**CS673 Software Engineering**

**Team 4 - MyDietHub**

**Tests Report**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Josephine But | QA Leader | *Josephine But* | June 21, 2021 |
| Ray Clark | Team Lead | *Ray Clark* | June 14, 2021 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| Iteration 2 | Josephine But | June 14, 2021 | **Test Cases for testing. Add Unit testing report and code coverage report** |
| Iteration 3 | Josephine But | June 19, 2021 | **Update Test Cases. Add Integration testing report and System Testing report** |

[Introduction](#_87t9hln2vjz0)

[Test Summary](#_sm5odwyvuk3j)

[Tests Reports](#_pqso2mbjyzx4)

[Testing Metrics](#_mtfbusfb0eq3)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Introduction

This document is the MyDietHub application test report for different iterations. It contains

the results of tests and to disclose all testing metrics. We are using the Python unittest

framework for unit testing and for end-to-end testing of the system, we are using the

Python Selenium module.

Developers are responsible for unit tests covering the user stories. Our QA Leader is

taking the lead for selenium end-to-end tests.

# Test Summary

* + UI Tests

These tests are performed manually. Testers are to verify the visual elements of the application to validate whether they accurately meet the expected performance and functionality. All the results will be documented in this document. We use black box testing for UI tests.

* + Unit Tests

The purpose of the unit testing is to validate that each unit of the application performs as designed based on the User Stories. Developers use the Python unittest module to develop the tests and re-use the unit tests any time the unit is changed. We use white box testing for Unit Tests

* + Integration Tests

The purpose of the Integration testing is to test the components when integrated to verify that they work as expected. These tests are performed manually. All the results will be documented in this document. Unit tests can be part of the continuous integration framework. We use white box testing for Integration Tests.

* + System Tests

The purpose of the system testing(end-to-end testing) is to test the system as a whole. All the components are integrated in order to verify if the system works as expected or not. These tests are performed automatically. Developers use the Python Selenium module to develop the tests and execute the tests for system testing to make sure the system works as expected as whole. We use black box testing for system tests.

# Tests Reports

**Iteration 2 results**

**UI Tests**

App startup:

* Test case ID : UI0001
* Test name : App startup
* New or old: NEW
* Test items: Testing myDietHub go to the landing page and all the menu items are there.
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): None
* Preconditions: (if any) : None
* input data: run flask app on command line
* Test steps:
  + Go to app directory
  + Make sure you have executed the command “export FLASK\_APP=app.py”
  + Run the following command : “flask run”
  + Navigate to IP address 127.0.0.1:5000 on any web browser
* Postconditions: None
* Expected output: MyDietHub homepage is displayed.
* Actual output: MyDietHub homepage is displayed
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

(You can use an additional spreadsheet for this section as well)

Navigation bar options:

* Test case ID : UI0002
* Test name : Navigation bar options
* New or old: NEW
* Test items: Testing myDietHub’s home page has the “Home”, “About”, “Login/Sign Up” and “Logout” links are on the navigation bar.
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): None
* Preconditions: (if any) : None
* input data: run flask app on command line
* Test steps:
  + Navigate to IP address 127.0.0.1:5000 on any web browser
  + MyDietHub homepage is displayed
* Postconditions: None
* Expected output: The “Home”, “About”, “Login/Sign Up” and “Logout” links are on the navigation bar.
* Actual output: The “Home”, “About”, “Login/Sign Up” and “Logout” links are on the navigation bar.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

User Registration:

* Test case ID : UI0003
* Test name : User Registration
* New or old: NEW
* Test items: Unregistered user registration
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): “Sign up here” link is available in the login page.
* Preconditions: (if any) : Registration page is available
* input data: Firstname(“test1F”), Lastname(“test1L”), username(“test1”), password(“test1”) and email address(“test1@123.com)
* Test steps:
  + Navigate to the landing page
  + Select the “Login/Sign Up” link from the navigation bar.
  + Select the “Sign up here” link to navigate to the registration page
  + Enter the input data
  + Click the “Sign Up” button
* Postconditions: User is redirected to the login page.
* Expected output: User is redirected to the login page
* Actual output: User is redirected to the login page
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

