**오픈 소스 전문 프로젝트**

**Report #07 최종 보고서**

****

|  |  |
| --- | --- |
| **학과** | **컴퓨터공학과** |
| **학번** | **2018037002 손지현**  **2018037004 이진희**  **2018037025 이도희**  **2018037026 윤소영**  **2018037054 신지애** |
| **교수님** | **박수창 교수님** |
| **제출일** | **2020/06/17** |

**목 차**

1. **포켓 폴리스 소개**
2. **센서 사용 및 수집 정보**
3. **데이터 베이스**
4. **앱 기본 기능**
5. **MLP 주요 기능**

**1. 포켓 폴리스 소개**

* **GPS, 가속도 센서**를 이용한 치안용 어플리케이션
* 사용자가 목적지를 설정하여 귀가하는 동안 위급한 상황으로 감지되었을 때 **주저앉는다면,** 위치 정보를 경찰서와 사전 지정 대상자에게 전송하는 기능
* 목적지 설정 범위 내에서 귀가 완료 버튼을 눌렀을 시 안전 귀가 메시지를 사전 지정자에게 전송하는 기능
* 안전 귀가 메시지 url을 클릭하면 지도를 기반으로 한 위치 정보를 보여주고, 위치 공유까지 가능한 기능

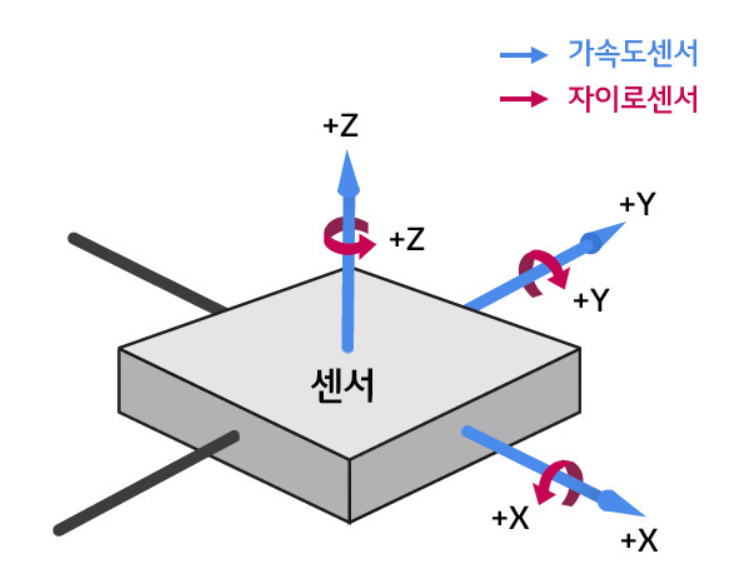
1. **센서 사용 및 수집 정보**

**STEP 1) 센서 활용**

1. **가속도 센서 정의**

* 물체의 속도의 변화(가속도)나 운동량의 변화(충격량) 등을 측정하는 센서이다. 물체의 운동 상태를 상세하게 감지할 수 있기 때문에 다양한 분야에서 필수적으로 사용되며, 그 활용도 또한 높다.

1. **가속도 센서 이해**



스마트폰의 기울기에 따라 가속도 센서 x/y/z축의 측정값이 변화함.

* 스마트폰이 주머니/가방 속에 있을 경우, 다양한 방향(x/y/z축)으로 가속도 센서 측정값이 나올 수 있기 때문에, 스마트폰 기울기를 다르게 하여 가속도 센서 측정값을 수집함.

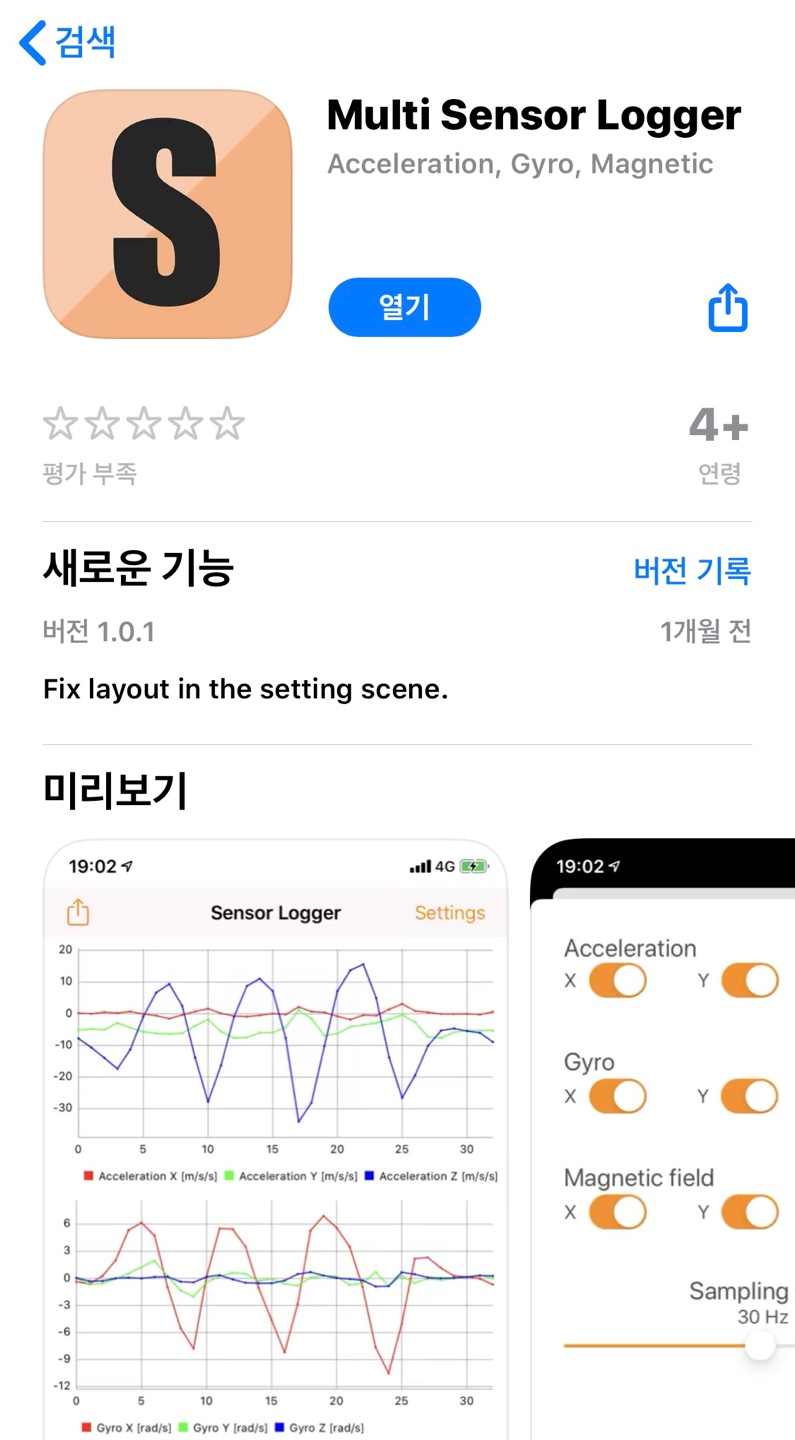
1. **가속도 센서 활용**

* 포켓폴리스는 스마트폰을 주머니/가방 속에 넣고 귀가하는 상황에서 위험 상황으로 감지될 경우, 주저앉음으로써 저장된 위치 정보를 전송 및 신고하는 치안 관련 어플.
* 즉 주저앉았음을 인식하기 위해, 물체의 운동 상태를 상세하게 감지할 수 있는 스마트폰에 내장된 가속도 센서를 활용.

