CS 495: Capstone Final Report
Tyler Allen
12/12/2014

Overview

This purpose of this capstone is to create a social networking mobile application targeted at the Android platform. This application is currently aimed towards college students. The primary purpose of this application is to assist students in connecting with others by name or mutual interest. Students will build a profile that lists some basic information about the student that other students may view. Students are to have the ability to create groups related to their specific interests. Students are also to be able to create “events” within the application and invite other students to attend these events. Events will contain a name, description, date, time, and location. Students will also be able to create groups for students with mutual interests to join. Groups primarily exist to allow students of mutual interests to find each other, and as a medium through which events can be communicated. Students will also be able to list ‘likes’. Likes appear on a student’s profile for others to see. Students can easily find other students with mutual interests based on their likes and groups. Students will also be able to provide their class schedule to the application. ‘Friends’ of these students will be able to view their schedules. Other students will be able to view a “busy” or “available” status on the student’s profile.

The purpose of this report is to detail the progress made in the development of this application thus far.

Development Progress

Report 1:

Development began with planning and design. It was decided that a back-end server housing a database would be necessary to keep data consistent across multiple clients. UML class diagrams were created to detail the structure of the application and the server. A storyboard was also created to show the usage and outline the appearance of the application. At this time, it was decided that the implementation language would be Java. Java is the supported primary language of the Android platform, and it was a logical decision to design the full application in the same language. The C/C++ and D languages were also considered for this project. It was decided that a C/C++ implementation would increase the development time considerably. I was unable to find a reliable database connector for the D language, as D support is still in development. It was also decided that Java Database Connectivity (JDBC) would be used as the database connector for this project because it is built into the Java Application Programming Interface (API) and is the Java standard. The Google Maps API would be used for mapping, because it is built into Android.

Report 2:

A database schema was developed to map out the server-side database. It was decided that PostgreSQL would be the database management system used to maintain data for this application. The schema was normalized in order to prevent redundancy. The layouts for the application were created. They contained all functional elements. No functionality was implemented at this time except for the input validation available to layout elements. The visual theme was not decided on at this time as the layouts are subject to change, but many

of the pages displayed different themes at this time in order to begin working towards a concrete theme. The maps page was implemented and the appropriate API keys were generated, but maps were not included in the application at this time.

Report 3:

User authentication was partially implemented at this time. Networking and network encryption were also implemented at this time. The development server was also set up at this time and the database was created on the development server. User authentication was implemented using android `Accounts`- a built in utility for managing accounts in the Android framework. Accounts are stored in the database on the server. The `Accounts` API provides the ability to store a user's authentication information on the client side. It also allows the usage of a `SyncService`. `SyncService` allows a multi-threaded approach to synchronizing data with the server, and is recommended by the Android documentation for this purpose. Within the `SyncAdapter` associated with the `SyncService`, the network communication with the server was defined. It was decided that SSL would be used to encrypt data transfer between the client and server. Secure Socket Layer (SSL) and Hypertext Transfer Protocol Secure (HTTPS) are recommended in the Android documentation; SSL was chosen because HTTPS adds an additional layer of data transfer as well as an additional level of complication server-side to interpret the Hypertext Transfer Protocol (HTTP) protocol. The application does not have a web component, so SSL was the obvious choice. If a web application were to be created later, it would be straightforward to implement HTTPS on top of SSL. SSL keystores were created and added to the development server as well as the android application. At this point, the client was capable of sending data to the server, but server-side user authentication

was not implemented. It was decided that a token authorization system may be implemented in the future to improve security.

Report 4:

User authentication was completed during this week, as well as the creation of default groups, likes, and a test account on the development database. Structured Query Language (SQL) statements and JDBC code were created to allow server-side user authentication. Users could now enter valid information and allowed access to the rest of the application. The users information was appropriately stored using `Accounts` client-side; default user information was previously used for account validation client-side. Users could now navigate through most of the application once logged in, but not all functionality was implemented. It was decided that a `ContentProvider` using SQLite would be necessary to store the volume of data to be received from the server for some functions. The SQLite database would contain the relevant subset of the data contained on the server. It was also decided that Javascript Object Notation (JSON) would be better suited for data transmission between the client and server.

Report 5:

JSON would be integrated at this time, and a `ContentProvider` using SQLite3 would be implemented. Users would also be able to view events. JSON support was added to the client side, as it was available through the Android API. The JSON.org JSON library was included to provider server-side JSON support. JSON objects were made the primary medium through which to transmit data through the server at this point. The `ContentProvider` was created, and SQL statements to create the table storing events were implemented. The query,

insertion, and deletion functionality was also added to the `ContentProvider`. Queries were written to retrieve event data from the server, and store it in the `ContentProvider`. Events were then viewable in the events list.

Report 6:

During this time, the search functionality was added, allowing users to search for other users, groups, and likes. This required the addition of multiple tables to the `ContentProvider`, and the creation of multiple SQL queries for the retrieval of this data. The user is able to put in a search term/phrase, and the query will match anything containing the search information. The results are viewable in a scrolling list. Selecting a user or group will allow the user to view the page associated with that user or group. If a user selects their own profile, they will be brought to the edit profile page. Users are now able to register from the registration page, and the new user will be inserted into the database. It was decided that a change in design should be implemented that this point- after registration, users would be require to log in for the first time. This would allow for a much simpler authentication process. In addition, spinning loading bars were implemented during synchronization with the server.

Summary

Communication from the development server and its clients has been implemented and secured. Databases now exist on both the client and server side. The ability to query data from the server has been mostly implemented, and the ability to create accounts has also been implemented. This means that registration, as well as the ability to view a user's own content, and search for other users and their content has been implemented. Future work includes the ability to modify all content created by the user, add friends, view locations, and

the ability to use the location/map services within the application, as well as modify the application settings. In conclusion, the backbone of the application has been created, as well as registration, and the ability to view and query most content.