L-Università ta' Malta
Faculty of Information & Communication Technology

Department of Computer Information Systems

Software Testing

Nicole Abdilla (0142902L)
B.Sc. (Hons) Computing and Business

Study-unit: **Fundamentals of Software Testing**
Code: **CPS3230**
Lecturer: **Dr Christian Colombo**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I, the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include 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.

Nicole Abdilla	
Student Name	Signature
Student Name	Signature
Student Name	Signature
Student Name	Signature

CPS3230 — Course Code

Fundamentals of Software Testing — Title of work submitted

18/12/2022 — Date

Table of Contents

1 Project

This section includes the links to access and download the source code for second and third section of the presented report.

The GitHub link to the project is below:

https://github.com/nicoleabdilla20/cps3230_softwaretesting2

The Google Drive link of the project is the below:

https://drive.google.com/drive/folders/1Ih65jQWFwMSesCSeD4ZzYCIRTbv0ZTRJ?usp=sharing

Accessing the Project

The below are instructions to load and run the tests of the project.

1. Download Folder from one of the above URLs.
2. Unzip Folder
3. Launch eclipse
4. Open folder part2
5. Run the project
6. Launch IntelliJ
7. Open folder part 3
8. Run the project from the test sub-folders

2 Specify the System

2.1 Planning Stage

The below are the planning steps taken before the design decisions. The Gherkin used in the first part of the assignment were copied to this section and started converting the Gherkin tests into monitors by following instructions found on [1, 2].

Scenario: Test 1: Valid Login

Given I am a user of marketalertum

- Observed navigate to https://marketalertum.com/

When I login using valid credentials

- Observed navigate to https://www.marketalertum.com/Alerts/Login
- Observed click on input field and enter credentials

Then I should see my alerts

- View Alert page

Scenario: Test 2: Invalid Login

Given I am a user of marketalertum

- Observed navigate to https://marketalertum.com/

When I login using invalid credentials

- Observed navigate to https://www.marketalertum.com/Alerts/Login/Login
- Observed click on input field enter bad credentials

Then I should see the login screen again

- Observed navigate back to https://www.marketalertum.com/Alerts/Login

Scenario: Test 3: Alert Layout

Given I am an administrator of the website and I upload 3 alerts

- Observed accessed Ebay
- Searched through Ebay
- Got details from Ebay
- Posted 3 alerts to https://www.marketalertum.com/Alerts

And I am a user of marketalertum

- Observed navigate to https://marketalertum.com/

When I view a list of alerts

- Observed navigate to https://www.marketalertum.com/Alerts/Login/Login
- Observed click on input field enter credentials
- View 3 Alerts https://www.marketalertum.com/Alerts

Then each alert should contain an icon

- View alert detail contains icon

And each alert should contain a heading

- View alert detail contains heading

And each alert should contain a description

- View alert detail contains description

And each alert should contain an image

- View alert detail contains image

And each alert should contain a price

- View alert detail contains price

And each alert should contain a link to the original product website

- View alert detail contains link

Scenario: Test 4: Alert limit

Given I am an administrator of the website and I upload more than 5 alerts

- Observed accessed Ebay
- Searched through Ebay
- Got details from Ebay
- Posted >5 alerts to https://www.marketalertum.com/Alerts

Given I am a user of marketalertum

- Observed navigate to https://marketalertum.com/

When I view a list of alerts

- Observed navigate to https://www.marketalertum.com/Alerts/Login/Login
- Observed click on input field enter credentials
- View 5 Alerts https://www.marketalertum.com/Alerts

Then I should see 5 alerts

- View 5 alerts

Scenario Outline: Test 5: Icon check

Given I am an administrator of the website and I upload an alert of type <alert-type>

- Observed accessed Ebay
- Searched through Ebay
- Got details from Ebay
- Posted 1 alert to https://www.marketalertum.com/Alerts

Given I am a user of marketalertum

- Observed navigate to https://marketalertum.com/

When I view a list of alerts

- Observed navigate to https://www.marketalertum.com/Alerts/Login/Login
- Observed click on input field enter credentials
- View 1 Alert https://www.marketalertum.com/Alerts