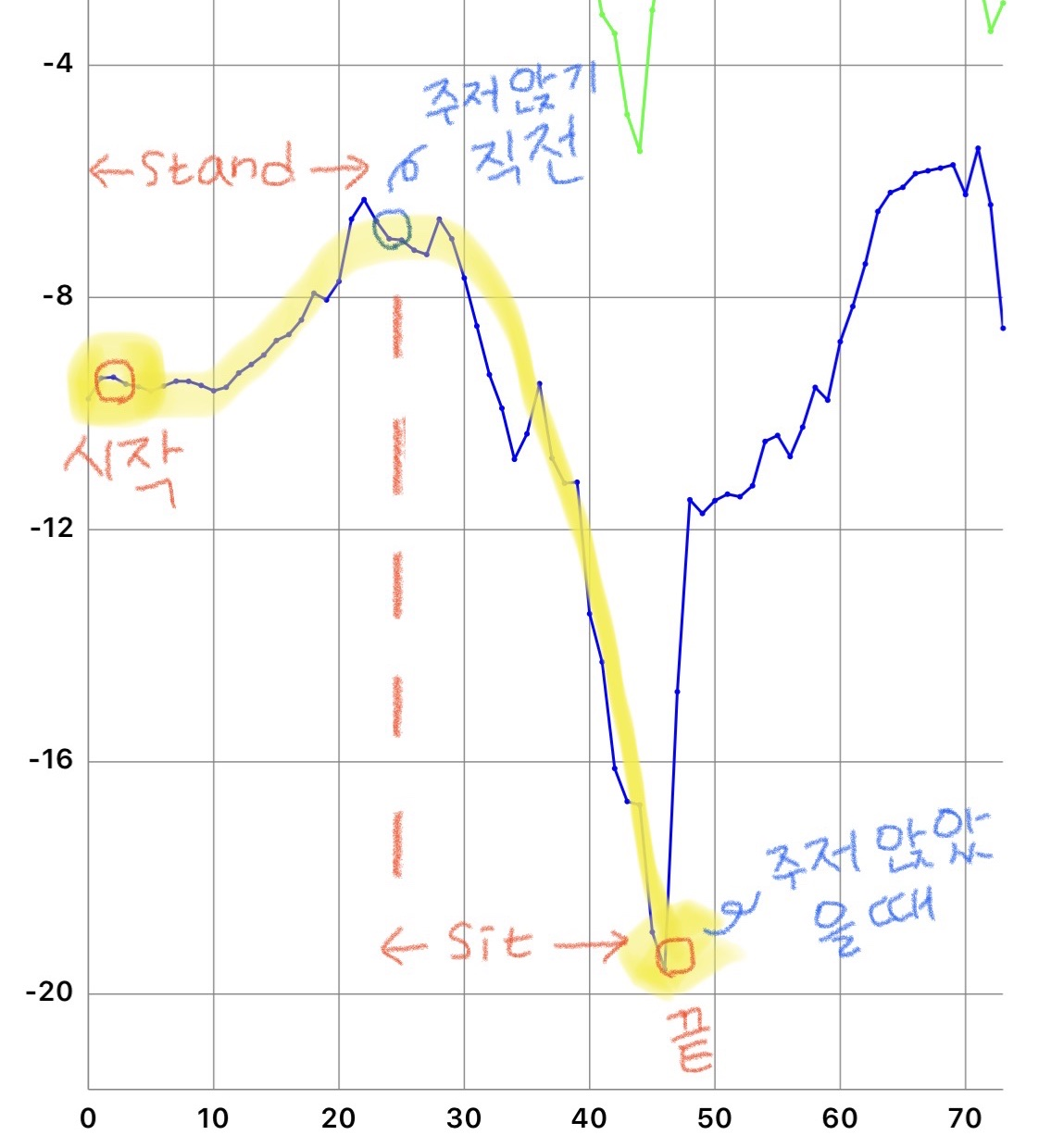
**STEP 2) 데이터 수집 정보**

1. **데이터 측정**

* 스마트폰 앱 Multi Sensor Logger을 통해 데이터를 측정.
* **Multi Sensor Logger**는 가속도, 자이로 스코프 및 자기장을 동시에 측정할 수 있는 로거 응용 프로그램. 이 앱은 측정 값을 CSV 파일로 출력 가능함.

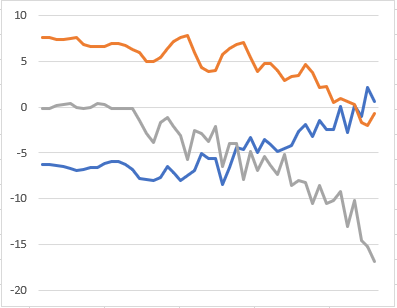


1. **데이터 수집**



* 가속도 그래프를 보았을 때, **서 있을 때의 값부터 주저 앉았을 때까지의 값**을 도출하여 CSV 파일로 데이터 수집
* 그 구간에서 시간에 따른 변화량 데이터 분석 및 머신 러닝 기법을 이용하여 사용자가 주저앉았을 때, 위험 상황으로 감지하여 서비스를 실행할 수 있도록 활용.

1. **데이터 수집 정보**



**<한번 앉았을 때의 그래프>**

* 주저앉기 시작할 때부터 주저앉았을 때까지의 데이터를 1000개 수집함.

1. **데이터 베이스**

* **Firebase을 활용하여 데이터 베이스 저장**
  + - * 1. **회원가입**
* 가입 조건에 맞아 회원 가입 성공 시, 입력한 회원 가입 정보를 받아옴

String email = user.getEmail();  
String uid = user.getUid();  
String pass= mPasswordText.getText().toString().trim();  
String name = mName.getText().toString().trim();

* 해쉬맵 테이블을 파이어베이스 데이터 베이스에 저장

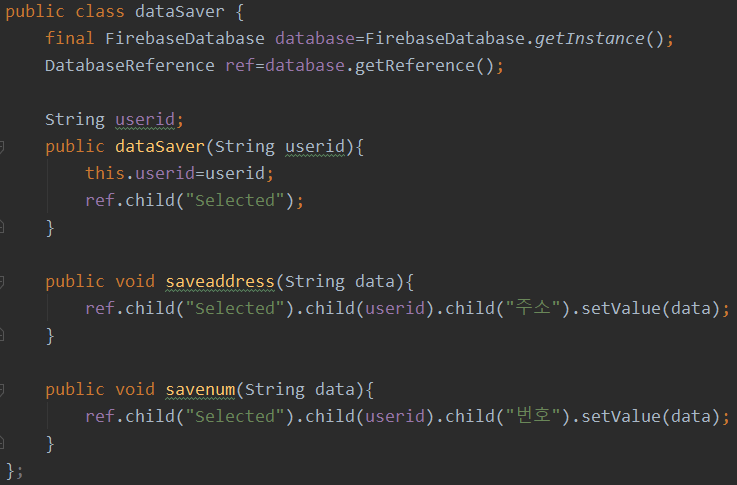
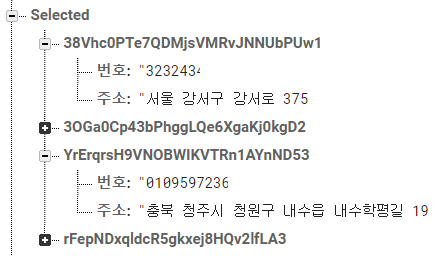
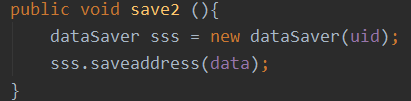
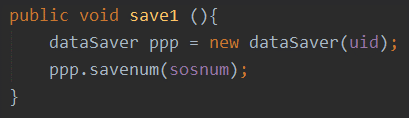
HashMap<Object, String> hashMap = new HashMap<>();  
  
hashMap.put("uid", uid);  
hashMap.put("name", name);  
hashMap.put("email", email);  
hashMap.put("password",pass);

