Sheffield Hallam University

Faculty of Arts, Computing, Engineering & Sciences

Department of Computing

Final Year Individual Project (SEGM)

[55-6727]

2016/17

Author:	**Steven Stretton**
Date Submitted:	**24/04/2017**
Supervisor:	**Tony Day**
Degree Course:	**BEng Software Engineering**
Title of Project:	**The development of a robust test automation library for Selenium WebDriver application**

Confidentially Required?

NO ☐

YES ☐

Sheffield Hallam University

The development of a robust test automation library for Selenium WebDriver application

Steven Stretton BEng (Hons) Software Engineering

Abstract

This work introduces a new automation test library to be conducted by quality assurance software engineers. It assesses the problems faced during the change process when webpages are altered, such as where a web form is upgraded from a basic HTML layout a bootstrap style page. Various testing libraries were examined including Cucumber (Java) and PhantomJS (JavaScript), with a common problem of webpage modifications A contemporary test procedure would automatically fail even if the website functionality is persistent. Therefore, solutions were developed, showing the design process written in Java, using agile project management. Using a Maven build tool on three test forms, the tests were successfully executed on the three test websites using the new automation library and no webpage modification was necessary. The development of the new automation library concept was successful as it eliminated the process of finding a path to an element, therefore it would reduce time of development and would lead to earlier releases of software for a business.

Preface

This report describes project work carried out within the Department of Computing at Sheffield Hallam University between September 2016 and April 2017.

The submission of the report is in accordance with the requirements for the award of the degree of BEng (Hons) Software Engineering under the auspices of the University.

This project was inspired by work done during placement in Engineering and Operations department at Plusnet Plc.

Table of Contents

1. Introduction

The aim of the project research is to develop of a robust test automation library for Selenium WebDriver application.

Testing frameworks have a part for developers to set conditions and execute tests onto applications such as websites, providing reassurance to the developer that the product developed fully functions as intended (Jamil, Arif, Abubakar, & Ahmad, 2016). There are various testing frameworks available for different platforms, ranging from Java to TypeScript that covers various techniques within the test driven development (TDD) domain, including behaviour driven development (BDD).

Selenium WebDriver is a testing framework which covers web based applications, written in various languages. As the Maven build tool is going to be used, the WebDriver will be using the Java client. The tests use Selenium to interact with content on a webpage being tested, including read, write and selecting content as if an end user was using the webpage in real life. The driver does have limitations regarding website modifications, if a webpage gets modified with new content, Selenium does not pick this up and requires test modifications per page element modification. Leading towards test failures and increasing the possibility of product release delays for a business.

From observations of the Selenium WebDriver, it relies on a hardcoded path such as CSS paths and XPath that links to a specific element on a page. A path represents a single element on a webpage, the hardcoded path is entered in manually by the developer. The path is set as part of a precondition so upon execution of the test, Selenium gets the path and sends it to the driver on the browser, where it selects the element on the page based on the value of the path. Any modifications to the page can change the path of the control, rendering the hardcoded path in the tests as incorrect, leading to test failures.

End users (including developers) can easily identify and interact with specific elements on a modified webpage of a website, prompting a question of if an end user can identify an existence of a webpage element, why cannot an automation system?

Automation libraries such as selenium are too dependent on the end user proving information of a webpage that it is testing for each element, in a state during implementation. When a

webpage is updated, depending on the enhancements carried out, the structure is gone. So, as an end user can see that a webpage element exists by visual observations, Selenium tests structure of the element has been changed, causing test failures.

A resolution to the issue is by developing a proof of concept of a new automation library, where it generates and send a path to the Selenium WebDriver. The automation library will be written in Java, along with the tests with the test sites written in hypertext mark-up language (HTML), deployed on a virtual server on a local machine to simulate the testing process in industry.

The outcome of the concept will be measured by observing the pass and fail rate between pre-library implementation and post-library implementation. With the expectation that the tests using the current Selenium WebDriver implementation will generate failures majority of the tests, and with the new automation library having a small or non-existent failure rate.

2. Problem

Selenium WebDriver is designed to automate webpages through a browser, providing a realistic representation as the driver is acting as a user selecting content on a page. For the WebDriver to find and interact with content (such as text input and button selection), an accurate path to the element is required. Selenium WebDriver supports both CSS selector and Xpath, which is required for a consistent structure of the webpage.

Carrying out modifications to a webpage which involves changing the html structure will have an impact on existing tests written up prior to the website modifications, due to CSS paths losing the structure of the specific target, resulting in test failures when executed. Software developers can identify a selenium test failure relating to a CSS path with a termination of the test along with an error message output on a console log (Figure 1).

Selenium can find elements with HTML tags, however an Automation Engineer is required to explicitly inform the test what HTML tag it must find to collect the attribute. Figure 1 shows the HTML tag of name contained in an id, however a case that a HTML file has a modified structure that an id tag no longer exists, but placed in another tag leading to Selenium unable to find the attribute. Selenium has become too dependent to accurate HTML tags where if a developer doesn't tell the test what HTML tag the attribute is in, Selenium cannot find it and the current driver does not have the functionality to scan and find the correct element.

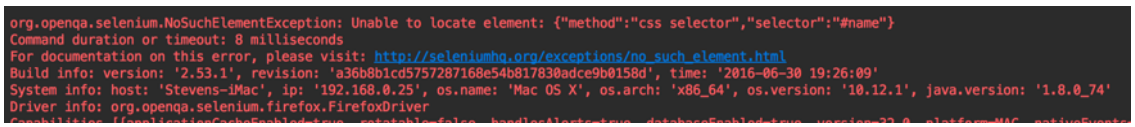


Figure 1: Selenium WebDriver automation test error displayed on IntelliJ IDE.

Automation testing is not just confined to an Automation Engineer or Quality Assurance. Developers may be restricted in creating dynamic website applications to replace existing websites that may not have the advanced functionality. The reason for the restrictions may vary from industry to industry, but the pressure of deadlines reduces the time for development. In some cases, developers may do a quick fix on the HTML structure for a test to pass on a modified webpage, including adding id tags to elements where CSS selectors fail to locate elements. However quick fixes can lead inconsistencies to the structure of the code, causing coding practices that may be applied to both an organisation and a key programming rule (Hegedüs, Horváth, & Ráth, 2011), risking infringement on a well-functioning system in release due to improving the test results.

Project managers fear projects could be delayed due to tests needing to be rewritten for significant webpage modifications, therefore adding pressure to software developers to complete a task within a limited period (Herzig, 2014). Another matter for the developer is ensuring that the automation test must be able to identify structural and content issues on a webpage, however at the same time the test must not purposely fail a page where there is nothing wrong (Vitó Ferreira & Langerman, 2014).

3. Hypothesis

3.1. Solution

To resolve the shortcomings of automation testing using Selenium WebDriver, a development of a new library is required, with the functionality of automatically generating a CSS path based on the structure of the HTML of a webpage during a test. The CSS path generated means that the developer is no longer required to collect a path manually using another application such as a Firefox plugin Firepath. The new library aims to eliminate a reiterating process for the developer of manual path collection.

It is intended for the library to generate a path to an element, and to minimise the complexity for the developer to pass the least quantity of values into the library. Rather than the developer using a WebElement object, a new object in the new automation library is used instead, specifically designed to collect values. The library handles the call to the WebElement once the path has been generated for a specified element.

Once the automation library generates the path, it is intended for the path to be outputted onto a terminal during any execution of the code (debugging through IDE or running through Maven). Outputting the generated paths can aid the developer to identify two scenarios; firstly, the developer can see any tests that have passed, along with any missing ids that the developer may have missed during development as the library will not be dependent on what tag should be collected. Secondly, the developer can find where a test may have failed, as the system outputs one path at a time. Identifying where a failure occurred would be significantly narrowed down.

The development of the library is aimed towards reducing the time required for developing automation tests. One key area is cutting the quantity of code that a developer needs to input. Currently Selenium tests need to include the name of the CSS tags as part of a WebElement, which the developer needs to collect manually (shown on Figure 2), and the setup needs to be done before the WebElement object can be called during the testing process. As a new library will provide its own object to replace WebElement, an aim of reducing the number of lines needed to call an element from 3 to 1, is to benefit the promotion of clean code and understanding of the system for other developers (Vasileva & Schmedding, 2016).

```
11      @FindBy(css = "#name")
12      private WebElement formName;
```

Figure 2: CSS tag name along with the WebElement name.

3.2. Probable Issues

Although the automation library would be able to find the elements and generate a path, there are some risks that may transpire during development. One risk highlighted is the possibility of the HTML structure containing multiple elements of the searched item, which can come in various forms. The multiple elements issue is demonstrated on a test page (Figure 3) displaying a label tag with a 'for' property containing the name.

```
<div class="form-group">
    <label class="col-md-4 control-label" for="name">Name</label>
    <div class="col-md-4">
        <input id="name" name="name" type="text" placeholder="your name here..." class="form-control input-m
    </div>
</div>
</div>
```

Figure 3: HTML structure with a bootstrap label and text box

However, the test may mistakenly identify the label as the element to enter in a value, which for the label demonstrated would be incorrect. To avoid the issue of selecting an incorrect label, the library needs to be able to identify the difference between a label and input tag, but the properties that the tags contains, can assist the system to know which tag to filter down to.

A possible solution is to create a filter consisting of HTML tags that the library should ignore during the search for the property, and as the system needs to be independent from which HTML tags it uses. It should be noted that it is likely that the library could disregard a correct HTML tag due to an inconsistent structure. Therefore, greater care and through testing of the application must be considered.

4. Research

4.1. History of Automation Testing

Automation testing has been around for over 30 years, with a common aim that has been persistent throughout all projects, to reach an achievable objective set by the developers (Graham & Fewster, 2012). Early automation testing was rudimentary, where manual testing followed with a straight to release of products was a common practice. When client and server based architectures were introduced, it caused an increase in software density, leading to an increase in the quantity of tests required (Horne, 2014).

The concept of Selenium WebDriver goes back the applications predecessor Selenium Remote Control (Selenium RC). Selenium RC was released back in 2004 by Jason Huggins (Selenium Project, 2012). Huggins developed the concept of developing a JavaScript-based library to dynamically interact with contents on a webpage during a project when Huggins wanted to find a better method of going back to the same test during every modification.

On the first release, Selenium RC (as part of Selenium 1) consists of a server (known as Selenium Server) and a client. The server associates the test (e.g. webpage) with the client, whereas the client proves the API (Application Programming Interface) that passes the commands to the server (Altaf, Dar, & Rashid, 2015). The release of Selenium 2 included greater framework integration and migrated Selenium away from RC to WebDriver, where it has greater browser vendor participation (Selenium Documentation, 2017).

4.2. Version Control Systems

Version control systems (VCS) is a tool that enables the development of code and allows separation of various coding tasks on the same project. The use of agile methodology such as Scrum can be dependent on version control projects as it allows groups to be able to work on implementing different features onto a code without the worry of saving and merging all the changes together. Version control doesn't just benefit group work, individual projects can manage their code more effectively so a developer can split up their tasks and create a log of what code has been implemented and to check for any outstanding tasks.

Research prior to development of the automation library was conducted to establish the best VCS to use. Three applications were selected based on reliability and performance, the applications include GitHub (see Appendix C, Section 1, Part A), Bitbucket (see Appendix C,

Section 1, Part B), and GitLab (see Appendix C, Section 1, Part C). To ensure a consistent assessment, all VCS will be assessed by the services offered from the basic and free plans.

Based on functionality and privacy, the assessment concluded that Bitbucket was the preferred version control for the project. Bitbucket had excellent integration with Git for both command line and SourceTree (which is also built by Atlessian). Unlike GitHub, the privacy of Bitbucket show that any pushed code won't be made public without the secure shell (SSH) address.

4.3. Existing Automation Products

There are multiple testing tools available for various programming languages, ensuring software integrity and developer confidence to ensure that the software produced correctly functions as intended (Yang, Li, & Weiss, 2009). Two automation tools will be examined on functionality and shortcomings where the new automation library could resolve.

4.3.1. Cucumber

Cucumber is an automation script that writes automation tests in plain English rather than a programming language. Therefore, it is good for quality assurance (QA) engineers who are not fully competent using programming languages. It still relies on the Selenium WebDriver to interact with the webpages for the Automation Tests (Sypolt, 2016).

Cucumber specifically outlines what the developer is intending to test, however it still relies on the Selenium WebDriver to find elements which are entered by the developer (Pannu, 2015). Although Cucumber is an automation script to enter in commands, these commands become redundant if the website has been upgraded, as Cucumber has not been designed to locate web elements. It cannot read the structure of the webpage but Selenium still expects an explicit path to the web elements.

Cucumber could potentially be integrated with the automation library as the developer would use the Cucumber framework to write and execute tests. The developer would need to test, but would not be required to interact with the web elements directly.

4.3.2. PhantomJS

PhantomJS is an automation library used in JavaScript to test the webpage content the same way as the Selenium WebDriver used in Java. Unlike Selenium which relies on a web browser,

PhantomJS simulates a web browser and directly interacts with the webpage (Christophe, De Roover, & De Meuter, 2013).

PhantomJS relies on the web page as the source of the test. An experienced JavaScript developer would need to write the PhantonJS tests, although the syntax is slightly different from Java based Selenium WebDriver test. Like Selenium, PhantomJS requires all the elements from the page to be entered in explicitly so any future changes on a page will still need to be updated.

4.4. Website Evolution

Websites are in constant competition with each other, especially as much of the global economy is online and this is expected to grow. Figures published by the Office for National Statistics (ONS) (2016) shows online activities ranging from goods and services to news has grown over 30% since 2007. The increasing number of online users has led to greater competition among rival organisations trying to attract more visitors. In response, websites have evolved with content modification and new scripting technologies.

Website design has constantly changed since the beginning of the information age, and in the last 5 years the design has been adapted for mobile as well as desktop. Early websites consisted of a basic structure of HTML and scripting languages such as Java were introduced in the early 2000s in part of IPv6 (Graba, 2007). As computer specifications increase, websites have become more dynamic with content plugins by software developers such as Macromedia, and the success has led to the company being brought by Adobe for $3.4 billion (£2.7 billion) (Adobe, 2005).

Recently, websites are increasingly being navigated with direct user interaction on a touchscreen portable tablet and smartphone device, and a well-designed website is quickly judged by an end user (Lombardi, 2014). The rise of responsive websites demonstrates that it is possible to have one webpage that works on any device regardless of resolution. However, research done by Glassman and Shen (2014), found developers faced the challenge of planning their website designs and content in a smaller scale compared to a few years ago, when developers had a set dimension for a website (1200px by 800px) to work on.

The material on websites vary over the years, but most websites retain their structure so users that are familiar or new to the website can find the content with ease. In relation to the new automation library, the system will need to be dependent on the structure as a comparison of

each HTML tag to search for the keyword entered by the developer. However, the new library is not dependent on how the structure is set out, as the system can be able to get the HTML structure by calling a function to get a source in a FirefoxDriver object.

The comparison has been conducted on four BBC homepages (Table 1) shows that even if the site has been updated with a different layout and appearance, the expected content is still there. All four pages contains keywords that would uniquely identify what it is (e.g. links to news, weather, etc.).

	Figure 4	Figure 5	Figure 6	Figure 7
Responsiveness	No responsive functionality	No responsive functionality	Introduction of responsiveness to the website	Page still retains its responsivity
Structure	Header with all the main links (e.g. news, sport). Body which shows the main content	No major alterations to the structure despite design modifications	There was a difference from a user prospective of the website with new responsive and interactive features.	Reduction interactivity from the same page a year ago
Links		Retained link from 2010, although major key links remain (e.g. news)	Still clear to find important information that is both present the 2010 and 2013 BBC homepage (such as news and sport)	Links location are same as in the previous figures, and in same area of the page as previous.

Table 1: Comparison of four versions of the same website, upgraded over the past 5 years

Figure 4: BBC Homepage dated from 31st July 2010

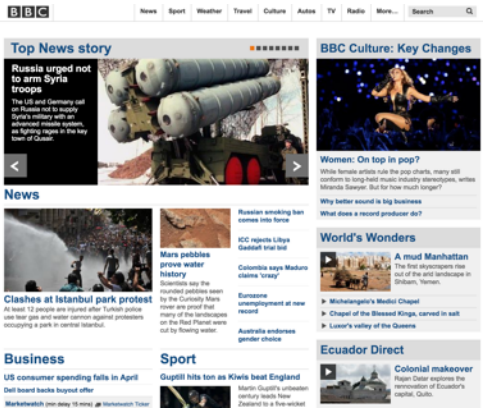


Figure 5: BBC Homepage dated from 1st June 2013

Figure 6: BBC Homepage dated from 14th June 2015

Figure 7: BBC Homepage dated from 15th November 2016

Based on the comparison of the four pages, the analysis has highlighted an important question, if an end user can find the content on the page, why does the automation test fail to find it? Selenium gets the structure from an exact path by the developer entering the pointer manually, pointing to the element. If the developer changes any properties of the HTML tag (e.g. id), the pointer may become redundant depending on the properties the pointers path contains. It is imperative that the new library can construct a consistent pointer that can be used on all controls.

4.5. Business and IT project impact

4.5.1. Importance of testing

Businesses can have a positive impact with automation testing, with assistance in finding some bugs before release therefore seeing a reduction in cost, and providing confidence to the developer ensuring that the code written functions as intended (Bartley, 2014). Testing is the last stage of software development before any product is released, with agile teams running tests of the teams' concern, before going to an infrastructure team who would run a full regression with every other teams' changes in place.

Testing applies to all types of industries; however, it can depend on the properties of the application along with the industry. Software quality standards can help to identify the level impact an application can have, with some failures causing negative effect towards the end user (customers), and the rest to the software developer which can link back to providing confidence to the developer (Galin, 2004).

4.5.2. Impact of release

Research from the Standish Group (2014), has highlighted issues regarding overruns of projects. The research suggested that both full and iterative rollouts can put pressure on developers, ensuring the releases are all tested and put into release. Any modifications of content for the release is prone for the automation tests needing modification, therefore increasing the prospect of an overrun and causing operational and cost implications to a business (shown on Figure 8). The overrun for automation testing depend on the nature of the test, some modifications can take a few days to over a week. The QA engineer will be required to modify the test plans and rerun all the tests, including untouched tests to ensure no new failures are present (Holtsnider, Wheeler, Stragand, & Gee, 2010).

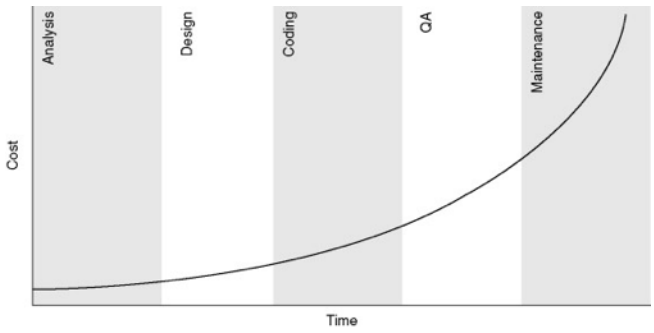


Figure 8: Cost of change curve (Holtsnider, Wheeler, Stragand, & Gee, 2010).

Based on the analysis of release impact to a business, the new automation library will aim to resolve the issues highlighted. One key area is for the new library to focus on is to reduce the

length of time a test takes to be modified, which can be achieved by eliminating a process of the developer needing to find the full path to individual HTML elements such as button and textboxes.

5. Approach

A java-based automation library has been developed which assists in building of a robust Webdriver based automation test suite that can adapt to changes onto a webpage. Each test website will need to be able to have the same behaviour from each other in terms of what the site is doing, so if the site is filling and submitting a form (with the same information) then any enhanced site must do the same thing. The objective is for the automation tests to pass a test consisting of three websites, each of them contains different structures in both front end and backend.

