CS 166 Final Project

Members: Kevin Chang
Yvette Hernandez

Objective: Create an intuitive, online messenger, that allows a user to send messages and search for connection.

Implementation

1. Loading Data: Editing the provided sql file, load_data.sql, we implemented the loading of data into the tables. This included extracting the many tables provided in the .xlsx file provided for the project. First, since the .xlsx file was not compatible with the project, it was saved as merely a .xls file. Then, each table in the .xls file was saved as .csv files.

Technical difficulties

- Since there were missing elements in the tables, implementing foreign key constraints were more of a hassle, since references to users that did not exist were referenced to in the other files.
- Another problem was in the loading of data from the Messages table. Since there were commas in the Messages table, we could not really use the conventional method of using commas as delimiters. Fortunately, openoffice had an option to save xls tables as csv files, as well as the function to set delimiters on the files.

2. Logging on/Off: Logging on and off consists of taking two inputs, one for a username, and one for a password. If both are correct, then the user will be shown a main menu, that provides the many options below.

3. Friendlist: Initially, viewing the friends list, there will be only one choice: the choice to view the user's friends. After that level, the user has a choice to either view friends of friends, or view profiles of their friends.

Technical Difficulties

- It was difficult the deeper level of friends. Essentially, we needed to know how to manage the list of friends.

4. Update Profile: To update a user's profile, a user will have to pick choices from a menu. From this menu, the user will be able to edit the many aspects of a user's profile. From the user's choices, the user will be editing the database.

5. Write a new message: When creating a draft for a new message, the user must first specify the receiver, and the contents of the message. If the user so happens to enter an invalid user, then the function will quit on the user. When the user inputs valid input, then the message will be sent.

Technical Difficulties

- Java is unforgiving when it comes to formatting. In order to provide the timestamp for the message class, we had to create a new Date object, and then get the Date's time with .getTime(). Using this information, we then had to create a new timestamp, which we then had to convert to a string.

6. Send Friend Request: Before a user can find usernames to connect with, the user will first be checked for connections. If a user has no connections, the user will be able to connect with anyone, disregarding existing connections. The user will be able to connect with anyone for five connections. If the user already exists, then the user will be limited to how many connections they can make, being limited to their connections.

Technical Difficulties

- We had difficulty in implementing the deeper level of connections.

7. View Profile: The program will print out information regarding the user, depending on values in the database.

8. View Messages: The user will be presented with a menu of choices. Based on the choice the user takes, the user will see their own sent or received messages.

9. Change Password: The user will change their password to a combination of characters of their own choosing, and then the password database will be updated.

10. Search People: The user will enter a search query, and then the program will look for a user with that name in the database. If no such person exists, then the program will say so.

11. View Requests: The user will be presented with a menu of choices. Based on the choice the user makes, the user will be able to see connection requests, accepted requests, and rejected requests.