Then I should see 1 alerts

- View 1 alert

And the icon displayed should be <icon file name>

- Details of icon should match icon set by admin

2.2 Design Decisions

There were two FSM diagrams designed to not overcrowd the diagram, which makes it difficult to read. The first diagram was designed based on the operations carried out in the first and second test scenarios. Figure 1 starts from a transition condition when a user takes the decision to access marketalertum.com where they are welcomed by the *Home Page* of the website. This decision was made by following the example presented in a lecture. It included a vending machine where the input were the coins. In this case, the input is a user's decision to navigate to the Home Page. The transition condition could have been left out in this diagram however, this makes it more readable to understand from where the FSM starts. The next condition is when the user navigates to the *Login Page,* where they enter their details. If the login is unsuccessful, there is an arc back on itself because the action was unsuccessful and directed again to the *Login Page* state. If login is successful, the user is directed to the *Alerts Page* state. In this state there is an arc back on itself because the user will view a number of alerts, noted by *num* in the diagram and the alerts details on the current state the user is at. Details refer to having an icon, heading, description, image, price, and link of a product.

The below is a list of assumptions made for Figure 1:

- Assumption 1 - Assume no issues encountered when accessing https://www.marketalertum.com
- Assumption 2 - Assume a user is logged out and not already signed in
- Assumption 3 - Assume that alerts' details are correct

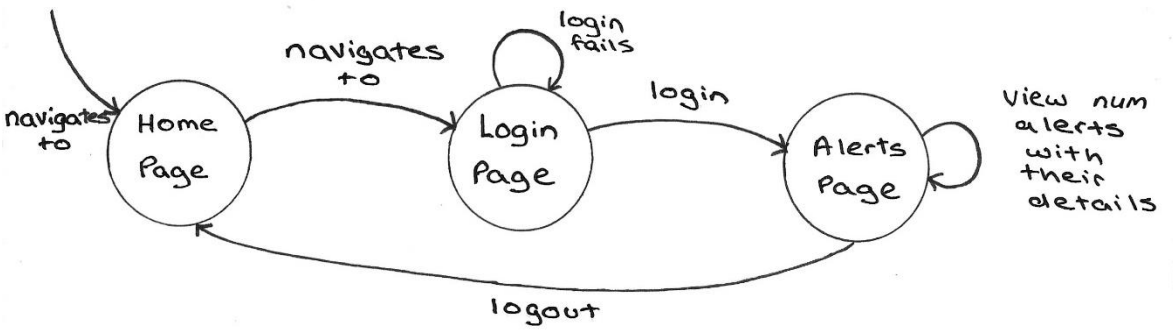

Figure 1 – Market Alert UM

The illustrated Figure 2 diagram starts with a transition condition where an admin decides to upload a number of alerts that are of a specific type, noted as alert-type in diagram. The first state is accessing the *Ebay Main Page* where a product is searched. The results of the search are outputted known as

the *Result Details Page*. The next transition condition is Query*. This has an asterisk to explain the actions carried out include more than one before proceeding to the next state.

Query*:

- Screen scrape results
- Check if *num* of alerts greater than 5 if true then *num* is equal to 5 alerts
- Post *num alerts of alert-type* from screen scrapped data if *alert-type* not set use default type
- Navigate to https://www.marketalertum.com

The last state, *Market Alert UM,* refers to the Figure 1 FSM process named Market Alert UM. This means the last state will go through all the steps of Figure 1. There is one assumption for Figure 2, which is assuming there are always results displayed for searching for a product.

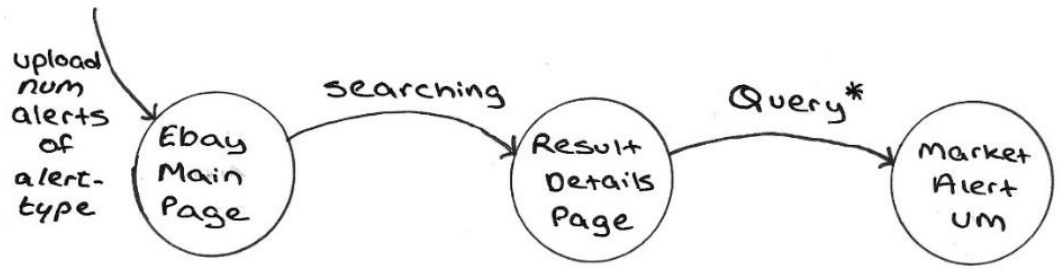

Figure 2 - eBay

However, after the Q&A session and attempting section 2 and 3 had to perform some alternations to the diagrams as below. The below are the final adjusted diagrams. The adjustments performed include:

- Rewording the states and some transitions as displayed in Figure 3, and
- The whole diagram of Figure 2 was changed as it made more sense to perform operations as displayed in Figure 4.

