ChihYung Wu

4. 6.2016

Assignment 3

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.

# At this stage, the application can allow users to sign up and sign in. After they sign in, users can see their profile info, and the friends on their contact list for future location detection purpose. The function of searching new and current friends on the map is still under construction.

# 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.

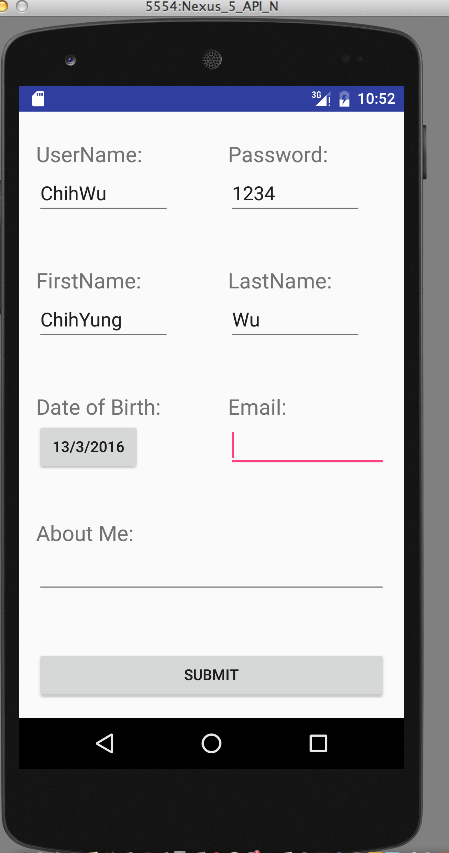
Application Feature 1:

User can choose to sign up and sign in on the homepage.



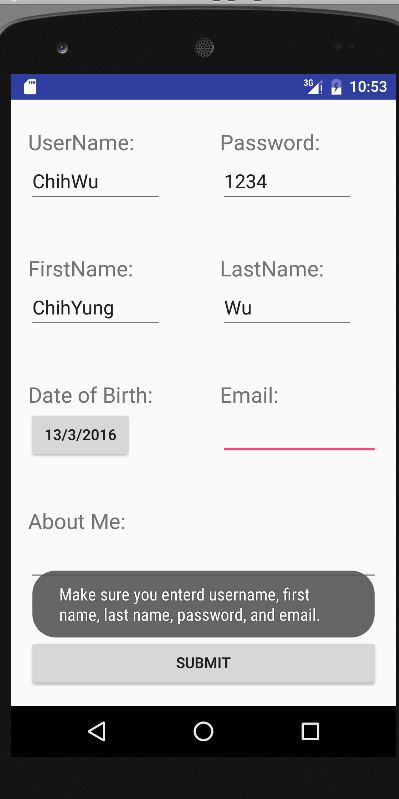
Application Feature 2:

The sign-up page allows users to enter their personal data and sign up.



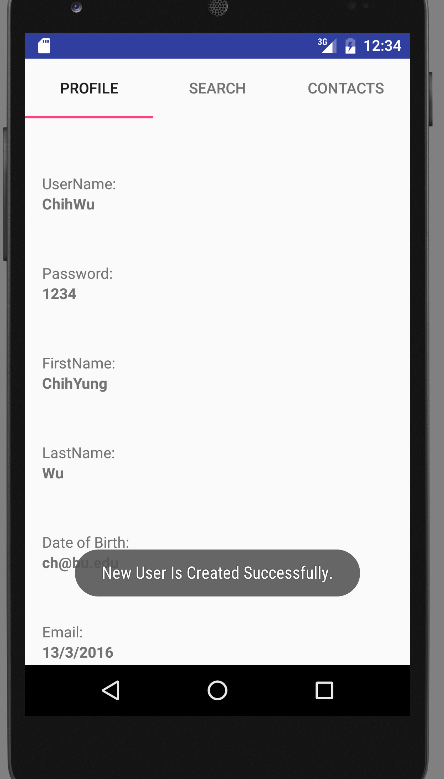
Application Feature 3:

The UserName, Password, FirstName, LastName, and Email fields are required fields. If any one is not filled, a Toast warning message will pop up.



