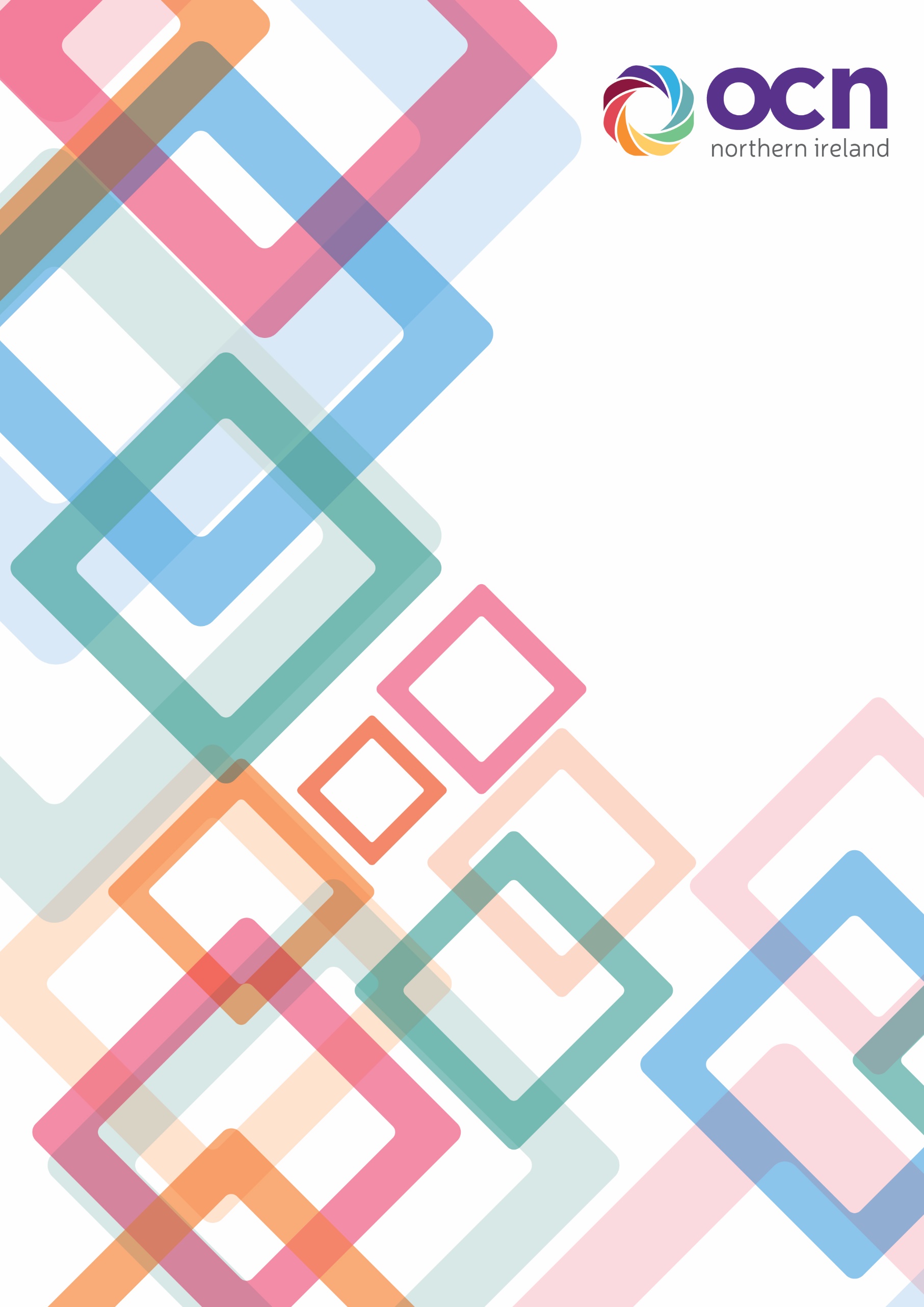
****

**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/No | 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/No | 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. – done   2. Critically compare the application of manual and automated software testing for a given application. – write up done need to show and example   3. Develop an optimal software testing plan |
| 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 – needs work   2. Explain Hyper Text Transfer Protocol (HTTP) and its ability to retrieve appropriate resources. – do this   3. Explain and demonstrate the use of at least four methods to assist the testing process available within a given testing tool.   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.   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. – do this |
| 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.   2. Summarise common exceptions and codes generated through different test actions.   3. Use an appropriate test strategy to effectively manage and interact with a new tab or window. |
| 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.   2. Create a basic test automation framework for a given application that features reusability of code and low-cost maintenance. |

**ASSESSMENT CRITERIA (AC)**

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

Explanation here on test automation etc.

