/*Hope your final app prototypes are coming along nicely! I wanted to send out a quick reminder that both the final prototype and the write-up are due on 10/12 at 12:59pm.
Please commit the source code and the write-up to your github group repository as usually. The write-up should include all group member's parts as well as the overall pitch and UML diagram in one PDF.*/

Luke Rehmann
Joyce Sakata
Mingjue Ni

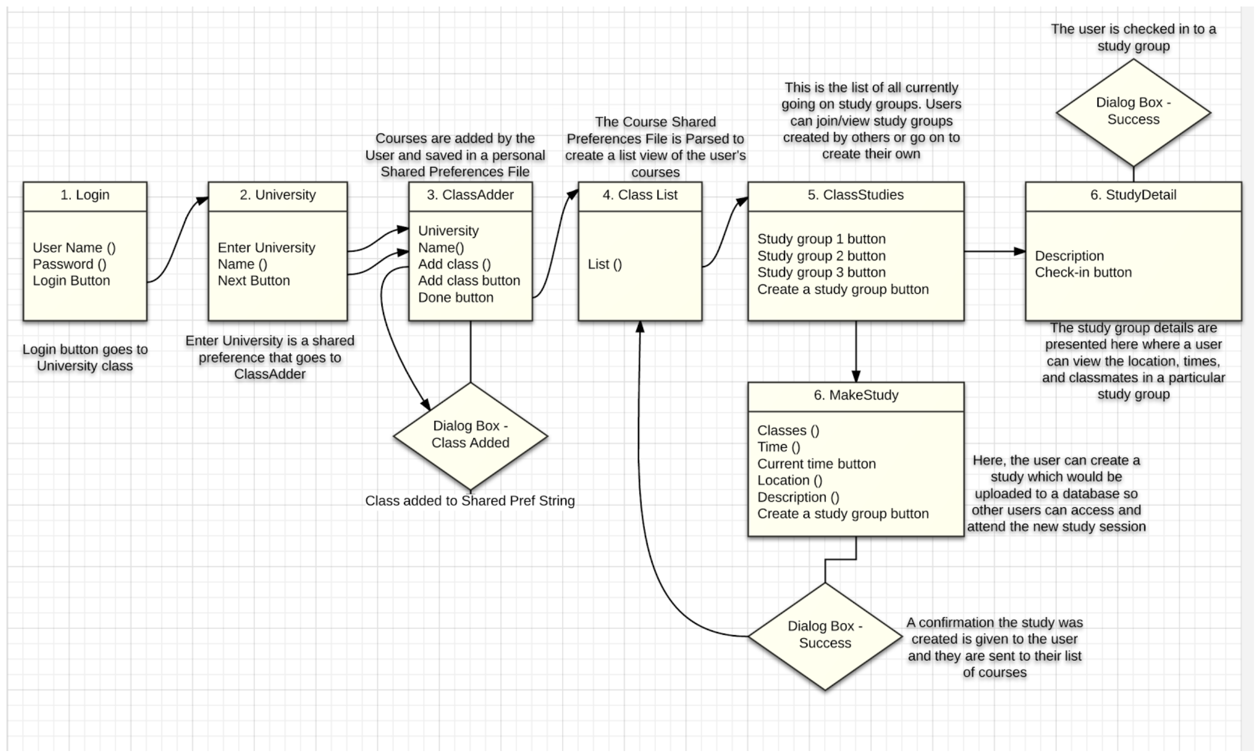
Final Writeup

Cahoot is a classwork collaboration app that focuses on bringing people together to better their education experience. Students can create public study groups for other students to join so that previously unconnected people can come together to work.

College classes are often very large with class sizes up to several hundred people. Meeting new people in this kind of setting can be intimidating, especially for introverted individuals. Research studies have shown that students who study in a group outperform students who study alone.[1] However, it is difficult to find a time that works for everyone and students may not know everyone in their classes. Build a prototype of a mobile application that allows users to create and see existing study groups for their classes. Students can also check into study groups. This mobile application allows students to easily create impromptu study sessions, meet new people and expand their networks while improving their performance in class.

