ChihYung Wu

4.26.2016

Assignment 6

The deliverables are:

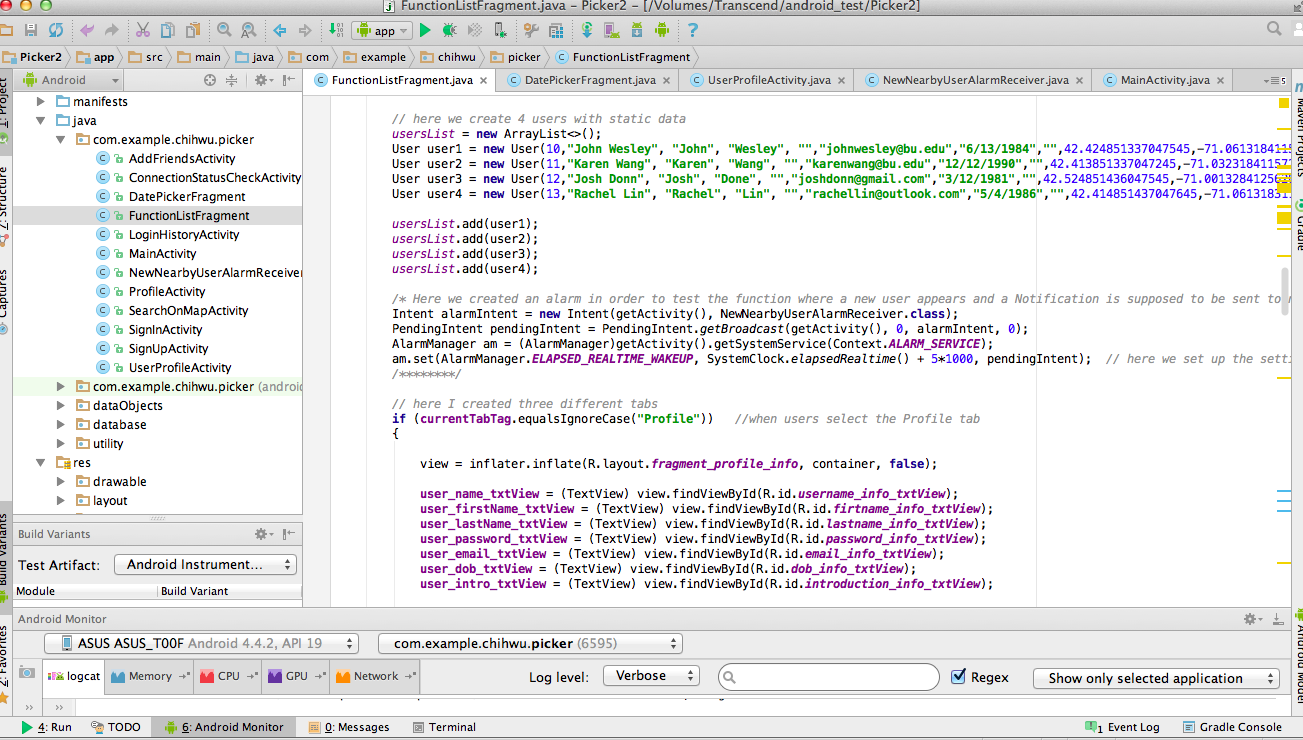
# Part 1—Overall Project Description:

Provide a paragraph (no more) outlining the Android application you’d like to build for this course. Don’t worry about being overly ambitious or not completely specific to begin with: we hold you only to the specific requirements in Part 3.

# This will be a cool-friend-making application that will allow its users to quickly find and make friends with someone in their proximity. In order to be able to use this application, the user will have to create an account by inputting their username, password, date-of-birth, introduction. The results will display the info of all the users in that user’s proximity.

# Currently, users can use Google Map Android API to locate his/her location on the map. The app now can show the distance of other users geographically close to the current user of the app and more functions, such as checking out other users profile and receiving notifications when new users log in.

