Monash University  
FIT5183 Mobile and Distributed Computing Systems  
Practical Assignment Phase 2 – 2017 (Semester 1)

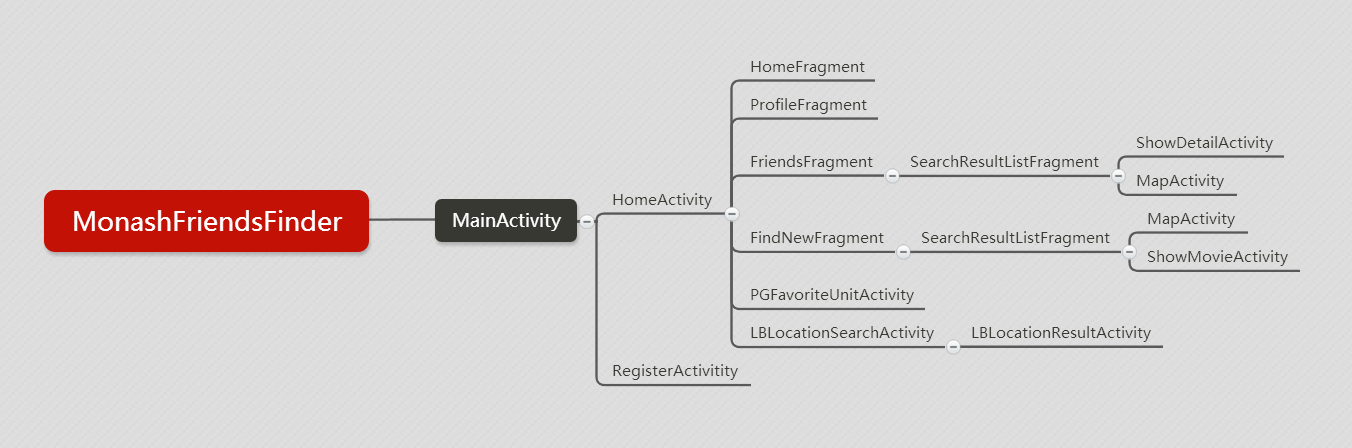
This assignment is a preferable attempt at an Android project and requires us to create a student oriented app. This task requires us to give app many functions, such as registration, login, personal information inquiries, personal information changes, weather forecasts, film presentations, access, Registration, login, personal information inquiries, personal information changes, weather forecasts, film presentations, adding friends and other complex functions and other complex functions Our team worked together to complete all of the requirements of the project. In addition, we add a lot of detail to the design that makes the project better. Next, I will introduce our work and summarize each of them in the order of task 1 to task 7. In the end, we will draw a conclusion and publish our thoughts on the work.

Task assignment:

Zhu Yilin: T1; T3-Register,Home; T4-search;T5-SQLite; T6;T7

Li Rongzhou: T1; T2; T3-Log; T4-updating,friends; T5-sharedPreference; T6

**Ⅰ. Flow Chart**



**Ⅱ. Main Task**

**Task 1: Invoking public web APIs (16 marks):**

**a)**

At first, we need to provide current weather information for location of the login student by the help of weather API. The weather information should displayed in the home page of this project.

Correspond to this requirement, we create the URL though an appropriate weather API and get the weather information of student location. However, the information from API is the type of JSON. In order to get valid weather information, we wrote the appropriate program to parse the JSON string acquired.

Weather API provider’s website: <https://www.heweather.com/documents/api/v5.>

**Returned json being parsed looks like below.**

{

"HeWeather5"**:** [

{

"basic"**:** {

//基本信息"city": "北京",

//城市名称"cnty": "中国",

//国家"id": "CN101010100",

//城市ID"lat": "39.904000",

//城市维度"lon": "116.391000",

//城市经度"prov": "北京",

//城市所属省份（仅限国内城市）"update": {

//更新时间"loc": "2016-08-31 11:52",

//当地时间"utc": "2016-08-31 03:52"//UTC时间

}

},

"now"**:** {

//实况天气"cond": {

//天气状况"code": "104",

//天气状况代码"txt": "阴"//天气状况描述

},

"fl"**:** "11",

//体感温度"hum": "31",

//相对湿度（%）"pcpn": "0",

//降水量（mm）"pres": "1025",

//气压"tmp": "13",

//温度"vis": "10",

//能见度（km）"wind": {

//风力风向"deg": "40",

//风向（360度）"dir": "东北风",

//风向"sc": "4-5",

//风力"spd": "24"//风速（kmph）

}

},

"status"**:** "ok"//接口状态

}

]

}

**Main code:**

**1. Get location city name**

// use Baidu API to get local location(latitude and longitude and city)

public class MyLocationListener implements BDLocationListener {

@Override

public void onReceiveLocation(BDLocation **location**) {

Double lat **=** location**.**getLatitude();

Double lon **=** location**.**getLongitude();

final String city **=** location**.**getCity();

Log**.**d("MainActivity", "lat" **+** lat);

Log**.**d("MainActivity", "lon" **+** lon);

Log**.**d("MainActivity", "city" **+** city);

editor**.**putString("city", city);

editor**.**commit();

**new** Handler()**.**postDelayed(**new** Runnable() {

public void run() {

FragmentManager fragmentManager **=** getFragmentManager();

//默认显示HomeFragment

fragmentManager**.**beginTransaction()**.**replace(R**.**id**.**content\_frame, **new**

HomeFragment())**.**commit();

}

}, 1000);

}

**2. GetWeather method in RestClient.java (passing city as param)**

**public** **static** String getWeather(String city) {

final String URI **=** "https://free-api.heweather.com/v5/now?city=" **+** city **+** "&key=" **+** "c91fa1f1c00147108ad7d7b51cb89eab";//initialise

URL **url** **=** null;

HttpURLConnection conn **=** null;

String textResult **=** "";//Making HTTP request

**try** {

**url** **=** **new** URL(URI);//open the connection

conn **=** (HttpURLConnection) **url**.openConnection();//set the timeout

conn.setReadTimeout(10000);

conn.setConnectTimeout(15000);//set the connection method to GET

conn.setRequestMethod("GET");//add http headers to set your response type to json

conn.setRequestProperty("Content-Type", "application/json");

conn.setRequestProperty("Accept", "application/json");//Read the response

Scanner inStream **=** **new** Scanner(conn.getInputStream());//read the input steream and store it as string

**while** (inStream.hasNextLine()) {

textResult **+=** inStream.nextLine();

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

conn.**disconnect**();

}

Log.d("RestClient", "获取到的weather数据是：" **+** textResult);

**return** textResult;

}

1. **invoke getWeather method and parse the return string for getting info we needed in HomeFragment.java**

**new** AsyncTask**<**String, Void, String**>**() {

@Override

protected String doInBackground(String... params) {

Log.d("HomeFragment", "city：" **+** finalCity);

String info = RestClient.getWeather(finalCity);

**return** info;

}

@Override

protected void onPostExecute(String info) {

**if** (**!**info.equals("{}")) {

//Log.d("HomeFragment", "weather info is：" + info);

info = info.substring(15, info.length() **-** 2);

Log.d("HomeFragment", "weather info is：" **+** info);

//将字符串转换成jsonObject对象

**try** {

JSONObject myJsonObject1 = **new** JSONObject(info);

//获取对应的值

String now, weather, cond, tmp, wind, windStrength, windDirection;

now = myJsonObject1.getString("now");

// Log.d("HomeFragment", "weather info--now is：" + now);

JSONObject myJsonObject2 = **new** JSONObject(now);

cond = myJsonObject2.getString("cond");

JSONObject myJsonObject3 = **new** JSONObject(cond);

weather = myJsonObject3.getString("txt");

Log.d("HomeFragment", "weather info--weather is：" **+** weather);

tmp = myJsonObject2.getString("tmp");

Log.d("HomeFragment", "weather info--tmp is：" **+** tmp);

wind = myJsonObject2.getString("wind");

JSONObject myJsonObject4 = **new** JSONObject(wind);

windDirection = myJsonObject4.getString("dir");

Log.d("HomeFragment", "weather info--windDirection is：" **+** windDirection);

windStrength = myJsonObject4.getString("sc");

Log.d("HomeFragment", "weather info--weather is：" **+** windStrength);

mtv\_country.setText(finalCity);

tmp = tmp **+** "℃";

mtv\_weather.setText(weather);

mtv\_temperature.setText(tmp);

mtv\_wind.setText(windDirection);

windStrength = "风力: " **+** windStrength;

mtv\_windStrength.setText(windStrength);

} **catch** (JSONException mE) {

mE.printStackTrace();

}

}

}

}.execute();

**Rendering:**

**we get the weather, wind direction, wind strength, time, city and temperature info and show in the home page(picture 1).**

****

**pic 1. Homepage showing time and weather info**

**b&c**

Secondly, we choose douban API to get a brief description of the student’s favourite movie. Douban movie is the most authoritative movie ratings and wonderful video sharing site, to provide the latest film introduction and comments.

Movie info API provider’s website: <https://movie.douban.com/.>

We use getMovieInfo method to get json response using this API and parson it for getting and storing important and brief movie information attracting us. Thanks to website: <http://www.bejson.com/.> This website providing function of formatting online json. By using this website, we can see where info we want and how to parsing them out.

**part json returned looks like below after being formatted:**

{

"count": 20,

"start": 0,

"total": 43,

"subjects": [

{

"rating": {

"max": 10,

"average": 7.5,

"stars": "40",

"min": 0

},

"genres": [

"悬疑",

"恐怖"

],

"title": "逃出绝命镇",

"casts": [

{

"alt": "https://movie.douban.com/celebrity/1316928/",

"avatars": {

"small": "http://img3.doubanio.com/img/celebrity/small/38838.jpg",

"large": "http://img3.doubanio.com/img/celebrity/large/38838.jpg",

"medium": "http://img3.doubanio.com/img/celebrity/medium/38838.jpg"

},

"name": "丹尼尔·卡卢亚",

"id": "1316928"

},

{

"alt": "https://movie.douban.com/celebrity/1318760/",

"avatars": {

"small": "http://img7.doubanio.com/img/celebrity/small/1402641640.85.jpg",

"large": "http://img7.doubanio.com/img/celebrity/large/1402641640.85.jpg",

"medium": "http://img7.doubanio.com/img/celebrity/medium/1402641640.85.jpg"

},

"name": "艾莉森·威廉姆斯",

"id": "1318760"

},

{

"alt": "https://movie.douban.com/celebrity/1002681/",

"avatars": {

"small": "http://img7.doubanio.com/img/celebrity/small/4291.jpg",

"large": "http://img7.doubanio.com/img/celebrity/large/4291.jpg",

"medium": "http://img7.doubanio.com/img/celebrity/medium/4291.jpg"

},

"name": "凯瑟琳·基纳",

"id": "1002681"

}

],

"collect\_count": 45059,

"original\_title": "Get Out",

"subtype": "movie",

"directors": [

{

"alt": "https://movie.douban.com/celebrity/1032808/",

"avatars": {

"small": "http://img7.doubanio.com/img/celebrity/small/40682.jpg",

"large": "http://img7.doubanio.com/img/celebrity/large/40682.jpg",

"medium": "http://img7.doubanio.com/img/celebrity/medium/40682.jpg"

},

"name": "乔丹·皮尔",

"id": "1032808"

}

],

"year": "2017",

"images": {

"small": "http://img3.doubanio.com/view/movie\_poster\_cover/ipst/public/p2416438068.webp",

"large": "http://img3.doubanio.com/view/movie\_poster\_cover/lpst/public/p2416438068.webp",

"medium": "http://img3.doubanio.com/view/movie\_poster\_cover/spst/public/p2416438068.webp"

},

"alt": "https://movie.douban.com/subject/26688480/",

"id": "26688480"

},

...

**Maincode:**

**1. getMovieInfo method used, this method return an array. we choose to get image, title, english title, released year, director, stars and introduction of a movie.**

**public** **static** String[] getMovieInfo(String movie) {

// store info in an array

String movieInfo[] **=** **new** String[7];

OkHttpClient client **=** **new** OkHttpClient();//

Request **request** **=** **new** Request.Builder().**url**("http://api.douban.com/v2/movie/search?q=" **+** movie).build();

**try** {

Response **response** **=** client.newCall(**request**).**execute**();

String responseData **=** **response**.body().string();

Log.d("RestClient", responseData);

JSONObject jsonObject **=** **new** JSONObject(responseData);

JSONArray subjects **=** jsonObject.getJSONArray("subjects");

JSONObject subject **=** subjects.getJSONObject(0);

String **title** **=** subject.getString("title");

StringBuilder starSb **=** **new** StringBuilder();

JSONArray casts **=** subject.getJSONArray("casts");

**for** (**int** j **=** 0; j < casts.**length**() **-** 1; j**++**) {

JSONObject cast **=** casts.getJSONObject(j);

starSb.append(cast.getString("name") **+** ",");

}

**if** (casts.**length**() > 0) {

starSb.append(casts.getJSONObject(casts.**length**() **-** 1).getString("name"));

}

StringBuilder directorsSb **=** **new** StringBuilder();

JSONArray directors **=** subject.getJSONArray("directors");

**for** (**int** k **=** 0; k < directors.**length**() **-** 1; k**++**) {

JSONObject director **=** directors.getJSONObject(k);

directorsSb.append(director.getString("name,") **+** ",");

}

**if** (directors.**length**() > 0) {

directorsSb.append(directors.getJSONObject(directors.**length**() **-** 1).getString("name"));

}

String **year** **=** subject.getString("year");

String englishTitle **=** subject.getString("original\_title");

String image **=** subject.getJSONObject("images").getString("large");

**int** id **=** subject.getInt("id");

**request** **=** **new** Request.Builder().**url**("http://api.douban.com/v2/movie/subject/" **+** id).build();

**response** **=** client.newCall(**request**).**execute**();

responseData **=** **response**.body().string();

JSONObject summaryJson **=** **new** JSONObject(responseData);

String Introduction **=** summaryJson.getString("summary");

movieInfo[0] **=** image;

movieInfo[1] **=** **title**;

movieInfo[2] **=** englishTitle;

movieInfo[3] **=** **year**;

movieInfo[4] **=** directorsSb.**toString**();

movieInfo[5] **=** starSb.**toString**();

movieInfo[6] **=** Introduction;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** movieInfo;

}

1. **invoke this method in ShowMovieActivity.java**

**new** AsyncTask**<**String, Void, String[]**>**() {

@Override

protected String[] doInBackground(String... params) {

String info = null;

