**CS673 Software Engineering** 

**Team 1 - cheffy**

**Tests Report**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Justin Fanning | Team Leader, Requirement Leader | *Justin Fanning* | 9/12/2021 |
| Kyle Mabry | Backup Team leader | *Kyle Mabry* | 9/12/2021 |
| Jean Shalenkova | Design and Implementation leader; Configuration leader | *Jean Shalenkova* | 9/12/2021 |
| Yanru Zhu | QA leader | *Yanru Zhu* | 9/12/2021 |
| Aidan Duffy | Security leader | *Aidan Duffy* | 9/12/2021 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Yanru Zhu | 10/11/2021 | Put in the first version of the testing strategy and testing results so far |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Test Summary](#_sm5odwyvuk3j)

[Tests Reports](#_pqso2mbjyzx4)

[Testing Metrics](#_mtfbusfb0eq3)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Introduction

This document provides an overview of the testing plan used by the Cheffy team, including the test methods the team will use, test reports, and test metrics used to evaluate the quality of this app. The cheffy app is being developed using a test-driven development methodology and both developers and QA leader are involved in the testing process.

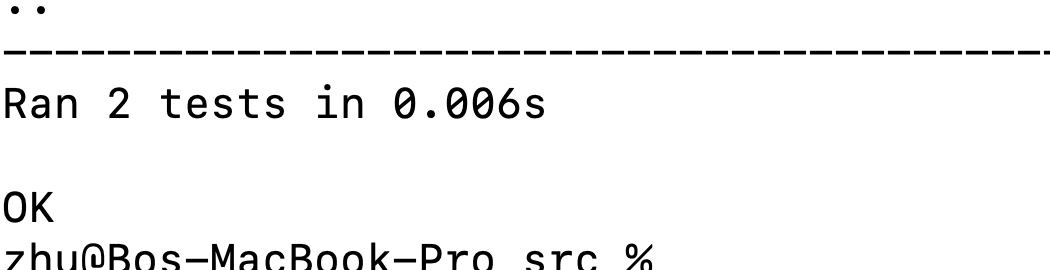
The cheffy app is built using Python, Flask as backend and Jinja templates as front-end. All testing is being managed by Python. We are mainly using unittest, selenium end-to-end tests, and manual tests for this app.