# Part 2—Screenshot of the directory structure (project or package):

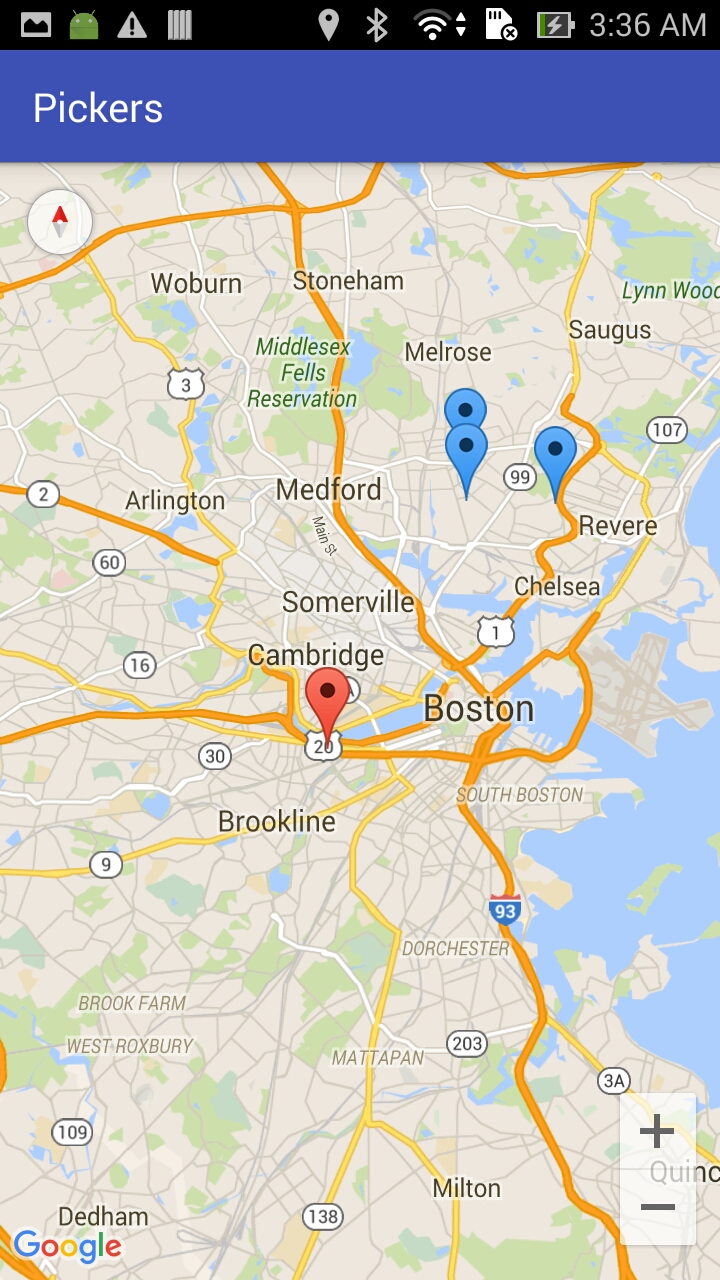


# Part 3—Application Features with output and tests: *(maximum 2 pages of 12-point text, including figures)*

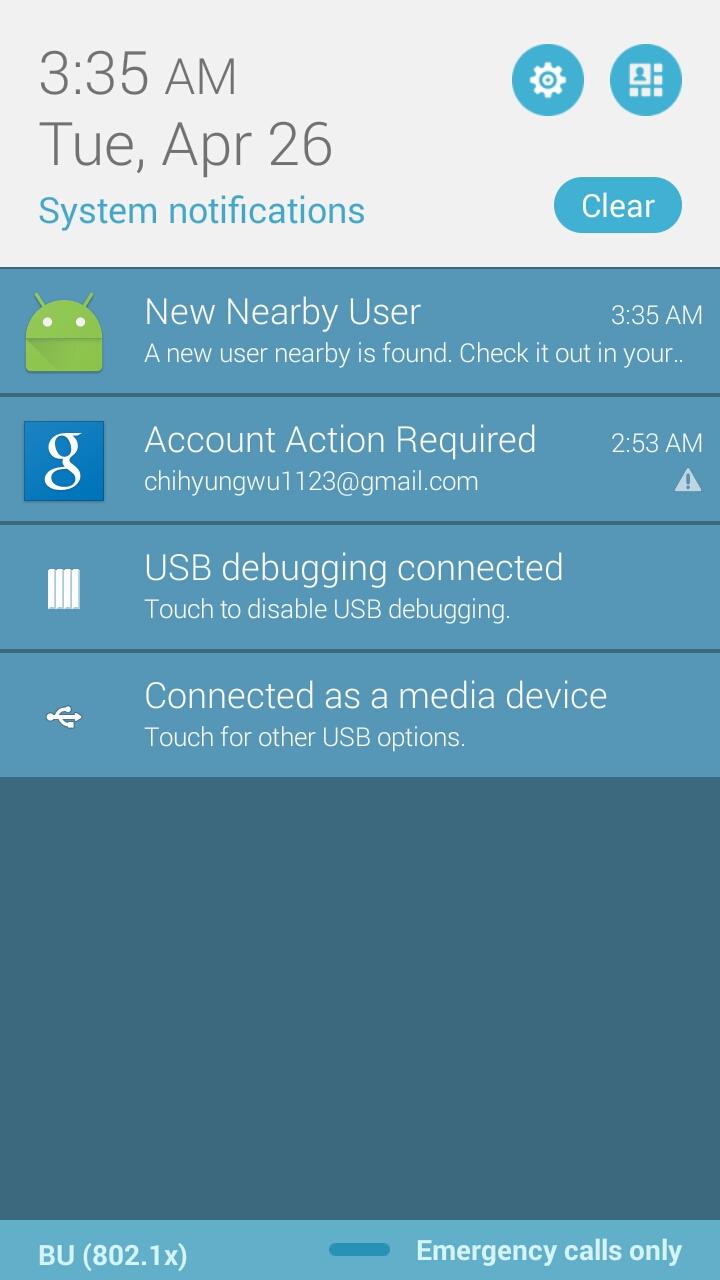
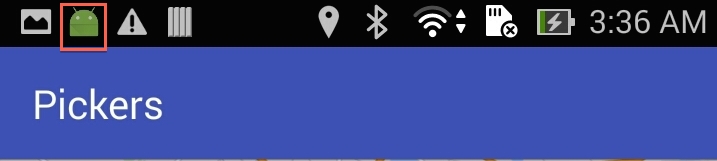
Specify the features of your application that you implemented. The priority is to get practice with as many as possible of the Android constructs covered in Module 1. Number your features 1, 2, 3, … . Each feature must accompanied by output or screen shot that show you accomplished it.

(New Features based on elements covered in this week’s module)

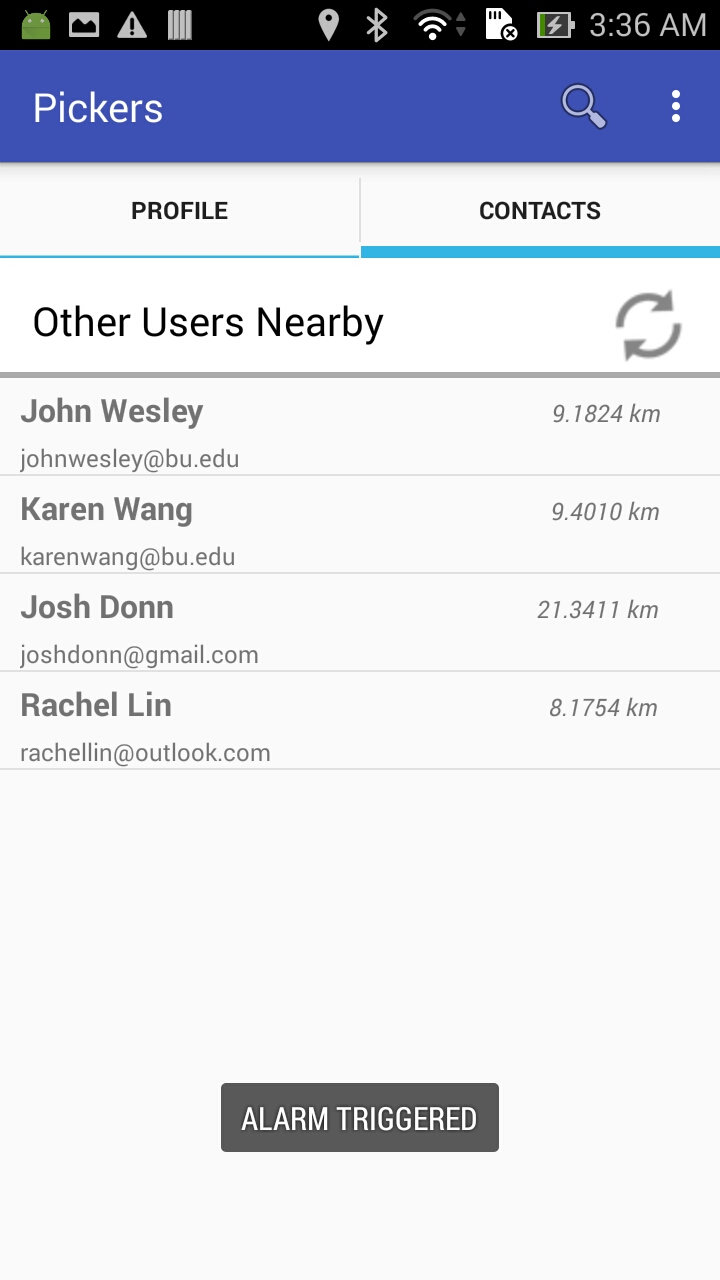
Application Feature 1: (The user of the app and other users nearby are displayed now on the map with markers of different colors.)



