CS 495: Capstone Report 4
Tyler Allen
11/3/2014

The goals for this week were to complete user authentication client and server authentication, create default likes, groups, and a test account in the database, and network communication will be complete and structured for additional features. Screens are also to be connected so that users can navigate through the application.

User authentication has been completed both client and server side. A user may enter their credentials at the login screen, press login, and a synchronization request will be sent to the server. The server will then validate that the credentials exist in the database, and the correct password is associated with the correct username. Every request will contain the user's credentials for now. The login screen should only need to be accessed on the first login, or if the user explicitly logs out. Error message now display if the server cannot be contacted, if no credentials are entered, or if the server denies the user's credentials. These currently exist as toasts; they will be changed to dialog boxes. Currently, to create this relationship between the login screen and the network interface, a broadcast receiver is being used.

During implementation, it became apparent that a content provider[1] would be optimal to represent all data received from the server. The content provider will be backed by an SQLite[2][3] database, which will contain a subset of the data on the server. SQLite is fully supported on the Android framework and is recommended by the android documentations for client side databases[4]. This subset will exclusively contain data that the current user needs, as opposed to a replica of the database. This will save a large amount of data needing to be written over the network. This can also be used to validate logins, and will be changed in the future when a content provider has been used. Properly passing data between the `Login` activity, and the `SyncAdapter` created the greatest difficulty during this stage, and it is apparent that in order to meet the large data requirements of this application, the content provider will be necessary. The content provider is currently in research, has been started, and will be complete by the next report.

The user accounts, client side, are now successfully registered with the Android account framework, as opposed to using a default account. The user's credentials are saved locally and privately to be re-entered so that, as mentioned previously, the login screen only needs to be accessed on first login.

[1] "Content Providers | Android Developers." 2009. 3 Nov. 2014 <http://developer.android.com/guide/topics/providers/content-providers.html>
[2] "android.database.sqlite | Android Developers." 2009. 3 Nov. 2014 <http://developer.android.com/reference/android/database/sqlite/package-summary.html>
[3] "SQLite Home Page." 2003. 3 Nov. 2014 <http://www.sqlite.org/>
[4] "Saving Data in SQL Databases | Android Developers." 2012. 3 Nov. 2014 <http://developer.android.com/training/basics/data-storage/databases.html>

Upon successful login, a user can now navigate through the majority of the application per the storyboard. Some of these can not be seen yet. For example, you can not examine an event page because the functionality to populate the event list is not in place yet.

Default groups and likes have been added to the database, as well as a test account. More test accounts will be added to test group s and invites in the future, as well as database continuity.

Hooks are now in place to extend the `SyncAdapter` to support receiving larger data sets from the server. During the next week, Javascript Object Notation (JSON) Support will be added so that data can be passed between client and socket in a more organized way[5]. Support for JSON is implemented directly into Android, and the same library is supported but not included in standard Java[6][7]. Some research does need to be done to validate that JSON can be integrated cleanly, but there are no foreseeable issues currently. JSON would be beneficial because it would provide an organized way to send and retrieve data over the network. JSON utilizes key-value pairs, which would be far simpler and less prone to error than constructing a packet protocol.

Future problems to address include ensuring network-dependent operations are atomic so that the application does not allow users that are not properly authenticated do not move forward into the application and create an invalid display, and that valid operations do not fail during intermittent network conditions.

[5] "JSON." 2003. 3 Nov. 2014 <http://www.json.org/>
[6] "JSONObject | Android Developers." 2009. 3 Nov. 2014 <http://developer.android.com/reference/org/json/JSONObject.html>
[7] "JSON in Java." 2006. 3 Nov. 2014 <http://www.json.org/java/>