SI 543 Java – ConnectMentor Final Report

Erica Chan, Kushank Raghav, Yi-Yin Wang (Group 12)

Purpose

Our app provides a less formal way for students to connect with other students and/or working professionals for academic and/or career advice.

User group

- High school students: Connecting with students in different universities for college classes/activities advice.
- College students: Connecting with upperclassmen for classes/academic major advice.
- College students: Connecting with upperclassmen/working professionals for internship/full time advice.
- Working professionals: Connecting with other working professionals to network and build relationship, as well as career progression advice.
- “You get by giving.”: The working professionals connecting with college students to give them advice and college students helping high school students for academic advice.

Why is it important?

Other people’s experience can be very valuable to someone who is trying to make a decision between different colleges or different jobs as it may change the path that they will take in life. In addition, the mentor/mentee relationship may translate to long lasting friendship. Through this app, students can seek useful academic and professional guidance to achieve their goals. In addition, working professionals can use the network to progress in their career.

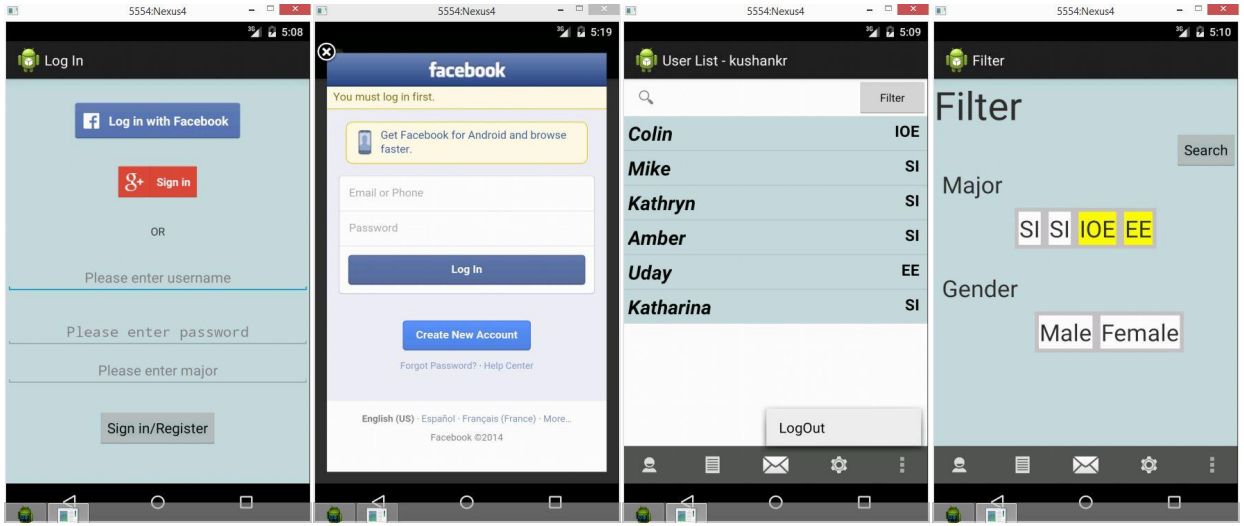
Competitors

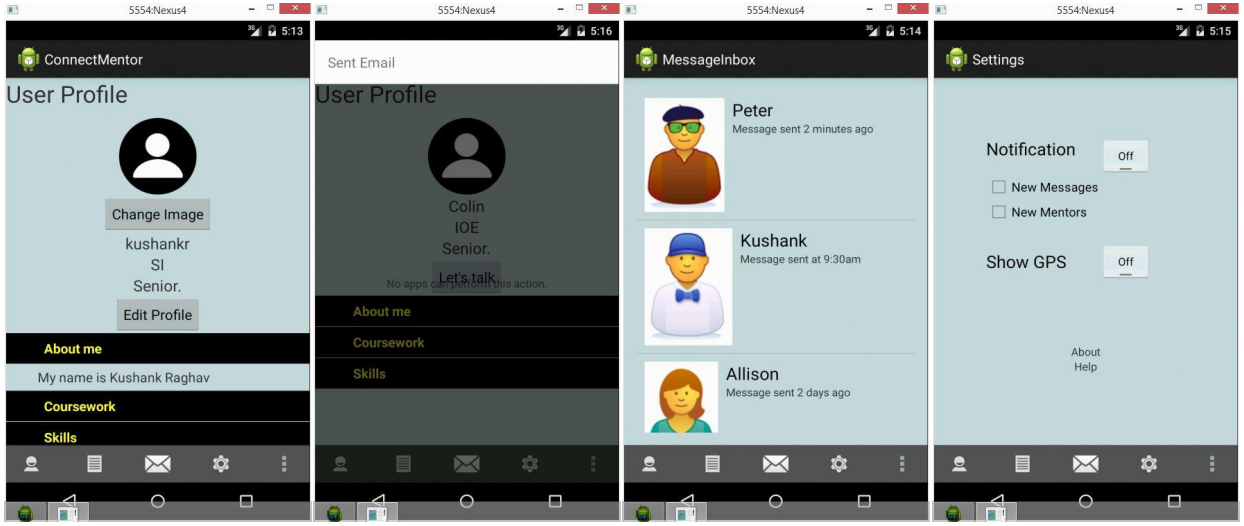
- Mentor (App)
- LinkedIn
- Quora

Why is your idea better?

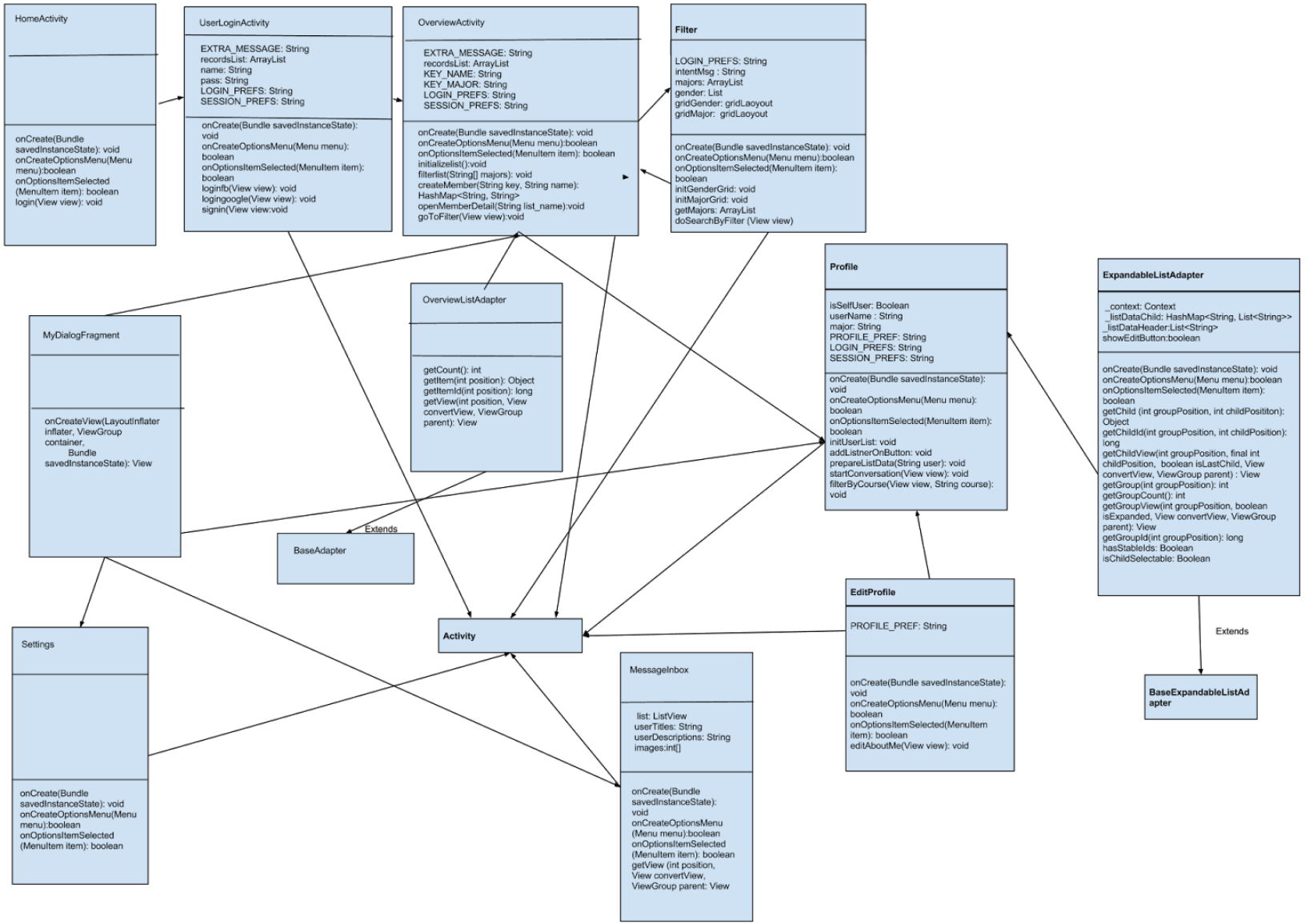
Our app targets a younger generation than other professional development app/sites. Our goal is to attract more high school and college students, who are more tech savvy and are untapped by LinkedIn or other professional development sites. The users then decide if they are interested in providing guidance to other students (mentor) through the app in addition to being a mentee. Therefore, the mentors are more likely to respond to incoming messages and to help these youngsters as they feel that they were once in their shoes. Most importantly, unlike Quora or other Q&A app, ConnectMentor provides a private platform where students can feel comfortable to share their queries.

