId, String conzoneName, String stdDate, String speed, String timeAvg, String grade) {  
 this.stdHour = stdHour;  
 this.routeNo = routeNo;  
 this.routeName = routeName;  
 this.updownTypeCode = updownTypeCode;  
 this.vdsId = vdsId;  
 this.trafficAmout = trafficAmout;  
 this.shareRatio = shareRatio;  
 this.conzoneId = conzoneId;  
 this.conzoneName = conzoneName;  
 this.stdDate = stdDate;  
 this.speed = speed;  
 this.timeAvg = timeAvg;  
 this.grade = grade;  
 }  
  
 public String getStdHour() {  
 return stdHour;  
 }  
  
 public void setStdHour(String stdHour) {  
 this.stdHour = stdHour;  
 }  
  
 public String getRouteNo() {  
 return routeNo;  
 }  
  
 public void setRouteNo(String routeNo) {  
 this.routeNo = routeNo;  
 }  
  
 public String getRouteName() {  
 return routeName;  
 }  
  
 public void setRouteName(String routeName) {  
 this.routeName = routeName;  
 }  
  
 public String getUpdownTypeCode() {  
 return updownTypeCode;  
 }  
  
 public void setUpdownTypeCode(String updownTypeCode) {  
 this.updownTypeCode = updownTypeCode;  
 }  
  
 public String getVdsId() {  
 return vdsId;  
 }  
  
 public void setVdsId(String vdsId) {  
 this.vdsId = vdsId;  
 }  
  
 public String getTrafficAmout() {  
 return trafficAmout;  
 }  
  
 public void setTrafficAmout(String trafficAmout) {  
 this.trafficAmout = trafficAmout;  
 }  
  
 public String getShareRatio() {  
 return shareRatio;  
 }  
  
 public void setShareRatio(String shareRatio) {  
 this.shareRatio = shareRatio;  
 }  
  
 public String getConzoneId() {  
 return conzoneId;  
 }  
  
 public void setConzoneId(String conzoneId) {  
 this.conzoneId = conzoneId;  
 }  
  
 public String getConzoneName() {  
 return conzoneName;  
 }  
  
 public void setConzoneName(String conzoneName) {  
 this.conzoneName = conzoneName;  
 }  
  
 public String getStdDate() {  
 return stdDate;  
 }  
  
 public void setStdDate(String stdDate) {  
 this.stdDate = stdDate;  
 }  
  
 public String getSpeed() {  
 return speed;  
 }  
  
 public void setSpeed(String speed) {  
 this.speed = speed;  
 }  
  
 public String getTimeAvg() {  
 return timeAvg;  
 }  
  
 public void setTimeAvg(String timeAvg) {  
 this.timeAvg = timeAvg;  
 }  
  
 public String getGrade() {  
 return grade;  
 }  
  
 public void setGrade(String grade) {  
 this.grade = grade;  
 }  
  
 private String stdHour;  
 private String routeNo;  
 private String routeName;  
 private String updownTypeCode;  
 private String vdsId;  
 private String trafficAmout;  
 private String shareRatio;  
 private String conzoneId;  
 private String conzoneName;  
 private String stdDate;  
 private String speed;  
 private String timeAvg;  
 private String grade;  
}**

**//고속도로 정체 상황 가져오기  
public void refresh(){  
 /\*레트로핏을 이용하여 데이터 가져오기\*/  
 GetDataHighway service1 = Retrofit\_Highway*.getInstance*().create(GetDataHighway.class);  
 Call<LoadDataHighway>call = service1.getInstanceHighway(key, "json");  
 //레트로핏 비동기 실행  
 call.enqueue(new Callback<LoadDataHighway>() {  
 @Override  
 public void onResponse(Call<LoadDataHighway> call, Response<LoadDataHighway> response) {  
 if(response.isSuccessful()){  
 result = (LoadDataHighway)response.body();  
 if(result.getDhh().size() != 0) {  
 for (int i = 0; i < result.getDhh().size(); i++) {  
 result\_data.add(  
 new DataHighway(result.getDhh().get(i).getStdHour(),  
 result.getDhh().get(i).getRouteNo(),  
 result.getDhh().get(i).getRouteName(),  
 result.getDhh().get(i).getUpdownTypeCode(),  
 result.getDhh().get(i).getVdsId(),  
 result.getDhh().get(i).getTrafficAmout(),  
 result.getDhh().get(i).getShareRatio(),  
 result.getDhh().get(i).getConzoneId(),  
 result.getDhh().get(i).getConzoneName(),  
 result.getDhh().get(i).getStdDate(),  
 result.getDhh().get(i).getSpeed(),  
 result.getDhh().get(i).getTimeAvg(),  
 result.getDhh().get(i).getGrade()));  
 }  
 }  
 OnSuccess(result\_data);  
 }  
 else{  
 Log*.v*("결과는 : ","실패");  
 }  
 }  
 @Override  
 public void onFailure(Call<LoadDataHighway> call, Throwable t) {  
 Log*.d*("결과는 : ","서버 연결 실패!");  
 }  
 });**