* Users 경로에 uid 기준으로 회원가입 정보를 저장

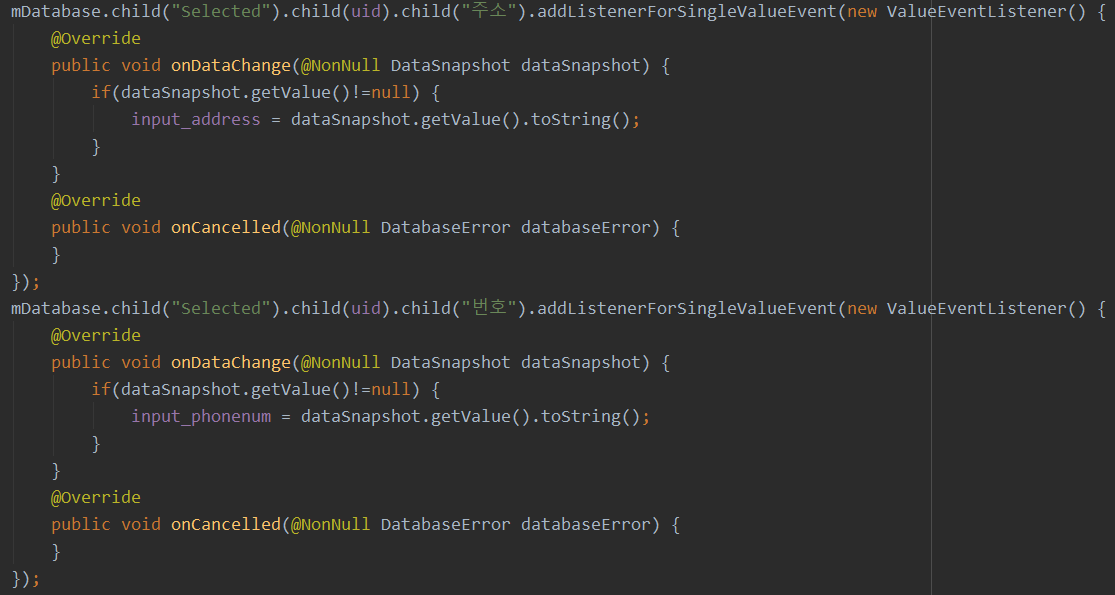
FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
DatabaseReference reference = database.getReference("Users");  
reference.child(uid).setValue(hashMap);



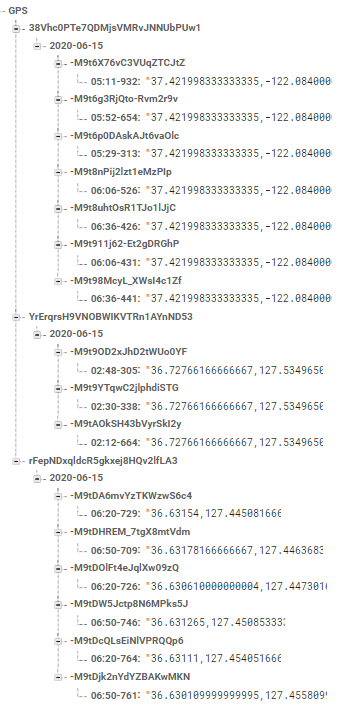
* + - * 1. **시작 페이지 사전 정보 설정**
* 데이터베이스 Selected 하위에 uid를 기준으로 사용자가 선택한 주소와 번호를 저장



* 기능 실행 시 사용자가 설정한 목적지와 연락처가 필요한 경우 데이터베이스에서 가져옴



GPS 위치 정보는 데이터 베이스의 GPS경로로 들어가게 되며 사용자 <GPS-uid-날짜(yyyy-mm-dd)-시간 별 위치정보(“위도, 경도")>의 구조로 저장된다.



현재 사용자의 uid를 가져와 해당 uid로 위치 정보를 저장한다.

FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
final String nowuser = mAuth.getCurrentUser().getUid();  
gpsSaver gps = new gpsSaver(nowuser);

GpsSaver의 saveGPS는 GPS의 정보를 업데이트 하는데 이때 Data 5 클래스를 사용하여 <시간: gps 정보>로 map을 만들어 업데이트 한다.

public void saveGPS(String gps){  
  
 String key = ref.child("GPS").child(userid).push().getKey();  
 Data5 data=new Data5(gps);  
 Map<String,Object> gpsValue=data.toMap();  
 Map<String,Object> childUpdates=new HashMap<>();  
 childUpdates.put("/GPS/"+userid+'/'+time+'/'+key,gpsValue);  
 ref.updateChildren(childUpdates);  
}

gps.saveGPS(lati + "," + longi);

locationManager와 locationListener를 사용하여 location을 받아 각각 위도와 경도를 다음과 같은 형태로 데이터 베이스에 저장되게 한다.

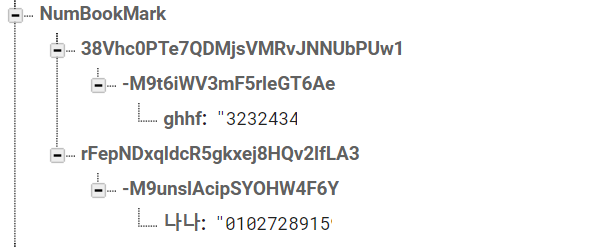
* + - * 1. **환경설정**
    1. **연락처 즐겨찾기**
* 데이터베이스에 NumBookMark를 추가하고 uid를 받아와 사용자별로 관리할 수 있게 함

public numbookmark(String uid){  
 this.user=nowuser;  
 ref.child("NumBookmark").child(uid);  
}

* NumBookMark와 uid 하위에 랜덤 값을 넣고 입력 받은 두 값을 저장

public void savenbm(String snbm\_info, String snbm){  
 String key = ref.child("NumBookmark").child(user).push().getKey();  
 Data3 data3=new Data3(snbm, snbm\_info);  
 Map<String,Object> Value=data3.toMap();  
 Map<String,Object> childUpdates=new HashMap<>();  
 childUpdates.put("/NumBookMark/"+user+'/'+key,Value);  
 ref.updateChildren(childUpdates);  
}

* 연락처의 별칭과 연락처를 key, value로 저장

class Data3{  
 public String snbm;  
 public String snbm\_info;  
  
 public Data3(){}  
 public Data3(String snbm, String snbm\_info){  
 this.snbm=snbm;  
 this.snbm\_info=snbm\_info;  
 }  
  
 public Map<String, Object> toMap(){  
 HashMap<String,Object> result=new HashMap<>();  
 result.put(snbm\_info,snbm);  
 return result;  
 }  
}

**uid**

**등록한 연락처**

* + 1. **목적지 즐겨찾기**
* 데이터 베이스에 DesBookmark를 추가하고 uid를 받아와 사용자별로 관리할 수 있게 함

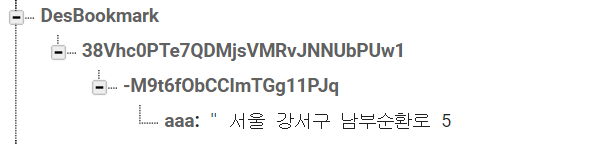
public desbookmark(String uid){  
 this.user=nowuser;  
 ref.child("DesBookmark").child(uid);  
}

* DesBookmark와 uid 하위에 랜덤 값을 넣고 입력 받은 두 값을 저장

public void savedbm(String sdbm\_info, String sdbm){  
  
 String key = ref.child("NumBookmark").child(user).push().getKey();  
 Data4 data=new Data4(sdbm, sdbm\_info);  
 Map<String,Object> gpsValue=data.toMap();  
 Map<String,Object> childUpdates=new HashMap<>();  
 childUpdates.put("/DesBookmark/"+user+'/'+key,gpsValue);  
 ref.updateChildren(childUpdates);  
 }  
};

* 목적지의 별칭과 목적지를 key, value로 저장

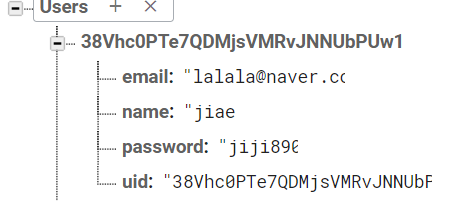
class Data4{  
 public String sdbm;  
 public String sdbm\_info;  
  
 public Data4(){}  
 public Data4(String sdbm, String sdbm\_info){  
 this.sdbm=sdbm;  
 this.sdbm\_info=sdbm\_info;  
 }  
  
 public Map<String, Object> toMap(){  
 HashMap<String,Object> result=new HashMap<>();  
 result.put(sdbm,sdbm\_info);  
 return result;  
 }  
}