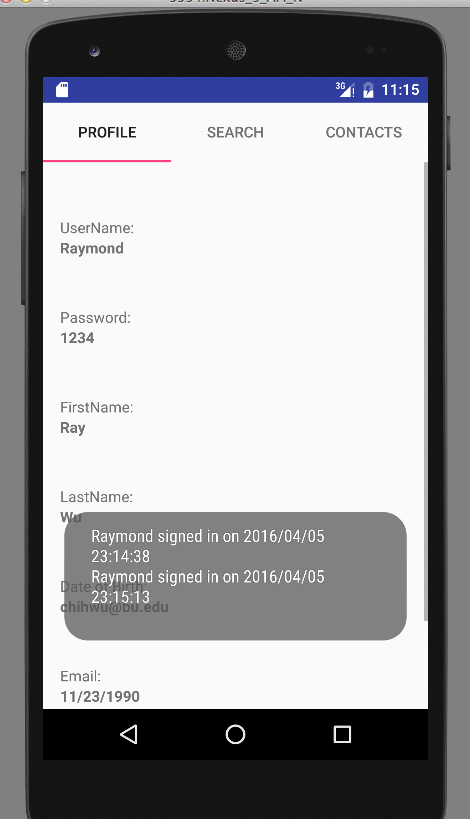
Application Feature 4:

Once users put in all the required info, they can be successfully signed up and led to the next profile page. (The application will only accept unique Email.)



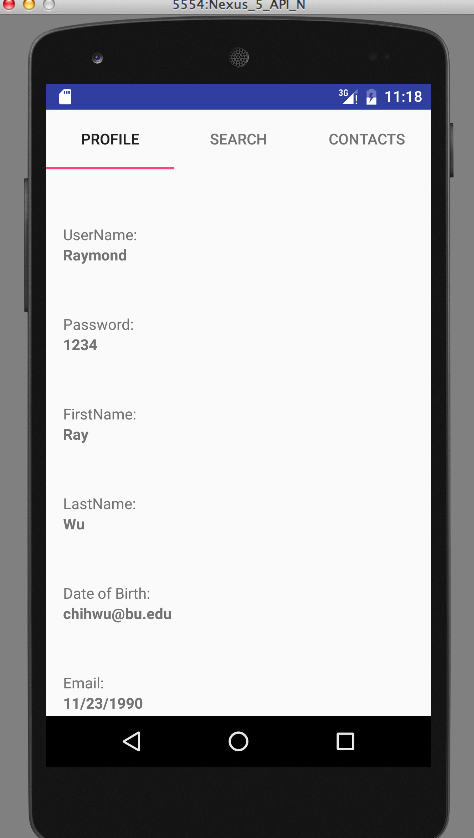
Application Feature 5:

The users can sign in, and right after they sign in a Toast message will show the history of their sign in records.



