Team: German Ostaszynski, Sarah Anderson, Troy Leonard

SI 543

December 10, 2014

Final Project Deliverable

LinguaMeet VC Pitch

LinguaMeet is an Android app for quickly connecting users who share the love of foreign language to one another for face-to-face conversations. Using a location-based identification and messaging service, this app brings the possibility of friendly exchanges in a user's second—or third, or fourth—language to an everyday reality.

Our primary target user for LinguaMeet is anyone trying to improve or maintain his or her foreign language skills, whether that be a novice looking to practice verb conjugation or a fluent speaker of a rare language who doesn't often find other speakers in everyday life. The instantaneous location-based connection method also means that on-the-go users won't have to add yet another meeting to their schedules; they can simply log in to see the polyglots nearest to their current location at the moment of their choosing.

Other foreign language apps on the market do not offer the unique blend of one-on-one interactions with location services that LinguaMeet does. Text-based and video-based apps cannot compare to the value of speaking to another individual in person for practicing and maintaining language skills. The one-on-one style also ensures that each user gets ample practice time (unlike interest-based group meeting apps), while still facilitating meeting new and interesting individuals.

As technology is making connecting to people across the globe easier than ever, it's equally as important to be able to converse with those people in meaningful ways. From foreign language students to travelers to those merely looking for engaging conversations, LinguaMeet facilitates the process of making these meaningful connections by helping users flex their second-language muscles in a convenient and simple way.

Changing Themes - Troy Leonard

A feature I implemented was changing themes across all activities. First, I created a new Java class called ThemeUtils.java. Within there are two methods, changeToTheme() and onActivityCreateSetTheme(). Both are public, meaning they can be accessed by all classes. The methods are also static, making them class methods. Being class methods also means that these methods can be called from another class without having to create a new instance of the ThemeUtils class. Next there are four variables declared: cTheme, C1, C2, and C3. cTheme is private, meaning only the ThemeUtils class can access it. The rest are public, and thus accessible to all other classes. All four variables are class variables, and they are all of the primitive data type integer. C1, C2, and C3 are set to 0, 1, and 2 respectively. These three class variables are also final, meaning they cannot be altered.

The onActivityCreateSetTheme() method takes an object from the Android Activity class as a parameter, in this case named ‘activity’. Next, this method uses a switch statement to check which theme is currently selected. The switch expression must use the same primitive type as the cases, which are ints here. Here, C1, C2, and C3 correspond to the Default, Dark, and UM themes respectively. The changeToTheme() method takes two parameters: an activity object and an int called ‘theme’. theme is made to equal cTheme (this determines the current theme depending on if C1, C2, or C3 is currently selected). It then ends the activity and reboots it with the new theme, which is done by the intent object in activity.startActivity(new Intent(activity, activity.getClass()));. How does the app know which theme is currently selected, though?

This aspect is handled in the Settings class. First, the onActivityCreateSetTheme() method is is called in the onCreate() method to set the current theme. The Settings class implements the Android class OnClickListener, which is used to “listen” to what button is pressed, with each corresponding to a respective theme. Below, another switch statement is used in conjunction with the OnClickListener. When the Dark theme button is clicked, for example, the switch statement, getting the id from the OnClickListener, sets the current theme by assigning the appropriate int value:

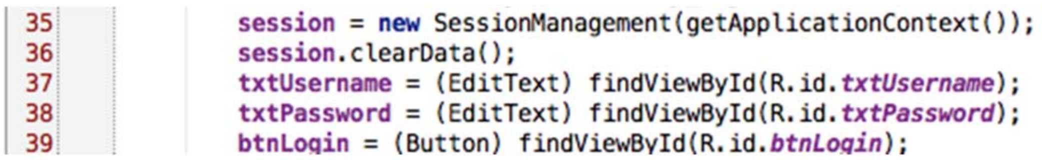
```
case R.id.custom2:
        ThemeUtils.changeToTheme(this, ThemeUtils.C2);
```

This carries over to the class method changeToTheme() which sees that C2 (which equals 1) is set. The theme is then changed to the corresponding selection. Applying it across all activities was as easy as making every class extend Settings. This could also have been done by putting the onActivityCreateSetTheme() class method in every other activity in their respective onCreate() methods.

