CSIT 6000B: Mobile Application Development

For

Project Report: Passer-By@UST

Prepared by

Rio He, 20227977, yheas@connect.ust.hk
Kevin Wang, 20225577, zwangbo@connect.ust.hk
Cedric Liang, 20220395, zliangag@connect.ust.hk

2014/12/14

Table of Contents

1. Introduction

1.1 User Requirements(Background)

Today, is an information explosion period, and opportunities are coming in everywhere, but they flow away in a flash. So if we want to catch the opportunity at once, people need a tool which could have instance feature to realize this requirement. In this position, it is our project background to let us make sure doing it project.

1.2 Approaches

To implement this application, instant message technology is basic. The real-time chatting will be held in the chatting room.

Additionally, if all the users are chatting within one chatting room, the situation will be quite complex and mess. Of course, users will not have good experience. So Passer-by@UST needs to get the location of users, and push users into a specific chatting room which only contains people who are all in the specific location range. So Google Maps APIs would be involved.

1.3 Project Goals

Passer-By@UST, the name of this application means that people may meet a lot of strangers in HKUST, is a social software that could let strangers chatting at the same time and in the same position to make friends. Before becoming friends with someone, that person is a stranger to us. Maybe, we have already missed chances to know people who should have become friends with us. Passer-By@UST aims to increase the possibility that people could find someone who has similar thoughts with them. Because of the same location and the same time, people may be doing the same thing. That is the key for figuring out new friends.

2. Design and Implementation

2.1 Requirements Analysis

Nowadays, there are a lot of social applications in APP store, for example, Wechat, Whatsapp and facebook. But after we do a research of them, we found that most of them are focus on the friends chatting part, not the strangers chatting part of the social networking, so it is a range that have not been exploited, and also, in this period, communicate with strangers are very normal and frequently, so these two points are the users requirements, so this application has huge demand and a big future.

2.2 Description of System Design

Passer-by@UST is designed especially for HKUST students to make friends with someone who are strangers to them. The core features of Passer-by@UST include searching for a specific chatting room, getting users' location and chatting with other users. Because this application needs combine the client-side and server-side. Apart from the android programming, on the server-side, there should be add database to distribute information to users among the same chatting room, also saving personally information. However, all the chatting history will not be stored in users' phone, just like Snapchat.

2.3 Approaches to Store Data

For personal setting part, Passer-by@UST is a chatting tool. So naturally when a user wants to communicate with others, he would be required to set his nickname and gender. Otherwise, it would very wired to chat.But the nickname and gender are not required to be passed to the server-side. So a local storage can be applied. Considering about current application specification and future development, using SQLite in this case is a good idea.

"SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain."

For message storage part,it is managed by a open & free software tool named Firebase. After user input a message and click send, both the message and his name

would be take into a socket object and push it to the online server, when server receive this socket, it would be stored in the server where have been created the corresponding room to store message with the format {key(name): value(message)} and at the same time send to the other user who are now living in the same room. The picture of server part shows in Appendix B part.

2.4 Specific Features and Important Components

Generally, there will be at least six activities in implementation:

2.2.1 Welcome Activity:

A welcome interface will be shown to users once starting the application. And the guideline of using Passer-by@UST will be animated to users if that is the first time they use this App.

2.2.2 Main UI Activity:

The main interface of Passer-by@UST, providing 'search room' and 'locate myself' features. Also users can modify their personal information here.

2.2.2.1 Input Activity:

Let user input a specific room number to enter into a specific chatting room directly.

2.2.2.2 Setting Activity:

Let user modify their personal settings, like user name, gender.

2.2.3 Result of Searching Room Activity:

This is the result interface to return back the searcher rooms, and if that is which room you want go in, just point it to enter room.

2.2.4 Chatting Activity:

The chatting happens in this activity.

2.2.5 We don't think we would use Content provider and Broadcast receiver, so they have no description. And we draw three diagrams to show our function and processing of project in the Appendix A.

2.2.6 Specific Features

Location Based Service(LBS) will be used to get user's location, and it may need the support of Google Map API and GPS to get a specific and accurate

location. Additionally, because the application requires real-time communication, so this technology will be a highlight, as well.

2.5 Implementation

For storage part.

