

# SW Engineering CSC648/848 2019

## Milestone 4

## TEAM 102

## “EzySort”

- IBRAHIM ABOUDAMOUS - TEAM LEAD

[ibrahimaboudamous@gmail.com](mailto:ibrahimaboudamous@gmail.com)

Abdi Mohamud[Backend lead]

Carolyn Chen [Scrum/Frontend]

Tianrong Zhen[Frontend lead/Github Master]

Anne Lanaza[Frontend]

Surabhi Chavan[Frontend]

John Sabour[Backend]

Date	Version	Description
12/03/19	1.0	First Draft

## 1. Product summary

a. The name of our product is Ezy-Sort. Ezy-Sort is a company that allows users to seamlessly scan their grocery receipts and manage what is inside of their fridge. A health page is also included to track how much is consumed and a calorie counter. Ezy-Sort also includes tracking expiration dates to consume food before it becomes waste.

b. Committed functions:

- Add item into inventory list for user.
- Remove item from inventory list for user.
- Placed a Log-In feature to differentiate between authorized-users and unauthorized-users. Which both include different privileges.
- Included a health page that allows user to monitor consumption and calorie counter.

c. Our product is included with a health page that allows user to monitor calories and how much is being consumed. Which can benefit a large demographic of people who want to track how much they consume.

d. <http://ec2-13-57-221-62.us-west-1.compute.amazonaws.com>

## 2. QA test plan:

- II. Unit Test

### **Upload recipe**

HW and SW setup:

HW setup: a laptop, a desktop and a mobile phone

SW setup:

window os or macOS, python, virtualenvironment, django

Actual test cases:

Tested on a regular receipt file: the program successfully extracted all the information and stored them into the database.

Tested on a blank paper: The program didn't extra any information.

Tested on a paper with random characters on it: The program didn't extract any information.

Any bugs: none

/\* \*\*\*\*\* \*/

### **List inventory**

HW and SW setup:

HW setup: a laptop, a desktop and a mobile phone

SW setup: window os or macOS, python, virtualenvironment, django

Actual test cases:

Tested on a regular receipt file: the program successfully extracted all the information and the webpage displayed all the useful information on the screen.

Tested on a blank paper: The program didn't extra any information and the webpage didn't display anything.

Tested on a paper with random characters on it: The program didn't extract any information

Any bugs: none

/ \*\*\*\*\* /

### **Login function**

HW and SW setup:

HW setup: a laptop or desktop

SW setup: window os or macOS, python, virtualenvironment, django

Actual test cases:

Login a user account with the correct username and password: we could successfully log into the user account and saw the information displayed correctly

Login a user account with an incorrect username or password: we could log into the user account

Any bugs: none

Analyze test coverage

Function coverage: for all the three functions above, each of them was called and tested individually. All three of them worked properly and there were no bugs.

Statement coverage: for all the three functions above, each statement inside the function were executed, and the outcome met the expectation.

Branch coverage: for all the three functions above, each case statement was tested and executed without errors.

## II. Integration test

### Functional requirements to be tested:

- Users can register an account.
- Users can view overall reports.
- Calories counter
- Able to add items into the shopping list.

### Hardware and Software Set-Up:

Device: Macbook air, Huawei, iPhone8, and Samsung.

Browsers: Safari and Google Chrome

URL: <http://ec2-13-57-221-62.us-west-1.compute.amazonaws.com>

### QA table:

Test Case ID	2	Test Case Description	Able to view overall report in Home Page				
Created By	Carolyn Chen	Reviewed By		Version	N/A		
QA Tester's Log							
Tester's Name	Carolyn Chen	Date Tested	December 2, 2019	Test Case (Pass/Fail/Not	Fail		
S #	Prerequisites:			S #	Test Data		
1	Access to Chrome Browser			1	Pass		
2	Can view how many items			2	PASS		
3	Can view closing expire items			3	PASS		
4	Can view nutrition informarion			4	The graph do not present nutrition detail of the month		
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://ec2-13-57-221-62.us-west-	Site should open	As Expected		Pass		
2	Click Login	Login and jump to Home	As Expected		Pass		
3	Check all information	Can see overall report	nutrition information missing		Fail		
4							

Test Case ID	1	Test Case Description	Test the Register			
Created By	Carolyn Chen	Reviewed By		Version	N/A	
QA Tester's Log						
Tester's Name	Carolyn Chen	Date Tested	December 2, 2019	Test Case (Pass/Fail/Not	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Chrome Browser		1	Pass		
2	Register an Account		2	Successfully Created an account		
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://ec2-13-57-221-62.us-west-	Site should open	As Expected		Pass	
2	Click Register	Open Register Page	As Expected		Pass	
3	Enter all data and submit	submitted and go to home	As Expected		Pass	
4						

Test Case ID	3	Test Case Description	Able to view Calories of an item			
Created By	Carolyn Chen	Reviewed By		Version	N/A	
QA Tester's Log						
Tester's Name	Carolyn Chen	Date Tested	December 2, 2019	Test Case (Pass/Fail/Not	PASS	
S #	Prerequisites:		S #	Test Data		
1	Access to Chrome Browser		1	PASS		
2	Can see the Calories of each items		2	PASS		
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://ec2-13-57-221-62.us-west-	Site should open	As Expected		Pass	
2	Click Login	Login and jump to Home	As Expected		Pass	
3	Check Refrigerator	Can see Calories of an item	nutrition information missing		PASS	
4						

