# **Task 1**

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

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.

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.

Table

Description automatically generated[Fig.2]

Text

Description automatically generated[Fig.3]

Text

Description automatically generated[Fig.4]

Text

Description automatically generated [Fig. 6]

# **Task 2**

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

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]

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.