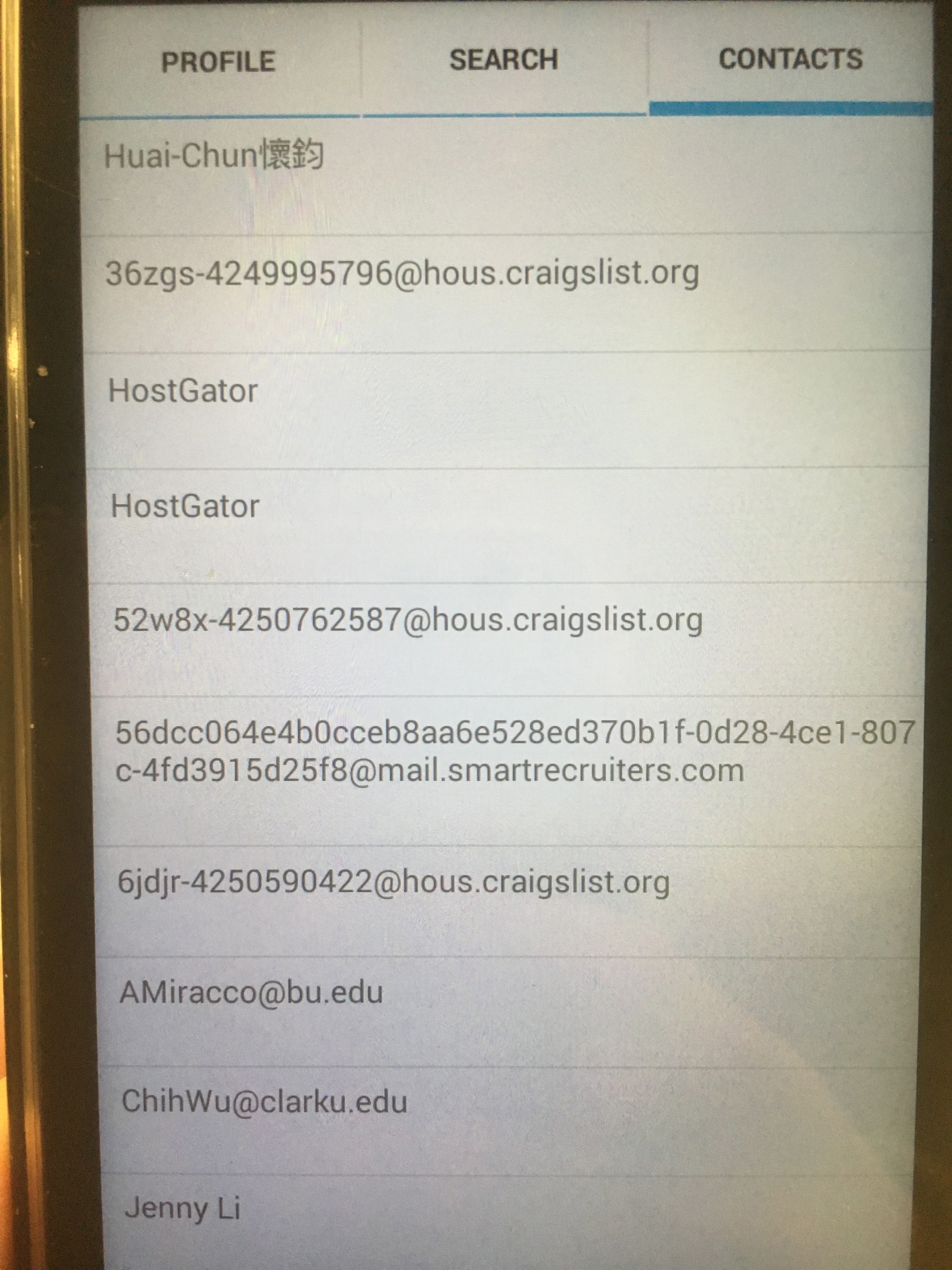
Application Feature 6:

After users sign in, then users will first view their profile and they can also select different functions using the tabs displayed at the top.



Application Feature 7:

Users can click the Contacts tab to see all the contacts found from their phone.

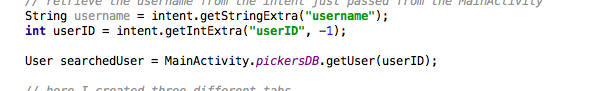


# Part 4—List of Android Elements: *(maximum 1 page of 12-point text)*

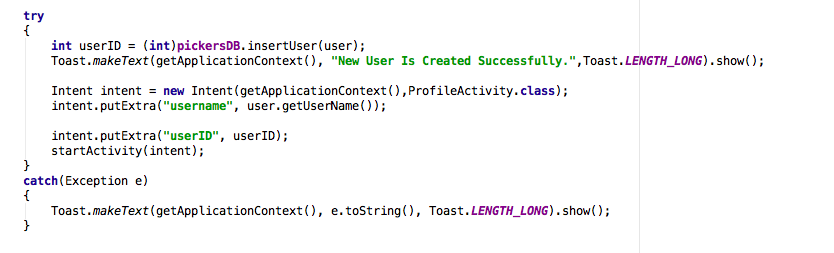
Use as many of the Android elements (e.g., built-in keywords, data types or functions) as possible covered in module 1. Provide lines from your code and where they were used once.