String[] movieInfo = RestClient.getMovieInfo(movie);

// invoke method in RestClient to get movie's info

**for** (int i = 0; i **<** movieInfo.length; i**++**) {

Log.d("ShowMovieActivity", "get movieinfo: " **+** movieInfo[i]);

}

**return** movieInfo;

}

@Override

protected void onPostExecute(String[] info) {

// set these info into textviews of activity

mwv.getSettings().setJavaScriptEnabled(true);

mwv.setWebViewClient(**new** WebViewClient());

mwv.loadUrl(info[0]);

mtv\_title.setText(info[1]);

mtv\_englishTitle.setText(info[2]);

mtv\_year.setText(info[3]);

mtv\_dirctor.setText(info[4]);

mtv\_star.setText(info[5]);

mtv\_introduction.setText(info[6]);

}

}.execute();

**Result：we show these info in a new dialog style activity. For realize this function, we setting a style in styles.xml and write a theme statement when we state this activity in manifest.xml.**

**style setting:**

**<**style name**=**"dialogstyle"**>**

<!--设置dialog的背景-->

**<**item name**=**"android:windowBackground"**>**@android:color/transparent**<**/item**>**

<!--设置Dialog的windowFrame框为无-->

**<**item name**=**"android:windowFrame"**>**@null**<**/item**>**

<!--设置无标题-->

**<**item name**=**"android:windowNoTitle"**>**true**<**/item**>**

<!--是否浮现在activity之上-->

**<**item name**=**"android:windowIsFloating"**>**true**<**/item**>**

<!--是否半透明-->

**<**item name**=**"android:windowIsTranslucent"**>**true**<**/item**>**

<!--设置窗口内容不覆盖-->

**<**item name**=**"android:windowContentOverlay"**>**@null**<**/item**>**

<!--设置动画，在这里使用让它继承系统的Animation.Dialog-->

**<**item name**=**"android:windowAnimationStyle"**>**@android:style/Animation.Dialog**<**/item**>**

<!--背景是否模糊显示-->

**<**item name**=**"android:backgroundDimEnabled"**>**true**<**/item**>**

**<**/style**>**

**theme statement:**

<activity

android:name=".UI.ShowMovieActivity"

android:theme="@style/dialogstyle"/>

**Rendering:**

****

**pic 2. showMovieInfoActivity**

For realizing the final effect of showing an activity looks like a dialog, we find a method in this blog: <http://blog.csdn.net/student9128/article/details/51895383.>

**Task 2 – Android Client of RESTful WS (6 marks):**

**a) HttpURLConnection:**

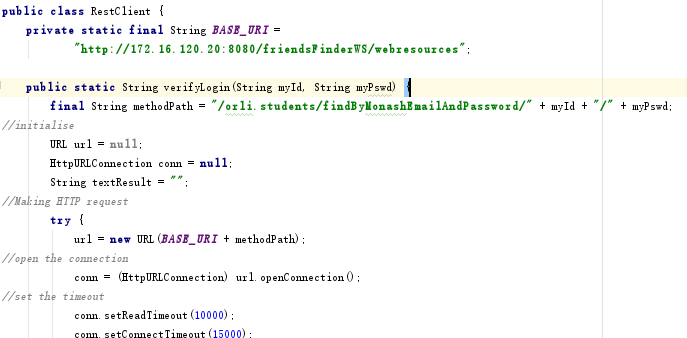
In this task, we need to create an android application client to connect to the server and consume the RESTful web service created in Phase 1. We use two kinds of connection ways: http HttpURLConnection and OkHttpClient. Most method are written by using HttpURLConnection. However, when we try to write getMovieInfo method using douban API, we found it’s not work but when we use OKHttpClient, json result can be got. OkHttp is a modern, fast and efficient Http client. It is the Http client for Android. Very efficient, support for SPDY, connection pool, GZIP and HTTP cache.

Beside this, we need using the AsynchTask approach to Access data and execute all the queries from the server side (and the web service).

Because of the partly repeat of task 2 , 4 , 6, we will just explain the basic function of HttpURLConnection and AsynchTask.

**HttpURLConnection:**

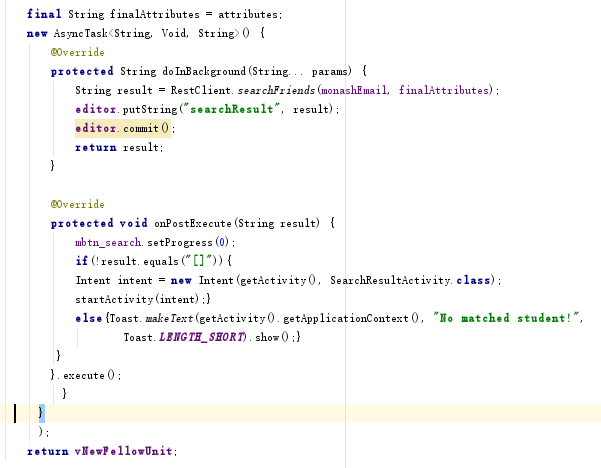
We create URL, which is equal to BASE\_URI plus methodpath. As the picture 3 shows, methodpath Correspond to the specific methods on the RESTful web service. Using the URL, we can get the information from the database.



**Pic 3. HttpURLConnection information**

**b) AsynchTask:**

Using AsynchTask to invoke compiled method to operate the database through URI. As picture 4 shows below, we create this AsyncTask to invoke searchFriends method which is compiled before to get the friends information with the same attributes.



**Pic 4. AsyncTask to get the friends information by the same features**

**Task 3 Login, Subscription, and Main Screen (20 marks):**

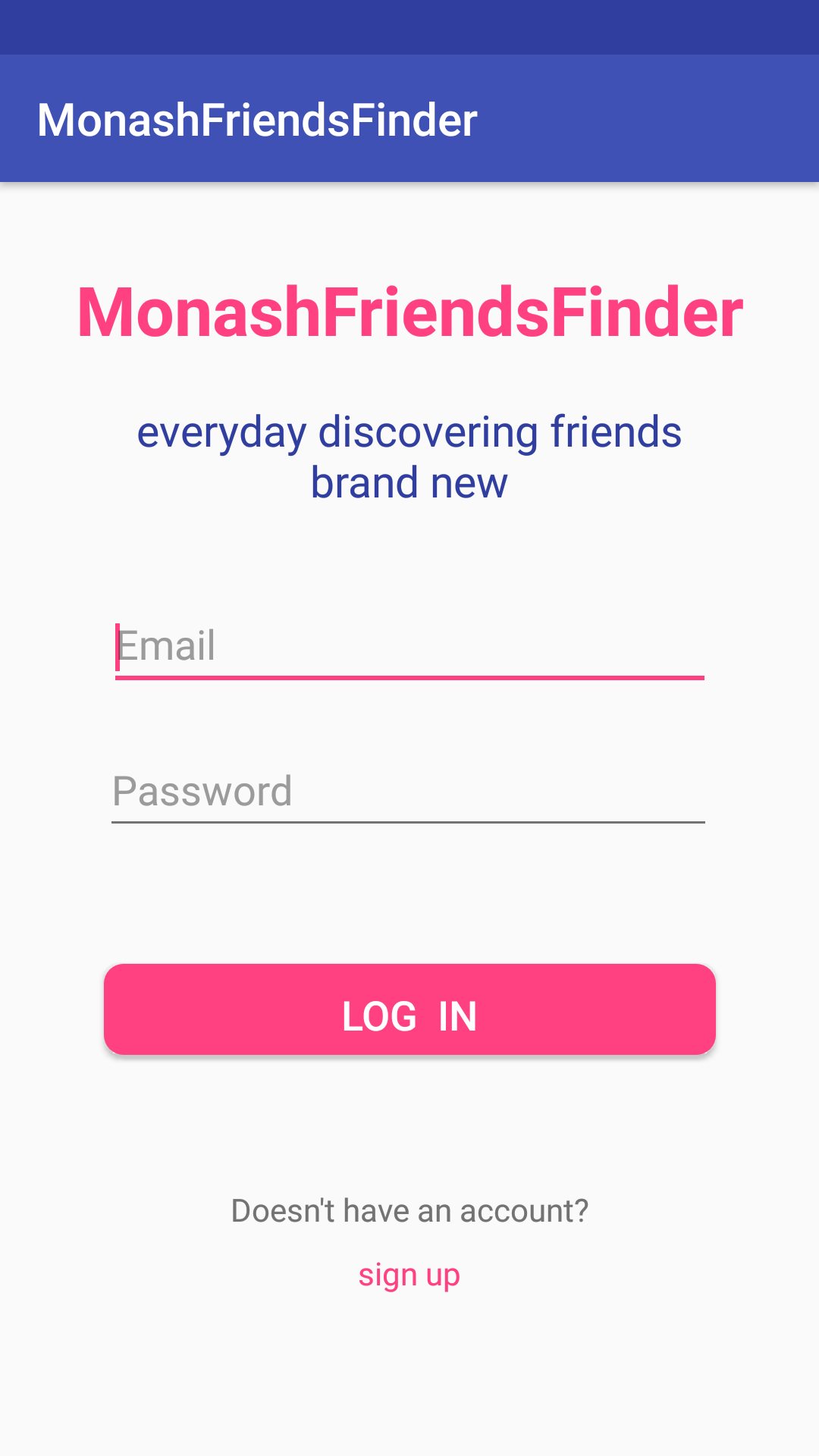
Task 3, mainly divided into 3 blocks, registration function, login function and home page.

**a)**

In login function, users type in their Email and password. After click button‘Log in’, project use the method verifyLogin(String myId, String myPswd) verify Email and password you type in with the information in database. When the Email which the user entered cannot query, the system will prompt ‘Incorrect username or password.’When the password entered by the user is incorrect, the system will prompt ‘Incorrect password’. If there is a network error and other problems, make the user unable to query the database information, the system will inform the user ’Connection time out!’ in a matter of seconds.

**Renderings:**

**MainActivity is witten as a login activity.**



**Pic 5. loginActivity**

**Main code:**

public class MainActivity extends AppCompatActivity {

//使用SharedPreferences进行读取

private SharedPreferences pref;

//使用SharedPreferences.Editor进行存储

private SharedPreferences**.**Editor editor;

@Override

protected void onCreate(Bundle **savedInstanceState**) {

**super.**onCreate(savedInstanceState);

setContentView(R**.**layout**.**activity\_main);

final EditText et\_ID **=** (EditText) findViewById(R**.**id**.**et\_ID);

final EditText et\_pswd **=** (EditText) findViewById(R**.**id**.**et\_pswd);

// CircularProgressButton element

final CircularProgressButton btn\_login **=** (CircularProgressButton) findViewById(R**.**id**.**btn\_login);

TextView tv\_sign\_up **=** (TextView) findViewById(R**.**id**.**tv\_sign\_up);

//第一个参数：文件名，没有则新建。第二个参数：写入模式-覆盖

pref **=** getSharedPreferences("admin", MODE\_PRIVATE);

//获取SharedPreferences.Editor对象

editor **=** pref**.**edit();

// turn on indeterminate progress

btn\_login**.**setIndeterminateProgressMode(true);

btn\_login**.**setOnClickListener(**new** View.OnClickListener() {

@Override

public void onClick(View **v**) {

btn\_login**.**setProgress(20);

final String myId **=** et\_ID**.**getText()**.**toString();

String myPswd **=** et\_pswd**.**getText()**.**toString();

// Validate user input

**if** (myId**.**isEmpty()) {

et\_ID**.**setError("email address is required!");

btn\_login**.**setProgress(0);

**return**;

}

**if** (myPswd**.**isEmpty()) {

et\_pswd**.**setError("password is required!");

btn\_login**.**setProgress(0);

**return**;

}

//create an anonymous AsyncTask

final String hashedPswd **=** md5Hash(myPswd)**.**substring(0,20);

final String finalMyPswd **=** myPswd;

Log**.**d("MainActivity", "hashedPswd is：" **+** hashedPswd);

**new** AsyncTask<String, Void, String>() {

@Override

protected String doInBackground(String... **params**) {

String info;

// Log.d("xxx", "你的信息1" RestClient.verifyLogin(myId,hashedPswd));

**if** ((info **=** RestClient**.**verifyLogin(myId, hashedPswd))**.**equals("[]")) {

// Log.d("xxx", "你的信息2" + RestClient.getId(myId));

**if** (**!**(RestClient**.**getId(myId))**.**equals("[]")) {

**return** "2";

}

**return** "1";

} **else** {

**return** info;

}

}

@Override

protected void onPostExecute(String **info**) {

**if** (info**.**equals("1")) {

et\_ID**.**setError("Incorrect username or password.");

btn\_login**.**setProgress(0);

} **else** **if** (info**.**equals("2")) {

et\_pswd**.**setError("Incorrect password.");

btn\_login**.**setProgress(0);

} **else** **if** (info**.**equals("")) {

btn\_login**.**setProgress(0);

Toast**.**makeText(getApplicationContext(), "Connection time out!", Toast**.**LENGTH\_SHORT)**.**show();

} **else** {

**try** {

JSONArray jsonArray **=** null;

jsonArray **=** **new** JSONArray(info);

JSONObject jsonObj **=** jsonArray**.**getJSONObject(0);

String firstName **=** (String) jsonObj**.**get("firstName");

final String surname **=** (String) jsonObj**.**get("surname");

String doB **=** (String) jsonObj**.**get("doB");

String gender **=** (String) jsonObj**.**get("gender");

String course **=** (String) jsonObj**.**get("course");

String studyMode **=** (String) jsonObj**.**get("studyMode");

String address **=** (String) jsonObj**.**get("address");

String suburb **=** (String) jsonObj**.**get("suburb");

String nationality **=** (String) jsonObj**.**get("nationality");

String nativeLanguage **=** (String) jsonObj**.**get("nativeLanguage");

String favoriteSport **=** (String) jsonObj**.**get("favoriteSport");

String favoriteMovieType **=** (String) jsonObj**.**get("favoriteMovieType");

String favoriteMovie **=** (String) jsonObj**.**get("favoriteMovie");

String favouriteUnit **=** (String) jsonObj**.**get("favouriteUnit");

String currentJob **=** (String) jsonObj**.**get("currentJob");

String subscriptionDate **=** (String) jsonObj**.**get("firstName");