Test Case ID	4	Test Case Description	Able to add items from Refrigerator to shoppinglist				
Created By	Carolyn Chen	Reviewed By		Version		N/A	
QA Tester's Log							
Tester's Name	Carolyn Chen	Date Tested	December 2, 2019		Test Case (Pass/Fail/Not	PASS	
S #	Prerequisites:			S #	Test Data		
1	Access to Chrome Browser			1	PASS		
2	Can add item into shopping list			2	PASS		
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://ec2-13-57-221-62.us-west-	Site should open	As Expected		Pass		
2	Click Login	Login and jump to Home	As Expected		Pass		
3	Check Refrigerator	Can see an shopping list icon	As Expected		PASS		
4	Click items and click the icon	Add items into shoppinglist	AS Expected		PASS		

### Analyze :

Users are able to Register an account and see some of the information on the home page now. Still missing Nutrition information. Calories are showing next to each item but may still need to work on total Calories of the month. Users can add an item from Refrigerator to the shopping list but not able to add a lot of new items(no in your refrigerator) to the shopping list.

## 3. BETA TEST PLAN

### Test Objective:

The objective of our beta test is to determine where both the faults and the successes of the product lie. By having a dedicated group of individuals who cover the roles of future users, we will be able to efficiently improve upon factors we missed during the previous development stage.

### System Setup:

To set up our testers with the optimal testing environment, all they will need is a computer/smart device and an appetite. The testers will be directed to our system using the

URL: <http://ec2-13-57-221-62.us-west-1.compute.amazonaws.com>

And proceed to register as a user and create their own profile.

#### Starting point/Intended users:

Our product appeals to a wide variety of users, ranging from a mother organizing her grocery agenda for the husband and kids at home to a bodybuilder with strict dietary planning needs. In order to effectively test we need not search far to find people who fit the criteria for a long time user. We will have each of the testers utilize the website in their own individual ways, spanning from strict nutritional and financial needs, to a lax grocery list they made in order to remember whether they need to pick up milk. Having all of these various types of starts

#### Feedback:

In order to collect feedback we will be having all of the beta testers send emails to our company email address. We will have an automated reply to each of these reports, so that testers will know that the bugs and issues they encounter are in fact being reviewed and not scrapped.

#### Reports:

In order to determine which features of our product are used the most we will be monitoring the frequency of user registrations, logins, and item additions/removals. Through this, we will be able to review what parts of the website are experiencing the highest traffic and focus on

improving the quality of life for those aspects. We will be able to see how long a user spends on each page by reviewing the refresh count for each specific page. The time between a user visiting their cart and reloading back to the home page will give us a good estimate as to the amount of time they spent navigating that page.

### Response Collection:

Realizing how many crashes occur and on which pages relies solely on the testers. Without the beta testers relaying to us information as to the count and cause of crashes, we would not be able to accurately determine these numbers.

### 3. Code Review

#### a) Coding Style:

To enforce these styling habits, having peer reviews or having someone go through the code like one person for the frontend and another for the backend. Checking the readability and whether the style is actually consistent all throughout the code.

a. Frontend: Google HTML/CSS

b. Backend: Pep8

#### b) Peer Review Sample:



Tianrong Zhen 10:41 PM

 **Tianrong Zhen** 10:41 PM


```
<script>
  function myCreateFunction(name, date, date2, cal) {
    var table = document.getElementById("myTable");
    var row = table.insertRow(-1); // 0
    var cell1 = row.insertCell(-1); // 0
    var cell2 = row.insertCell(-1); // 1
    var cell3 = row.insertCell(-1); // 1
    var cell4 = row.insertCell(-1);
    cell1.innerHTML = name;
    cell2.innerHTML = date;
    cell3.innerHTML = data2;
    cell4.innerHTML = cal;
  }

  function myDeleteFunction() {
    document.getElementById("myTable").deleteRow(1);
  }

</script>
```


I guess this is the most lengthy part of code I can find, others are way too short

Anne L 11:06 PM

 **Anne L** 11:06 PM


The coding style looks fine to me, all that I see is that closing script isn't aligned with opening script.

Tianrong Zhen 11:07 PM

 **Tianrong Zhen** 11:07 PM


All right, do I need to write any comments?

Anne L 11:17 PM

 **Anne L** 11:17 PM

Nope. This is pretty much good

Anne L 11:17 PM

 **Anne L** 11:17 PM

Nope. This is pretty much good

```
<script>
  function myCreateFunction(name, date, date2, cal) {
    var table = document.getElementById("myTable");
    var row = table.insertRow(-1); // 0
    var cell1 = row.insertCell(-1); // 0
    var cell2 = row.insertCell(-1); // 1
    var cell3 = row.insertCell(-1); // 1
    var cell4 = row.insertCell(-1);
    cell1.innerHTML = name;
    cell2.innerHTML = date;
    cell3.innerHTML = data2;
    cell4.innerHTML = cal;
  }

  function myDeleteFunction() {
    document.getElementById("myTable").deleteRow(1);
  }

</script>
```

#### 4. Self-Check

- The 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**
- The 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**
- The 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). **ON TRACK**