(New Features For this Module)

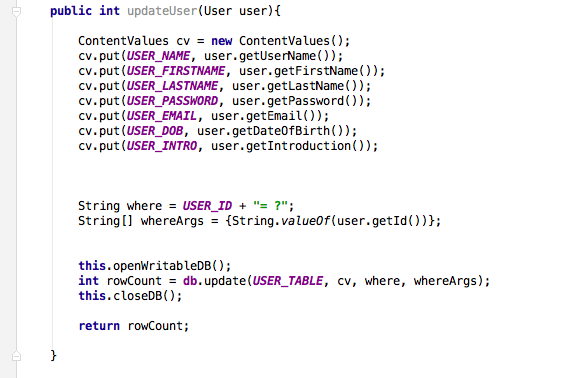
Android Feature 1: (Query Database)



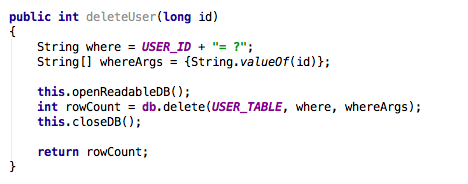
Android Feature 2: (Insert Into Database)



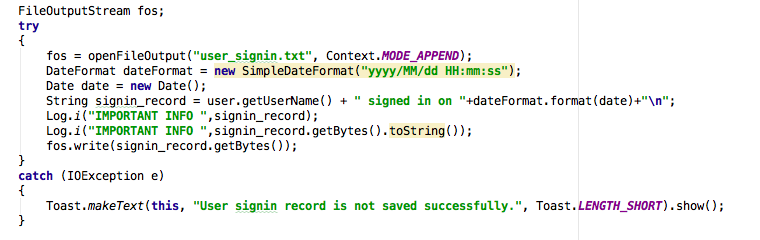
Android Feature 3: (Update Database)



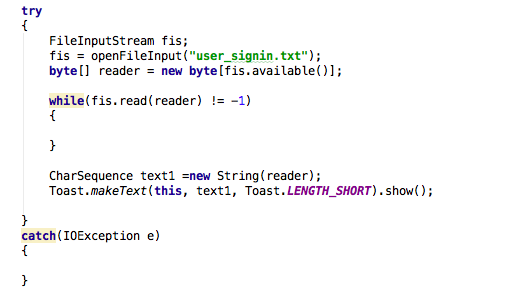
Android Feature 4: (Delete From Database)



Android Feature 5: (Writing to a Text File)



Android Feature 6: (Reading from a Text File)



Android Feature 7: (Content Provider)



Android Feature 8: (Tabs Manager)



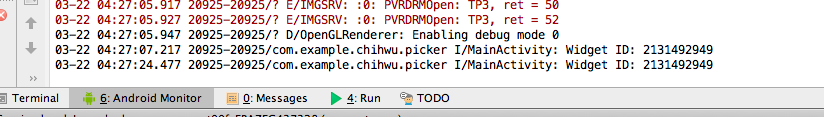
Android Feature 9. (Saving an Activity States using SharedPreferences):

Macintosh HD:Users:ChihWu:Desktop:Screen Shot 2016-03-22 at 5.15.10 AM.png

(Features In the Past)

Android Feature 1: **(Logging):**

Macintosh HD:Users:ChihWu:Desktop:Screen Shot 2016-03-22 at 5.14.19 AM.png



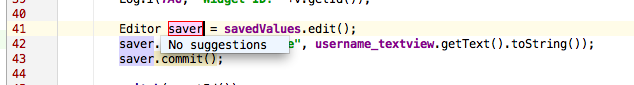
Android Feature 2. **(Saving an Activity States using SharedPreferences):**