**Each answer to an assessment criterion should be on a separate page where possible**

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.

**(AC-2.1 - Research the advantages and disadvantages for test automation practices and their associated resources ) – need to talk about resource not sure what is expected for this if it’s the libraries used for test automation or what.**

Test automation is a widely adopted practice in the software industry due to its numerous advantages, but it also comes with several drawbacks. Therefore, it is crucial to evaluate the pros and cons before implementing automation. One of the primary benefits of automation is its ability to enhance testing speed and efficiency compared to manual testing. Automated tests can run repeatedly and consistently without human error, offering prompt feedback on whether the application functions correctly. Besides, it facilitates comprehensive test coverage by enabling easier testing of various scenarios and edge cases. This, in turn, ensures thorough testing and the detection of potential issues early in the development cycle, reducing the overall cost of testing and preventing costly production errors. Furthermore, automated tests can be integrated into continuous integration processes, leading to automatic and quick testing of code changes, minimizing the possibility of regressions and ensuring the application remains stable.

However, despite these benefits, test automation has several disadvantages. Firstly, it involves a significant initial investment in developing and maintaining automated tests, which can be costly. Additionally, automation requires specialized skills that may not be available in-house, necessitating additional resources and expertise. Secondly, not all tests can be automated, particularly those that require human intuition and creativity, such as usability or exploratory testing. Consequently, manual testing is still necessary in some scenarios. Thirdly, automated tests require maintenance, like any other code, to remain relevant as the application evolves, which can be time-consuming and require additional resources. Fourthly, automated tests can report false positives, leading to frustration among developers and testers and reducing confidence in the automated testing process. Finally, automated tests can only test what they are programmed to test, potentially missing issues that a human tester would notice. Therefore, it is essential to supplement automated testing with other methods to ensure comprehensive testing of the application.

**(AC 2.2- Critically compare the application of manual and automated software testing for a given application) – need to apply this to a given application.**

Manual testing and automated testing are two primary methods of software testing. Both methods have their advantages and disadvantages, and the choice of which one to use depends on various factors such as the nature of the application, the level of testing required, the time and budget constraints, and the availability of resources.

Manual testing involves a tester executing test cases manually, without the use of any automated tools. The tester simulates user actions to ensure that the application behaves as expected. Manual testing requires a skilled tester who can identify defects that an automated tool may miss. Manual testing is useful when the application's user interface is complex or when the application has many different test cases that are difficult to automate.

One of the advantages of manual testing is that since it is performed by a human, it can identify issues that may be missed by automated testing tools, making it more thorough. Another advantage is that it is less expensive than automated testing since it does not require the use of specialized tools or equipment. Manual testing is also useful when testers need to perform ad-hoc testing or exploratory testing to uncover hidden defects. However, manual testing can be time-consuming, especially when testing large or complex applications. It is not scalable as it requires additional resources and time for each new test case, and it is subject to the tester's bias, which can lead to non-repeatable results.

Automated testing involves using specialized tools to execute test cases automatically. Automated testing can save time and effort, and it can be useful for testing complex or repetitive tasks. Automated testing is suitable when the application has many test cases that are well-defined and repeatable.

One of the advantages of automated testing is that it can save time and effort since it can execute test cases much faster than a human tester. It is also scalable, as it can execute many test cases in a short period. Additionally, it is more repeatable than manual testing since the same tests can be executed multiple times with consistent results. However, automated testing requires specialized skills to develop and maintain automated test scripts. It can have a high initial cost for acquiring and setting up the necessary tools and equipment, and it may not be effective in identifying defects that require human judgment, such as visual defects or usability issues.

In conclusion, the choice of manual or automated testing for a given application depends on various factors, including the application's complexity, the level of testing required, and the available resources. Manual testing is better suited for ad-hoc testing and complex user interfaces, while automated testing is more appropriate for repeatable tasks and scalable testing. Ultimately, a combination of both manual and automated testing may provide the most effective testing approach. It is essential to consider the advantages and disadvantages of each method to make an informed decision that meets the needs of the specific application being tested.

