SW Engineering CSC648/848

“FoodsOnly”

Section 01 Team 06

Issac Moreno (Team Lead)

Noah James Yamsuan (Scrum Master)

Anshav Upadhyay Nepal (Front End Lead)

Terrell Enoru (Back End Lead)

Kayla Young (GitHub Master)

Karl Xavier Layco

Milestone 4

5/06/2024

1) Unit Testing
    I. **Unit Testing URL:**
    https://github.com/CSC-648-SFSU/csc648-01-sp24-team06/tree/feature/application/frontend/Test
    https://github.com/CSC-648-SFSU/csc648-01-sp24-team06/tree/feature/application/backend/backend-test
        a. Description of 5 P1 features and GitHub URL of test cases directory.
            1. **Home**
            Our home page contains a welcoming array of information that is immediately available to the user. We show current recipe images to entice the user. Further, our home page also features a trending section, which is based on our "like" feature.
            2. **Sign Up**
            Our signing-up feature is a main function of our application because upon signing up, users can create their profiles and customize their preferences, skill levels, and allergies.
            3. **View Post**
            Viewing posts is another major feature of our application. While viewing a post, users can like and bookmark these posts as well. Viewing posts on our website is crucial because we allow users to access our recipes, as well as third-party API recipes.
            4. **Search**
            Our search function has multiple functionalities. The primary focus of searching is to search our database for recipes. The second feature is profile searching. Upon a "@" prefix, you can search our application for other users. The third function of searching is search with AI. Our search with AI function employs an AI to personally recommend recipes to users by searching an extensive third-party recipe database.
            5. **Login**
            Our login feature is very important for protecting our application from unauthorized use. We make sure most of our application features are restricted to users who create accounts. By doing so, we can ensure that users have the best personal experience on our website.
    II. Integration Testing (at least 10 P1 features)
        a. **Description of test cases and test results for each of 10+ P1 features. (*In spreadsheet link*)**
            1. **Home**

Our home page unit test tests our trending recipes function on our landing page. Our first test case is displaying recipes when we have trending recipes to display. Our second test case is for no recipes available, so our error handling proves it works as well. The final test case is an internal error, that should display an error message and prevent any further action.

2. **Sign Up**
   Our unit test for signing up primarily tests different cases for sending different types in the form. The first test case is to ensure that the page loads at all. The second test case sends clean data to the registration form and makes sure the profile is set up correctly as well. In the future, we will have other test cases that send dirty data through the form and will only accept clean data.
3. **View Post**
   For testing view posts, we ensure that recipes are displaying the right data. Our first test case ensures that the page renders without issues. The second case checks that all expected data is on the page and accessible to the user. The final test case is to check for internal errors and prevent further action from the user.
4. **Search**
   To test our search, we want to ensure that queries pull up correct results. Our first test case ensures the page renders correctly. The second test case is for ensuring that relevant results are displayed (as expected for search). The third test case tests when there are no results, we should see a message explaining why. The fourth test case tests our function to search profiles. On success, we see the profile, and on failure, we get the same message from case 3.
5. **Bookmark**
   Our bookmark unit test is for toggling bookmarks. Upon bookmarking a recipe, the button should be pushed and the page should be updated. On failure, we should display a message and expect error results.
6. **Login**
   The login unit test is a backend test for testing our connection to the database with clean/dirty data. The first test case is correct login, which should allow a user to log in without difficulty. The second test case is for logging in with incorrect password credentials. The case prevents anyone with an incorrect password from accessing accounts that aren't theirs. The third case is for an invalid username, which should not allow the user to log in.

7. **Logout**
   The logout unit test is for testing the logout function for a user that is logged in. The first test logs in a test user and creates a session and cookie, then executes the logout function and tests that the session and cookie are cleared, returning an error if unsuccessful. The second test ensures the session is cleared by requesting session info, if it could not be retrieved it means the logout was successful and cleared cookie and session data. The third case tests that logout cannot be executed unless a user is logged in. The expected result is a 401 unauthorized response.
8. **Like**
   The like unit test ensures the like functionality works on a recipe. First a test recipe entry is created, and deleted after the tests are complete. The first test logs in a test user, and adds a like to the recipe from the user. The second test makes the user toggle like again, making sure the like is removed. The third tests for liking a recipe that does not exist and ensures an error is returned.

Description of functional and statement coverage.
Our functional coverage is minimal, but it covers most cases. Our project design groups necessary functions together in files, so testing a small portion confirms that the rest works as well. For each of our unit tests, we test most of the functions that are called by users. Our statement coverage is fairly extensive. We attempt to test the entire React pages and service files that support our application. For each unit test, the execution statements are ensured that they output correct results.

b. **URL of the bug tracker system which records your failed integration test results (and integration testing).**
   https://docs.google.com/spreadsheets/d/1kvtcKxjpf2zI9M-U_Dn1UPadLuWeT94tG37eUf3SX9Y/edit?usp=sharing
c. **Description which analyzes coverage.**
   Our coverage outlines testing strategies with minimal to full functional coverage, but extensive statement coverage. We focus on grouping functions in files for efficient testing and testing most user-called functions through unit tests. Further, we emphasize verifying correct outputs of execution statements and extending testing to entire React pages and service files, ensuring thorough coverage across frontend and backend components.

2) Coding Practices

I. Coding Style

a. **Please describe what coding style you chose and how to enforce it by the tool or plug-in.**
   We chose to use *Prettier* as our coding style for our team. We are enforcing this code style through plugins on VsCode and installing dependencies in our package.json.
b. **Please list source files related to 5 P1 features which demonstrate your coding style.**

   - Home:
     csc648-01-sp24-team06/application/frontend/src/Home.js
   - Sign-Up:
     csc648-01-sp24-team06/application/frontend/src/Signup/Signup.js
   - View Post/Bookmark:
     csc648-01-sp24-team06/application/frontend/src/Post/ViewPost.js
   - Search:
     csc648-01-sp24-team06/application/frontend/src/Nav_Footer/Search.js
   - Login:
     csc648-01-sp24-team06/application/frontend/src/Login/loginService.js