School of Information Technologies
Faculty of Engineering & IT

# ASSIGNMENT COVERSHEET
# GROUP ASSIGNMENT

**Unit of Study**: Mobile Computing

**Assignment name**: Build a Mobile App – Proposal

**Assignment group members:**

YUMING JIANG 460444981

YUHAO CHENG 450580673

## Abstract

Recently, notes are not only related to the time but also the location. People want to check what they should do in a specific building when they walk into that place. A good Note-Taking App should notify users the information based on the alert location. To solve this problem, our project team develop this easy-to-use “Note” App. Users can not only set the alert time but also the notification location. With this App, people can remember things much more easily and naturally.

## Table of Contents

**Abstract**........................................................1
Background & Significance........................................................3
Objective........................................................3
Comparing with related Work........................................................3
①Evernote........................................................3
②Todoist........................................................3
App Storyboards........................................................4
1.0 SplashActivity........................................................4
2.0 fblogin........................................................4
3.0 MainActivity........................................................5
4.0 EditActivity........................................................8
5.0 AlertTime........................................................10
6.0 LocationService........................................................11
7.0. MapsActivity........................................................11
Reflection of the report........................................................12
Reflection of the presentation........................................................13
References........................................................13

## Background & Significance

Nowadays, taking notes on the smart phone is essential in people's daily life. There are also many useful notes-taking apps such as Evernote and ToDoist, developed for Android to help people memory things easily. However, all of these apps have a significant problem that when the number of notes grow to hundreds or thousands, it is really difficult to find the note that the user exactly wants. Furthermore, these apps are always focus on alerting users with time, which is not enough to match people's need.

## Objective

This project aims to help users find necessary notes with both passive and active ways. For passive way, users can find a specific note with orders of distance and alert time. For active ways, this App will remind users the notes automatically according to the user-setting position and time.

## Comparing with related Work

To determine whether this project is valuable, two notes-taking app developed for Android is compared with the objective of this app. Those apps are Evernote and ToDoist.

①Evernote

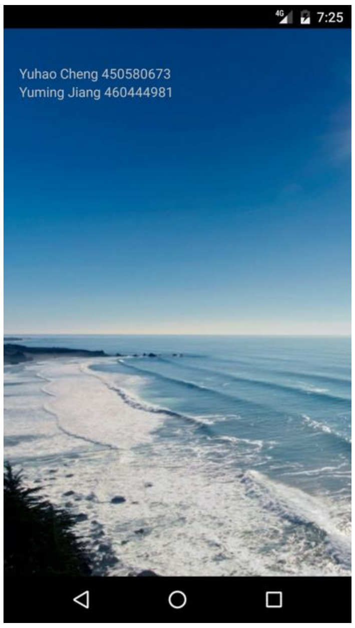
Evernote is a kind of notebook application which has multiple functions. It was founded in America on May 2012. [1]
Nowadays, this application already has a large amount users and be highly acclaimed. But it still has some problems when comparing with our product.
The format of every file is very complicated.
Users may be crazy when they are going to finish the articles format, it is even more complicated than WORD! But our product is clear, easy, understandable and user-friendly note application. It is just a note collector.
The easiest way for users to find a note is searching the name or keyword of the note. Honestly, it is a good function but it can not be the only function. So our product provides the sorting function according to the distance and time.
The functions in free edition is limited. Users cannot take notes when it is offline and can not set passwords for notes if they do not pay the money. The good news is our product is totally free! Further more, it still has some drawbacks such as non-supporting adjusting images size. Anyway, it is a good note application but not a good editing application.

②Todoist

Todoist is also very famous among notes-taking app users. [4] The date and excerpt of each note is displayed in the list in the main interface. There is a search button in the top right of the main interface. Users can sort the notes by date, priority and name.

However, this app can not link the note with the location. All of reminders are not free. The final app of this project will actively remind users things based on positions for free.