5.1. Methodology

The automation library has been developed in Java using IntelliJ integrated development environment (IDE), with the whole development being made on Mac OS Sierra operating system. The project files (which contains the library), consists of a Selenium WebDriver and FirefoxDriver. Selenium WebDriver is an application programmable interface (API) that can natively drive a web browser in the same way as a user (Selenium WebDriver, 2017). The driver contains various browser drivers including Chrome, but a decision was made to use Firefox driver due to greater performance and stability.

A version control system was used to allow the separation of various coding tasks on the same project. The use of agile methodology such as Scrum can be dependent on version control projects as it allows groups to be able to work on implementing different features onto a code without the worry of saving and merging all the changes together. Version control doesn't just benefit group work, individual projects can manage their code more effectively so a developer can split up their tasks and create a log of what code has been implemented and to check for any outstanding tasks.

To ensure that progress is maintained throughout development of the automation library, a Scrum approach was implemented throughout using an online management tool Trello, with monthly Scrum meetings arranged with the project supervisor. Each ticket (represents the task) in Trello is then undertaken by creating a new branch in Bitbucket, by matching up the project information from the ticket (Figure 9) onto the branch name (Figure 10). Upon completion of the

Figure 9: Trello ticket.

task, the branch is then merged onto a develop branch (containing the previous tasks from other merged branches), which is then tested before it finally gets merged with the master branch.

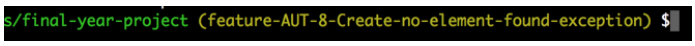


Figure 10: Branch name in Terminal.

Each segment of the scrum board is divided into six segments (Figure 11), representing each stage of the task (Table 2). For each task, a ticket is created representing a specific task for the project, and each ticket has either the two categories (Automation and Web, Table 3).

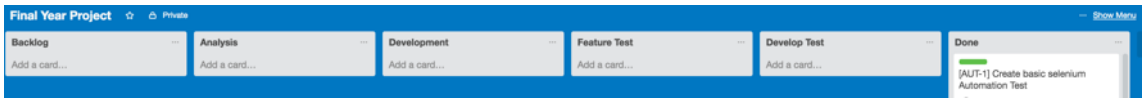


Figure 11: Trello scrum board showing the six segments.

List Name	Purpose
Backlog	Holds all the upcoming tasks that needs to be completed.
Analysis	Checks the validity of the task (e.g. does it still need to be done?), and carry out any required investigation to undertake a task.
Development	Carrying out the development of the task, which can be creating or modifying a piece of code.
Feature Test	Verifies that the developed code works on the current feature branch, and ensures that no other functionality is compromised unless otherwise stated on the individual task. Feature testing must be completed before progressing any further in the process.
Develop Test	The completed feature branch is merged onto the local develop branch, and tested to ensure that no additional functionality is compromised on the develop branch. Once testing is completed and passed, the branch is then pushed onto remote, a force push may be required but only if necessary.
Done	Holds all completed tasks.

Table 2: Trello scrum board list names and details.

Ticket Type	Abbreviation Used	Details
Automation	AUT-*n*	Carry out tasks to impalement features that involves work on the WeleniumLibrary (main and tests) and UIBox (tests only)

Web	WEB-*n*	Carry out tasks which involved the development of the UIBox (web only)

Table 3: Ticket identifiers and details

When a new ticket is created, the value increments depending on the maximum number of tickets for each type. (e.g. AUT-3)

The automation library works by finding webpage elements using the HTML tag properties from a keyword entered in by the end user. During the test execution, the library will collect the HTML source from the Firefox driver, where the HTML source gets split up line by line into an array. Once the split is completed, the library will collect the keyword entered in earlier by the user, and use the keyword to illiterate through each of the HTML tags to find any properties that contains the value of the keyword. If a keyword is found, the iteration process will cease and moves into the path build process, and will start assembling the CSS paths required to point and interact towards the web object, depending on the method the end user has called from the pageObjects. As soon as the CSS path is build, the process will then validate that the path is correct and passes the generated CSS path to the WebDrivers findElement function, where Selenium takes over the rest of the automation process.

5.2. Design

The concept for the idea of the application has materialised from the observations of numerous automation test failures in industry, with some failures occurring due to structural page modifications rather than incorrect data on the page. As it is obvious for an end user to visually identify the location of the content on page such as input forms, libraries such as Selenium WebDriver cannot identify such precise locations, due to its reliance on the precise path to the object based on the HTML structure. Therefore, a concept for an automation library that can work on top of Selenium has been identified to resolve the concern.

To ensure that the full potential of the library can be fulfilled, a single test scenario has been devised for the automation library to be able to identify and interact with the elements included on the page. As a single test is used where it primarily focuses on one behaviour of filling in and submitting a form, different variants of the page was also developing with careful consideration of not changing the functionality and the purpose of the page to ensure a consistent test is maintained. Based on the variants required for an effective test, three web applications that does the same functionality were developed (Figure 12), with each site having its own properties such as any Cascading Style Sheets (CSS) and Javascript that each may use, so no script for each website is used across more than one test site. The three test sites are ordered based on complexity, with Test A being the least dynamic in terms of design and scripting used, whereas Test C is the most dynamic. The variants levels of dynamics across the three websites will demonstrate the level of flexibility the automation library can undergo. Test B and Test C are both written in bootstrap, however to test the automation flexibility, the id tag attributes on the HTML in Test C was removed and instead the library should identify other HTML tag attribute.

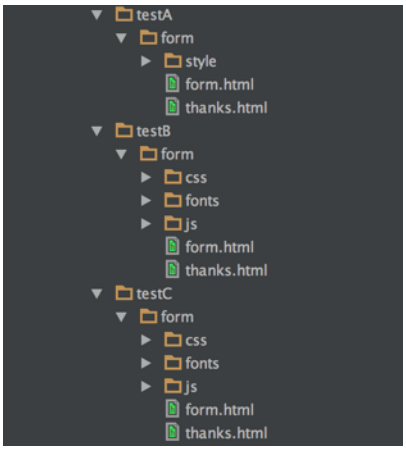


Figure 12: Three test websites that the library will interact with.

As the library will need to know the webpage source to complete its operation, the FirefoxDriver is included as part of the package. The library will contain the instance of the FirefoxDriver that is passed through to the tests rather than the tests instantiating the new instance of the driver, as the driver needs to know the page structure that can only be collected by the FirefoxDriver.

5.3. Development

The impact of development based from the design is understanding how the automation library can communicate to the tests, but understanding the basic elements of a website without the library being tailored to a single website. A scrum based agile approach intends to phase in certain development aspects, with the objective of the library to function with one site at a time, only using the HTML structure that a webpage uses along with the UIBox package to write the tests that link the library and test pages together. For instance, the library will not fully work for test site B whilst the library is still being developed to pick up HTML elements for test website A, as the library will encounter different structures for each website, along with an objective to identify all three HTML structures and scenarios. The same scenario will apply for test C

respectively. Therefore, the automation library will work around one website at a time, with the outcome expected for all websites to pass the tests. Additional library maintenance will take place after all three test websites are completed. The Automation scrum are provided in Appendix G Section 1, and the Web scrum tickets are provided in Appendix G Section 2.

During development, a version control system was used to ensure that the application was developed in a consistent approach based from the Scrum methodology, and to reduce the impact of any failures of development by staging each part of a successful development. Due to the flexibility and popularity of the version control system, git was used to host code onto a Bitbucket repository so the project can be worked on any computer without depending on a removable storage device.

The project on Git was using three levels of the branching structure, feature, develop, and master. The feature branch represents each task created on the Scrum board ticket, a new feature branch is created off from the develop branch. The develop branch contains all the merged feature branches once they are done, so once a feature branch is merged onto develop, the changes are verified to detect any faults before the branch is pushed. The develop branch was created from the master branch, and the final product will be merged to master prior to testing. The master branch contains the final version of the application, with the branch being created upon the creation of the repository.

The project is developed using a Maven framework which is designed for build management of projects, consisting of two java modules used as part of the automation project, UIBox and WeleniumLibrary. Each module is generated by Maven, automatically creating a main and test directory along with a Maven Project Object Model (POM) XML file used to store project configuration such as dependencies. Maven does contain a parent POM file outside of the modules, so if multiple module uses the same dependency, it can be distributed from the parent to all the child POM files.

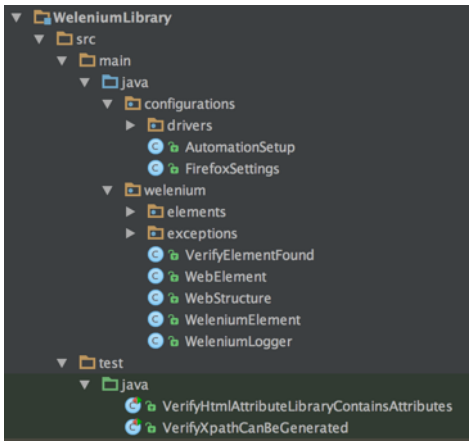


Figure 13: WeleniumLibrary module and its containing packages.

WeleniumLibrary consists of two sub packages (Figure 13), configurations which contains the configurations for the FirefoxDriver, and the Welenium folder contains all the java classes for the generation of paths. UIBox module (Figure 14) contains of a set of tests containing both Selenium and Welenium tests to demonstrate the application with and without the library. Additionally, the UIBox module contains a collection of test website used to demonstrate the robustness of the automation library. The WeleniumLibrary module communicates to the UIBox module but cannot work the other way, as Maven triggers a cyclic referencing exception.

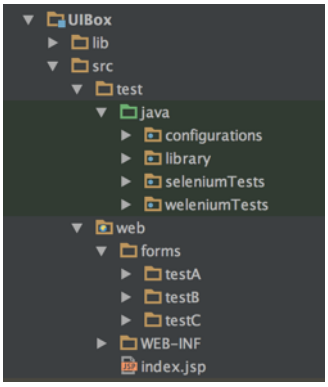


Figure 14: UIBox module and the containing packages.

The methodology of the way the application will be developed is by taking an object orientated approach for both modules, as some custom types will be used in the WeleniumLibrary package as part of the path build process, and for the UIBox to use as part of the automation tests. Custom types can be seen in Appendix D, Section 3, and in Appendix F, Section 1 Using custom types is an essential activity in programming with Java that is encouraged (Eckel, 2000). The automation library will be taking on a process that will follow a structure of data collection from both the keyword provided by the end user and the appropriate HTML tag that is collected, with the process using the collected data for the system to establish in creating a path that the WebDriver can detect. By taking on an object orientated approach, the project will aim to reduce the number of no return values and increase of return values for string and any necessary custom classes that may be created.

The automation library will contain literal string values for the system to identify HTML elements, where the same value literal string may be used in various locations in the application. For string literal source code, please see Appendix D Section 1 (for UIBox), and Appendix D Section 4 (for WeleniumLibrary). The automation library will contain a package just for literal strings stored in an enumeration file, therefore simplifying maintenance and preserving consistency across the whole system.

Although the library will be built to function on top of Selenium, the methods the library will provide to the user will be different but consisting of a cleaner structure to promote clean code practices and transferability of knowledge to other developers. The automation library will process one element at a time regardless on which page it is on, which keeps in line of the similar behaviour as Selenium WebDriver.

Some of the classes in the UIBox will inherit from the WeleniumLibrary, therefore replacing the Selenium code that would normally be used. It is important that any of the automation library methods used is simple enough for the developer to understand what the methods purpose is based on and what the developer wants to achieve using them. The automation library will contain three different functions (Table 4), where no more than two parameters will be passed to ensure developers can easily remember the data that needs to be passed (Fowler, Beck, Brant, Opdyke, & Roberts, 1999).

Method name (including parameters)	**Code in use (Screenshot)**	**Description**
readFromPage(String readElement)	`formPage.readFromPage(title);`	Reads webpage text based on the passed into the readElement string. Can be used to validate existence of text.
writeToPage(String value, String keyword)	`formPage.writeToPage(name, "name");`	Write the value into an element on a webpage which matches the element keyword. (e.g. textbox id would be name, therefore keyword finds 'name' in HTML tag properties and matches it to the element)
selectOnPage(String item)	`formPage.selectOnPage("checkbox");`	Selects a UI (e.g. button) on the webpage based on the item string which finds the control based on the UI's HTML tag properties.

Table 4: WeleniumElement methods which the end user has access to in the UIBox, and demonstrating the methods in use in code.

During development, that the structure to begin with is likely to be modified, nerveless it is imperative that the objective of the automation library is not changed.

5.4. Testing

There are various tests in the automation library, there are some Junit tests located in the WeleniumLibrary package and automation tests located in the UIBox package. The UIBox package is specifically designed to test the automation library from a front end prospective to a scenario of the test websites.

There will be three automation tests that provide the same behaviour, with the only difference being that each test goes to a different test site, but the actions are the same. The three automation tests could be all merged into one test, but it is not intended to run all test websites as that can be done as part of a test suite instead. Each test will fill out a form on the test website, select a control such as a button, and validate a message on the completion page.

Executing the automation tests will be done using Maven as it handles build management on projects. Prior to executing Maven, the test websites will need to be deployed onto a local virtual server to provide the test website, a simulated online state. Due to the virtual server having enhanced integration on IntelliJ, Glassfish has been selected to host the test websites on the UIBox package (Figure 15).

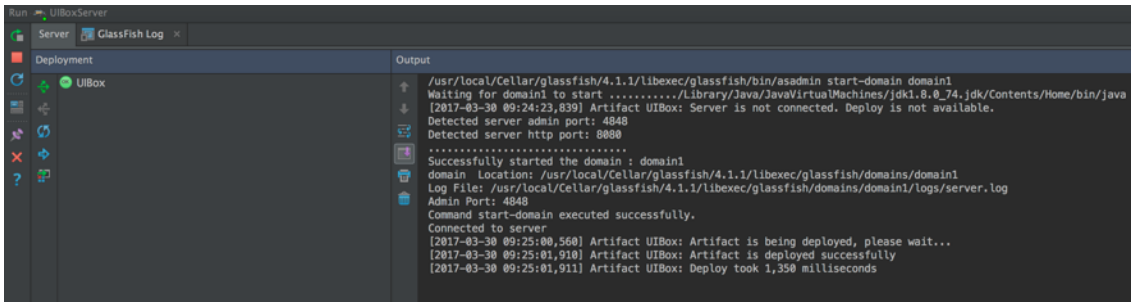


Figure 15: UIBoxServer Glassfish server in deployment

Once Glassfish is deployed, the automation tests will be ready to be executed. Two identical test suites will be used; the first test suite will be using Selenium without the automation library, whilst the second test suite will be using the automation library, to demonstrate the outcome of the tests between the two libraries.

The expected outcome of the automation tests is for the Selenium test to pass one of three test cases, as the first website will demonstrate as Selenium is set up for test A, no adaptations to the test will have taken place on the page objects for test B and test C. However, it is expected that new automation library tests in the UIBox will pass on all three tests as Selenium WebDriver tests need to know the full path of the website it is testing prior to deployment, where the new automation library locates and generates the path during deployment. A pass or failed result will be generated depending on the method of deployment, however any results regardless of deployment method will be the same. Deployment will be primarily being carried out through Maven commend on Mac terminal, and during development some tests will be carried out through IntelliJ. The terminal output is shown on Appendix E.

6. Evaluation

The development of the automation library has been a success, in terms of reaching the objective of generating a path to an element which the developer is looking for using a keyword. The development has highlighted a few issues particularly with Maven and path collection, nevertheless there were resolutions around the issues where the actual outcome has not affected the expected outcome of the application.

6.1. Issues During Development

6.1.1. Maven and WebDrivers

The development of the automation library has highlighted an error when using Maven. The original design was to have FirefoxDriver located on the UIBox package as the tests need to send Firefox the universal resources locator (URL) to navigate. As WeleniumLibrary needs to get the page source from current page the UIBox is on, the scenario produced a problem having both packages synchronised together. A failed attempt was conducted of passing the FirefoxDriver to the WeleniumLibrary package but Maven produced a circular referencing exception on build, as one package can inherit another but communication between the two packages is strictly one way. A solution was produced where the FirefoxDriver was relocated from the UIBox to the WeleniumDriver, and a new java class was created in UIBox specifically for WebDriver to get the automation libraries driver properties. So, as well as the library providing the methods for finding and interacting with content on a webpage, the library additionally provided the Firefox drivers.

For FirefoxDriver to work, Mozilla Firefox must be installed on the computer, however there is a known bug on the driver that does not allow newer versions of Firefox to work. A resolution was found when an older version of Firefox was compatible with the driver, therefore Firefox 32.0, the last version of the browser to work with the driver was installed. Although future versions of the library could include other browser drivers such as Google Chrome.

6.1.2. Paths

The largest challenge during the development of the automation library was the paths for Selenium to point to the element on the page. The original plan was for the automation library to generate a CSS selector for Selenium to use, however creating the CSS selector requires the automation library to go through every open HTML tag, and as the structure involved other open HTML tags that are part in the same tag, an incorrect CSS path would be generated. It has

become apparent that CSS path generation was not the solution due to the complexities highlighted earlier, therefore another path format was required.

Selenium did have other path options as well as CSS selector, where XPath was identified and selected as the best path option. XPath was the dynamic choice due to its operational flexibility, low coupling with the HTML structure, and a consistent path build structure; the automation library was modified to assemble a path.

XPath uses a contains tag to find the tag property based on its name but cannot isolate which HTML attribute it is on; therefore, a solution was found by creating a series of Java classes which can isolate the HTML tags per line, and exclude the properties of the tag depending on the function that is passed on the UIBoxs page objects. For instance, if a developer wants to enter a string in a textbox and calls the relevant method, the function will find paths that will contain the correct HTML tag properties and checks the value against the keyword the developer has passed through. Once the correct HTML tag is found, the path is then generated with the same tag that is required for the XPath formula. The generated path formula is then sent to another Java class where it sends the path to Selenium where it finds the element and performs the necessary action.

6.1.3. Ambiguous names

The automation library relies on the HTML tag attributes to have responsible names relating to the purpose of each HTML attribute serves on a webpage. For example, a textbox used for a phone number should contain in the id a keyword that associates to telephone, therefore ‘telephone’ would be used as part of the id attribute value. The rule for a consistent and exclusive id attribute does form as part of the World Wide Web consortium (W3C) guidelines (W3C, 2014), as developers (particularly web) should be aware of W3C guidelines.

During development, a challenge was presented that highlighted the automation library was becoming too dependent on the id. A solution was needed to increase HTML attribute coverage to find keywords by not limiting towards the id attributes. At the end of development, the automation library can collect name, placeholder, and type attributes as well as the id to compensate for the eventuality of missing attributes. To demonstrate the automation framework for ambiguous names, test C in the UIBox has the id attributes removed to determine that the framework can pick up other attributes apart from id.

Despite the full coverage of the automation library, any test failures relating to the system unable to find the matching attribute value based from the keyword entered in by the end user is not the responsibility of the library, but the developer is reasonable for the webpages and tests written up.

6.2. Impact on the findings

Automation tests using the new automation library is significantly different compared to the current Selenium tests, by eliminating the process of CSS selectors and using a simplified object orientation approach that a developer can easily learn.

Industries such as banking, retail, manufacturing, telecommunications, government, etc. utilise web based product which requires extensive testing before release. These would be positively impacted by the automation library because they can get improvements rolled out quicker with less cost and with greater flexibility with function.

For example: A bank, finding out their current site poses a greater risk of fraud, would ask the developers to improve the site. Ensuring retention of functionality, yet changing from an older HTML based system to a framework (e.g. Angular2) containing additional dependencies that can enhance security onto the page. Currently the testers would need to rewrite the whole automation test from scratch if they were using CSS selectors, which is most commonly used. Therefore, additional time and resources are used to fix a problem which could be exploited at the detriment of the bank. If the bank was using the new automation library, the test rewrite would take no time other than amending the tests for additional functionality of the website. As the saying goes, time is money.

By reducing the user interaction with the test, the rate of errors reduces over time. If the test is right once, its right continuously.