Therefore, Figures 1 and 2 are considered as the first draft of the automata which are more specific to the first assignment. The screen scrapper was not modelled as advised in the Q&A session. The state eBay refers to the process that the actual screen scrapper would have gone through. So, it is representing the website used in the first assignment and the steps mentioned in Query*.

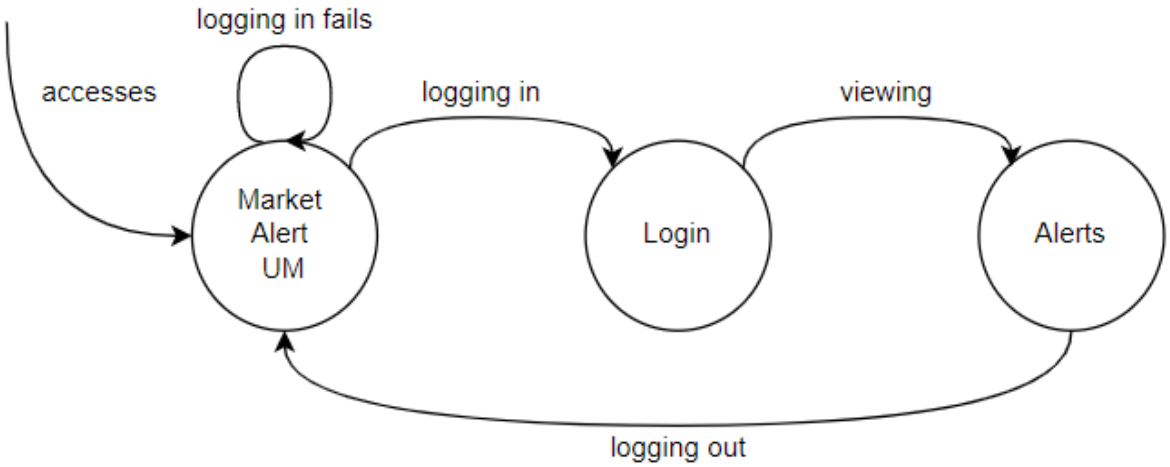

Figure 3 – Market Alert UM FSM Model

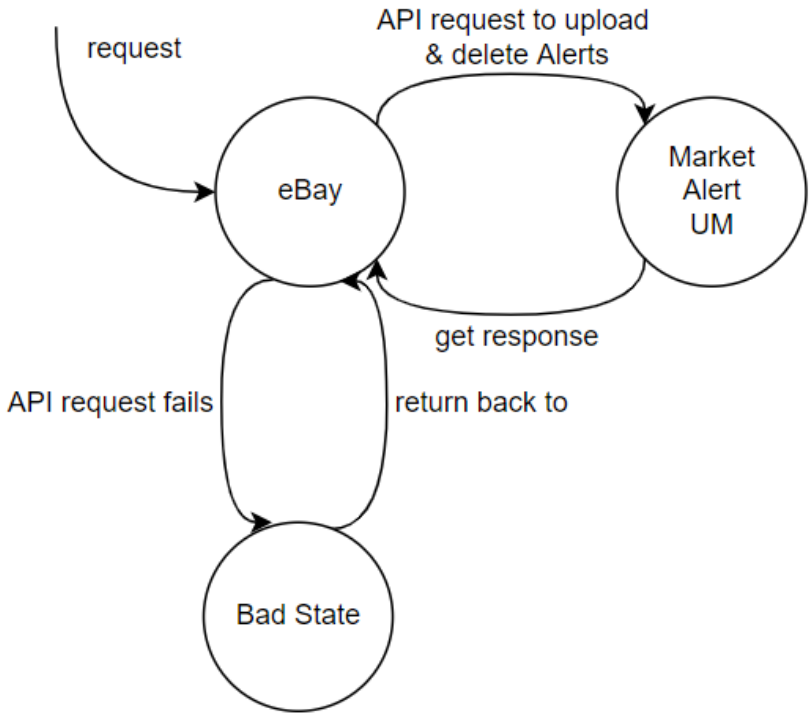

Figure 4 – eBay's website with results and API requests

3 Runtime Verification

3.1 Documenting the steps taken

Accessed marketalertum.com

Went to Postman Get request: https://api.marketalertum.com/EventsLog/5e383da7-d1b0-4b72-b9d3-31c15a9b9e44