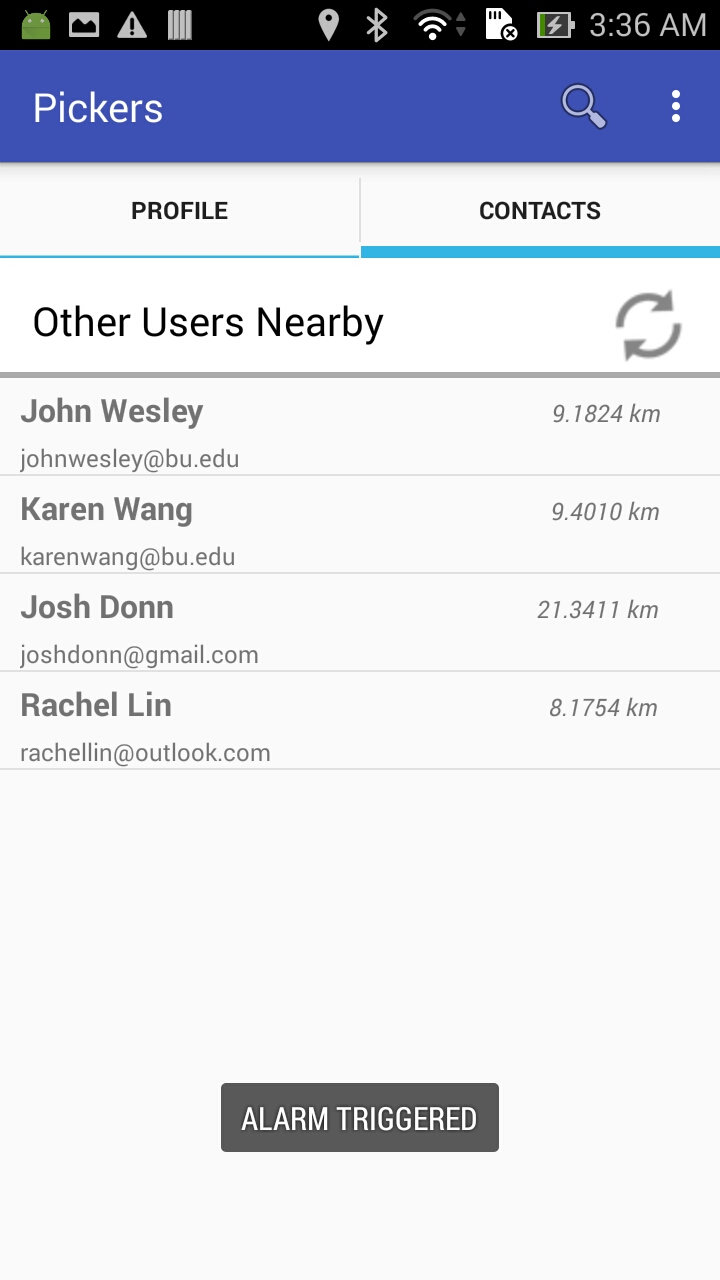
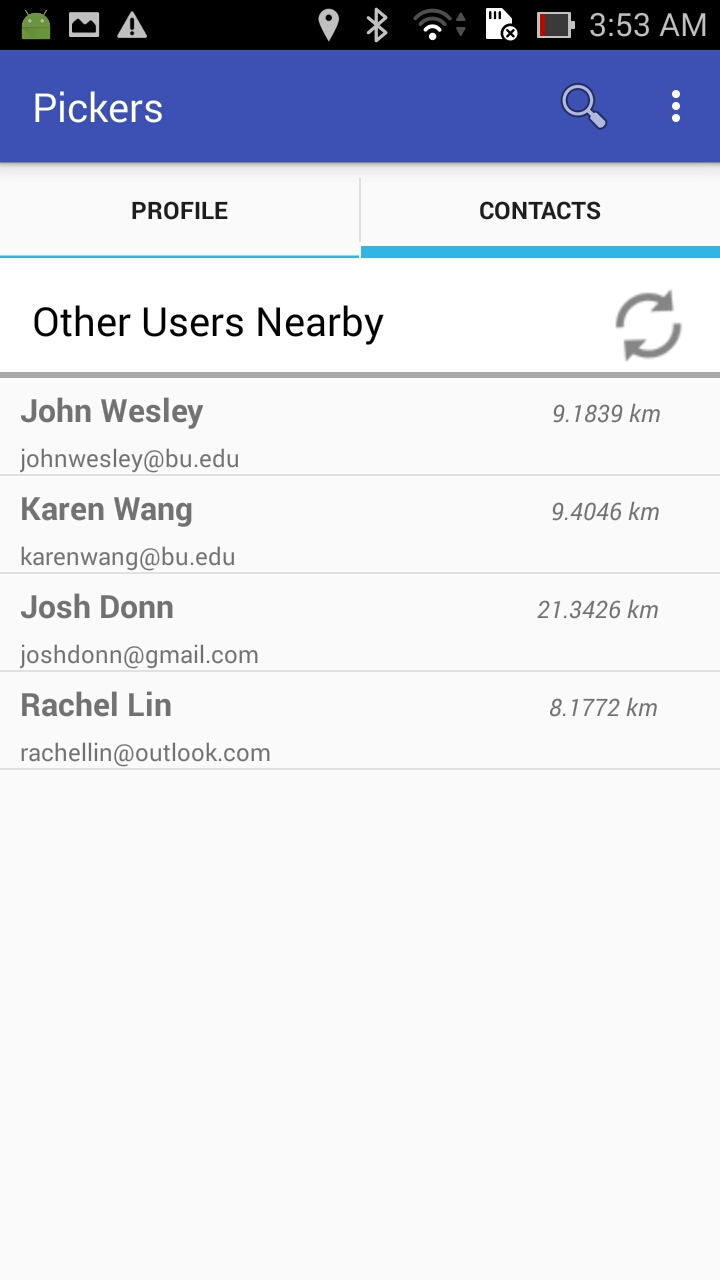
Application Feature 2: (When there is an user nearby, the app will send a notification, which will vibrate user’s phone.)