A difficulty in assessing the impact of the automation library is finding participants to write an automation test and compare it to the new automation library. This in the authors experience is due to a general lack of enthusiasm in quality assurance (QA) amongst developers. Testing the application in industry would be initially cumbersome and would require significant time and training to implement this due to the many standards and practices within various industries (Coy, 2017).

The automation library will have an impact on how web developers and QA engineers develop and test an application. Web developers will have increased flexibility in developing applications by not being constrained by a website structure as long the HTML attributes are correctly used. QA engineers will see the greatest impact from the automation library, with the lines of code (LOC) from three in the current Selenium tests, down to one from the new automation library, leading to the reduction in both resources and the task difficulty including coding and updating (specially to programming illiterate QA engineers), to perform the same functionality as current Selenium tests (Bhatia & Malhotra, 2014).

Software engineers will still be required to update automation tests, even with the use of the automation library. As content on a webpage may change overtime, such as new form fields and buttons being added in, but also HTML tags being removed such as redundant text boxes which is no longer required. However, amending tests where additional pages will need to be tested will be reduced as exact HTML element paths no longer need to be collected, but rather the keyword that is associated to the HTML element.

6.3. Future Improvements

The as the automation library contains a FirefoxDriver so the library can collect the HTML source from the rendered webpage, other WebDrivers could also be used. ChromeDriver can be implemented into the automation library package as an alternative to the FirefoxDriver. Additional settings for ChromeDriver will be required as each driver has its own configuration, but the tests and the rest of the library is unlikely to be effected.

Other coding platforms could be used for the automation library to expand on. In C#, a NuGet package manager would be used to get all the dependencies required, including Selenium WebDriver (Sharma, 2015). Execution of the automation test in C# can be done through visual studio, however visual studio is limited to the Window operating system. Other languages for Seleium WebDriver are available, including Python and Ruby, where the library can be rewritten for those platforms.

6.4. Personal Development

The learning experience in developing the automation library has been a challenging but rewarding experience. The development and research of the project has bridged the gap between the industries who require testing, and the automation tools available, that are a necessity but weaken productivity for early release.

Having some experience developing Selenium based automation tests provided a platform to begin with. Undertaking the project has provided a challenge by creating a Maven project from scratch for the first time. To overcome this, I consulted Mavens developers Apache book, and referred to various blogs and forums but these did not feel very trustworthy. Therefore, I referred to an official QA tools website, that covers technical tutorials regarding testing. This website has helped me to understand and acquire a new skill detailing the project object model (POM) structure, as part of Maven, written in XML.

As well as developing research skills, I've also developed skills in creativity to resolve the issue of how to create the XPath in the automation library.

Beneficial outcomes from the project have been that, I have become more independent in researching, designing and developing solutions. The solution was by manually getting the path and identified a pattern that XPath uses, this was also attempted for CSS selectors but as each selector is different, there was no structure that I could write for the system. As soon as I've done experimenting with XPath, I then had an idea for how the system can identify what path is required for each scenario along with how the path is constructed using a step by step process.

Bibliography

Adobe. (2005). *Adobe to acquire Macromedia*. Retrieved from Adobe: https://www.adobe.com/aboutadobe/invrelations/adobeandmacromedia.html

Altaf, I., Dar, J., & Rashid, F. (2015). Survey on selenium tool in software testing. *015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 1378-1383.

Bartley, M. (2014, April 29). *Achieving business benefits through automated software testing.* Retrieved February 19, 2017, from bcs.org: http://www.bcs.org/upload/pdf/test-automation-mbartley.pdf

Bhatia, S., & Malhotra, J. (2014). A survey on impact of lines of code on software complexity. *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, 1-4.

Christophe, L., De Roover, C., & De Meuter, W. (2013). Study on the Practices and Evolutions of Selenium Test Scripts. *Software Evolution Research Seminar*, 14.

Coy, A. (2017, February 5). Conversation on attitudes towards testing. (S. Stretton, Interviewer)

Eckel, B. (2000). *Thinking in Java.* Upper Saddle River: Prentice-Hall Inc.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code* (1st Edition ed.). Reading, Massachusetts: Addison Wesley Longman, Inc.

Galin, D. (2004). *Software Quality Assurance: From theory to implementation.* London: Pearson Education Limited.

Glassman, N. R., & Shen, P. (2014). One Site Fits All: Responsive Web Design. *Journal of Electronic Resources in Medical Libraries*, 78-90.

Graba, J. (2007). *An Introduction to Network Programming with Java.* Sheffield: Springer Science+Business Media, LLC.

Graham, D., & Fewster, M. (2012). *Experiences of Test Automation: Case Studies of Software Test Automation.* Upper Saddle River: Pearson Education, Inc.

Hegedüs, Á., Horváth, Á., & Ráth, I. (2011). Quick fix generation for DSMLs. *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 17-24.

Herzig, K. (2014). Using Pre-Release Test Failures to Build Early Post-Release Defect Prediction Models. *2014 IEEE 25th International Symposium on Software Reliability Engineering*, 300-311.

Holtsnider, B., Wheeler, T., Stragand, G., & Gee, J. (2010). *Agile Development & Business Goals The Six Week Solution.* Burlington: Elsevier Science.

Horne, G. (2014, October 7). *A (Very) Brief History of Test Automation*. Retrieved March 16, 2017, from Linkedin Pulse: https://www.linkedin.com/pulse/20141007123253-16089094-a-very-brief-history-of-test-automation

Jamil, M. A., Arif, M., Abubakar, N. S., & Ahmad, A. (2016). Software Testing Techniques: A Literature Review. *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, 177-182.

Lombardi, G. (2014, June 19). 5 Ways to Tell If Your Website Needs an Upgrade. *Marketing your practice on the Internet*, 51-52.

Office for National Statistics. (2016, August 4). *Internet access – households and individuals: 2016*. Retrieved from Office for National Statistics: https://www.ons.gov.uk/peoplepopulationandcommunity/householdcharacteristics/homeinternetandsocialmediausage/bulletins/internetaccesshouseholdsandindividuals/2016#activities-completed-on-the-internet

Pannu, Y. S. (2015). Test Automation Using Cucumber and Selenium Webdriver. *International Journal on Advance Computer Theory and Engineering*, 26-28.

Selenium Documentation. (2017, March 14). *Migrating From Selenium RC to Selenium WebDriver*. Retrieved March 16, 2017, from SeleniumHQ: http://www.seleniumhq.org/docs/appendix_migrating_from_rc_to_webdriver.jsp

Selenium Project. (2012, August 26). *Selenium Documentation Release 1.0.* Retrieved from scholar.harvard.edu: http://scholar.harvard.edu/files/tcheng2/files/selenium_documentation_0.pdf

Selenium WebDriver. (2017, March 20). *Selenium WebDriver*. Retrieved March 21, 2017, from seleniumhq.org: http://www.seleniumhq.org/projects/webdriver/

Sharma, L. (2015, October 15). *Set Up Selenium WebDriver with Visual Studio in C#.* Retrieved April 4, 2017, from ToolsQA: http://toolsqa.com/selenium-webdriver/c-sharp/set-up-selenium-webdriver-with-visual-studio-in-c/

Sypolt, G. (2016, April). *Writing Great Cucumber Tests.* Retrieved October 4, 2016, from Sauce Labs: https://saucelabs.com/blog/write-great-cucumber-tests

The Standish Group. (2014, July 2). *Big Bang Boom.* Retrieved February 19, 2017, from Standish Group: https://www.standishgroup.com/sample_research_files/BigBangBoom.pdf

Vasileva, A., & Schmedding, D. (2016). How to Improve Code Quality by Measurement and Refactoring. *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 131-136.

Vitó Ferreira, N. N., & Langerman, J. J. (2014). Proving that the Release Management Process can Enhance Throughput in Software Development Projects. *The 9th International Conference on Computer Science & Education (ICCSE 2014)*, 419-424.

W3C. (2014, October 28). *3 Semantics, structure, and APIs of HTML documents*. Retrieved April 3, 2017, from w3.org: https://www.w3.org/TR/html5/dom.html#the-id-attribute

Yang, Q., Li, J. J., & Weiss, D. M. (2009). A Survey of Coverage-Based Testing Tools. *The Computer Journal, 52*(5), 589-597.

Appendices

Appendix A – Project Specification Document

PROJECT SPECIFICATION - SEGM 2016/17

Student:	**Steven Stretton**
Date:	**20/10/2016**
Supervisor:	**Tony Day**
Degree Course:	**BENG Software Engineering**
Title of Project:	**The development of a robust test automation library for Selenium WebDriver application**

Elaboration

I will be developing a prototype Java-based automation test library to assist in the building of a robust web driver based test suite that can adapt to small changes on a webpage. The main goal is for the automation test to interact with a webpage, even if the webpage structure is upgraded, but keep the page functionality the same.

As the scope of this project could potentially be large, the focus is navigating and filling in a set form within three locally hosted test websites.

Each of the website test areas will need to be able to have the same objective across all the test pages in terms of what the site is doing. So, if the site is filling and submitting a form (with the same information) then any enhanced site must do the same thing. Overall the whole of the project will be a demonstration that the system works.

Project Aims

The project of developing a prototype of the robust automation system is aiming to achieve the following:

- Run an automation test on different versions of the same page and pass within the set criteria of the test.
- To create an ease of use method implementation for the end user that can be applied to the fundamentals of test driven development.
- Utilise the clean code concepts into the library, this will include implementing an object orientation.
- To output test results of each found element onto a terminal.

Project deliverable(s)

I will be developing a prototype automation test library that will be able to adapt to any webpage upgrades.

The automation test library will be developed in Java using an IDE application with the execution of the test and version control done in terminal. The test script will be running on a web browser, whereas the results will be outputted on a console terminal.

The software and hardware tools required for this will be my own. For software I'm intending to use IntelliJ IDEA for Java (for the automation test) and HTML (for the webpage), Terminal and Mozilla Firefox. The hardware side will be developed on a MacBook Pro and iMac, although development could also be done on Ubuntu.

To ensure that the work is backed up, this project will be saved using a version control system, in this case Git will be used, whilst all the projects (and wikis) will be stored on a private Git repository. An agile development of the project will be used; in this case it will be a Scrum/Kanban model.

Action plan

Task	Description	Deadline
Find a project supervisor	Email module leader supervisor's name	07/10/2016
Create a basic testing template (Test A)	This testing template will be a HTML page (uploaded online) that has a basic form. The form details do not need to be saved	07/10/2016
Develop Current Selenium Test	Create a basic Selenium WebDriver test that uses a testing template.	07/10/2016
Research	Look at Existing Automation Frameworks, and see what functionality and any shortcomings that my application may address.	07/10/2016
Submit project specification and ethics form	For formative assessment and moderation, this is done electronically through Blackboard.	21/10/2016
Create a Class Diagram	Develop a class diagram that shows the outline of the library. Tests will not need to be included on this	21/10/2016
Attend research workshop	1pm in Adsetts 6624.	31/10/2016
Develop the Objects	Develop the objects and methods that may be needed	04/11/2016

	in IntelliJ, a library will not be needed yet.	
Research	• Webpage Upgrades • Libraries	11/11/2016
	• Clean Code • Object Orientated Programming	25/11/2016
	• Human Computer Interaction • Test Driven Development	09/12/2016
Create a Bootstrap testing template (Test B)	This will be a near replica of the basic HTML page but using the Bootstrap framework. A basic Bootstrap cookie banner will be added as part of a separate test.	18/11/2016
Information review	This can be either submitted via Blackboard or discussion with project supervisor	02/12/2016
Develop first prototype	The automation code will be able to run test A and B with minimal issues on two pages for both the basic and Bootstrap template.	16/12/2016
Create an advanced testing template (Test C)	The template will need to have some modified controls (e.g. changing a radio button with dropdown) but retain the same behaviour. A advanced cookie banner will be added as part of a separate test.	20/01/2017
Submit provisional contents page	This can be either submitted via Blackboard or discussion with the project supervisor.	10/02/2017
Develop second prototype	The second prototype should be able to run Test A and B with minimal issues.	17/02/2017
Draft critical evaluation	This can be either submitted via Blackboard or discussion with the project supervisor.	31/02/2017

Selection of draft report	This can be either submitted via Blackboard or discussion with the project supervisor.	31/02/2017
Develop final prototype	The final prototype should be able to run Test A, B and C without any issues.	17/03/2017
Proof read and reference check	Proof read the report to ensure content is ok, and check that the references are correct.	10/04/2017
Submit the project report to TurnitinUK	Submit electronically to module's Blackboard site.	24/04/2017
The Project Report and electronic copies of the deliverable	Bound copies to Cantor Reception + CD/DVD/pen drive, upload to BB &/or clear instructions for accessing my work.	26/04/2017
Demonstration of my work	This will need to be agreed with the project supervisor and second marker.	Tbc (before 12/05/2017)

Ethics

Please see the Ethics form.

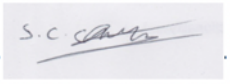
Appendix B – The Ethics form

Ethics Checklist – Project (SEGM) 55-6727

If the answer to any question is 'yes' the issue **MUST** be discussed with your project supervisor.

Question		***Yes/No***
1.	Does the project involve human participants? This includes surveys, questionnaires, observing behaviour, testing etc.	Yes
2.	Does the project involve the use of live animals?	No
3.	Does the project involve an external organisation? If yes, please write the name of the organisation here:	*No*
4.	Does the project require access to any private or otherwise sensitive material?	*No*
5.	Does the project require the reproduction (beyond normal academic quotations) of materials authored by a source other than yourself?	*No*

Adherence to SHU policy & procedures

Declaration
I can confirm that: • I have read the Sheffield Hallam University Research Ethics Policy (available at http://www.shu.ac.uk/_assets/pdf/research-ethics-policy.pdf) • I agree to abide by its principles. Signature …  ……Print Name…Steven Stretton…… Date …19/10/2016…..

Appendix C – Research

Section 1: Existing Version Control Products

Part A: Github

GitHub is a cloud-based hosting application provided by GitHub, Inc. It works by storing source code that can be pushed or pulled by a computer that has git installed through either a GUI-based application or terminal command. GitHub does have branch capability so any branch worked on can be pushed from one computer and pulled from another computer (ensuring that the end user has the clone address). GitHub is free to use however the repository is only available publically, although for a fee (or free for students) the repository can be made private.

Part B: Bitbucket

Bitbucket is cloud-based hosting application provided by Atlassian. Like GitHub, it stores source code that can be pushed or pulled by a computer that has git installed through either a GUI-based application or terminal command. Bitbucket does provide the option of both public and private repositories which are both free (with a limit of 5 users).

Part C: GitLab

GitLab is a cloud-based hosting application provided by GitLab Inc. The application runs at an agile level with all the code, tests with an integrated CI, and deployment. Gitlabs does provide a free account with unlimited public or private repositories. Like Bitbucket and GitHub, it stores source code that can be pushed or pulled by a computer that has git installed, but it only works through a terminal command, as GitLab has yet to make it available for GUI applications such as SourceTree.

Section 2: Possible System Code: Fake Code

Part A: Overview

The library will still need to use the selenium WebDriver, however the approach that it will work will be modified so rather having an explicit path entered in by default it will get the path by searching the HTML structure of the site by taking in the keywords that the developer would enter and search by various Id's. It is intended that this system won't work for every scenario as it will need to be clear where the test will need to pass and fail.

The way that Automation tests cases are written will not require any changes, however the page objects will require modification.

Part B: Proposed Objects

The following concepts are proposed to be implemented:

- A set of new objects in the library would be created, listed below is the proposed objects and their purposes that they will serve:

Object Name	**Purpose**
WebElements	Collect the text from the page object (entered in by the developer) and will host the process objects which regardless of the outcome will pass it to the VerifyElementFound object.
WebStructure	Read and process the webpage structure. DomStructure will find element that may be either controls or element groups (e.g. radio buttons). WebStructure could also be adapted to get all tag names and id's (e.g. div id ='name'). All the elements found would be passed to FoundWebElements.
FoundWebElements	FoundWebElements will collect the elements found from the WebStructure object and will organise the results into groups (for example if it finds a button, it will add it to a category to uiControls)
VerifyElementFound	VerifyElement will check for any matching keywords that is found from the DomStructure to the WebStructure. If a keyword given is found it will get the full path of the element in which the webdriver will do the rest, otherwise it will call a MultipleElementFoundException object
MultipleElementFoundException	When multiple elements are found, a MultipleElementFoundException will be thrown in which the test will fail.
NoElementFoundException	If the system cannot find the keyword that is found, a NoElementFoundException will throw, resulting in a test failiure.

Part C: Proposed Code

A proposed code concept is for the user to search the page by various of control groups in which it will tell the library what to look for by searching the HTML structure and finding the elements from there. The system will need to search for multiple elements as if it finds multiple elements with the same keyword then the test would need to fail.

Part D: Page Objects

In standard Automation Test convention, the interfaces would belong in the Automation code and because of this the new concept will remove the interface as the library will work out the path to the control that the developer tells it to find.