Output of Postman for user **5e383da7-d1b0-4b72-b9d3-31c15a9b9e44**


```
[
    {
        "id": "6dc99cde-6b7d-4b2a-9481-c8b202d6b429",
        "timestamp": "2022-11-29T08:41:30.569994Z",
        "eventLogType": 5,
        "userId": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
        "systemState": {
            "userId": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
            "loggedIn": true,
            "alerts": [
                {
                    "id": "c9044341-1bfc-42e6-8794-b17d57208a46",
                    "alertType": 6,
                    "heading": "Jumper Windows 11 Laptop",
                    "description": "Jumper Windows 11 Laptop 1080P Display,12GB RAM
 256GB SSD",
                    "url": "https://www.amazon.co.uk/Windows-Display-Ultrabook-
Processor-Bluetooth",
                    "imageURL": "https://m.media-
amazon.com/images/I/712Xf2LtbJL._AC_SX679_.jpg",
                    "postedBy": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
                    "priceInCents": 24999,
                    "postDate": "2022-11-28T09:38:11.7508074Z"
                }
            ]
        }
    },
    {
        "id": "e10ea6f0-4a4f-4358-af63-d44d49541876",
        "timestamp": "2022-11-29T08:41:30.8946257Z",
        "eventLogType": 7,
        "userId": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
        "systemState": {
            "userId": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
            "loggedIn": true,
            "alerts": [
                {
                    "id": "c9044341-1bfc-42e6-8794-b17d57208a46",
                    "alertType": 6,
                    "heading": "Jumper Windows 11 Laptop",
```

```
                    "description": "Jumper Windows 11 Laptop 1080P Display,12GB RAM
 256GB SSD",
                    "url": "https://www.amazon.co.uk/Windows-Display-Ultrabook-
Processor-Bluetooth",
                    "imageURL": "https://m.media-
amazon.com/images/I/712Xf2LtbJL._AC_SX679_.jpg",
                    "postedBy": "5e383da7-d1b0-4b72-b9d3-31c15a9b9e44",
                    "priceInCents": 24999,
                    "postDate": "2022-11-28T09:38:11.7508074Z"
                }
            ]
        }
    }
]
```


The Java Code for Get request:

```
Unirest.setTimeouts(0, 0);
HttpResponse<String> response = Unirest.get("https://api.marketalertum.com/EventsLo
g/5e383da7-d1b0-4b72-b9d3-31c15a9b9e44")
  .asString();
```

The next steps taken were creating an AspectJ project folder in eclipse. Issues encountered included the version of eclipse since the PC used was blocking the software from installing. The issue was resolved by downloading it from another search engine and it worked. Overall, this task was challenging in understanding the requirements, as it was not clear what was required. This task was attempted by trying the best effort.

The below are some snippets of the outputs.

```
[MarketAlertUMProperty]AUTOMATON::> MarketAlertUMProperty() STATE::>MarketAlertUM
Good login observed. Good logins: 1
[MarketAlertUMProperty]MOVED ON METHODCALL: void main.MarketAlertUM.goodLogin() TO STATE::> Login
[MarketAlertUMProperty]AUTOMATON::> MarketAlertUMProperty() STATE::>Login
Viewing alerts ... true
[MarketAlertUMProperty]MOVED ON METHODCALL: void main.MarketAlertUM.view(boolean) TO STATE::> Alerts
[MarketAlertUMProperty]AUTOMATON::> MarketAlertUMProperty() STATE::>Alerts
Logged Out!
[MarketAlertUMProperty]MOVED ON METHODCALL: void main.MarketAlertUM.logout() TO STATE::> MarketAlertUM
```

