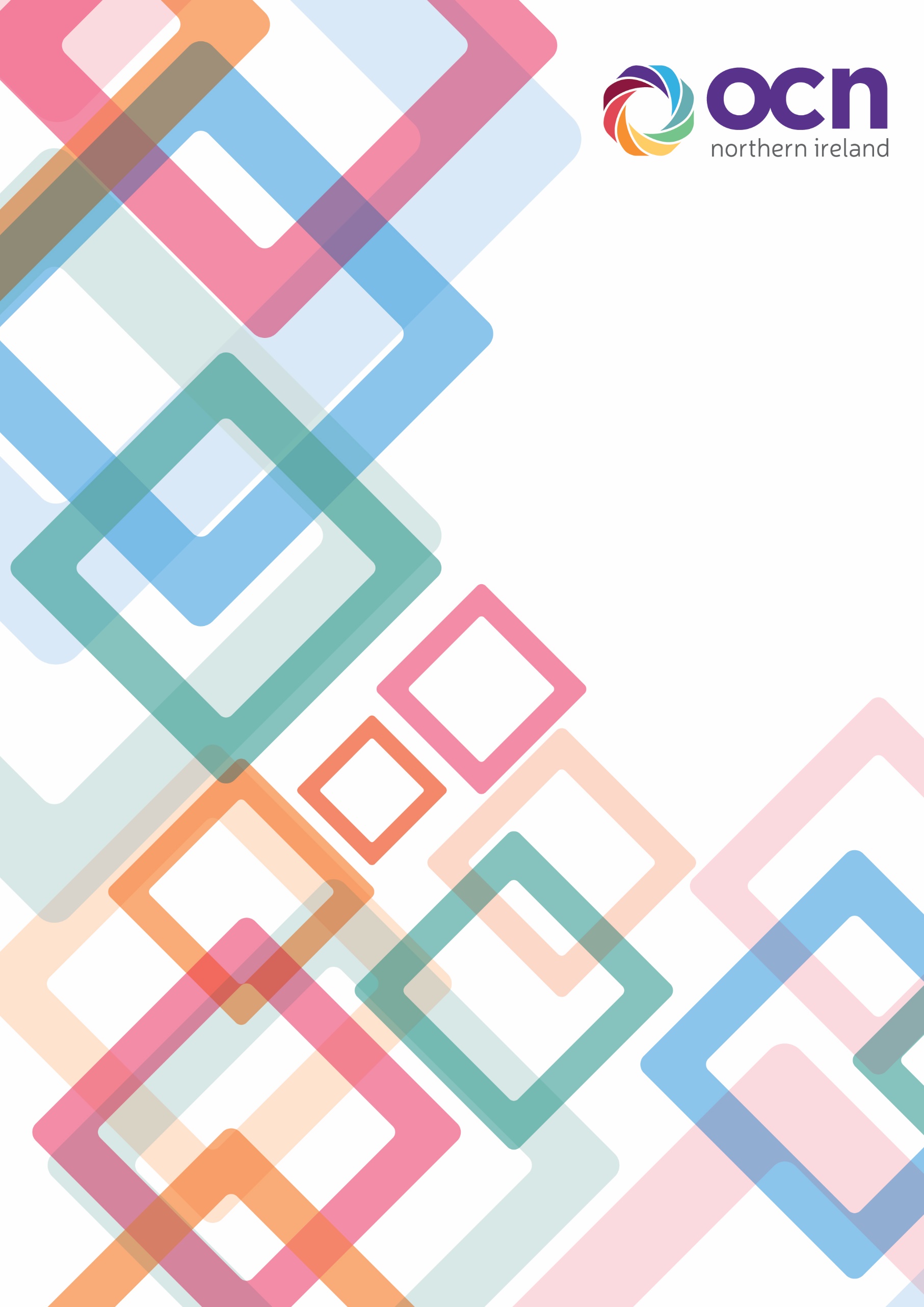
****

**Automated Software Testing Fundamentals**

**(L/617/8670)**

**Learner Assessment  
Booklet**

**Learner Assessment Booklet**

**Level 4 Diploma in Software Testing**

**Unit:** Automated Software Testing Fundamentals

**Credit Value:** 16

**Unit Reference Number:** (L/617/8670)

***This form is used to record and confirm that assessment has taken place and the learner has achieved the unit detailed on pages 5 & 6***

|  |  |
| --- | --- |
| **Learner Name:** | Ryan McKee |
| **Assessor name:** |  |
| **Internal Verifier name:** |  |
| **Course:** |  |
| **Course code:** |  |
| **Completion Date:** |  |

|  |
| --- |
| **Achievement of unit is confirmed: Yes/No**  **Signature of Assessor:**  **Date:** |

|  |
| --- |
| **I certify that all the work in this booklet is my own.**  **Learner Signature:**  **Date:** |

**Assessment Grid: *Completed by Assessor***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Learning Outcomes** | **Achieved**  **(Delete)** | **Criteria** | **Booklet Page** | **Assessment Date**  **(If Applicable)** | **Date of Verification**  **(If Applicable)** |
| **1** | Yes | 1.1  1.2 |  |  |  |
| **2** | Yes/No | 2.1  2.2  2.3 |  |  |  |
| **3** | Yes/No | 3.1  3.2  3.3  3.4  3.5 |  |  |  |
| **4** | Yes/No | 4.1  4.2  4.3 |  |  |  |
| **5** | Yes/No | 5.1  5.2  5.3 |  |  |  |
| **6** | Yes | 6.1  6.2 |  |  |  |
| **7** | Yes/No | 7.1  7.2 |  |  |  |

***Note to Assessors:***

***This assessment record book is a template that you may choose to use to record summative assessment activities. It is intended to offer a framework for recording summative assessment. It is not intended to be prescriptive about assessment tasks. Assessors should amend the assessments and the booklet accordingly, taking account of the needs of their specific learner group and any opportunities for using combined assessments that cover multiple assessment criteria or learning outcomes.***

**Assessment record: *Completed by Assessor***

|  |
| --- |
| **Assessor Assessment decisions:** |

**Signature of Assessor:**

**Date:**

**All the following Learning Outcomes and Assessment Criteria MUST be met:**

**This unit has seven learning outcomes**

|  |  |
| --- | --- |
| **Learning Outcomes** | **Assessment Criteria** |
| **The learner will:** | **The learner can:** |
| 1. Understand what is meant by automated testing and its application. | * 1. Explain what is meant by test automation and the main two testing disciplines   2. Analyse at least three different features of web applications for which automated testing would be the optimal approach. |
| 1. Be able to research test automation practices in order to develop a software testing plan. | * 1. Research the advantages and disadvantages for test automation practices and their associated resources.   2. Critically compare the application of manual and automated software testing for a given application. – given application would be an api talk about some of the things that would need manually tested that api tests could not or selenium tests could now.   3. Develop an optimal software testing plan – need to show test plan for api test |
| 1. Be able to set up and use a web-based automated software testing tool. | * 1. Explain the setup procedures for using a web-based automated software-testing tool. – do this   2. Explain Hyper Text Transfer Protocol (HTTP) and its ability to retrieve appropriate resources.   3. Explain and demonstrate the use of at least four methods to assist the testing process available within a given testing tool. POST, PUT UPDATE AND DELETE screenshot the code for each of these and explain a bit about them.   4. Demonstrate the use of conditions to check for validity of tests   5. Demonstrate the effective use of a wait to assist with confirmation that a specific testing task has been successful. |

|  |  |
| --- | --- |
| **Learning Outcomes** | **Assessment Criteria** |
| **The learner will:** | **The learner can:** |
| 1. Be able to locate and test web elements. | * 1. Explain and demonstrate at least three different techniques to reference web elements on a webpage. – needs work got screen shots and stuff now show how I got it on the browser.   2. Explain what is meant by the Document Object Model (DOM) and how it allows a Hypertext Markup Language (HTML) webpage to be accessed.   3. Demonstrate the use of browser developer tools for testing. – needs to be done. |
| 1. Be able to manage and interact with notifications within a browser. | * 1. Critically evaluate different types of notifications and responses received from interactions within a browser. – responses are response codes, 200 working, 500, 400 so on. Evaluate the response code for something like a delete code where even tho 200 shows that the api works as expected but in the body it returns false telling the test an item was not actually deleted.   2. Summarise common exceptions and codes generated through different test actions. – exceptions that cause the exception codes shown above. Examples could be a connect exception thrown when a test is run.   3. Use an appropriate test strategy to effectively manage and interact with a new tab or window. – selenium pop up tests notes in email |
| 1. Be able to research and create a test automation suite.   “ A test suite is just several different classes testing classes within a software solution “ | * 1. Research and evaluate the advantages and disadvantages of implementing a test suite.   2. Create a test automation suite for use with multiple test cases for test execution purposes. |
| 1. Be able to research and create a test automation framework. | * 1. Research and evaluate the advantages and disadvantages of using a test automation framework. – can do this as its just theory. 6.1 and 7.1 are sort of interlinked   2. Create a basic test automation framework for a given application that features reusability of code and low-cost maintenance. – can’t look at this currently complete after easter. |

**ASSESSMENT CRITERIA (AC)**

**AC 1.1:** Explain what is meant by test automation and the main two testing disciplines.

Test automation refers to the use of software tools to execute tests automatically, without requiring manual intervention. Test automation is particularly useful for repetitive tests, where manual testing would be time-consuming and error prone. There are two main disciplines withing test automation the first being functional testing which involves testing whether the system or software application is functioning as expected, by verifying its behaviour against a set of predefined requirements. Functional testing includes testing individual functions or features of the system or software application, as well as testing the overall behaviour of the system examples of functional tests would be smoke testing, regression testing, integration testing unit testing, alpha and beta testing and user acceptance testing just to name a few.