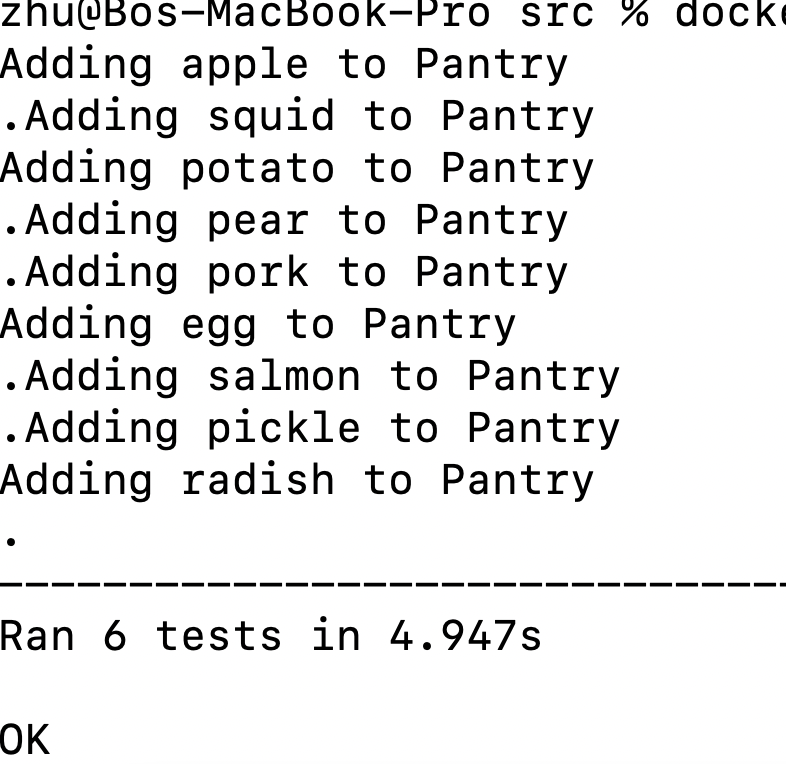
# Test Summary

* **Unit Tests:** The purpose of the unit test is to make sure each component developed is meeting quality standards. ThePython unittest module is used to develop the tests and this is done by the developers working on the specific feature. The responses library is used in some tests as well since this app is mainly using a third party API and so web response is something we need to make sure that’s working well. This is a white-box testing. Example of the unit tests include:
  + The function of searching a recipe works by checking if the recipe data returned matched the expectation.
  + The function of getting a correct specific recipe page by checking if the response gotten from the call matches what’s expected.
* **System Tests:** The QA leader is leading the system tests to make sure the app is built meeting the requirements, so the test is following the user stories we have in [PivotalTracker](https://www.pivotaltracker.com/n/projects/2531653) by testing the main feature, which is mainly the search feature. Selenium is used to do this end-to-end system testing so every test is automatic and efficient. Features that have been tested using Selenium include:
  + Visit the site.
  + Search the recipe after entering ingredients.
  + Display recipe details after clicking on a specific recipe.
* **Manual Tests:** The QA leader is leading the manual tests by manually following specific steps to make sure all the features mentioned in [PivotalTracker](https://www.pivotaltracker.com/n/projects/2531653) are built correctly. The acceptance test description in each user story is used to see if the requirements are met. Features that have been tested include but not limited to:
  + Sign up for cheffy using an email address and password.
  + Log into cheffy using correct email address and password, and incorrect email address and password.
  + Search recipes using ingredients.
  + Display a recipe with recipe details.
  + Add/delete ingredients into/from a personal pantry.

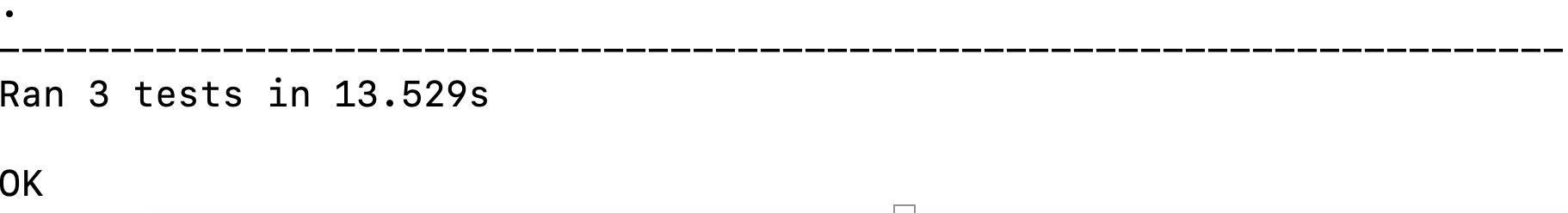
# Tests Reports

* **Unit tests:** [This](https://github.com/BUMETCS673/BUMETCS673OLF21P1/tree/main/src/tests) test folder contains the unit test files written by developers, so far there have had two features (get a recipe detail and personal pantry) being tested using unit tests. Below is a screenshot of the test result - all tests (8) have passed. To run the test you need to install the responses library.