Macintosh HD:Users:ChihWu:Desktop:Screen Shot 2016-03-22 at 5.15.10 AM.png

Android Feature 3. **(Relative Layout):**



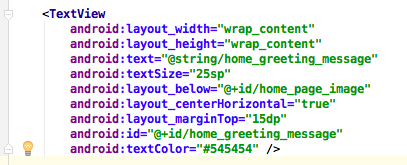
Android Feature 4. **(Use of Refactoring):**



Android Feature 5. **(ImageView):**



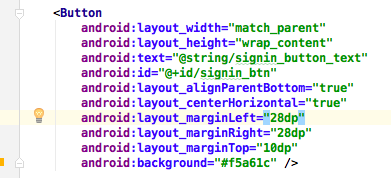
Android Feature 6. **(TextView):**



Android Feature 7. **(EditText):**

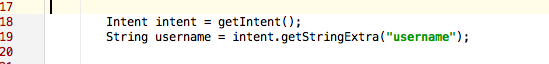


Android Feature 8. **(Button):**

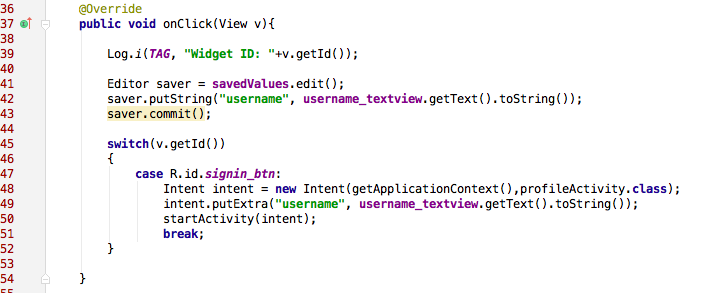


Android Feature 9. **(Use of Intent):**





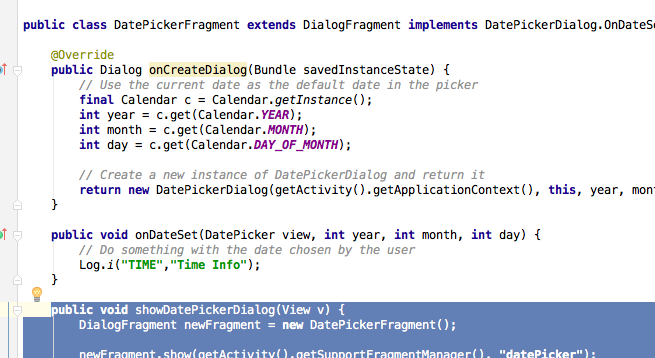
Android Feature 10. **(Use of ClickEventHandler):**



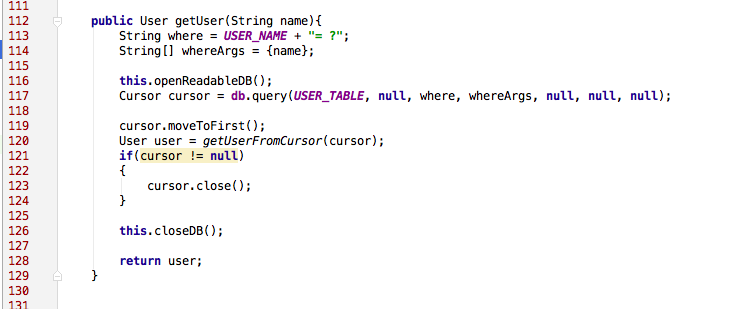
Android Feature 10. **(Use of Scrollview):**



Android Feature 11. **(Use of DatePickerFragment):**



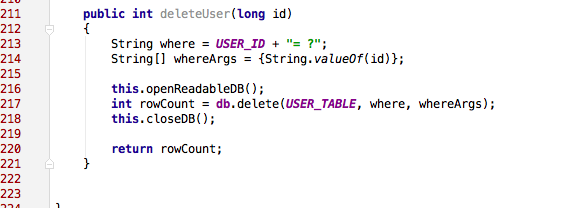
Android Feature 12:(Query data from database)