## App Storyboards



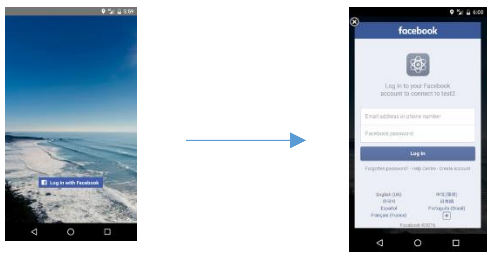
1.0 SplashActivity
This is an Activity that let our application open smoothly. Users often see a white screen before entering the application without splash screen. (Splash screen will jump out automatically and immediately when users click to open the application icon. Developers can put their basic information they want to show to users in this screen.)
Implement:
We simply used a new activity called splashActivity (Layout: activity_splash).
Set an integer called SPLASH_DISPLAY_LENGTH = 2000, which means the application will automatically jump to the main system after 2 seconds wait.
We imported Intent class and bundle class to control jumping between this activity and the next activity.
Here is the main code:

```
new android.os.Handler().postDelayed(new Runnable() {
    public void run() {
        Intent mainIntent = new Intent(SplashActivity.this,
                fblogin.class);
        SplashActivity.this.startActivity(mainIntent);
        SplashActivity.this.finish();
    }

}, SPLASH_DISPLAY_LENGHT);}
```

By the way, we settled this activity launch firstly in Manifest.

2.0 fblogin
In this activity, we let users to login with their Facebook accounts. All users with Facebook accounts can login our system. And they can share their notes after logging in. The activity will jump to Facebook login page when clicking the Facebook login button.
Furthermore, if the system checked the users are already logged in, users will directly enter the main page instead of this page.



Implement: In this activity, we mainly used 3 methods

### 2.1 onCreate()

This method is mainly control the log in button. When this button is clicked, the application will show a page that let users to login in with account names and passwords. And judge what will happen if it success or not. (CallbackManage is provided by Facebook SDK)

2.2 ForwardtoActivity()

This method is called when logging successfully and let program jump to Main activity. (We used the Intent class to reuse the parameter).

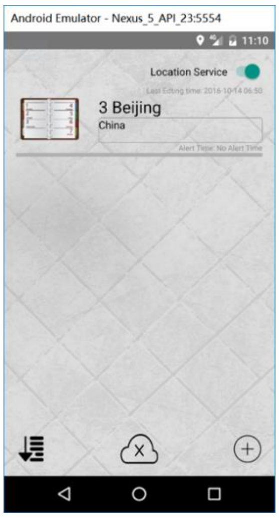
```
public void ForwardtoMainactivity() {
    Intent intent = new Intent(fblogin.this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
    finish();}
```

2.3 isLoggedIn()

If users have logged in, the program will directly jump to main activity. We used the accessToken class to access the logging information.

```
public boolean isLoggedIn() {
    AccessToken accessToken = AccessToken.getCurrentAccessToken();
    return accessToken != null;
}
```

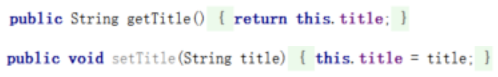
## 3.0 MainActivity


Android Emulator - Nexus_5_API_23:5554
11:10
Location Service
Last Editing time: 2016-10-14 06:50
3 Beijing
China
Alert Time: No Alert Time

After users login with their Facebook account, they will get into this activity. In this activity, users can see the notes they save in the phone. They can add new notes with the Add New Note Button, sort the notes with the Sort Button, logout Facebook with the Logout Button and open or close the background location check service with the Location Service Switch. The layout of the MainActivity is shown below.

### 3.1 Note Class

All of notes are defined as Note object and stored in the local database. Typically, there will be five String: title, time, message, imageURI, alertTime, and two double: latitude and longitude stored in one Note object. Users can get and change information from Note object simply with usual methods just like getTitle() and setTitle(String) for title.

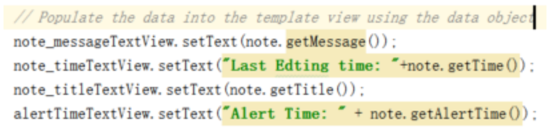


```
public String getTitle() { return this.title; }

public void setTitle(String title) { this.title = title; }
```

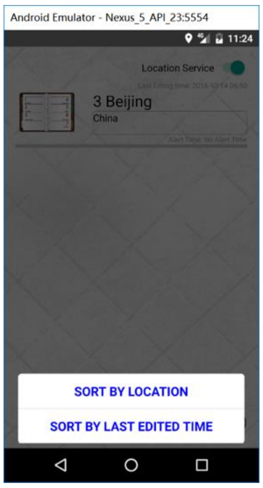
## 3.2 UserAdapter Class

UserAdapter extends ArrayAdapter, it defines the layout of the notes shown in the MainActivity.



```
// Populate the data into the template view using the data object
note_messageTextView.setText(note.getMessage());
note_timeTextView.setText("Last Edting time: "+note.getTime());
note_titleTextView.setText(note.getTitle());
alertTimeTextView.setText("Alert Time: " + note.getAlertTime());
```

## 3.3 Sort Button



As shown in the figure, after clicking on the sort button, there will be two options displayed. The default sort order is LAST EDITED TIME. If users click on the SORT BY LOCATION, the notes will be sorted by the alert location.