//清空旧的cookie

editor**.**clear();

editor**.**commit();

editor**.**putString("monashEmail", myId);

// Log.d("mainActivity", "获取到的info是：" + info);

// Log.d("mainActivity", "获取到的address是：" + address);

// store brief info in sharepreference

editor**.**putString("password", finalMyPswd);

editor**.**putString("firstName", firstName);

editor**.**putString("surname", surname);

editor**.**putString("doB", doB);

editor**.**putString("gender", gender);

editor**.**putString("course", course);

editor**.**putString("studyMode", studyMode);

editor**.**putString("address", address);

editor**.**putString("suburb", suburb);

editor**.**putString("nationality", nationality);

editor**.**putString("nativeLanguage", nativeLanguage);

editor**.**putString("favoriteSport", favoriteSport);

editor**.**putString("favoriteMovieType", favoriteMovieType);

editor**.**putString("favoriteMovie", favoriteMovie);

editor**.**putString("favouriteUnit", favouriteUnit);

editor**.**putString("currentJob", currentJob);

editor**.**putString("subscriptionDate", subscriptionDate);

editor**.**commit();

} **catch** (JSONException mE) {

mE**.**printStackTrace();

}

**new** Handler()**.**postDelayed(**new** Runnable() {

public void run() {

Toast**.**makeText(getApplicationContext(), "welcome back!", Toast**.**LENGTH\_SHORT)**.**show();

Intent intent **=** **new** Intent(MainActivity**.this**, HomeActivity**.**class);

//start intent

startActivity(intent);

finish();

}

}, 1000);

}

}

}**.**execute();

}

});

tv\_sign\_up**.**setOnClickListener(**new** View.OnClickListener() {

@Override

public void onClick(View **v**) {

Intent intent **=** **new** Intent(MainActivity**.this**, RegisterActivity**.**class);

//start intent

startActivity(intent);

finish();

}

});

}

public boolean onKeyDown(int **keyCode**, KeyEvent **event**) {

**if** (keyCode **==** KeyEvent**.**KEYCODE\_BACK) {

// 创建退出对话框

AlertDialog isExit **=** **new** AlertDialog.Builder(**this**)**.**create();

// 设置对话框标题

isExit**.**setTitle("SYSTEM HINT");

// 设置对话框消息

isExit**.**setMessage("Do you want to quit?");

// 添加选择按钮并注册监听

isExit**.**setButton(DialogInterface**.**BUTTON\_NEGATIVE, "NO", listener);

isExit**.**setButton(DialogInterface**.**BUTTON\_POSITIVE, "YES", listener);

// 显示对话框

isExit**.**show();

}

**return** false;

}

/\*\*

\* 监听对话框里面的button点击事件

\*/

DialogInterface**.**OnClickListener listener **=** **new** DialogInterface.OnClickListener() {

public void onClick(DialogInterface **dialog**, int **which**) {

**switch** (which) {

**case** AlertDialog**.**BUTTON\_POSITIVE**:**// "确认"按钮退出程序

finish();

**break**;

**case** AlertDialog**.**BUTTON\_NEGATIVE**:**// "取消"第二个按钮取消对话框

**break**;

**default:**

**break**;

}

}

};

public static String md5Hash(String **string**) {

**if** (TextUtils**.**isEmpty(string)) {

**return** "";

}

MessageDigest md5 **=** null;

**try** {

md5 **=** MessageDigest**.**getInstance("MD5");

byte[] bytes **=** md5**.**digest(string**.**getBytes());

String result **=** "";

**for** (byte b **:** bytes) {

String temp **=** Integer**.**toHexString(b **&** 0xff);

**if** (temp**.**length() **==** 1) {

temp **=** "0" **+** temp;

}

result **+=** temp;

}

**return** result;

} **catch** (NoSuchAlgorithmException e) {

e**.**printStackTrace();

}

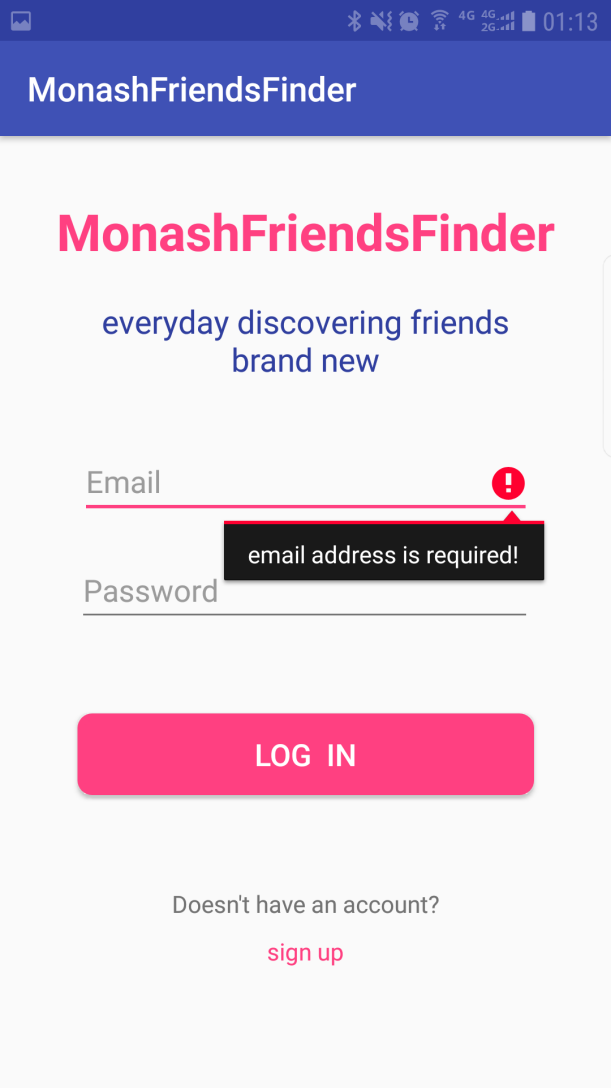
**return** "";

}

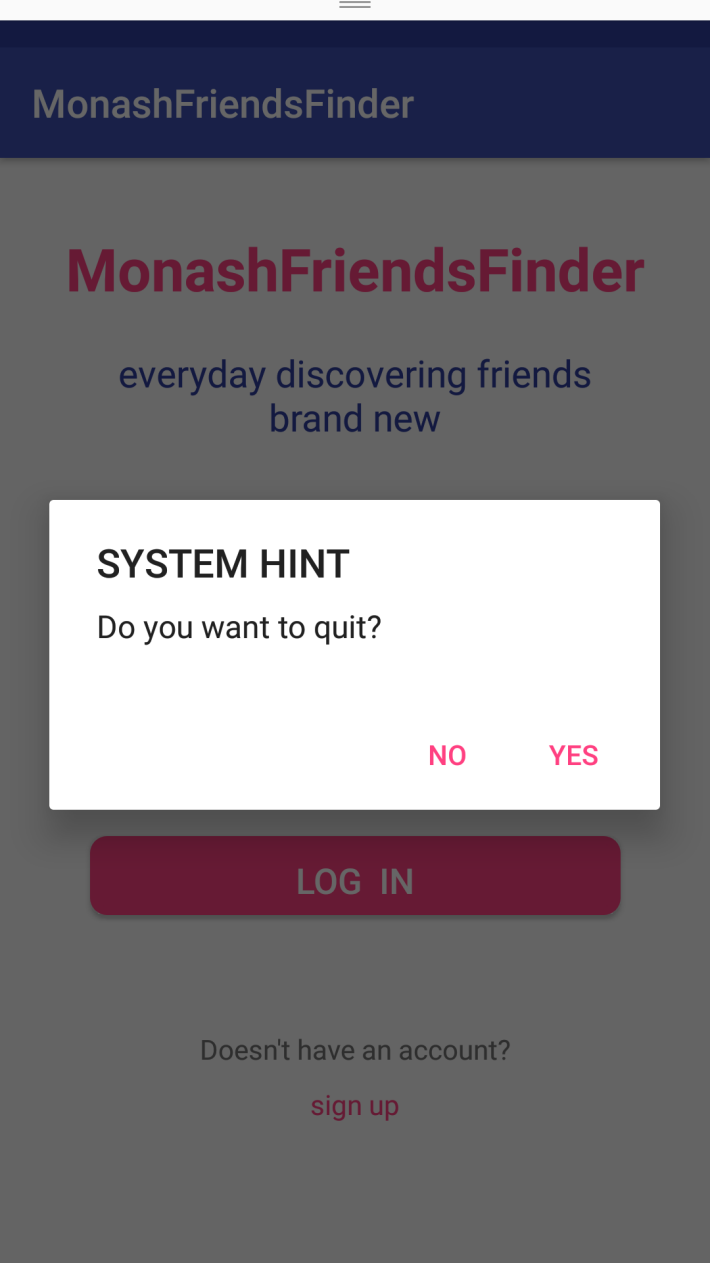
}

We made many human design in this app including in login page. For example, when a user does not enter any character at the Email text, there is a mark in the shape of exclamation appearing in this line. The icon will prompt your Email address is required and a hint for reminding you of inputting wrong email or password(Show in Picture 6&7). When you move the mouse over the icon. Beside this, when you click the button to leave the app, the system will let you confirm if you really want to leave. (Show as Picture 8).

For realizing ‘skip login’, we set that when a person successful registered, app will auto jump to the home page rather than letting he/she log in again.

Picture 6&7 Warning of email address and wrong password



**Pic 8. Confirming whether quit or not**

**b)**

**we use MD5 for hashing password and storeit in dataBase. This method is used in situation like Registerring, after inputting password in loginActivity and uploading profile.**

**Main code:**

public static String md5Hash(String string) {

**if** (TextUtils.isEmpty(string)) {

**return** "";

}

MessageDigest md5 = null;

**try** {

md5 = MessageDigest.getInstance("MD5");

byte[] bytes = md5.digest(string.getBytes());

String result = "";

**for** (byte b : bytes) {

String temp = Integer.toHexString(b & 0xff);

**if** (temp.length() **==** 1) {

temp = "0" **+** temp;

}

result **+**= temp;

}

**return** result;

} **catch** (NoSuchAlgorithmException e) {

e.printStackTrace();

}

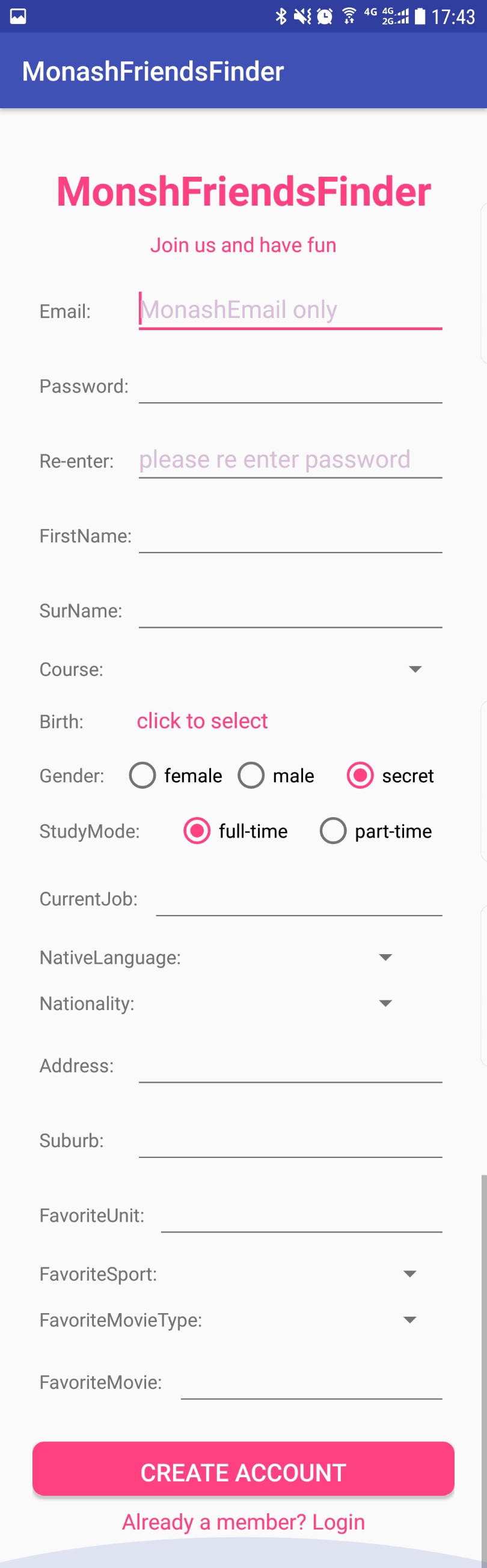
**return** "";

}

**c)**

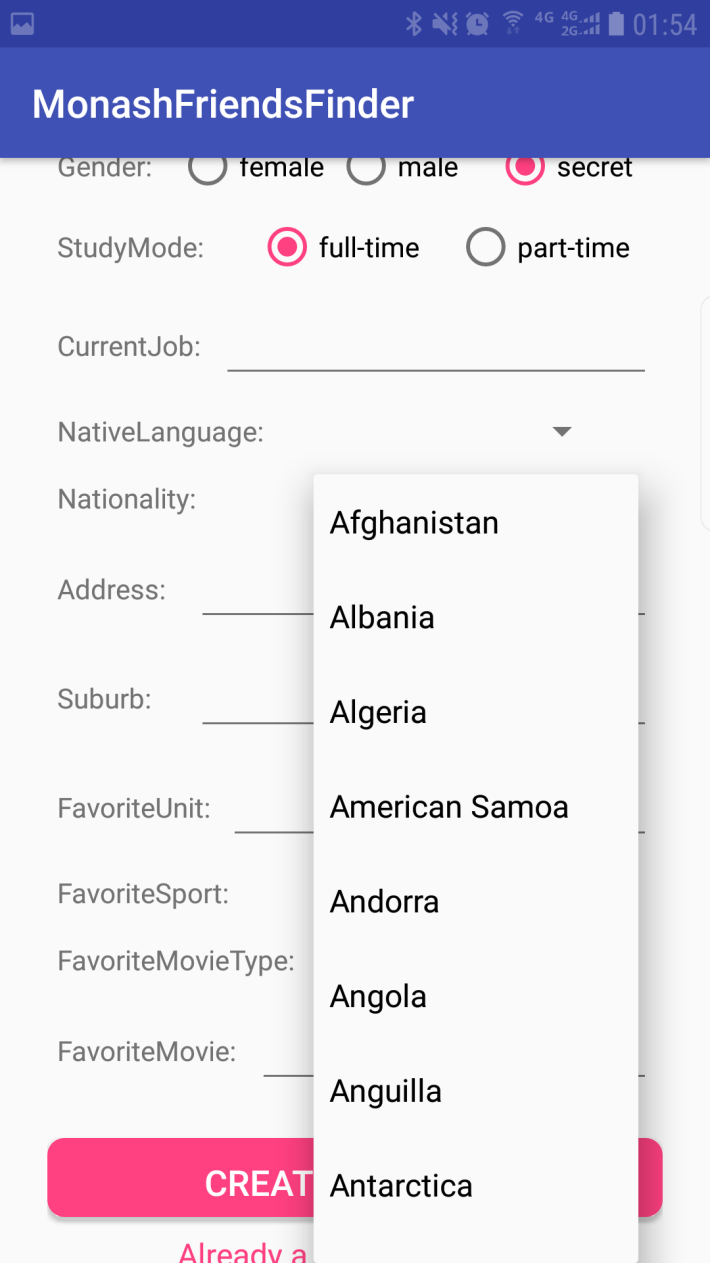
In registration function, new users can create their own accounts. New students need to input their own personal information, like Email, password, name etc. (show as Picture 9). We make the design that email, password, first name, surname must be entered.