LogIn - German Ostaszynski

We implemented the login feature using sharedPrefences. Once the login screen is encountered, an existing user, in the context of this project the username would be “Troy” and the password is “stuff.” When these values are prompted the application allows the user to navigate through the different views, and even change the themes of the application. If “Troy” is not logged in, the user would be taken to the main screen where he or she will have to prompt the user values, not doing so will make the rest of the application non-navigable.

Before being able to check if the username and password matches with “Troy” and “stuff” we created an object created by the instantiation by the sessionManagement class. These objects can be seen in the picture bellow, (session, txtUsername, txtPassword and btnLogin).



```
        session = new SessionManagement(getApplicationContext());
        session.clearData();
        txtUsername = (EditText) findViewById(R.id.txtUsername);
        txtPassword = (EditText) findViewById(R.id.txtPassword);
        btnLogin = (Button) findViewById(R.id.btnLogin);
```

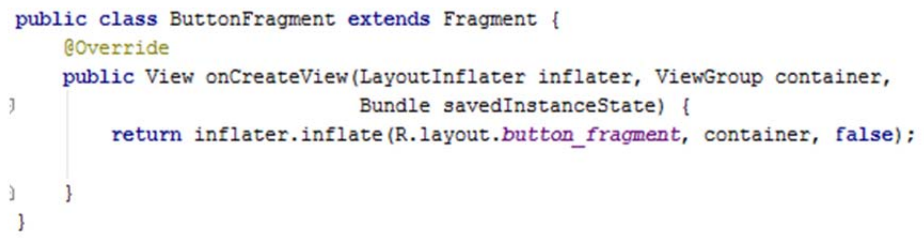
In order to check if the values entered in the username and password fields matches “Troy” and “stuff” respectively, we created an “if-statement.” Before comparing the values, we retrieve the values entered by using a class method, “onClickListener()” to the submit button. After obtaining the values, we create two method string variables where we convert and store the retrieved information. By having a the two string values we are able to check if they exist by checking if the variables are greater than zero. If that is true then we check that username equals “Troy” and password equals “stuff.” If that statement is also true then the program creates an intent that takes the user to the home screen. If either of these if-loops is not true, the user will obtain one of two errors asking them to re-enter the information because it does not match or alerting him or her than neither of the fields can be blank. Lastly in order to check that the user is logged in, two lines of code must be added in the following activities:

```
session = new SessionManagement(getApplicationContext());
session.checkLogin();
```

These lines of code will create a new object, here named “session,” that will call the method checkLogin() on it. This method checks for if the user matches the existing user in sharedPrefereces by calling the method isLoggedIn() to “this.” If that is true it will allow the user to continue to the next views. If he or she is not logged in then checkLogin() would generate a flag that will redirect the user to login, forcing him or her to LogIn.

Button Fragments - Sarah Anderson

The feature that I implemented was fragments for the buttons at the bottom of each screen in the LinguaMeet app. To start, I created a new Java class for the fragment that I named ButtonFragment.java. I extended the Android Fragment Superclass for my ButtonFragment class, which means that ButtonFragment class inherits its states and behaviors from the Fragment class. I override the Fragment class with a public OnCreateView() method that takes three parameters: inflator, container, and savedInstanceState. This method is called so the fragment instantiates the button_fragment.xml file associated with the Fragment. The inflator represents the inflator object used to display the fragment and references the button_fragment.xml file for what should be shown.



