Group: Anjali Maheshwari, Alex Wysoczanski, Jack Goettle, Eli Kalish and Carly Ryan
CIS 350: Iteration 3 Report

Contributions:

Alex: I wrote AsyncClient with Jack. When the user enters a username, we attempt to connect to the URL specific to that user. If we are able to successfully connect, we check if the password matches the expected value from the user. If we cannot connect or the passwords do not match, we throw an error in the app. For creating users, I assisted with editing the EJS file for a user so it can handle creating a user based on a request from the app through the '/jazz' route.

Anjali: I wrote the AsyncCreateClient for the create account feature. I attempt to connect to the URL based on the user's username. I open a connection based on the URL. Then,I created a JSON object that is populated with the information of the current user. Using this JSON object, send the JSON object as a String to the server. Lastly, I close the connection.

Carly: I used the singleton pattern to create a new class, CurrentUser.java, that stores the user currently logged into the app. When a user has successfully logged in or registered, I create a User instance which I then pass to my CurrentUser constructor. Anytime we need the current user to populate data in our app, we call getCurrentUser, from which we can grab the necessary fields of our User object. Additionally, I worked with Eli and Anjali in writing AsyncCreateClient so that we can write new users to our database and implemented its functionality to initialize the current user at the end of the CreateAccountActivity and then update it with the survey results after the SurveyActivity.

Eli: I wrote much of the initial code for adding users to the database when they create an account. I wrote the '/jazz' route in index.js which takes in a JSON object containing the fields of a new user. I also added in a new TextView into the createAccount screen in Android that indicates whether or not the input username is already associated with a username in the database (using the getUser method) and if so, prompts the user to enter a new username.

Jack: I wrote AsyncClient.java with Alex to support login functionality in our app. After the user has entered their username and password, I first check to see if the username is valid by searching our user database in mongo. If the user exists, I'll check that the password entered matches the one in the database. If this all matches properly, I pull the user information and populate a singleton instance of our User.java

class which stores all the appropriate information needed for our app. Additionally, I provided field checks for registration methods to make sure a chosen username doesn't already exist and that a user completes all the necessary information when registering. If there is an error in the login or registration process, an AlertDialog message will appear with an appropriate error message.

User Stories:

1. As a user, I want to be able to create a new account. (2)
2. As a user, I want to be able to take the survey and have my results stored in a database. (1)
3. As a user, I want to be able to sign back in using my created information. (2)
4. As a user, I want to be informed if I am attempting to create an account for a username that already exists. (1)
5. As a user, I want my information to be stored throughout my use of the app so that my experience is fully individualized.
6. As an administrator, I want to be able to ensure that two users cannot create an account with the same username. (1)
7. As an administrator, I want to be able to ensure that no one can access the information of an account without the password corresponding to that account. (1)
8. As an administrator, I want to be able to ensure that no one can access the application without signing in to an existing account or creating a new one. (1)
9. As an administrator, I want to ensure that two users cannot be logged into the app on the same device at the same time. (1)