According to requirements, we use spinners (Picture 10) for choosing course, favoriteMovieType, nationnality and nativeLanguage. attributes and a date picker (Picture 11) to pick a date. The chosen units of favourite sports and movie spinners are written in String.xml, and chosen units of the other two spinners are written at assets as the type of txt. The content mentioned earlier is also mentioned in task 5. When you click the button(CREATE ACCOUNT), the information user has already register will be added to the database via the method signUp(Student stu), and then you can login to the app via your registered information.



**Pic 9. New user registration interface**

We write five spinners. Three of them(course favoriteSport and favoriteMovieType) load message from string Array in strings.xml. The others(nationality and nativeLanguage) load message from SQLiteDataBase for the rason that them need a great deal of information. We store these info in two txt files in assert file and load them to SQLiteDataBase when we initialize them dataBase and create two tables for storing them two. Then when we initialize the registerActivity and profileFragment, they will read info from DataBase and stringArrays and fill themselves.



**Pic 10. spinner example of nationality**

**Main code for spinner:**

// listener for spinner

msp\_nation.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {

public void onItemSelected(AdapterView<?> parent, View view,

int position, long id) {

**if** (isSpinnerFirst[1]) {

//第一次初始化spinner时，不显示默认被选择的第一项即可

view.setVisibility(View.INVISIBLE) ;

}

isSpinnerFirst[1] = false ;

}

@Override

public void onNothingSelected(AdapterView<?> parent) {

}

});

// load nation info form SQLiteDateBase, we need invoke this method in onCreat().

**public** **void** loadNationSpinnerData() {

**try** {

dbManager.**open**();

} **catch** (SQLException e) {

e.printStackTrace();

}

// Spinner Drop down elements

List<String> lables **=** dbManager.getAllNation();

// Creating adapter for spinner

ArrayAdapter<String> dataAdapter **=** **new** ArrayAdapter<String>(**this**,

android.R.layout.simple\_spinner\_item, lables);

// Drop down layout style - list view with radio button

dataAdapter

.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

// attaching data adapter to spinner

msp\_nation.setAdapter(dataAdapter);

dbManager.**close**();

}

// stringArray in strings.xml

<string-array name="sports\_array">

<item>American Football</item>

<item>Badminton</item>

<item>Baseball</item>

<item>Basketball</item>

<item>Boxing</item>

<item>Cricket</item>

<item>Cycling</item>

<item>Football/Soccer</item>

<item>Golf</item>

<item>Gymnastics</item>

<item>Rugby</item>

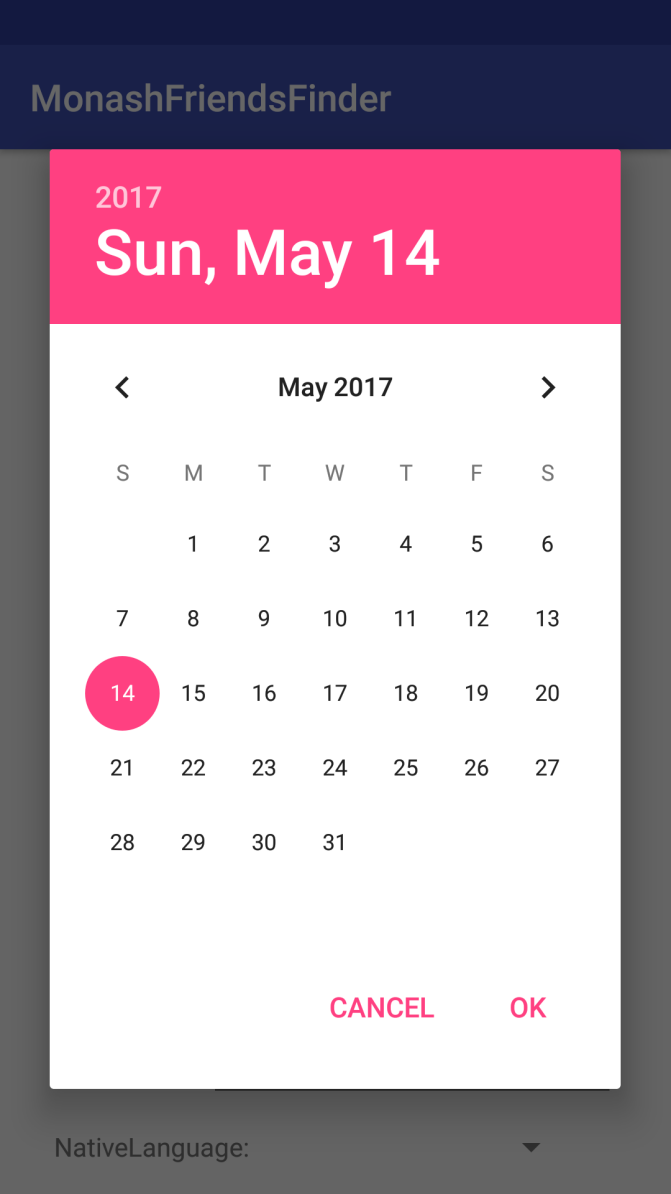
<item>Swimming</item>

<item>Table Tennis</item>

<item>Tennis</item>

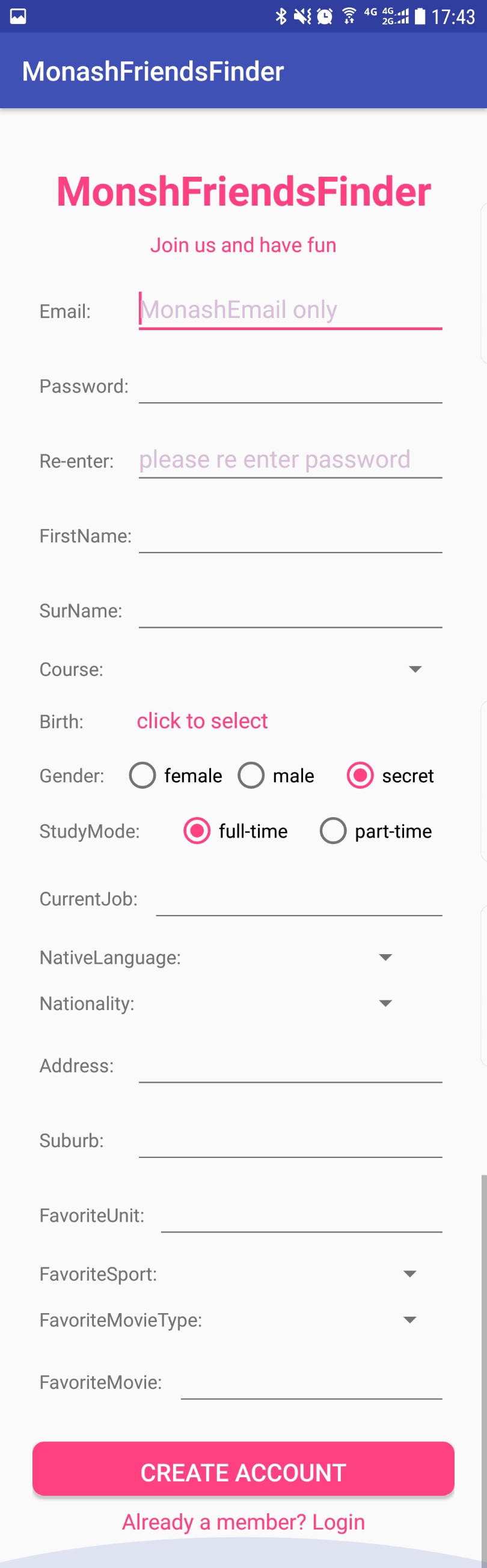
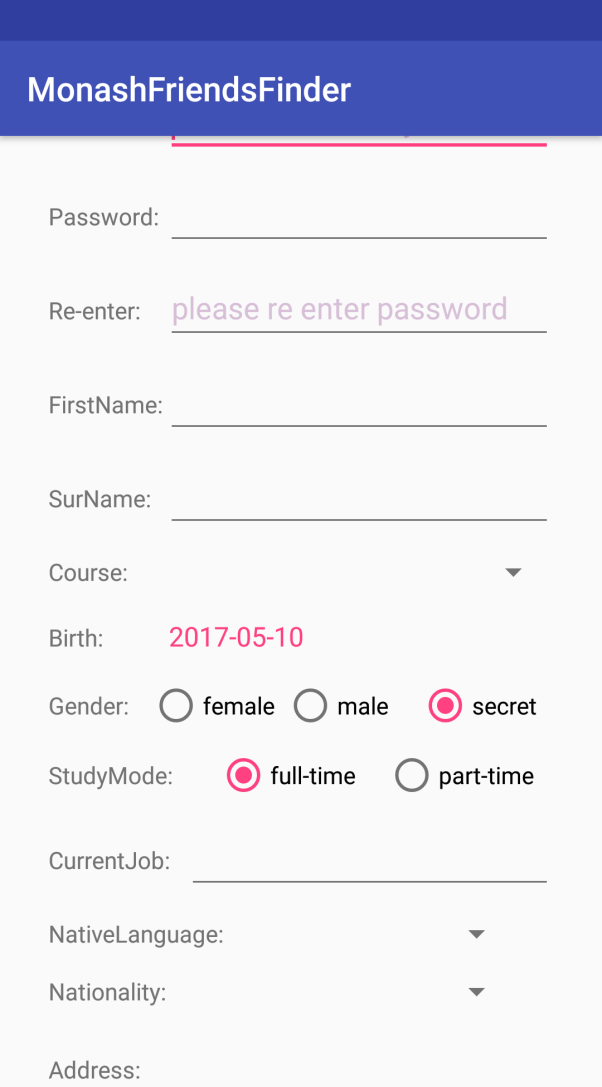
<item>Volleyball</item>

</string-array>



**Pic 11. DatePicker**

**We use the datePicker fragment default of android. It’s a clickable textView showing “click to select”, when you click it, it show this datePicker and then show the result after your choice.(see in picture 12 and picture 13).**

** **

**pic 12. before set date of birth pic 13. after chose the birthday**

**Main code for datePicker:**

// 1. implement interface[DatePickerFragment.OnDateSetListener] declared in fragment

public class RegisterActivity extends AppCompatActivity implements DatePickerFragment.OnDateSetListener {

**// 2. write showDataPickDiaglog in this activity**

**public** **void** showDatePickerDialog(View v) {

DialogFragment newFragment **=** **new** DatePickerFragment();

newFragment.**show**(getFragmentManager(), "datePicker");

}

// 3. write a fragment

**public** class DatePickerFragment extends DialogFragment

**implements** DatePickerDialog.OnDateSetListener {

// 4. Define a callback interface in this fragment

**public** **interface** OnDateSetListener {

// method to be implemented in activity

**void** onDateSet(**int** **year**, **int** month, **int** day, **int** cur);

}

**public** **void** onDateSet(DatePicker view, **int** **year**, **int** month, **int** day) {

// Do something with the date chosen by the user

// 5. call method in interface to send values in this fragment

Log.d("DatePickFragment--cur--", String.**valueOf**(cur));

mListener.onDateSet(**year**, month, day, cur);

}

// 6. onAttach() detect whether the Activity has inherited the interface in this fragment

// if not it will throw an exception

// Called when a fragment is first attached to its context

**public** **void** onAttach(Activity activity) {

**super**.onAttach(activity);

**try** {

mListener **=** (OnDateSetListener) activity;

} **catch** (ClassCastException e) {

**throw** **new** ClassCastException(activity.**toString**() **+** " must implement OnDateSetListener");

}

}

// 7. Implement method in interface in activity

public void onDateSet(int year, int month, int day,int cur) {

String s\_year;

String s\_month;

String s\_day;

if (year < 100) {

s\_year = Integer.**toString**(year);

s\_year = "00" + s\_year;

} else if (year < 1000) {

s\_year = Integer.**toString**(year);

s\_year = "0" + s\_year;

} else {

s\_year = Integer.**toString**(year);

}

if (month < 9) {

s\_month = Integer.**toString**(++month);

s\_month = '0' + s\_month;

} else {

s\_month = Integer.**toString**(++month);

}

if (day < 10) {

s\_day = Integer.**toString**(day);

s\_day = '0' + s\_day;

