CPS3230 – Fundamentals of Software Testing

ASSIGNMENT PART 1 OF 3 – MARKET ALERT UM

A picture containing graphical user interface

Description automatically generated

Robert Fenech Adami

B.Sc. (Hons.) Information Technology (Software Development)

November 2022

**FACULTY OF INFORMATION AND**

**COMMUNICATION TECHNOLOGY**

Declaration

Plagiarism is defined as “the unacknowledged use, as one’s own work, of work of another person, whether or not such work has been published” (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I / We\*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our\* work, except where acknowledged and referenced.

I / We\* understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

\* Delete as appropriate.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Robert Fenech Adami

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Student Name | Signature |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Student Name  CPS3230 | Signature |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Assignment Part 1 of 3 – Market Alert UM  Course Code Title of work submitted | |

13/11/2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date

# **Task 1**

Github repository: <https://github.com/robbyfa/marketAlertTask1>

Google Drive: <https://drive.google.com/drive/folders/1wZVTt368XZxm4r4mASsn_p7OxxzBagyF?usp=sharing>

The SUT implemented consists of an ‘Alert’ object which includes all its relative details. An ‘AlertCatalogue’ class is implemented. This class contains a list of all alert objects created, together with relative methods to gather data through web scraping, publish alerts to marketAlertUM and, also, delete all alerts from the site, among other methods. A utilities folder which consists of several interfaces is also used. The architecture for the system under test can be seen in figure 1 below. All data was gathered from:

<https://www.ultimate.com.mt/product-category/tv-audio/audio/headphones/>

Graphical user interface, text, application, chat or text message

Description automatically generated[Fig.1]

Several tests have been implemented for each aspect of the system using a test-driven approach, thus obtaining 100% test coverage, as seen in Figure 2 below. Each test was implemented using a ‘setup, verify, exercise, and verify architecture; Furthermore, these tests also make use of spies in order to verify the number of alerts being saved to the alerts list and to verify the number of alerts being published to marketAlertUM via the REST API [fig. 3] [fig. 4]. The tests also make use of Mockito to assign the alert type to each alert being published.

Table

Description automatically generated [Fig.2]

Text

Description automatically generated [Fig.3]

It is assumed that all data being gathered via the Chrome driver web scraper fall under the electronics category when publishing alerts via the REST API. Furthermore, it is assumed that all data, for 5 alerts, is being stored at one go, rather than adding and publishing one alert at a time. Further to this, the method used to call the web scraper simply implements a loop to gather alert data until the maximum limit is reached.

Another measure taken to ensure proper web scraping is the use of Gson and FileWriter dependency and library. These are used to store all data retrieved from the automated web scraper to a JSON file, as seen below in figure 5.

Text

Description automatically generated[Fig.4]

Text

Description automatically generated [Fig. 5]

The below figure shows the success status of all tests, with 15 of 15 passes.

A picture containing text

Description automatically generated

AlertCatalogueTests:

|  |  |
| --- | --- |
| **Test Name** | **Scope** |
| testEmptyAlertCatalogue | Assure alerts list is empty |
| testAlertCatalogueWith5Alerts | Assure web scraper gathers data for 5 alerts |
| testProductSavedToFile | Assure 5 alerts are saved to file |
| testNumberOfAlertsPosted | Assure number of alerts being posted |
| testHttpRequest | Assure successful use of API endpoint |
| testRequestUri | Assure proper HTTP request URI |
| testDeleteAlertsRequest | Assure successful use of API ‘Delete’ endpoint |
| testClearLists | Assure alerts list is emptied |

AlertTests:

|  |  |
| --- | --- |
| **Test Name** | **Scope** |
| testHeading | Assure heading variable is assigned |
| testDescription | Assure description variable is assigned |
| testUrl | Assure url variable is assigned |
| testImageUrl | Assure imageUrl variable is assigned |
| testPostedBy | Assure postedBy variable is assigned |
| testAlertType | Assure alertType variable is assigned |
| testPriceInCents | Assure priceInCents variable is assigned |

|  |  |
| --- | --- |
| **Name** | **Scope** |
| FileManagerSpy | Used to store number of alerts saved |
| PostSpy | Used to store number of alerts being posted |
| AlertType (Mockito) | Used to set alert type for alerts |

# **Task 2**

Github repository: <https://github.com/robbyfa/marketAlertTask2>

Google Drive: <https://drive.google.com/drive/folders/1wZVTt368XZxm4r4mASsn_p7OxxzBagyF?usp=sharing>

With regards to the SUT to be tested, all methods are declared within one ‘MarketAlert’ class. All feature files have been stored in the ‘resources’ folder in the test suite and all respective steps and runners have been stored in separate folders, i.e., the steps and runner for test1 are stored in the same package, while the steps and runner for test 2 are stored together in a different package. This was done to allow different features to have identical steps and thus avoid repetition of similar methods in the SUT. The project architecture can be seen below in Figure 6.

Text

Description automatically generated [Fig. 6]

Upon running tests, as seen in figure 7 below, all 5 tests are successful.

A screenshot of a computer

Description automatically generated with medium confidence[Fig. 7]

With regards to the testability of ‘marketalertum.com’, it is suggested that an alert counter element is implemented in the ‘My Alerts’ page to allow quicker verification when verifying the number of alerts published.