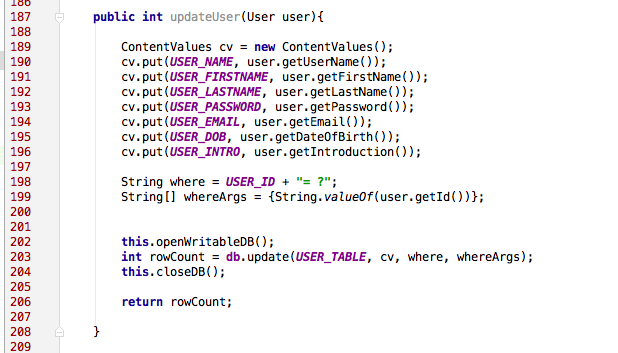
Android Feature 12:(Insert data into database)



Android Feature 13: (Delete from database)



Android Feature 14:(Update data from database)



# 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 think that best represents my work for this module are the file PickerDB.java and FunctionListFragment.java. The PickerDB.java has all the database logics, and the FunctionListFragment.java has all the logics for the tab functions of my app and all the functions my app provides so far.

**PickerDB. Java**

**import** android.content.ContentValues;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase.CursorFactory;  
**import** android.database.Cursor;  
  
**import** dataObjects.User;  
  
*/\*\*  
 \* Created by ChihWu on 4/5/16.  
 \*/***public class** PickersDB {  
  
 **public static final** String ***DB\_NAME*** = **"pickers2.db"**;  
 **public static final int *DB\_VERSION*** = 2;  
  
 **public static final** String ***USER\_TABLE*** = **"user"**;  
  
 **public static final** String ***USER\_ID*** = **"\_id"**;  
 **public static final int *USER\_ID\_COL*** = 0;  
  
 **public static final** String ***USER\_NAME*** = **"user\_name"**;  
 **public static final int *USER\_NAME\_COL*** = 1;  
  
 **public static final** String ***USER\_FIRSTNAME*** = **"user\_firstname"**;  
 **public static final int *USER\_FIRSTNAME\_COL*** = 2;  
  
 **public static final** String ***USER\_LASTNAME*** = **"user\_lastname"**;  
 **public static final int *USER\_LASTNAME\_COL*** = 3;  
  
 **public static final** String ***USER\_PASSWORD*** = **"user\_password"**;  
 **public static final int *USER\_PASSWORD\_COL*** = 4;  
  
 **public static final** String ***USER\_DOB*** = **"user\_dob"**;  
 **public static final int *USER\_DOB\_COL*** = 5;  
  
 **public static final** String ***USER\_EMAIL*** = **"user\_email"**;  
 **public static final int *USER\_EMAIL\_COL*** = 6;  
  
 **public static final** String ***USER\_INTRO*** = **"user\_intro"**;  
 **public static final int *USER\_INTRO\_COL*** = 7;  
  
  
 **public static final** String ***CREATE\_USER\_TABLE*** =  
 **"CREATE TABLE "** + ***USER\_TABLE*** + **" ("**+  
 ***USER\_ID*** + **" INTEGER PRIMARY KEY AUTOINCREMENT, "** +  
 ***USER\_NAME*** + **" TEXT NOT NULL, "**+  
 ***USER\_FIRSTNAME*** + **" TEXT, "**+  
 ***USER\_LASTNAME*** + **" TEXT, "**+  
 ***USER\_PASSWORD*** + **" TEXT NOT NULL, "**+  
 ***USER\_EMAIL*** + **" TEXT NOT NULL UNIQUE, "**+  
 ***USER\_DOB*** + **" TEXT, "**+  
 ***USER\_INTRO*** + **" TEXT);"**;  
  
 **public static final** String ***DROP\_USER\_TABLE*** =  
 **"DROP TABLE IF EXISTS "**+***USER\_TABLE***;  
  
  
 **private static class** DBHelper **extends** SQLiteOpenHelper{  
  
 **public** DBHelper(Context context) {  
 **super**(context, ***DB\_NAME***, **null**, ***DB\_VERSION***);  
 }  
  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db){  
 db.execSQL(***CREATE\_USER\_TABLE***);  
  
 db.execSQL(**"INSERT INTO user VALUES(1,'Raymond','Ray','Wu','1234', 'chihwu@bu.edu','11/23/1990','Hi, this is Ray.')"**);  
 }  
  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion)  
 {  
 db.execSQL(***DROP\_USER\_TABLE***);  
 onCreate(db);  
 }  
  
