ACE ASE Second Iteration:

MyPlace

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

During this iteration, we focused on expanding and hardening our MVPs, such that MyPlace has more usability. We accomplished adding a logging out feature, forget password feature, search location ability, and the ability to add a selected searched location to a user's account (by adding it under her username in the places table in the database with a tag to denote which tab it is being added to -- "Places you'll go" or "Places you've been"). Additionally, we added a commentary functionality in which users have the ability to add commentary to the locations that they are adding to their lists. These user stories 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. After meeting with our IA after the first iteration, we decided on creating two tables for the data in this program for the second iteration (as in the first iteration we only had one table): one table called accounts for static data, including username, password and security question answers, and a second table called places, which includes username, list (places visited and future places), name, and address.

Use Cases:

1. **Title:** User Creating Account
    a. **Actor(s):** MyPlace user
    b. **Description:** User creates an account when the MyPlace program opens. Creating this account will include entering a username, password, and providing answers to two predetermined security questions.
    c. **Basic Flow:** User clicks the sign up button on main page. User enters username and password into the text fields "username" and "password". She selects the first security question and enters the response on the "question #1" textfield, then selects the second security question and enters the response on the "question #2 field." She clicks the "register" button and the information entered in the textfield are added as an account to the MyPlace database.
    d. **Alternate Flow:** User clicks the sign up button on main page. User clicks the "register" button without entering any characters into the text fields or entering values into some of the text fields. User is redirected to another page stating "No

fields can be left blank!" User clicks "try again" button and is redirected back to the "register" page containing all of the empty text fields. User clicks "Back to sign in" button and main page that appears when program starts appears on screen.

2. **Title:** User Logging On
    a. **Actor(s):** MyPlace user
    b. **Description:** User logs into the application so that she can add places to her account (and see places that she has already added to her account, although this part of the functionality has not yet been added).
    c. **Basic Flow:** From first page when app opens, the user enters a valid username and password and then clicks login. Once she clicks login, a page opens where she can choose to look at the tab with the places she has been to or the tab with the places she wants to visit.
    d. **Alternate Flow:** From first page when app opens, the user does not enter anything into either the username or password text fields and then clicks login or the user enters an invalid username or password into the text fields. In either of these cases, when the user clicks login, the app opens a page that says "Wrong Password or Username". The user can then click the "Try Again" button, which will take her back to the first page of the app where she can try to login again.
3. **Title:** User Forgot Password
    a. **Actor(s):** MyPlace user
    b. **Description:** User can retrieve her password based on her username and security question answers in case she forgets her password.
    c. **Basic Flow:** User opens the program and the main page of MyPlace appears. User clicks the "Forgot Password?" button, which redirects the user to the "Forgot Password" page. User enters her username in the "userName" text field, clicks the "Security Question #1" drop down menu and selects the question. In the first "answer" text box, the user enters the answer. User clicks the "Security Question #2" drop down menu and enters answer into the "answer" text box. User clicks "Retrieve Password" button and the password of that user's account is displayed as a text in the format of "Password: 'password'". User clicks "Back" button and is redirected to the main page of MyPlace.
4. **Title**: User Searching Places
    a. **Actor(s):** MyPlace user
    b. **Description:** User can search for a certain place to add by specifying its name and city, thus allowing her to add the place (name and address) to her account.
    c. **Basic Flow:** User opens the program and logs in. Then, the user either clicks on the "Places you'll go" or "Places you've been" tab. Once in either of those tabs, the user clicks on the "+" button in the top right corner. This opens a page where

the user enters the name of the place she's looking for and the city into the specified text fields. She then clicks "Search". Then, there is a drop-down menu that shows all of the results that the FourSquare API returned for her query. She can then choose one from this drop-down menu to add to her account.

5. **Title**: User Adding Future Places to Database
    a. **Actor(s):** MyPlace user
    b. **Description:** User can add places to a given tab in her account so that they are saved under her username in the places table of the myplace.db database.
    c. **Basic Flow:** User opens the program and logs in. Then, the user either clicks on the "Places you'll go" or "Places you've been" tab. The user then clicks on the "+" button and enters the parameters she wants into the text fields and clicks "Search". Then, she clicks on the drop down and menu and selects the place that she wants to add. She then adds her username into the username text field above the drop down. After that she clicks "Add place" which adds this place to the places table in the myplace.db database under her username so that it is associated with her account.
6. **Title:** User Logging Out
    a. **Actor(s):** MyPlace user
    b. **Description:** User is able to logout of account which terminates the use of the application.
    c. **Basic Flow:** User opens the program and logs in. Then the user clicks "Logout" which logs her out of her account and terminates the program.
    d. **Alternate Flow:** User opens the program and logs in. Then the user clicks on either the "Places You'll Go" or "Places You've Been" tab. From either of those tabs, the user clicks "Logout" which logs her out of her account and terminates the program.
    e. **Alternate Flow:** User opens the program and logs in. Then the user clicks on either the "Places You'll Go" or "Places You've Been" tab. From either of those tabs the user clicks the "+" button in the top right hand corner. Then the user clicks "Logout" which logs her out of her account and terminates the program.
7. **Title**: User Adding Commentary
    a. **Actors:** MyPlace User
    b. **Description**: User is able to add commentary to the location they are adding to either of their lists.
    c. **Basic Flow:** User is searching for a location to add to a given list. User adds commentary about the place he or she is looking for.
    d. **Alternate Flow:** Use decides not to add commentary to a given location. He or she can still add the given place to his or her list.
8. **Title:** User Views List of Places User Has Been

    a. **Actors:** MyPlace User
    b. **Description:** User is able to view the list of places that have been added to the list of places a user has been.
    c. **Basic Flow:** User is on the page for the list of places a user has been. User clicks the "See List" button and the list of places appears.
9. **Title:** User Views List of Places User Will Go
    a. **Actors:** MyPlace User
    b. **Description:** User is able to view the list of places that have been added to the list of places a user will go.
    c. **Basic Flow:** User is on the page for the list of places a user will go. User clicks the "See List" button and the list of places appears.