To use SQLite in Android development, there should be a concept called DatabaseHelper involved. DatabaseHelper is used to simplify the procedures of constructing database. And there is another class called DatabaseManager which is the direct class to connect database and Android Activity.

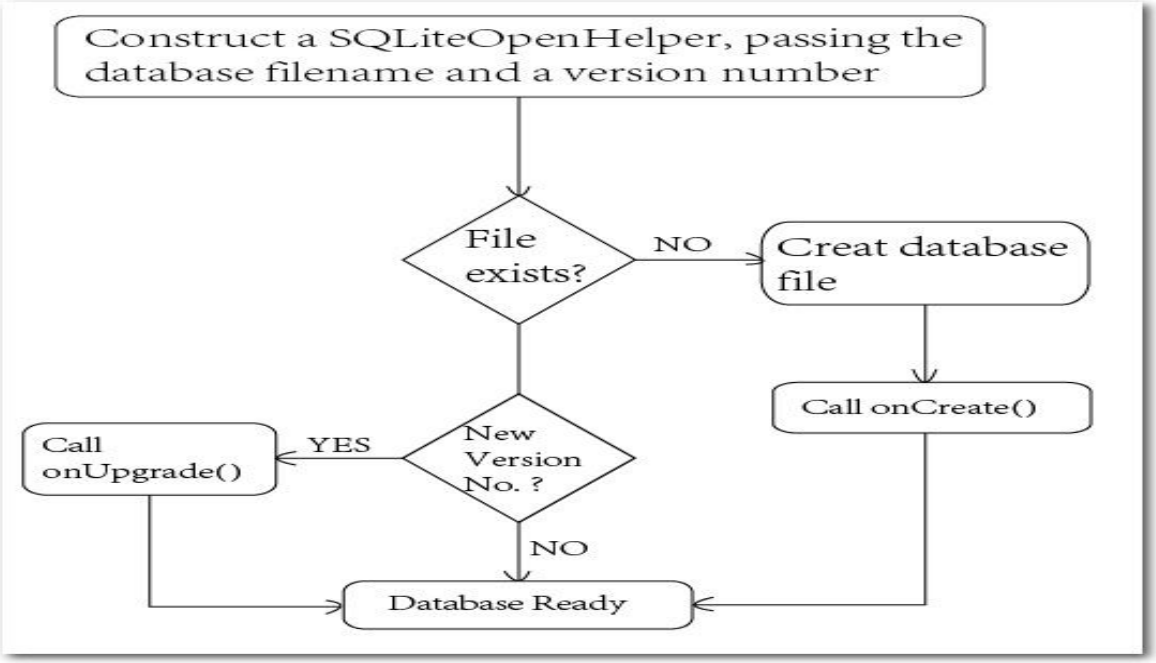
The details of DatabaseHelper is listed the following:

+ Constructor: used to make the instantiation of DatabaseHelper
DatabaseHelper(Context context, String name, CursorFactory factory, int version)
DatabaseHelper(Context context, String name, CursorFactory factory, int version, DatabaseErrorHandler errorHandler)
DatabaseHelper(Context context)
+ Methods: the operations on the calling object
onCreate(SQLiteDatabase db): only when database is created, the method will be called.
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): when version of database is changed, this method will be called
onOpen(SQLiteDatabase db): once the data is required to open,

The details of DatabaseManager listed the following:

+ Constructor:
DatabaseManager()
DatabaseManager(Context context)
+ Methods:
createUser(User user): insert new users into database
uptUser(String prevName): update a specific user's info
queryTheCursor(): get all the data records in the database, which will be used in query()
query(): select current user
closeDB(): close the database

The overview of SQLite is showing below:



Future Work of this part.

Although in current stage, using SQLite looks like too complex and time consuming. If the requirement of storing data is only the current user holding the device, Preferences will be more proper. But in the future, the system of this application can be enlarged. Right now people cannot add strangers as their friends. But maybe in the future the friends system can be added in. Then the SQLite can be used to store those kind of data to enhance the performance. If users do not change their friends list, then there will be no HTTP request for getting data. Just using the static one on local storage.