### 3.3.1 Sort by last edited time

The notes edited by the user will stored in the end of the database. To sort by LAST EDITED TIME, just read all of notes from the database and show them from down to top.

### 3.3.2 Sort by location

As the distance should be calculated based on the current location, the distance is not covered in the Note object. To make the compare simple, our project team define a class called NoteDistanceCompare which stores the Note object and the distance between the alert location stored in the Note and the current location of users. After sorting the ArrayList of NoteDistanceCompare with java.util.Collections and java.util.Comparator, the notes can be shown in the ascending order of the distance between the alert location and the current location.

```
Collections.sort(arrayOfNotesDistanceCompare, new Comparator<NoteDistanceCompare>() {
    @Override
    public int compare(NoteDistanceCompare noteDistanceCompare, NoteDistanceCompare t1) {
        //Calculate the distance and the difference
        double temp = noteDistanceCompare.getDistance() - t1.getDistance();
        if (temp>0){
            //If noteDistanceCompare is far than t1, return 1
            return 1;
        }
        if (temp<0){
            //If t1 is far than noteDistanceCompare, return -1
            return -1;
        }
        //If they have the same distance, return 0
        return 0;
    }
});
```

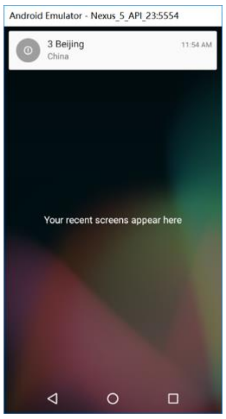
## 3.4 Facebook logout button

After clicking on the logout button, the current token of Facebook will be destroyed. This app will jump to the fblogin activity, users should login with their Facebook account again. The methods used to destroy the current token is shown below.

```
public void facebookLogOut(View view){
    LoginManager.getInstance().logOut();
    Intent intent = new Intent(this, fblogin.class);
    startActivity(intent);
    finish();

}
```

## 3.5 Location Service Switch


Android Emulator - Nexus_5_API_23:5554
3 Beijing
China
11:54 AM
Your recent screens appear here

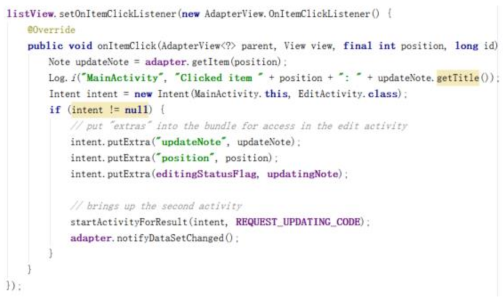
This switch controls the on/off of this App's background location change detecting service. It will be set off for the first time of using this App. After setting this switch on, the background location change detecting service will be enabled. As shown in the figure, this means that even the users close the App, it will still send notifications to users when users' current locations are the same with alert locations. The methods to achieve this function will be talked about in the section 6.0 LocationService in this report.

## 3.6 Add New Note Button

After clicking on this button, users can add their new notes to the local database. This App will jump to the EditActivity.

## 3.7 Short click on the note

Users can also short click on the notes displayed on the MainActivity to edit the note stored before. After short clicking on the note, this App will jump to the EditActivity and display the information of the clicked note. As shown below, to implement this function, our project team uses the OnItemClickListener.