There is no direct competitors to our application. Applications that allow users to create groups based on similar interests or classes include: Facebook Groups, Piazza, Ctools Forum, and Meetup.com. Although many of these sites are popular, they are not geared towards more impromptu or informal meet ups. Cahoot provides a space for students to create study groups with very low effort and requires no planning or coordinating based on other students' schedules.

[1] https://net.educause.edu/ir/library/pdf/ffp0604.pdf

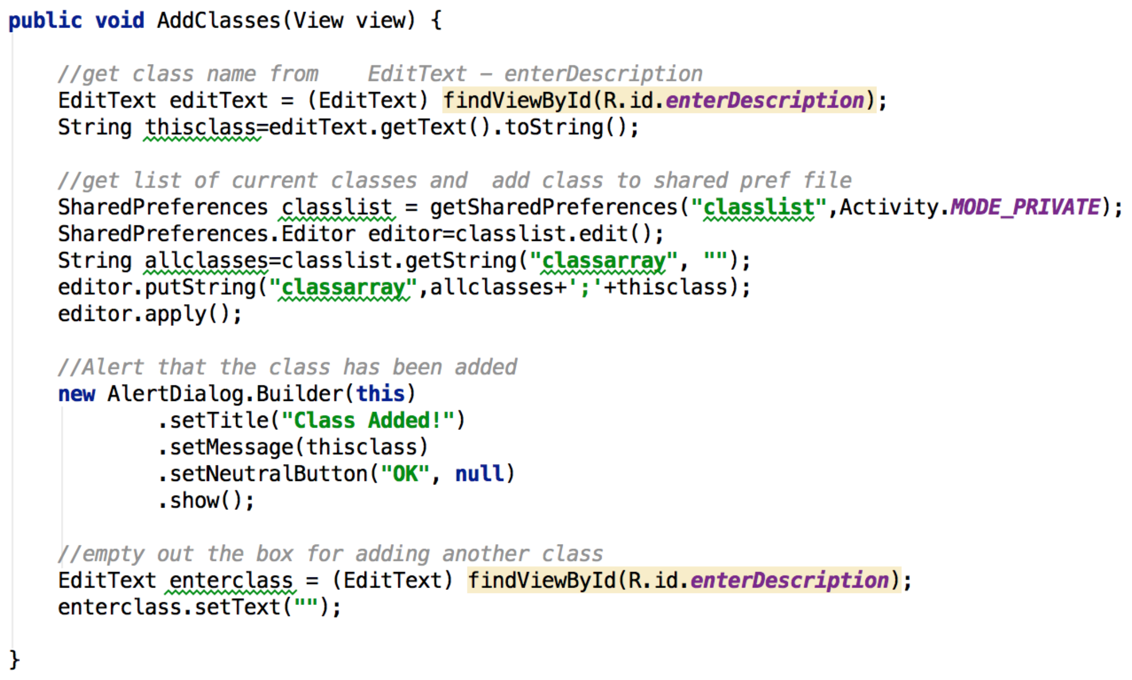

1. Login
User Name ()
Password ()
Login Button
Login button goes to University class
2. University
Enter University Name ()
Next Button
Enter University is a shared preference that goes to ClassAdder
Courses are added by the User and saved in a personal Shared Preferences File
3. ClassAdder
University Name()
Add class ()
Add class button
Done button
Dialog Box - Class Added
Class added to Shared Pref String
The Course Shared Preferences File is Parsed to create a list view of the user's courses
4. Class List
List ()
This is the list of all currently going on study groups. Users can join/view study groups created by others or go on to create their own
5. ClassStudies
Study group 1 button
Study group 2 button
Study group 3 button
Create a study group button
6. MakeStudy
Classes ()
Time ()
Current time button
Location ()
Description ()
Create a study group button
Here, the user can create a study which would be uploaded to a database so other users can access and attend the new study session
Dialog Box - Success
A confirmation the study was created is given to the user and they are sent to their list of courses
The user is checked in to a study group
Dialog Box - Success
6. StudyDetail
Description
Check-in button
The study group details are presented here where a user can view the location, times, and classmates in a particular study group