User Login:

* Test case ID : UI0004
* Test name : User Login
* New or old: NEW
* Test items: Registered user login
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): “Login/Sign up link is available on the navigation bar
* Preconditions: (if any) : Login page is available
* input data: Username(“test1”) and password(“test1”)
* Test steps:
  + Navigate to the landing page.
  + Select the “Login/Sign Up” link from the navigation bar.
  + Enter the input data
  + Click the “Sign In” button
* Postconditions: User is redirected to the “Create a Meal” page.
* Expected output: User is redirected to the “Create a Meal” page
* Actual output: User is redirected to the “Create a Meal” page
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Create a meal:

* Test case ID : UI0005
* Test name : Create a meal
* New or old: NEW
* Test items: User can create a meal
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): User can login into the system
* Preconditions: (if any) : User must be logged into the system and can be landed to the “Create a meal” page after login
* input data: Dinner, Rice, Beef
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Enter the input data
  + Click the “Submit” button
* Postconditions: “Meal Added” message should be displayed
* Expected output: “Meal Added” message is displayed
* Actual output: “Meal Added” message is displayed
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

View a meal

* Test case ID : UI0006
* Test name : View a meal
* New or old: NEW
* Test items: User can view a meal
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): User can login into the system and at least one meal is already created.
* Preconditions: (if any) : User must be logged into the system and at least one meal is created.
* input data: meals data are in the system
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Click the “View Meals” button
* Postconditions: List of past meals should be displayed
* Expected output: List of past meals is displayed
* Actual output: List of past meals is displayed
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Update a meal

* Test case ID : UI0007
* Test name : Update a meal
* New or old: NEW
* Test items: User can update a meal
* Test priority (high/medium/low) : Medium
* Dependencies (to other test case/requirement if any): User can login into the system and at least one meal is already created.
* Preconditions: (if any) : User must be logged into the system and at least one meal is created.
* input data: meal data is already created
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Click the “View Meals” button to view the user’s meals
  + Click the “Update” link for a meal
  + Change the values in the Meal Type, Food Item 1 and Food Item 2
  + Click the “Update” button
* Postconditions: The List of past meals page should be displayed and the meal is updated.
* Expected output: The List of past meals page is displayed and the meal is updated.
* Actual output: The List of past meals page is displayed and the meal is updated.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Delete a meal

* Test case ID : UI0008
* Test name : Delete a meal
* New or old: NEW
* Test items: User can delete a meal
* Test priority (high/medium/low) : Medium
* Dependencies (to other test case/requirement if any): User can login into the system and at least one meal is already created.
* Preconditions: (if any) : User must be logged into the system and at least one meal is created.
* input data: the meal data from the system
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Click the “View Meals” button to view the user’s meals
  + Click the “Delete” link to delete the meal
* Postconditions: The List of past meals page should be displayed and the meal is deleted.
* Expected output: The List of past meals page is displayed and the meal is deleted.
* Actual output: The List of past meals page is displayed and the meal is deleted.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Admin Login:

* Test case ID : UI0009
* Test name : Admin Login
* New or old: NEW
* Test items: Administrator login
* Test priority (high/medium/low) : MEDIUM
* Dependencies (to other test case/requirement if any): “Login/Sign up link is available on the navigation bar
* Preconditions: (if any) : Administrator account is already created in the database
* input data: Username (“admin1”) and password(“admin1”)
* Test steps:
  + Navigate to the landing page.
  + Select the “Login/Sign Up” link from the navigation bar.
  + Enter the input data
  + Click the “Sign In” button
* Postconditions: User is redirected to the “Retrieve Userl” page.
* Expected output: User is redirected to the “Retrieve User” page
* Actual output: User is redirected to the “Retrieve User” page
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Admin to retrieve user profile:

* Test case ID : UI0009
* Test name : Retrieve user profile
* New or old: NEW
* Test items: Retrieve user profile
* Test priority (high/medium/low) : MEDIUM
* Dependencies (to other test case/requirement if any): Admin is able to login
* Preconditions: (if any) : Admin must be logged into the “Retrieve User” page and the user account is created.
* input data: user’s username(“test1”)
* Test steps:
  + Admin login to the “Retrieve User” page
  + Enter the users’s username
  + Click the “Retrieve User” button
* Postconditions: User is redirected to the “User Details” page and the details of user is displayed.
* Expected output: User is redirected to the “User Details” page and the details of user is displayed.
* Actual output: User is redirected to the “User Details” page and the details of user is displayed.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Admin to update user profile:

* Test case ID : UI0010
* Test name : Update user profile
* New or old: NEW
* Test items: Update user profile
* Test priority (high/medium/low) : MEDIUM
* Dependencies (to other test case/requirement if any): Admin is able to login
* Preconditions: (if any) : Admin must be logged into the “Retrieve User” page and the user account is created.
* input data: user’s firstname(“test1FUpate”), lastname(“test1LUpdate”), username(“test1Update”), password(“1234”) and email(“test1Update@123.com)
* Test steps:
  + Admin login to the “Retrieve User” page
  + Enter the user’s username
  + Click the “Retrieve User” button
  + Click the “Update” link
  + Update each fields with input data
  + Click the “Submit” button
* Postconditions: User is redirected to the “Retrieve user” page and the user’s profile should be updated
* Expected output: User is redirected to the “Retrieve user” page and the user’s profile is updated.
* Actual output: User is redirected to the “Retrieve user” page and the user’s profile is updated.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Admin to delete user profile:

* Test case ID : UI0011
* Test name : Delete user profile
* New or old: NEW
* Test items: Delete user profile
* Test priority (high/medium/low) : MEDIUM
* Dependencies (to other test case/requirement if any): Admin is able to login
* Preconditions: (if any) : Admin must be logged into the “Retrieve User” page and the user account is created.
* input data: user’s username(“test1”)
* Test steps:
  + Admin login to the “Retrieve User” page
  + Enter the users’s username
  + Click the “Retrieve User” button
  + Click the “Delete” link
* Postconditions: The “Retrieve User” page should be displayed and the user is deleted.
* Expected output: The “Retrieve User “ page is displayed and the user is deleted.  
  Actual output: The “Retrieve User” page is displayed and the user is deleted.
* Pass or Fail: PASS
* Bug id/link: (this should link to your github issue id)
* Additional notes:

Meal Update with invalid food items :

* Test case ID : UI0013
* Test name : Meal Update with invalid food items
* New or old: NEW
* Test items: Meal Update
* Test priority (high/medium/low) : LOW
* Dependencies (to other test case/requirement if any): User can login into the system and at least one meal is already created.
* Preconditions: (if any) : User must be logged into the system and at least one meal is created.
* input data: food type: “LunchUpdate”, foodItem1: “riceUpdate”, foodItem2: “beefUpdate”
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Click the “View Meals” button to view the user’s meals
  + Click the “Update” link for a meal
  + Change the values in the Meal Type, Food Item 1 and Food Item 2 with input data
  + Click the “Update” button
* Postconditions: The List of past meals page should be displayed and the meal is updated.
* Expected output: The List of past meals page is displayed and the meal is updated.
* Actual output: The List of past meals page is displayed and the meal is updated.
* Pass or Fail: FAILURE
* Bug id/link: [Issue #5](https://github.com/BUMETCS673/BUMETCS673OLSum21P4/issues/5#issue-924253016)
* Additional notes:
  + The List of past meals page is not displayed with meals.
  + Bug fixed. Retest and closed the bug.