For Location Based Service part,

```
//set location manager and location variable
private LocationManager locationManager;
private Location location;
//use x and y to record users location
private double x;
private double y;
//Class position information
private final double CLASS_2464_Y = 114.26333902908577;
private final double CLASS_2464_X = 22.33760550858573;

private final double CLASS_LSK_Y = 114.26464130298577;
private final double CLASS_LSK_X = 22.33347407989984;

private final double CLASS_4619_Y = 114.26401378535724;
private final double CLASS_4619_X = 22.335136092645868;
//default x,y
```

```
                x = 0.0;
                y = 0.0;
                //set location manager value
                locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);
        //every 2 seconds get users location
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
           2000, 8, new LocationListener() {
              @Override
              public void onStatusChanged(String provider, int status,
                    Bundle extras) {
              }
                 @Override
                 public void onLocationChanged(Location location) {
                          // TODO Auto-generated method stub
                          location = locationManager
                                   .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                 }
                 @Override
                 public void onProviderEnabled(String provider) {
                          // TODO Auto-generated method stub
                          location = locationManager
                                    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                 }

                 @Override
                 public void onProviderDisabled(String provider) {
                          // TODO Auto-generated method stub
                 }
           });
                 //set button
                 Button autoLocate = (Button) findViewById(R.id.autoLocate);
                 //set listener
                 autoLocate.setOnClickListener(new OnClickListener() {
                          @Override
                          public void onClick(View v) {
                                   location = locationManager
                              .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                                   //get the locate
                                   if (location != null) {
                                            x = location.getLatitude();
                                   y = location.getLongitude();
                                   } else {
                          x = 0;
                          y = 0;
                        }
                 //Log.i("location", sb.toString());
                 double distance_2464 = Math.pow(x - CLASS_2464_X, 2) + Math.pow(y -
CLASS_2464_Y, 2);
                 double distance_lsk = Math.pow(x - CLASS_LSK_X, 2) + Math.pow(y - CLASS_LSK_Y,
2);
                 double distance_4619 = Math.pow(x - CLASS_4619_X, 2) + Math.pow(y -
CLASS_4619_Y, 2);
```

```
            // jump to search result activity, show search result
            if(distance_2464 <= distance_lsk && distance_2464 <= distance_4619){
                    Intent intent = new Intent(LocateActivity.this, SearchResultActivity.class);
                    intent.putExtra("u_room", "Academic");
                    LocateActivity.this.startActivity(intent);
            }
            else if(distance_lsk <= distance_2464 && distance_lsk <= distance_4619){
                    Intent intent = new Intent(LocateActivity.this, SearchResultActivity.class);
                    intent.putExtra("u_room", "lsk");
                    LocateActivity.this.startActivity(intent);
            }
            else{
                    Intent intent = new Intent(LocateActivity.this, SearchResultActivity.class);
                    intent.putExtra("u_room", "Enterprise");
                    LocateActivity.this.startActivity(intent);
            }

            }
    });
}
```

3. Testing and Evaluation

1.Version control:
Github : https://github.com/lzlcss/passby

2.Integration Testing:
The program integrates well with the other core device features.
It works well when suspending and resuming.
It works well when interrupting by coming message, calls.

3.Internationalization Testing:
Only show English version whatever the system language is.

4.Usability Testing:
The application is very easy to use.
Power consumption is low.

5.Performance Testing:
Resource and memory usage is normal. No memory leaks.

6.Conformance Testing:
No policies issues due to the name of user is hiding.

7.Screen Orientation Change Testing:
Layout is the same proportion as original.

8.Configuration Changes Testing:
It works well when change keyboard and language.

9.External Resources Dependence Testing:
Wifi or GPS is required by the application.

4. Conclusions

In user requirement area, an android application will be developed for social networking purpose, which is named as Passer-by@UST. However, different from the applications have already published on public, Passer-By focuses on HKUST students and tries to provide a vintage feel to users by using the concept of ‘chatting room’.

In technique area, because of the usage of Passer-by@UST, instant message and Google Maps tech are involved. For the implementation part, there mainly six activities which will offer users setting, searching rooms, getting their location and chatting with others.

In conclusion, we make a social software to let strangers chatting at the same time and in the same position for mainly the same thing. It is helpful of information exchanging among the people. And finally, shorten the distance between people.