UML Diagram

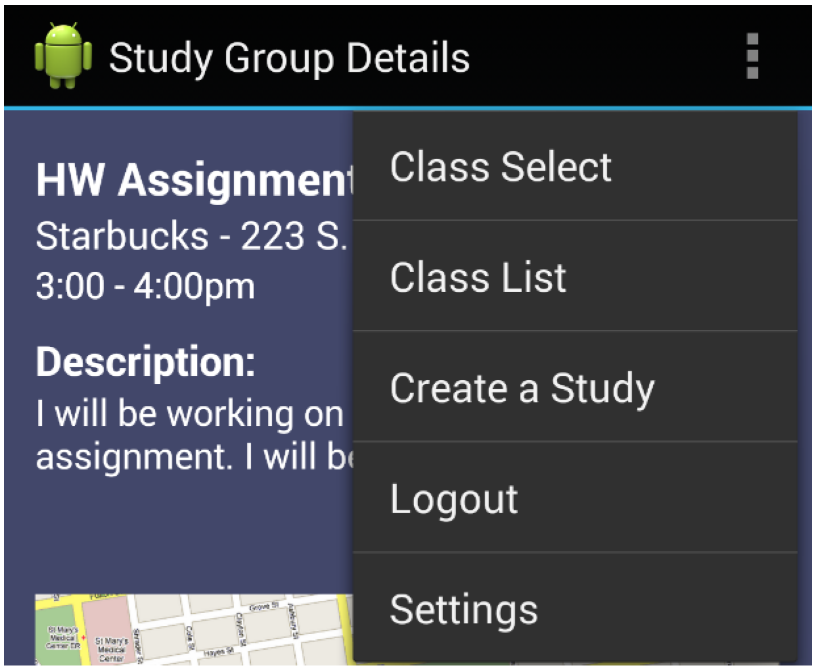
Luke - Shared Preferences

One of my roles in creating this app was adding shared preferences to make the hard coded activities more dynamic. Although some of the activities would have been much better served with a database backend, through shared preferences, we were able to emulate that and give the app a more real feel. Sharedpreferences allowed us to build in variables that were translated to arrays and then front end views; this would be very similar to implementing a database so I feel we are prepared for a theoretical next step.

I also played a large role in integrating proper intents in the app, in particular, the intents and xml changes that take place once a dialog box is clicked seemed to be more difficult to trigger so I programmed an “extra message” to be sent with the intent to later trigger a dialog box at the destination. Overall, I feel we worked well as a group and spent much time sharing activities that we worked on, debugging each others code when our sore eyes were unable to figure out what was wrong.



```
public void AddClasses(View view) {

    //get class name from    EditText - enterDescription
    EditText editText = (EditText) findViewById(R.id.enterDescription);
    String thisclass=editText.getText().toString();

    //get list of current classes and  add class to shared pref file
    SharedPreferences classlist = getSharedPreferences("classlist",Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor=classlist.edit();
    String allclasses=classlist.getString("classarray", "");
    editor.putString("classarray",allclasses+';'+thisclass);
    editor.apply();

    //Alert that the class has been added
    new AlertDialog.Builder(this)
            .setTitle("Class Added!")
            .setMessage(thisclass)
            .setNeutralButton("OK", null)
            .show();

    //empty out the box for adding another class
    EditText enterclass = (EditText) findViewById(R.id.enterDescription);
    enterclass.setText("");

}
```