**uid**

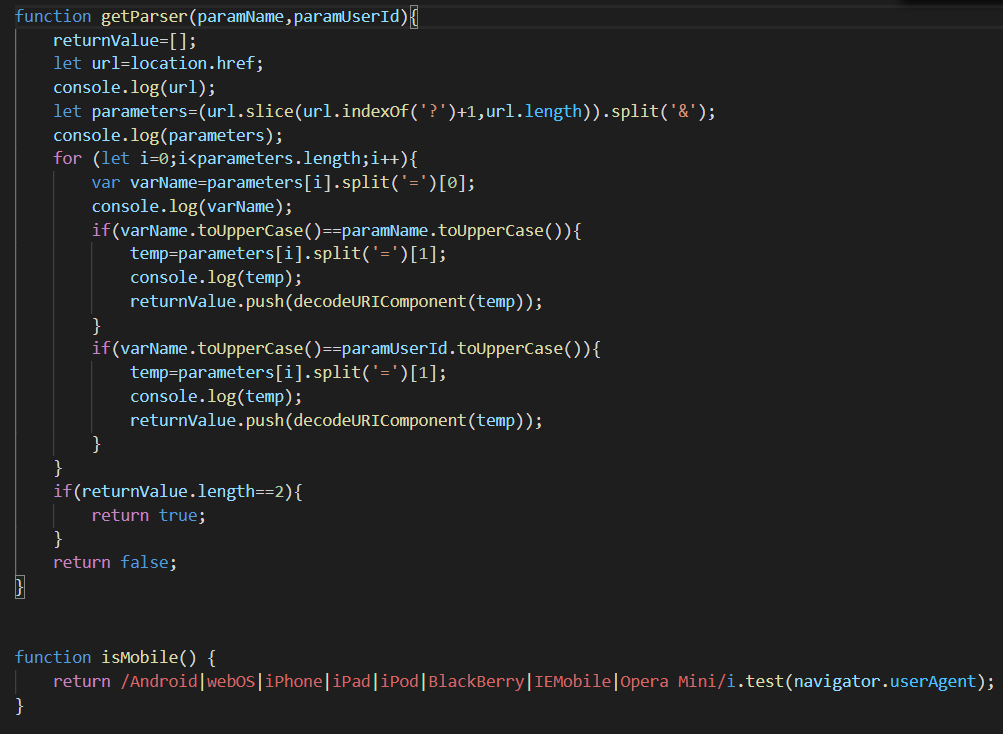
**등록한 주소**

* + - * 1. **위치 공유 기능**
* SMS로 url에사용자의 uid와 name함께보내 웹서버에서 get방식으로 가져와 데이터베이스에서 uid와 name이 일치하는지 확인하고 일치하는지 확인해야 지도 표시

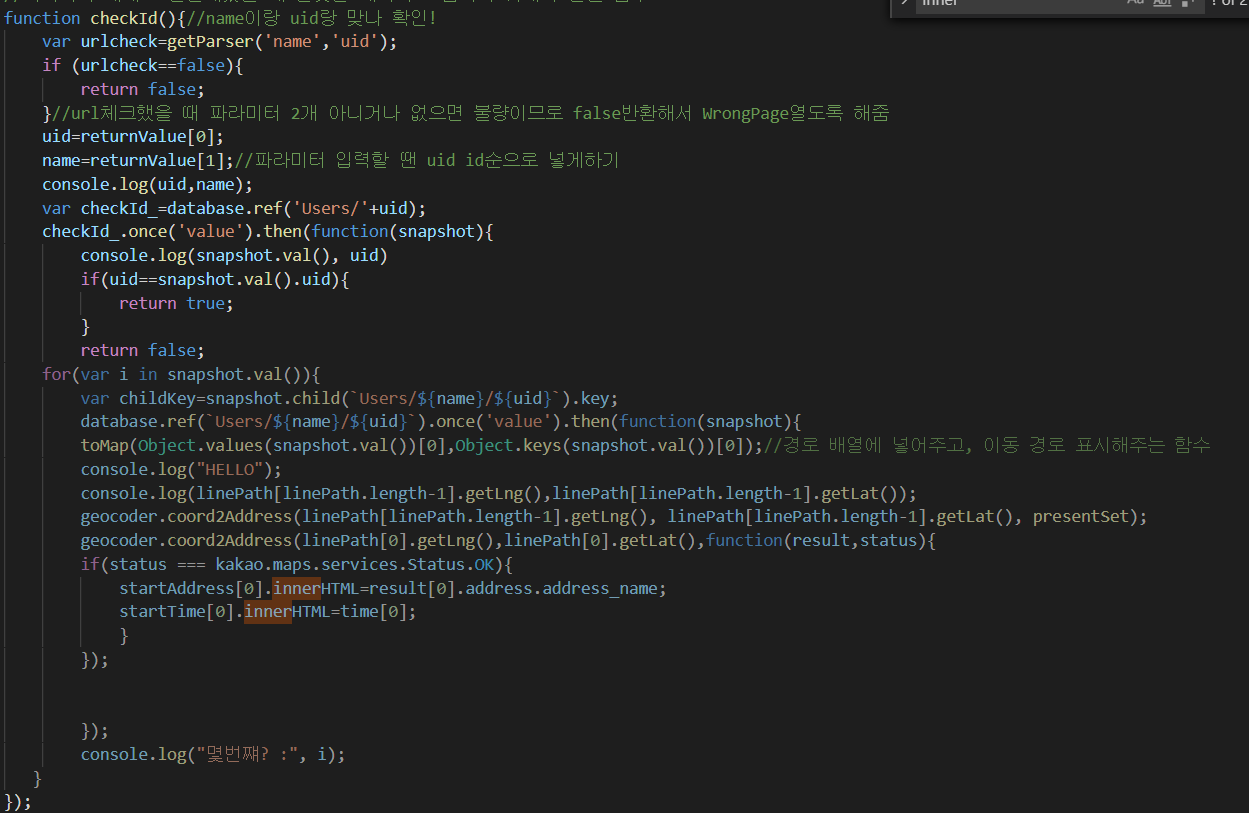




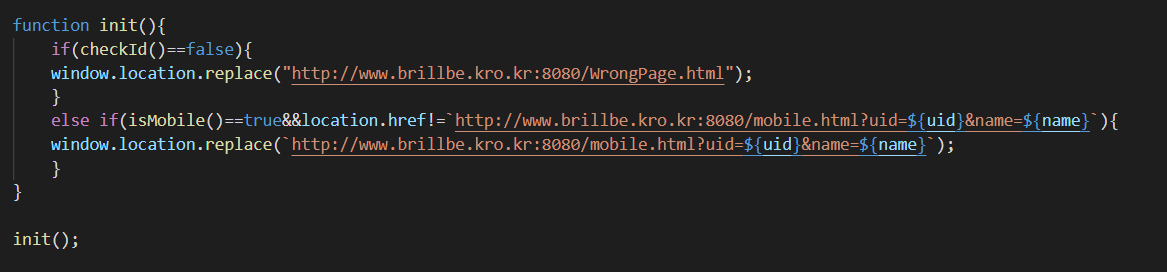
**<정보 일치시>**



<get방식으로 uid와 name가져옴&모바일 기기인지 체크>

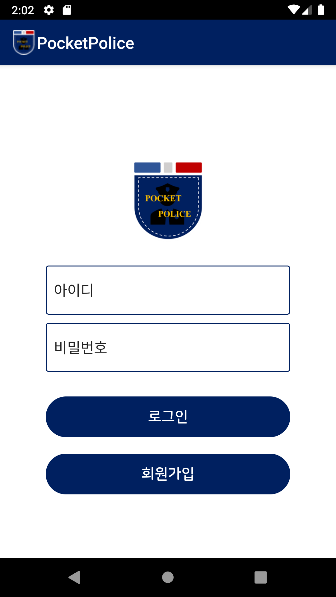


<데이터베이스에 있는 uid값 name값과 같은지 비교>

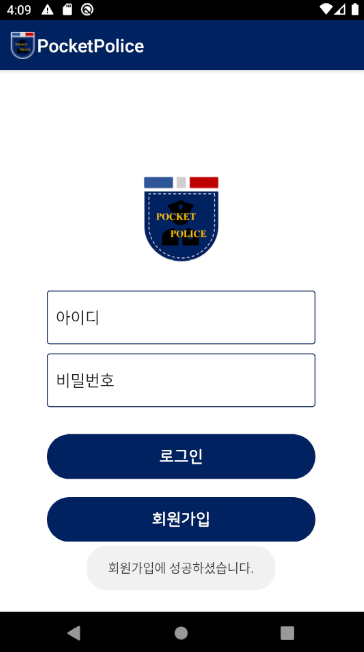
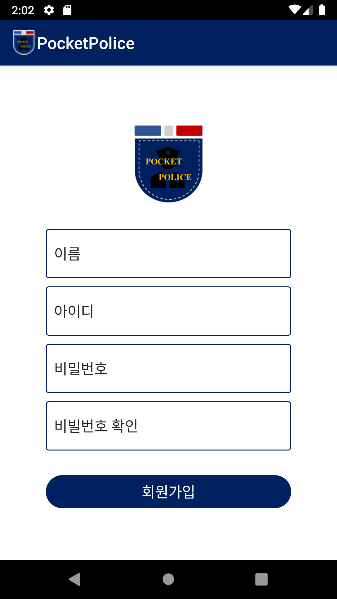