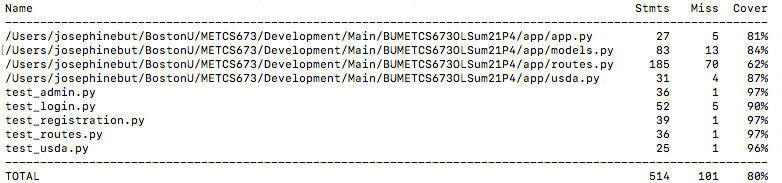
Wrong calories retrieve for a food item with separated word:

* Test case ID : UI0014
* Test name : Wrong calories retrieve for a food item with separated word
* New or old: NEW
* Test items: Create a Meal, Update a Meal
* Test priority (high/medium/low) : HIGH
* Dependencies (to other test case/requirement if any): User can login into the system and at least one meal is already created.
* Preconditions: (if any) : User must be logged into the system and at least one meal is created.
* input data: food type: “Lunch”, foodItem1: “beef”, foodItem2: “Apple Juice”
* Test steps:
  + Login into the system to the “Create a meal” page.
  + Enter the food type and food items with input data
  + Click the "Submit" button
  + Click the “View Meals” button to view the user’s meals

* Test Steps:
  + Click the “Update” link for a meal
  + Change the values in the Food Item 2 with "Orange Juice"
  + Click the “Update” button
* Postconditions: The calories of the food items are displayed with correct calculations.
* Expected output: The calories of the food items are displayed with correct calculations.
* Actual output: The calculations for the calories of the food items( Apple Juice and Orange Juice) are incorrect.
* Pass or Fail: FAILURE
* Bug id/link: [Issue #6](https://github.com/BUMETCS673/BUMETCS673OLSum21P4/issues/6#issue-924262857)
* Additional notes:
  + The calculations of the calories for the food items should be displayed correctly in "Create A Meal" and after updating a meal.
  + Bug fixed, retest and closed the bug.

# Testing Metrics

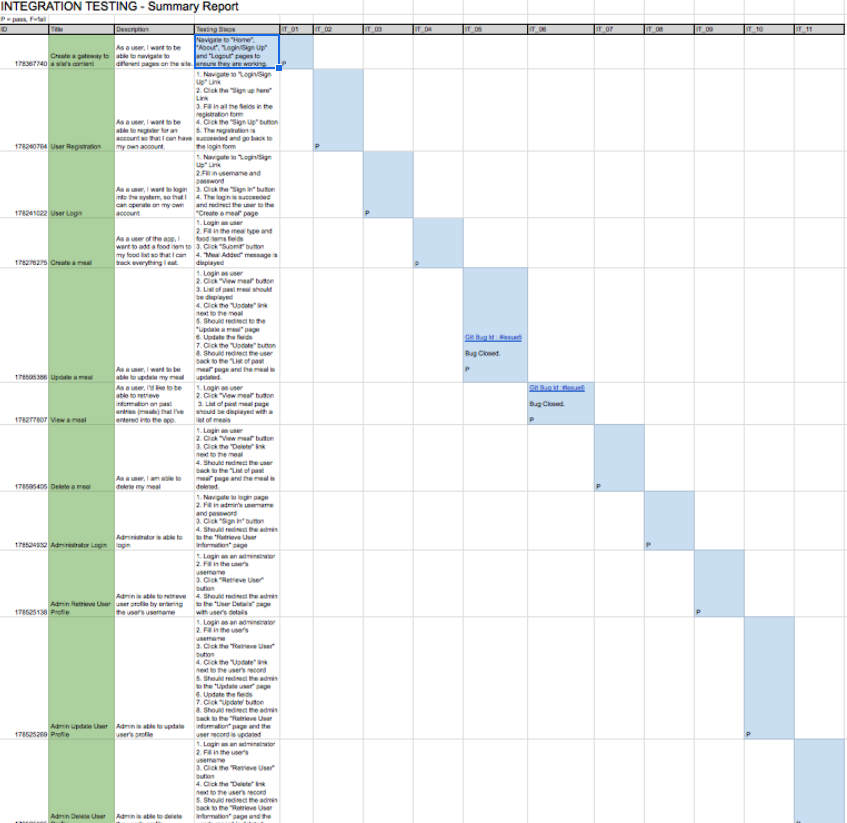
The document below details the unit testing coverage rate for each unit test file as well as the entire application.