The second method of automation testing is non-functional testing this is when non-functional aspects of the system or software application, such as performance, reliability, scalability and security. Are tested. Non-functional testing focuses on the systems’ ability to meet performance requirements, handle varying loads, and function effectively and securely in different environments. Some examples of non-functional tests include performance tests, load tests stress tests and security tests.

Overall test automation can be applied to both functional and non-functional testing disciplines. Automating tests can significantly reduce the time and effort required to perform tests, while also improved their accuracy and repeatability.

**AC 1.2:** Analyse at least three different features of web applications for which automated testing would be the optimal approach.

Automated testing has become an indispensable practice in modern web application development as it helps to ensure that software meets its requirements and operates as expected. There are several features of web applications where automated testing is optimal, including UI testing, API testing, and performance testing.

Automated UI testing is an optimal approach for web applications that have a large user interface, such as e-commerce sites, content management systems, and social media platforms. The user interface is a critical aspect of web applications as it's the primary way users interact with them. Automated UI testing frameworks, such as Selenium, can simulate user interactions and verify the expected output against the actual output. This can help to ensure that the application's interface is functioning as expected and is responsive to user interactions.

Web applications are often built using APIs, which expose data and functionality to other applications. Automated API testing is optimal for applications that rely heavily on APIs, such as those with complex data structures, multiple endpoints, or those that integrate with third-party APIs. Automated API testing can verify that the API endpoints return the expected response and that the response is correctly formatted. It can also ensure that the API performs as expected under varying loads and network conditions. Tools such as Postman or Newman can be used to automate API testing.

Web applications must perform well to provide a satisfactory user experience. Automated performance testing is optimal for web applications that have high traffic and a large user base, such as e-commerce sites, social media platforms, or video streaming services. Automated performance testing can help to ensure that the application can handle the expected load and is responsive under varying traffic and network conditions. Tools such as JMeter can be used to simulate traffic and measure performance metrics like response times, throughput, and error rates.

In conclusion, automated testing is essential for ensuring the quality and performance of web applications. UI testing, API testing, and performance testing are three critical areas where automated testing can be an optimal approach to achieving quality and reliability in web application development. By incorporating automated testing into the development process, web developers can minimize the risk of errors and ensure that their applications meet user requirements and provide a satisfactory user experience.

**Section 2**

**AC 2.1:** Research the advantages and disadvantages for test automation practices and their associated resources

Test automation is the use of software tools to perform testing tasks automatically, without requiring human intervention. The advantages of test automation include faster and more efficient testing, increased test coverage, and the ability to run tests repeatedly with little or no additional cost. Additionally, test automation can detect defects earlier in the development process, reducing the overall cost of fixing defects and enhancing the quality of the software product.

On the other hand, test automation also has some disadvantages. Initial setup and configuration of the automation tools can be time-consuming and costly. Maintenance of test scripts and automation frameworks can also be challenging and require continuous effort. Test automation cannot completely replace manual testing, as some aspects of testing still require human intelligence and creativity. Finally, the return on investment for test automation needs to be carefully evaluated to ensure that the benefits of automation outweigh the costs.

In terms of specific tools, TestNG is a testing framework that provides features such as grouping, sequencing, and parallel execution of test cases. Rest Assured is a Java library used to test RESTful web services. Selenium is an open-source tool used to automate web browsers for testing web applications. Each tool has its own advantages and disadvantages, and the selection of a particular tool depends on the specific requirements of the project and the expertise of the testing team.

Overall, test automation can be a valuable addition to a testing process, but it should not be viewed as a complete replacement for manual testing. It is important to carefully evaluate the benefits and costs of automation, select appropriate automation tools, and ensure that the testing team has the necessary skills and expertise to effectively use the automation tools.

**AC 2.2:** Critically compare the application of manual and automated software testing for a given application

The Staff API is a REST API application that includes endpoints for push, put, delete, and get operations. In order to ensure its reliability and scalability, testing is essential. Manual testing and automated testing are two approaches that can be used to test the Staff API, each with its own advantages and disadvantages.

Manual testing involves a tester manually interacting with the application by sending requests to the API endpoints and checking the responses. This approach offers more flexibility and adaptability in testing, as testers can easily change test cases and provide a more subjective evaluation of the application. However, manual testing is time-consuming, error-prone, and expensive since it requires human effort.

On the other hand, automated testing involves using tools to automate the testing process. Tools like Postman, Rest Assured, and TestNG can be used to create and execute automated tests. Automated testing is faster, more reliable, and less error-prone than manual testing. Additionally, it is useful for repetitive testing tasks and can easily be integrated with Continuous Integration and Continuous Delivery pipelines, making code maintainable and expandable. However, automated testing requires a higher initial investment in terms of setup and tooling, and can only be as effective as the test cases it is built upon.

Both manual and automated testing can be useful for different purposes in testing the Staff API. Manual testing is particularly useful for exploratory testing, usability testing, and testing of edge cases. On the other hand, automated testing is particularly useful for regression testing, where the same test cases are repeatedly executed to ensure that changes made to the API did not introduce new errors or regressions. It is also useful for load testing, where multiple requests are sent simultaneously to test the performance and scalability of the API.

Both manual and automated testing can be applied to specific endpoints. For instance, automated tests can be created for the "get-all-staff" endpoint to verify that the list contains the expected number of staff members. Similarly, for the "add-new-staff-member" or "update-staff" endpoints that add or update staff information, automated tests can be created to verify that the data has been updated as expected. For the "delete-staff-member" endpoint, a test can be written to verify that the management record number has been reduced by 1 and that the correct staff member was deleted.

In conclusion, both manual and automated testing have their own advantages and disadvantages, and they can be applied to all the endpoints within the staff-API system. However, based on the current needs of the system, automated testing is the best approach to ensuring reliability and scalability into the future. This is because there are no complex endpoints that require user authentication, involve complex business logic or interact with external systems of dependencies. Automated testing is faster, more reliable, and less error-prone, making it the ideal approach for testing the Staff API.

**Table

Description automatically generatedAC 2.3:** Develop an optimal software testing plan