<불일치지 wrongpage로 모바일시 모바일페이지로 변환>

1. **앱 기본 기능**
   * + - 1. **회원 가입/ 로그인 기능**

▶ 로그인 화면 ▶ 로그인 실패 ▶ 로그인 성공



▶ 회원 가입 화면 ▶ 회원 가입 성공

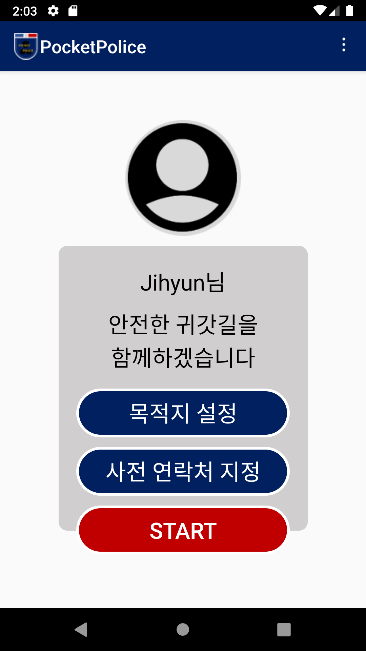
* 로그인 버튼이 눌리면, 데이터 베이스에 저장된 정보를 얻어옴

String email = mEmailText.getText().toString().trim();  
String pwd = mPasswordText.getText().toString().trim();  
firebaseAuth.signInWithEmailAndPassword(email, pwd)

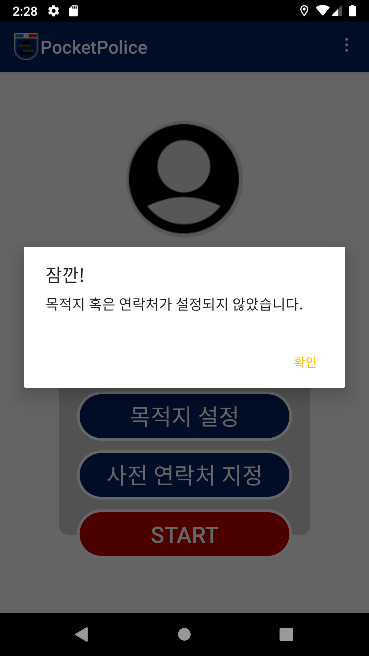
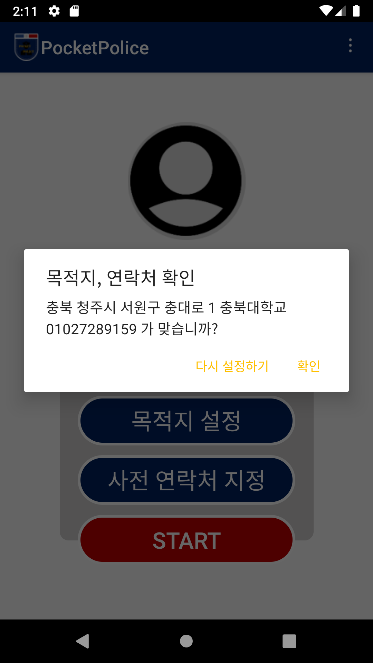
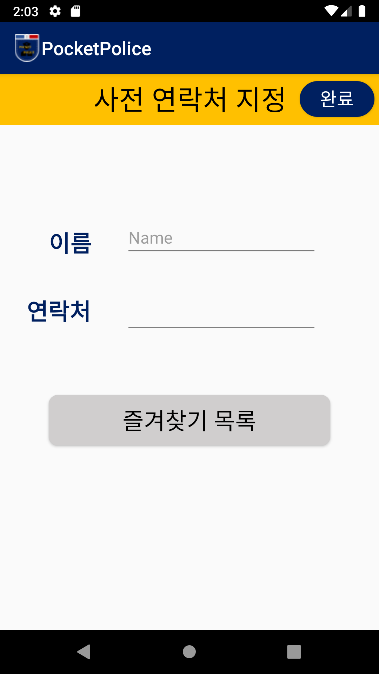
* 로그인 성공 시, “로그인이 성공하였습니다” 라는 팝업창을 띄우며 메인 화면 이동

if (task.isSuccessful()) {  
 Toast.*makeText*(MainActivity.this, "로그인에 성공하셨습니다.", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(MainActivity.this, home.class);  
 startActivity(intent);

* + - * 1. **시작 페이지 사전 정보 설정**



▶ 홈 화면 ▶ 목적지 설정 화면



▶ 연락처 설정 화면 ▶ START버튼 누른 후 화면

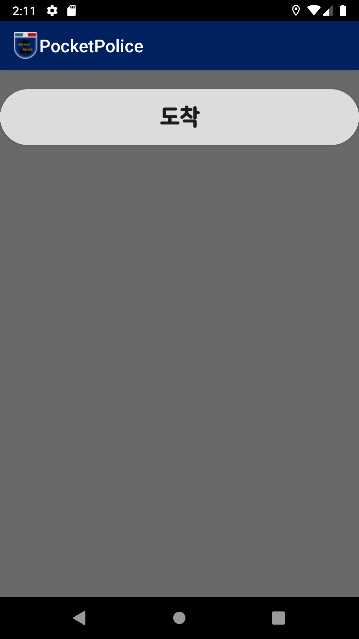
* START버튼이 눌리면

1. 데이터베이스에 저장된 사용자가 선택한 목적지와 연락처를 받아옴

2. 목적지와 연락처 둘 다 저장되어 있는 경우 재확인할 수 있는 알림창 띄움

3. 둘 중 하나라도 저장되어 있지 않는 경우 다시 설정하고 오게끔 경고창 띄움

사전 정보를 입력 후 START를 누르면 기능 시작 화면으로 넘어가면서 GPS 권한이 요청된다. 권한이 허용되어 있다면 정상적으로 페이지가 시작되며, 허용되어 있지 않다면 허용을 요청하는 팝업이 뜨게 된다.



▶ 위치 권한 설정 화면 ▶ 기능 시작 페이지

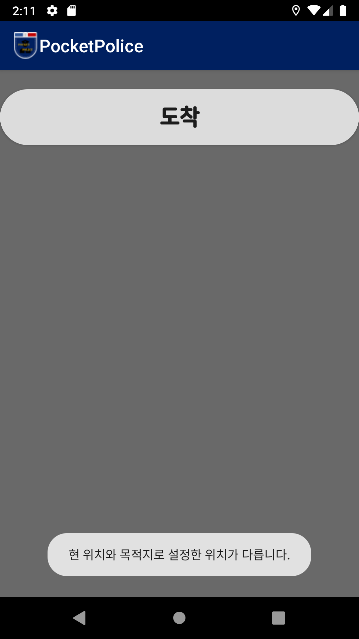
페이지가 정상적으로 시작되면 GPS정보 저장 기능이 시작된다.

locationManager과 locationListener를 사용하여 location을 받는다.

locationManager.requestLocationUpdates(LocationManager.*NETWORK\_PROVIDER*, 30000, 0, locationListener);  
locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 30000, 0, locationListener);

이때 30000ms 단위로 gps 정보가 업데이트 되도록한다.