} else {

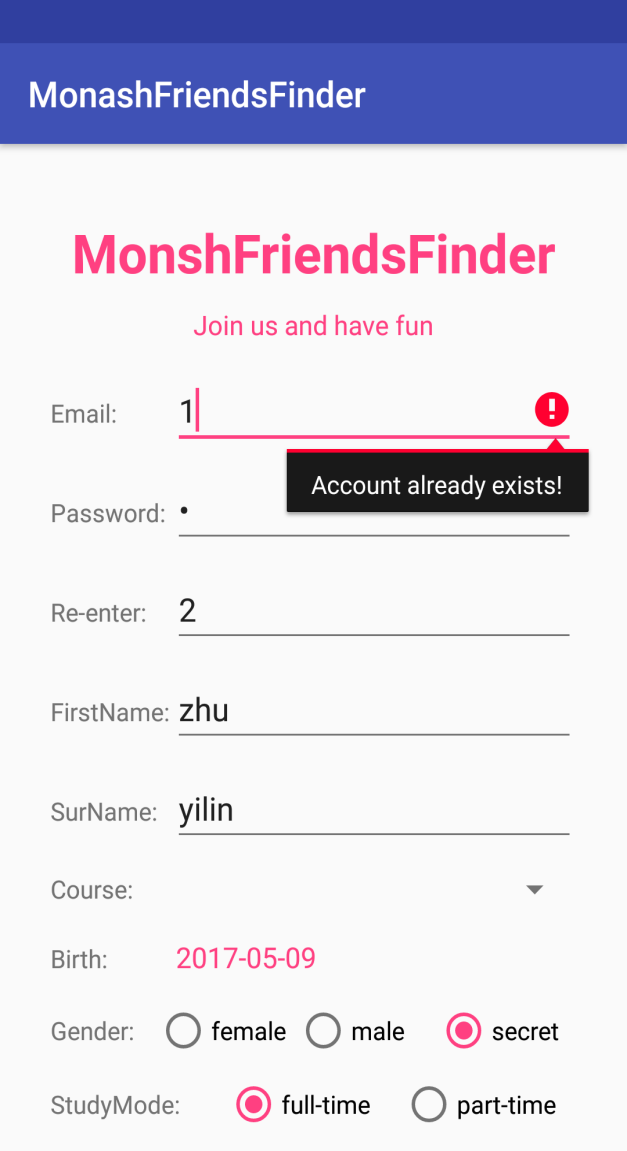
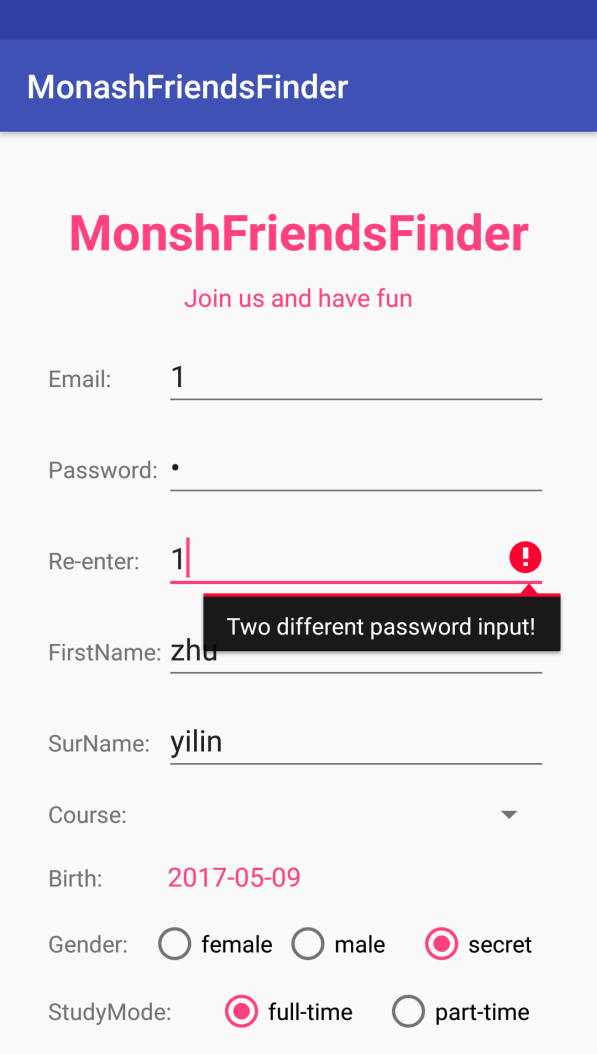
s\_day = Integer.**toString**(day);

}

mtv\_showDoB.**setText**(s\_year + "-" + s\_month + "-" + s\_day);

}

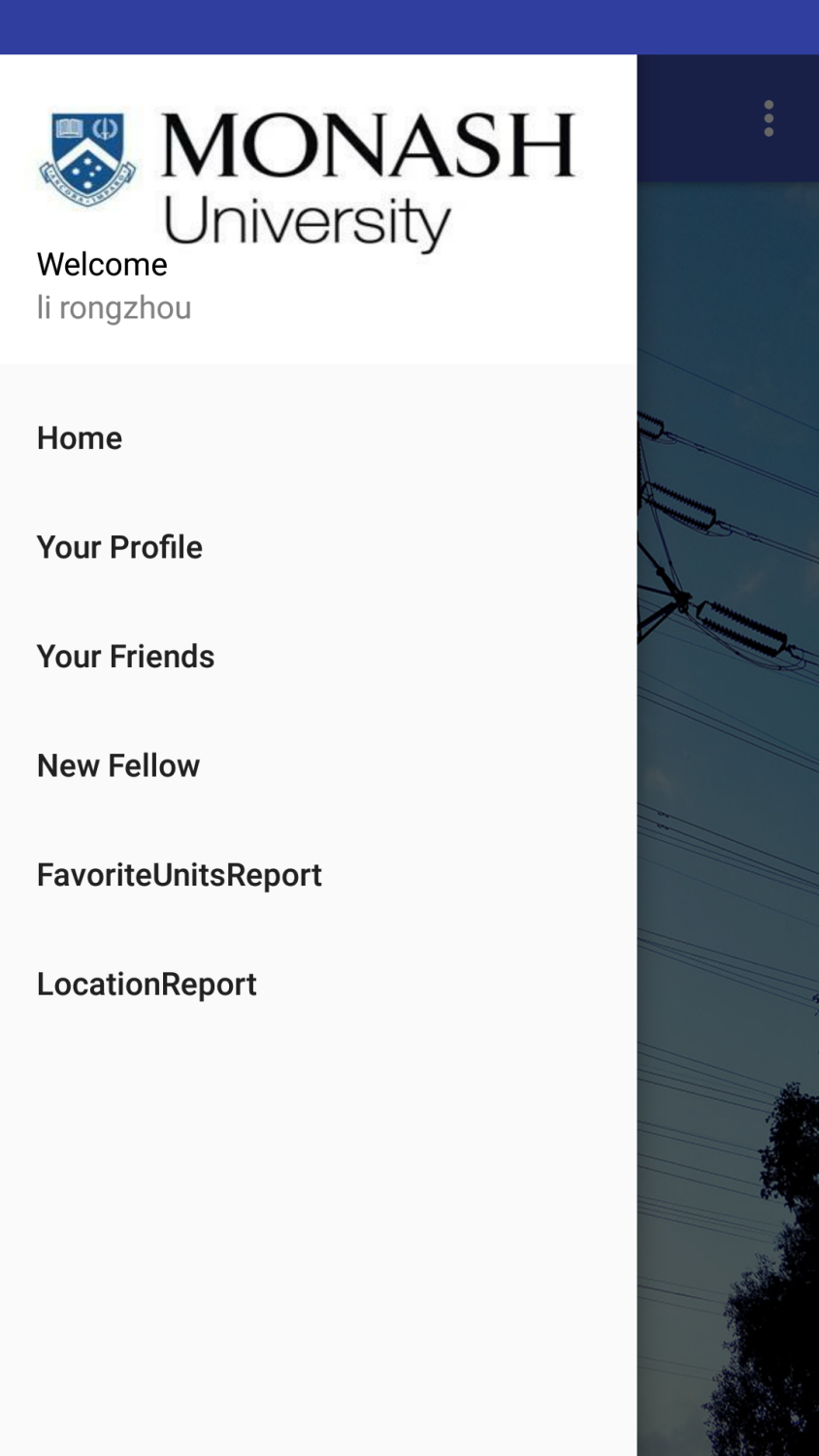
We have many user-friendly designs for this task. In the user registration, if not filled the Email, password and other important parameters, it will mention you that registration are not success. In addition, when the user registers, the registered user name (Email) will be compared with the existing user name in the database. If the comparison result has repeat, the project will mention you ‘Account already exists!’ and a function of checking whether twice inputting of passwords are write(Show as Picture 14&15).

**Pic 14&15 account and password check**

**d)**

homepage already showed in task1, the navigation drawer. welcome and student’s name in the navigation drawer header. The other function of this app are listed below the header(see picture 16).



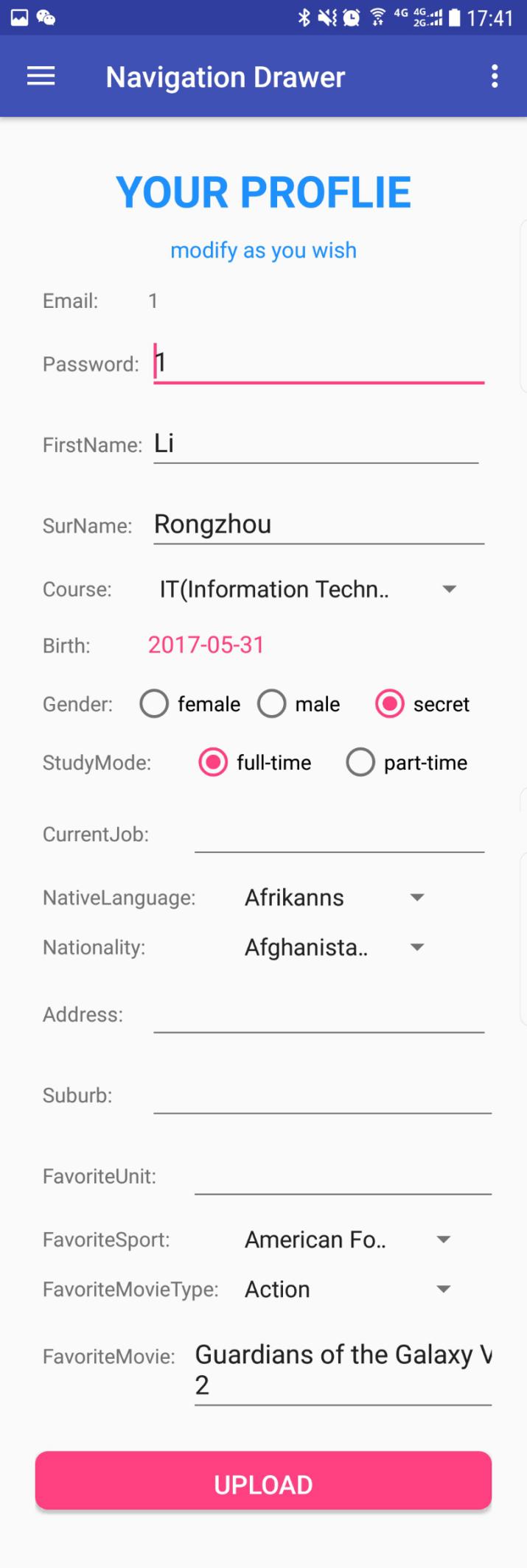
**Picture 16. Navigation drawer**

**HomePage is an Activity and there are four fragments in it (homeFragment, ProfileFragment,FriendFragment andNewFellowFragment). These two reports function are realized in two activity.**

**Task 4 Editing Records and Search Screen (22 marks):**

a) Screen for updating profiles:

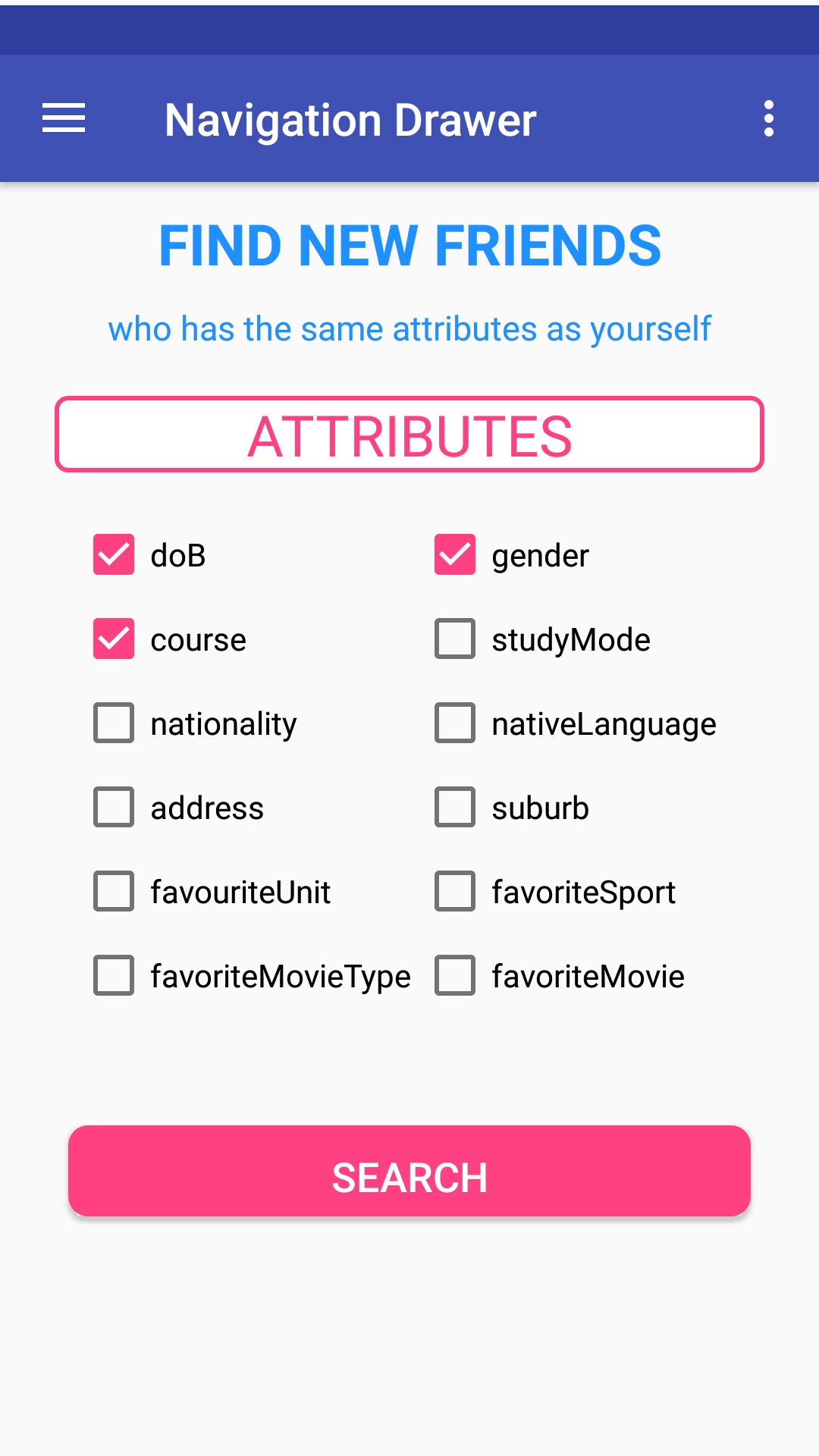
Users can reach this screen for updating profiles though navigation drawer. On this page, users can see all the information before they change. The user can select the personal information he wishes to modify, and finally click the button ‘update’ to store the modified information in the database. (as picture 8 shows) This system update the information from the login user by the method ‘updateProfile(String myId, Student student)’, which connect with the database.