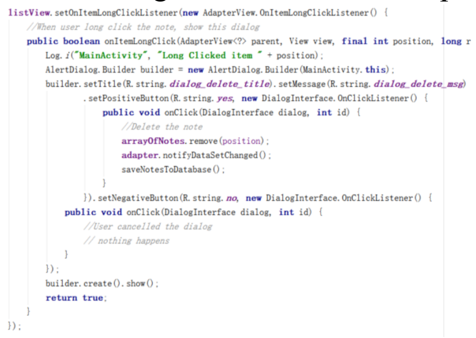


```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {
        Note updateNote = adapter.getItem(position);
        Log.i("MainActivity", "Clicked item " + position + ": " + updateNote.getTitle());
        Intent intent = new Intent(MainActivity.this, EditActivity.class);
        if (intent != null) {
            // put "extras" into the bundle for access in the edit activity
            intent.putExtra("updateNote", updateNote);
            intent.putExtra("position", position);
            intent.putExtra(editingStatusFlag, updatingNote);

            // brings up the second activity
            startActivityForResult(intent, REQUEST_UPDATING_CODE);
            adapter.notifyDataSetChanged();
        }
    }
});
```

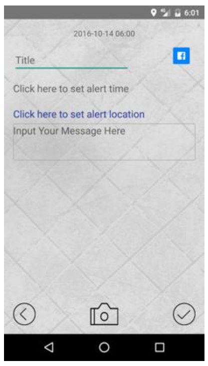
## 3.8 Long click on the note

Users can long click on the notes to delete them, to implement this function, one OnItemLongClickListener is set up.



```
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    //When user long click the note, show this dialog
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int position, long r
        Log.i("MainActivity", "Long Clicked item " + position);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setTitle(R.string.dialog_delete_title).setMessage(R.string.dialog_delete_msg)
                .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        //Delete the note
                        arrayOfNotes.remove(position);
                        adapter.notifyDataSetChanged();
                        saveNotesToDatabase();
                    }
                }).setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                //User cancelled the dialog
                // nothing happens
            }
        });
        builder.create().show();
        return true;
    }
});
```

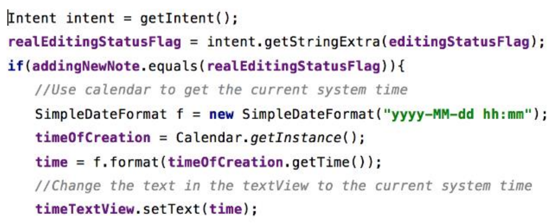
## 4.0 EditActivity


6:01
2016-10-14 06:00
Title
Click here to set alert time
Click here to set alert location
Input Your Message Here

When clicking the add new button on the main activity, users will jump to this activity.
In this Layout, several buttons, textviews, imageview and editviews are settled. And they all have different functions which are important to the whole system.

## 4.1 Last edited time textview
2016-10-14 08:46

On the top of the layout, the textview is settled to show the last edited time of the note. It will also be shown on the main activity.
Implement: We used the Calendar class to get the system time and SimpleDateFormat class to format the string, then used Intent (Flag) class to show in the textview.



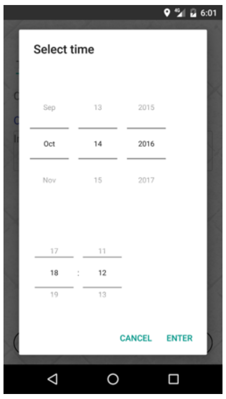
```
Intent intent = getIntent();
realEditingStatusFlag = intent.getStringExtra(editingStatusFlag);
if(addingNewNote.equals(realEditingStatusFlag)){
    //Use calendar to get the current system time
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");
    timeOfCreation = Calendar.getInstance();
    time = f.format(timeOfCreation.getTime());
    //Change the text in the textView to the current system time
    timeTextView.setText(time);
```

Every time the note is edited, the time will change and save automatically.

4.2 Title editview

This is the editview to enter the title of the note and will be shown on the main screen.

4.3 Set Alert time



This is a textview to show and set the alert time. It will jump out a dialog to let people choose what time should be alerted, then save or cancel it. Then the settled time will be shown on the textview and main activity.