도착 버튼을 누르게 되면 사용자가 목적지로 설정한 주소와 비교하여 오차 범위 이내라면 도착 완료 화면으로 넘어가게 되고, 오차 범위 밖이라면 목적지와 일치하지 않는다는 알림을 띄운다.



▶ 목적지 범위 밖에서 도착버튼 눌렀을 시 ▶ 목적지 범위 내에서 도착버튼 눌렀을 시

Location nowlocation = locationManager.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*);

현재 사용자 위치를 nowlocation에 저장

Location input\_loc=*findGeoPoint*(this,data\_address);

사용자가 사전에 설정한 목적지를 주소로 받아와 위치 좌표로 변환하여 input\_loc에 저장한다.

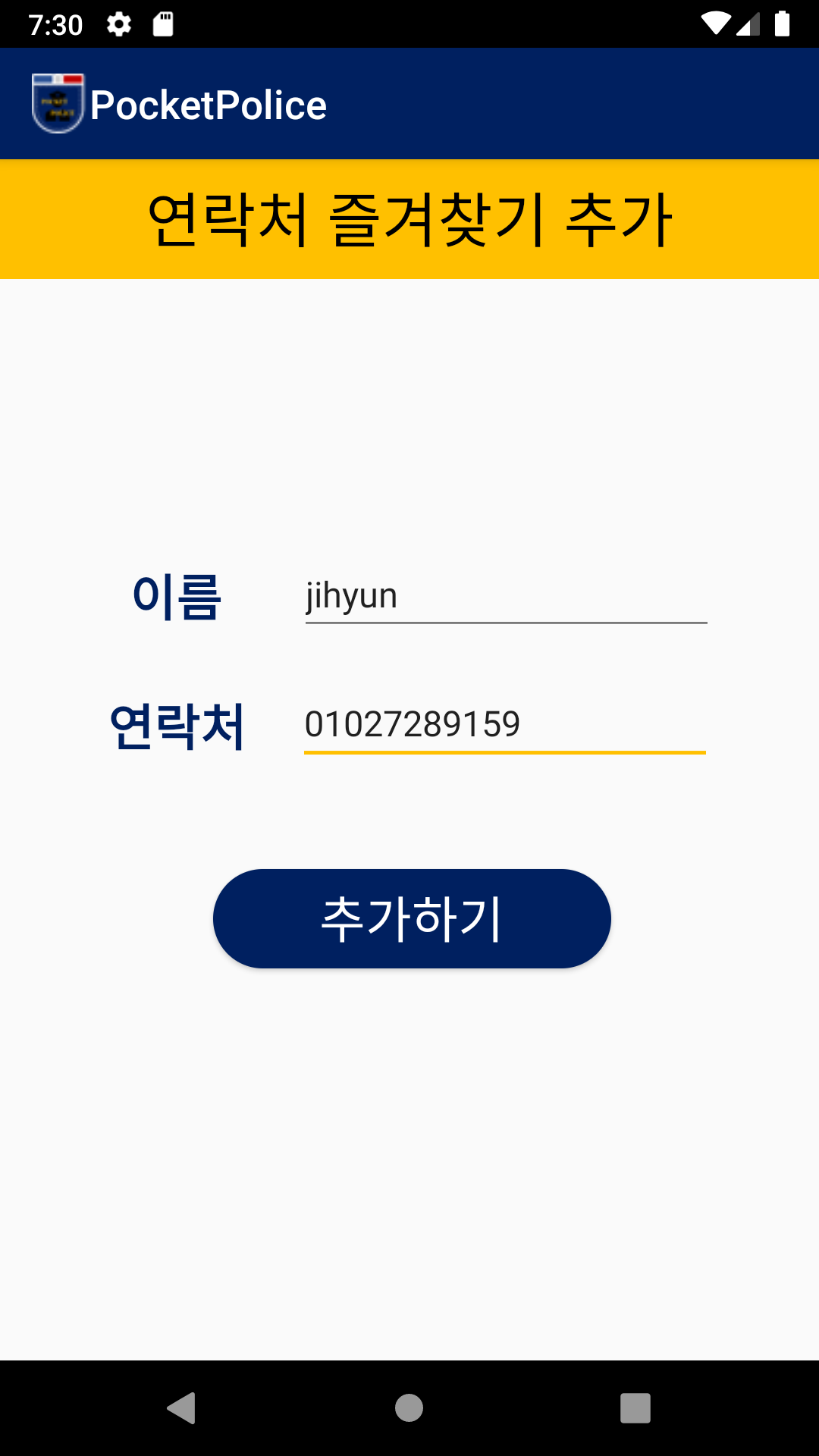
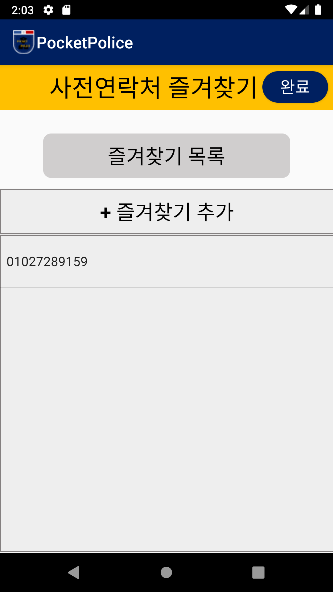
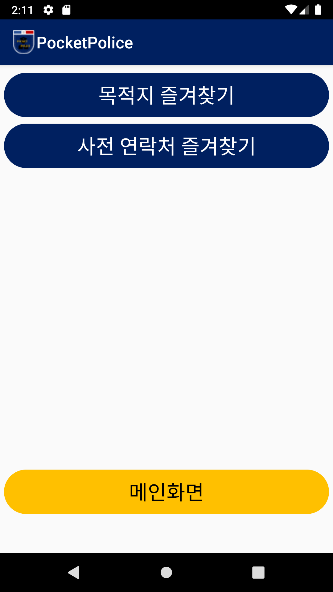
이때 String 주소를 Location으로 변환하는 findGeoPoint()는 지오코딩을 사용하였다.

if(now\_lati>=goal\_latitude-0.002&&now\_lati<=goal\_latitude+0.002&&now\_longi>=goal\_longitude-0.002&&now\_longi<=goal\_longitude+0.002){  
 Intent intent1 = new Intent(getApplicationContext(), arriving\_complete.class);  
 startActivity(intent1);  
 locationManager.removeUpdates(locationListener);  
 }  
 else{  
 Toast.*makeText*(getApplicationContext(),"현 위치와 목적지로 설정한 위치가 다릅니다.",Toast.*LENGTH\_LONG*).show();  
 }

이때 저 두 location 값의 위도 경도차가 0.002 이하일때 목적지에 도착한 것으로 판단

도착 완료 페이지에서 메시지 전송을 누르면 사전 설정한 번호로 안전 귀가 메시지를 보낼 수 있다.

* + - * 1. **환경설정**



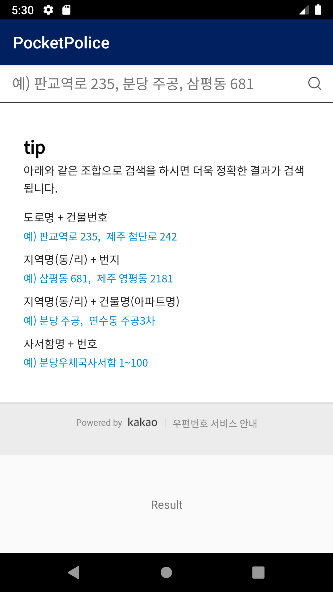
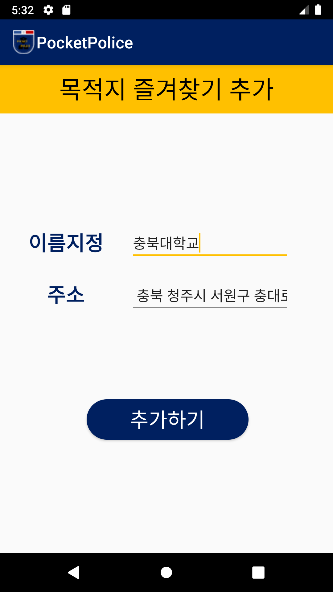
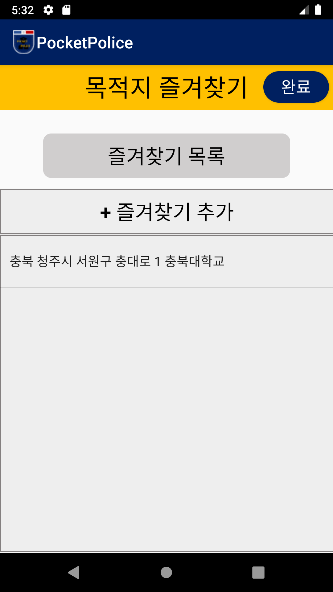
▶ 설정 화면 ▶ 연락처 즐겨찾기 표시 ▶ 즐겨찾기 추가