**Pic 17. Screen for updating profiles**

**b) Search screen：**

After click “New Fellow” button in navigation drawer. A search screen will be shown. This search screen allow the student to select multiple matching attributes from a group of checkboxes and then jump to an activity and view a list of other students who meet these attributes with their information. The following image (picture 18 and 19) show the function for selecting features and the list of classmates with the same attributes.



**Pic 18.select attributes**



**Picture 19 Search screen of student with the same information you choose**

This screen find the information from the database by the method searchFriends (monashEmail, finalAttributes).

**Main code for searchFriends method:**

**public** **static** String searchFriends(String myId, String query) {

final String methodPath **=** "/orli.students/findFriendByAnyKey/" **+** myId **+** "/" **+** query;//initialise

URL **url** **=** null;

HttpURLConnection conn **=** null;

String textResult **=** "";//Making HTTP request

**try** {

**url** **=** **new** URL(BASE\_URI **+** methodPath);//open the connection

conn **=** (HttpURLConnection) **url**.openConnection();//set the timeout

conn.setReadTimeout(10000);

conn.setConnectTimeout(15000);//set the connection method to GET

conn.setRequestMethod("GET");//add http headers to set your response type to json

conn.setRequestProperty("Content-Type", "application/json");

conn.setRequestProperty("Accept", "application/json");//Read the response

Scanner inStream **=** **new** Scanner(conn.getInputStream());//read the input steream and store it as string

**while** (inStream.hasNextLine()) {

textResult **+=** inStream.nextLine();

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

conn.**disconnect**();}

Log.d("RestClient", "匹配到的同好是：" **+** textResult);

**return** textResult;

}

Main coed for add friends:

public static int createFriendship(Friendship fsp) {

//initialise

URL url = null;

HttpURLConnection conn = null;

final String methodPath = "/orli.friendship/";

**try** {

Gson gson = **new** Gson();

String stringCourseJson = gson.toJson(fsp);

url = **new** URL(BASE\_URI **+** methodPath);//open the connection

conn = (HttpURLConnection) url.openConnection();//set the timeout

conn.setReadTimeout(10000);

conn.setConnectTimeout(15000);//set the connection method to POST

conn.setRequestMethod("POST");//set the output to true

conn.setDoOutput(true);//set length of the data you want to send

conn.setFixedLengthStreamingMode(stringCourseJson.getBytes().length);//add HTTP headers

conn.setRequestProperty("Content-Type", "application/json");//Send the POST out

PrintWriter **out** = **new** PrintWriter(conn.getOutputStream());

**out**.print(stringCourseJson);

**out**.close();

Log.i("error", **new** Integer(conn.getResponseCode()).toString());

**return** conn.getResponseCode();

} **catch** (Exception e) {

e.printStackTrace();

**return** 400;

} **finally** {

conn.disconnect();

}

}

For making sure that there is no redundant pair in DataBase, a comparison and change should be added before store this friendship.In our cwe make the small email first.

**if** (myMonashEmail**.**compareTo(Friendemail) **>** 0) {

String tmp **=** "";

tmp **=** Friendemail;

Friendemail **=** myMonashEmail;

myMonashEmail **=** tmp;

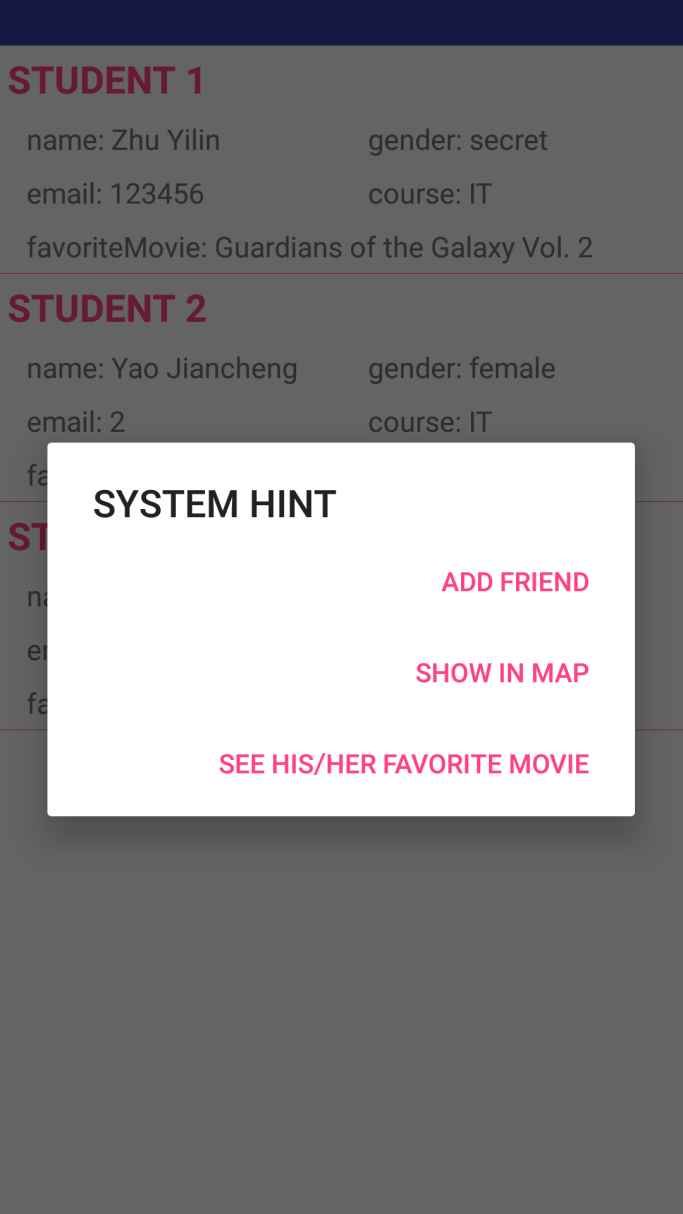
}

**c)**

When you click at each student, several options to be selected will appear. If users choose to click see his/her favourite movie, the information and image about the movie will appear. (shown in before).

**d)**

When you click “add friend”, if you are not friends you will see a toast”added successful” else it will toast”you are friends already”.



**Picture 20. Search screen after you click each student you interest**

**e)**

The way you get the movie information is similar to the way to get the weather information. We need to use API to obtain the official picture/image of the movie as a supplement to the film introduction. If you click the button “add friend”, this student will be add as your friend. (with help of the method createFriendship(Friendship fsp))

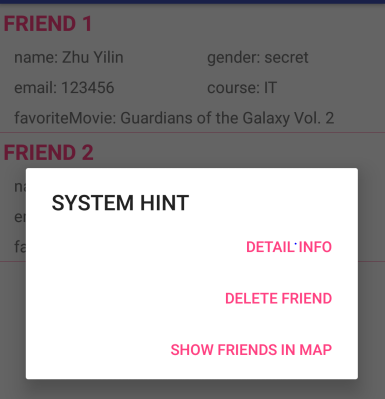
3. Friends screen

The friends screen allow the users to view the list of his/her current friends and delete any of them. (show as picture 21)

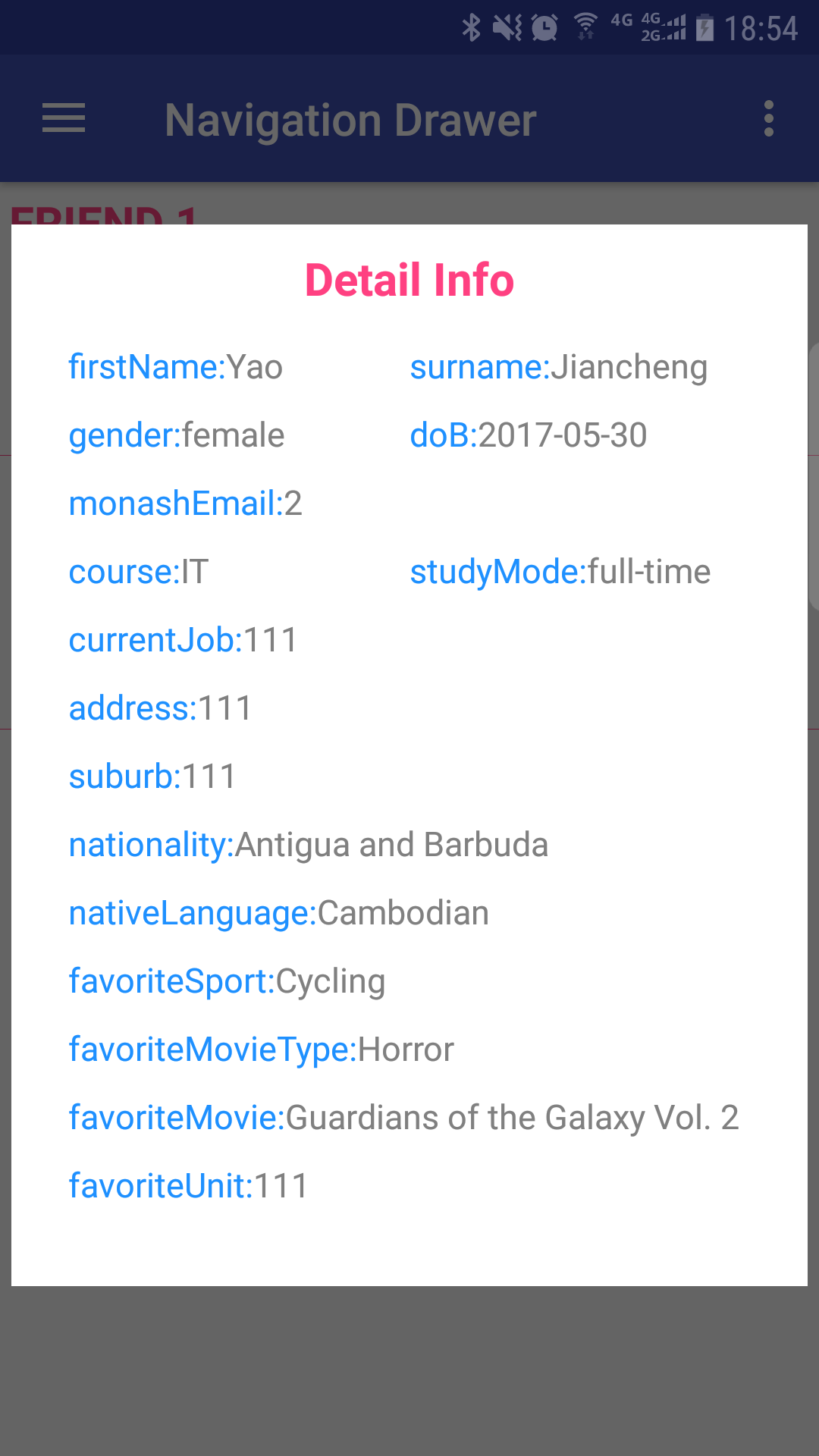
. 

**Picture 21. friends list**

When you click at each of your friend, several options to be selected will appear. (picture 22) If users choose to click delete friend, this friend will be deleted from you friendship. And detail of this friend will be shown when you click “Detail info”(see picture 23).



**Picture 22. friends’ information after you click at each of them**

****

**Picture 23. friends’ detail information**

This system get the information of your friends from the database by the method getFriendship1(String myId) , and delete his/her friends by the method deleteFriendship(String myId,String FriendId).

**Main code for delete friends:**

**public** **static** **void** deleteFriendship(String myId, String FriendId) {

final String methodPath **=** "/orli.friendship/somePath;myMonashEmail=" **+** myId **+** ";friendMonashEmail=" **+** FriendId;

URL **url** **=** null;

HttpURLConnection conn **=** null;

String txtResult **=** "";// Making HTTP request

**try** {

**url** **=** **new** URL(BASE\_URI **+** methodPath);//open the connection

conn **=** (HttpURLConnection) **url**.openConnection();//set the connection method to GET

conn.setRequestMethod("DELETE");

Log.i("delete\_log", **new** Integer(conn.getResponseCode()).**toString**());

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

conn.**disconnect**();

}

}

**Task 5 Local Storage (8 marks):**

**a)**

We should to store part of the data locally on the mobile phone’s SQLite database. We decide on store the data of lists of nationalities and nativelanguages that could be used to populate the spinners in the subscription and profile screen. The data of lists of countries or languages is stored under assert file called nationData.txt and nativeLanguageData.txt. When we create dataBase, them will be read and store in it.

**Main code for method of reading info in txt file and store in database:**

// 1. read nation txt from assets and insert db

private void readNativeLanguageFromAssets(SQLiteDatabase db) {

**try** {

ContentValues values **=** **new** ContentValues();

InputStream is **=** context**.**getAssets()**.**open("nativeLanguageData.txt");

InputStreamReader reader **=** **new** InputStreamReader(is);

BufferedReader br **=** **new** BufferedReader(reader);

String s1;

// read line by line

**while** ((s1 **=** br**.**readLine()) **!=** null) {

//add key-value pairs in values

values**.**put(DBStructure**.**tableNativeLanguage**.**COLUMN\_NAME, s1);

//insert record

db**.**insert(DBStructure**.**tableNativeLanguage**.**TABLE\_NAME, null, values);

}

br**.**close();

reader**.**close();

} **catch** (Exception e) {

// TODO Auto-generated catch block

e**.**printStackTrace();

}

}

//2. invoke this method when initialize SQLiteDataBase

@Override

public void onCreate(SQLiteDatabase db){

db**.**execSQL(SQL\_CREATE\_TABLE\_NATIONALITY);

db**.**execSQL(SQL\_CREATE\_TABLE\_NATIVELANGUAGE);

// call this method here

// onCreate will only be called once(only when there didn't exist a user.db)

readNationFromAssets(db);

readNativeLanguageFromAssets(db);

}

**b)**

In addition to that, we also need to store some data using SharedPreferences which is suitable for small set of key-value pairs. We use sharedPreferences to store the user's basic information key pairs when login.

