****

## School of Computing, Engineering and Built Environment

## 

## Programming Paradigms - MHI325688

## Coursework Testing

## 

## Ewan Allison - S2004763

## **Testing for Food Prices Application**

**Test Objective**

Bugs or errors can be detected and dealt with through rigorous testing on the food prices application. This is performed, not only to improve quality but also to ensure the application meets the requirements.

**Test Plan**

The chosen functional testing strategy was performed and documented within the test results section of this document. To test all functions and analysis, the test scope is defined as the following:

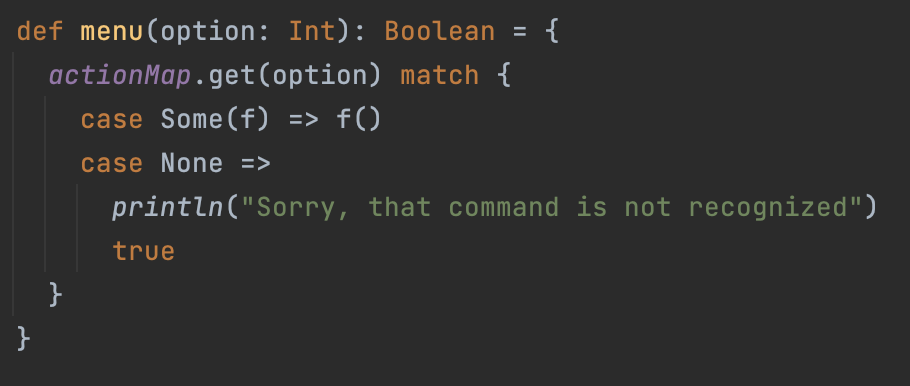
1. Menu Navigation
2. Data Loading
3. Analysis