**Iteration 2 Code Coverage Re****sult:**

Currently, the unit tests cover approximately 80% of the application.

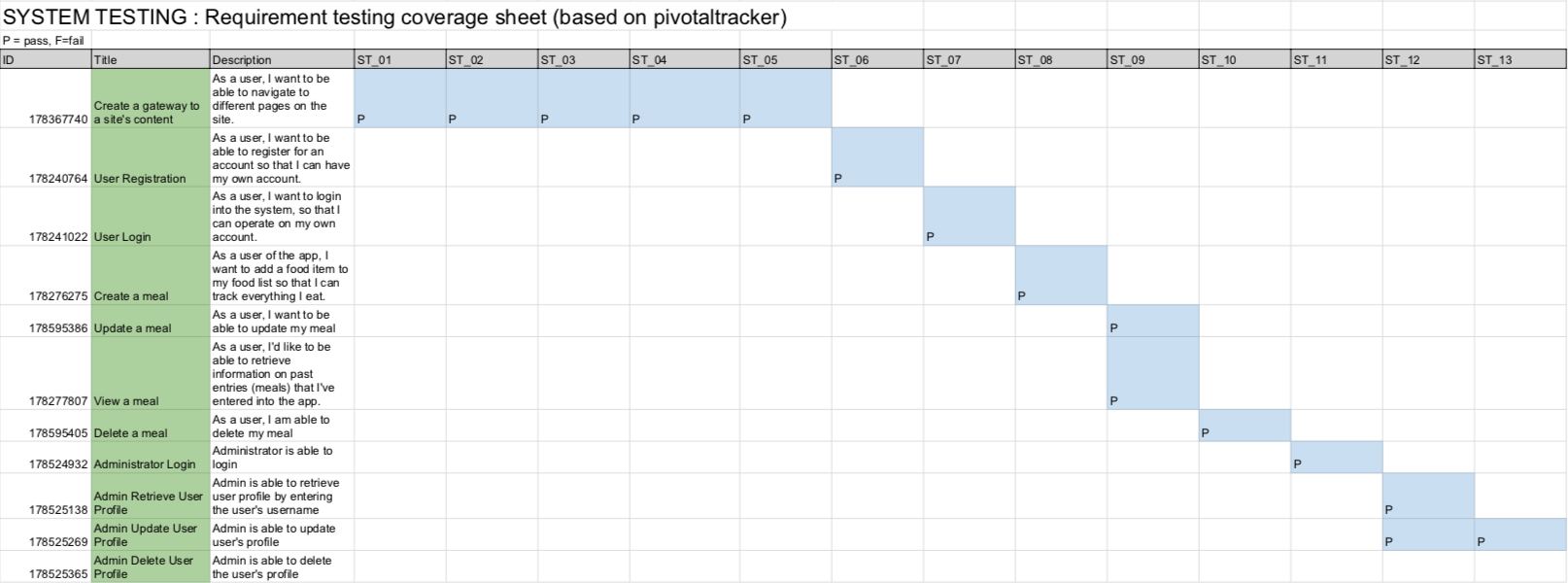
**Integration Testing Result :**

[**Integration Testing Report Link**](https://drive.google.com/file/d/18V0pDhNlqjEGLVXQyw3czA-b8Fhr97cN/view?usp=sharing)