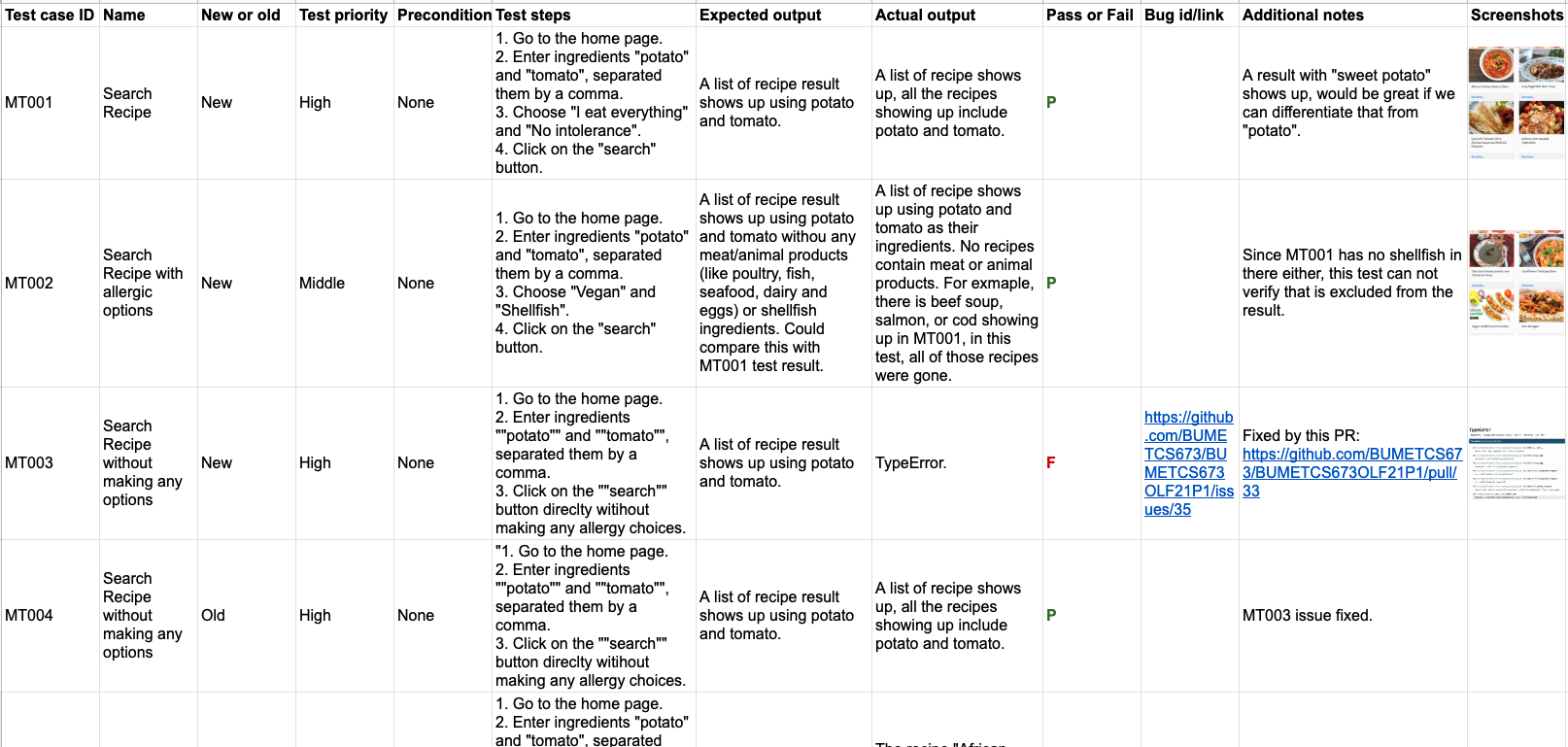




* **System tests:** [This file](https://github.com/BUMETCS673/BUMETCS673OLF21P1/blob/main/src/tests/selenium_tests.py) contains the selenium test code and below is the screenshot of the test result - all tests have passed. To run the test you need to install selenium.



* **Manual tests:** [Here](https://docs.google.com/spreadsheets/d/1Q_5AYMsETIKbEJEdUeavqGPvvWOEixiszPwasDHC_Cw/edit#gid=1057987399) is the spreadsheet that records the manual test and below is a screenshot of the test. So far we’ve found a few defects when doing manual tests and have those defects fixed as soon as we can.



# Testing Metrics

Testing metrics are recorded following each iteration and metrics were chosen at the beginning of the project and explained in the [SPPP](https://docs.google.com/document/d/1tyz7XQUj_klupyI85l6bEhwMsBlASTLptq9FQ-e93F4/edit) document.

* Product metrics: Below metrics are tracked to determine the quality of the product.

| **Metric Name** | **Iteration 0** | **Iteration 1** | **Iteration 2** | **Iteration 3** |
| --- | --- | --- | --- | --- |
| Unit Tests Passing Rate (Unit Tests Passed/Unit Tests Written) | None | None | 8/8 = 100% |  |
| System Tests Passing Rate (System Tests Passed/System Tests Written) | None | None | 3/3 = 100% |  |
| Manual Tests Passing Rate (Manual Tests Passed/Manual Tests Written) | None | None | 10/12 = 83% |  |
| User story counts | None | 8 (finished 1) | 8 (finished 7) |  |

* Process Metrics: We use below metrics to measure the effectiveness of the process.

| **Metric Name** | **Iteration 0** | **Iteration 1** | **Iteration 2** | **Iteration 3** |
| --- | --- | --- | --- | --- |
| Defects fix rate (defects fixed/total defect counts) | None | 1/1 = 100% | 3/3 = 100% |  |
| Project effort (total hours spent) | 7+8+8+9+10+5 =47 | (8+8)+(7+8)+(8+8)+(8+8)+(8+8)+(4+5) = 88 | (22) + (8.5+20.5) + (10+15) + (10+19) + (10) = 115 |  |

# References

* [CS673 Module 4](https://onlinecampus.bu.edu/bbcswebdav/pid-9237538-dt-content-rid-57219237_1/courses/21fallmetcs673_o1/course/module4/allpages.htm)
* [Python unittest](https://docs.python.org/3/library/unittest.html)
* [Selenium](https://www.selenium.dev/)

# Glossary

* Unit Tests: Testing individual components (units). A unit is usually a class or a method. Sometimes it can be a group of classes or methods. The goal is to confirm that the individual component is correctly coded and carries out the intended functionality. It is usually performed by developers and executed automatically.
* System Tests: Testing the entire system, including its integration with other software and hardware systems. The goal is to determine whether the system meets functional and nonfunctional requirements.
* Manual Tests: Test cases are developed in documents and performed manually by people.