CSC 648-03 Fall 2019 Software Engineering

Milestone 4:
Testing

Team 103 - *Fridge Tracker 9000*

Due: December 3, 2019

Joshua Lizak **Project Lead** *jlizak@mail.sfsu.edu*	Nina Mir **Scrum Master / Front End Developer**	Abhishek Mane **Back End Lead**
Wilson Xie **Full Stack Developer**	Douglas Hebel **Front End Lead**	Gangzhaorige Li **GitHub Master / Back End Developer**

Date Submitted For Review:	Date Revised After Feedback:
December 3, 2019	December 19, 2019

---

1. Product Summary

- **Name**: Fridge App 9000
- **Major Committed Functions**
    - User Login
        - Login of existing users
            - Users contain accessible fridges, basic user information, and personal notes
        - Creation of new users
            - Username
            - Email
            - Full Name
        - Logout

    - Multiple Owned/Friended Fridge Management
        - Manual item addition/deletion
        - View item details: Icon, Calories, Added Date, Expiration Date, Adder
        - Easy item deletion
        - Item expiration color coding
        - Simple switching between owned fridges
        - Simple switching between friended fridges
        - Setting a primary fridge to be loaded on new log-on
        - Changing name of owned fridge(s)
        - Deletion of owned fridge(s)
        - Adding/Removing friends to owned refrigerators
    - Grocery List
        - Individual per refrigerator
        - Automatically tracked items based on user selection (set by the owner)
        - Manual list of items that can be added to by anyone with access to the refrigerator
        - Easy search for items within manual item selection window
    - Receipt Upload
        - Automatic text recognition and item discovery
        - Manual confirmation of discovered receipt items
        - Addition of selected receipt items into the currently selected refrigerator
    - Recipe Discovery
        - Select items from the currently selected refrigerator to find recipes from
        - Use of recipe API to return a list of available recipes
            - Recipes can be clicked on to view the original webpage of the recipe
        - Discovered recipes can be saved to the user's profile for easy viewing at a later time
    - Profile Page
        - Display list of all currently owned refrigerators
        - Display list of all refrigerators you have been added to as a friend
        - Personal Notepad
            - User editable text field which is saved to a user's profile and can be viewed anytime the user is logged in
- **Unique Feature:** In an effort to ease the ability of a fridge owner to share their refrigerator with those that they are connected to, we developed friended fridges. The concept surrounds the idea that a fridge has a singular owner and anyone else who may need access to this fridge can be granted the capability. This singular owner is able to provide access to their fridge to others by adding these other people as 'Friends' of the

refrigerator. Working in this manner, a refrigerator is easily seen as an entity with one administrator and multiple accessors.

- **Product URL:** fridgeapp9000.com
    - Secondary link: http://ec2-34-224-23-175.compute-1.amazonaws.com:8000
    - Secondary link may change. Primary is preferred.

2. QA Test Plan

Unit Test

A. Features Tested:
    1. Receipt Upload - Upload Mechanism, Text Recognition, Item Selection
    2. Fridge Manipulation - Fridge Creation, Adding/Removing Friends, Ensuring Added Friends Have Access

B. Hardware & Software Setup:
    1. Hardware: iPhone 11, iPhone 7, iPhone 6s, Motorola G7,
    2. Software: Desktop Safari 13.0.3, Desktop Google Chrome 78.0.3904.108, Mobile Safari iOS 13.2.3, Mobile Google Chrome 78.0.3904.84

C. Test Cases:
    1. Receipt Upload and saving to server under general circumstances. This test is to be conducted when there is a user currently logged in. As a non-logged-in user does not have access to the receipt upload mechanism it is only of concern to verify that the basic upload functionality is operating properly when a user is currently logged into the system.
    2. Adding an item to a refrigerator. Unit testing will be conducted on the basis that a user is already logged in and has a refrigerator available. The item will be added to a fridge that is the one that is currently selected and viewable to the user. These refrigerators that the item will be added to are owned and friended in order to provide a wider test situation that may lead to a fault in the item addition system.

D. Test Setup/Analysis:

In order to perform our testing on the above listed functions of our refrigerator manager system, we opted to use the unit tests that are built into the Python/Django system that runs our whole application. Using the tools built into Django, we are able to easily integrate detailed testing parameters into the core functions of our application. As we are using Docker to run our Django server and MySQL database within a virtual machine, we were required to run the tests with

Docker currently running. Running the tests in terminal would result in a pass or fail depending on the conditions set before the tests were run.

E. Bugs Discovered:
    1. The uploading of a receipt to the system when a user does not currently have access to either an owned refrigerator or a friended refrigerator leads to the item detection system returning null values. While this situation will not raise any alarm for the end user, no indication was given to the user that there might have been something that was amiss. In effect, the user just uploaded a receipt and nothing proceeded to happen. In an ideal scenario the user would be informed that they do not currently have any fridges at their repertoire and thus the receipt upload functionality cannot proceed as is intended.
    2. In the situation where a receipt is uploaded that has item content that is not within our hardcoded database of items, the text recognition system will return no items and provide the user with no indication that something is amiss. This may confuse the end user and will have to be addressed in our continued builds.