(AC 3.1 - Explain the setup procedures for using a web-based automated software-testing tool.)

3.1. is project api:

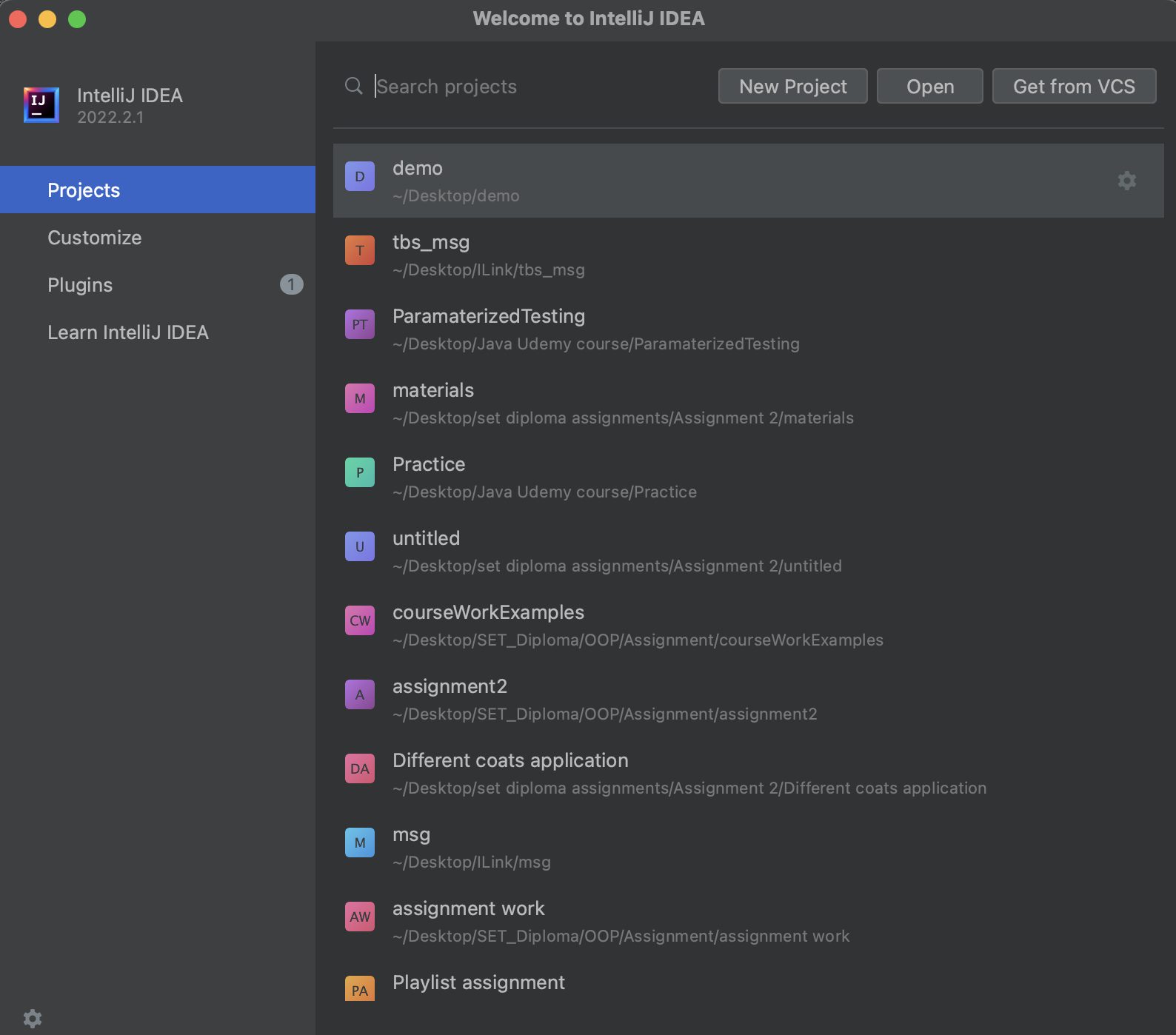
Talk through steps of setting up a api testing environment and show screenshots

