Second Iteration Demo

ACE ASE Team

Group Members:
Michaella Schaszberger (mls2290)
Julia Sheth (jns2157)
Gabi Munoz (gmm2172)
Francesca Callejas (ffc2108)

Details of Demo:

We completed the demo at 7:00 PM on Monday, December 3rd.

Overall, our demo went smoothly. We demo'ed the user stories completed for this iteration (described in detail below), the unit tests, and the coverage report. There were not any major challenges presented; however, we discussed changing the first search box prompt on the search page because "place name" was found to be too ambiguous and confusing to the user. Additionally, we discussed handling the error for a nonexistent city entered into the second search box on the search page. We have already handled the errors for spaces in a city's name, leaving either or both of the search boxes blank when attempting to search a place, and entering special characters into the input. To handle city verification, we would check the input against a list of the 50 largest U.S. cities, but realized during the demo that we should instead verify the city input by checking the API response after every connection during a search. Additionally, we discussed handling the error of a user entering another user's username when adding a place. We will implement error handling that will check that only the user logged in is allowed to add a place to her lists by verifying the username entered and the account logged in to the program. Lastly, we went over code coverage and discussed how we could increase coverage in DataConnection.java, which has the most code coverage so far, and PlaceSearch.java by implementing more test cases and dividing longer methods into smaller ones, which should allow for more tests and increase coverage.

User Stories Demonstrated:

The logging out feature, forget password feature, search location ability, and the ability to add a selected searched location to a user's account were added in addition to the user stories accomplished during the first iteration, which were the "user signing up" story and the "user logging in" story.

The use cases demonstrated in this demo were the "User Creating Account", "User Logging On", "User Forgot Password", "User Searching Places", "User Adding Future Places to Database", and "User Logging Out."

During the demo, we opened MyPlace and demonstrated the "User Creating Account" use case by creating a new account by clicking the sign in button, which redirected the user to the register page, where we entered the username, password, and answers to two security questions and clicked the "register" button. We returned to the main page where we demonstrated the "User Forgot Password" use case. We clicked the "Forgot Password?" link, which directed the user to a pop up asking for the username and answers to the two security questions. When all the fields were entered correctly and the "Retrieve Password" button was clicked, the password for the given username appeared on the screen and we clicked the button to return to the sign in page. We demonstrated the "User Logging On" use case by entering the username and password of the account made earlier in the demo and clicking the "Log In" button, which redirected us to a new page with links to a list for "Places you'll go" and "Places you've been."

We demonstrated the "User Searching Places" use case by clicking the "Places you'll go" list, which redirected the user to another page which currently contains an empty chart with information about the places in that list, such as "Place Name", "Address", and "Comment." In order to search a place, we demonstrated how a user has to press the "+" button, which redirected the user to a "Locate Page" page. We entered "pizza" in the "place name" text box and "chicago" in the "city" textbox and pressed search, which retrieved the results of the search using the Foursquare API and displayed the results in a drop down list called "Potential Matches."

To demonstrates the "User Adding Future Places to Database" use case we had to have the user enter her username after she selected a location from the list, such that the location information would be stored in the correct list and with the correct username. The user then clicked "Add Place" and the location information was added to the database. Since the results of the places table does not appear on the page yet, we proved that the locations were being added to the database by making a SQL query displaying all the results of the places table. As discussed, the error handling of this use case will involve preventing any user from typing in another user's username and

adding a place to another user's lists. Additionally, after the 1st iteration and meeting with our IA, we realized it was necessary to use two tables in the database: one for account information and one for places information, both of which have the "username" field. We implemented this change in realizing that the account information will be more static than the places information, some of which could be deleted or edited by the user.

Lastly, to demonstrate the "User Logging Out" use case, we clicked the "logout" button on the page displaying the two lists, which was where the user was directed after adding a location to her list. Once clicked, the window displaying the program closed. Additionally, our conditions for satisfaction for the logout user story changed and now the user is no longer prompted a line "You have logged out successfully" when closing the window. This prompt only appears when the user clicks the logout button, which is on almost every screen.

Brief Overview of our CI Mechanisms:

TravisCI utilized our Maven build to notify us if there was a build failure any time that we pushed code to our github repository. TravisCI does so by running the test cases that our Maven build runs.

For this iteration we created unit tests to test all the new interactions that we implemented in this iteration. We created unit tests to isolate and test the proper implementation of our Forget Password functionality, Search Place functionality, Add Location functionality, and Delete Location functionality. We used a tool called CodeCov to generate a coverage report. This visual tool provides an interface that shows how many and which lines of code are being tested by our unit tests after every push to GitHub.

Our testing allowed for 86.7% coverage of our interactions between the database. We also have 15.22% coverage in the interactions between the user and the backend before sending any API request. This occurs in the PlaceSearch class. We discussed ways to increase coverage in both of these classes. We have not yet begun testing on any interactions between the user and the GUI. This is because JavaFX (which we used for the interface) does not have many ways to test itself. Because of this, our overall coverage was 20%.

GitHub Repository:

The link to our github repository is: https://github.com/francescacallejas/MyPlace_ASE
The link to our coverage report is: https://codecov.io/gh/francescacallejas/MyPlace_ASE/tree/master/MyWorld/src/application

Everything that is in these repositories was shown in this demo.