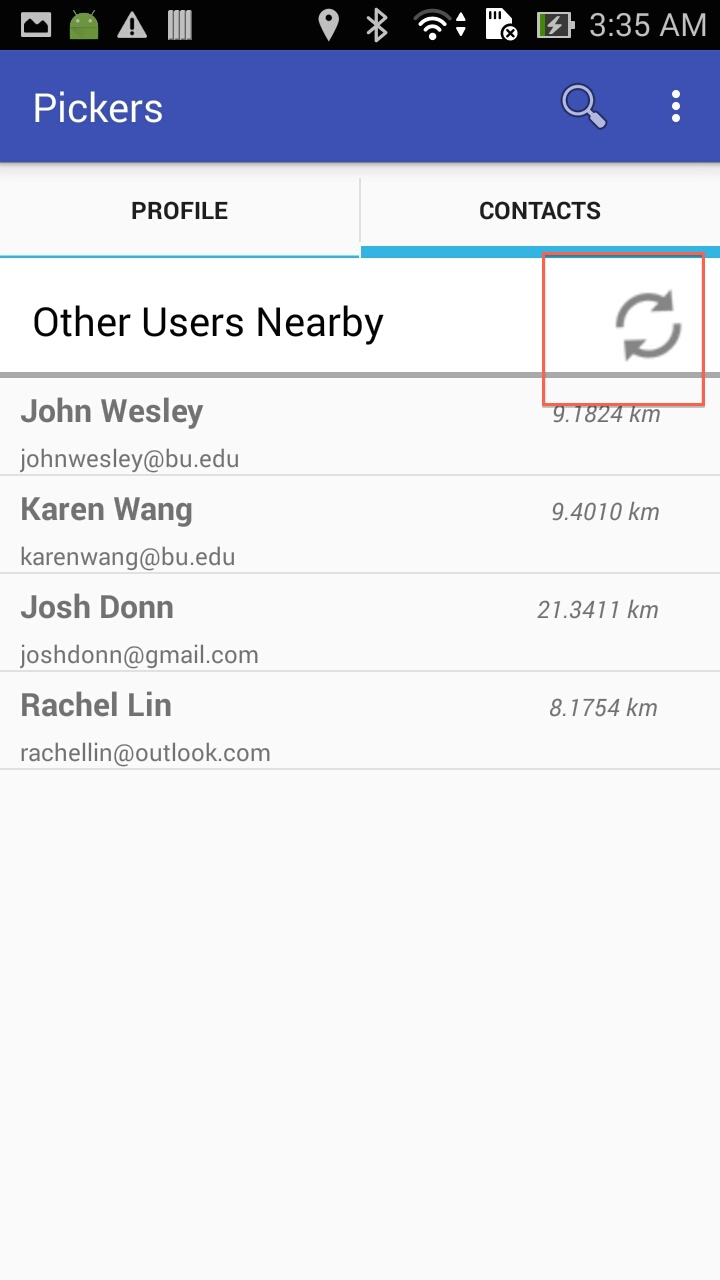
Application Feature 3: (Now all the users displayed on the map will also be shown on the page with their distance info displayed.)



