Web Scrapper Documentation

# Task 1

## Implementation

Scan Malta was selected as the target for scrapping data. It is important to note that the image link is scrapped successfully but due the image is still unable to be displayed in the website.

The following classes were created for creating a web scrapper and posting program:

|  |  |
| --- | --- |
| ScrapAndPostClass | This class contains a method inside of it which connects various methods. It scraps and posts alerts based on the amount provided (int numberOfScrapes). |
| Variables | Contains a list of variables used throughout the program, such as links, login IDs, etc. |
| WebClientConfig | Used to connect to the web, and more specifically the website from which data is scrapped. |
| Scrapper | Scraps the data from the web. Uses “getByXPath” to find the proper css elements that we wish to scrape and save them in appropriate variables. |
| Transcript | Populates the class Transcript with data obtained from Scrapper. All fields are filled in from scrapper except for “postedBy”, which is already predefined in the Variables class. |
| Conversion | This class is used to convert “Transcript” into JSON using Gson. The returned jsonRequest will be used to post the alert type. |
| PostClass | Contains the method postToWeb which uses REST API to post a JSON request |

## Design

It was decided to split the program in various classes to make it clear what the methods are performing. ScrapAndPostClass is the class that calls all the methods together to make everything work.

The class was also designed in a way such that when the method is called, the user can specify the number of scrapes and posts they wish to do, and they can also easily alter the alert type. The default alert type is 1.

WebClient was used to setup the connection with the Scan website.

# Task 2

## Implementation

The test cases given were separated into three separate features:

* Login.feature: Contains Test 1 & 2
* Alert.feature: Contains Test 3 & 4
* Alert.feature: Contains Test 5

All features are run from the same TestRunner class.

For this task, an additional class, “WebAutomation” was created. This class uses WebDriver and contains a method “userLogin(String userID)” which is used to login to the website using credentials userID. This method is essential for tests 1 and 2 which deal in checking the user login.

Additionally, in the PostClass class, a deleteFromWeb() method was created, which uses Rest API to purge the website from any alert notifications that may be present. This method is used in the setup phase.

## Video Demonstration

Link: <https://drive.google.com/file/d/1e87xC49pi-VhgMvuuny__dAo8hcqV5hB/view?usp=share_link>

## Key Decisions Taken

Setup & Teardowns

In the setup, before each case, the list of alerts is purged from the website. This was done so that the test cases are always working on a blank slate. For example, when checking that 3 alerts have been uploaded successfully to the website, it would not make sense if there were already 2 alerts present, making a total of 5 alerts visible on the page.

Test case changes

Certain test cases were slightly altered from the requested one, to make allow the test to run better. For example, in the 4th Test given (Alert limit), an extra step was added called, “Then I should see 3 alerts”. Additionally, the previous, “Then each alert should contain an icon” was altered to be “And each alert should include an icon”.

This change was done so that in the new step, the program can check that when the administrator uploads 3 alerts, the website does indeed show 3 alerts. Additionally, a counter is also present in this step to count that each alert contains icons, heading, description, etc.

## Testability

Checking for elements on a website with no IDs present was not ideal. This caused situations in which the test case was searching for element “//table[1]/tbody/tr[1]/td[1]/h4/img” instead of simply using the ID. Ideally, headings, icons, descriptions, etc. should have IDs that properly relate to what they are referring to.

While testing the website, it was also found that two icons, “icon-property-rent” and “icon-property-sale” were of type JPG not PNG. Proper corrections should ideally be implemented to make sure that all icons are indeed of type PNG.

## Test Coverage

After all testing is done, 77% of all classes along with 94% of lines were covered. The only class which was not covered is the Variables class as this contains no methods, but rather various variables which are used throughout the program.

Text

Description automatically generated

## Code

Github link: <https://github.com/neilBugeja00/ScreenScrapper>