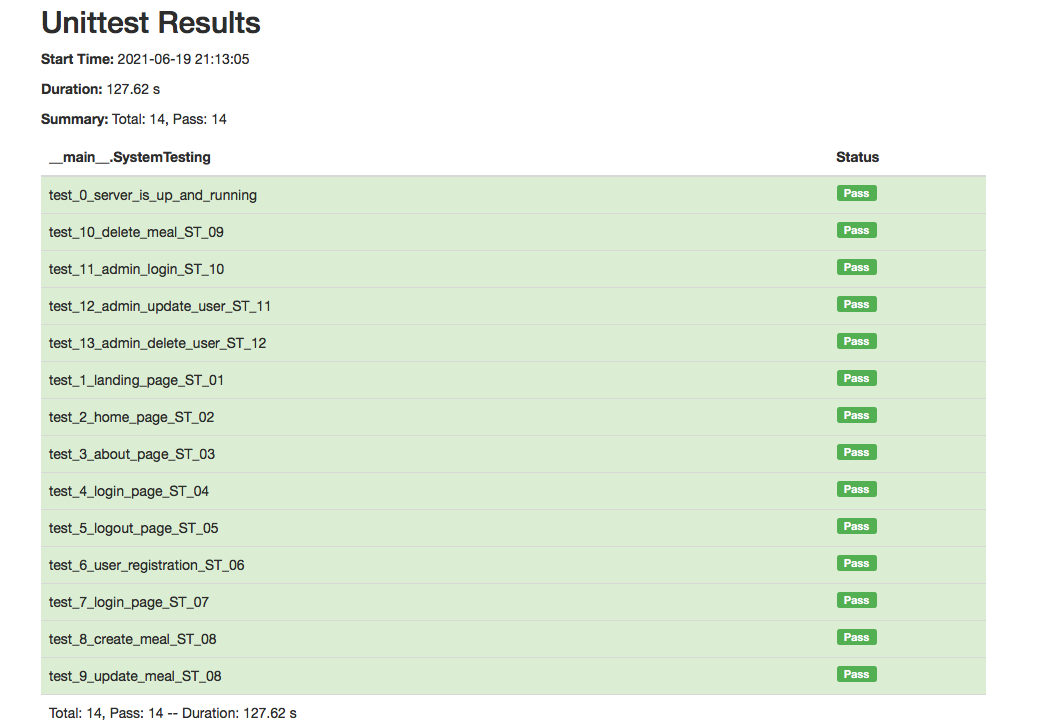


**System Testing Results for Iteration 3 :**

[**System Testing Report Link**](https://drive.google.com/file/d/1HA6cM_c-vSkbADzeiViENZCigLuDxwZO/view?usp=sharing)



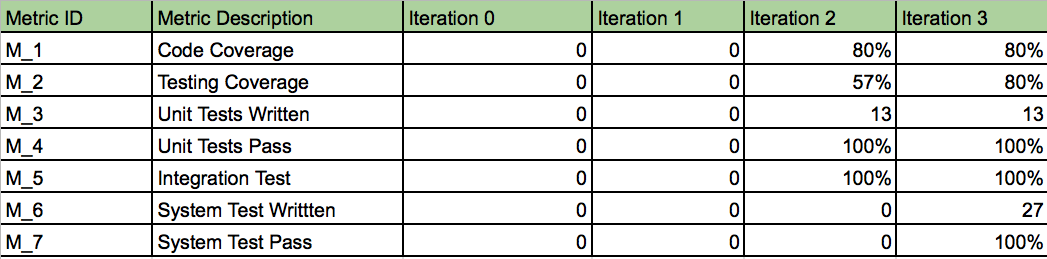
**System Testing Result:**



In Iteration 3, we developed 14 test cases with Selenium for system testing and the result is 100% pass rate.

* **Functionality coverage** 
  + 100%. Every single feature has been tested. 30% higher than iteration 2
  + Functionality of view the meal, update the meal and delete the meal, admin update user profile and delete the user account are included in the iteration 3 for System Testing.
* **Defect Rate**
  + Including all manual and automated tests, 11% defect rate

**Overall Testing Metrics :**



# References

# Glossary

In Code Coverage Result:

* Stmts : Statements.
* Miss: Number of statements that were not covered by testing
* Cover : Percentage of statements included in a unit test over the total number of statements in the application.