Week 11 - Testing

Team Number: 17-03

Feature 1: Login Page: Password needs to be 8 characters minimum - Validate the password

Test cases:

- Test for character length to ensure it is at least 8
- When a user is creating a password and does not have at least 8 characters, 1 special character, or 1 capital letter, should ask the user to put in a different password.
- Test for special characters, should have at least 1 for security purposes
- Test for capital letters, should have at least 1 for security purposes
- When given the wrong password, a message is sent saying “password is incorrect”
- Test for spaces in the password. Should not have any.
- “Remember password” features should work accordingly if you ask the machine to save it for you..

Test plans:

- We will create a login page and test the different edge cases. User should be able to login and if the password is wrong, it should notify the user. Will create multiple accounts and test to see if this feature is working correctly.

Test Environment:

- This will be performed on our local machines first and tested to make sure it is working properly.

Test Results:

- We expect the result of the login page to either have the user login successfully to their respective profile page, or to notify the user of an incorrect login and to retry at the login screen.

User Acceptance Testers:

- While in the local environment, the primary testers will be the developers. When we move our application to the server, testers will likely be CU Boulder students, as they frequent campus and will have access to the application.

Feature 2: Add Meal: When the user clicks on the Add Meal option, a modal should pop up with options to enter meal or information or select from an existing meal.

Test Cases:

- A meal name should be provided, either manually entered or selected from the library.

- Optionally, there should be nutrition information (calories, protein, sodium, fiber) that may be entered.
- As the user begins typing, suggestions from the library should appear, if the input matches something that exists in the library.
- When the user hits submit, the meal should appear in the meal log.
- If the user selected ‘Save to library’ (in the event the meal that they are entering did not already exist in the library), it should be saved to the library.
- If the user changes their mind about adding a meal, there should be a cancel button, that will close the modal and leave the library and day unchanged.

Test Data:

- Test data will be user input or input from the library depending on the user’s selection.

Test environment:

- Initial testing environment will be local machines; subsequent user testing will occur on the server.

Test results:

- User enters the meal name. If the user selects from the library, the details are retrieved from the library database and shown in the input fields. If the user inputs a new name, the user (optionally) enters nutrition information and selects whether or not to save the meal to the library if not already saved. The user hits submit, and the meal displays as part of the log.

User Acceptance Testers:

- Preliminary testing will be done by the developer; subsequent user testing will be conducted with CU students or similar persons which have access to the app.

Feature 3: Request for displaying meal library

Test Cases:

- When given a username/userID, the backend returns an accurate list of recipes associated with the user through the Library table
- The returned data should populate a list of recipes viewable by the user in the Library tab

Test data:

- We will create model data and insert it into the database. This data will include a few instances of the user, recipe, and library table. The users will have different recipe libraries, which will be represented in the library table. When creating our test data, we will keep track of what users have what recipes in their libraries in order to verify our test results.

Test environment:

- Initially, testing will be performed on our local machines. Eventually, when our application is deployed to the server, testing will be done on the server to ensure everything is working properly

Test results:

- We expect the request to return a JSON object containing the recipes associated with the given user. We will be able to verify that the list of recipes includes the proper data as we will know what to expect after inserting our test data.

User acceptance testers:

- While in the local environment, the primary testers will be the developers. When we move our application to the server, testers will likely be CU Boulder students, as they frequent campus and will have access to the application.

Feature 4: Calendar Page Meal Log Display (Popup): When the user clicks on a weekday, the meal log for the day should display.

Test Cases:

- If the day has meal log information, it should be displayed. If there is no information for that day, it should indicate that by displaying a message (“No meals logged for this day.”)
- In the display for a day there should be an add meals button.
- When the user is finished adding/editing, there should be an option to cancel or save the changes. Upon hitting this, the user should be returned to the updated calendar page.

Test Data:

- The data will be retrieved from the chosen day’s logged meals in the database for display (if there is data).
- The data will be input by the user or retrieved from the recipe library if a new meal is added (see add meal functionality).

Test Environment:

- Initial testing environment will be local machines; subsequent user testing will occur on the server.

Test Results:

- User will view the meals logged for the selected day, assuming they exist, if not it will inform the user that the day has none logged.
- If the user chooses to delete a meal from the log, the meal will be removed from the display.
- If the user chooses to add a meal, then the add meal modal will pop up and they will interact with the modal accordingly, and the meal will be added to the display log.

- When the user is done adding or deleting from the log, then they will select ‘Done’ to confirm their changes, otherwise cancel to undo their alterations and be returned to the calendar page.

User Acceptance Testers:

- Preliminary testing will be done by the developer; subsequent user testing will be conducted with CU students or similar persons which have access to the app.

Feature 5: Profile Page: User should have the option to have a public vs private profile page

Test Cases:

- Users should have the ability to update their profile pages whenever they’d like. Should have an edit profile feature.
- If the user asks to edit their profile but doesn’t make any changes and clicks save, a message will appear saying no edits were made.
- If the user wants to have a partially public profile but also private features (such as the statistics that count your nutrition), they should be able to perform this and allow only part of the profile to be public.
- Users should have the ability to friend other profiles.
- Profile page should have an about me, profile picture, and a list of all the statistics calculated from the database.
- If a different profile accesses a private account, all the user should see is a screen that says “Sorry, this account is private”.

Test Data:

- The data will be provided from whatever the user decides to put on his profile page.
- Will test with 2 different profile pages, one public and one private to see the differences and ensure they are working correctly.

Test Environment:

- Initial testing will be done on our local machines, and then will be added to the server.

Test results:

- We expect that the user will have the freedom to make their profile either private or public.
- If viewing a private profile, the user should be directed to a screen that says “Sorry, this account is private”.
- If viewing a public profile, the user viewing that persons’ page should have the ability to view all their statistics and descriptions.

User Acceptance Testers:

- While in the local environment, the primary testers will be the developers. When we move our application to the server, testers will likely be CU Boulder students, as they frequent campus and will have access to the application.

Feature 6: 3rd party API recipe book

Test Cases:

- The user should be able to navigate to the relevant page and see a list populated with recipes via 3rd party API
- The user should be able to add a recipe to their library via button
- The user should be able to provide input to the API to return a filtered/specified list of recipes

Test Data:

- The data for this test will consist of hypothetical user input to the API in order to return a more specific list of recipes

Test environment:

- Initially, testing will be performed on our local machines. Eventually, when our application is deployed to the server, testing will be done on the server to ensure everything is working properly

Test results:

- We expect the data returned by the API to be a list of relevant recipes. These recipes should populate on the appropriate page.
- When a recipe is added, two inserts must occur within the database. One in the recipes table with the information regarding the recipe, and the following insert will be in the Library table connecting the user session to the added recipe.

User acceptance testers:

- While in the local environment, the primary testers will be the developers. When we move our application to the server, testers will likely be CU Boulder students, as they frequent campus and will have access to the application.