**Table

Description automatically generated**

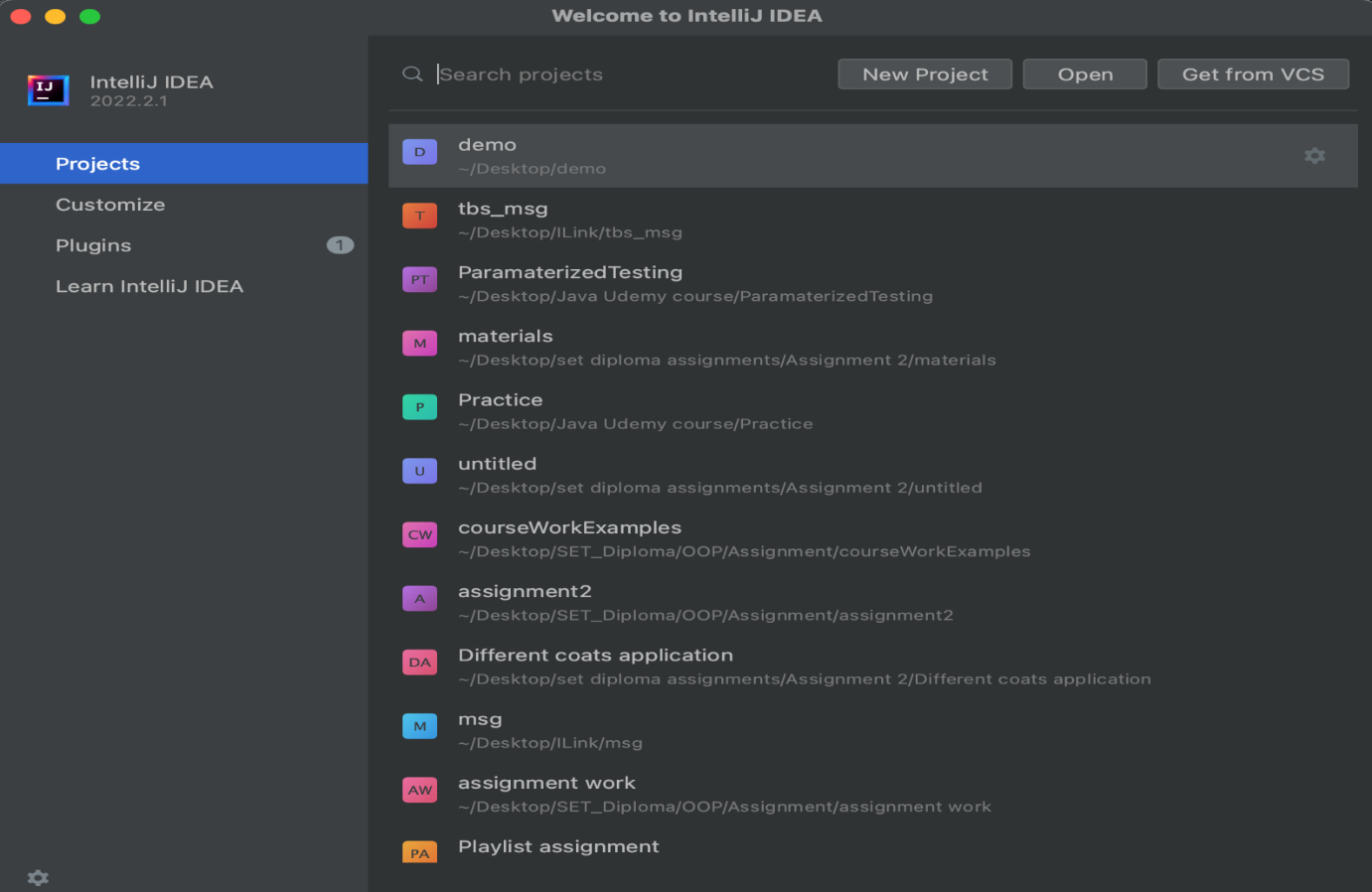
**Table

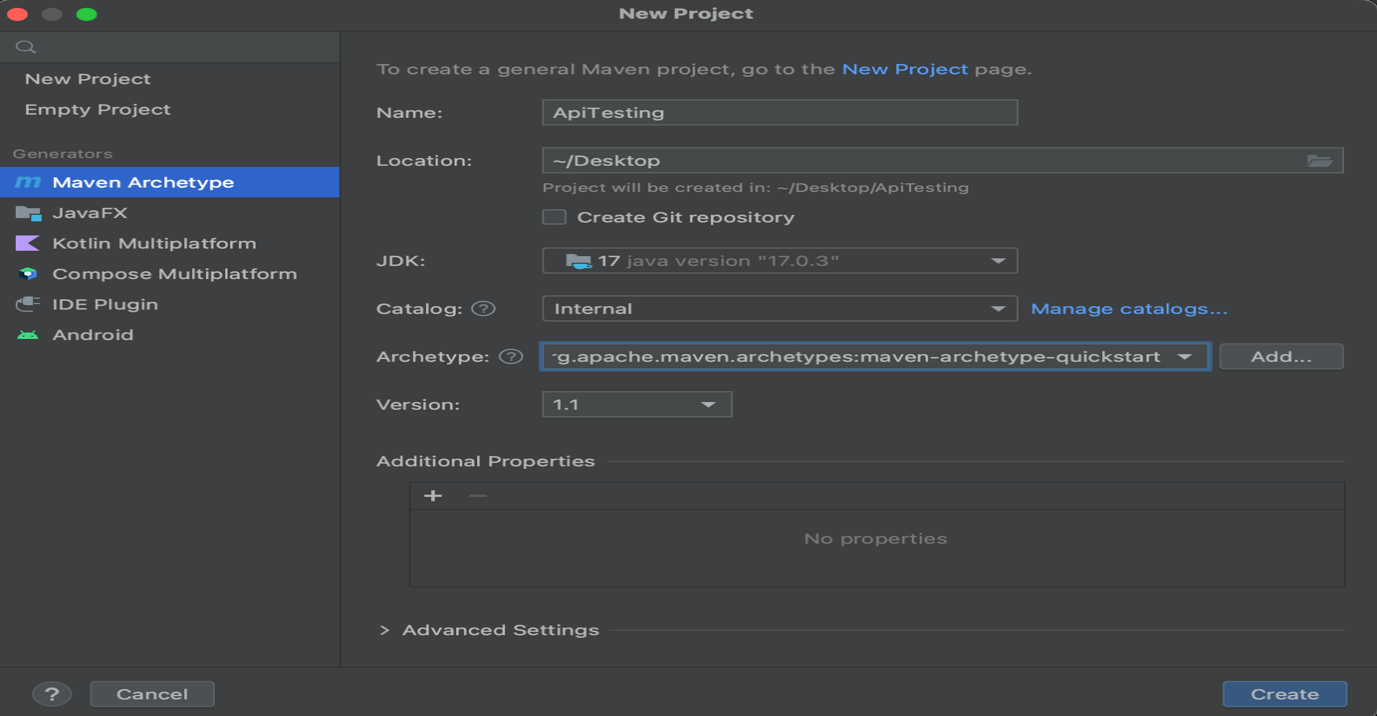
Description automatically generated**

**Section 3**

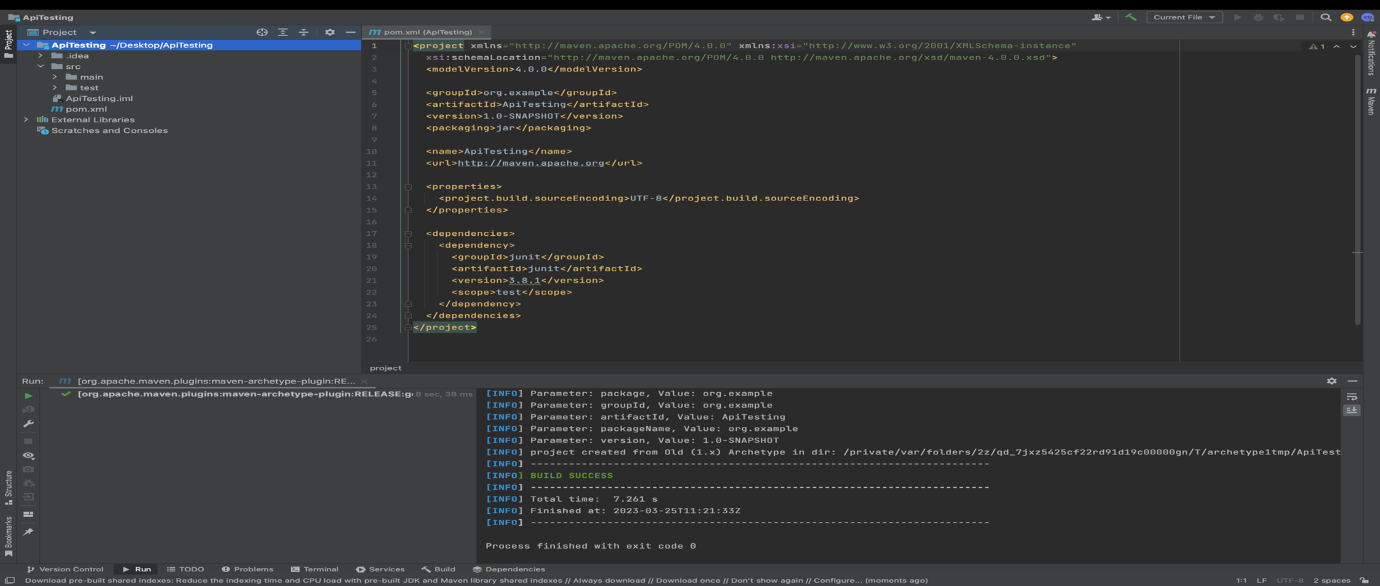
**AC 3.1:** Explain the setup procedures for using a web-based automated software-testing tool

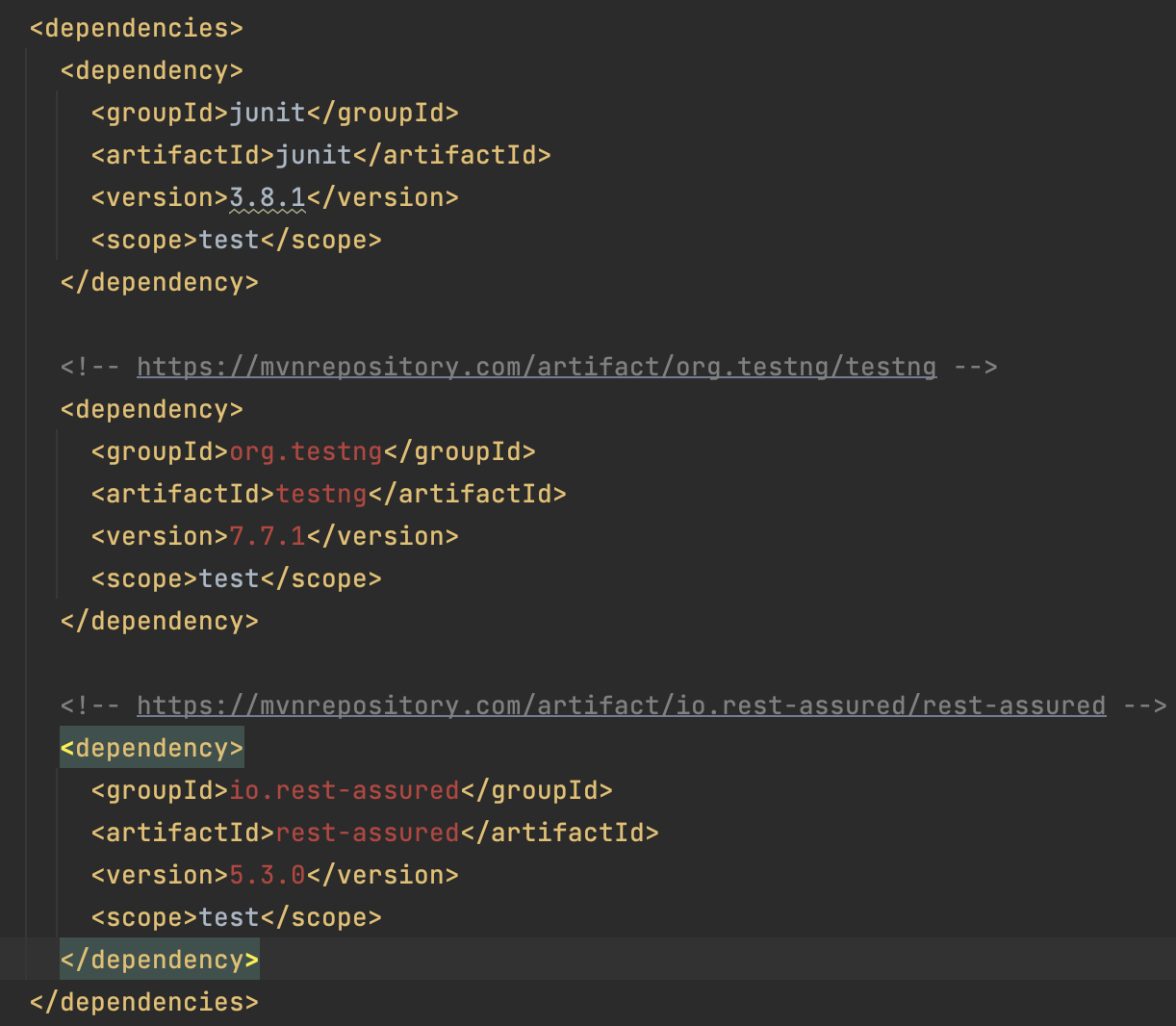
This step involves creating a new Maven project, which is relatively simple to accomplish in most IDEs. A Maven project provides a structured and standardized approach to software development and is widely used in Java-based web applications.



