SW Engineering CSC 648/848 Fall 2019
Milestone 4

Team 101
“Pantry Raid”

Brian Nguyen - Team Lead - bnguyen18@mail.sfsu.edu
Malik Iscondari - Scrum Master
Vincent Wu - Github Master
Jeffrey Piercy - Backend Lead
Yiyu Zhang - Backend Assistant
JianQiao Xie - Frontend Lead

Version	Date Submitted for Review	Date Revised
1.0	12/3/2019	

Product Summary

Pantry Raid is a smart refrigerator mobile application which allows users to manage, build, and keep track of their refrigerator inventory at all times. Alongside with tracking and managing inventory, Pantry Raid allows its users to upload and search for recipes throughout the application. For the sake of convenience of our customers, Pantry Raid also allows for users to add and edit to a shopping list from their mobile device as they plan their next grocery store trip. A unique feature of our product is our Meal Plan feature which allows users to allocate a set amount of ingredients for a set amount of days in order to give them a more organized. We plan to market this product as a solution which saves time for busy users who are looking for the assistance of technology to ease their lives.

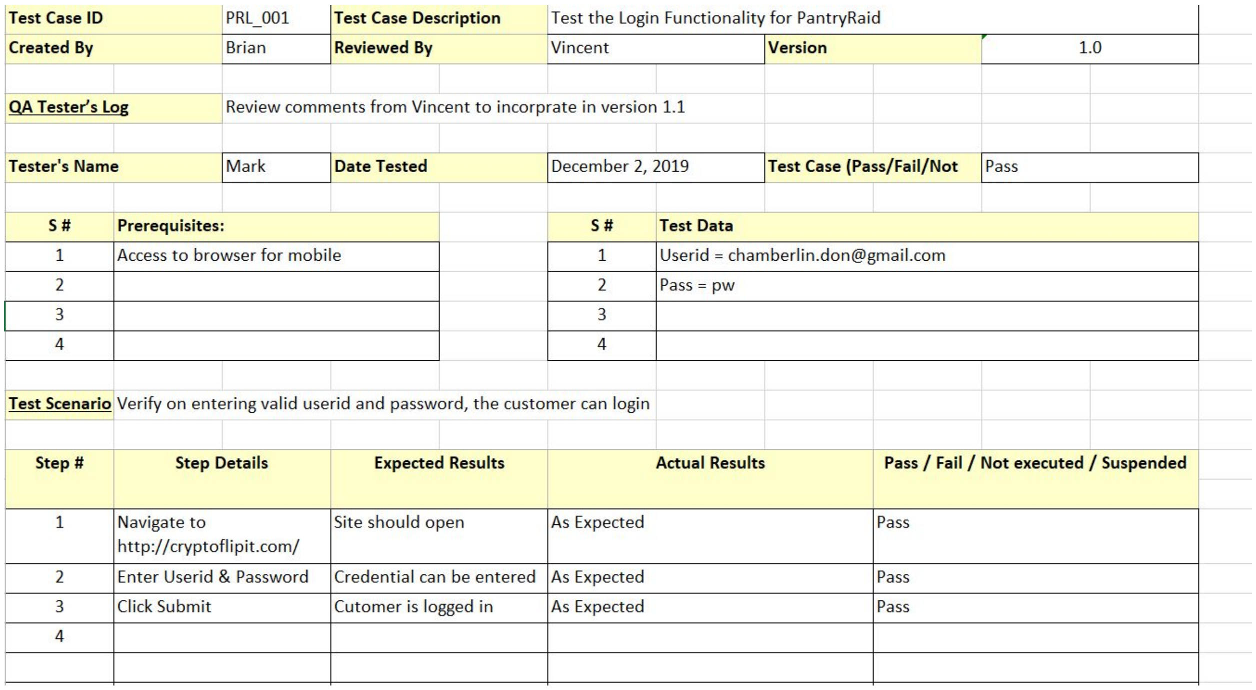
Features List:

- Inventory
    - List Ingredients in inventory
    - Add Ingredients to inventory
    - Remove ingredients from inventory
- Recipe
    - Search for recipes
    - Display recipes for the specific search
    - Add/Upload recipes ingredients to shopping list
- Shopping List
    - Create a shopping list
    - Add ingredients to shopping list with a certain amount of quantities for that specific ingredient
- Unique Feature: Meal Plan
    - Adds how many meals to meal plan
    - Adds recipes to meal plan
    - Generate meal plan for x amount of days from calendar

URL: www.cryptoflipit.com

QA Test plan

a) FUNCTIONS TO UNIT TEST: Upload Recipe, List Inventory
Test tool to test react application will be Jest



Test Case ID	PRL_001	**Test Case Description**	Test the Login Functionality for PantryRaid		
Created By	Brian	**Reviewed By**	Vincent	**Version**	1.0

QA Tester's Log	Review comments from Vincent to incorprate in version 1.1

Tester's Name	Mark	**Date Tested**	December 2, 2019	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**		**S #**	**Test Data**
1	Access to browser for mobile		1	Userid = chamberlin.don@gmail.com
2			2	Pass = pw
3			3	
4			4	

Test Scenario	Verify on entering valid userid and password, the customer can login

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to http://cryptoflipit.com/	Site should open	As Expected	Pass
2	Enter Userid & Password	Credential can be entered	As Expected	Pass
3	Click Submit	Cutomer is logged in	As Expected	Pass
4				

b) Integration Test

Beta Test Plan

a) Test objectives
To gather data on how users interact with our features and functionality to show if our components are working properly like intended. We would like to know what part of the website that users frequent to improve those specific feature qualities. We would also like to gather data on who our audiences are (age group, occupation, etc.)

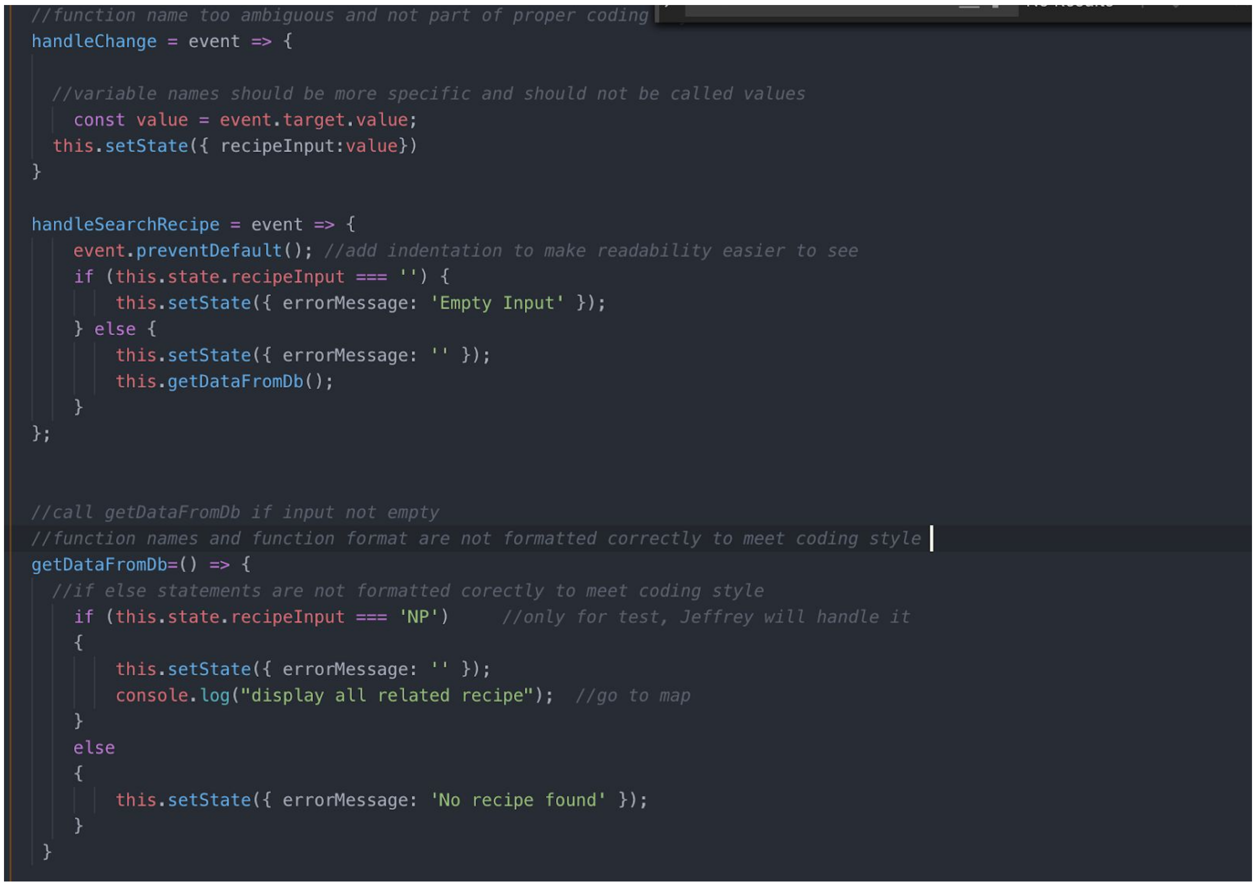
b) Test plan
-System Setup: Need Phone
-Starting point: http://cryptoflipit.com/
-Who is the intended user: A busy worker
-How to collect user feedback: google form survey
-How to automatically collect the behavior of your product (which features are used the most, how many crashes occurred, how long the user spent for each