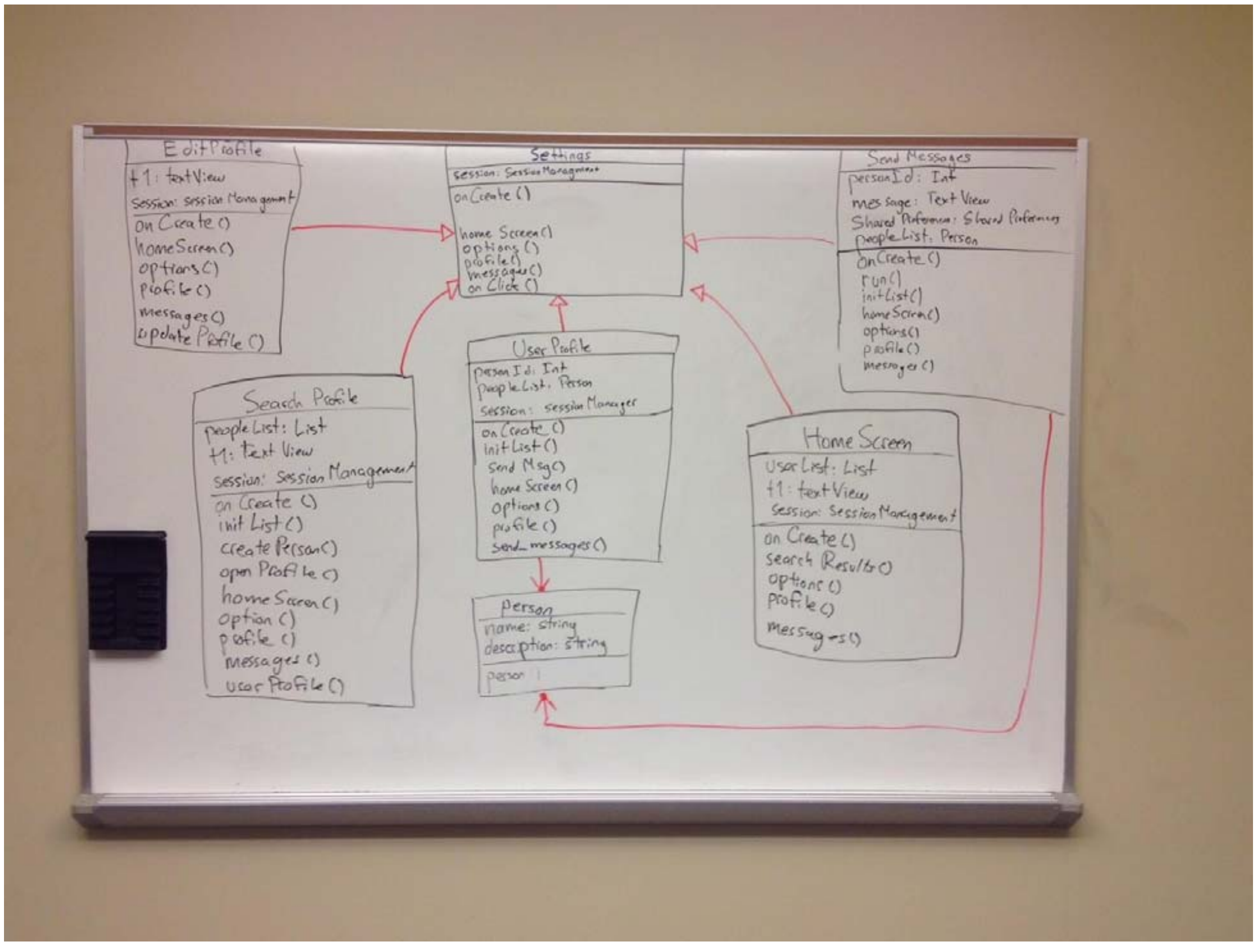
```java
public class ButtonFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.button_fragment, container, false);

    }
}
```

Within the button_fragment.xml file are the respective widgets and attributes for each of the four buttons (settings, homescreen, messages, and profile. Each button is represented by an ImageButton widget, has specific drawables (linked to via the src attribute) and several layout-specific attributes. Additionally, each button is given a unique onClick attribute that names an onClick method called on each individual .java activity when the user clicks the button.