Input the name of your testing application, select the JDK version, and choose the appropriate Archetype from the available templates. In this step, you will specify the name of your testing application and select the version of the Java Development Kit (JDK) you wish to use. Additionally, you will choose an Archetype, which is a collection of pre-defined templates that provide a starting point for your project. This selection determines the structure and dependencies of your project. Ensure that you choose the appropriate Archetype that suits your project requirements.

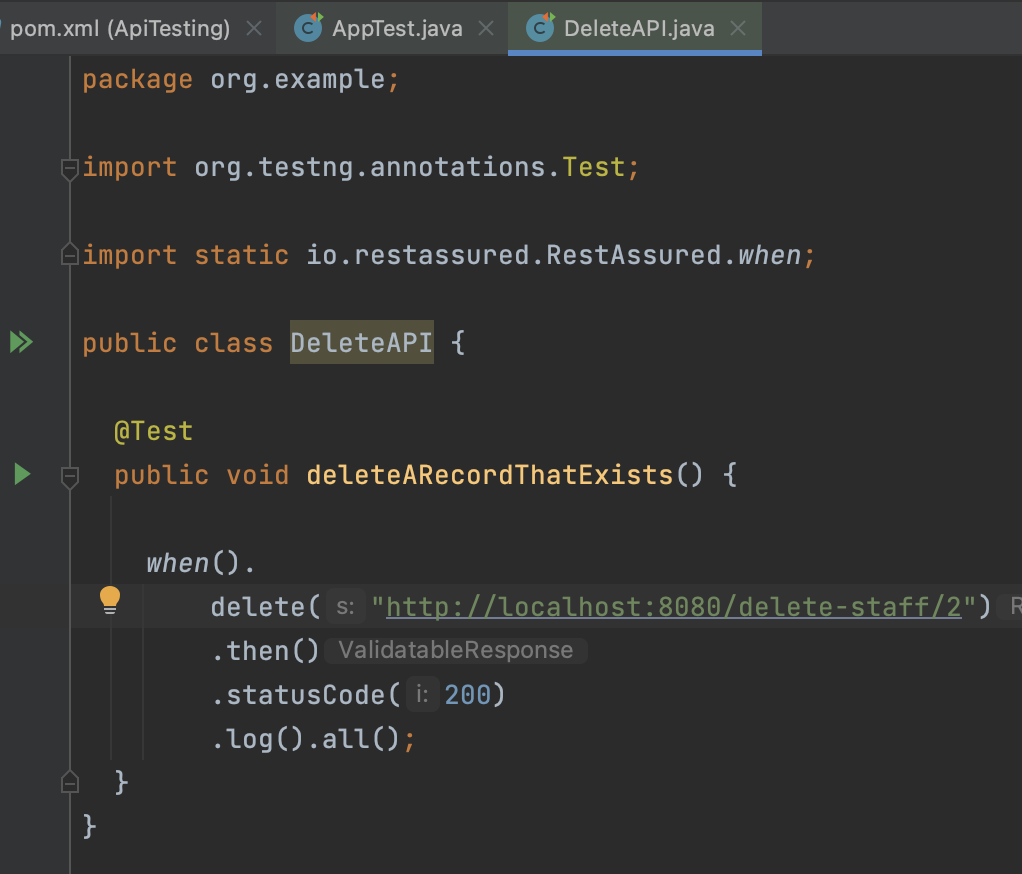
Once you have completed Step 2, your project will open up, and Maven will automatically begin generating the project based on the templates defined in the pom.xml file.

At this point, the project structure and dependencies are defined, and Maven is configuring the project based on the selected Archetype. The pom.xml file is a configuration file that defines the project's build process, dependencies, and other details. Maven uses this file to manage the project's build and deployment process. With the project generation underway, proceed to the next step.

In this step, you will add import statements for the libraries required by your test automation project to the dependencies section of the pom.xml file. These libraries include Rest Assured and TestNG, which are essential for implementing automated tests in web applications. By adding these libraries to the project, you can leverage their functionality in your test scripts. Ensure that the import statements are correctly formatted and located within the dependencies section of the pom.xml file.

**Graphical user interface, text

Description automatically generated**Click install and maven will install these new project dependencies.

****With the project dependencies in place, you can now create your API test. Below is an example of a Rest Assured test that checks the response when a delete function occurs on the localhost.

**AC 3.2:** Explain Hyper Text Transfer Protocol (HTTP) and its ability to retrieve appropriate resources.

Hyper test transfer protocol (HTTP) is a protocol used for communication between web servers and clients, such as web browsers. The HTTP protocol governs how requests from clients are made to the server, and how the server responds to those requests with the appropriate resources like HTML pages, images, documents videos and audio files and web services.

HTTP uses uniform resource locators (URLs) to identify resources on the web. A URL is a unique address that identifies a resource such as a web page, image or document. The URL consists of a protocol (http:// or https://), a domain name example.com which will translate to the physical address e.g. 198.102.434.8 and a path to the resource on the server e.g. ( /documents/foo )

HTTP requests can be made using several different methods. The most common HTTP methods are GET, POST, PUT or DELETE.

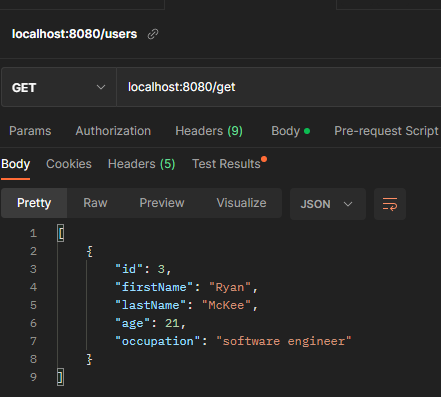
**Get:** This method is used to retrieve a resource from the server. When a client sends a GET request to a server, the server response by sending the requested resource back to the client. GET requests can be cached by clients and servers, making them useful for retrieving resources that do not change frequently.

Figure : The image on the left shows an example of a get request for a localhost server which gets a staffs details. The resource is returned in JSON format.

**POST:** This method is used to submit data to the server. When a client sends a POST request to a server, it typically includes data in the request body that the server can use to perform some action, such as updating a database. Post requests are not cached by clients of servers, making them useful for submitting data that should not be cached.

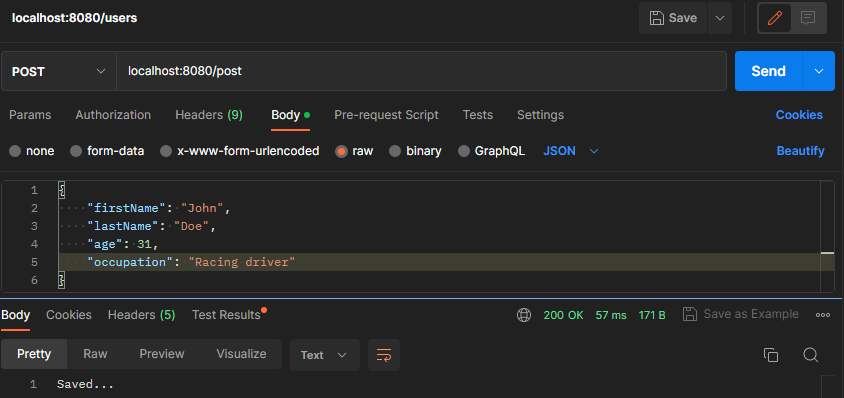
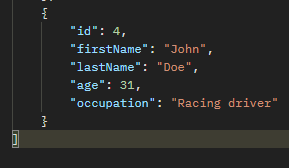
****

Figure : Above shows an example of a POST request. When endpoint /post is used the information in json format in the request body is sent to the server this is confirmed by the response body at the bottom returning the string saved and the status code 200 on the bottom right in green.

PUT: This method is used to update an existing resource on the server. When a client sends a PUT request to a server, it typically includes the updated data in the request body, and the server updates the resource with the new data. PUT requests can be used to create new resources as well, if the server supports this.

Above shows the record with id 4 before the PUT operation was performed.

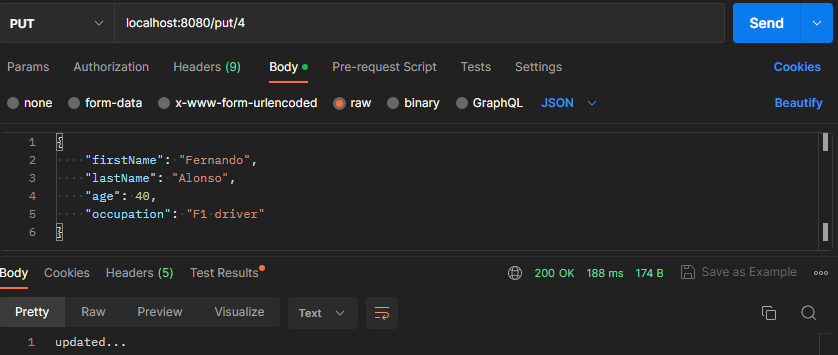
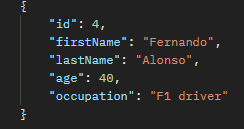
****

Figure : Above shows and example of the end point put which also takes the id of the record that is being updated. In this example the record with id 4 is updated to contain the information sent in the message body. Below shows the output of that PUT request which was received from the response body.

****

**DELETE:** This method is used to delete a resource from the server. When a client sends a DELETE request to a server, the server deletes the resource and responds with a confirmation message. DELETE requests can be used to remove resources that are no longer needed.

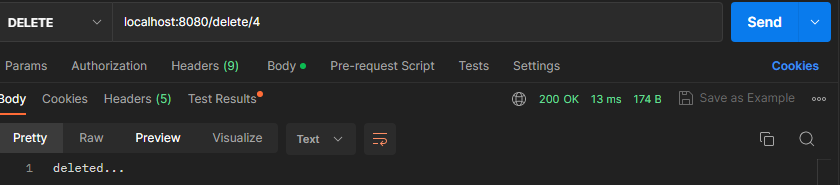
****

Figure : Above shows an example of the delete end point it takes a path variable and as long as there is a record in the database with this id the record will be deleted as shown in the example above.