feature: Hotjar is a data analysis tool that records user's interactions of our features. It displays visual heatmap
-URL of the system to be tested: http://cryptoflipit.com/

Code Review

c) The Javascript coding style that will be enforced on every team member will be Airbnb Style. The Python style will be PEP 8.

d) How To enforce style
Team members should use a style checking tool (e.g. eslint, pycodestyle) before committing code. When reviewing pull requests, the reviewer must ensure that it is in the correct style.



```
//function name too ambiguous and not part of proper coding
handleChange = event => {

  //variable names should be more specific and should not be called values
    const value = event.target.value;
  this.setState({ recipeInput:value})
}

handleSearchRecipe = event => {
    event.preventDefault(); //add indentation to make readability easier to see
    if (this.state.recipeInput === '') {
        this.setState({ errorMessage: 'Empty Input' });
    } else {
        this.setState({ errorMessage: '' });
        this.getDataFromDb();
    }
};


//call getDataFromDb if input not empty
//function names and function format are not formatted correctly to meet coding style
getDataFromDb=() => {
  //if else statements are not formatted corectly to meet coding style
    if (this.state.recipeInput === 'NP')     //only for test, Jeffrey will handle it
    {
        this.setState({ errorMessage: '' });
        console.log("display all related recipe");  //go to map
    }
    else
    {
        this.setState({ errorMessage: 'No recipe found' });
    }
 }
```

1) A Coder submits code to a reviewer
2) Here, a team member evaluates another team member's code through comments.
3) Comments are included inside and outside of the function

4) Once the changes are modified to meet requirements, the programmer resubmits the code to checked and approved

<u>Self-check: Adherence to original non-functional specs</u>

e) Application shall be developed, tested, and deployed using tools and servers reviewed by class TA
   i) ON TRACK
f) Application shall be optimized for mobile browsers
   i) ON TRACK
g) Data shall be stored in team's chosen database technology on team's deployment server
   i) DONE
h) Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users
   i) FIRST PART DONE. SECOND PART ON TRACK
i) Application shall be very easy to use and intuitive
   i) ON TRACK
j) Pay functionality shall not be implemented
   i) DONE
k) Site security: besic best practices shall be applied
   i) DONE
      Passwords are hashed with the PBKDF2-HMAC-SHA256 algorithm using 600,000 rounds. Cookies are signed using the BLAKE2b algorithm. When any protected call is made to the backend, a signed session cookie is sent and verified before responding.
l) Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
   i) ON TRACK
m) 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.
   i) ON TRACK