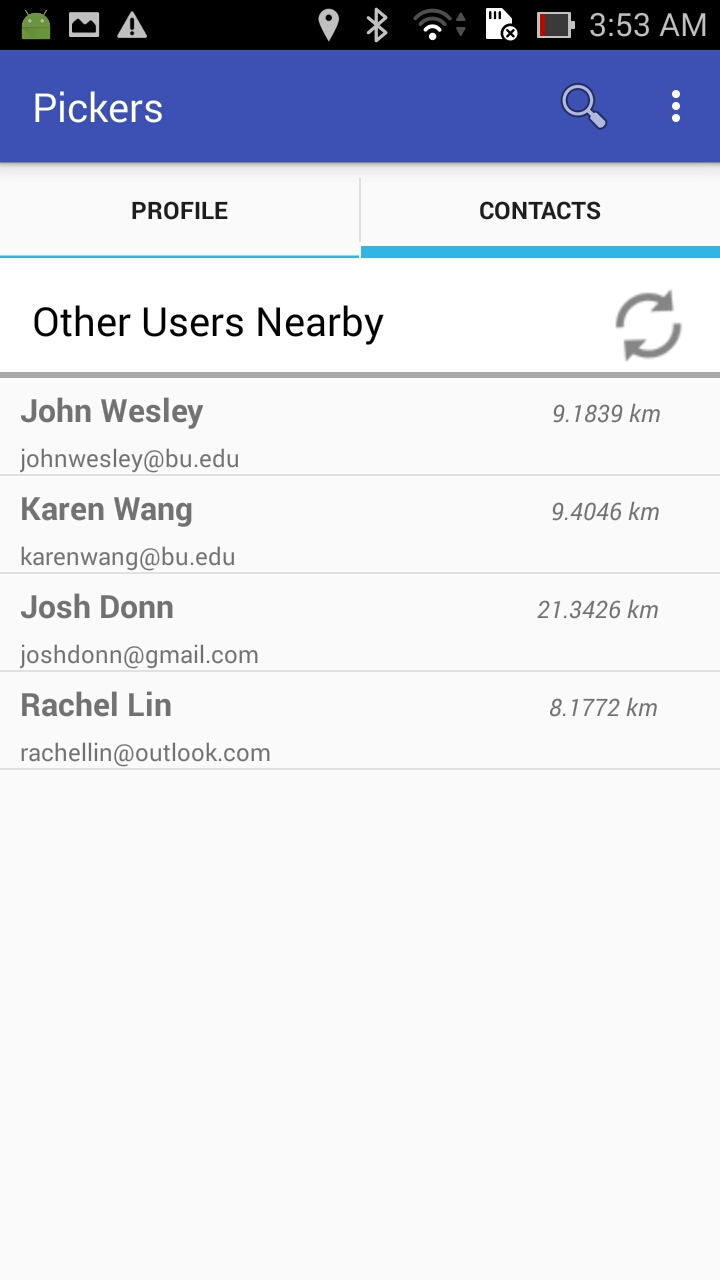
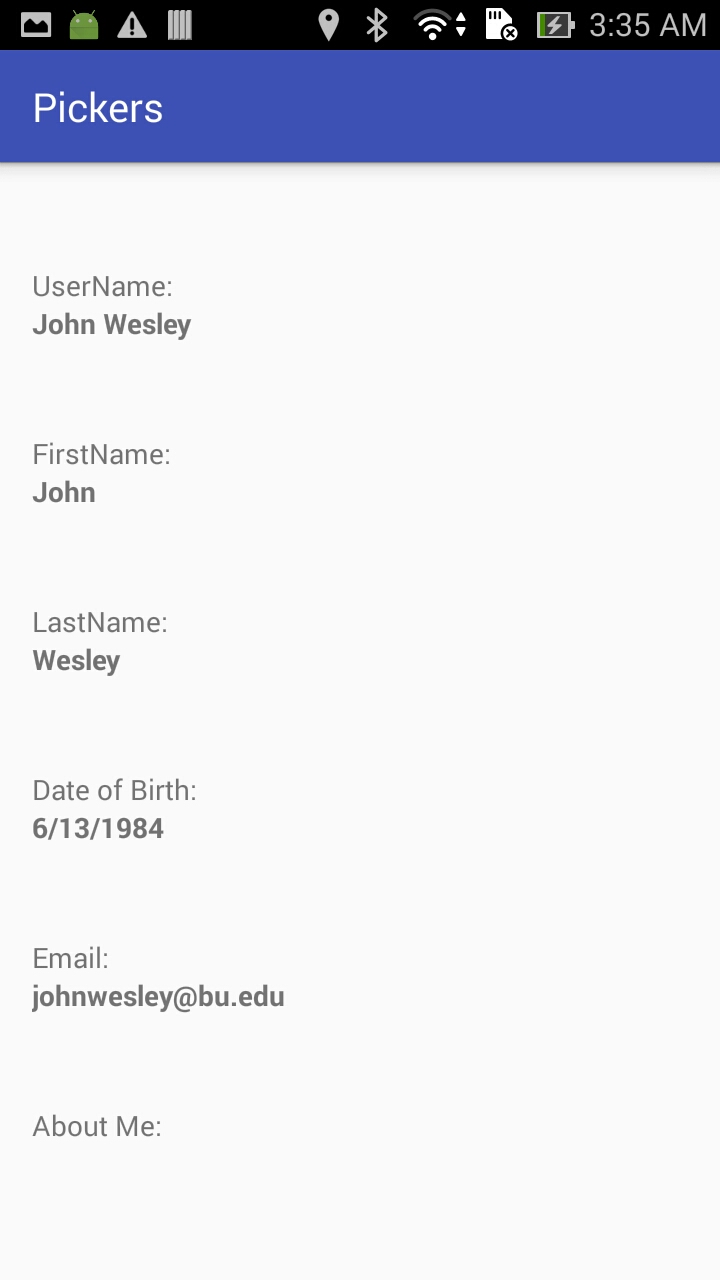
Application Feature 4: (When users click the Refresh icon on the top-right corner, the distance data mentioned above will be recalculated.)

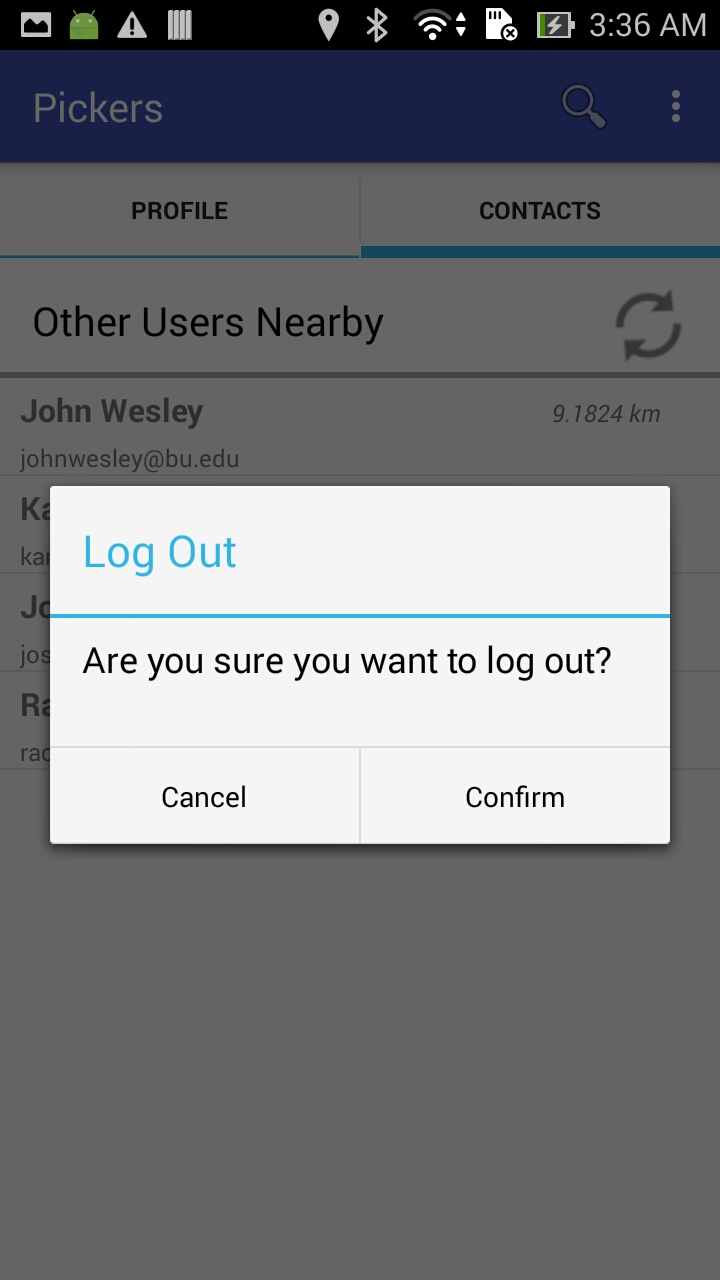
Application Feature 5: (The refresh button is fully animated when users click it, it will rotate until the distance data is calculated.)