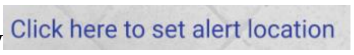
Implement: We used the onTouch() method via View class and AlertDialog class to make the dialog then used the Calendar get time and date from system and set time and date in timepicker and datepicker.

```
public boolean onTouch(View v, MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        View view = View.inflate(this, R.layout.date_time_dialog,
null);
```

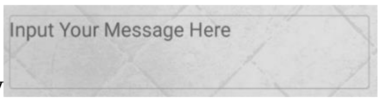
```
@Override
public void onClick(DialogInterface dialog, int which)
```

```
@Override
public void onClick(DialogInterface dialog, int which) {
    dialog.cancel();//nothing happens
}
```

4.4 Set location textView 

Users can click the locationTextView to set the alert location. If users do not set it, the current location will automatically be set as the current location and stored in the database. After clicking this text view, this App will jump to the MapsActivity. To realize this function, a OnClickListener is set up.

```
locationTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(EditActivity.this, MapsActivity.class);
        intent.putExtra("latitude", latitude);
        intent.putExtra("longitude", longitude);
        startActivityForResult(intent, REQUEST_MAP_CODE);
    }
});
```

4.5 Message Editview 

This editview is used to input the message, no words limited and return button is settled to change lines.
Implement: the method it used is same with the title editview.
4.6 Photo button
When clicking the photo button, two secondary buttons will come out. And we can choose take photo and choose photos from gallery. Then the photo will be shown in the imageview.

Implement:

4.6.1 Show(View view)

This method is used to show the secondary buttons which used the dialogclass.

```
public void show(View view){
        dialog = new Dialog(this,R.style.ActionSheetDialogStyle);
        inflate =
LayoutInflater.from(this).inflate(R.layout.dialog_layout, null);  }
```

```
dialogWindow.setAttributes(lp);
dialog.show();//
```

4.7 Take photo button(onTakePhotoClick)

Used the intent class to open the camera and transfer the photo to a uri to save and load.

```
public void onTakePhotoClick() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    String photoDir = photoFileName + time + ".jpg";
    intent.putExtra(MediaStore.EXTRA_OUTPUT, getFileUri(photoDir));
startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

4.8 Choose from gallery(onLoadPhotoClick)

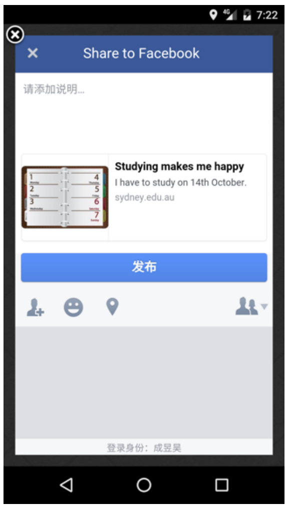
Used the intent class to get the uri of the photo file from system gallery then load and save it.

```
public void onLoadPhotoClick() {
    Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, PICK_PHOTO_CODE);
}
```

4.9 Finish button

When clicking this button, the note will save to the database and show on the main screen.

4.10 Share button



Studying makes me happy
I have to study on 14th October.
sydney.edu.au

When clicking this button, the basic information will be share to Facebook (title, time and message)

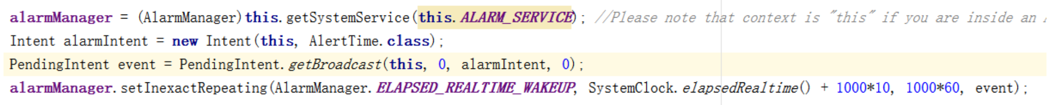
Implement:

```
ShareLinkContent content = new ShareLinkContent.Builder()
        .setContentTitle(fbShareTitle)
        .setContentDescription(fbShareDescription)
        .setImageUrl(myUri)
        .setContentUrl(Uri.parse("http://sydney.edu.au/"))
        .build();
```

```
fbShareButton.setShareContent(content);
```

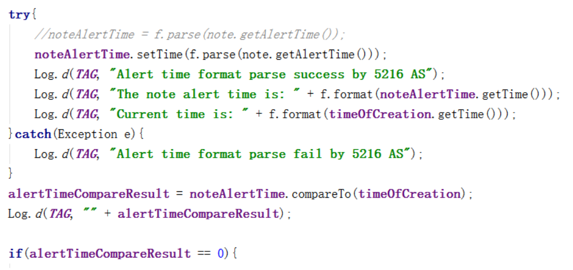
## 5.0 AlertTime

The AlertTime class extends BroadcastReceiver. In the MainActivity, an alarmManager is set to send broadcast every minute.