Mingjue -

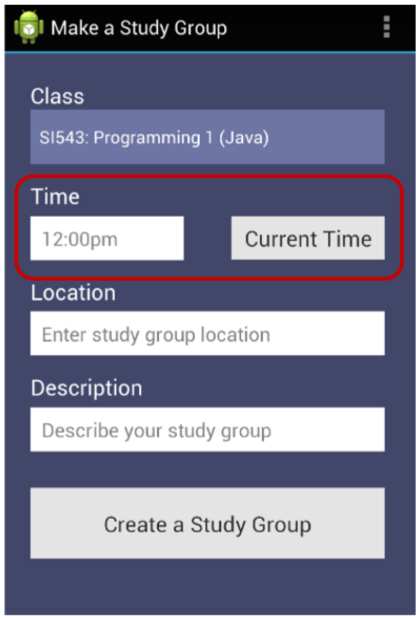
My primary roles in creating Cahoot involved the design of the Action-bar and menu as well as work on the class list views. The action bar allows users to perform common actions from anywhere in the app as well as the traditional logout action that is typically found in this location. For our app, the actionbar is replicated across all activities, and it is reduced to show only “settings” when a user is not logged in. In order to implement this, I created menu xml fragments that were integrated into each activity through java code as menu items.

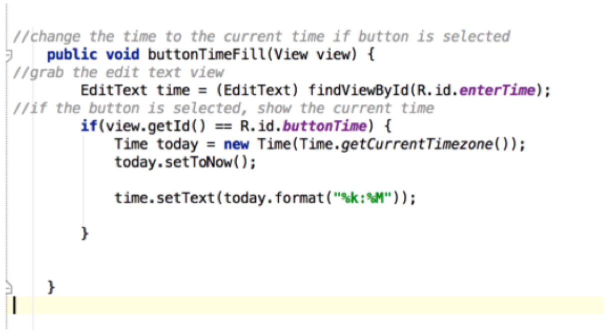
Our original design showed the list of classes as buttons, this worked well for the alpha design, but for the beta, we decided to integrate a more dynamic list that had the potential to easily show more than just 3 or 4 classes.

Joyce - Programmatic Changes

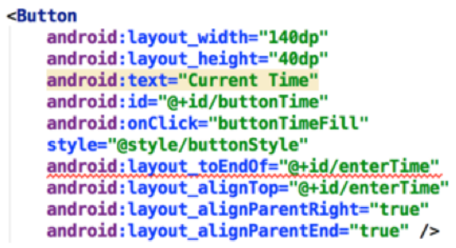
For this project, I am most proud of completing the programmatic time changes for the application. One of the primary goals and differentiating features of cahoot is to help students create impromptu study sessions. If a student finds himself at a coffee shop studying, he can instantly open his study session to others. In order to make it easier to create impromptu study groups, we incorporated programmatic changes so that when users create a study group, they can easily set the start time to the current time. This greatly improves the efficiency of creating a study group.

To programmatically change the time, I first created a button and a text view in the xml file. The id for this button is “buttonTime” and the id of the text view is “enterTime”. Within the MakeStudy class, I created the method “buttonTimeFill”. The purpose of this method is to change the time to the current time if “buttonTime” button is selected. In the next step, I grab the text view, “enterTime” and type cast it as EditText, which is a subclass of view. From there, I created an if statement. In this if statement, I state that if the button, “buttonTime” is selected, then set it to the current time. I used the native Time class and the existing java methods “getCurrentTimezone” and “setToNow” to grab the current time zone of the users computer and set it to the current time. I also told java to format the text in a particular format, separated by a colon using “%k:%M”. Back in the XML, I used the android onClick handler to call the method “buttonTimeFill”, which connects the java code to the xml interface.





```
//change the time to the current time if button is selected
    public void buttonTimeFill(View view) {
//grab the edit text view
        EditText time = (EditText) findViewById(R.id.enterTime);
//if the button is selected, show the current time
        if(view.getId() == R.id.buttonTime) {
            Time today = new Time(Time.getCurrentTimezone());
            today.setToNow();

            time.setText(today.format("%k:%M"));

        }


    }
```



```
<Button
    android:layout_width="140dp"
    android:layout_height="40dp"
    android:text="Current Time"
    android:id="@+id/buttonTime"
    android:onClick="buttonTimeFill"
    style="@style/buttonStyle"
    android:layout_toEndOf="@+id/enterTime"
    android:layout_alignTop="@+id/enterTime"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
```