Test Plan:

For this iteration, we focused on pining locations that a user has been to or would like to visit. For this reason, we have focused our testing on the add location, delete location, and search location functionality.

Below we have outlined the equivalence classes for each function that we are testing. Successful tests should result in a change to the list being displayed to the user by updating the GUI. However, we have focused our testing on ensuring the backend is working as expected so that will not be a part of our test suite.

Search Location

The user has the opportunity to search for a place in order to add it to their list. We have limited the searchable locations to the top 50 cities in the US and created a database containing these cities. We have done so by creating a database with these cities. This is our criteria for a "valid city" in this iteration. We test this functionality using the following equivalence classes:

1. The user attempts to search for a location using valid criteria. The result is that the location is determined . We will be testing the following conditions to ensure that that this functions properly for all edge cases:
    a. Valid search criteria and valid single-word city
        i. "Pizza", "Chicago"
    b. Empty search criteria and valid city
        i. "", "Chicago
    c. Valid search criteria and valid city that contain a space in their values
        i. "Pizza place", "New York"
    d. Valid single-word criteria and valid city that contains a space in it
        i. "pizza", "New York"

2. ~~The user attempts to search for a location using invalid criteria. These tests should result in an error message and nothing being added to the database. We will be testing the following cases to ensure that invalid inputs are not added to the database:~~
    a. ~~Valid search criteria and invalid city (one that is not in the database we have created of valid cities)~~ The invalid city is determined by the API, and if the city is not found by the API, then an error is thrown and the program advises the user to input valid data (error statement printed to console). Since this isn't determined unless an API call has been made and an HTTP request has been sent, we have omitted this from unit testing.
3. The user leaves at least one text field blank. There are two text fields for every search, name and city. If the user enters a valid name and city, the result is the user is able to click search and results will appear. We will be testing the following conditions to ensure that that this functions properly for all edge cases:
    a. Valid name and city entered:
        i. name: "pizza" , city: "chicago"
    b. Valid name and blank text field for city:
        i. name: "pizza" , city: ""
    c. Invalid name and blank text field for city:
        i. name: "pi@zz#a" , city: ""
    d. Blank text field for name and valid city:
        i. name: "" , city: "new york"
    e. Blank text field for name and invalid city:
        i. name: "" , city: "new!! york!!*"
    f. Blank text field for name and city:
        i. name: "", city: ""

Add Location

The user has the opportunity to add a location to their list if they choose to. We test this function using the following equivalence class:

1. We are currently testing to ensure that the addLocation function is working as expected.
    a. Add a dummy place into the database and ensure it actually appears in database

Delete Location

The user has the opportunity to delete a location from their list if they choose to. We test this function using the following equivalence class:

2. We are currently testing to ensure that the deleteLocation function is working as expected.
    a. Add a dummy place into the database, delete it, and ensure that it is no longer in the database

Forgot Password

The user has the opportunity to retrieve their password if they forgot it. We test this functionality using the following equivalence classes:

1. The user attempts to retrieve their password for a valid account.
    a. Valid username, valid password, valid, security question 1, valid security question 2
2. The user attempt to retrieve the password for an invalid account
    a. Invalid username, invalid password, invalid security question, invalid security question
    b. Valid username, invalid password, invalid security question, invalid security question
    c. Invalid username, valid password (one that exists in database), invalid security question 1, invalid security question 2
    d. Invalid username, invalid password, valid security question 1, invalid security question 2
    e. Invalid username, invalid password, invalid security question 1, valid security question 2
    f. User leaves a single field blank

User Adding Commentary

The user has the opportunity to add commentary on a specific location that he or she wants to add to his or her list. We tested this functionality using the following equivalence classes:

1. Adding commentary to a location
    a. Ensure that it enters the database
2. Not adding commentary to a location
    a. Ensure that nothing is added to the database.

User Loading Commentary

The comment for a specific location in a list can be loaded. This comment does not have any restrictions on the characters that can be entered; however, the GUI catches the number of characters a user is allowed to type for one location added. We are currently testing that loadComment functions as expected by:

1. Loading a comment that contains characters
    a. Ensure that it returns the entered character
2. Loading a comment without any characters
    a. Ensure that the comment returned is an empty string

For the feature to view the information for a selected location (place name, address, and comment) on a user's list, we decided against unit testing because this feature is mainly involved with the GUI.

Code Coverage

We have used JUnit to run our test suites. We used cobertura to generate a coverage report of how much of our code our testing covers. Although right now the numbers are low, it is because most of our classes are Controllers for FXML files that are not being tested since they are part of the GUI.

GitHub Repository:
https://github.com/francescacallejas/MyPlace_ASE

Location of Test Suite in GitHub Repository:
https://github.com/francescacallejas/MyPlace_ASE/tree/master/MyWorld/src/test

CodeCoverage Report:
https://codecov.io/gh/francescacallejas/MyPlace_ASE