* 즐겨 찾기에 값을 추가하면 추가한 연락처를 listview로 표시한다.

public void onDataChange(DataSnapshot dataSnapshot){  
 adapter.clear();  
  
 for (DataSnapshot nbmData : dataSnapshot.getChildren()){  
 for (DataSnapshot nbmdata2 : nbmData.getChildren()) {  
 String nbmdata3 = nbmdata2.getValue().toString();  
 Array.add(nbmdata3);  
 adapter.add(nbmdata3);  
 }  
 }  
 adapter.notifyDataSetChanged();  
 listView.setSelection(adapter.getCount()-1);  
}

* 즐겨 찾기 목록에 있는 값을 누르면 메인 화면으로 돌아가고 값은 저장된다.

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
 @Override  
 public void onItemClick(AdapterView<?> adapterView,  
 View view, int position, long id) {  
   
 selected\_item = (String)adapterView.getItemAtPosition(position);  
   
 bm\_save1();  
 Intent go1 = new Intent(getApplicationContext(), home.class);  
 startActivity(go1);  
 }  
});



▶ 목적지 즐겨찾기 표시 ▶ 즐겨찾기 추가 ▶ 주소 검색

* 즐겨 찾기에 값을 추가하면 추가한 연락처를 listview로 표시
* 즐겨 찾기 목록에 있는 값을 누르면 메인 화면으로 돌아가고 값은 저장된다.
* 주소 입력창을 누르면 주소 검색창을 이용하여 주소를 입력할 수 있다.

<클릭 시 주소검색페이지로 이동>

EditText find\_address = (EditText) findViewById(R.id.*find\_address*);  
find\_address.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent pg4 = new Intent(getApplicationContext(), DaumWebViewActivity.class);  
 startActivity(pg4);  
 }  
});

<주소검색페이지에서 얻은 값 즐겨찾기 추가 페이지로 전송>

Intent intent\_ad = new Intent(getApplicationContext(), dbmupdate.class);  
intent\_ad.putExtra("input\_address", input\_add);  
startActivity(intent\_ad);

<즐겨찾기 추가 페이지에서 값 받음>

Intent ad = getIntent();  
final String desti = ad.getStringExtra("input\_address");

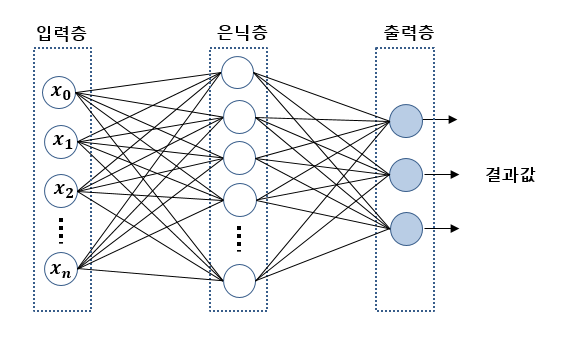
* + - * 1. **위치 공유 기능**
* 사용자가 위급 상황시(여기서는 사용자가 주저 앉을 때)즉각 신고되어야하는데 112에는 sms신고 기능이 있으므로 SMS으로 링크를 공유해 사용자의 위치를 즉각 알 수 있게 하기 위하여 웹에서 사용자의 이동경로를 표시하게 했다.
* Uid와 name값 이용해 데이터베이스에서 실시간 GPS와 시간 값 가져와 실시간으로 표시
* <사진 후에 첨부>
* PC/mobile접속화면 각각 보기 편하게 만들었다.

1. **MLP 주요 기능**

* **MLP 이론**

**MLP (다중 퍼셉트론, Multilayer Perceptron)**

: 단층 퍼셉트론의 비선형적으로 분리되는 데이터에 대해서는 제대로 된 학습이 불가능하다는 한계를 극복하기 위해 고안되었다. 다층 퍼셉트론은 **입력층과 출력층 사이에 하나 이상의 중간층을 두어 비선형적으로 분리되는 데이터에 대해서도 학습이 가능**하도록 한다.

단층 퍼셉트론의 동작 원리는 활성 함수가 내놓는 결과값이 실제값과 오류가 최소가 되도록 입력층에서 전달되는 가중치의 값을 결정하는 것인데, 다층 퍼셉트론의 동작원리 또한 이와 크게 다를 바 없지만 다층 퍼셉트론은 은닉층과 출력층에 존재하는 활성함수가 여러 개고, 이에 다른 가중치도 여러 개다.

**다층 퍼셉트론의 동작 순서**

1. 각 층에서 가중치를 임의의 값(일반적으로 0에 가까운 값)으로 설정한다. 각 층에서 바이어스 값은 1로 설정한다.

2. 하나의 트레이닝 데이터에 대해서 각 층에서의 순입력 함수값을 계산하고 최종적으로 활성 함수에 의한 출력값을 계산한다.

3. 출력층의 활성 함수에 의한 결과값과 실제값이 허용 오차 이내가 되도록 각층에서의 가중치를 업데이터 한다.

4. 모든 트레이닝 데이터에 대해서 출력층의 활성 함수에 의한 결과값과 실제값이 허용 오차 이내가 되면 학습을 종료한다.