 }  
  
  
 **private** SQLiteDatabase **db**;  
 **private** DBHelper **dbHelper**;  
  
 **public** PickersDB(Context context)  
 {  
 **dbHelper** = **new** DBHelper(context);  
 }  
  
 **private void** openReadableDB(){  
 **db** = **dbHelper**.getReadableDatabase();  
 }  
  
 **private void** openWritableDB()  
 {  
 **db** = **dbHelper**.getWritableDatabase();  
 }  
  
 **private void** closeDB()  
 {  
 **if**(**db** != **null**)  
 {  
 **db**.close();  
 }  
 }  
  
  
 **public** User getUser(**int** id){  
 String where = ***USER\_ID*** + **"= ?"**;  
 String[] whereArgs = {Integer.*toString*(id)};  
  
 **this**.openReadableDB();  
 Cursor cursor = **db**.query(***USER\_TABLE***, **null**, where, whereArgs, **null**, **null**, **null**);  
  
 cursor.moveToFirst();  
 User user = *getUserFromCursor*(cursor);  
 **if**(cursor != **null**)  
 {  
 cursor.close();  
 }  
  
 **this**.closeDB();  
  
 **return** user;  
 }  
  
 **public** User getUserByEmail(String email){  
 String where = ***USER\_EMAIL*** + **"= ?"**;  
 String[] whereArgs = {email};  
  
 **this**.openReadableDB();  
 Cursor cursor = **db**.query(***USER\_TABLE***, **null**, where, whereArgs, **null**, **null**, **null**);  
  
 cursor.moveToFirst();  
 User user = *getUserFromCursor*(cursor);  
 **if**(cursor != **null**)  
 {  
 cursor.close();  
 }  
  
 **this**.closeDB();  
  
 **return** user;  
 }  
  
  
 **private static** User getUserFromCursor(Cursor cursor){  
  
 **if**(cursor == **null** || cursor.getCount() == 0)  
 {  
 **return null**;  
 }  
 **else** {  
 **try**{  
 **return new** User(  
 cursor.getInt(***USER\_ID\_COL***),  
 cursor.getString(***USER\_NAME\_COL***),  
 cursor.getString(***USER\_FIRSTNAME\_COL***),  
 cursor.getString(***USER\_LASTNAME\_COL***),  
 cursor.getString(***USER\_PASSWORD\_COL***),  
 cursor.getString(***USER\_EMAIL\_COL***),  
 cursor.getString(***USER\_DOB\_COL***),  
 cursor.getString(***USER\_INTRO\_COL***)  
 );  
  
  
  
  
 }  
 **catch**(Exception e)  
 {  
 **return null**;  
 }  
 }  
  
  
 }  
  
  
 **public long** insertUser(User user){  
  
 ContentValues cv = **new** ContentValues();  
 cv.put(***USER\_NAME***, user.getUserName());  
 cv.put(***USER\_FIRSTNAME***, user.getFirstName());  
 cv.put(***USER\_LASTNAME***, user.getLastName());  
 cv.put(***USER\_PASSWORD***, user.getPassword());  
 cv.put(***USER\_EMAIL***, user.getEmail());  
 cv.put(***USER\_DOB***, user.getDateOfBirth());  
 cv.put(***USER\_INTRO***, user.getIntroduction());  
  