**Test Results**

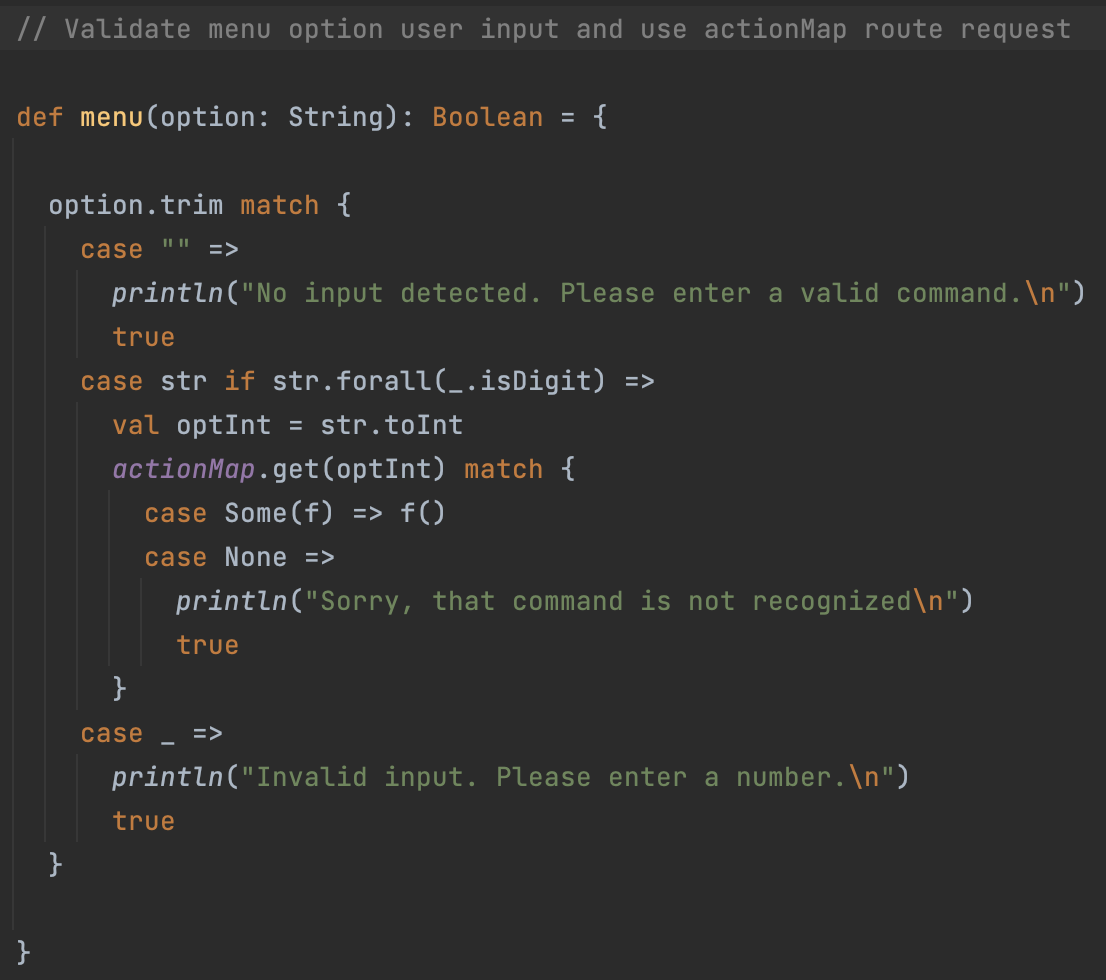
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Action** | **Expected Outcome** | **Pass/Fail** | **Evidence** |
| 1.1 | Type a valid integer menu option and press enter | The application takes you to the appropriate functionality and waits for the user to press “Enter” before returning to the menu. | **PASS** |  |
| 1.2 | Type an invalid integer menu option and press enter | The application outputs an error message and awaits valid input. | **PASS** |  |
| 1.3 | Type a string and press enter | The application outputs an error message and awaits valid input. | **FAIL** |  |
| 1.4 | Type a character and press enter | The application outputs an error message and awaits valid input. | **FAIL** |  |
| 1.5 | Only press enter | The application outputs an error message and awaits valid input. | **FAIL** |  |
| 2 | Start the application | “Data.txt” is successfully read in as a Map with the format “Map[String, List[Int]]”. | **PASS**  println(foodPriceData) temporarily added |  |
| 3.1 | Enter 1, view the result, then press enter | A table view of all food items and their most recent price is shown. The app waits for the user to press “enter” to return to the menu. | **PASS** |  |
| 3.2 | Enter 2, view the result, then press enter | A table view of all food items and their highest and lowest prices is shown. The app waits for the user to press “enter” to return to the menu. | **PASS** |  |
| 3.3 | Enter 3, view the result, then press enter | A table view of all food items and their median prices is shown. The app waits for the user to press “enter” to return to the menu. | **PASS** |  |
| 3.4 | Enter 4, view the result, then press enter | A table view of the highest rising food, its price increase, and percentage increase is shown. The app waits for the user to press “enter” to return to the menu. | **PASS** |  |
| 3.5.1 | Enter 5, then enter two valid food items, view the result, then press enter | The average over 12 months of both entered food items is presented to the user. The app waits for the user to press “enter” to return to the menu. | **PASS** |  |
| 3.5.2 | Enter 5, then enter invalid food items | An error message prompts the user that their imputed food was not found in the list, reattempt until success. | **PASS** |  |
| 3.6.1 | Enter 6, enter 1, then enter a valid food item and quantity, and finally view the results. Repeat the above steps to add more items or enter 2 to return to the menu. | Item basket is displayed with correct individual and total prices. | **PASS** |  |
| 3.6.2 | Enter 6, enter 1, then enter an invalid food item | An error message prompts the user that their imputed food was not found in the list, reattempt until success. | **PASS** |  |
| 3.6.3 | Enter 6, enter 1, then enter a valid food item but an invalid quantity. | An error message prompts the user their input was invalid, and returns to menu. | **PASS** |  |

**Actions Taken**

Relating to failed tests 1.3, 1.4, and 1.5, this is the original code snippet that handles user input for menu navigation:

****

As it is written to receive and validate only integer inputs by the user, this explains how 1.2 passes (entering out-of-bounds integers) but any relating to character, string, or empty inputs fail. To solve this issue and handle the error, the menu function has been altered like so:



By altering the code to read the menu option as a string, it becomes easier to handle invalid user input. Valid input can be converted back into an int and sent into the actionMap to handle the request.

**Calculations**

To validate the application functionality, each analysis will be manually calculated and compared to the actual result:

*Find Recent Price (Apple)*

Applications Result = £2.81

Actual Result = £2.81

Manually Calculated

*View Highest/Lowest Price (Apple)*

Applications Result = £2.81 high & £2.00 low

Actual Result = £2.81 high & £2.00 low

Manually Calculated

*View Median Price (Apple)*

Applications Result = £2.25

Actual Result = £2.25

Calculations:

A screenshot of a calculator

Description automatically generated

*View Highest Riser*

Applications Result = Chicken

Actual Result = Chicken

Manually Calculated

*Item Basket total (1kg Apple, 100g butter, 500g flour)*

Applications Result = £3.95

Actual Result = £3.95

Manually Calculated