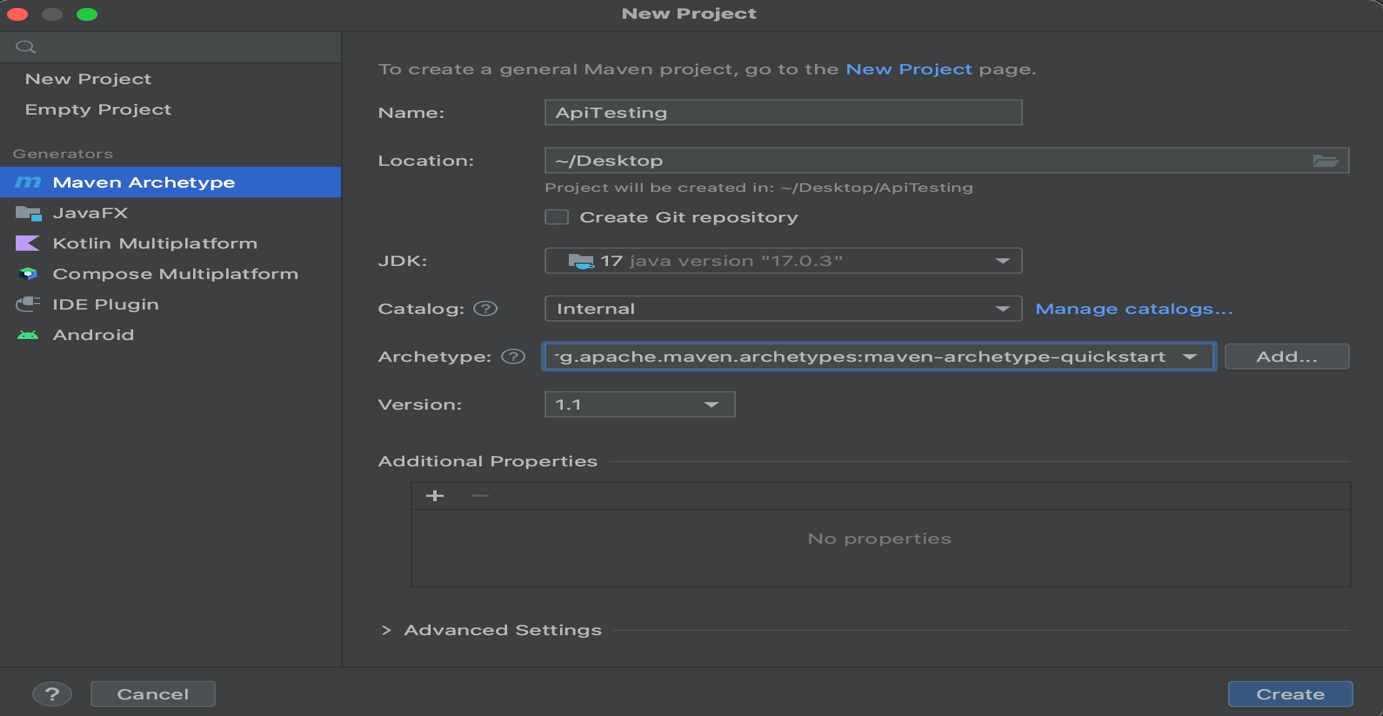
Step 1: create maven project then

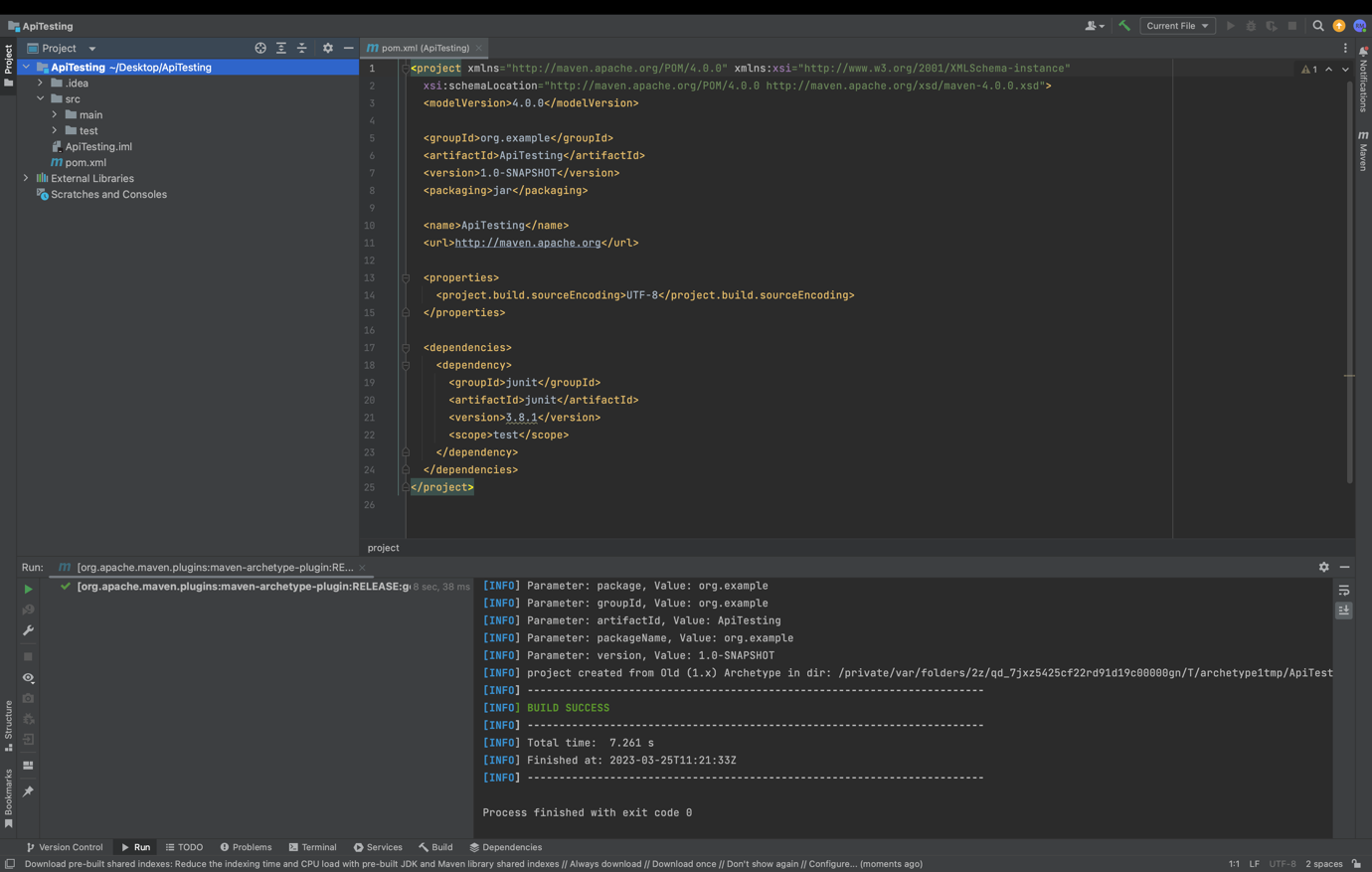
* go to pom import key dependencies testNG to allow for tests to be run rest assured to use rest assured interface and pick up on them classes json too.
* last thing show the method deleteARecordThatDoesExists which is an example of a test that is run after this setup is complete