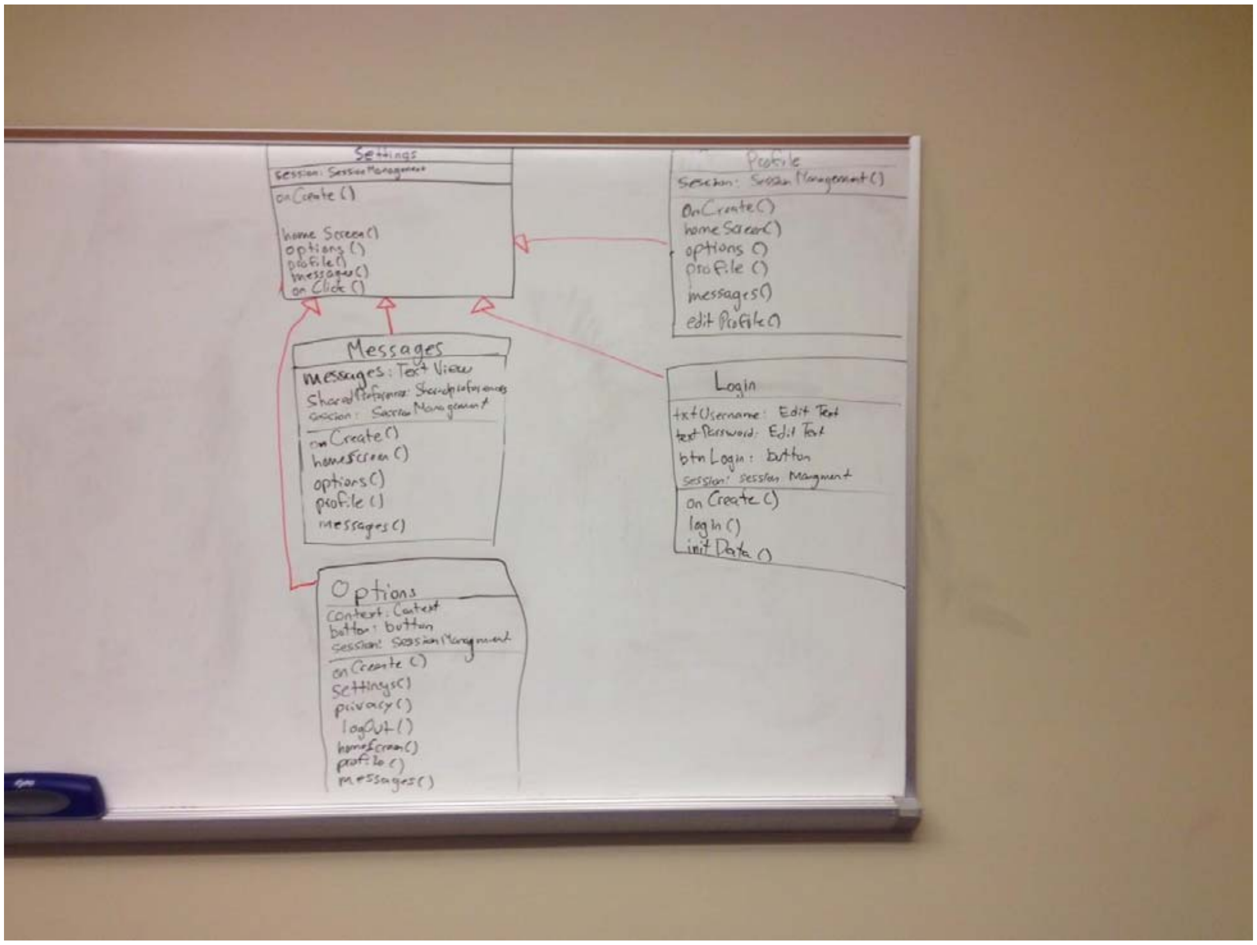
With the above two pieces in place, I applied code to each activity’s .xml layout so the fragment would show up. To do this, I added a fragment under the linear layout section that had a “android:name=” stating where the associated parent ButtonFragment activity could be found in the project ("com.example.troy.linguameethome.ButtonFragment").

Finally, as briefly mentioned above, the button_fragment.xml file allows for an explicit intent to start another activity via the “onClick” method attribute, which is then referenced in each activity’s .java page. For example, the “homeScreen(View view)” represents an onClick() method where a new intent object is formed (homeScreen_intent). This new intent has two parameters: “this,” which represents Context, and “HomeScreen.class,” which is where the homeScreen_intent will be delivered.