```
alarmManager = (AlarmManager)this.getSystemService(this.ALARM_SERVICE); //Please note that context is "this" if you are inside an
Intent alarmIntent = new Intent(this, AlertTime.class);
PendingIntent event = PendingIntent.getBroadcast(this, 0, alarmIntent, 0);
alarmManager.setInexactRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, SystemClock.elapsedRealtime() + 1000*10, 1000*60, event);
```

When the AlertTime class receive the broadcast every minute, it will load all of the notes from the database and check the alert time stored in the note. If it is the same with the current time, a notification will be sent to users.

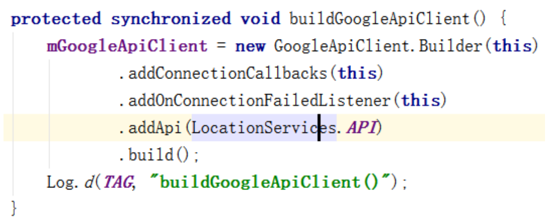


```
try{
    //noteAlertTime = f.parse(note.getAlertTime());
    noteAlertTime.setTime(f.parse(note.getAlertTime()));
    Log.d(TAG, "Alert time format parse success by 5216 AS");
    Log.d(TAG, "The note alert time is: " + f.format(noteAlertTime.getTime()));
    Log.d(TAG, "Current time is: " + f.format(timeOfCreation.getTime()));
}catch(Exception e){
    Log.d(TAG, "Alert time format parse fail by 5216 AS");
}
alertTimeCompareResult = noteAlertTime.compareTo(timeOfCreation);
Log.d(TAG, "" + alertTimeCompareResult);

if(alertTimeCompareResult == 0){
```

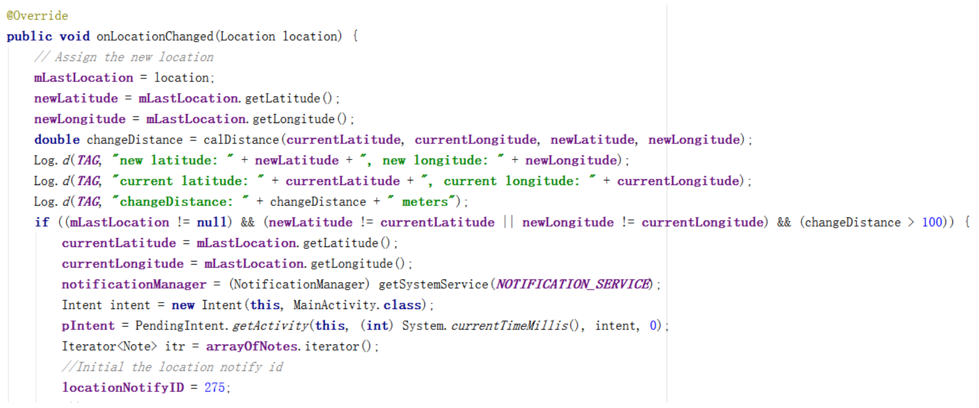
## 6.0 LocationService

To check the current location periodically in the background, the LocationServie class extends Service and implements GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener, LocationListener. Firstly, the Location Service will build a GoogleApiClient and connect to it to get the current location.



```
protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();
    Log.d(TAG, "buildGoogleApiClient()");
}
```

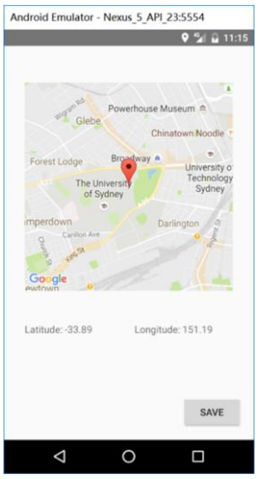
By implementing the LocationListener, when the current location changed, the method onLocationChanged method will be invoked to check the distance between the current location and the alert location stored in the note. If the distance is less than fifty meters, this App will send notifications to users.