출처: <https://m.blog.naver.com/samsjang/221030487369>

1. **목적:**

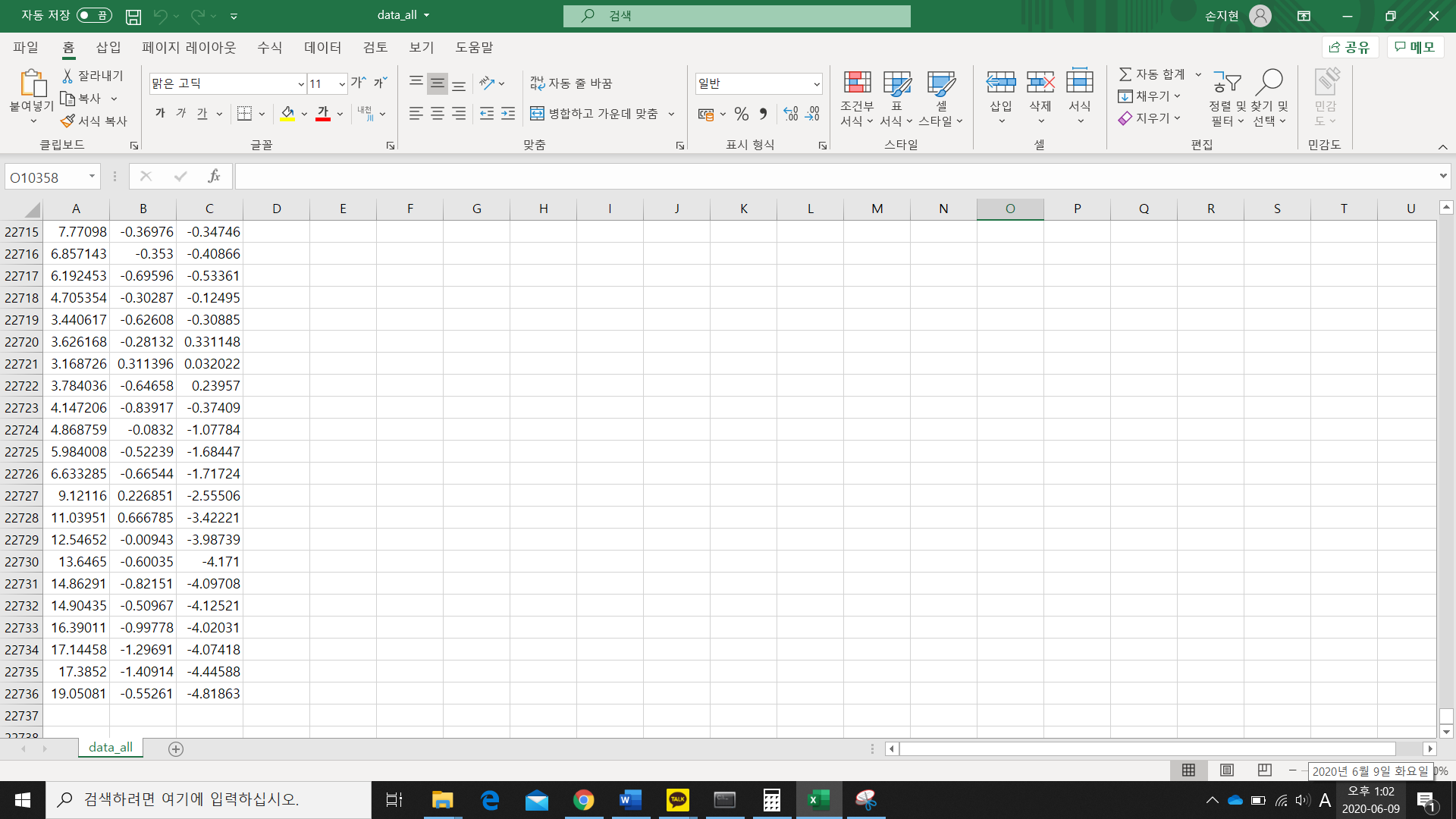
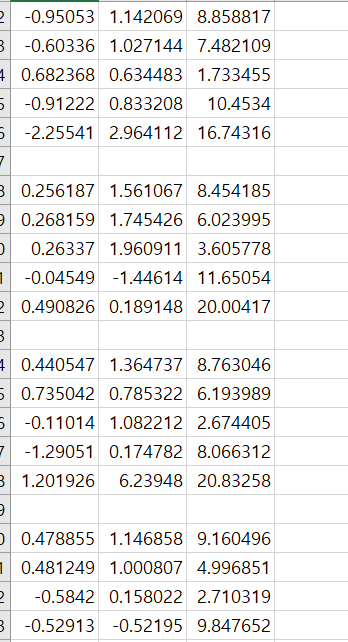
가속센서를 통해 얻은 데이터 값을 입력으로 하고 이진분류모델을 사용하여 **사용자가 주저 앉았는가를 예측**

1. **데이터 및 전처리과정:**

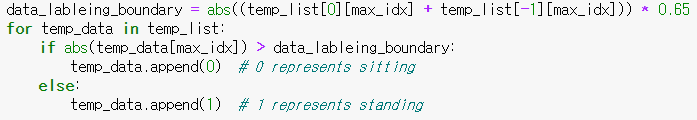
* **데이터셋 구성 필요조건**

MLP는 지도학습에 기반한 모델로서 학습을 위해서는 입력데이터와 출력층의 정답여부를 확인할 수 있는 라벨이 필요하다.

데이터는 **입력 데이터 + 라벨 데이터**로 구성된다.

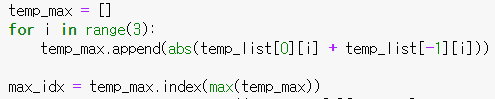
* **데이터셋 수집 및 라벨링**
  + 1. 가속 센서의 연속적인 측정값을 저장한 데이터가 있다.

**한번 앉을 때를 기준으로 나눈다**

* + 1. 연속된 데이터에서 앉음과 서있음을 구별하기위해 측정값의 시작과 끝 사이의 특정 지점을 기준으로 라벨링을 진행하도록 전처리를 함.

최저점과 최고점의 65%지점을 섬과 앉음의 구분점으로 사용

* + 1. 측정값의 양상이 상황에 따라 음수일때도 있고 양수일때도 있다. 따라서 구분점과 축의 데이터를 절대값 처리 후 비교
    2. 주저앉았을 때 가속 센서의 측정값이 크게 변화하는 양상을 보인다. 즉 축별로 측정된 값의 절대값이 커진다는 점을 참고하여 라벨링을 위한 특정지점을 정하는 데 활용하였다.
    3. 각 축의 변화량(기울기)을 체크하여 가장 큰 값이 앉고 서있음을 구별하도록 하는 구분자이므로 이를 특이점을 결정하는데 사용

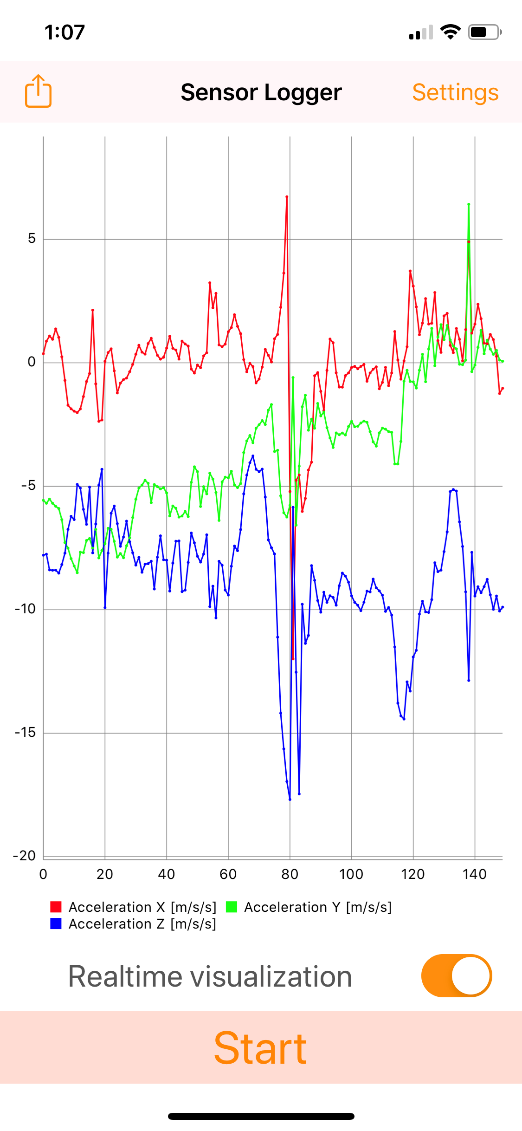


* **데이터 활용**

: 학습데이터셋과 테스트데이터셋은 70:30의 비율로 나누어 사용하였다.

* **데이터 결과**

: Standing status: 1/ Sitting status: 0



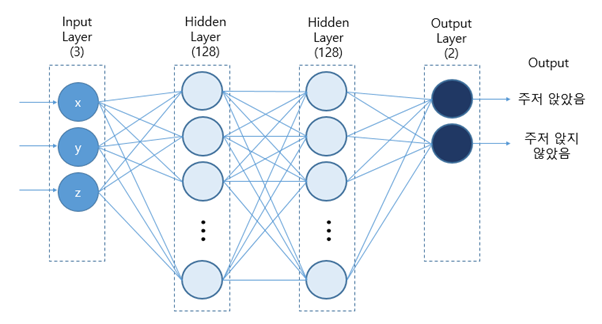
Input

측정된 가속도 센서의 x, y, z 값

Output

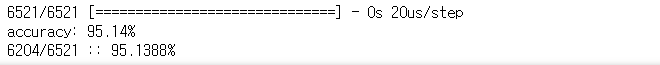
사용자가 주저 앉았음 여부

1. **MLP 구성**



1. **MLP 학습 결과**

* 분류 모델을 통한 예측 성능 -> **정확도: 95.14%**

(학습 환경: epoch 150, batch size: 100)

▶ 6521개의 학습 데이터 중 6204개

* **앱 MLP 주요 기능**

앱 start시 서버로 가속도 정보를 6000ms단위로 모델이 저장되어 있는 서버로 전송한다.(소켓 프로그래밍) x,y,z,값을 바이너리로 보낸다음 서버에서 다시 float값으로 바꾼다. x,y,z세개의 정보를 4개씩 묶어서 입력한다. 앉았다면 0인상태이고, 다시 정보를 가져와 앱에서 바로 신고기능으로 넘어간다.