**Main code for store sharedPreference:**

public class MainActivity extends AppCompatActivity {

//1. 使用SharedPreferences进行读取

private SharedPreferences pref;

//2. 使用SharedPreferences.Editor进行存储

private SharedPreferences**.**Editor editor;

@Override

protected void onCreate(Bundle **savedInstanceState**) {

//3. 第一个参数：文件名，没有则新建。第二个参数：写入模式-覆盖

pref **=** getSharedPreferences("admin", MODE\_PRIVATE);

//4. 获取SharedPreferences.Editor对象

editor **=** pref**.**edit();

// 5. save key-value sharedPreference

editor**.**putString("favoriteMovieType", favoriteMovieType);

editor**.**putString("favoriteMovie", favoriteMovie);

editor**.**commit();

**Task 6 Maps (14 marks):**

**a)**

Firstly, the search screen will allow users to navigate to a map screen where the matching students will be shown on a map. The map will show the current location of the student and all the matching students using two different marker colours to differentiate between them. When users click the button, the student’s detail info will be show in a textview floating behind the map marker.

We use Baidu API to realize all the map function. As you can see in picture 24, the user is marked by a blue point, while the other matched students are marked with red map markers. Because we use one MapActivity to show friends and matched students, so picture 25(show user’s friends) looks the same as 24. This part spent most of our time. For reaching the final effect, Baidu API Document has been read by us. At the same time ,we sincerely thank other students helping us in solving this problem.



**Picture 24. show matched students in map**

**b)**

****

**Picture 25. show friends in map**

**Main code in MapActivity.**

// we jump to one same MapActivity from SearchResultListFragment and

// FriendsFragment.(Fragment to Activity)

// thus we need to do a complex way for realizing this request

// what we need to do:

//1. realize a new Intent method in activity

public static Intent newIntent(Context packageContext, int flag) {

Intent intent **=** **new** Intent(packageContext, MapActivity**.**class);

intent**.**putExtra("flag", flag);

**return** intent;

}

// 2. passing different flag for differ their request

// 2.1 write intent in SearchResultListFragment

**case** AlertDialog**.**BUTTON\_NEGATIVE**:**// show in map

Integer flag **=** 1;

Intent intent1 **=** MapActivity**.**newIntent(getActivity(),flag);

//start intent

startActivity(intent1);

**break**;

// 2.2 write intent in FriendsFragment

**case** AlertDialog**.**BUTTON\_NEUTRAL**:**// show in map

Integer flag **=** 2;

Intent intent1 **=** MapActivity**.**newIntent(getActivity(),flag);

//start intent

startActivity(intent1);

**break**;

//3. get the flag

int flag **=** **this.**getIntent()**.**getIntExtra("flag", 0);

Log**.**d("ShowMovieActivity", "get flag: " **+** flag);

// 4. invoke different method being relevant with flag

**if** (flag **==** 1) {

**try** {

initMarkerList1();

} **catch** (JSONException mE) {

mE**.**printStackTrace();

}

} **else** **if** (flag **==** 2) {

**try** {

initMarkerList2();

} **catch** (JSONException mE) {

mE**.**printStackTrace();

}

}

**Main code in RestClient for getting and updating a student’s latest location info(latitude and longitude).**

**public** **static** String getLatestLocation(String email)

{

final String methodPath **=** "/orli.location/returnlatestLocation/"**+**email;//initialise

URL **url** **=** null;

HttpURLConnection conn **=** null;

String textResult **=** "";//Making HTTP request

**try** {

**url** **=** **new** URL(BASE\_URI **+** methodPath);//open the connection

conn **=** (HttpURLConnection) **url**.openConnection();//set the timeout

conn.setReadTimeout(10000);

conn.setConnectTimeout(15000);//set the connection method to GET

conn.setRequestMethod("GET");//add http headers to set your response type to json

conn.setRequestProperty("Content-Type", "application/json");

conn.setRequestProperty("Accept", "application/json");//Read the response

Scanner inStream **=** **new** Scanner(conn.getInputStream());//read the input steream and store it as string

**while** (inStream.hasNextLine()) {

textResult **+=** inStream.nextLine();

}

Log.d("RestClient", "获取到的location数据是：" **+** textResult);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

conn.**disconnect**();

}

**return** textResult;

}

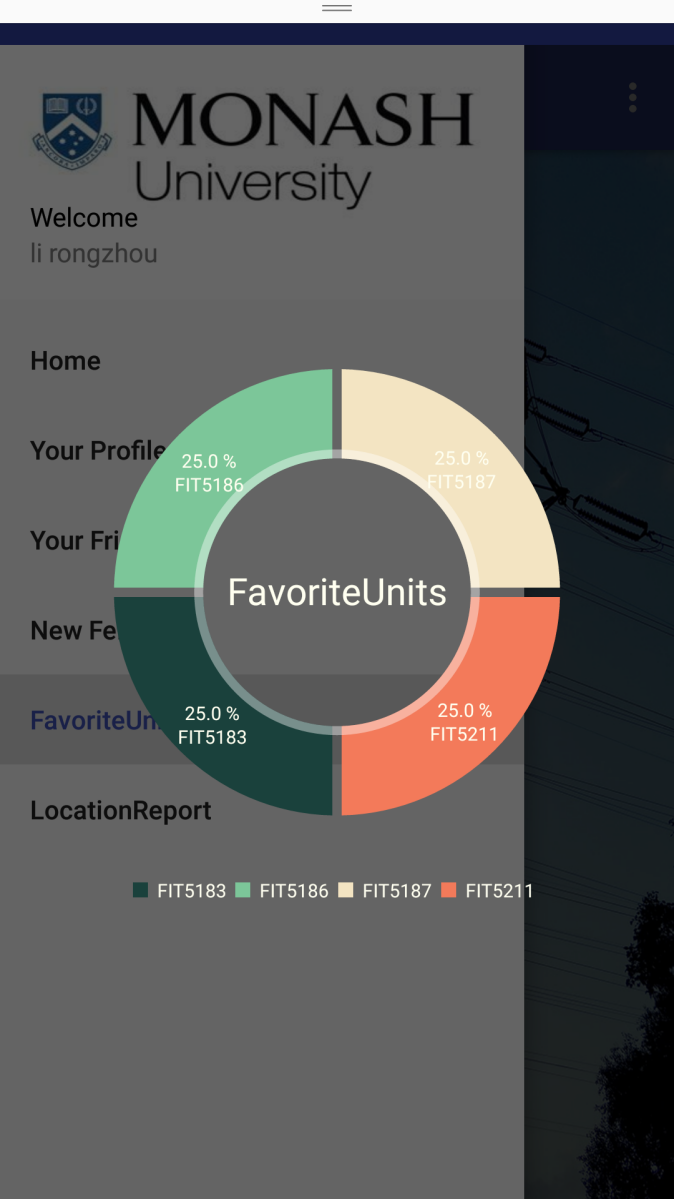
**Task 7 Reports (14 marks):**

1. **Common attributes pie graph screen:**

When you click “FavoriteUnits Report” , a float Pie Chart(see in picture 26) will shown in the center of your phone screen. This chart include all Units’ name and their percentage in all units being favorited. You can click any of them to see detail and even you can play it like drag and rotate it.

We realize this effect by show this piechart in a dialogStyle activity mentioned before. Surprising us, it has this magic effect. Due to limited time, we will study it later.

Codes for drawing a pie chart came from a open resource called [MPAndroidChart](https://github.com/PhilJay/MPAndroidChart) in github. We download and modified them according to our needs. Here is the github link: <https://github.com/PhilJay/MPAndroidChart>. And with the help of serveral blogs in CSDN, like <http://blog.csdn.net/shineflowers/article/details/44701645>.



**Picture 26. Pie Chart for reporting favoriteUnits**

**Main code for countFavoriteUnits**

public static String countFavoriteUnits() {

final String methodPath **=** "/orli.students/countFavoriteUnits";//initialise

URL url **=** null;

HttpURLConnection conn **=** null;

String textResult **=** "";//Making HTTP request

**try** {

url **=** **new** URL(BASE\_URI **+** methodPath);//open the connection

conn **=** (HttpURLConnection) url**.**openConnection();//set the timeout

conn**.**setReadTimeout(10000);

conn**.**setConnectTimeout(15000);//set the connection method to GET

conn**.**setRequestMethod("GET");//add http headers to set your response type to json

conn**.**setRequestProperty("Content-Type", "application/json");

conn**.**setRequestProperty("Accept", "application/json");//Read the response

Scanner inStream **=** **new** Scanner(conn**.**getInputStream());//read the input steream and store it as string

**while** (inStream**.**hasNextLine()) {

textResult **+=** inStream**.**nextLine();

}

} **catch** (Exception e) {

e**.**printStackTrace();

} **finally** {

conn**.**disconnect();

}

Log**.**d("RestClient", "获取到的favoriteUnit数据是：" **+** textResult);

**return** textResult;

}

Write a favorite java class. (wrting this class will make json parsing be easier by using Gson).

public class FavoriteUnitsFrequency {

private String favoriteUnit;

private String frequency;

public FavoriteUnitsFrequency(String **mFavoriteUnit**, String **mFrequency**) {

favoriteUnit **=** mFavoriteUnit;

frequency **=** mFrequency;

}

public String getFavoriteUnit() {

**return** favoriteUnit;

}

public void setFavoriteUnit(String **mFavoriteUnit**) {

favoriteUnit **=** mFavoriteUnit;

}

public String getFrequency() {

**return** frequency;

}

public void setFrequency(String **mFrequency**) {

frequency **=** mFrequency;

}

}

Main code for parsing response json.

**new** AsyncTask<Void, Void, String>() {

@Override

protected String doInBackground(Void... **params**) {

String result **=** RestClient**.**countFavoriteUnits();

**return** result;

}

@Override

protected void onPostExecute(String **info**) {

**try** {

JSONArray jsonArray **=** **new** JSONArray(info);

**for** (int i **=** 0; i **<** jsonArray**.**length(); **++**i) {

JSONObject jsonObject **=** jsonArray**.**getJSONObject(i);

frq**.**add(**new** Gson()**.**fromJson(jsonObject**.**toString(), FavoriteUnitsFrequency**.**class));

}

sum\_count **=** 0;

int i;

**for** (i **=** 0; i **<** frq**.**size(); **++**i) {

unitFrequencyMapList**.**put(frq**.**get(i)**.**getFavoriteUnit(),

Integer**.**parseInt(frq**.**get(i)**.**getFrequency()));

sum\_count **+=** Integer**.**parseInt(frq**.**get(i)**.**getFrequency());

}

unit\_count **=** i;

Log**.**d("unit\_count: " **+** unit\_count, " sum\_count: " **+** sum\_count);

PieData mPieData **=** getPieData(unit\_count, sum\_count, unitFrequencyMapList);

showChart(mChart, mPieData);

} **catch** (JSONException mE) {

mE**.**printStackTrace();

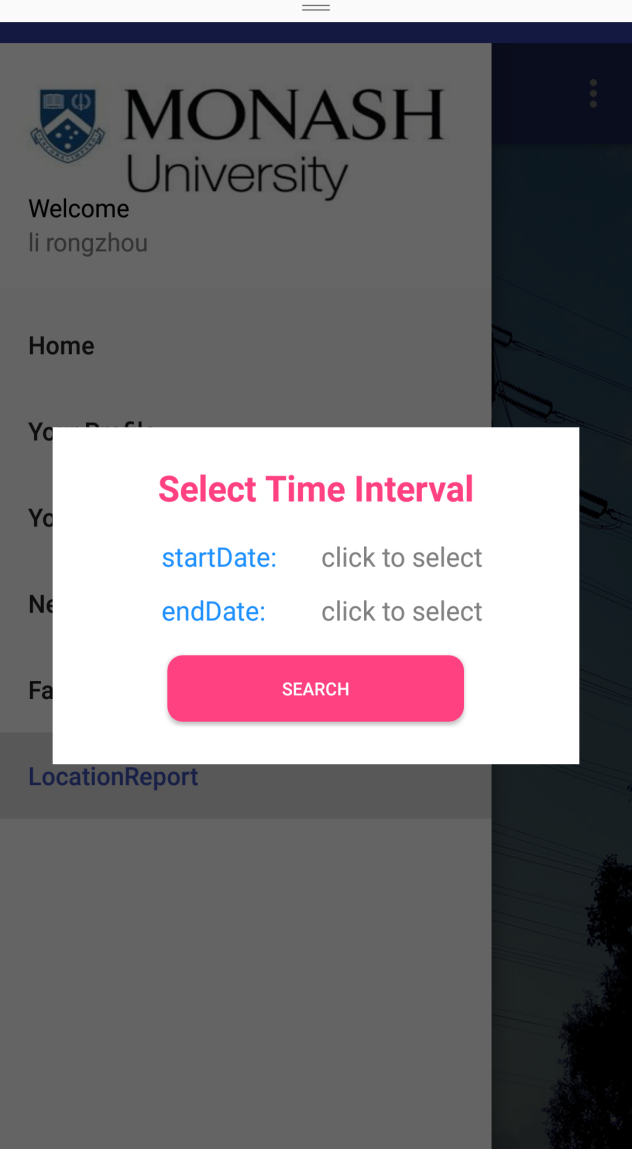
}

}

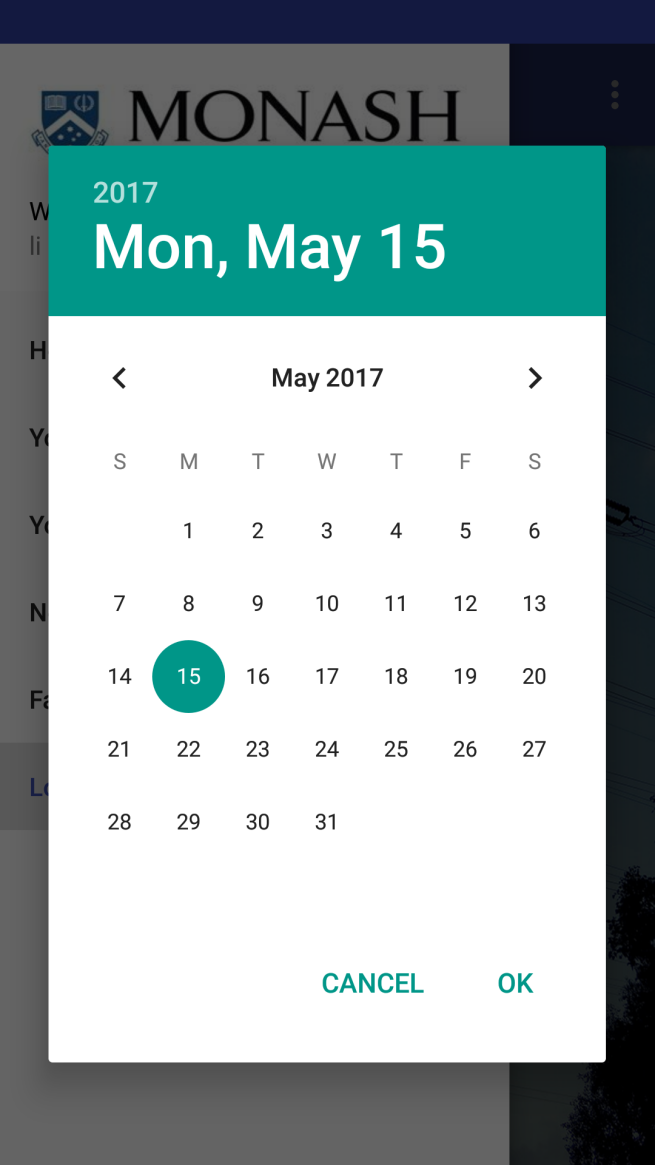
}**.**execute();