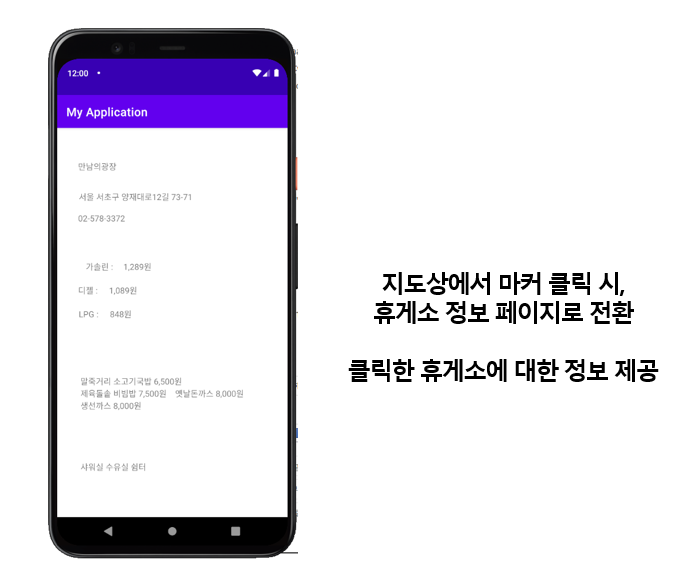
**//API는 정체인 구간만 나옴, 그러므로 원래 데이터와 비교하여 나와야함  
//등급은 높을수록 막힘.  
public void change\_data() {  
 if(result\_data.size() != 0) {  
 for (int i = 0; i < result\_data.size(); i++) {  
 for (int j = 0; j < list.size(); j++) {  
 //막힌 경로가 똑같이 있다면?  
 if (result\_data.get(i).getConzoneName().equals(list.get(j).getConzoneName())) {  
 list.get(j).setGrade(result\_data.get(i).getGrade());  
 }  
 }  
 }  
 }  
}**

## **4. 구현 결과**

### **4-1. 휴게소 페이지**



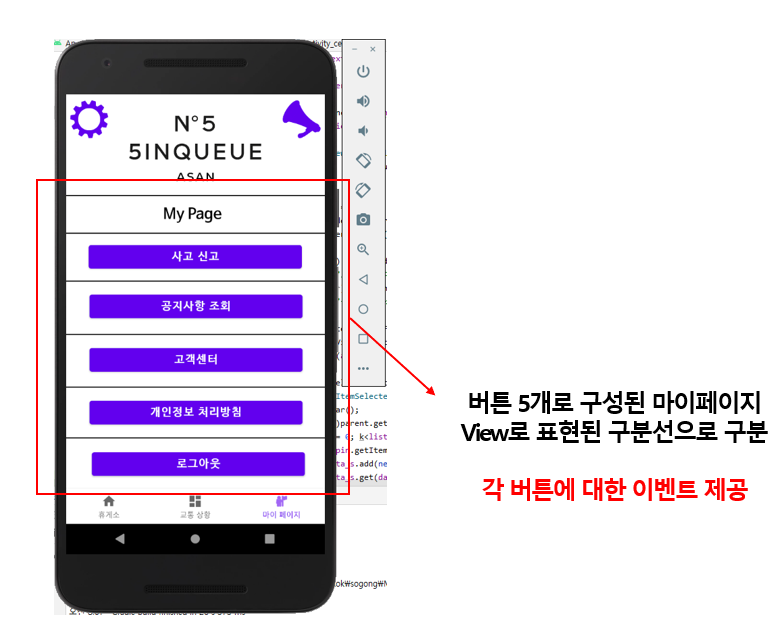
### **4-2. 휴게소 정보 페이지**

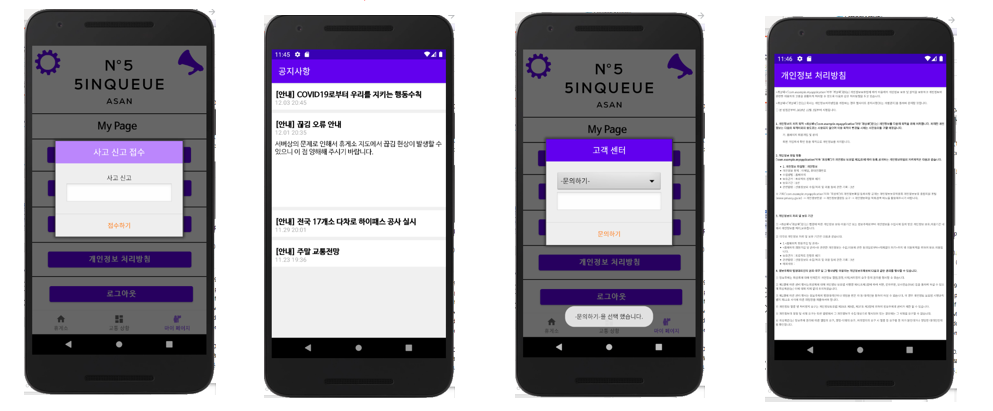


### **4-3. 교통상황 페이지**



### **4-4. 마이 페이지**





### **4-5. 설정**