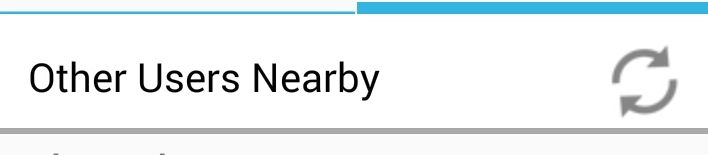
Application Feature 6: (Now when user clicks any user in user list, it will take them to each of those user’s profile page.)

Application Feature 7: (Users now can log out and an alert dialog will appear for confirmation.)



Application Feature 8: (We use custom graphics to create a border for the header bar for displaying nearby user.)

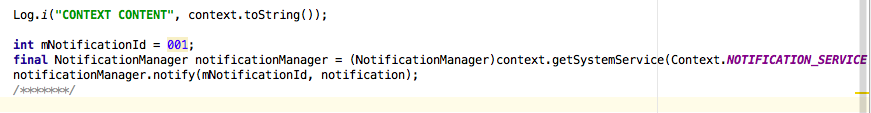


# Part 4—List of Android Elements: *(maximum 1 page of 12-point text)*

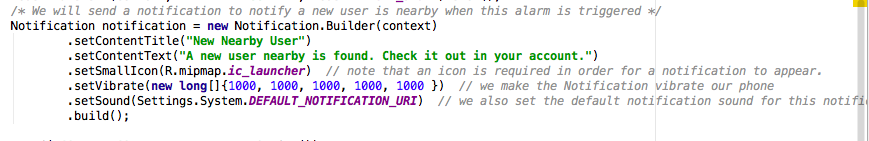
Use as many of the Android elements (e.g., built-in keywords, data types or functions) as possible covered in module 1. Provide lines from your code and where they were used once.

(New elements covered in this week’s module)

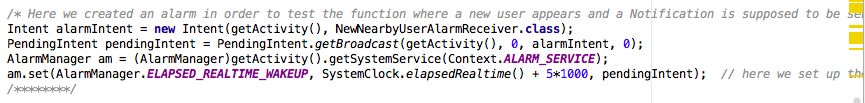
**New Element #1 (Use of Notification.)**



**New Element #2 (Use of Virbation for Notification.)**

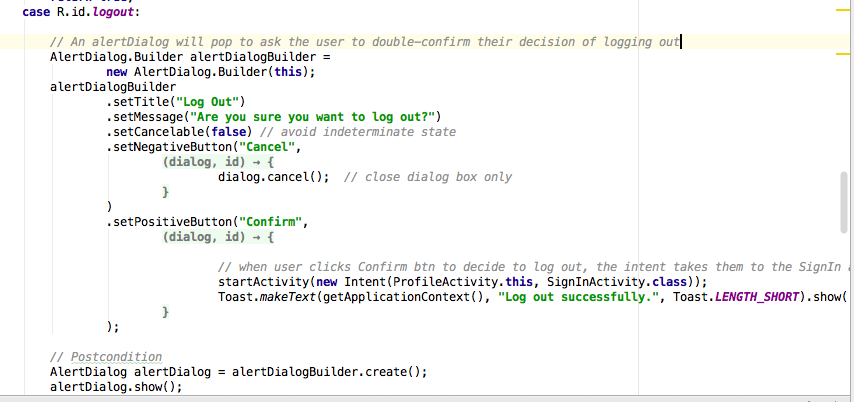


**New Element #3 (Use of Alarms.)**



Macintosh HD:Users:ChihWu:Desktop:Screen Shot 2016-04-26 at 4.35.24 AM.png

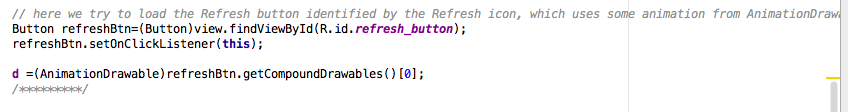
**New Element #4 (Use of Alerts.)**



**New Element #5 (Drawing with XML – use of shape tag)**

****

**New Element #6 (Drawing with Drawable Class)**

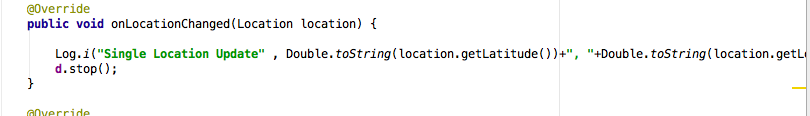
****

**New Element #7(Use of ImageView)**

****

**New Element #8(XML animation)**

Macintosh HD:Users:ChihWu:Desktop:Screen Shot 2016-04-26 at 4.31.47 AM.png



**New Element #9(Single Location Request—requestSingleUpdate())**



# Part 5—Example Code

Provide a page—in 10-point Ariel Narrow font—of your best (clearly documented)[[1]](#footnote-1) code. You may precede it with a paragraph (not counted in the page limit) explaining where and how it fits with the rest of your project.

The code I selected is the one in the FunctionListFragment.java file as it contains many features covered in this module, such as: Notification, Alert, Alarm, and Drawable class, etc.

**FunctionListFragment.java**

**package** com.example.chihwu.picker;  
  
  
  
  
**import** android.app.AlarmManager;  
**import** android.app.PendingIntent;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.database.Cursor;  
**import** android.graphics.drawable.AnimationDrawable;  
**import** android.location.Location;  
**import** android.location.LocationManager;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.os.SystemClock;  
**import** android.support.v4.app.Fragment;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.AdapterView;  
**import** android.widget.Button;  
**import** android.widget.ImageButton;  
**import** android.widget.ListView;  
**import** android.widget.SimpleAdapter;  
**import** android.widget.TabHost;  
**import** android.widget.TextView;  
**import** android.util.Log;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** android.view.View.OnClickListener;  
**import** android.location.LocationListener;  
**import** android.widget.AdapterView.OnItemClickListener;  
**import** android.widget.Toast;  
**import** android.support.v4.app.FragmentManager;  
**import** android.location.Criteria;  
**import** dataObjects.User;  
**import** utility.util;  
**import** database.PickersDB;  
  
  
**import** android.provider.ContactsContract.Contacts;  
**import** android.provider.ContactsContract.Data;  
**import** android.provider.ContactsContract.CommonDataKinds.Phone;  
  
**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.SupportMapFragment;  
**import** com.google.android.gms.maps.model.CameraPosition;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
  