Screenshots





UML CLASS DIAGRAM



Individual Write ups

Yi-Yin Wang

Profile Activity

The Profile activity is the screen where the users can view/edit their own profile or browse other users' info. To display correct info, the app need to determine which profile to show and populate the view dynamically. Further, there are four ways that user can land on Profile Activity. Each way would sent different intent and contain different message as listed below:

1. From Action Bar at the bottom - With Extra-message : current user id
2. From Overview Activity - With Extra-message: name and major of selected user
3. From EditProfile Activity -
    a. Back to parent activity: Without Extra-message
    b. Finish edit: With Extra-message : current user id

To solve the problem, I set up a global boolean variable to determine whether the user is viewing his own profile. I also use an error handler to process the intent and get the extra message accordingly. Once we decide which user's profile to show, the program retrieve data from shared preference and populate the views with an ExpandistViewAdapter.

```
/* Get the message from the intent */
Intent intent = getIntent();
userName = intent.getStringExtra(OverViewActivity.EXTRA_MESSAGE);
try {
    //userName = intent.getStringExtra(OverViewActivity.EXTRA_MESSAGE);
    userid = (int) Long.parseLong(userName);
    isSelfUser = true;
}catch (RuntimeException e){
    //  Intent from Overview or from EditProfile
    isSelfUser = (userName==null)?true:false;
}
```

Figure 1: the error handler to get extra message

Erica Chan

I implemented the Settings Screen, MessageBox Screen and Email Activity Screen. For the purpose of this write-up, I am selecting the email activity.

Email Activity

For the app, we decided that it is more convenient for the users to send emails to the mentors instead of building an in-house messager since people check their emails more often. Therefore, we decided to use an email function to have them communicate with one another. After clicking 'Let's talk' on a person's profile, an email box with pop-up.