**b)Location bar chart screen:**

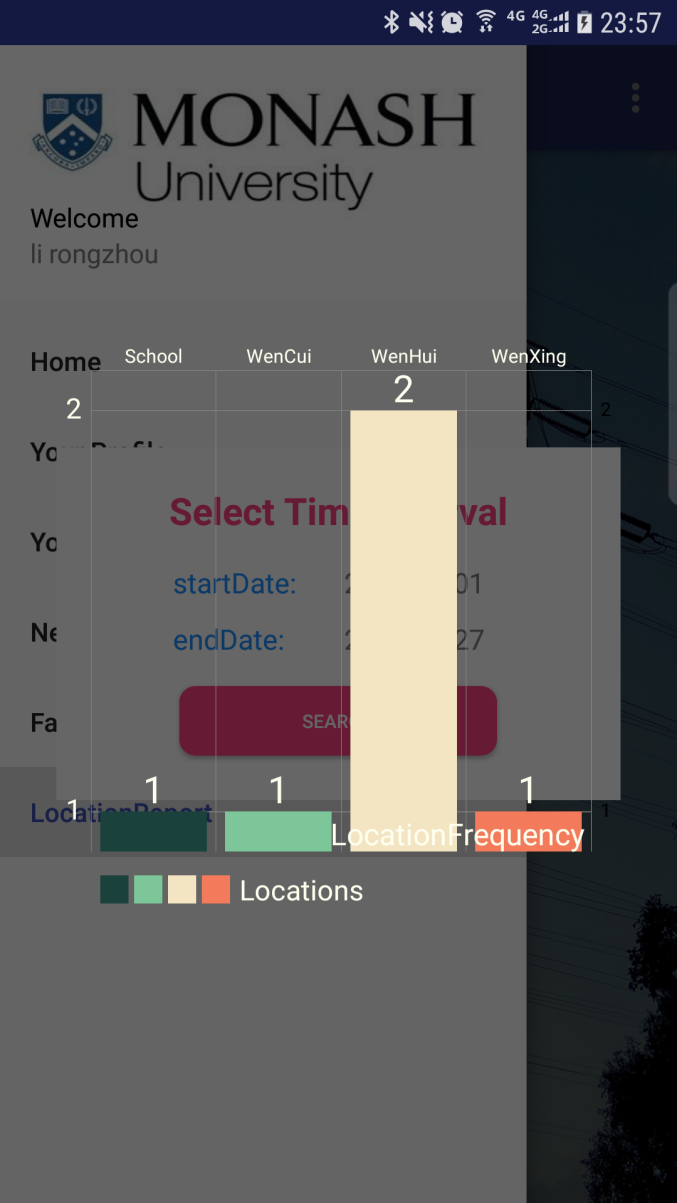
Codes for this also came form the open resource where the pie chart came from. We also use them and made personal design(see in picture 27). The four locations, WenHui, WenXing, WenCui, School and users’ sum times went to them in a time period are shown. This part is realized in this way: 1. User click LocationReport in navigationDrawer; 2. A dialogStyle activity will jump out; 3. Pick startDate and endDate(realized use DatePicker)(see in picture 27 and 28); 4. A barChart shown after “search” button clicked.



**Picture 26. activity for set date period**



**Picture 26. An another style dataPicker**



**Picture 27. Pie Chart for reporting favoriteUnits**

**Main code of lacationFrquency class**

public class LocationFrequency {

private String location;

private String frequency;

public LocationFrequency(String **mLocation**, String **mFrequency**) {

location **=** mLocation;

frequency **=** mFrequency;

}

public String getLocation() {

**return** location;

}

public void setLocation(String **mLocation**) {

location **=** mLocation;

}

public String getFrequency() {

**return** frequency;

}

public void setFrequency(String **mFrequency**) {

frequency **=** mFrequency;

}

}

**Main code in Activity for parsing json result.**

**new** AsyncTask<Void, Void, String>() {

@Override

protected String doInBackground(Void... **params**) {

String info;

**if** (**!**(info **=** RestClient**.**getLocationFrequency(monashEmail, startDate, endDate))**.**equals("[]")) {

**return** info;

} **else** {

**return** "1";

}

}

@Override

protected void onPostExecute(String **info**) {

**if** (info**.**equals("1")) {

finish();

}

**try** {

JSONArray jsonArray **=** **new** JSONArray(info);

**for** (int i **=** 0; i **<** jsonArray**.**length(); **++**i) {

JSONObject jsonObject **=** jsonArray**.**getJSONObject(i);

placeFrequencyList**.**add(**new** Gson()**.**fromJson(jsonObject**.**toString(),

LocationFrequency**.**class));

}

int i;

**for** (i **=** 0; i **<** placeFrequencyList**.**size(); **++**i) {

placeFrequencyMapList**.**put(placeFrequencyList**.**get(i)**.**getLocation(),

Integer**.**parseInt(placeFrequencyList**.**get(i)**.**getFrequency()));

}// count = i;

i **=** 0;

**for** (Map**.**Entry<String, Integer> m **:** placeFrequencyMapList**.**entrySet()) {

labels**.**add(m**.**getKey());

entries**.**add(**new** BarEntry(m**.**getValue(), i));

**++**i;

}

showChart(barChart, entries);

} **catch** (Exception e) {

e**.**printStackTrace();

}

}

}**.**execute();

**Main code in RestClinet**

public static String getLocationFrequency(String monashEmail, String startDate, String endDate) {

final String methodPath **=** "/orli.location/recordTrack/"**+**monashEmail**+**"/"**+**startDate**+**"/"**+**endDate;//initialise

URL url **=** null;

HttpURLConnection conn **=** null;

String textResult **=** "";//Making HTTP request

**try** {

url **=** **new** URL(BASE\_URI **+** methodPath);//open the connection

conn **=** (HttpURLConnection) url**.**openConnection();//set the timeout

conn**.**setReadTimeout(10000);

conn**.**setConnectTimeout(15000);//set the connection method to GET

conn**.**setRequestMethod("GET");//add http headers to set your response type to json

conn**.**setRequestProperty("Content-Type", "application/json");

conn**.**setRequestProperty("Accept", "application/json");//Read the response

Scanner inStream **=** **new** Scanner(conn**.**getInputStream());//read the input steream and store it as string

**while** (inStream**.**hasNextLine()) {

textResult **+=** inStream**.**nextLine();

}

} **catch** (Exception e) {

e**.**printStackTrace();

} **finally** {

conn**.**disconnect();

}

Log**.**d("RestClient", "获取到的frequency数据是：" **+** textResult);

**return** textResult;

}

**Ⅲ.Problems summary**

Here is the list of part problems and their resolution we met in constructing this app:

**1.Gradle sync failed.**

Cause: paste GSON library in wrong gradle file, not in gardle file [build.gardle (module: app)].

**2. Crash once starting this app.**

Probability cause:

1)Set wrong setContentView(R.layout.xxx);

2)No initializing variables before use;

Resolution: read logcat and find the last related error record(about your own activity).

**3.fragment communicate with activity.**

Read developer document and a blog : <http://blog.csdn.net/huangyabin001/article/details/35256379>

Resolution: use interface.

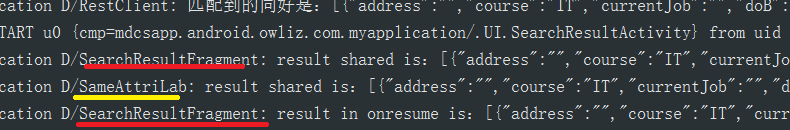
method need connecting with internet should write in AsyncTask.

**4. fragment1->activity1->fragment1->activity1. activity1 doesn’t refresh with neww data from fragment1 the second time.**

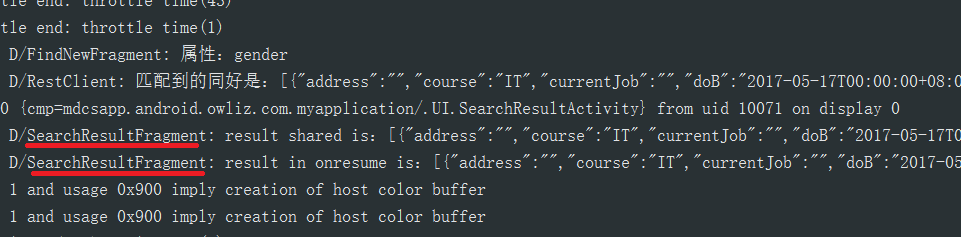
Rsolution：

First I use log to trace info change.I notice this change.

1)First time generate result list view

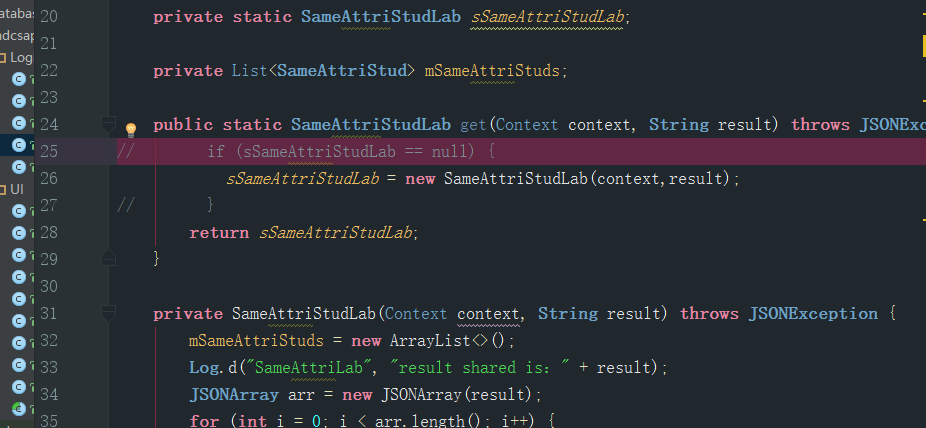


2)Second time generate result list view



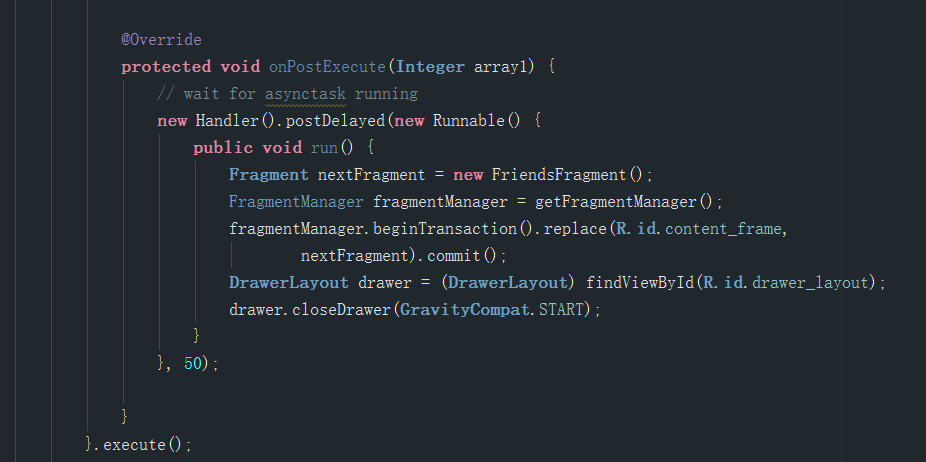
SameAttriLab hasn’t been generated after first time.

i find I use java Singleton to storage matched students’ info for displaying. And in a typical Singleton, it has this judge to make sure the data has only one version. So I make it change everytime I go into this result activity.



Resolution：handler

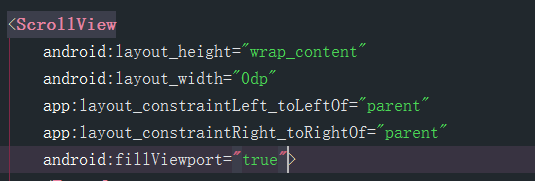
write jump fragment method in a handler().postDelayed().



**10. Set android:layout\_width="match\_parent" of ScrollView in ConstraintLayout, it will auto change to android:layout\_height="368dp", which nakes the fragment can't fill this ScrollView.**

Resolution: find answer in stackoverflow.

This setting of setting android:layout\_width="match\_parent" isn't supported by ConstraintLayout. It should write in this way:



**Ⅳ. Conclusion**

After this task, we gained a lot. The importance of base: Because Android language was rarely used before, there were many difficulties in the early stages of programming. Thanks to detailed tutorials on the Moodle and enthusiastic coders in the stack overflow ,CSDN and other place sharing their knowledge and helping us in the Internet. Due to the limited time, our app still has some problems to be further improved.

What we learn:

1. Ability to learn and communicate: Our IT worker's most important ability in life is probably the ability to learn and communicate. How to integrate the innumerable network resources to use, this is my weak place. We have made a lot of progress through this project.

2. Spirit of persistence: When writing programs, you often encounter difficult problems to solve. Only by persevering and solving problems creatively can we finally get growth from difficulties.