SW Engineering CSC648/848 Fall 2019
Milestone 4

Team 106
"<u>Cloud Fridge</u>"

George Butler: gbutler@mail.sfsu.edu - Team Lead, Github Master

Sydni Starling - Scrum Master
James Day - Back-End Lead
Tommy Nguyen - Back-End Developer
Aung Hein - Front-End Lead
Raymond Gee - Front-End Developer
Jian Xin Qi- Developer

Date	Version	Description
12/3/2019	1.0	First Draft

1) Product summary (e.g. how would you market and sell your product – max ¾ page)

- Name of the product: Cloud Fridge
- Explicit list of **ALL major committed functions (for which you will be graded) your team shall deliver and test for. This is your FINAL functional commitment e.g. failure to deliver on some of these functions results in a reduced grade. Please write it in the list format (each item max 1-3 lines) so it is easy to check. List of functions to be written in regular English, not like specs.**
    1. Can take the contents of an uploaded image of a receipt and the contents are automatically inserted into Food List
    2. Can manually add items to the fridge with user input
    3. Items on the Food List will appear in different colors based on the expiration date (red if expired, yellow if expires in a few days, and teal if it has more than few days till expire)
    4. Weekly food report based on consumption
- What is a unique feature in your product
    - Can upload an image of a receipt and the contents are automatically inserted into Food List
- URL to your product accessible to instructors, on the deployment server
    - https://gbutler.dev/

Write this in style targeted for a wide audience: executives, marketing, customers, beta testers. **The list of final functions will be checked on your final delivery for functionality and correct operation as you specified them. Failure to deliver a complete list of these committed functions may result in a reduced grade.**

2) QA test plan - max 2.5 pages
For the function, write a QA unit & integration test plan (check class slides), see below. Basically, it has to contain:

1. Unit Test

Select two-three functions to be tested with your focus. We recommend search or upload recipe/list inventory and recipe recommendation.

- Features to be tested
    1. OCR - Adding items through receipt and manually
    2. Food report - Weekly consumption

- HW and SW setup
- Actual test cases: test cases and results of testing them on your system. It is recommended to run the test by the tool such as JUnit.

  Testing tool: PHPUnit (Test functions created in Laravel)

  Test Cases:

  1. Adding items manually
     a. Add three items manually, inputting the name and other details of the item
     b. Check if the items show up in the fridge after manual input
     c. Check to see if each item's details match what was initially inputted

  2. Adding items through a receipt
     a. Check if items are added to the fridge
     b. Check if item details match expected output based off the receipt used
     c. I.e. only actual food items should show up instead of unnecessary information like the store name

  3. Weekly food report
     a. Mark which items in fridge have been consumed
     b. Check if items are added to food report, including details such as quantity intake
- Analyze test coverage (It is recommended to use the tool)
  1. Test coverage with PHPUnit
  2. Main concern is if-statements that will check the items being added through receipt, check branch coverage for actual food items

2. Integration Test

Select half of all required features out of your committed functional requirements.

- Functional requirements to be tested (please specify the number of Functional requirements in your requirement documentation).
- HW and SW setup
- A suggested format for QA Table columns: test case id; test case description; dates tested, Test scenario with steps to follow, Prerequisites, Test data and Test results (PASS or FAIL for each tested browser).
- Analyze your test coverage for your user story.

Features to test:

1. Fridge inventory
    1.1. Add items manually and through receipt
    1.2. Delete items
    1.3. View items
    1.4. Sort items
    1.5. Search items
    1.6. Notification of food expiration by color
3. Shopping list
    3.1. Create shopping list by manually adding items
    3.2. Add items on shopping list to fridge inventory (as if the user bought the items on the list and wants to transfer them to the inventory)
4. Food report
    4.1. Display user's weekly consumption

Test Case Template: Test Case ID, test description, date tested, test priority, test scenario: test steps with expected and actual results, evaluated by PASS/FAIL

b) Perform the testing as per the plan above and record the results in a form above on your mobile browser. (If you test on a PC browser, you will get the penalty).

3) Beta test plan – 2 pages max
Write a beta test plan for **all** required features out of the functional requirements. Please consult the class material. This test plan is to contain:

- Test objectives: 0.5 page
- Test plan:
    - System setup,
    - starting point,
    - who is the intended user,
    - how to collect users feedback (web portal or email),
    - how to automatically collect the behavior of your product (which features are used the most, how many crashes occur, how long user spent for each feature)
    - URL of the system to be tested.

You can also ask your friends or team members to do this test

Beta Test Plan

- **Test objectives**: To test the P1 functional requirements: Adding items through uploaded receipt image, adding or deleting items from Food List manually, view item information at a glance (quantity, expiration date), search for items by name or brand, notify user of food expiration
- **Test plan**:
  - System Setup: Have the users access the app through the URL on their respective mobile devices (no set OS or specific type of device to gather data from across all types of devices).
  - Starting Point: They start at the welcome page, which is the only page Unregistered User(s) can access.
  - Who is the Intended User: People who need to keep track of the contents of their fridge with other people, or just for themselves. Typically need to be able to see things at a glance when they are out and about. Applies to families, or young adults who are beginning to be more responsible and organized.
  - How to Collect Users Feedback (web portal or email): Google Forms that encompasses how they would rate the high priority features of our app, and a textbox that allows for them to give comments on the app overall, or whatever they would like to say about the app.
  - How to Automatically Collect the Behavior of Your Product (which features are used the most, how many crashes occur, how long user spent for each feature): Error log
  - URL of the System to be Tested: https://gbutler.dev/

4) Code Review:

1. A coding style. In the report say what coding style you chose, and how to enforce it to all members of the team
1. Chose the code (a substantial portion of it) related to the feature you used for QA. You need to submit an example of the code (or part of it – 2 pages or so MAX) for this function to be peer-reviewed, and document this as follows:
1. One team member should submit code to another team member (s) for peer review.
2. Peer review should be performed by another group member (s) (1 review is OK).
3. Comments are to be included in the code
4. Submit the code again

Important: It is critical that code reviews are friendly and helpful, intended to help and education, and not to criticize. It is strongly suggested that you use peer review in the development of the whole system. Reviewers should also check for at least minimal code header and in-line comments and flag this as a problem if this is not adequate

5) Self-check: Adherence to original Non-functional specs
Copy all original non-functional specs as in high level application document published at the very beginning of the class and then for each say DONE if it is done (which is expected and required); ON TRACK if it is in the process of being done and you are sure it will be completed on time; or ISSUE meaning you have some problems and then explain it.

Non-Functional Requirements from High Level Project Description (from Milestone 1)

- **Application shall be developed, tested and deployed using tools and servers reviewed by Class TA (Nicholas Olegovich Stepanov) 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).**
  DONE

- **Application shall be optimized for mobile browsers.**
  DONE

- **Data shall be stored in the team's chosen database technology on the team's deployment server.**
  DONE

- **Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.**
  DONE

- **Application shall be very easy to use and intuitive.**
  DONE

- **Pay functionality, if any (e.g. paying for goods and services) shall not be implemented.**
  DONE

- **Site security: basic best practices shall be applied**
  DONE

- **Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development**
  DONE

- **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).**
  DONE