Appendix D – Source Code

Section 1: UIBox test

WeleniumFirefox.java

```
package configurations;

import configurations.drivers.WeleniumFirefoxDriver;
import org.openqa.selenium.support.PageFactory;
import welenium.WebElement;
import weleniumTests.pageObject.WebForm;

import static configurations.FirefoxSettings.driver;

/**
 * Created by stevenstretton on 04/01/2017.
 */
public class WeleniumFirefox
{
    private WebElement element = new WebElement();

    public void navigateToPage(String ADDRESS)
    {
        pageFactoryConfig();
        WeleniumFirefoxDriver.goToPage(ADDRESS);
        element.collectElement();
    }

    public void getElement()
    {
         element.collectElement();
    }

    private void pageFactoryConfig()
    {
        PageFactory.initElements(driver, WebForm.class);
    }

}
```

FormUtil.java

```
package library;

/**
 * Created by stevenstretton on 23/03/2017.
 */
public enum FormUtil {

    NAME_STEVEN_STRETTON("Steven Stretton"),
    ADDRESS_SOMEWHERE_ST("123 Somewhere Street"),
    EMAIL_STEVEN("stevenstretton1@email.com"),
    PHONE_LOCAL("0114123456"),
    DATE_2017("01-01-2017"),
    MESSAGE("Hold on Tight!!!");

    private final String formData;

    FormUtil(String formInformation) {
        formData = formInformation;
    }

    public final String getFormData()
    {
```

```
        return formData;
    }
}
```

NavigationUtil.java

```
package library;

/**
 * Created by stevenstretton on 23/03/2017.
 */
public enum NavigationUtil {

    NAVIGATE_TO_TEST_A("A"),
    NAVIGATE_TO_TEST_B("B"),
    NAVIGATE_TO_TEST_C("C");

    private final String testPage;

    NavigationUtil(String testString) {
        testPage = testString;
    }

    public final String getTestPage()
    {
        return testPage;
    }

}
```

TestA.java

```
package seleniumTests.automationTests;

import configurations.AutomationSetup;
import org.junit.Test;
import seleniumTests.pageObject.WebForm;
import seleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.NavigationUtil.NAVIGATE_TO_TEST_A;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestA extends WebForm
{
    @SuppressWarnings("Duplicates")
    @Test
    public void testA() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_A.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();

        fillInForm(
                NAME_STEVEN_STRETTON.getFormData(),
```

```
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );

        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);

        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

TestB.java

```
package seleniumTests.automationTests;

import configurations.AutomationSetup;
import org.junit.Test;
import seleniumTests.pageObject.WebForm;
import seleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.FormUtil.DATE_2017;
import static library.FormUtil.MESSAGE;
import static library.NavigationUtil.NAVIGATE_TO_TEST_B;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestB extends WebForm {
    @SuppressWarnings("Duplicates")
    @Test
    public void testB() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_B.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();


        fillInForm(
                NAME_STEVEN_STRETTON.getFormData(),
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );
```

```
        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);

        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

TestC.java

```
package seleniumTests.automationTests;

import configurations.AutomationSetup;
import org.junit.Test;
import seleniumTests.pageObject.WebForm;
import seleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.FormUtil.DATE_2017;
import static library.FormUtil.MESSAGE;
import static library.NavigationUtil.NAVIGATE_TO_TEST_C;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestC extends WebForm {
    @SuppressWarnings("Duplicates")
    @Test
    public void testC() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_C.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();


        fillInForm(
                NAME_STEVEN_STRETTON.getFormData(),
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );

        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);
```

```
        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

CompletedFormPage.java

```
package seleniumTests.pageObject.formObjects;


import configurations.WeleniumFirefox;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

/**
 * Created by stevenstretton on 27/09/2016.
 */
public class CompletedFormPage extends WeleniumFirefox
{
    @FindBy(css = "#name")
    private WebElement formMessage;

    public void pageTitle(String title)
    {
        System.out.println(title);
    }
}
```

FormObject.java

```
package seleniumTests.pageObject.formObjects;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

/**
 * Created by stevenstretton on 27/09/2016.
 */
public class FormObject
{
    @FindBy(css = "#name")
    private WebElement formName;
    @FindBy(css = "#address")
    private WebElement formAddress;
    @FindBy(css = "#email")
    private WebElement formEmail;
    @FindBy(css = "#telephone")
    private WebElement formTelephoneNumber;
    @FindBy(css = "#date")
    private WebElement formDateOfAvailability;
    @FindBy(css = "#additional")
    private WebElement formAdditionalInformation;
    @FindBy(css = "#checkbox")
    private WebElement formConfirmTermsAndConditions;
    @FindBy(css = "html>body>table>tbody>tr>td>a>button")
    private WebElement formSubmit;


    public void enterFormName(String name)
    {
```

```
        formName.sendKeys(name);
    }

    public void enterFormAddress(String address)
    {
        formAddress.sendKeys(address);
    }

    public void enterFormEmail(String email)
    {
        formEmail.sendKeys(email);
    }

    public void enterFormTelephoneNumber(String telephone)
    {
        formTelephoneNumber.sendKeys(telephone);
    }

    public void enterFormDateOfAvailability(String
dateOfAvailability)
    {
        formDateOfAvailability.sendKeys(dateOfAvailability);
    }

    public void enterFormAdditionalInformation(String
additionalInformation)
    {

formAdditionalInformation.sendKeys(additionalInformation);
    }

    public void selectReadTermsAndConditions()
    {
        formConfirmTermsAndConditions.click();
    }

    public void submitForm()
    {
        formSubmit.click();
    }

}
```

IndexObject.java

```
package seleniumTests.pageObject.formObjects;

import configurations.FirefoxSettings;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

/**
 * Created by stevenstretton on 06/12/2016.
 */
public class IndexObject
{
    @FindBy(css = "#itemList>ul>li:nth-child(1)>a")
    private WebElement testA;
    @FindBy(css = "#itemList>ul>li:nth-child(2)>a")
    private WebElement testB;
    @FindBy(css = "#itemList>ul>li:nth-child(3)>a")
    private WebElement testC;

    public void selectTestA()
```

```
    {
        String TEST_A_URL =
"http://localhost:8080/UIBox/forms/testA/form/form.html";
        FirefoxSettings.navigateToPage(TEST_A_URL);
    }

    public void selectTestB()
    {
        String TEST_B_URL =
"http://localhost:8080/UIBox/forms/testB/form/form.html";
        FirefoxSettings.navigateToPage(TEST_B_URL);
    }

    public void selectTestC()
    {
        String TEST_C_URL =
"http://localhost:8080/UIBox/forms/testC/form/form.html";
        FirefoxSettings.navigateToPage(TEST_C_URL);
    }

    public void invalidTestSelection()
    {
        System.out.println("invalid test selected");
    }
}
```

WebForm.java

```
package seleniumTests.pageObject;

import configurations.FirefoxSettings;
import configurations.WeleniumFirefox;
import org.openqa.selenium.support.PageFactory;
import seleniumTests.pageObject.formObjects.FormObject;
import seleniumTests.pageObject.formObjects.IndexObject;


/**
 * Created by stevenstretton on 25/09/2016.
 */
public class WebForm
{
    private FormObject formObject;

    protected void start()
    {
        formObject =
PageFactory.initElements(FirefoxSettings.driver,
FormObject.class);
    }

    @SuppressWarnings("Duplicates")
    protected void navigateToFormWebpage(Character
testToExecute)
    {
        String SIMPLE_FORM_URL = "http://localhost:8080/UIBox/";
        WeleniumFirefox weleniumFirefox = new WeleniumFirefox();
        weleniumFirefox.navigateToPage(SIMPLE_FORM_URL);

        IndexObject indexObject = new IndexObject();

        switch (testToExecute)
        {
            case 'A':
```

```
                indexObject.selectTestA();
                break;
            case 'B':
                indexObject.selectTestB();
                break;
            case 'C':
                indexObject.selectTestC();
                break;
            default:
                indexObject.invalidTestSelection();
                break;
        }

    }

    @SuppressWarnings("Duplicates")
    protected void fillInForm(String name, String address,
String email, String telephone,
                              String date, String information,
boolean termsAndConditions)
    {
        formObject.enterFormName(name);
        formObject.enterFormAddress(address);
        formObject.enterFormEmail(email);
        formObject.enterFormTelephoneNumber(telephone);
        formObject.enterFormDateOfAvailability(date);
        formObject.enterFormAdditionalInformation(information);
        isTermsAndConditionsChecked(termsAndConditions);
    }

    protected void submitForm()
    {
        formObject.submitForm();
    }

    private void isTermsAndConditionsChecked(boolean
termsAndConditions)
    {
       if (termsAndConditions)
       {
           formObject.selectReadTermsAndConditions();
       }

    }
}
```

TestA.java

```
package weleniumTests.automationTests;

import configurations.AutomationSetup;
import weleniumTests.pageObject.WebForm;
import org.junit.Test;
import weleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.FormUtil.DATE_2017;
import static library.FormUtil.MESSAGE;
import static library.NavigationUtil.NAVIGATE_TO_TEST_A;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestA extends WebForm {
```

```
    @SuppressWarnings("Duplicates")
    @Test
    public void testA() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_A.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();

        fillInForm(
                NAME_STEVEN_STRETTON.getFormData(),
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );

        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);

        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

TestB.java

```
package weleniumTests.automationTests;

import configurations.AutomationSetup;
import org.junit.Test;
import weleniumTests.pageObject.WebForm;
import weleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.FormUtil.DATE_2017;
import static library.FormUtil.MESSAGE;
import static library.NavigationUtil.NAVIGATE_TO_TEST_B;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestB extends WebForm {
    @SuppressWarnings("Duplicates")
    @Test
    public void testB() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_B.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
```

```
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();

        fillInForm(
                NAME_STEVEN_STRETTON.getFormData(),
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );

        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);

        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

TestC.java

```
package weleniumTests.automationTests;

import configurations.AutomationSetup;
import org.junit.Test;
import weleniumTests.pageObject.WebForm;
import weleniumTests.pageObject.formObjects.CompletedFormPage;

import static library.FormUtil.*;
import static library.FormUtil.DATE_2017;
import static library.FormUtil.MESSAGE;
import static library.NavigationUtil.NAVIGATE_TO_TEST_C;

/**
 * Created by stevenstretton on 25/09/2016.
 */
public class TestC extends WebForm {
    @SuppressWarnings("Duplicates")
    @Test
    public void testC() throws InterruptedException {

        char navigationChar =
NAVIGATE_TO_TEST_C.getTestPage().charAt(0);
        AutomationSetup automationSetup = new AutomationSetup();
        CompletedFormPage completedFormPage = new
CompletedFormPage();
        automationSetup.executeInitialisationSettings();
        navigateToFormWebpage(navigationChar);
        start();

        fillInForm(
```

```
                NAME_STEVEN_STRETTON.getFormData(),
                ADDRESS_SOMEWHERE_ST.getFormData(),
                EMAIL_STEVEN.getFormData(),
                PHONE_LOCAL.getFormData(),
                DATE_2017.getFormData(),
                MESSAGE.getFormData(),
                true
        );

        Thread.sleep(1000);

        submitForm();

        Thread.sleep(1000);

        completedFormPage.pageTitle("Thank You!");

        Thread.sleep(1000);

        automationSetup.endOfAutomationTest();

    }

}
```

CompletedFormPage.java

```
package weleniumTests.pageObject.formObjects;

import configurations.WeleniumFirefox;
import welenium.WeleniumElement;

/**
 * Created by stevenstretton on 27/09/2016.
 */
public class CompletedFormPage extends WeleniumFirefox
{
    private WeleniumElement formPage = new WeleniumElement();

    public void pageTitle(String title) {
        formPage.readFromPage(title);
    }
}
```

FormObject.java

```
package weleniumTests.pageObject.formObjects;

import configurations.WeleniumFirefox;
import welenium.WeleniumElement;

/**
 * Created by stevenstretton on 27/09/2016.
 */
public class FormObject
{
    private WeleniumElement formPage = new WeleniumElement();
    private WeleniumFirefox weleniumFirefox = new
WeleniumFirefox();

    public void enterFormName(String name)
    {
        formPage.writeToPage(name, "name");
```

```
    }

    public void enterFormAddress(String address)
    {
        formPage.writeToPage(address, "address");
    }

    public void enterFormEmail(String email)
    {
        formPage.writeToPage(email, "email");
    }

    public void enterFormTelephoneNumber(String telephone)
    {
        formPage.writeToPage(telephone, "telephone");
    }

    public void enterFormDateOfAvailability(String
dateOfAvailability)
    {
        formPage.writeToPage(dateOfAvailability, "date");
    }

    public void enterFormAdditionalInformation(String
additionalInformation)
    {
        formPage.writeToPage(additionalInformation,
"additional");
    }

    public void selectReadTermsAndConditions()
    {
        formPage.selectOnPage("checkbox");
    }

    public void submitForm()
    {
        formPage.selectOnPage("submit");
        weleniumFirefox.getElement();
    }

}
```

IndexObject.java

```
package weleniumTests.pageObject.formObjects;

import configurations.WeleniumFirefox;

/**
 * Created by stevenstretton on 06/12/2016.
 */
public class IndexObject
{
    private WeleniumFirefox weleniumFirefox = new
WeleniumFirefox();

    public void selectTestA()
    {
        String TEST_A_URL =
"http://localhost:8080/UIBox/forms/testA/form/form.html";
        weleniumFirefox.navigateToPage(TEST_A_URL);
    }
```

```
    public void selectTestB()
    {
        String TEST_B_URL =
"http://localhost:8080/UIBox/forms/testB/form/form.html";
        weleniumFirefox.navigateToPage(TEST_B_URL);
    }

    public void selectTestC()
    {
        String TEST_C_URL =
"http://localhost:8080/UIBox/forms/testC/form/form.html";
        weleniumFirefox.navigateToPage(TEST_C_URL);
    }

    public void invalidTestSelection()
    {
        System.out.println("invalid test selected");
    }
}
```

WebForm.java

```
package weleniumTests.pageObject;

import configurations.FirefoxSettings;
import configurations.WeleniumFirefox;
import org.openqa.selenium.support.PageFactory;
import weleniumTests.pageObject.formObjects.FormObject;
import weleniumTests.pageObject.formObjects.IndexObject;
import welenium.WeleniumLogger;


/**
 * Created by stevenstretton on 25/09/2016.
 */
public class WebForm
{
    private FormObject formObject;
    private WeleniumLogger weleniumLogger = new
WeleniumLogger();

    protected void start()
    {
        formObject =
PageFactory.initElements(FirefoxSettings.driver,
FormObject.class);
        weleniumLogger.init();
    }

    @SuppressWarnings("Duplicates")
    protected void navigateToFormWebpage(Character
testToExecute)
    {
        String SIMPLE_FORM_URL = "http://localhost:8080/UIBox/";
        WeleniumFirefox weleniumFirefox = new WeleniumFirefox();
        weleniumFirefox.navigateToPage(SIMPLE_FORM_URL);

        IndexObject indexObject = new IndexObject();

        switch (testToExecute)
        {
            case 'A':
                indexObject.selectTestA();
                break;
```

```
            case 'B':
                indexObject.selectTestB();
                break;
            case 'C':
                indexObject.selectTestC();
                break;
            default:
                indexObject.invalidTestSelection();
                break;
        }


    }

    @SuppressWarnings("Duplicates")
    protected void fillInForm(String name, String address,
String email, String telephone,
                              String date, String information,
boolean termsAndConditions)
    {
        formObject.enterFormName(name);
        formObject.enterFormAddress(address);
        formObject.enterFormEmail(email);
        formObject.enterFormTelephoneNumber(telephone);
        formObject.enterFormDateOfAvailability(date);
        formObject.enterFormAdditionalInformation(information);
        isTermsAndConditionsChecked(termsAndConditions);
    }

    protected void submitForm()
    {
        formObject.submitForm();
    }

    private void isTermsAndConditionsChecked(boolean
termsAndConditions)
    {
       if (termsAndConditions)
       {
           formObject.selectReadTermsAndConditions();
       }

    }

}
```

Section 2: UIBox Web

Form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Basic Web Form</title>
    <link href="style/css/style.css" rel="stylesheet"
type="text/css">
</head>
<body>
<table>
    <tr>
        <th colspan="2">
            <h1 id>Simple Form!</h1>
        </th>
    </tr>
    <tr>
```

```
        <th align="right">
            Name:
        </th>
        <td>
            <input id="name" type="text"/>
        </td>
    </tr>
    <tr>
        <th align="right">
            Address:
        </th>
        <td>
            <input id="address" type="text"/>
        </td>
    </tr>
    <tr>
        <th align="right">
            Email:
        </th>
        <td>
            <input id="email" type="email"/>
        </td>
    </tr>
    <tr>
        <th align="right">
            Telephone Number:
        </th>
        <td>
            <input id="telephone" type="number"/>
        </td>
    </tr>
    <tr>
        <th align="right">
            Date of Availability:
        </th>
        <td>
            <input id="date" type="date"/>
        </td>
    </tr>
    <tr>
        <th align="right">
            Additional Information:
        </th>
        <td>
            <textarea id="additional"></textarea>
        </td>
    </tr>
    <tr>
        <th align="right">
            Confirm Terms and Conditions
        </th>
        <td>
            <input id="checkbox" type="checkbox"/>
        </td>
    </tr>
    <tr>
        <td colspan="2" align="right">
            <a href="thanks.html">
                <button type="submit">Submit</button>
            </a>
        </td>
    </tr>
    <tr>
        <td align="right">

        </td>
    </tr>
```

```
</table>
</body>
</html>
```

Thanks.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Basic Web Form</title>
    <link href="style/css/style.css" rel="stylesheet"
type="text/css">
</head>
<body>
<h1>Thank You!</h1>
</body>
</html>
```

Form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Test B Form</title>
    <link type="text/css" rel="stylesheet"
href="css/bootstrap.min.css">
</head>
<body>
<form class="form-horizontal" action="thanks.html">
    <fieldset>

        <legend>Test B</legend>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="name">Name</label>
            <div class="col-md-4">
                <input id="name" name="name" type="text"
placeholder="your name here..." class="form-control input-md">
            </div>
        </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="email">Email</label>
            <div class="col-md-4">
                <input id="email" name="email" type="text"
placeholder="your email..." class="form-control input-md">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="address">Address</label>
            <div class="col-md-4">
                <input id="address" name="address" type="text"
placeholder="your address..."
                       class="form-control input-md">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="telephone">Telephone</label>
            <div class="col-md-4">
```

```
                <input id="telephone" name="telephone"
type="text" placeholder="your telephone..."
                       class="form-control input-md">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="date">Date</label>
            <div class="col-md-4">
                <input id="date" name="date" type="text"
placeholder="your date..." class="form-control input-md">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="additional">Additional Information</label>
            <div class="col-md-4">
                <textarea class="form-control" id="additional"
name="additional"></textarea>
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label">Read terms and
conditions</label>
            <div class="checkbox">
                <label>
                    <input type="checkbox" name="checkboxes"
id="checkbox" value="read">
                    I've read the terms and conditions
                </label>
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-4 control-label"
for="submit">Submit</label>
            <div class="col-md-8">
                <button id="submit" name="submit" class="btn
btn-success" type="submit">Submit</button>
                <button id="cancel" name="cancel" class="btn
btn-danger" type="reset">Cancel</button>
            </div>
        </div>

    </fieldset>
</form>
</body>
</html>
```

Thanks.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Basic Web Form</title>
    <link href="css/bootstrap.min.css" rel="stylesheet"
type="text/css">
</head>
<body>
<div class="container">
    <h1 class="page-header">Thank You!</h1>
</div>
</body>
</html>
```

Form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Test C Form</title>
    <link type="text/css" rel="stylesheet"
href="css/bootstrap.min.css">
    <link type="text/css" rel="stylesheet" href="css/main.css">
</head>
<body>
<div class="container">
    <div class="row main">
        <div class="panel-heading">
            <div class="panel-title text-center">
                <h1 class="title">Test C</h1>
                <hr/>
            </div>
        </div>
        <div class="main-login main-center">
            <form class="form-horizontal" method="post"
action="thanks.html">

                <div class="form-group">
                    <label for="name" class="cols-sm-2 control-
label">Your Name</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-user" aria-hidden="true"></i></span>
                            <input name="name" type="text"
placeholder="your name here..."
                                   class="form-control input-
md">
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="email" class="cols-sm-2 control-
label">Your Email</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-envelope"

aria-hidden="true"></i></span>
                            <input name="email" type="text"
placeholder="your email..." class="form-control input-md">
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="address" class="cols-sm-2
control-label">Your Address</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-home" aria-hidden="true"></i></span>
                            <input name="address" type="text"
placeholder="your address..."
                                   class="form-control input-
md">
```

```
                        </div>
                    </div>
                </div>


                <div class="form-group">
                    <label for="telephone" class="cols-sm-2
control-label">Your Telephone</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-phone" aria-hidden="true"></i></span>
                            <input name="telephone" type="text"
placeholder="your telephone..."
                                   class="form-control input-
md">
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="date" class="cols-sm-2 control-
label">Your Date</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-calendar"

aria-hidden="true"></i></span>
                            <input name="date" type="text"
placeholder="your date..." class="form-control input-md">
                        </div>
                    </div>
                </div>


                <div class="form-group">
                    <label for="additional" class="cols-sm-2
control-label">Additional Info</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-pencil"

aria-hidden="true"></i></span>
                            <textarea class="form-control"
name="additional"></textarea>
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="checkbox" class="cols-sm-2
control-label">Accept Terms?</label>
                    <div class="cols-sm-10">
                        <div class="input-group">
                            <span class="input-group-addon"><i
class="glyphicon glyphicon-star" aria-hidden="true"></i></span>
                            <input type="checkbox"
name="checkboxes" value="read">
                        </div>
                    </div>
                </div>


                <div class="form-group">
                    <button type="submit" class="btn btn-primary
```

```
btn-lg btn-block login-button">Submit</button>
                </div>
                <div class="login-register">
                    <a type="reset">Cancel</a>
                </div>
            </form>
        </div>
    </div>
</div>

<script type="text/javascript"
src="js/bootstrap.min.js"></script>
</body>
</html>
```

Thanks.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Basic Web Form</title>
    <link href="css/bootstrap.min.css" rel="stylesheet"
type="text/css">
</head>
<body>
<div class="alert alert-success" role="alert">Thank You!</div>
</body>
</html>
```

Section 3: WeleniumLibrary Configurations

AutomationSetup.java

```
package configurations;


/**
 * Created by stevenstretton on 25/09/2016.
 */
public class AutomationSetup {

    private FirefoxSettings firefoxSettings = new
FirefoxSettings();

    public void executeInitialisationSettings() {
        firefoxSettings.getBrowser();
    }

    public void endOfAutomationTest() {
        firefoxSettings.closeBrowser();
    }