To send an email from our application, I decided to use the default Email app provided from Android. I wrote an Activity that uses an Intent (intent= new Intent(Intent.ACTION_SEND)) to launch the default Email app. I start by using a Button (Let's talk) that enables the Intent when it's clicked. I then set the Intent type to message/rfc822, which prompt the email providers. Then, the user can choose which application (Mail/Gmail) to handle the Intent (chooser=Intent.createChooser(intent, "Sent Email")) I then set my email as a string so it's a default send to. In addition, I made the default subject to 'Invitation from ConnectMentor' and the body to contain 'Sent from ConnectMentor.'

```
/* Intents*/
public void startConversation(View view){
    Intent intent=null, chooser=null;

    if(view.getId()==R.id.btnTalk)
    {
        intent=new Intent(Intent.ACTION_SEND);
        intent.setData(Uri.parse("mailto:"));
        String[] to={"ericachan11@gmail.com"};
        intent.putExtra(Intent.EXTRA_EMAIL, to);
        intent.putExtra(Intent.EXTRA_SUBJECT,"Invitation from ConnectMentor");
        intent.putExtra(Intent.EXTRA_TEXT,"Sent from ConnectMentor");
        intent.setType("message/rfc822");
        chooser=Intent.createChooser(intent,"Sent Email");
        startActivity(chooser);
```

Kushank Raghav

I implemented the following screens and all functionalities that is found in them: Home Activity Screen, User Login Activity Screen, and OverView Activity Screen. However, for the purpose of this write-up, I will select one and that functionality is populating user list in Overview Activity through Shared Preferences and through filter.

OverView Activity

```
//Get names and major stored in shared preferences and display them (Except current user)
SharedPreferences loginsharedpref = getSharedPreferences(LOGIN_PREFS, Activity.MODE_PRIVATE);
SharedPreferences sessionpref = getSharedPreferences(SESSION_PREFS, Activity.MODE_PRIVATE);
String user = sessionpref.getString("Login", "");
Map<String, ?> allEntries = loginsharedpref.getAll();
for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
    if (!entry.getKey().toString().equals(user)) {
        recordsList.add(createMember(entry.getKey().toString(), entry.getValue().toString().split(":")[1]));
    }
```

In this code, I am creating two sharedpreferences objects - 'loginsharedpref' and 'sessionpref'. Login SharedPreferences file contains username as key and "password+":"+major" as value. Session SharedPreferences file contains "Login" as key and currently logged in username as value. By using getAll() function, I am getting key-value pairs present in Login SharedPreferences file and storing it in a Map object - "allEntries". I am using 'for' loop to loop through each entry. Within the loop, I am checking if the key of current entry (username) is equal to current logged in user. If this condition is false, I am creating a hashmap object (using createMember function that returns this object) with username as key and major as value (parameters to createMember function) and adding this to an existing Array List (recordsList - class variable). The if condition is being used to ensure that current logged in username is not displayed in the user list.

```
private void filterlist(String[] majors) {
    //Source:http://stackoverflow.com/questions/15979828/loop-through-an-arraylist-of-hashmaps-java
    //Remove from records list the rows that are not selected in filter => Only show rows selected by Filter
    // Use iterator to avoid a concurrent exception i.e. avoid error due to looping through array list being
    // modified by the program
   Iterator<HashMap<String, String>> it = recordsList.iterator();
    while (it.hasNext()) {
        HashMap<String, String> hmap = (HashMap<String, String>) it.next();
        if (!Arrays.asList(majors).contains(hmap.get("major").toString())) {
            it.remove(); // avoids a ConcurrentModificationException
        }
```

This code is a function that filters an existing Array List (recordsList - class variable) based on majors sent by the Filter screen. It takes a string array as parameters. This array was earlier formed by splitting filter message received by Overview Activity. My approach is to remove any row in the userlist that does not contain the elements in the majors array. Only rows with majors present in the majors array are displayed. The reason to use 'Iterator' class is because if one modifies an existing array list while simultaneously looping over it, it throws a Concurrent Modification Exception and this class helps us to avoid this exception. We first create an Iterator object - 'it'. This object is returned by calling 'iterator()' function on recordsList. This will help us loop through the list. We use a while loop and put a condition which will return true as long as there are more elements in the list. 'hmap' is an object that stores the next Hashmap object in the Array List. The 'if' condition is true if array list of majors (selected in filter screen) does not contain value corresponding to "major" key of hmap object (by using get(key_name) function of hashmap object). When this condition is true, we call the 'remove' function of iterator object to remove the current element.