  
*//Known issues: Please note that for the content provider for the contacts to work properly,  
// one needs to use a real android device rather than the emulator.***public class** FunctionListFragment **extends** Fragment **implements** OnClickListener, LocationListener, OnItemClickListener {  
  
 **private** TextView **functionTextView**;  
 **private** String **currentTabTag**;  
 **private** LocationManager **locationManager**;  
  
 */\*Variables for Profile Info Tab\*/* **private** TextView **user\_name\_txtView**;  
 **private** TextView **user\_firstName\_txtView**;  
 **private** TextView **user\_lastName\_txtView**;  
 **private** TextView **user\_password\_txtView**;  
 **private** TextView **user\_email\_txtView**;  
 **private** TextView **user\_dob\_txtView**;  
 **private** TextView **user\_intro\_txtView**;  
 */\*\*\*\*\*/  
  
 /\*Variables for the contacts Tab\*/* **private final** Uri **DATA\_URI** = Data.***CONTENT\_URI***; *//the CONTENT\_URI is a constant containing the Contacts ContentProvider's info from the class android.provider.ContactsContract* **private** ListView **contact\_list\_view**; *// for displaying all the contacts in a ListView in the tab Contacts of this fragment  
 /\*\*\*\*/  
  
  
 /\*Variables for the Contacts Tab\*/* **private** AnimationDrawable **d**;  
 **public static** ArrayList<User> *usersList*; *// this list is static and will be shared with other activities so far to share the info of other users of the application.  
 /\*\*\*\*/* @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 TabHost tabHost = (TabHost) container.getParent().getParent();  
 **currentTabTag** = tabHost.getCurrentTabTag();  
 View view = **null**;  
  
 Intent intent = getActivity().getIntent();  
 *// retrieve the username from the intent just passed from the MainActivity* String username = intent.getStringExtra(**"username"**);  
 **int** userID = intent.getIntExtra(**"userID"**, -1);  
  
 */\*find the id of the user who is logging in\*/* User searchedUser = MainActivity.*pickersDB*.getUser(userID);  
  
  
 *// here we create 4 users with static data  
 usersList* = **new** ArrayList<>();  
 User user1 = **new** User(10,**"John Wesley"**, **"John"**, **"Wesley"**, **""**,**"johnwesley@bu.edu"**,**"6/13/1984"**,**""**,42.424851337047545,-71.06131841156281);  
 User user2 = **new** User(11,**"Karen Wang"**, **"Karen"**, **"Wang"**, **""**,**"karenwang@bu.edu"**,**"12/12/1990"**,**""**,42.413851337047245,-71.03231841157281);  
 User user3 = **new** User(12,**"Josh Donn"**, **"Josh"**, **"Done"**, **""**,**"joshdonn@gmail.com"**,**"3/12/1981"**,**""**,42.524851436047545,-71.00132841256281);  
 User user4 = **new** User(13,**"Rachel Lin"**, **"Rachel"**, **"Lin"**, **""**,**"rachellin@outlook.com"**,**"5/4/1986"**,**""**,42.414851437047645,-71.06131831157281);  
  
 *usersList*.add(user1);  
 *usersList*.add(user2);  
 *usersList*.add(user3);  
 *usersList*.add(user4);  
  
 */\* Here we created an alarm in order to test the function where a new user appears and a Notification is supposed to be sent to notify the situation \*/* Intent alarmIntent = **new** Intent(getActivity(), NewNearbyUserAlarmReceiver.**class**);  
 PendingIntent pendingIntent = PendingIntent.*getBroadcast*(getActivity(), 0, alarmIntent, 0);  
 AlarmManager am = (AlarmManager)getActivity().getSystemService(Context.***ALARM\_SERVICE***);  
 am.set(AlarmManager.***ELAPSED\_REALTIME\_WAKEUP***, SystemClock.*elapsedRealtime*() + 5\*1000, pendingIntent); *// here we set up the setting that we will receive notification of new user nearby in 5 seconds after the switch to the ProfileActivity  
 /\*\*\*\*\*\*\*\*/  
  
 // here I created three different tabs* **if** (**currentTabTag**.equalsIgnoreCase(**"Profile"**)) *//when users select the Profile tab* {  
  
 view = inflater.inflate(R.layout.***fragment\_profile\_info***, container, **false**);  
  
 **user\_name\_txtView** = (TextView) view.findViewById(R.id.***username\_info\_txtView***);  
 **user\_firstName\_txtView** = (TextView) view.findViewById(R.id.***firtname\_info\_txtView***);  
 **user\_lastName\_txtView** = (TextView) view.findViewById(R.id.***lastname\_info\_txtView***);  
 **user\_password\_txtView** = (TextView) view.findViewById(R.id.***password\_info\_txtView***);  
 **user\_email\_txtView** = (TextView) view.findViewById(R.id.***email\_info\_txtView***);  
 **user\_dob\_txtView** = (TextView) view.findViewById(R.id.***dob\_info\_txtView***);  
 **user\_intro\_txtView** = (TextView) view.findViewById(R.id.***introduction\_info\_txtView***);  
  
 **user\_name\_txtView**.setText(searchedUser.getUserName());  
 **user\_firstName\_txtView**.setText(searchedUser.getFirstName());  
 **user\_lastName\_txtView**.setText(searchedUser.getLastName());  
 **user\_password\_txtView**.setText(searchedUser.getPassword());  
 **user\_email\_txtView**.setText(searchedUser.getEmail());  
 **user\_dob\_txtView**.setText(searchedUser.getDateOfBirth());  
 **user\_intro\_txtView**.setText(searchedUser.getIntroduction());  
  
 **functionTextView** = (TextView) view.findViewById(R.id.***function\_textView***);  
  
  
 }  
 **else if** (**currentTabTag**.equalsIgnoreCase(**"Contacts"**)) *//when users select the Contacts tab* {  
  
 */\*try to get the current location of the current logged-in user\*/* **if**(**locationManager** == **null**)  
 {  
 **locationManager** = (LocationManager)getActivity().getSystemService(Context.***LOCATION\_SERVICE***);  
 }  
  