5. References

[1] Android Instant Messaging Application, Pirngruder,
https://github.com/Pirngruber/AndroidIM

[2] Android Location Based Services, tutorialspoint, Retrieved Nov. 3rd,
http://www.tutorialspoint.com/android/android_location_based_services.htm

[3] Location Strategies, Retrieved Nov. 3rd,
http://developer.android.com/guide/topics/location/strategies.html

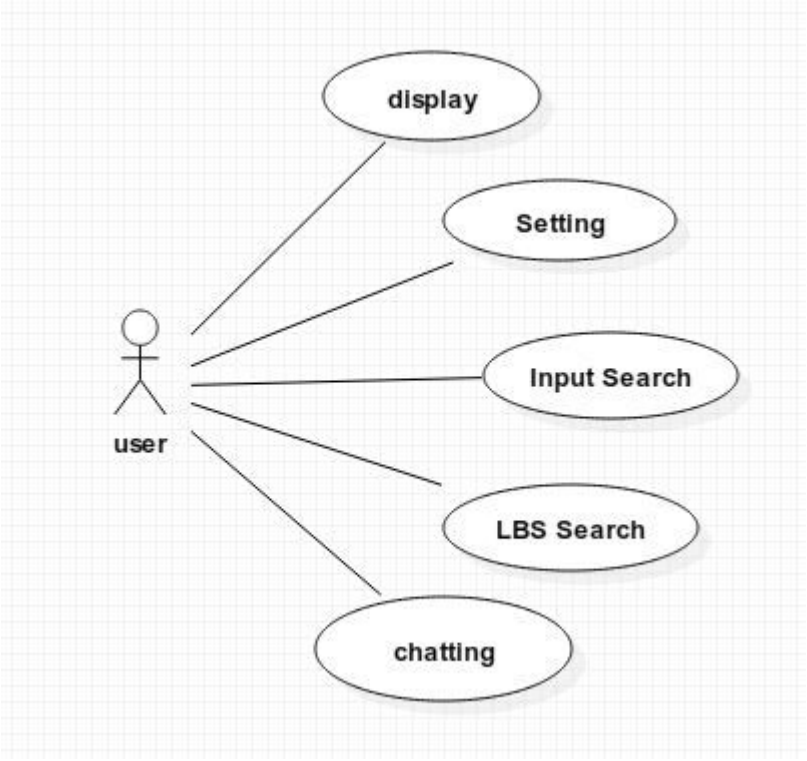
[4] Google Maps Android API v2, Retrieved Nov. 3rd,
http://developer.android.com/google/play-services/maps.html