Figure 5 - MarketAlertUM's output

```
[eBayProperty]AUTOMATON::> eBayProperty() STATE::>eBay
Good API request: 1
[eBayProperty]MOVED ON METHODCALL: void main.eBay.goodRequest(boolean) TO STATE::> MarketAlertUM
[eBayProperty]AUTOMATON::> eBayProperty() STATE::>MarketAlertUM
Getting Response
[eBayProperty]MOVED ON METHODCALL: void main.eBay.getResponse(boolean) TO STATE::> eBay
[eBayProperty]AUTOMATON::> eBayProperty() STATE::>eBay
Bad request observed. Bad requests: 1
[eBayProperty]MOVED ON METHODCALL: void main.eBay.badRequest(boolean) TO STATE::> !!!SYSTEM REACHED BAD STATE
aspects._asp_scraper0.ajc$before$aspects__asp_scraper0$5$e9e987b5(_asp_scraper0.aj:52)
main.eBay.run(eBay.java:30)
main.eBay.main(eBay.java:51)
[eBayProperty]AUTOMATON::> eBayProperty() STATE::>BadState
Returning to scraper ... true
[eBayProperty]MOVED ON METHODCALL: void main.eBay.returnTo(boolean) TO STATE::> eBay
[eBayProperty]AUTOMATON::> eBayProperty() STATE::>eBay
```

Figure 6 - eBay's output

4 Model-based Testing

The model tests were tested by using *Asserts.assertEquals* and when it is not true, a bug will rise, letting the developer know at which state it was detected. For this report, a line of code was adjusted to raise a bug. The below was displayed:

```
done (MARKET_ALERT_UM, loggingIn, LOGIN)
FAILURE: failure in action viewing from state LOGIN due to java.lang.AssertionError: The SUT state does not match to the model's Alert state.. Actual: true
```

Therefore, if a bug occurs the code should be able to detect it and raise it.

4.1 Results

MarketAlertUM which is portrayed in Figure 3.

Action coverage: 4/4

State coverage: 3/3

Transition-pair coverage: 6/6

```
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, loggingIn, LOGIN)
done (LOGIN, viewing, ALERTS)
done Random reset(true)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, loggingIn, LOGIN)
done (LOGIN, viewing, ALERTS)
done (ALERTS, loggingOutFromAlerts, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, loggingIn, LOGIN)
done (LOGIN, viewing, ALERTS)
done Random reset(true)
done (MARKET_ALERT_UM, badLoggingIn, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, loggingIn, LOGIN)
done (LOGIN, viewing, ALERTS)
action coverage: 4/4
state coverage: 3/3
transition-pair coverage: 6/6

Process finished with exit code 0
```

Figure 7 – Results of Market Alert UM Test

The screen scrapper used to scrape data from eBay as portrayed in Figure 4.

Action coverage: 4/4

State coverage: 3/3

Transition-pair coverage: 6/6

```
done (MARKET_ALERT_UM, response, EBAY)
done (EBAY, badApiRequest, BAD_STATE)
done (BAD_STATE, returnToScraper, EBAY)
done (EBAY, goodApiRequest, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, response, EBAY)
done (EBAY, badApiRequest, BAD_STATE)
done (BAD_STATE, returnToScraper, EBAY)
done (EBAY, badApiRequest, BAD_STATE)
done (BAD_STATE, returnToScraper, EBAY)
done (EBAY, badApiRequest, BAD_STATE)
done (BAD_STATE, returnToScraper, EBAY)
done (EBAY, goodApiRequest, MARKET_ALERT_UM)
done (MARKET_ALERT_UM, response, EBAY)
done (EBAY, badApiRequest, BAD_STATE)
done Random reset(true)
done (EBAY, badApiRequest, BAD_STATE)
done (BAD_STATE, returnToScraper, EBAY)
done (EBAY, goodApiRequest, MARKET_ALERT_UM)
action coverage: 4/4
state coverage: 3/3
transition-pair coverage: 6/6

Process finished with exit code 0
```

Figure 8 – Results of eBay Test

Overall, this assignment had its challenges however it was attempted by completing all the requirements. This project has learnt the individual about new technologies and tools available.

5 References

[1] - C. Colombo, M. Micallef, and K. Spiteri, “Extracting Runtime Monitors from Tests.” Accessed: Dec. 11, 2022. [Online]. Available: http://staff.um.edu.mt/__data/assets/pdf_file/0003/273396/KeithSpiteriWhitePaper.pdf

[2] - “L06: Finite State Machines,” *computationstructures.org*. https://computationstructures.org/lectures/fsm/fsm.html (accessed Dec. 11, 2022).

[3] - C. Colombo, and L. Chircop. Model Based Testing Tool Examples – ModelJunit [PPT document]. Available: https://www.um.edu.mt/vle/mod/resource/view.php?id=945578