 **this**.openWritableDB();  
  
 **long** rowID = **db**.insert(***USER\_TABLE***, **null**, cv);  
 **this**.closeDB();  
  
 **return** rowID;  
 }  
  
  
 **public int** updateUser(User user){  
  
 ContentValues cv = **new** ContentValues();  
 cv.put(***USER\_NAME***, user.getUserName());  
 cv.put(***USER\_FIRSTNAME***, user.getFirstName());  
 cv.put(***USER\_LASTNAME***, user.getLastName());  
 cv.put(***USER\_PASSWORD***, user.getPassword());  
 cv.put(***USER\_EMAIL***, user.getEmail());  
 cv.put(***USER\_DOB***, user.getDateOfBirth());  
 cv.put(***USER\_INTRO***, user.getIntroduction());  
  
  
  
 String where = ***USER\_ID*** + **"= ?"**;  
 String[] whereArgs = {String.*valueOf*(user.getId())};  
  
  
 **this**.openWritableDB();  
 **int** rowCount = **db**.update(***USER\_TABLE***, cv, where, whereArgs);  
 **this**.closeDB();  
  
 **return** rowCount;  
  
 }  
  
  
 **public int** deleteUser(**long** id)  
 {  
 String where = ***USER\_ID*** + **"= ?"**;  
 String[] whereArgs = {String.*valueOf*(id)};  
  
 **this**.openReadableDB();  
 **int** rowCount = **db**.delete(***USER\_TABLE***, where, whereArgs);  
 **this**.closeDB();  
  
 **return** rowCount;  
 }  
  
  
}

**FunctionListFragment. Java**

**package** com.example.chihwu.picker;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.v4.app.Fragment;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**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** dataObjects.User;  
**import** database.PickersDB;  
  
  
**import** android.provider.ContactsContract.Contacts;  
**import** android.provider.ContactsContract.Data;  
**import** android.provider.ContactsContract.CommonDataKinds.Phone;  
  
  
**public class** FunctionListFragment **extends** Fragment {  
  
 **private** TextView **functionTextView**;  
 **private** String **currentTabTag**;  
  
 */\*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***;  
 **private** ListView **contact\_list\_view**;  
 */\*\*\*\*/* @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);  
  
 User searchedUser = MainActivity.*pickersDB*.getUser(userID);  
  
 *// here I created three different tabs* **if**(**currentTabTag**.equalsIgnoreCase(**"Profile"**))  
 {  
 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(**"Search"**))  
 {  
 view = inflater.inflate(R.layout.***fragment\_function\_list***, container, **false**);  
  
 **functionTextView** = (TextView)view.findViewById(R.id.***function\_textView***);  
 }  
 **else if**(**currentTabTag**.equalsIgnoreCase(**"Contacts"**))  
 {  
 view = inflater.inflate(R.layout.***fragment\_contacts***, container, **false**);  
  
 String[] columns = {  
 Data.***\_ID***, *// primary key* Contacts.***DISPLAY\_NAME***, *// person's name* Data.***DATA1***, *// phone number* Data.***DATA2*** *// phone type (mobile, home, work, etc.)* };  
  
 *//String where = "("+Data.MIMETYPE + "='"+Phone.CONTENT\_ITEM\_TYPE+"')";  
 //String orderBy = Contacts.TIMES\_CONTACTED+" DESC";* Cursor cursor = getActivity().getContentResolver().query(**DATA\_URI**, columns, **null**, **null**, **null**);  
 ArrayList<User> contactList = **new** ArrayList<User>();  
  
 **while**(cursor.moveToNext())  
 {  
 User user = **new** User();  
 user.setUserName(cursor.getString(1));  
  
 contactList.add(user);  
 }  
  
  
 ArrayList<HashMap<String, String>> data = **new** ArrayList<>();  
  
 **for**(User user:contactList)  
 {  
 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);  
 **contact\_list\_view** = (ListView)view.findViewById(R.id.***contactsListView***);  
 **contact\_list\_view**.setAdapter(adapter);  
  
 }  
  
  
  
 refreshFragment();  
  
 **return** view;  
 }  
  
  
 **public void** refreshFragment()  
 {  
 String text = **"This is the "**+**currentTabTag**;  
 *//functionTextView.setText(text);* }  
  
 @Override  
 **public void** onResume()  
 {  
 **super**.onResume();  
 refreshFragment();  
 }  
  
}

# 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)