```
@Override
public void onLocationChanged(Location location) {
    // Assign the new location
    mLastLocation = location;
    newLatitude = mLastLocation.getLatitude();
    newLongitude = mLastLocation.getLongitude();
    double changeDistance = calDistance(currentLatitude, currentLongitude, newLatitude, newLongitude);
    Log.d(TAG, "new latitude: " + newLatitude + ", new longitude: " + newLongitude);
    Log.d(TAG, "current latitude: " + currentLatitude + ", current longitude: " + currentLongitude);
    Log.d(TAG, "changeDistance: " + changeDistance + " meters");
    if ((mLastLocation != null) && (newLatitude != currentLatitude || newLongitude != currentLongitude) && (changeDistance > 100)) {
        currentLatitude = mLastLocation.getLatitude();
        currentLongitude = mLastLocation.getLongitude();
        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        Intent intent = new Intent(this, MainActivity.class);
        pIntent = PendingIntent.getActivity(this, (int) System.currentTimeMillis(), intent, 0);
        Iterator<Note> itr = arrayOfNotes.iterator();
        //Initial the location notify id
        locationNotifyID = 275;
```

## 7.0. MapsActivity

After clicking on the set alert location button, this App will jump to this activity, the layout is shown below.


Android Emulator - Nexus_5_API_23:5554

11:15

Latitude: -33.89 Longitude: 151.19

SAVE

A google map will be shown in this activity. Users can drag the marker to set the alert location. Users can also click on the map, the marker will be moved to that place automatically. After clicking the save button, this App will save the alert location to the database and jump back to the EditActivity. To realize this function, the MapsActivity implements OnMapReadyCallback. When the google map is ready, method onMapReady(GoogleMap googleMap) will be invoked to show the marker.

```java
// Add a marker of the user's alert location and move the camera
LatLng userLocation = new LatLng(latitude, longitude);
userMarker = mMap.addMarker(new MarkerOptions()
        .position(userLocation)
        .draggable(true)
        .title("user location"));
// Move the camera instantly to location with a zoom of 15.
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLocation, 15));
// Zoom in, animating the camera.
mMap.animateCamera(CameraUpdateFactory.zoomTo(14), 2000, null);
```

A OnMarkerDragListener is set to handle the dragging event.

```java
mMap.setOnMarkerDragListener(new GoogleMap.OnMarkerDragListener() {
    @Override
    public void onMarkerDragStart(Marker marker) {
        Toast.makeText(MapsActivity.this, "Dragging Start",
                Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onMarkerDragEnd(Marker marker) {
        LatLng position = marker.getPosition();
        latitude = position.latitude;
        longitude = position.longitude;
        String strLat = "" + latitude;
        String strLon = "" + longitude;
        latitudeTextView.setText("Latitude:  " + strLat.format("%.2f", latitude));
        longitudeTextView.setText("Longitude:  " + strLon.format("%.2f", longitude));
        Toast.makeText(
                MapsActivity.this,
                "Lat " + strLat.format("%.2f", latitude) + "\n"
                        + "Long " + strLon.format("%.2f", longitude),
                Toast.LENGTH_LONG).show();
    }
```

A OnMapClickListener is also set to handle the click event.

```java
mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
    @Override
    public void onMapClick(LatLng latLng) {
        latitude = latLng.latitude;
        longitude = latLng.longitude;
        String strLat = "" + latitude;
        String strLon = "" + longitude;
        latitudeTextView.setText("Latitude:  " + strLat.format("%.2f", latitude));
        longitudeTextView.setText("Longitude:  " + strLon.format("%.2f", longitude));
        userMarker.remove();
        LatLng userLocation = new LatLng(latitude, longitude);
        userMarker = mMap.addMarker(new MarkerOptions()
                .position(userLocation)
                .draggable(true)
                .title("user location"));
    }
});
```

## Reflection of the report

To write this report, our project team search the Internet to find the guideline for writing report. Finally, we decide to use the book written by John Wiley & Sons as our report writing guideline. Based on the guideline and the real situation, we divide our report into eight sections.

## Reflection of the presentation

After searching the Internet, we decide to use the presentation guideline in the ww2.amstat.org website to prepare our presentation. With this guideline, we well organized our presentation slides, controlled our presentation time and aimed to prepare the best presentation for our audiences.

## References

[1] https://evernote.com/?var=c&noredirect
[2] https://www.google.com/keep/
[3] https://www.any.do/
[4] https://todoist.com/
[5] https://developer.android.com/reference/android/widget/Button.html#
[6] https://developer.android.com/reference/java/util/Collections.html#
[7] https://developer.android.com/reference/android/widget/ListView.html
[8] Lichtenberger, E. O., Mather, N., Kaufman, N. L., & Kaufman, A. S. (2012).*Essentials of assessment report writing* (Vol. 22). John Wiley & Sons.
[9] https://ww2.amstat.org/meetings/jsm/2014/effectivepresentations.cfm