}
```

FirefoxSettings.java

```
package configurations;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxProfile;
```

```
/**
 * Created by stevenstretton on 04/09/2016.
 */
public class FirefoxSettings {

    private FirefoxProfile firefoxProfile = new
FirefoxProfile();

    public static WebDriver driver = new FirefoxDriver();

    public static WebDriver navigateToPage(String webpage) {
        driver.get(webpage);
        return driver;
    }

    WebDriver getBrowser() {
        setupBrowser();
        return driver;
    }

    WebDriver closeBrowser() {
        driver.quit();
        return driver;
    }

    private WebDriver setupBrowser() {
        setBrowserSize();
        setBrowserHomepage();
        clearAllBrowserCookies();

        return driver;
    }

    private WebDriver setBrowserSize() {
        driver.manage().window().maximize();
        return driver;
    }

    private WebDriver setBrowserHomepage() {
        String BLANK_HOMEPAGE = "about:blank";

        firefoxProfile.setPreference("browser.startup.homepage",
BLANK_HOMEPAGE);

firefoxProfile.setPreference("startup.homepage_welcome_url",
BLANK_HOMEPAGE);

firefoxProfile.setPreference("startup.homepage_welcome_url.addit
ional", BLANK_HOMEPAGE);
        firefoxProfile.setPreference("network.proxy.type", "1");

        return driver;
    }

    private WebDriver clearAllBrowserCookies() {
        driver.manage().deleteAllCookies();
        return driver;
    }

}
```

WeleniumFirefoxDriver.java

```java
package configurations.drivers;

import configurations.FirefoxSettings;
import org.openqa.selenium.WebDriver;

import static configurations.FirefoxSettings.driver;

/**
 * Created by stevenstretton on 02/01/2017.
 */
public class WeleniumFirefoxDriver {

    public static void goToPage(String link) {
        goToFirefoxURL(link);
    }

    public static String getFirefoxSource() {
        return driver.getPageSource();
    }

    private static WebDriver goToFirefoxURL(String link) {
        return FirefoxSettings.navigateToPage(link);
    }

}
```

Section 4: WeleniumLibrary Welenium

VerifyElementFound.java

```java
package welenium;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import welenium.exceptions.ElementMissingFromPathException;

import static configurations.FirefoxSettings.driver;

/**
 * Created by stevenstretton on 25/10/2016.
 */
public class VerifyElementFound {
    private WeleniumLogger logger = new WeleniumLogger();

    public void processElementVerificationOnSendKey(String
value, String extractedElement)
            throws ElementMissingFromPathException {
        if (containsElement(extractedElement)) {
            processElementRequestOnSendKeys(value,
extractedElement);
        } else {
            throw new
ElementMissingFromPathException(extractedElement);
        }
    }

    public void processElementVerificationOnClick(String
extractedElement) throws ElementMissingFromPathException {
        if (containsElement(extractedElement)) {
            processElementRequestOnClick(extractedElement);
        } else {
            throw new
ElementMissingFromPathException(extractedElement);
        }
```

```
    }

    public void processElementVerificationOnRead(String
extractedElement) throws ElementMissingFromPathException {
        if (containsElement(extractedElement)) {
            processElementRequestOnRead(extractedElement);
        } else {
            throw new
ElementMissingFromPathException(extractedElement);
        }
    }

    private boolean containsElement(String extractedElement) {
        return extractedElement != null;
    }

    private void processElementRequestOnSendKeys(String value,
String passedElement) {
        By path = By.xpath(passedElement);
        logger.getGeneratedPathOnSendKey(passedElement);

        WebElement element = driver.findElement(path);
        element.sendKeys(value);
    }

    private void processElementRequestOnClick(String
passedElement) {
        By path = By.xpath(passedElement);
        logger.getGeneratedPathOnClick(passedElement);

        WebElement element = driver.findElement(path);
        element.click();
    }

    private void processElementRequestOnRead(String
passedElement) {
        By path = By.xpath(passedElement);
        logger.getGeneratedPathOnTextSearch(passedElement);

        WebElement element = driver.findElement(path);
        element.getText();
    }
}
```

WebElement.java

```
package welenium;

import welenium.exceptions.ElementMissingFromPathException;
import welenium.exceptions.NoElementFoundException;

/**
 * Created by stevenstretton on 25/10/2016.
 */
public class WebElement {
    private static WebStructure structure = new WebStructure();
    private static VerifyElementFound verify = new
VerifyElementFound();

    public void collectElement() {
        structure.getDomStructure();
    }

    void findStructureByElementForSendKey(String value, String
elementName)
```

```
            throws NoElementFoundException,
ElementMissingFromPathException {
        structure.scanStructure(elementName, true);
        verify.processElementVerificationOnSendKey(value,
structure.getExtractedPath());
    }

    void findStructureByElementForOnClick(String elementName)
            throws NoElementFoundException,
ElementMissingFromPathException {
        structure.scanStructure(elementName, true);

verify.processElementVerificationOnClick(structure.getExtractedP
ath());
    }

    void findStructureByElementForRead(String value) throws
ElementMissingFromPathException {
        structure.scanStructureForText(value);

verify.processElementVerificationOnRead(structure.getExtractedPa
th());
    }

}
```

WebStructure.java

```
package welenium;

import configurations.drivers.WeleniumFirefoxDriver;
import welenium.elements.XpathElementCollector;
import welenium.elements.XpathExpressionBuilder;
import welenium.exceptions.NoElementFoundException;

/**
 * Created by stevenstretton on 14/12/2016.
 */
public class WebStructure {
    private String extractedPath;
    private String domSource;

    String getExtractedPath() {
        return extractedPath;
    }

    void getDomStructure() {
        domSource = WeleniumFirefoxDriver.getFirefoxSource();
    }

    void scanStructure(String keyword, Boolean
isSendKeyAndClick) throws NoElementFoundException {
        if (containsValidValue(domSource, keyword)) {
            extractedPath = extractPathFromStructure(keyword,
isSendKeyAndClick);
        } else {
            throw new NoElementFoundException("The element
keyword you have entered" +
                    " is not associated to this webpage.
Element: " + keyword);
        }
    }

    void scanStructureForText(String text) {
        extractedPath = extractPathFromStructureForText(text);
```

```
    }


    private boolean containsValidValue(String
extractableDomSource, String keyword) {
        return extractableDomSource.contains(keyword);
    }

    private String extractPathFromStructure(String keyword,
Boolean isSendKeyAndClick) {
        XpathElementCollector xpathElementCollector = new
XpathElementCollector();
        xpathElementCollector.getHtmlElements(domSource,
keyword, isSendKeyAndClick);

        return xpathElementCollector.getGeneratedXpath();
    }

    private String extractPathFromStructureForText(String text)
{
        XpathExpressionBuilder xpathExpressionBuilder = new
XpathExpressionBuilder();
        xpathExpressionBuilder.createExpressionFromText(text);

        return xpathExpressionBuilder.getGeneratedXpath();
    }


}
```

WeleniumElement.java

```
package welenium;

import welenium.exceptions.ElementMissingFromPathException;
import welenium.exceptions.NoElementFoundException;

/**
 * Created by stevenstretton on 14/12/2016.
 */
public class WeleniumElement extends WebElement
{
    public void writeToPage(String value, String keyword) {
        try {
            findStructureByElementForSendKey(value, keyword);
        } catch (NoElementFoundException |
ElementMissingFromPathException e) {
            e.printStackTrace();
        }
    }

    public void readFromPage(String readElement)  {
        try {
            findStructureByElementForRead(readElement);
        } catch (ElementMissingFromPathException e) {
            e.printStackTrace();
        }
    }

    public void selectOnPage(String item)  {
        try {
            findStructureByElementForOnClick(item);
        } catch (NoElementFoundException |
ElementMissingFromPathException e) {
            e.printStackTrace();
```

```
        }
    }

}
```

WeleniumLogger.java

```
package welenium;

/**
 * Created by stevenstretton on 24/02/2017.
 */
public class WeleniumLogger {

    private int pathLineCount = 0;

    public void init() {
        header();
    }

    public void outputInfo(String message) {
        System.out.println("[INFO] " + message);
    }

    public void outputError(String message) {
        System.out.println("[ERROR] " + message);
    }

    void getGeneratedPathOnSendKey(String path) {
        System.out.println("[" + pathLineCount + "] <OnSendKey>
===> " + path);
        pathLineCount++;
    }

    void getGeneratedPathOnClick(String path) {
        System.out.println("[" + pathLineCount + "] <OnClick>
======> " + path);
        pathLineCount++;
    }

    void getGeneratedPathOnTextSearch(String path) {
        System.out.println("[" + pathLineCount + "] <OnTextFind>
==> " + path);
        pathLineCount++;
    }

    public void getTestOutputOnSuccess(String functionName,
String className) {
        System.out.println("[" + pathLineCount + "] Test Passed
==> "
                + functionName + " ON " + className);

        pathLineCount++;
    }

    public void getTestOutputOnFail(String functionName, String
className, String expected, String actual) {
        System.out.println("[" + pathLineCount + "] Test Failed
==> "
                + functionName + " ON " + className);
        System.out.println(verticalSingleDivider());
        System.out.println("Expected ==> " + expected
                + " || But was ==> " + actual);
        System.out.println(verticalSingleDivider());
```

```
        pathLineCount++;
    }

    private void header() {
        System.out.println(verticalDoubleDivider());
        System.out.println(title());
        System.out.println(verticalSingleDivider());
        System.out.println(author());
        System.out.println(verticalDoubleDivider());
        System.getProperty("line.separator");
    }


    private String title() {
        return "__          __   _            _
\n" + "\\ \\     " +
                " / /__| | ___ _ __ (_)_   _ _ __ ___  \n" +
                " \\ \\ /\\ / / _ \\ |/ _ \\ '_ \\| | | | '_ `
_ \\ \\n" +
                "  \\ V  V /  __/ |  __/ | | | | |_| | | | | | |
|\n" +
                "   \\_/\\_/ \\___|_|\\___|_| |_|_|\\__,_|_| |_|
|_|";
    }

    private String author() {
        return "Developed by Steven Stretton";
    }

    private String verticalSingleDivider() {
        return "----------------------------------------------
-----";
    }

    private String verticalDoubleDivider() {
        return
"==========================================================";
    }

}
```

HtmlAttributeLibrary.java

```
package welenium.elements.library;

/**
 * Created by stevenstretton on 17/01/2017.
 */
public enum HtmlAttributeLibrary {
    //html area
    HTML("html"),
    BODY("body"),

    //table
    TABLE("table"),
    TR("tr"),
    TD("td"),

    //form
    FORM("form"),
    FIELDSET("fieldset"),
    INPUT("input"),
    LABEL("label"),
    BUTTON("button"),
    TEXTAREA("textarea"),
```

```
    //attribute elements
    TAG("<"),
    ID("id"),
    CLASS("class"),
    NAME("name"),
    TYPE("type"),
    PLACEHOLDER("placeholder"),
    H_ONE("h1"),
    FOR("for"),

    ATTRIBUTE_EQUAL("=\"");

    private final String elementName;

    HtmlAttributeLibrary(String htmlName) {
        elementName = htmlName;
    }

    public final String getHtmlAttributeName() {
        return elementName;
    }


}
```

XpathLibrary.java

```
package welenium.elements.library;

/**
 * Created by stevenstretton on 21/02/2017.
 */
public enum XpathLibrary {

    XPATH_CONTAINS_OPENING_TAG("//*["),
    XPATH_CONTAINS_QUERY("contains("),
    XPATH_CONTAINS_CLOSING_TAG("')]"),
    XPATH_AND_TAG("') AND ('"),

    XPATH_TYPE_TAG("@type,'"),
    XPATH_ID_TAG("@id,'"),
    XPATH_TEXT_TAG("text(),'"),
    XPATH_NAME_TAG("@name,'"),
    XPATH_PLACEHOLDER_TAG("@placeholder,'");

    private final String tagName;

    XpathLibrary(String htmlName) {
        tagName = htmlName;
    }

    public String getTagName() {
        return tagName;
    }


}
```

HtmlAttributeFinder.java

```
package welenium.elements;

import welenium.elements.library.HtmlAttributeLibrary;
```

```
import static welenium.elements.library.HtmlAttributeLibrary.*;

/**
 * Created by stevenstretton on 07/02/2017.
 */
public class HtmlAttributeFinder {

    private String htmlElementFromExtractor;
    private String collectedAttribute;
    private final String START_ATTRIBUTE_TAG = "=\"";
    private final String END_ATTRIBUTE_TAG = "\"";

    public String getAttributeFromElement(String
htmlElementFromExtractor) {

        this.htmlElementFromExtractor =
htmlElementFromExtractor;

        HtmlAttributeLibrary attributeCollection[] = {ID, NAME,
PLACEHOLDER, TYPE};

        for (HtmlAttributeLibrary givenAttribute :
attributeCollection) {
            if
(htmlElementFromExtractor.contains(givenAttribute.getHtmlAttribu
teName())) {

callAttributeProcess(givenAttribute.getHtmlAttributeName());
                break;
            }
        }

        return collectedAttribute;
    }

    private void callAttributeProcess(String providedAttribute)
{
        if (providedAttribute.equals(ID.getHtmlAttributeName()))
{
            processIdAttribute();

        } else if
(providedAttribute.equals(NAME.getHtmlAttributeName())) {
            processNameAttribute();

        } else if
(providedAttribute.equals(PLACEHOLDER.getHtmlAttributeName())) {
            processPlaceholderAttribute();

        } else if
(providedAttribute.equals(TYPE.getHtmlAttributeName())) {
            processTypeAttribute();

        }
    }

    private String processIdAttribute() {
        String[] trimElementLeft =
htmlElementFromExtractor.split(ID.getHtmlAttributeName() +
START_ATTRIBUTE_TAG);
        String[] trimElementRight =
trimElementLeft[1].split(END_ATTRIBUTE_TAG, 2);

        return collectedAttribute = trimElementRight[0];
    }
```

```
    private String processNameAttribute() {
        String[] trimElementLeft =
htmlElementFromExtractor.split(NAME.getHtmlAttributeName() +
START_ATTRIBUTE_TAG);
        String[] trimElementRight =
trimElementLeft[1].split(END_ATTRIBUTE_TAG, 2);

        return collectedAttribute = trimElementRight[0];
    }

    private String processPlaceholderAttribute() {
        String[] trimElementLeft =
htmlElementFromExtractor.split(PLACEHOLDER.getHtmlAttributeName(
)
                + START_ATTRIBUTE_TAG);
        String[] trimElementRight =
trimElementLeft[1].split(END_ATTRIBUTE_TAG, 2);

        return collectedAttribute = trimElementRight[0];
    }

    private String processTypeAttribute() {
        String[] trimElementLeft =
htmlElementFromExtractor.split(TYPE.getHtmlAttributeName() +
START_ATTRIBUTE_TAG);
        String[] trimElementRight =
trimElementLeft[1].split(END_ATTRIBUTE_TAG, 2);

        return collectedAttribute = trimElementRight[0];
    }

}
```

HtmlExtractor.java

```
package welenium.elements;

import static welenium.elements.library.HtmlAttributeLibrary.*;

/**
 * Created by stevenstretton on 31/01/2017.
 */
public class HtmlExtractor {

    String getContentHtml(String htmlIsolatedElement) {
        return findHtmlContent(htmlIsolatedElement);
    }

    String getAttributesHtml(String htmlAttributes) {
        return findHtmlAttributes(htmlAttributes);
    }

    String getElementType(String htmlElementType) {
        return findHtmlElementByType(htmlElementType);
    }

    private String findHtmlContent(String htmlIsolatedElement) {
        String REMOVED_CHAR = "";

        String[] trimElementLeft =
htmlIsolatedElement.split("(?=" + TAG.getHtmlAttributeName() +
")");
        String[] trimElementRight = trimElementLeft[1].split("
", 2);
```

```
        return
trimElementRight[0].replaceAll(String.valueOf(TAG.getHtmlAttribu
teName()), REMOVED_CHAR);
    }

    private String findHtmlElementByType(String htmlElementType)
{
        HtmlAttributeFinder htmlAttributeFinder = new
HtmlAttributeFinder();

        return
htmlAttributeFinder.getAttributeFromElement(htmlElementType);
    }

    private String findHtmlAttributes(String htmlAttributes) {
        String foundHtmlAttribute = "";

        if (findHtmlAttributeById(htmlAttributes)) {
            foundHtmlAttribute = ID.getHtmlAttributeName();
        } else if (findHtmlAttributeByName(htmlAttributes)) {
            foundHtmlAttribute = NAME.getHtmlAttributeName();
        } else if
(findHtmlAttributeByPlaceholder(htmlAttributes)) {
            foundHtmlAttribute =
PLACEHOLDER.getHtmlAttributeName();
        }

        return foundHtmlAttribute;
    }

    private boolean findHtmlAttributeByName(String
htmlAttributes) {
        return
htmlAttributes.contains(NAME.getHtmlAttributeName());
    }

    private boolean findHtmlAttributeById(String htmlAttributes)
{
        return
htmlAttributes.contains(ID.getHtmlAttributeName());
    }

    private boolean findHtmlAttributeByPlaceholder(String
htmlAttributes) {
        return
htmlAttributes.contains(PLACEHOLDER.getHtmlAttributeName());
    }


}
```

XpathAttributeGenerator.java

```
package welenium.elements;

import welenium.elements.library.XpathLibrary;

import static welenium.elements.library.HtmlAttributeLibrary.*;


/**
 * Created by stevenstretton on 23/02/2017.
 */
public class XpathAttributeGenerator {
```

```
    private XpathAttributeTagFinder xpathAttributeTagFinder =
new XpathAttributeTagFinder();
    private String setAttribute = "";

    public String getAttribute(String element, String keyword,
String attribute) {
        typeAllocator(element, keyword, attribute);

        return setAttribute;
    }

    public String getAttributeForText(String text) {
        return setAttribute = getTextAttribute(text);
    }

    private void typeAllocator(String element, String keyword,
String attribute) {
        if (element.equals(BUTTON.getHtmlAttributeName())) {
            setAttribute = getButtonAttribute(keyword);

        } else if (element.equals(INPUT.getHtmlAttributeName()))
{
            setAttribute = getInputAttribute(keyword,
attribute);

        } else if
(element.equals(TEXTAREA.getHtmlAttributeName())) {
            setAttribute = getTextAreaAttribute(keyword,
attribute);

        } else {
            System.out.println("Attribute not in catalog");
        }
    }

    private String getInputAttribute(String keyword, String
attribute) {
        return XpathLibrary.XPATH_CONTAINS_QUERY.getTagName()
                +
xpathAttributeTagFinder.getXpathAttributeTag(attribute)
                + keyword;
    }

    private String getTextAreaAttribute(String keyword, String
attribute) {
        return XpathLibrary.XPATH_CONTAINS_QUERY.getTagName()
                +
xpathAttributeTagFinder.getXpathAttributeTag(attribute)
                + keyword;
    }

    private String getButtonAttribute(String keyword) {
        return XpathLibrary.XPATH_CONTAINS_QUERY.getTagName()
                + XpathLibrary.XPATH_TYPE_TAG.getTagName()
                + keyword;
    }

    private String getTextAttribute(String keyword) {
        return XpathLibrary.XPATH_CONTAINS_QUERY.getTagName()
                + XpathLibrary.XPATH_TEXT_TAG.getTagName()
                + keyword;
    }
}
```

XpathAttributeTagFinder.java

```java
package welenium.elements;

import welenium.WeleniumLogger;
import welenium.elements.library.XpathLibrary;