In summary, HTTP is a protocol that allows clients and servers to communicate and exchange resources. HTTP request can be made using different methods (verbs) such as GET, POST, PUT, and DELETE, each serving a different purpose in retrieving and manipulating resources on the server.

**AC 3.3:** Explain and demonstrate the use of at least four methods to assist the testing process available within a given testing tool.

When testing an API, it is crucial to create a test suite that covers all its functionalities. We can automate these tests using TestNG as the testing library and Rest Assured, which provides a DSL for testing RESTful APIs like the one in the example. Rest Assured allows for tests to be written in a simple and readable manner, making it easier to validate the API's functionality. It is a Java library that offers a DSL specifically for testing RESTful APIs.

**GET test**

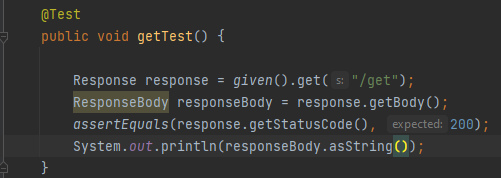
****

Figure : Above is a GET test that checks if the /get endpoint returns a status code of 200, indicating that the request was successfully received, understood, and accepted by the server. This means that the server has processed the request and is returning the requested information to the client in the message body of the response. To ensure that the response body is not null, the test prints it to the console manually.

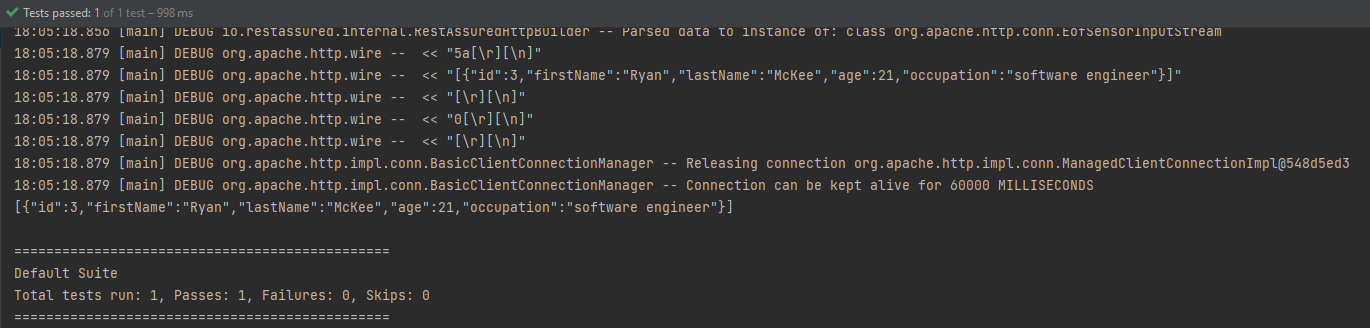
****

Figure 6: Above is the console output after running the getTest. It displays the returned response body and confirms that the test has passed.

****

Figure : Above is another example of a GET request, this time using a different endpoint that takes a specific ID of a staff member (in this case, number 3). The test checks if the response body contains "Ryan" (to confirm that this is the expected staff member) and if the status code is 200, indicating a successful response.

**PUT test**

**Text

Description automatically generated**

Figure : This is an example of a PUT test function. It checks if sending employee data in JSON format via the request body to the /update-staff/4 endpoint results in a successful request with a status code of 200.

**POST test**

**Text

Description automatically generated**

Figure : This is an example of a POST REST test function. The function checks if, when using the /add-new-staff-member endpoint and passing an employee's details in JSON format in the request body, the response code is 200 and the size of the employee list has incremented by 1.

**Text

Description automatically generatedText

Description automatically generatedDELETE TEST**

Figure : This code contains a unit test for the delete-member-endpoint. The test will fail if there are no records in the database, as it requires at least one record to be present. The test retrieves the ID of the last record in the database using a GET request, and then calls the delebeByIdAndGetResponse function to delete the record using this ID. It checks the response code to ensure that it is 200, indicating that the delete was successful, and also verifies that the database size has decreased by 1.

Figure : This function is used in the test function shown in Figure 7. It takes an ID as an argument and uses a REST Assured DELETE statement to call the “/delete-staff-member/” endpoint, with the ID given as a path parameter.

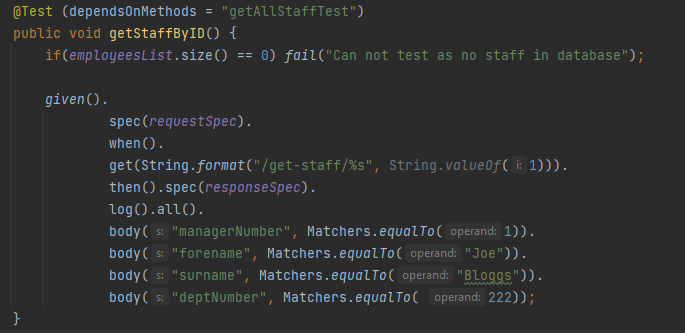
**AC 3.4:** Demonstrate the use of conditions to check for validity of tests – use ham-crest matchers for the conditions

Conditionals provide a flexible and expressive approach to assert statements in automated tests. The Ham crest framework offers an extensive range of matchers that can be used in conjunction with unit testing libraries, such as TestNG, and mocking frameworks, such as Mockito. These matchers provide a powerful and intuitive way of writing assertions that are both expressive and concise, enabling developers to understand the test's objectives better.

Using Ham crest matchers has several benefits, one of which is the ability to create more robust tests that are less vulnerable to codebase changes. The matchers are designed to be highly reusable, allowing developers to test various scenarios and conditions while maintaining a test suite over time**.**

**examples:**

Figure : Above screenshot shows an example of matchers being used to assert that each of the response body values are equal to what is expected from the path parameter used: 1.

**Matcher equal To**

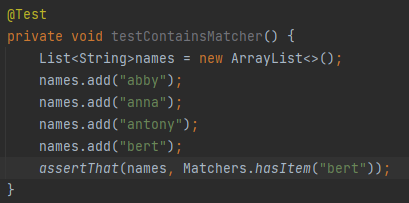
**Matcher has Item**

Figure : Above is an example of the matcher has Item being used to assert that Bert exists in a list of names. Since it does this test will return true.

**Matcher contains string**

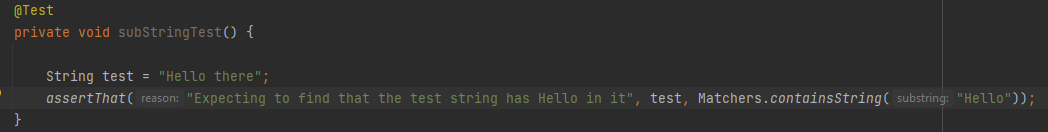
****

Figure : This shows an example of the Matcher.containsString in action. In this sample test it is checking if Hello is contained in Hello there which it is therefore making the test return true.

**Asserting that a value is greater than a given value.**

****

Figure : Line 63 on this code shows another example of a greater Than matcher. In this test this matcher is used to ascertain if the current records after an object have been posted to the spring API has increased by 1.

**AC 3.5 + 3.4:** Demonstrate the effective use of a wait to assist with confirmation that a specific testing task has been successful.

**The example below shows an example of a wait to assist. This wait to assist is a response time test this is used to measure the time taken by the spring boot API endpoints to respond to a request. This helps in assessing the performance of the API and identifying any potential bottlenecks or issues in the system. By monitoring the response time in these tests, the API can be assured to meet the performance requirements and can handle the expected loads. It also allows for identification of slow endpoints which can be further optimized to improve the overall system performance. In addition, response time can be used as a benchmark for future performance improvements and can provide valuable data for capacity planning and resource allocation.**

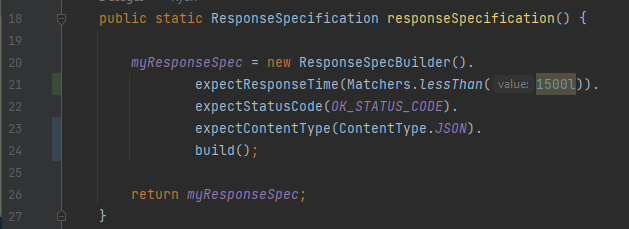
****

Figure : Line 21 on this code shows the code added for checking the response time is less that 1.5 seconds. This response specification static function is called in most of the API endpoint tests for ensuring that the response headers of these test responses to requests have the contain the expected values. This code further emphasises the use of matchers for testing as above I show that in the expectResponseTime function matcher less Than is used to ensure that the response time is less than 1.5 seconds.



Figure : Above shows the outputs after the response time test is added on the right hand side of each of these tests. As all the tests meet optimal performance standards by taking under 1.5 seconds they all have passed.

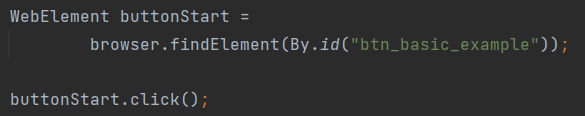
**Section 4**

**AC 4.1:** Explain and demonstrate at least three different techniques to reference web elements on a webpage

Below contains a number of methods for locating and manipulating specific elements like buttons links and, forms and other interactive elements on a web page using unique identifiers.

Element ID

Referencing the element ID, a web automation tool or test script can easily interact with the targeted element, such as clicking a button, filling in a form field, or validating its content. This method is widely used approach in web automation and testing and is supported by most web automation frameworks and libraries like selenium.