 *// note that here we try to check if the proper permissions are granted before we can use the GPS and Network connection for location detection* PackageManager pm = getContext().getPackageManager();  
 **int** hasPerm = pm.checkPermission(android.Manifest.permission.***ACCESS\_FINE\_LOCATION***, getContext().getPackageName());  
 **if** (hasPerm == PackageManager.***PERMISSION\_GRANTED***) {  
 Log.*i*(**"PERMISSION GRANTED"**, **"We got the permission!!!!!!!!!!!!!"**);  
 *//note that here we try to use both Network and GPS for detecting the location updates just in case either of them might not work all the time* **locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***, 10000, 0, **this**);  
 **locationManager**.requestLocationUpdates(LocationManager.***NETWORK\_PROVIDER***, 10000, 0, **this**);  
 }  
  
  
 view = inflater.inflate(R.layout.***fragment\_contacts***, container, **false**);  
  
  
 */\* Populate the contact list with other users' data below \*/* ArrayList<HashMap<String, String>> data = **new** ArrayList<>();  
  
 **for** (User user : *usersList*) {  
  
 HashMap<String, String> map = **new** HashMap<String, String>();  
  
 map.put(**"contact\_name"**, user.getUserName());  
 map.put(**"contact\_email"**, user.getEmail());  
 data.add(map);  
  
 }  
  
 **int** resource = R.layout.***contacts\_listview\_item***;  
 String[] from = {**"contact\_name"**, **"contact\_email"**};  
 **int**[] to = {R.id.***contact\_name***, R.id.***contact\_email***};  
  
 SimpleAdapter adapter = **new** SimpleAdapter(getContext(), data, resource, from, to);  
 */\*\*\*\*\*\*\*\*\*/  
  
 // here we try to load the Refresh button identified by the Refresh icon, which uses some animation from AnimationDrawable class.* Button refreshBtn=(Button)view.findViewById(R.id.***refresh\_button***);  
 refreshBtn.setOnClickListener(**this**);  
  
 **d** =(AnimationDrawable)refreshBtn.getCompoundDrawables()[0];  
 */\*\*\*\*\*\*\*\*\*/* **contact\_list\_view** = (ListView)view.findViewById(R.id.***contactsListView***);  
 **contact\_list\_view**.setAdapter(adapter);  
  
 **contact\_list\_view**.setOnItemClickListener(**this**);  
  
  
 }  
  
 **return** view;  
 }  
  
  
 @Override  
 **public void** onResume() {  
 **super**.onResume();  
  
 }  
  
  
  
 @Override  
 **public void** onClick(View v) {  
  
 **switch** (v.getId())  
 {  
 *// when users click the refresh button, the distance info of each other user is recalculated.* **case** R.id.***refresh\_button***:  
 **d**.start(); *// where users click the refresh button, the animation for this button will start  
  
 // Next we will re-calculate and reload the distance data of each nearby users.* Criteria criteria = **new** Criteria();  
 criteria.setAccuracy(Criteria.***ACCURACY\_FINE***);  
 criteria.setPowerRequirement(Criteria.***POWER\_HIGH***);  
  
 PackageManager pm = getContext().getPackageManager();  
 **int** hasPerm = pm.checkPermission(android.Manifest.permission.***ACCESS\_FINE\_LOCATION***, getContext().getPackageName());  
 **if** (hasPerm == PackageManager.***PERMISSION\_GRANTED***) {  
  
 *// note that here we only try to update our location just once to prevent unnecessary continuous updating in the thread.* **locationManager**.requestSingleUpdate(criteria, **new** LocationListener() {  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
  
 Log.*i*(**"Single Location Update"** , Double.*toString*(location.getLatitude())+**", "**+Double.*toString*(location.getLongitude()));  
 **d**.stop();  
 }  
  
 @Override  
 **public void** onProviderDisabled(String provider) {  
 *//* ***TODO Auto-generated method stub*** }  
  
 @Override  
 **public void** onProviderEnabled(String provider) {  
 *//* ***TODO Auto-generated method stub*** }  
  
 @Override  
 **public void** onStatusChanged(String provider, **int** status,  
 Bundle extras) {  
 *//* ***TODO Auto-generated method stub*** }  
  
 }, **null**);  
 }  
  
 **break**;  
 }  
  
 }  
  
  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View v, **int** position, **long** id)  
 {  
 Log.*i*(**"ITEM CLICKED"**, Integer.*toString*(position));  
 Intent intent = **new** Intent(getActivity(), UserProfileActivity.**class**);  
 intent.putExtra(**"user\_index"**,position);  
 startActivity(intent);  
 }  
  
  
  
 *// This method is triggered whenever there is an update in users' location* @Override  
 **public void** onLocationChanged(Location location) {  
  
 **for** (**int** index=0 ; index<*usersList*.size(); index++) {  
 View view = **contact\_list\_view**.getChildAt(index);  
 TextView distance\_txt\_view = (TextView)view.findViewById(R.id.***contact\_distance***);  
  
 *// here we use an utility function created ourselves to calculate the user's distance with other nearby users using the latitude and logitude* **double** distance = util.*distance*(location.getLatitude(),location.getLongitude(),*usersList*.get(index).getCurrentLat(),*usersList*.get(index).getCurrentLong(),**"K"**);  
 distance\_txt\_view.setText(String.*format*(**"%.4f"**, distance)+**" km"**);  
 }  
  
 Log.*i*(**"LOCATION "**, **"Latitude:"** + location.getLatitude() + **", Longitude:"** + location.getLongitude());  
 *//Toast.makeText(getActivity(), "Latitude:" + Double.toString(location.getLatitude()) + ", Longitude:" + Double.toString(location.getLongitude()), Toast.LENGTH\_LONG).show();* }  
  
 @Override  
 **public void** onProviderDisabled(String provider) {  
 Log.*d*(**"Latitude"**, **"disable"**);  
 }  
  
 @Override  
 **public void** onProviderEnabled(String provider) {  
 Log.*d*(**"Latitude"**, **"enable"**);  
 }  
  
 @Override  
 **public void** onStatusChanged(String provider, **int** status, Bundle extras) {  
 Log.*d*(**"Latitude"**, **"status"**);  
 }  
  
}

# Below is the matrix that will be used to evaluate your response:



A regular A translates as 95, A-=93, B+=87, B=85, B-=83, C+=77, C=75, C-=73, D+=67, F=0 etc. To get an A grade for the course, your weighted average should be >93. A-:>=90. B+:>=87. B:>83. B-:>=80. C+:>=77. C:>73. C-:>=70 etc.

1. Document intentions—don’t paraphrase code. Nontrivial functions should have (an informal) *Intent* statement*,* (precise) *Preconditions* (if any), *Returns* (if any),and *Postconditions* (always). Each block of code should be preceded by its intended objectives. [↑](#footnote-ref-1)