/**
 * Created by stevenstretton on 03/03/2017.
 */
public class XpathAttributeTagFinder {

    private WeleniumLogger weleniumLogger = new
WeleniumLogger();
    private String attribute;
    private String collectedAttributeTag;

    public String getXpathAttributeTag(String attribute) {
        this.attribute = attribute;
        findTagAssociateToKeyword();

        return this.collectedAttributeTag;
    }

    private void findTagAssociateToKeyword() {
        switch (attribute) {
            case "id":
                getIdTag();
                break;
            case "name":
                getNameTag();
                break;
            case "placeholder":
                getPlaceholderTag();
                break;
            case "type":
                getTypeTag();
                break;
            default:
                weleniumLogger.outputError("invalid attribute");
                break;
        }
    }

    private void getIdTag() {
        this.collectedAttributeTag =
XpathLibrary.XPATH_ID_TAG.getTagName();
    }

    private void getNameTag() {
        this.collectedAttributeTag =
XpathLibrary.XPATH_NAME_TAG.getTagName();
    }

    private void getPlaceholderTag() {
        this.collectedAttributeTag =
XpathLibrary.XPATH_PLACEHOLDER_TAG.getTagName();
    }

    private void getTypeTag() {
        this.collectedAttributeTag =
XpathLibrary.XPATH_TYPE_TAG.getTagName();
    }
}
```

XpathElementCollector.java

```
package welenium.elements;

import org.springframework.util.StringUtils;
import welenium.elements.library.HtmlAttributeLibrary;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by stevenstretton on 17/01/2017.
 */
public class XpathElementCollector {

    private String element;
    private String generatedXpath = "";
    private List<String> elements = new ArrayList<>();

    public void getHtmlElements(String htmlSource, String
keyword, Boolean isSendKeyAndClick) {
        this.element = keyword;
        collectAttributes();
        generateXpathElement(htmlSource, isSendKeyAndClick);
    }

    public List<String> getCollectedAttributes() {
        return this.collectAttributes();
    }

    public String getGeneratedXpath() {
        return generatedXpath;
    }

    private List<String> collectAttributes() {
        for (HtmlAttributeLibrary htmlTagLibrary :
HtmlAttributeLibrary.values()) {
            elements.add(htmlTagLibrary.getHtmlAttributeName());
        }

        return elements;
    }

    private String[] breakDownStructure(String htmlSource) {
        final String NEW_LINE = "\n";

        int ONE = 1;
        int newLineCount =
StringUtils.countOccurrencesOf(htmlSource, NEW_LINE) + ONE;
        int newLine = 0;
        String[] elementPerLine = new String[newLineCount];

        for (String temporaryElement :
htmlSource.split(NEW_LINE)) {
            getElementPerLine(elementPerLine, temporaryElement,
newLine);
            newLine++;
        }

        return elementPerLine;
    }

    private String getElementPerLine(String[] elementPerLine,
String temporaryElement, int newLine) {
        return elementPerLine[newLine] = temporaryElement;
    }
```

```
    private void generateXpathElement(String htmlSource, Boolean
isSendKeyAndClick) {
        HtmlExtractor htmlExtractor = new HtmlExtractor();

        for (String foundElementInStructure :
breakDownStructure(htmlSource)) {

            if (foundElementInStructure.contains(element)) {

                if (isSendKeyAndClick) {

                    if
(foundElementInStructure.contains(element) &&

!foundElementInStructure.contains(HtmlAttributeLibrary.FOR.getHt
mlAttributeName()
                                    +
HtmlAttributeLibrary.ATTRIBUTE_EQUAL.getHtmlAttributeName()
                                    + element)
                            ) {

                        if
(!foundElementInStructure.contains(HtmlAttributeLibrary.CLASS.ge
tHtmlAttributeName()
                                +
HtmlAttributeLibrary.ATTRIBUTE_EQUAL.getHtmlAttributeName()
                                    + element)) {
                            processElementToPath(htmlExtractor,
foundElementInStructure);
                            break;
                        }
                    }

                } else {
                    processElementToPath(htmlExtractor,
foundElementInStructure);
                    break;
                }

            }

        }
    }

    private void processElementToPath(HtmlExtractor
htmlExtractor, String foundElementInStructure) {
        String getContentElement =
htmlExtractor.getContentHtml(foundElementInStructure);
        String getContentAttribute =
htmlExtractor.getAttributesHtml(foundElementInStructure);
        String getContentType =
htmlExtractor.getElementType(foundElementInStructure);

        validateXPathElement(getContentElement);
        validateXPathElement(getContentAttribute);

        generateXpathStructure(getContentElement,
getContentType, getContentAttribute);
    }


    private boolean validateXPathElement(String
collectedElement) {
        boolean doesMatch = false;

        for (String foundElement : elements) {
            if (collectedElement.equals(foundElement)) {
```

```
                doesMatch = true;
                break;
            }
        }

        return doesMatch;
    }

    private void generateXpathStructure(String element, String
type, String attribute) {
        XpathExpressionBuilder xpathExpressionBuilder = new
XpathExpressionBuilder();
        xpathExpressionBuilder.createExpression(element, type,
attribute);
        generatedXpath =
xpathExpressionBuilder.getGeneratedXpath();
    }


}
```

XpathExpressionBuilder.java

```
package welenium.elements;

import welenium.elements.library.XpathLibrary;

import java.util.List;

/**
 * Created by stevenstretton on 31/01/2017.
 */
public class XpathExpressionBuilder {

    private String finalisedXpath = "";

    public void createExpression(String element, String type,
String attribute) {
        if (isElementValid(element)) {
            finalisedXpath = attributeAllocator(element, type,
attribute, false);
        }
    }

    public void createExpressionFromText(String text) {
        finalisedXpath = attributeAllocator(null, text, null,
true);
    }

    public String getGeneratedXpath() {
        return finalisedXpath;
    }

    private String attributeAllocator(String element, String
type, String attribute, boolean forTextSearch) {
        XpathAttributeGenerator xpathAttributeGenerator = new
XpathAttributeGenerator();
        String setAttribute;

        if (forTextSearch) {
            setAttribute =
xpathAttributeGenerator.getAttributeForText(type);
        } else {
            setAttribute =
xpathAttributeGenerator.getAttribute(element, type, attribute);
```

```
        }

        return String.valueOf(

XpathLibrary.XPATH_CONTAINS_OPENING_TAG.getTagName()) +
                setAttribute +

XpathLibrary.XPATH_CONTAINS_CLOSING_TAG.getTagName();
    }


    private boolean isElementValid(String element) {
        boolean isConfirmed = false;

        XpathElementCollector xpathElementCollector = new
XpathElementCollector();
        List<String> collectedAttributes =
xpathElementCollector.getCollectedAttributes();

        for (String collectedAttribute : collectedAttributes) {
            if (collectedAttribute.equals(element)) {
                isConfirmed = true;
                break;
            }
        }

        return isConfirmed;
    }
}
```

ElementMissingFromPathException.java

```
package welenium.exceptions;

/**
 * Created by stevenstretton on 25/10/2016.
 */
public class ElementMissingFromPathException extends Exception {
    public ElementMissingFromPathException(String message) {
        super("Element is missing from generated Xpath. Produced
path: " + message);
    }
}
```

NoElementFoundException.java

```
package welenium.exceptions;

/**
 * Created by stevenstretton on 25/10/2016.
 */
public class NoElementFoundException extends Exception {
    public NoElementFoundException(String message) {
        super(message);
    }
}
```

Section 5: WeleniumLibrary Tests

VerifyHtmlAttributeLibraryContainsAttributes.java

```
import org.junit.Test;
import welenium.WeleniumLogger;
import welenium.elements.library.HtmlAttributeLibrary;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * Created by stevenstretton on 21/03/2017.
 */
public class VerifyHtmlAttributeLibraryContainsAttributes {

    private List<String> elements = new ArrayList<String>();
    private WeleniumLogger weleniumLogger = new
WeleniumLogger();
    private String testClass = this.getClass().getName();

    @Test
    public void verifyHtmlAttributeLibraryContainsAttributes() {
        weleniumLogger.outputInfo("Welenium Library Test " +
testClass);

        this.getHtmlLibraryContent();

        this.validateElements("id");
        this.validateElementSize(20);

    }

    private List<String> getHtmlLibraryContent() {
        for (HtmlAttributeLibrary htmlTagLibrary :
HtmlAttributeLibrary.values()) {
            elements.add(htmlTagLibrary.getHtmlAttributeName());
        }

        return elements;
    }

    private void validateElements(String element) {
        String currentMethod = "validateElements";

        if (elements.contains(element)) {
            weleniumLogger.getTestOutputOnSuccess(currentMethod,
testClass);
        } else {
            weleniumLogger.getTestOutputOnFail(currentMethod,
testClass,
                    element,
Arrays.toString(elements.toArray()));
        }
    }

    private void validateElementSize(int size) {
        String currentMethod = "validateElementSize";

        if (elements.size() == size) {
            weleniumLogger.getTestOutputOnSuccess(currentMethod,
testClass);
        } else {
            weleniumLogger.getTestOutputOnFail(currentMethod,
testClass,
                    String.valueOf(size),
String.valueOf(elements.size()));
        }
    }
```

```
}
```

VerifyXpathCanBeGenerated.java

```
import org.junit.Test;
import welenium.WeleniumLogger;
import welenium.elements.library.XpathLibrary;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * Created by stevenstretton on 21/03/2017.
 */
public class VerifyXpathCanBeGenerated {

    private List<String> elements = new ArrayList<>();
    private WeleniumLogger weleniumLogger = new
WeleniumLogger();
    private String testClass = this.getClass().getName();

    @Test
    public void verifyXpathCanBeGenerated() {
        weleniumLogger.outputInfo("Welenium Library Test " +
testClass);

        this.getHtmlLibraryContent();

        this.validateElements("@type,'");
        this.validateElementSize(9);

    }

    private List<String> getHtmlLibraryContent() {
        for (XpathLibrary xpathLibrary : XpathLibrary.values())
{
            elements.add(xpathLibrary.getTagName());
        }

        return elements;
    }

    private void validateElements(String element) {
        String currentMethod = "validateElements";

        if (elements.contains(element)) {
            weleniumLogger.getTestOutputOnSuccess(currentMethod,
testClass);
        } else {
            weleniumLogger.getTestOutputOnFail(currentMethod,
testClass,
                    element,
Arrays.toString(elements.toArray()));
        }
    }

    private void validateElementSize(int size) {
        String currentMethod = "validateElementSize";

        if (elements.size() == size) {
            weleniumLogger.getTestOutputOnSuccess(currentMethod,
testClass);
        } else {
```

```
            weleniumLogger.getTestOutputOnFail(currentMethod,
testClass,
                    String.valueOf(size),
String.valueOf(elements.size()));
        }
    }
}
```

Appendix E – Terminal Log Output

Section 1: Test A

```
stevenstretton@Stevens-iMac ~/git/repos/final-year-project (master) $mvn verify -f UIBox/pom.xml -Dtest=weleniumTests.automationTests.TestA
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building UIBox 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ UIBox ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ UIBox ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ UIBox ---
[INFO] Surefire report directory: /Users/stevenstretton/git/repos/final-year-project/UIBox/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running weleniumTests.automationTests.TestA
collecting DOM element...
collecting DOM element...
=====================================================
__        __   _                            _
\ \      / /__| | ___ ___  _ __ ___   ___  | |_ ___
 \ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \ | __/ _ \
  \ V  V /  __/ | (_| (_) | | | | | |  __/ | || (_) |
   \_/\_/ \___|_|\___\___/|_| |_| |_|\___|  \__\___/
-------------------------------------------------------
Developed by Steven Stretton
=====================================================
[0] <OnSendKey> ===> //*[contains(@id,'name')]
[1] <OnSendKey> ===> //*[contains(@id,'address')]
[2] <OnSendKey> ===> //*[contains(@id,'email')]
[3] <OnSendKey> ===> //*[contains(@id,'telephone')]
[4] <OnSendKey> ===> //*[contains(@id,'date')]
[5] <OnSendKey> ===> //*[contains(@id,'additional')]
[6] <OnClick> =====> //*[contains(@id,'checkbox')]
[7] <OnClick> =====> //*[contains(@type,'submit')]
collecting DOM element...
[8] <OnTextFind> ==> //*[contains(text(),'Thank You!')]
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 46.604 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ UIBox ---
[WARNING] JAR will be empty - no content was marked for inclusion!
[INFO] Building jar: /Users/stevenstretton/git/repos/final-year-project/UIBox/target/UIBox-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 56.777 s
[INFO] Finished at: 2017-04-10T18:11:39+01:00
[INFO] Final Memory: 13M/125M
[INFO] ------------------------------------------------------------------------
```

Section 2: Test B

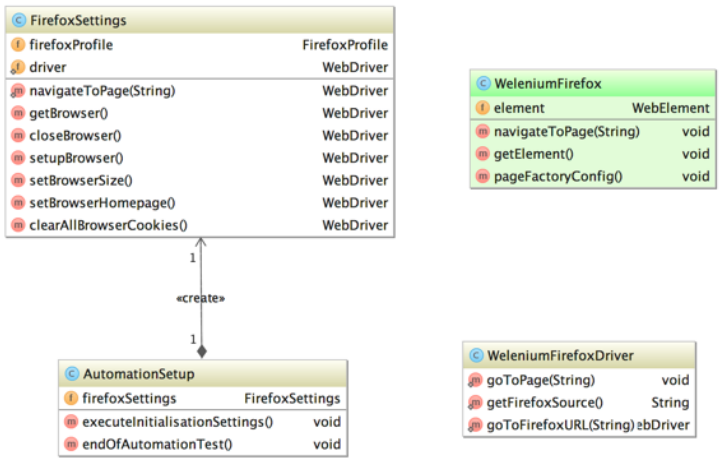
```
stevenstretton@Stevens-iMac ~/git/repos/final-year-project (master) $mvn verify -f UIBox/pom.xml -Dtest=weleniumTests.automationTests.TestB
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building UIBox 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ UIBox ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ UIBox ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ UIBox ---
[INFO] Surefire report directory: /Users/stevenstretton/git/repos/final-year-project/UIBox/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running weleniumTests.automationTests.TestB
collecting DOM element...
collecting DOM element...
=======================================================
__          __   _                            _
\ \        / /__| | ___ ___  _ __ ___   ___  | |
 \ \  /\  / / _ \ |/ __/ _ \| '_ ` _ \ / _ \ | |
  \ \/  \/ /  __/ | (_| (_) | | | | | |  __/ |_|
   \_/\_/ \___|_|\___\___/|_| |_| |_|\___| (_)
-------------------------------------------------------
Developed by Steven Stretton
=======================================================
[0] <OnSendKey> ===> //*[contains(@id,'name')]
[1] <OnSendKey> ===> //*[contains(@id,'address')]
[2] <OnSendKey> ===> //*[contains(@id,'email')]
[3] <OnSendKey> ===> //*[contains(@id,'telephone')]
[4] <OnSendKey> ===> //*[contains(@id,'date')]
[5] <OnSendKey> ===> //*[contains(@id,'additional')]
[6] <OnClick> ======> //*[contains(@id,'checkbox')]
[7] <OnClick> ======> //*[contains(@type,'submit')]
collecting DOM element...

 [8] <OnTextFind> ==> //*[contains(text(),'Thank You!')]
 Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 24.741 sec

 Results :

 Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

 [INFO]
 [INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ UIBox ---
 [WARNING] JAR will be empty - no content was marked for inclusion!
 [INFO] ------------------------------------------------------------------------
 [INFO] BUILD SUCCESS
 [INFO] ------------------------------------------------------------------------
 [INFO] Total time: 32.528 s
 [INFO] Finished at: 2017-04-10T18:16:11+01:00
 [INFO] Final Memory: 13M/125M
 [INFO] ------------------------------------------------------------------------
```

Section 3: Test C

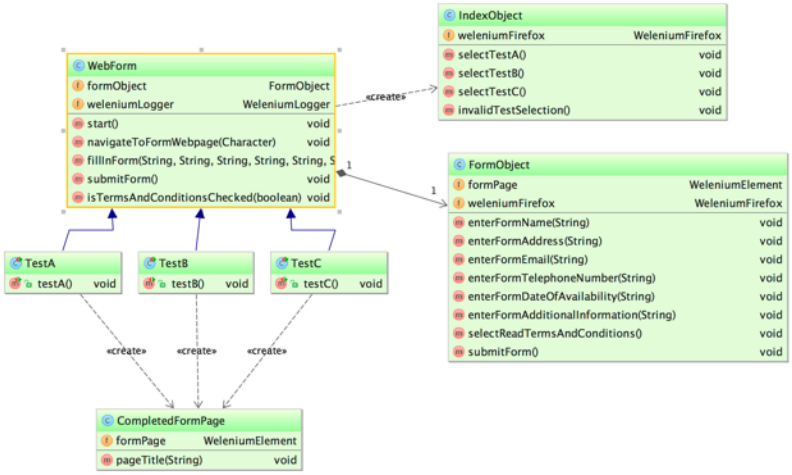
```
stevenstretton@Stevens-iMac ~/git/repos/final-year-project (master) $mvn verify -f UIBox/pom.xml -Dtest=weleniumTests.automationTests.TestB
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building UIBox 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ UIBox ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ UIBox ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/stevenstretton/git/repos/final-year-project/UIBox/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ UIBox ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ UIBox ---
[INFO] Surefire report directory: /Users/stevenstretton/git/repos/final-year-project/UIBox/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running weleniumTests.automationTests.TestB
collecting DOM element...
collecting DOM element...
=======================================================
__        __   _                            _
\ \      / /__| | ___ _ __ (_)_   _ _ __ ___
 \ \ /\ / / _ \ |/ _ \ '_ \| | | | | '_ ` _ \
  \ V  V /  __/ |  __/ | | | | |_| | | | | | |
   \_/\_/ \___|_|\___|_| |_|_|\__,_|_| |_| |_|
-------------------------------------------------------
Developed by Steven Stretton
=======================================================
[0] <OnSendKey> ===> //*[contains(@id,'name')]
[1] <OnSendKey> ===> //*[contains(@id,'address')]
[2] <OnSendKey> ===> //*[contains(@id,'email')]
[3] <OnSendKey> ===> //*[contains(@id,'telephone')]
[4] <OnSendKey> ===> //*[contains(@id,'date')]
[5] <OnSendKey> ===> //*[contains(@id,'additional')]
[6] <OnClick> =====> //*[contains(@id,'checkbox')]
[7] <OnClick> =====> //*[contains(@type,'submit')]
collecting DOM element...
[8] <OnTextFind> ==> //*[contains(text(),'Thank You!')]
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 24.741 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ UIBox ---
[WARNING] JAR will be empty - no content was marked for inclusion!
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 32.528 s
[INFO] Finished at: 2017-04-10T18:16:11+01:00
[INFO] Final Memory: 13M/125M
[INFO] ------------------------------------------------------------------------
```

Appendix F – Class Diagram

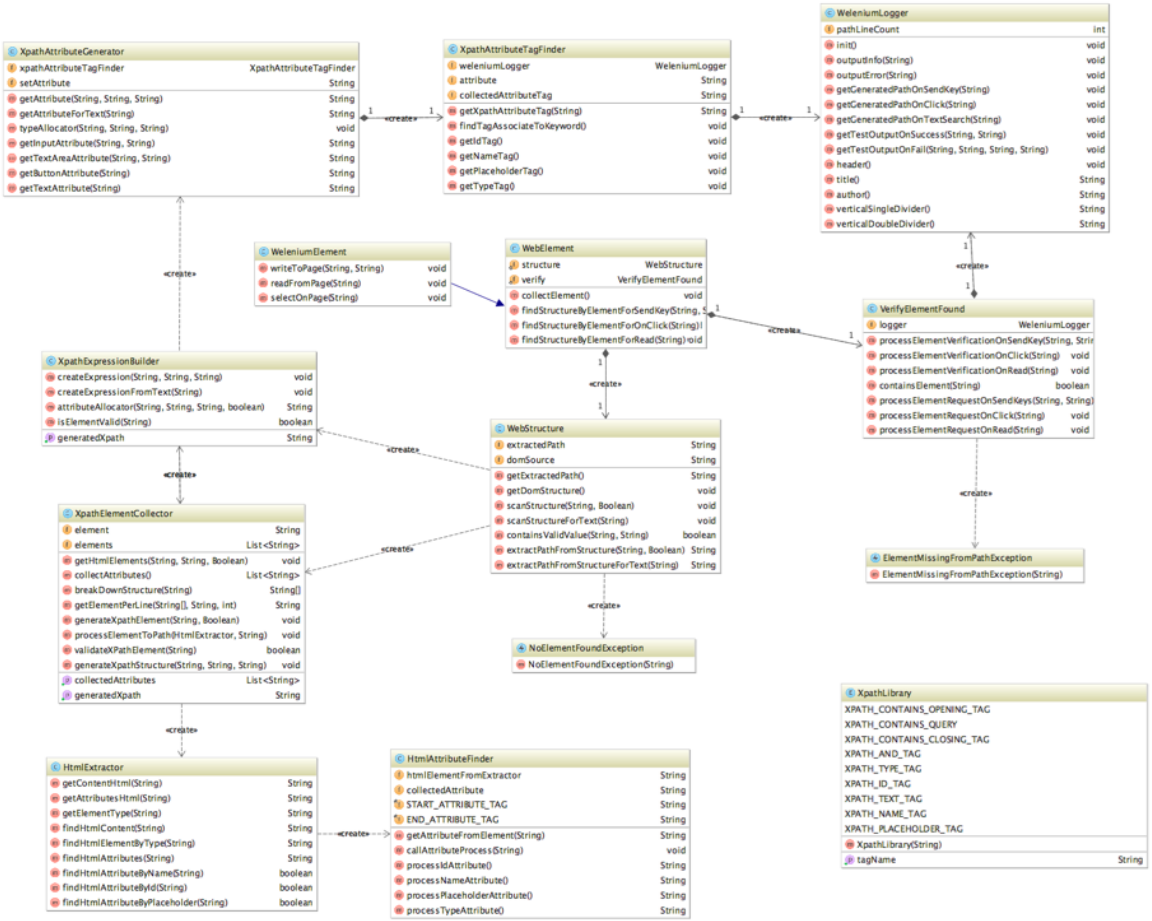
Section 1: Configurations



Section 2: WeleniumTests



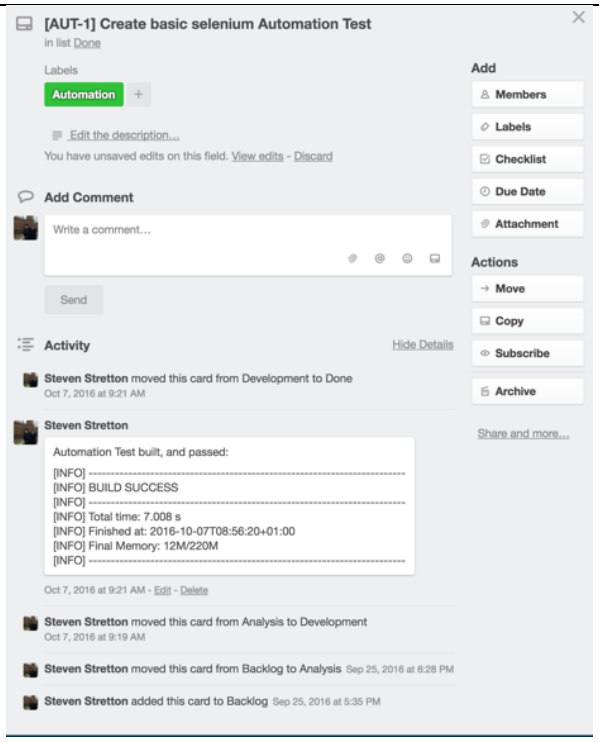
Section 3: Welenium



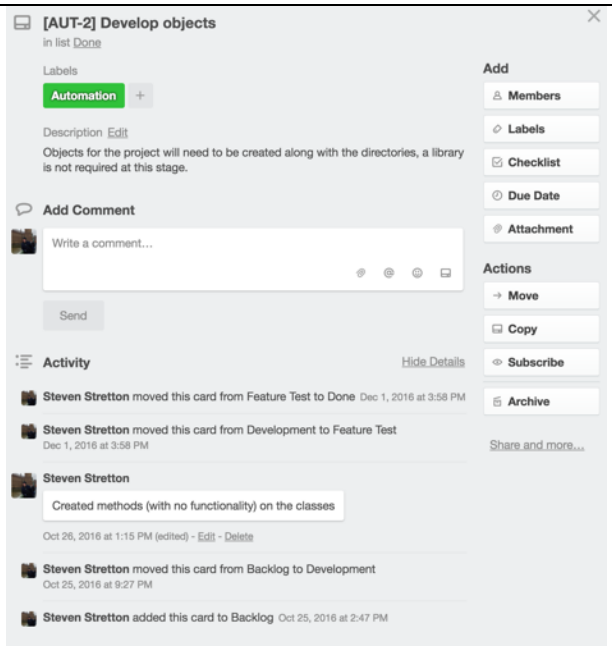
Appendix G – Agile

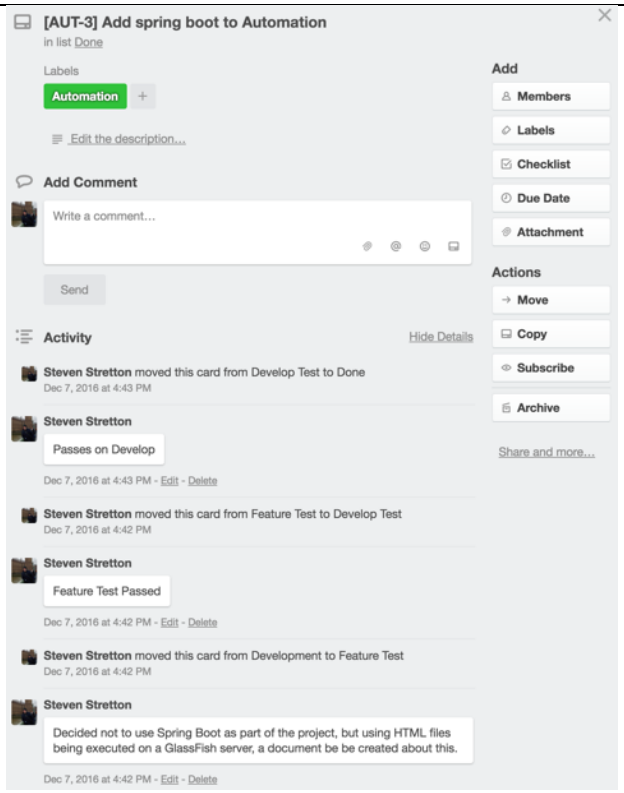
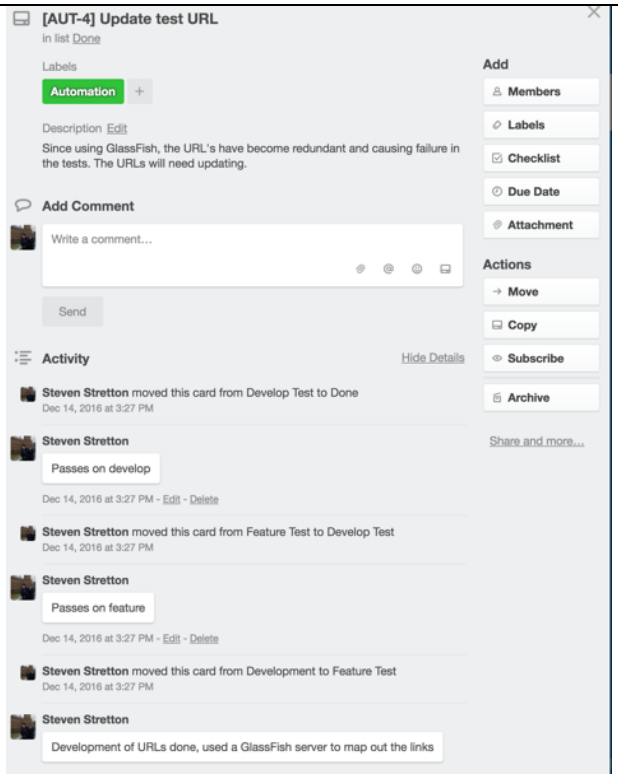
Section 1: Automation Tickets

AUT-1

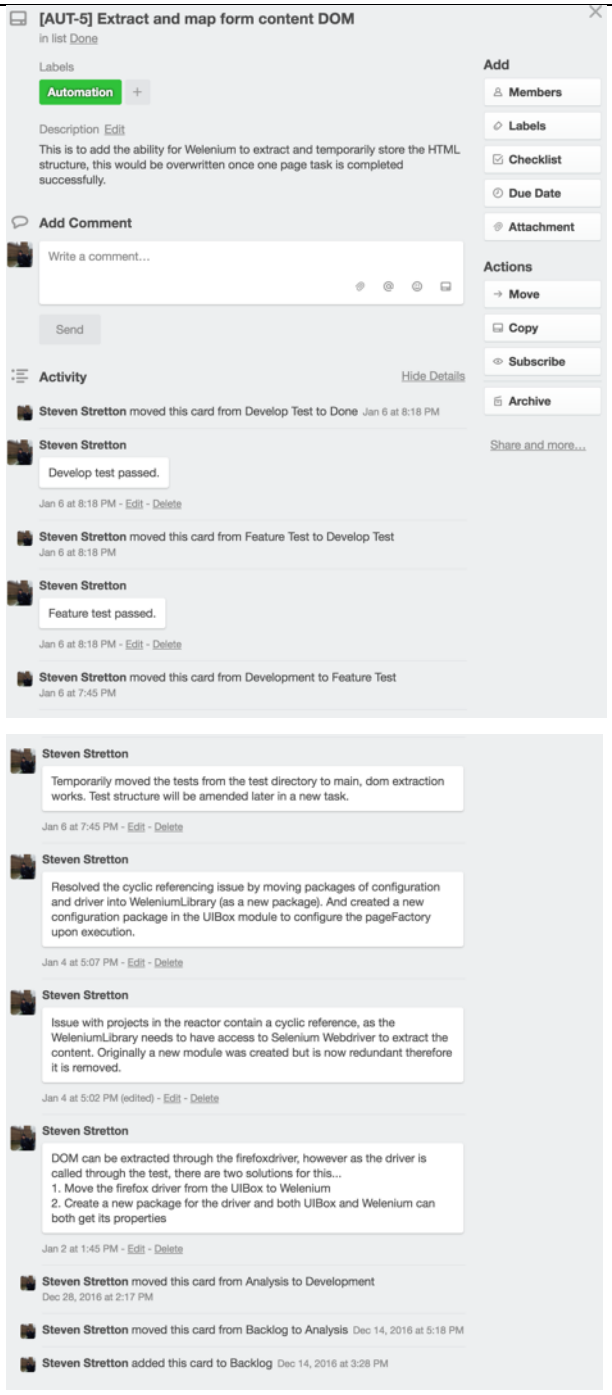


AUT-2

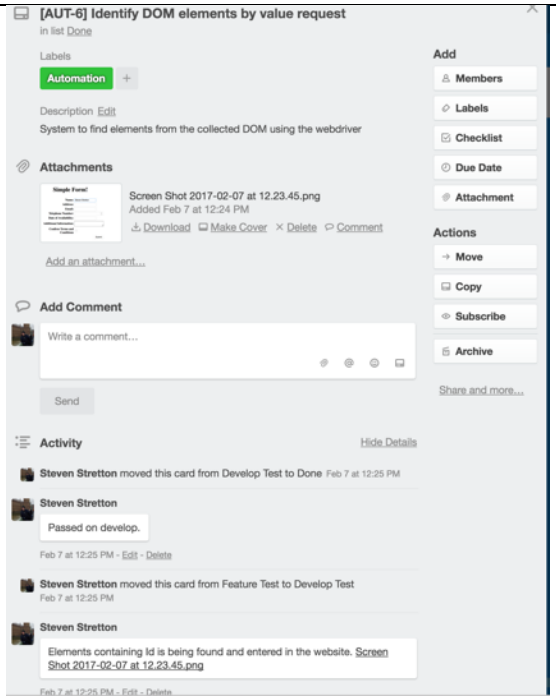


AUT-3	 [AUT-3] Add spring boot to Automation in list Done Labels Automation Edit the description... Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Dec 7, 2016 at 4:43 PM Steven Stretton Passes on Develop Dec 7, 2016 at 4:43 PM - Edit - Delete Steven Stretton moved this card from Feature Test to Develop Test Dec 7, 2016 at 4:42 PM Steven Stretton Feature Test Passed Dec 7, 2016 at 4:42 PM - Edit - Delete Steven Stretton moved this card from Development to Feature Test Dec 7, 2016 at 4:42 PM Steven Stretton Decided not to use Spring Boot as part of the project, but using HTML files being executed on a GlassFish server, a document be be created about this. Dec 7, 2016 at 4:42 PM - Edit - Delete Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more...
AUT-4	 [AUT-4] Update test URL in list Done Labels Automation Description Edit Since using GlassFish, the URL's have become redundant and causing failure in the tests. The URLs will need updating. Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Dec 14, 2016 at 3:27 PM Steven Stretton Passes on develop Dec 14, 2016 at 3:27 PM - Edit - Delete Steven Stretton moved this card from Feature Test to Develop Test Dec 14, 2016 at 3:27 PM Steven Stretton Passes on feature Dec 14, 2016 at 3:27 PM - Edit - Delete Steven Stretton moved this card from Development to Feature Test Dec 14, 2016 at 3:27 PM Steven Stretton Development of URLs done, used a GlassFish server to map out the links Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more...

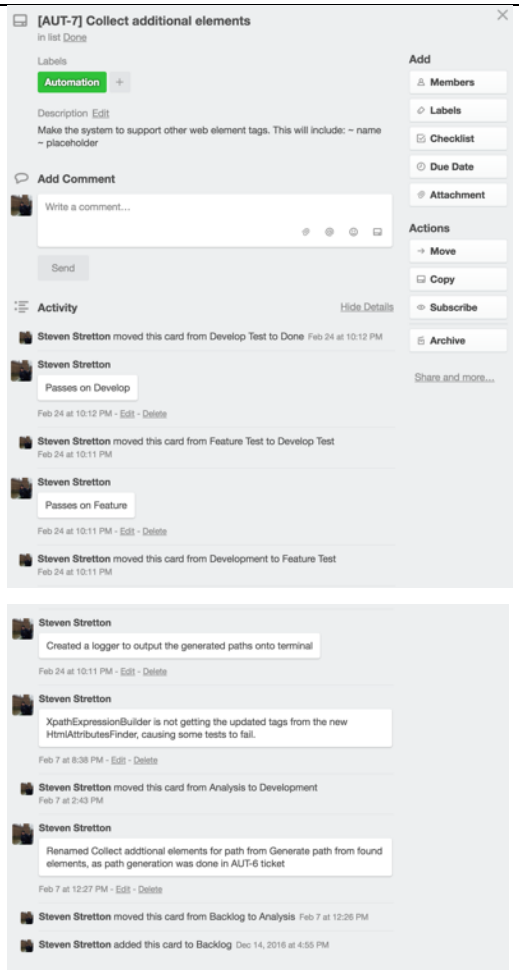
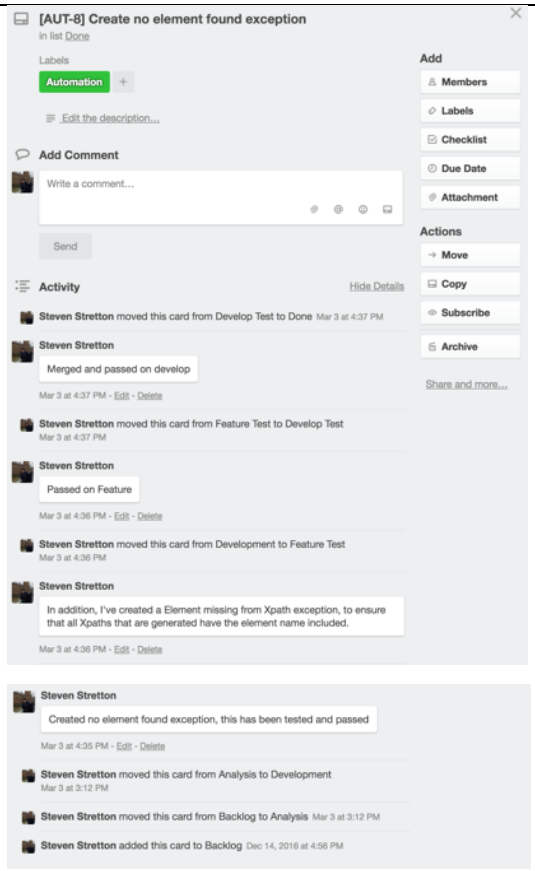
AUT-5

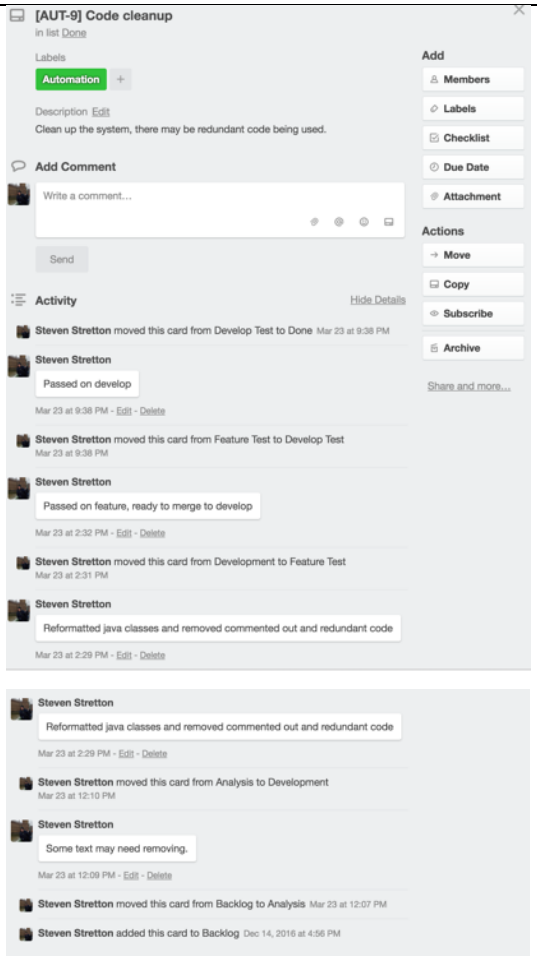

[AUT-5] Extract and map form content DOM
in list Done
Labels
Automation
Description Edit
This is to add the ability for Welenium to extract and temporarily store the HTML structure, this would be overwritten once one page task is completed successfully.
Add Comment
Write a comment...
Send
Activity
Hide Details
Steven Stretton moved this card from Develop Test to Done Jan 6 at 8:18 PM
Steven Stretton
Develop test passed.
Jan 6 at 8:18 PM - Edit - Delete
Steven Stretton moved this card from Feature Test to Develop Test
Jan 6 at 8:18 PM
Steven Stretton
Feature test passed.
Jan 6 at 8:18 PM - Edit - Delete
Steven Stretton moved this card from Development to Feature Test
Jan 6 at 7:45 PM
Add
Members
Labels
Checklist
Due Date
Attachment
Actions
Move
Copy
Subscribe
Archive
Share and more...
Steven Stretton
Temporarily moved the tests from the test directory to main, dom extraction works. Test structure will be amended later in a new task.
Jan 6 at 7:45 PM - Edit - Delete
Steven Stretton
Resolved the cyclic referencing issue by moving packages of configuration and driver into WeleniumLibrary (as a new package). And created a new configuration package in the UIBox module to configure the pageFactory upon execution.
Jan 4 at 5:07 PM - Edit - Delete
Steven Stretton
Issue with projects in the reactor contain a cyclic reference, as the WeleniumLibrary needs to have access to Selenium Webdriver to extract the content. Originally a new module was created but is now redundant therefore it is removed.
Jan 4 at 5:02 PM (edited) - Edit - Delete
Steven Stretton
DOM can be extracted through the firefoxdriver, however as the driver is called through the test, there are two solutions for this...
1. Move the firefox driver from the UIBox to Welenium
2. Create a new package for the driver and both UIBox and Welenium can both get its properties
Jan 2 at 1:45 PM - Edit - Delete
Steven Stretton moved this card from Analysis to Development
Dec 28, 2016 at 2:17 PM
Steven Stretton moved this card from Backlog to Analysis Dec 14, 2016 at 5:18 PM
Steven Stretton added this card to Backlog Dec 14, 2016 at 3:28 PM

AUT-6


[AUT-6] Identify DOM elements by value request
in list Done
Labels
Automation
Description Edit
System to find elements from the collected DOM using the webdriver
Attachments
Screen Shot 2017-02-07 at 12.23.45.png
Added Feb 7 at 12:24 PM
Download Make Cover Delete Comment
Add an attachment...
Add Comment
Write a comment...
Send
Activity
Hide Details
Steven Stretton moved this card from Develop Test to Done Feb 7 at 12:25 PM
Steven Stretton
Passed on develop.
Feb 7 at 12:25 PM - Edit - Delete
Steven Stretton moved this card from Feature Test to Develop Test
Feb 7 at 12:25 PM
Steven Stretton
Elements containing Id is being found and entered in the website. Screen Shot 2017-02-07 at 12.23.45.png
Add
Members
Labels
Checklist
Due Date
Attachment
Actions
Move
Copy
Subscribe
Archive
Share and more...


Steven Stretton moved this card from Development to Feature Test
Feb 7 at 12:21 PM
Steven Stretton
Found a issue with CSS selectors due to the complex structure, implementing this would take too long. So it was decided to use XPath as although user interaction isnt recommended, no user on Welenium will need to touch XPath
Feb 3 at 3:03 PM - Edit - Delete
Steven Stretton
Getting a InvocationTargetException on findContentHtml(), caused by a StringIndexOutOfBoundsException - this has been resolved
Jan 21 at 2:30 PM (edited) - Edit - Delete
Steven Stretton
ArrayIndexOutOfBoundsException is resolved by incrementing the new line count by 1
Jan 21 at 2:09 PM - Edit - Delete
Steven Stretton
Resolved the collection issue by removing a inner for loop that is only getting the end result of the DOM, this has fixed the first issue however encountered a ArrayIndexOutOfBoundsException caused by the line counter being equal to the size of the array.
Jan 19 at 6:12 PM - Edit - Delete
Steven Stretton
collection is only collecting </body> and </html> tags
Jan 19 at 3:47 PM - Edit - Delete
Steven Stretton moved this card from Analysis to Development