Tag names

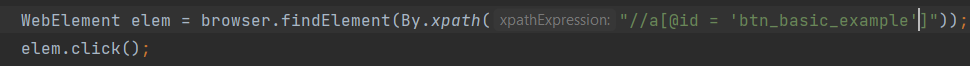
This is a method of referencing a group of elements on a web page based on their HTML tag name. HTML tags are used to structure and define the content of a web page, and each tag has a specific purpose such as defining headings, paragraphs, lists and other elements This method is useful when the specific element ID or class is not available or when interacting with a group of elements that share the same characteristics. However, it may not be as precise as other methods since multiple elements can have the same tag name, and it may require additional filtering or validation to ensure the correct element is targeted.

Graphical user interface, text, website

Description automatically generated

XPath

Referencing a web element by XPath is a method of locating and manipulating specific web element on a web page using its XML path. XPath is a language used to query and navigate XML documents, including HTML pages, and can be used to locate specific elements based on their attributes, text content, or relative position within the HTML hierarchy. By using XPath expressions, a web automation tool or test script. XPath provides a powerful and flexible method for referencing web elements and is widely used in web automation and testing. However, XPath expressions can be complex and may require additional validation to ensure the correct element is targeted, and changes to the HTML structure may require updates to the XPath expressions.

****

**AC 4.2:** Explain what is meant by the Document Object Model (DOM) and how it allows a Hypertext Markup Language (HTML) webpage to be accessed.

The document object model is a programming interface for web documents that allows scripts to dynamically access and manipulate the content, structure, and style of a hypertext markup language (HTML) webpage. The DOM represents the webpage as a hierarchical tree structure of objects, where each element, attribute, and text node is represented as a node in the tree. The DOM provides a way for scripts to traverse and manipulate this tree structure, allowing them to interact with the webpage’s content and behaviour.

To access an HTML webpage using the DOM, a web browser loads the page and parses its HTML code to create a DOM tree that represents the page’s structure and content. The DOM tree can then be accessed and manipulated using JavaScript or other programming languages, allowing developers to modify the content, style and behaviour of the webpage dynamically. For example, a script can use the DOM to add and or remove elements, change the style of content of an element, or respond to user events such as clicks or keystrokes. The DOM provides a powerful and flexible way to interact with webpages and is widely used to web development and testing.

To access the document object model using a web browser you can use the browser’s built-in developer tools.

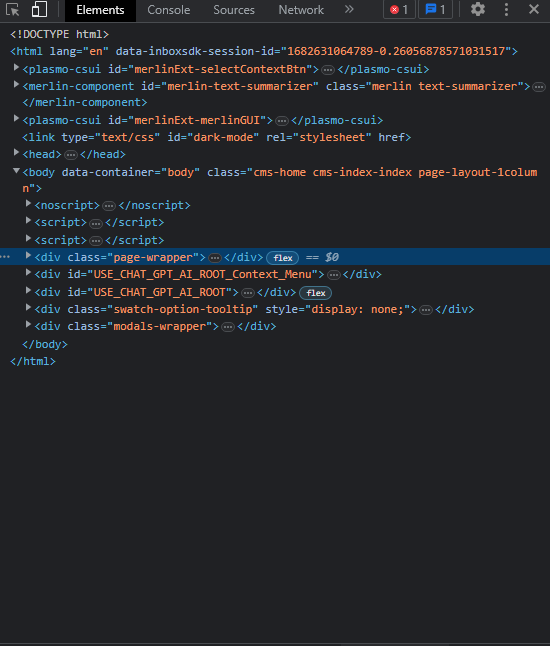
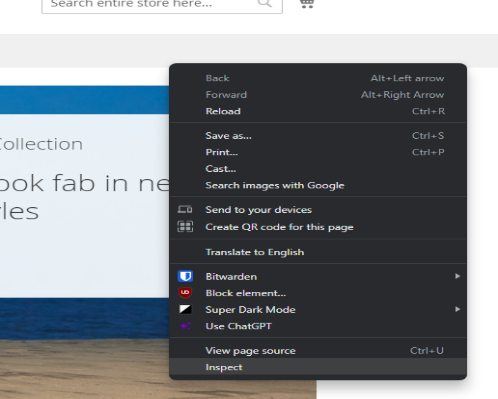
****To open the Document object model

Figure : Start by clicking on a webpage and selecting "inspect" form the context menu.

Figure : Then this element page will open up.

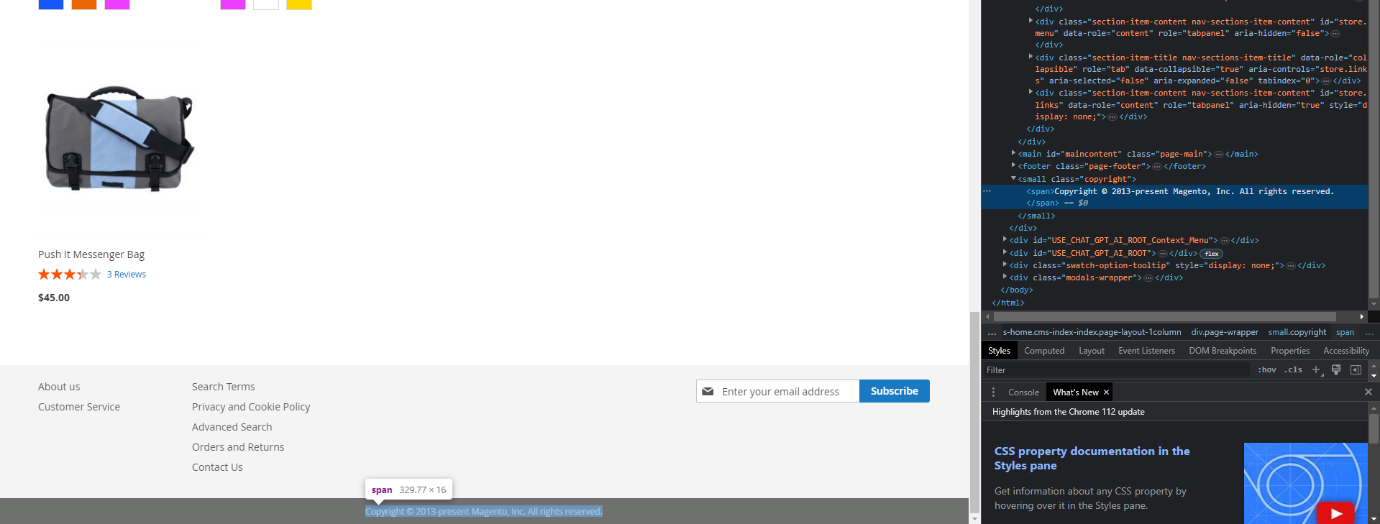
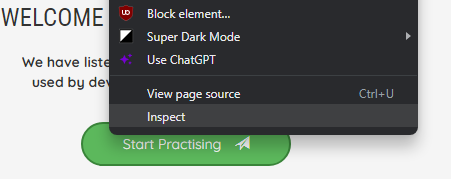
****Now within the DOM you can navigate and manipulate the DOM tree by selecting and editing elements in the elements panel.

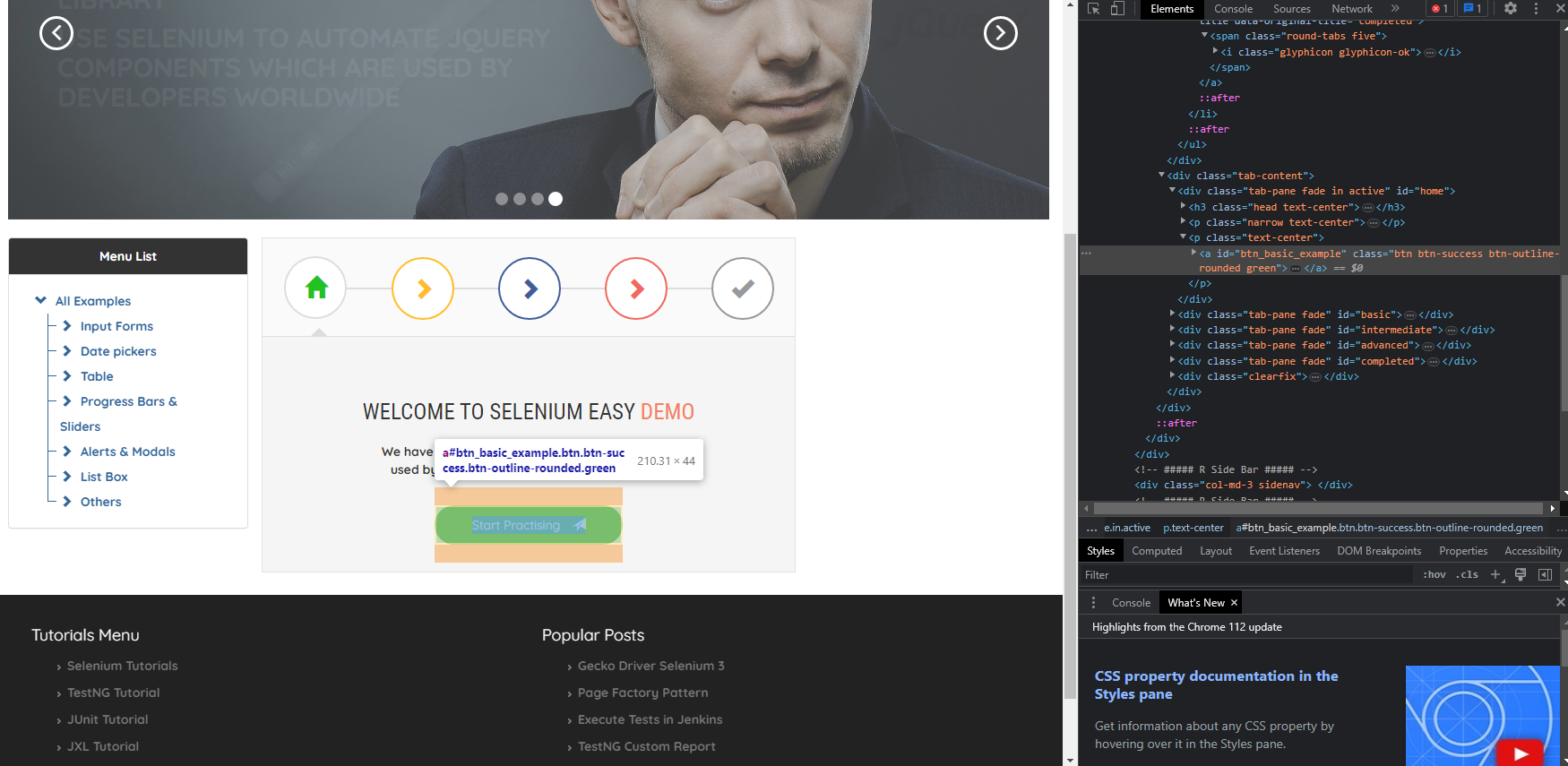
Figure : This shows the DOM after the changes are made the copyright label and the changed text.