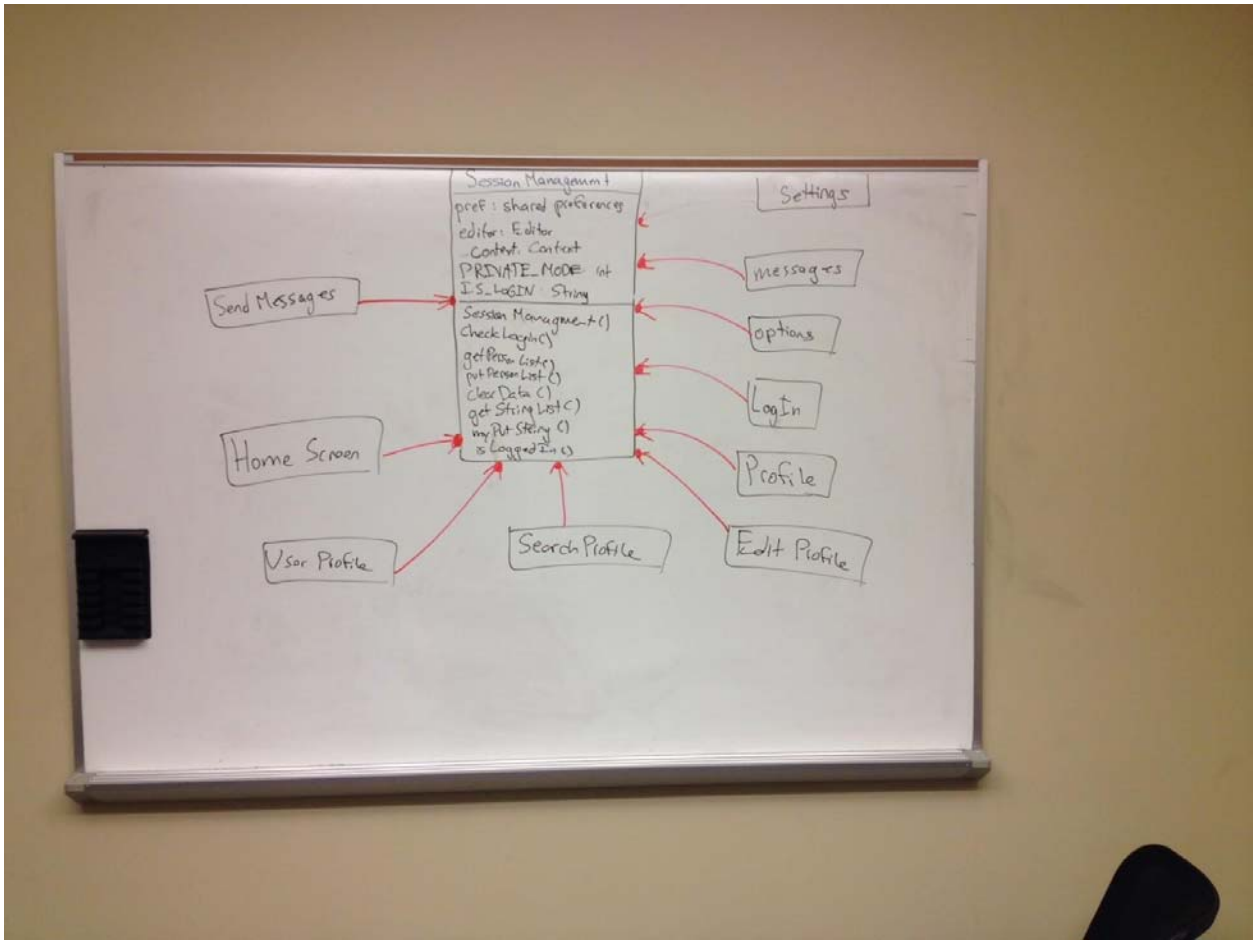
UML Visualization



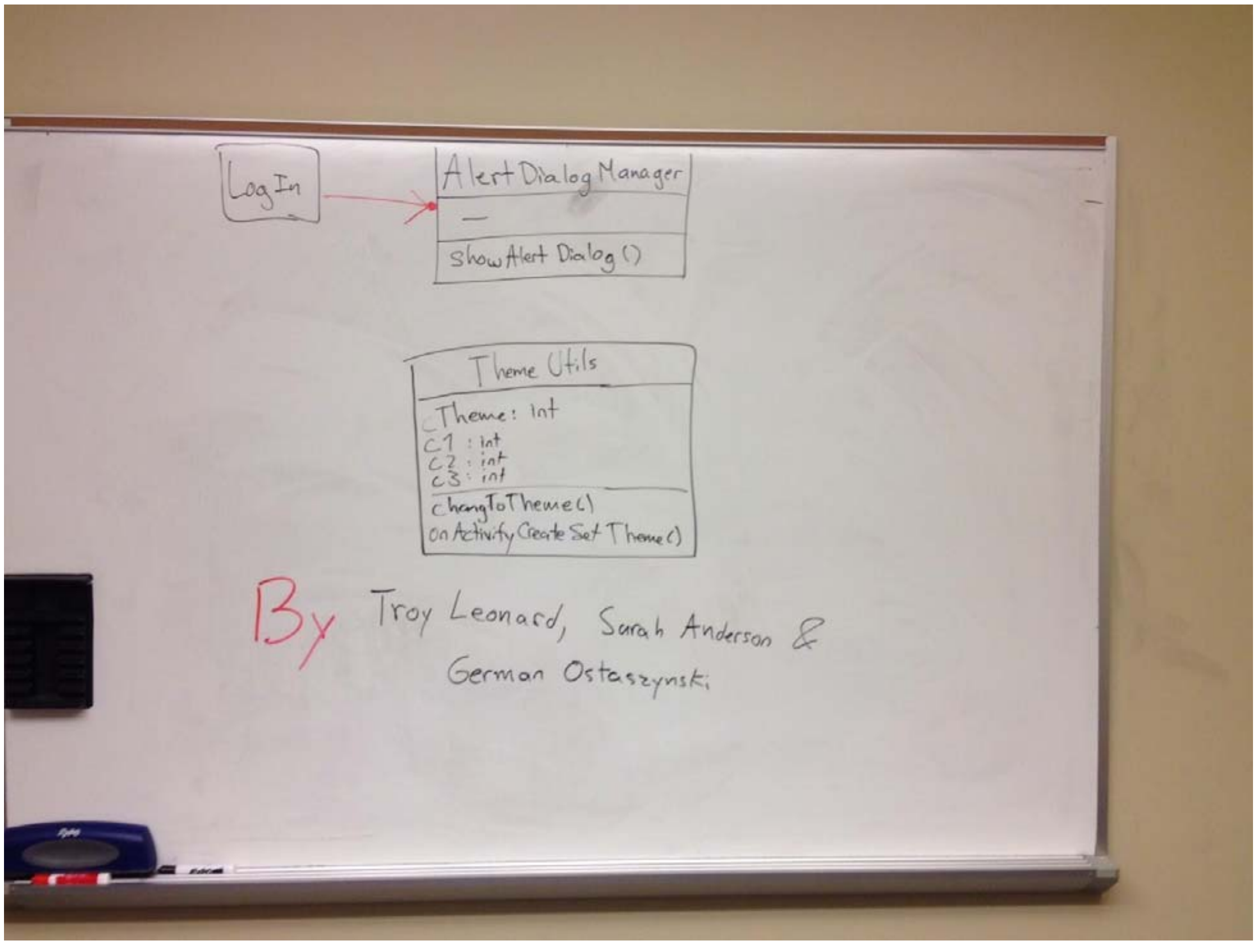
The above board represents the first half of our classes that inherit from the Settings class (see the next slide for the additional classes meeting this criteria).



These are the remaining classes that inherit from the Settings class.



Above is a depiction of each class that is associated with the SessionManagement class. We chose to not write out the full class descriptions for each of the classes above besides SessionManagement because those details could be seen in one of the previous pictures.



For our final picture, we have the AlertDialogManager class and the ThemeUtils class.