Jan 12 at 10:23 AM
Steven Stretton moved this card from Backlog to Analysis Jan 8 at 8:06 PM
Steven Stretton added this card to Backlog Dec 14, 2016 at 3:29 PM

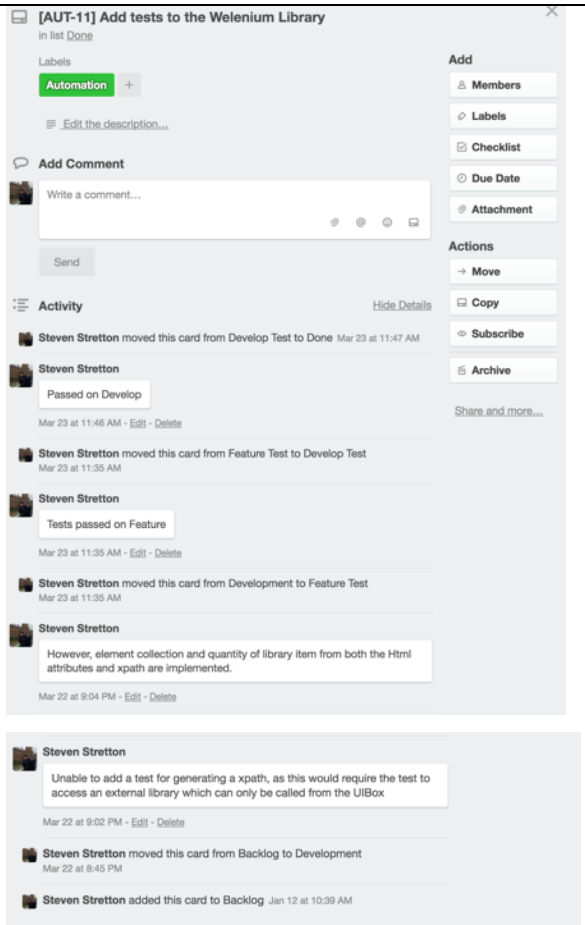
AUT-7	 [AUT-7] Collect additional elements in list Done Labels Automation Description Edit Make the system to support other web element tags. This will include: ~ name ~ placeholder Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Feb 24 at 10:12 PM Steven Stretton Passes on Develop Feb 24 at 10:12 PM - Edit - Delete Steven Stretton moved this card from Feature Test to Develop Test Feb 24 at 10:11 PM Steven Stretton Passes on Feature Feb 24 at 10:11 PM - Edit - Delete Steven Stretton moved this card from Development to Feature Test Feb 24 at 10:11 PM Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more... Steven Stretton Created a logger to output the generated paths onto terminal Feb 24 at 10:11 PM - Edit - Delete Steven Stretton XpathExpressionBuilder is not getting the updated tags from the new HtmlAttributesFinder, causing some tests to fail. Feb 7 at 8:38 PM - Edit - Delete Steven Stretton moved this card from Analysis to Development Feb 7 at 2:43 PM Steven Stretton Renamed Collect addtional elements for path from Generate path from found elements, as path generation was done in AUT-6 ticket Feb 7 at 12:27 PM - Edit - Delete Steven Stretton moved this card from Backlog to Analysis Feb 7 at 12:26 PM Steven Stretton added this card to Backlog Dec 14, 2016 at 4:55 PM
AUT-8	 [AUT-8] Create no element found exception in list Done Labels Automation Edit the description... Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Mar 3 at 4:37 PM Steven Stretton Merged and passed on develop Mar 3 at 4:37 PM - Edit - Delete Steven Stretton moved this card from Feature Test to Develop Test Mar 3 at 4:37 PM Steven Stretton Passed on Feature Mar 3 at 4:36 PM - Edit - Delete Steven Stretton moved this card from Development to Feature Test Mar 3 at 4:36 PM Steven Stretton In addition, I've created a Element missing from Xpath exception, to ensure that all Xpaths that are generated have the element name included. Mar 3 at 4:36 PM - Edit - Delete Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more... Steven Stretton Created no element found exception, this has been tested and passed Mar 3 at 4:35 PM - Edit - Delete Steven Stretton moved this card from Analysis to Development Mar 3 at 3:12 PM Steven Stretton moved this card from Backlog to Analysis Mar 3 at 3:12 PM Steven Stretton added this card to Backlog Dec 14, 2016 at 4:56 PM

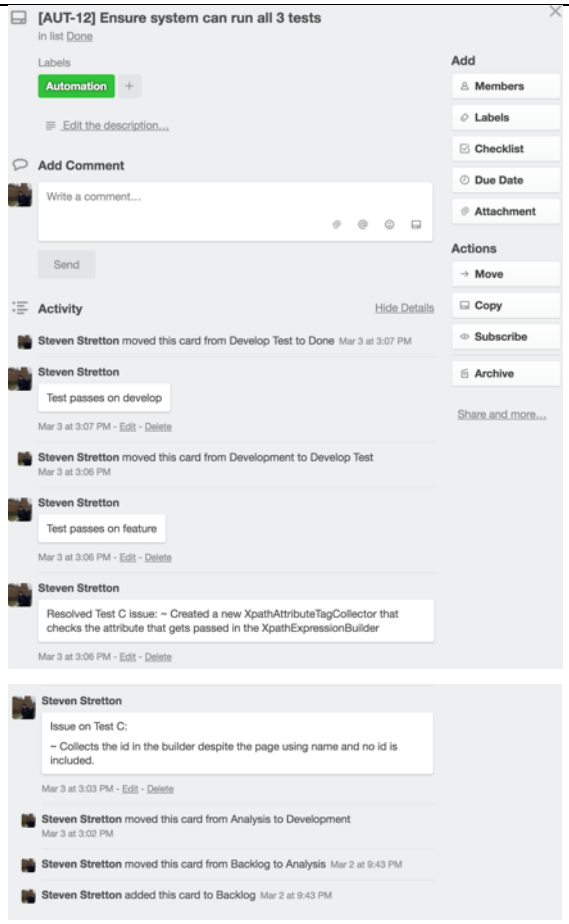
AUT-9	 [AUT-9] Code cleanup in list Done Labels Automation Description Edit Clean up the system, there may be redundant code being used. Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Mar 23 at 9:38 PM Steven Stretton Passed on develop Mar 23 at 9:38 PM - Edit - Delete Steven Stretton moved this card from Feature Test to Develop Test Mar 23 at 9:38 PM Steven Stretton Passed on feature, ready to merge to develop Mar 23 at 2:32 PM - Edit - Delete Steven Stretton moved this card from Development to Feature Test Mar 23 at 2:31 PM Steven Stretton Reformatted java classes and removed commented out and redundant code Mar 23 at 2:29 PM - Edit - Delete Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more... Steven Stretton Reformatted java classes and removed commented out and redundant code Mar 23 at 2:29 PM - Edit - Delete Steven Stretton moved this card from Analysis to Development Mar 23 at 12:10 PM Steven Stretton Some text may need removing. Mar 23 at 12:09 PM - Edit - Delete Steven Stretton moved this card from Backlog to Analysis Mar 23 at 12:07 PM Steven Stretton added this card to Backlog Dec 14, 2016 at 4:56 PM

AUT-10

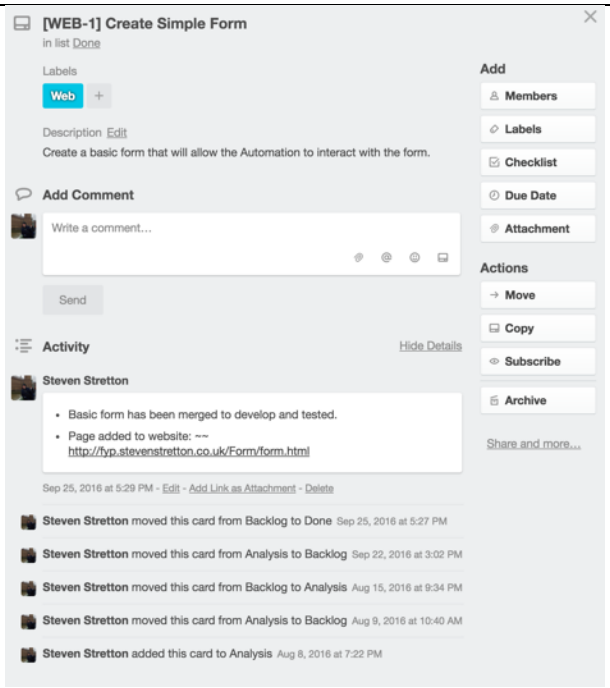

[AUT-10] Divide both Selenium and Welenium Tests
in list Done
Labels
Automation
Description Edit
Divide both the tests (pageObjects) so a comparison can be shown, it is expected that selenium will fail some tests whereas Welenium will not.
Add Comment
Write a comment...
Send
Activity
Hide Details
Steven Stretton moved this card from Develop Test to Done Mar 2 at 5:10 PM
Steven Stretton
Successfully merged and passed on develop
Mar 2 at 5:10 PM - Edit - Delete
Steven Stretton moved this card from Feature Test to Develop Test
Mar 2 at 5:09 PM
Steven Stretton
Passes on feature
Mar 2 at 4:13 PM - Edit - Delete
Steven Stretton moved this card from Development to Feature Test
Mar 2 at 4:13 PM
Add
Members
Labels
Checklist
Due Date
Attachment
Actions
Move
Copy
Subscribe
Archive
Share and more...
Steven Stretton
Test B of selenium does locate the text boxes and checkbox, however it fails to find the button. However Welenium does locate the button successfully.
Mar 2 at 4:12 PM - Edit - Delete
Steven Stretton
Selenium and Welenium tests divided successfully.
Mar 2 at 4:11 PM - Edit - Delete
Steven Stretton moved this card from Analysis to Development
Mar 2 at 4:11 PM
Steven Stretton moved this card from Backlog to Analysis Feb 28 at 12:57 PM
Steven Stretton added this card to Backlog Jan 12 at 10:21 AM

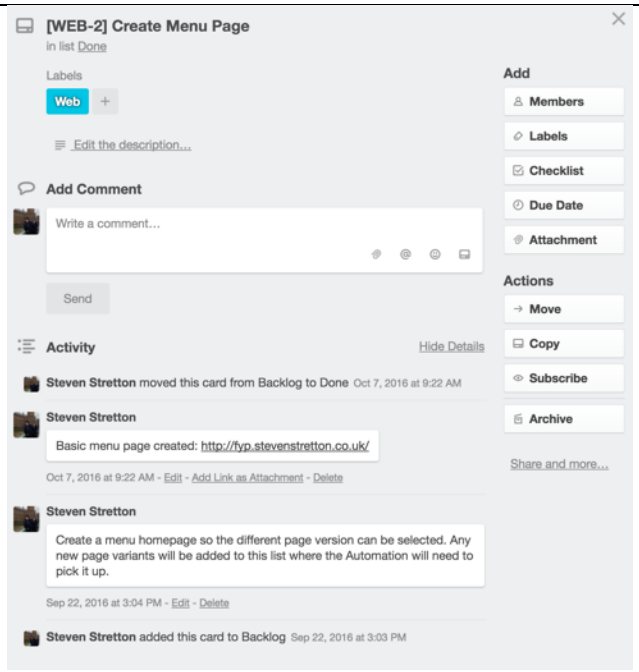
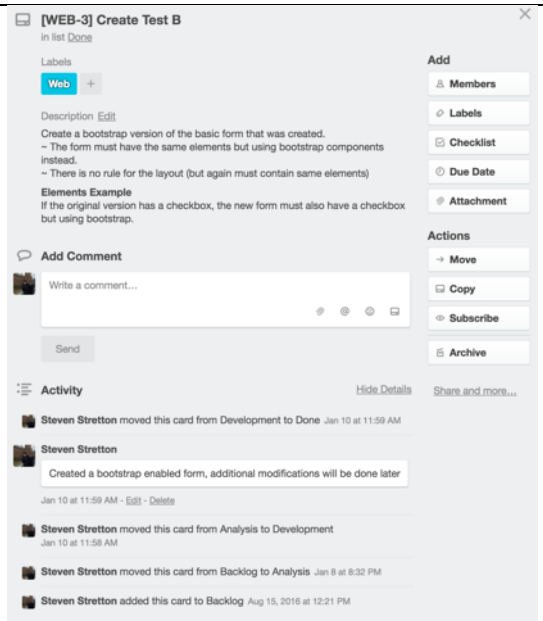
AUT-11

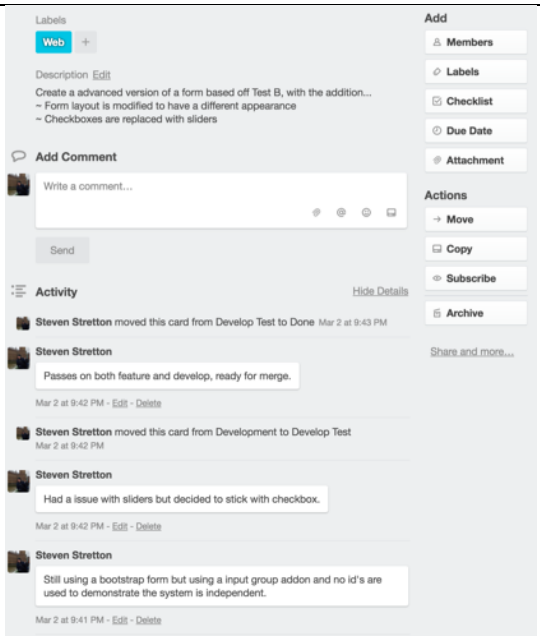

[AUT-11] Add tests to the Welenium Library
in list Done
Labels
Automation
Edit the description...
Add Comment
Write a comment...
Send
Activity
Hide Details
Steven Stretton moved this card from Develop Test to Done Mar 23 at 11:47 AM
Steven Stretton
Passed on Develop
Mar 23 at 11:46 AM - Edit - Delete
Steven Stretton moved this card from Feature Test to Develop Test
Mar 23 at 11:35 AM
Steven Stretton
Tests passed on Feature
Mar 23 at 11:35 AM - Edit - Delete
Steven Stretton moved this card from Development to Feature Test
Mar 23 at 11:35 AM
Steven Stretton
However, element collection and quantity of library item from both the Html attributes and xpath are implemented.
Mar 22 at 9:04 PM - Edit - Delete
Add
Members
Labels
Checklist
Due Date
Attachment
Actions
Move
Copy
Subscribe
Archive
Share and more...
Steven Stretton
Unable to add a test for generating a xpath, as this would require the test to access an external library which can only be called from the UIBox
Mar 22 at 9:02 PM - Edit - Delete
Steven Stretton moved this card from Backlog to Development
Mar 22 at 8:45 PM
Steven Stretton added this card to Backlog Jan 12 at 10:39 AM

AUT-12	

Section 2: Web Tickets

WEB-1	

WEB-2	 [WEB-2] Create Menu Page in list Done Labels Web Edit the description... Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Backlog to Done Oct 7, 2016 at 9:22 AM Steven Stretton Basic menu page created: http://fyp.stevenstretton.co.uk/ Oct 7, 2016 at 9:22 AM - Edit - Add Link as Attachment - Delete Steven Stretton Create a menu homepage so the different page version can be selected. Any new page variants will be added to this list where the Automation will need to pick it up. Sep 22, 2016 at 3:04 PM - Edit - Delete Steven Stretton added this card to Backlog Sep 22, 2016 at 3:03 PM Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more...
WEB-3	 [WEB-3] Create Test B in list Done Labels Web Description Edit Create a bootstrap version of the basic form that was created. ~ The form must have the same elements but using bootstrap components instead. ~ There is no rule for the layout (but again must contain same elements) Elements Example If the original version has a checkbox, the new form must also have a checkbox but using bootstrap. Add Comment Write a comment... Send Activity Hide Details Share and more... Steven Stretton moved this card from Development to Done Jan 10 at 11:59 AM Steven Stretton Created a bootstrap enabled form, additional modifications will be done later Jan 10 at 11:59 AM - Edit - Delete Steven Stretton moved this card from Analysis to Development Jan 10 at 11:58 AM Steven Stretton moved this card from Backlog to Analysis Jan 8 at 8:32 PM Steven Stretton added this card to Backlog Aug 15, 2016 at 12:21 PM Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive

WEB-4	 Labels Web Description Edit Create a advanced version of a form based off Test B, with the addition... ~ Form layout is modified to have a different appearance ~ Checkboxes are replaced with sliders Add Comment Write a comment... Send Activity Hide Details Steven Stretton moved this card from Develop Test to Done Mar 2 at 9:43 PM Steven Stretton Passes on both feature and develop, ready for merge. Mar 2 at 9:42 PM - Edit - Delete Steven Stretton moved this card from Development to Develop Test Mar 2 at 9:42 PM Steven Stretton Had a issue with sliders but decided to stick with checkbox. Mar 2 at 9:42 PM - Edit - Delete Steven Stretton Still using a bootstrap form but using a input group addon and no id's are used to demonstrate the system is independent. Mar 2 at 9:41 PM - Edit - Delete Add Members Labels Checklist Due Date Attachment Actions Move Copy Subscribe Archive Share and more...	