Figure : Below shows an example the DOM being expanded to show the properties of teh class copyrights text. Within here I change edit the text to show something else.

You can also utilize the DOM in order to write selenium tests for example If I wanted to write a test that would check if a webpage button took you to another web page, I would start by right clicking the button I am testing which will open a drop down, then clicking inspect which will open up the element properties

****

Within the DOM I can then retrieve information which will be used for the selenium test in this example we are going to use the button id shown in the highlighted region of the DOM file.

****

This button ID is then used within the test below in order to retrieve the element, Then the element Click() is used the transition is shown below.

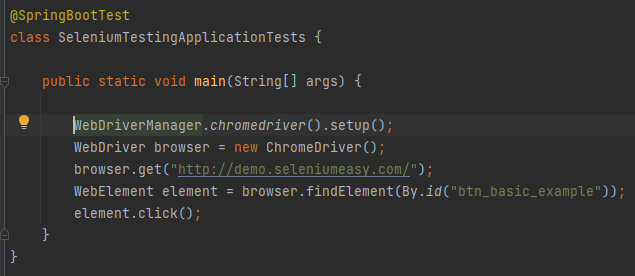
****

Figure : Selenium test using element ID retrieved from the DOM.

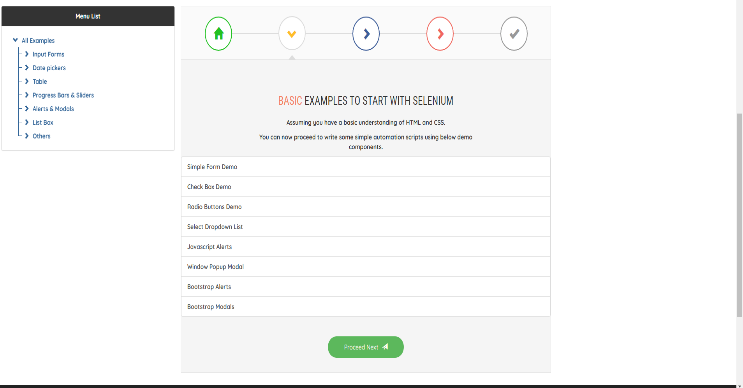
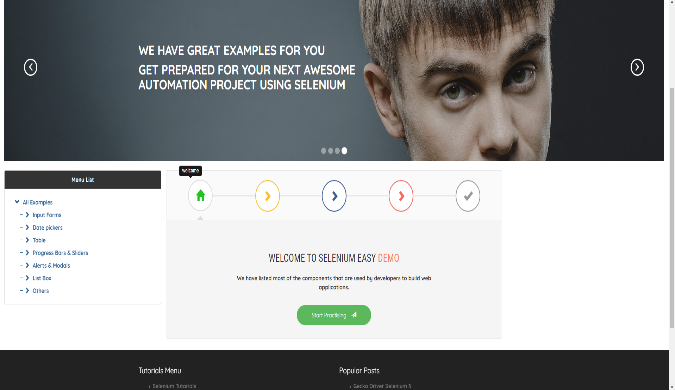
****

Figure : Web page at start of test.

Figure : Web page at end after web elements .click function was called.

****

**A.C 4.3:** Demonstrate the use of browser developer tools for testing – need to show some examples of testing including screenshots**. Show examples of developer toolds gor aapi tests and such test the website functions and what not. Inspect elements for look for locators for selenium for example.**

Browser developer tools are essential for testing web applications. There are several tools that can be utilized to aid testing some of these examples are shown below.

Inspect HTML and CSS: The developer tools allow you to inspect and manipulate the HTML and CSS of a web page. This can be useful for identifying layout issues, checking the box model, and testing responsive design. An example of the browser developer tools being used to inspect the HTML is shown in the example shown in 3.2

* Debug JavaScript: You can use the console to debug JavaScript code. This can be useful for identifying errors and testing the behaviour of your code.
* Network analysis: You can use the network tab to analyse network traffic, including requests and responses. This can be useful for identifying slow loading resources and checking for errors in HTTP requests.
* Testing mobile views: The developer tools allow you to simulate different screen sizes and resolutions, which can be useful for testing how your website looks on different devices.
* Testing user interactions: You can use the developer tools to simulate used interactions, such as clicks and scrolls. This can be useful for testing the functionality of your website.
* Performance testing: You can use the performance tab to analyse the performance of your website, including page load times, resource loading, and network activity.
* Accessibility testing: The developer tools have built-in accessibility auditing tools that can be used to test the accessibility of your website, including checking for keyboard navigation and screen reader support.

Overall browser developer tools are a powerful resource for testing web applications, and their many features can help developers ensure that their websites are running smoothly and efficiently.

**AC 5.1:** Critically evaluate different types of notifications and responses received from interactions within a browser

Interacting with a web browser can generate various types of notifications and responses. Each type has its advantages and disadvantages. Pop-up notifications, for instance, are small windows that appear on the screen and provide users with important information or alerts. They are often used to prompt users to take certain actions or provide updates about their activities. While pop-up notifications are attention-grabbing and can quickly convey important information to users, they can also be annoying and disruptive, especially if they appear too frequently or interrupt the user's browsing experience. Moreover, pop-up blockers may prevent users from seeing these notifications altogether.

Toast notifications, on the other hand, are brief, unobtrusive messages that appear at the bottom or top of the screen. They are usually used to notify users about actions that have been completed, such as a successful login or a completed file upload. Toast notifications are less disruptive than pop-up notifications and can still convey important information to users. However, they can be easily missed if the user is not paying attention or if they disappear too quickly. Browser alerts are notifications that are built into the browser itself, rather than being generated by a website. These alerts are often used to warn users about potential security risks, such as when they are about to download a file from an untrusted source. The advantage of browser alerts is that they are highly reliable and difficult to ignore. However, they can also be overly cautious and may generate false positives, leading to frustration and annoyance for users. Error messages are notifications that appear when something goes wrong with a website or application. They can provide users with information about what went wrong and how to fix it. The advantage of error messages is that they can help users troubleshoot problems and recover from errors. However, they can also be confusing and frustrating if they are too technical or if they do not provide enough information to help users understand. When web-related errors occur, status codes are returned as responses to reflect these errors. Some websites and applications have error handling to provide clear information about what happened when this error occurred. However, some don't and will just show the response code. Without technical knowledge, the meaning of these codes means nothing.

In conclusion, each type of notification and response has its advantages and disadvantages. Designers and developers need to carefully consider the type of notification or response that is most appropriate for each situation and design them in a way that is clear, informative, and user-friendly. It is essential to strike a balance between grabbing users' attention without causing annoyance, providing informative alerts without being overly cautious, and conveying errors in a clear, understandable manner without overwhelming the user with technical details. Ultimately, a well-designed notification and response system can improve the user experience and lead to increased engagement and satisfaction.

**AC 5.2:** Summarise common exceptions and codes generated through different test actions here you would talk about error codes and status codes and exceptions– exceptions that cause the exception codes shown above. Examples could be a connect exception thrown when a test is run. Element not found selenium exceptions from api tests and so on. Like connection exceptions and the status codes associated with them

HTTP protocol is used to communicate with a web browser and HTTP response status codes are sent in response to a message sent to a browser. There are several types of response codes, including informational responses (1xx), successful responses (2xx), redirection messages (3xx), Client error responses (4xx)

3xx 300 – 399 redirection messages

4xx 400 – 499 client error responses

400 bad request the server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing.

401 unauthorised: Although the http standard specifies unauthorized, semantically this response means, “authenticated” that is, the client must authenticate itself to get the requested response.

402 payment required

404 not found

The server cannot find the requested resource. In the browser, this means the URL is notrecognized.

**AC 5.3: Use an appropriate test strategy to effectively manage and interact with a new tab or window.** – how to use selenium to test popup windows work as expected. Talk about the parent window (being the window first open by the selenium test, the child window being the popup) and the child window

**Section 6**

**AC 6.1:** Research and evaluate the advantages and disadvantages of implementing a test suite

In today's software development industry, the importance of testing cannot be overstated. A test suite is a collection of tests designed to test the functionality of a system or application. One of the popular testing frameworks used to implement a testing framework is TestNG, which is used for functional, unit, and integration testing. While implementing a test suite has several benefits, it also has some drawbacks that must be considered.