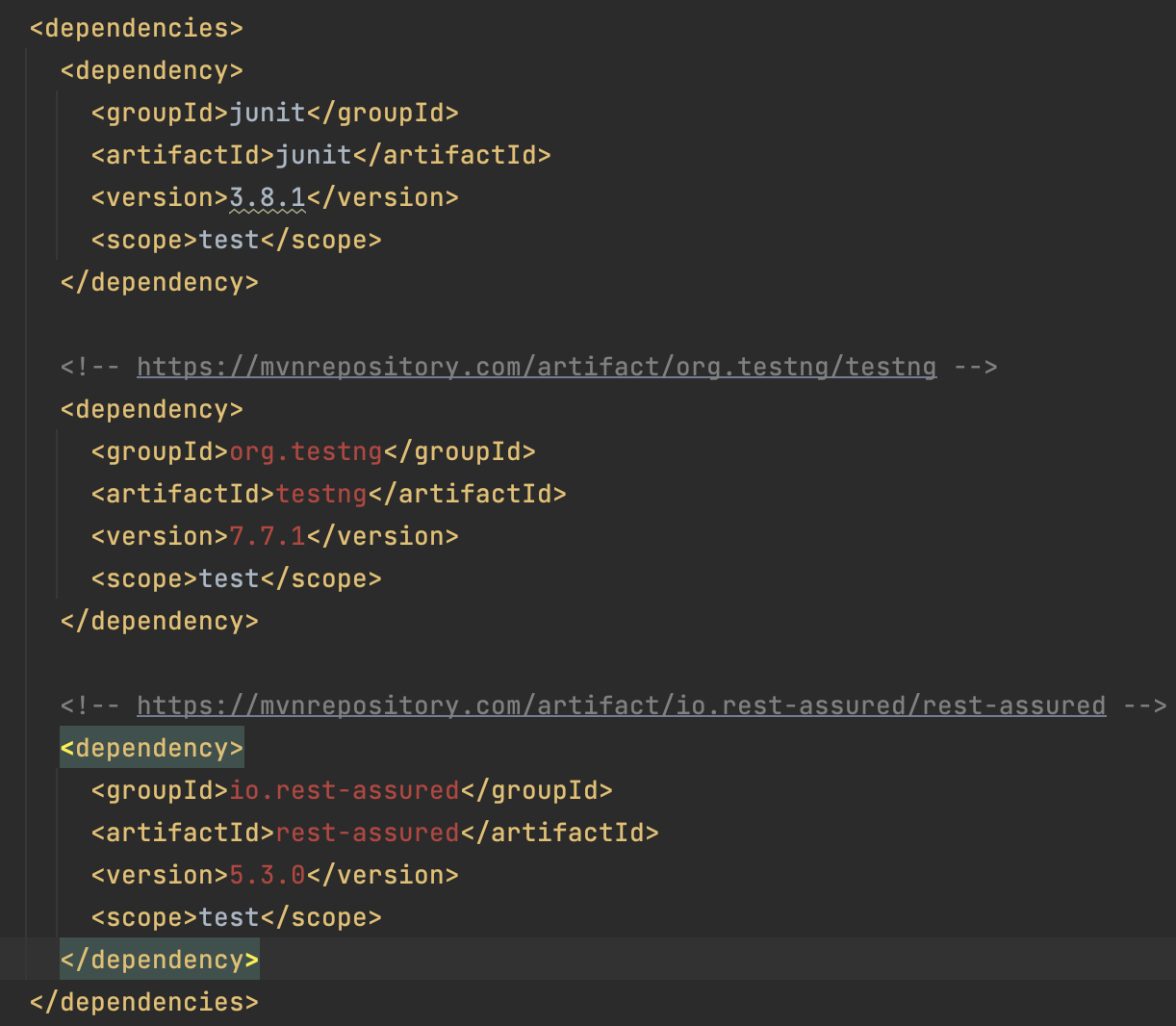
Setting up a web-based automated software testing tool is relativaly simple

Step 1: In this step in whatever you create a new maven project in whatever IDE you are using









**Graphical user interface, text

Description automatically generated**

**(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.

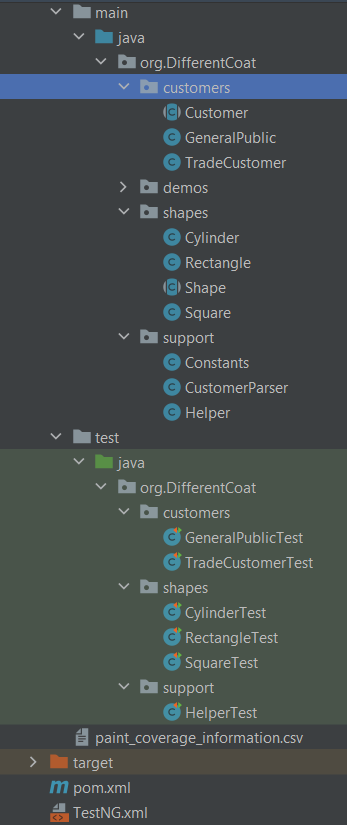
One of the primary benefits of implementing a test suite is that it increases the reliability of the system. A test suite ensures that all the critical functionality of the system is thoroughly tested, and the system is validated against various possible scenarios. This helps in early detection of defects, 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.

**(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.