Integration Test

A. Functional Requirements Tested:
    1. Receipt upload with and without an accessible refrigerator
        a) Text recognition and proper item discovery
        b) Manual user selection of discovered items and addition of selected items into the current refrigerator
    2. Creation of a new refrigerator when a user has an account with no currently accessible fridges (eg. a new user)
    3. Creation of a new refrigerator using the modal in the sidebar. This is to be used when the user already has access to a refrigerator and would like to create a new one.
    4. Addition of a friend to a refrigerator that the user owns. Then switching refrigerators to another fridge owned by a user and attempting to add a friend to this other refrigerator. Check the friend's account to ensure the refrigerator has been added properly.
    5. Removal of a friend from the currently selected refrigerator when the fridge has no friends and removal of a friend when the fridge does have friends
B. Hardware & Software Setup:

1. Hardware: iPhone 11, iPhone 7, iPhone 6s, Motorola G7,
2. Software: Desktop Safari 13.0.3, Desktop Google Chrome 78.0.3904.108, Mobile Safari iOS 13.2.3, Mobile Google Chrome 78.0.3904.84

C. Results:

Case ID	Description	Dates Tested	Test Steps	Results
1	Creating a new fridge as a new registered user.	Dec 3 7:42	Register-Login-CreateFridge	PASS
2	Creating/adding a new fridge when user already have an fridge(2 or more fridge)	Dec 3 7:46	Register-Login-CreateFridge-Addfridge	PASS
3	Adding a friend to a owned fridge	Dec 3 7:52	Create another user.Using the first user add the second user via email. Second user can choose the user1 fridge to view its content	PASS
4	Removing a friend from an owned fridge	Dec 3 7:57	First user goes to edit friend to remove a friend. Check if the second user is still there. Goes back to second user to check if he can see the first user's fridge	FAIL
5	Receipt Upload with a fridge Discovered item	Dec 3 8:01	User with a fridge clicks on add receipt button. Scans the receipt. It shows indeed what it found.	PASS
6	Receipt Upload without a fridge	Dec 3 8:17	Create new user. Clicks on receipt upload. Scans receipt. Discovered items shows up. Can not add the items	FAIL
7	Receipt Upload + Adding Item	Dec 3 9:27	User with fridge clicks on receipt upload button. Check if chosen item is added to the fridge.	PASS

D. Analysis: For Case Id 1 - 3 it worked perfectly fine. For Case id: 4 A bug was discovered that user can not remove the friend from a fridge at all which is not supposed to be happening. Id: 5 Passed with no problem. Id: 6 A user can still click on the upload receipt button and it shows the discovered item but unable to

add them into fridge because there is no fridge at all. The button for adding via receipt should be disabled when there is no fridge selected at all.

E. Bugs: 1. Fridge owner can not remove the friend. 2 Disable button for the user who has no fridge at all. 3. Disable some functionality if user does not have a fridge

3. Beta Test Plan

Test objectives:

- Stalk the application while the tester uses it to collect more information about errors, bugs, and crashes
- Feedback on UI (what needs to improve) from testers
- Is it user friendly?
- Performance of the application (Fast response time)
- Initial thought about the application from testers

Test Plan

- System setup: we create a beta aws server for people to test our mobile web application.
- Starting point: Register-Login
- Intended user: family members, roommates, friends, and sfsu students.
- How to collect user feedback: manual survey printed out or electronic.
- How to automatically collect the behavior of your product: stalk our application while the tester use it.
- URL of the system to be tested: http://fridgeapp9000.com/

4. Code Review

Coding Style: snake_case(underscore). Everyone should request a code review to git master and team leader for approval

Before The following code might look out of control because of the google doc.

```
def getCurrentFridge(self):

       if(self.session['current_fridge_id']):

           current_fridge = Fridge.objects.filter(

               id=self.session['current_fridge_id']).get()
```

```
            return current_fridge

        else:

            return None



    def getCurrentFridgeContent(self):

        fridge_content = FridgeContent.objects.filter(

            Q(fridge_id=self.session['current_fridge_id']))

        return fridge_content



    def getCurrentFridgeContentByExpiration(self):

        fridge_content_expiration = FridgeContent.objects.filter(


Q(fridge_id=self.session['current_fridge_id'])).order_by('expirationdate')

        return fridge_content_expiration



    def changeCurrentFridge(self, fridge_id):

        self.session['current_fridge_id'] = fridge_id

        return self.session['current_fridge_id']



    def setPrimaryFridge(self, fridge_id):
```

```
        user =
User.objects.filter(id=self.session['current_user_id']).get()

        if (fridge_id):

            user.primary_fridge = fridge_id

        else:

            user.primary_fridge = -1

        user.save()
```

After:

```
def get_current_fridge(self):

        if(self.session['current_fridge_id']):

            current_fridge = Fridge.objects.filter(

                id=self.session['current_fridge_id']).get()

            return current_fridge

        else:

            return None



    def get_current_fridgeContent(self):

        fridge_content = FridgeContent.objects.filter(

            Q(fridge_id=self.session['current_fridge_id']))

        return fridge_content
```

```
    def get_current_fridge_content_by_expiration(self):
        fridge_content_expiration = FridgeContent.objects.filter(
Q(fridge_id=self.session['current_fridge_id'])).order_by('expirationdate')
        return fridge_content_expiration

    def change_current_fridge(self, fridge_id):
        self.session['current_fridge_id'] = fridge_id
        return self.session['current_fridge_id']

    def set_primary_fridge(self, fridge_id):
        user =User.objects.filter(id=self.session['current_user_id']).get()
        if (fridge_id):
            user.primary_fridge = fridge_id
        else:
            user.primary_fridge = -1
        user.save()
```

2. Self-Check: Adherence to Original Non-Functional Specs

C - Completed NC - Not Completed CTC - close to complete

- Password encryption C
- Application shall be optimized for mobile browsers. C

- Application shall be developed, tested and deployed using tools and servers reviewed by Class TA in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be reviewed by class TA). C
- Data shall be stored in the team's chosen database technology on the team's deployment server. C
- Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. C
- Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development C
- Site security: basic best practices shall be applied C
- The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2019. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). CTC