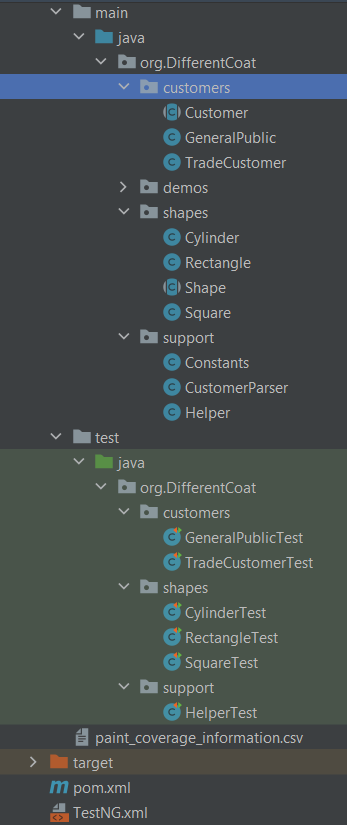
One of the primary benefits of implementing a test suite is that it increases the reliability of the system. A test suite ensures that all the critical functionality of the system is thoroughly tested, and the system is validated against various possible scenarios. This helps in early detection of defects, which can then be fixed before the system is deployed. This helps reduce the cost of fixing bugs later in the development cycle. Consistency is another advantage of using a test suite. Test suites ensure that the system behaves consistently across different scenarios, helping ensure that the system meets the required specifications and provides a stable user experience. Regression testing, which is essential to ensure that new features or changes do not break existing functionality, can also be performed using a test suite. This ensures that the system remains stable and consistent even after changes are made. Additionally, a test suite helps improve productivity as with a test suite in place, developers can quickly identify and fix defects, leading to increased productivity.

While implementing a test suite has several benefits, it also has some disadvantages that must be considered. One of the significant drawbacks is that it is costly to develop and maintain a test suite, especially for complex systems. This may lead to a higher development cost, which can be a significant drawback for smaller organizations or start-ups. It is also time-consuming to develop and run test suites, especially for large systems, which can lead to delays in the development cycle, which may not be acceptable in certain situations. Moreover, a test suite can give a false sense of security, leading to inadequate testing of critical functionality or failure to detect certain types of defects. Maintaining a test suite requires ongoing effort and resources, including updating tests as the system changes, fixing false positives, and addressing issues in the test environment. Finally, a test suite can only test for known scenarios, which may not cover all possible scenarios, leading to untested scenarios or unanticipated issues in the production environment.

In conclusion, implementing a test suite has several advantages, including increased reliability, early defect detection, consistency regression testing, and improved developer productivity. However, it does have some disadvantages, including cost, time consumption, false sense of security, maintenance overhead, and limited scope. Ultimately, the decision to implement a test suite should be based on the specific needs and requirements of the system being developed. While a test suite can increase the reliability of the system, it is important to weigh the benefits against the costs and potential drawbacks of implementing a test suite. Therefore, developers should carefully consider the pros and cons of implementing a test suite to determine whether it is appropriate for their specific needs.

**AC 6.2: Create a test automation suite for use with multiple test cases for test execution purposes**

The test automation suite for the paint coverage calculation solution designed for different coats is presented in Figure 1. The project includes two folders: "java" for project implementation and "test" for several tests that utilize the TestNG library to ensure the system's functionality as expected. Notably, the package and class structure within the "test" folder mirrors that of the "java" folder, which is a best practice to ensure that tests are easily maintainable and locatable. In addition to the test classes, the "main" folder also includes a file named "testNG.xml" that defines the hierarchy of the test classes to run when a regression test is conducted. This file serves as another valuable tool for testing purposes.

****By implementing a comprehensive test automation suite with organized and easily maintainable tests, the paint coverage calculation solution can be thoroughly tested and optimized for maximum efficiency and effectiveness.

**Text

Description automatically generated**Displayed in the figure below are the contents of the TestNG.xml file. Although not a mandatory requirement for conducting regression tests via Maven, this file can significantly aid in organizing the test class execution order. It is recommended to run the test classes for "shapes" and "support" first, as they serve as essential dependencies for the "customer" classes to run as expected.

Below is one of the test classes belonging to the test suite. It contains four tests for the rectangle class and will be the third test class ran based on the TestNG.xml file.

**Text

Description automatically generated**

Text

Description automatically generated with medium confidenceAfter using the mvn test function a regression test is run and all the test classes within the test suite are ran. Below shows the final output of these tests in console.

**Section 7 NEED TO FIX 7.1**

**AC 7.1: Research and evaluate the advantages and disadvantages of using a test automation framework** – Apply this to rest assured as it is a test automation framework.

Test automation frameworks have become an essential tool for software development teams looking to optimize their testing processes. They offer numerous advantages, such as code reusability, scalability, consistency, accuracy, and faster feedback, which can significantly improve the efficiency and effectiveness of testing. However, like any tool, they also come with their share of disadvantages, which must be carefully evaluated before implementing a test automation framework.

One of the most significant advantages of using a test automation framework is code reusability. This feature allows development teams to reduce the effort required to write and maintain test cases, resulting in faster test case development and execution. Additionally, automated test cases can be easily modified and reused across multiple projects and releases, reducing the time and effort required for regression testing.

Another advantage of test automation frameworks is scalability. As organizations grow, they need to expand their testing efforts to support multiple projects, test cases, and environments. Test automation frameworks can be easily scaled to accommodate these needs, allowing development teams to quickly add new test cases or make changes to existing ones.

Test automation frameworks also provide consistency, which is particularly important for complex applications that require extensive testing to ensure their functionality. Automated tests run the same way every time, providing consistent results and reducing the likelihood of errors or inconsistencies.

Automated tests are less prone to errors than manual tests, making them more accurate and reliable. Automated tests follow a predefined script and are not subject to human error, resulting in more accurate and reliable test results. This helps teams identify and fix issues more quickly, improving the overall efficiency of the testing process.

Finally, automated tests can be executed much faster than manual tests, providing faster feedback on the quality of the application under test. This allows development teams to identify issues and address them more quickly, resulting in faster delivery times and a more efficient testing process.

However, implementing a test automation framework also comes with its share of disadvantages. For example, setting up a test automation framework can be time-consuming and require significant resources. The framework must be configured correctly, and the tests must be designed to work with the framework, resulting in delays in the testing process and increased costs.

Test automation frameworks require ongoing maintenance to ensure that they remain up to date and continue to work correctly. This includes updating scripts to reflect changes in the application under test and addressing issues that arise during testing. Additionally, test automation frameworks can be expensive to implement, particularly for small organizations with limited resources, resulting in a higher upfront cost for testing, which may be difficult to justify for organizations with limited budgets.

Automated tests are not always suitable for all types of testing, particularly those that require subjective analysis or user interaction, limiting the effectiveness of automated testing in certain scenarios and requiring additional manual testing. Finally, automated testing can provide a false sense of security if it is not properly designed and executed, resulting in inaccurate test results.

In conclusion, test automation frameworks offer significant benefits, including increased reusability, scalability, consistency, accuracy, and faster feedback. However, they also come with drawbacks, such as the initial setup time, ongoing maintenance, cost, limited scope, and the potential for a false sense of security. Organizations considering implementing a test automation framework should carefully evaluate these advantages and disadvantages to determine whether it is the right choice for their specific needs and circumstances. A well-designed and properly executed test automation framework can significantly improve the efficiency and effectiveness of testing, resulting in faster delivery times and higher-quality software.

AC-7.2: Create a basic test automation framework for a given application that features reusability of code and low-cost maintenance – talk about Jackson and json data driven tests and the response and request header reuse.

I have developed a test automation framework for the staff API, as depicted in the accompanying screenshots. The figure below displays the project structure of the test suite, with three primary packages: "all Tests", "apiTestMethods" and “helper. The "all-tests" package comprises four classes, each with tests for a specific request type (POST, PUT, GET, and DELETE). Collectively, these classes cover all five of the API's endpoints (get-all-staff, get-staff/{ID}, delete-staff/{ID}, update-staff/{id}, add-new-staff-member and delete-staff-member/{ID}) each of these test classes use the data-driven testing method of software testing which involves using different types of data to test various scenarios of a software application allowing for efficient and comprehensive test cases that cover multiple scenarios in a single test case I have done this by using testNG parameterized testing. This test automation framework has been created with the use of several test frameworks used in conjunction to create a cost effective and reusable test suite which will allow for low cost maintenance of the staffAPI by allowing for automated testing and allowing for regression testing.

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

Description automatically generated

Figure : The project structure of the test suite is displayed above.

Post API

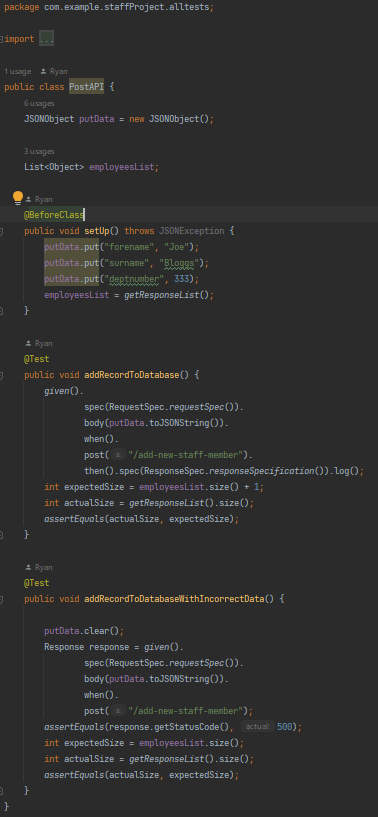
This class contains tests covering the post request "add-new-staff-member" for this API. The class includes two tests: one to verify if a request with data in the correct format was sent would this record be added to the database, and another to check if data with incorrect formatting is rejected.

Figure : Above shows the setup and fields for the PostTests



Figure : Above shows the two test functions in this class the first checks that the status code 200 representing success is returned and that the management records in the database are increased by 1. The second tries to post invalid data and checks if the status code 500 is returned representing that internal server error occurred due to the unexpected condition due to incomplete data being sent in the request body.

Put - API test class

Below shows the test class for the PUT request end point /update-staff/{id} there are a number of tests ensuring that this endpoint works as expected.

Figure : This is the PUT API testing class it has 3 test functions the first checks if the update-staff endpoint works as expected. The second sees check to see if an invalid request body is sent will the correct error type be thrown and the third checks what happens when the path variable is some random text instead of a manager Number existing in the staff table.