[5] Firebase tools docs,
https://www.firebase.com/docs.html

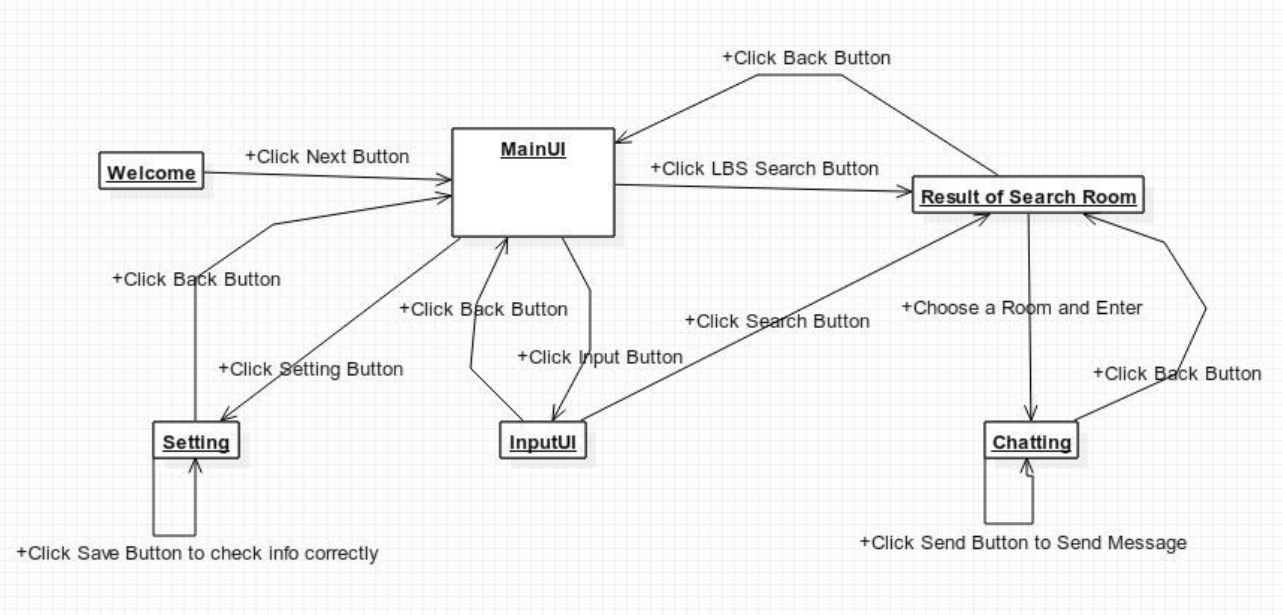
[6] SQLite docs,
http://www.sqlite.org/

Appendix A: Analysis Models

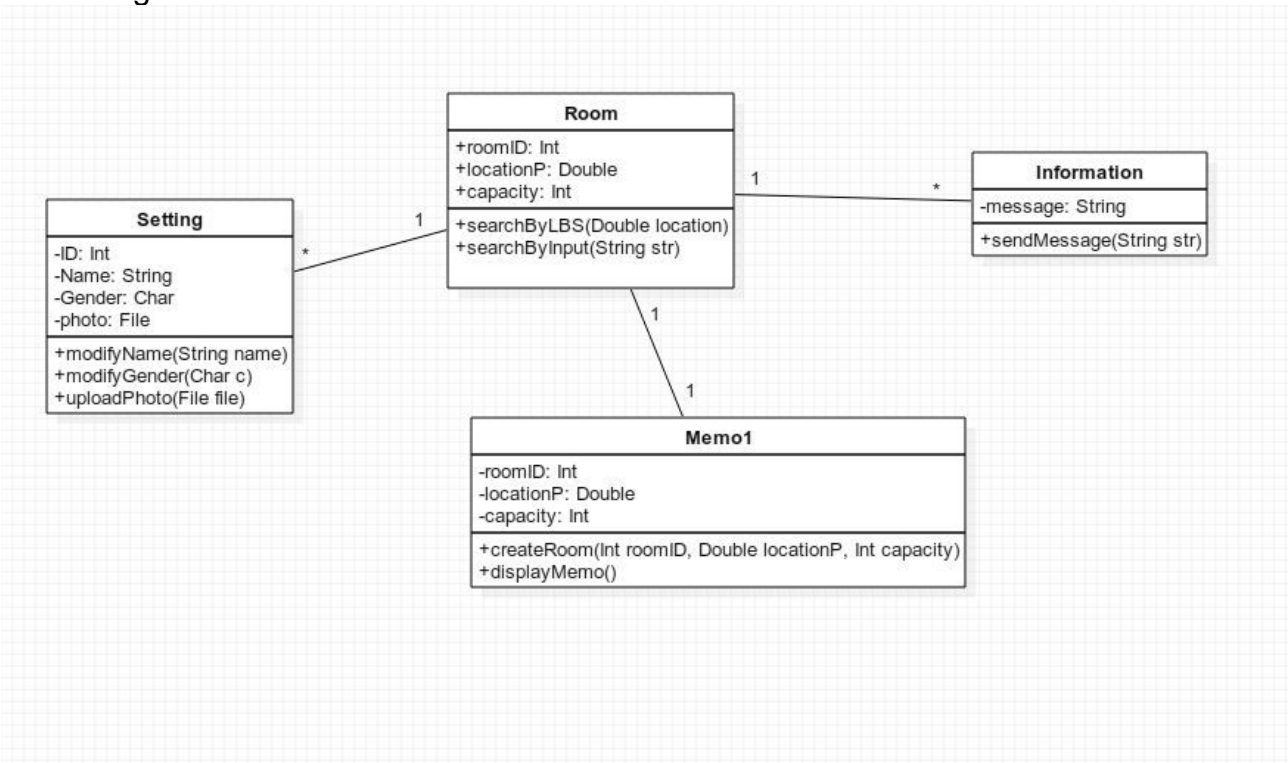
There are three diagrams to use: Use case diagram, state-transition diagram and class diagram.
Use case diagram:



State-transition diagram:



Class diagram:



Appendix B: Server Storage Part