|  |  |
| --- | --- |
| **Course** | Level 4 Diploma in Software Testing |
| **Course Code** | 603/5262/0 |
| **Unit Name** | Software Testing |
| **Unit Number** | CBE720 |
| **Assignment No.** | 1 |
| **Assignment Name** | Creation and Management of the Test Process |
| **Assessor Name** | Kevin McLaughlin |
| **Internal Verifier Name** | Michelle Simpson |
| **Internal Verifier Signature** | Michelle Simpson |
| **Internal Verification Date** | 19/10/2022 |
| **Assignment Release** | 21/10/2022 |
| **Assignment Submission** | 04/11/2022 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Learning Outcomes (LO) Covered By Assignment** | | | | | |
| **LO 1** | | Understand the process of software development | | | |
| **Criteria to be Assessed (Tick if Achieved)** | | | | | |
| **1.1** | | **1.2** | | | |
| **LO 2** | | Understand testing strategies, techniques and management | | | |
| **Criteria to be Assessed (Tick if Achieved)** | | | | | |
| **2.1** | **2.2** | | **2.3** | **2.4** | **2.5** |
| **LO 4 (Part)** | | Understand how to design a test strategy | | | |
| **Criteria to be Assessed (Tick if Achieved)** | | | | | |
| **4.3** | | | | | |

|  |  |
| --- | --- |
| **Learner Name** | Ryan McKee |
| **Date Submitted** | 04/11/2022 |

|  |  |
| --- | --- |
| **Assessor Comments** |  |

|  |  |
| --- | --- |
| **Assessor Declaration** | **I believe that the evidence submitted for this assignment is the learners own** |
| **Assessor Signature** |  |
| **Learner Declaration** | **I certify that the evidence submitted for this assignment is my own. I have clearly referenced any sources used in this work.** |
| **Learner Signature** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment Verified (Y/N)** | |  | | |
| **Grade Agreed (Y/N)** |  | | **Signature** |  |

**Tasks**

**Task 1**

The systems development lifecycle (SDLC) process was designed to ensure end-state solutions, meet user requirements in support of business strategic goals and objectives. A SDLC model is composed of several clearly defined and distinct work phases which are used by analysts, developers and testers to plan for, design, build, test, and deliver a software product.

Nowadays, computer systems have become more complex and undergo various development and maintenance needs on a regular basis. To manage this complexity, several specific models have been created. This complexity also adds to the increasing need for a comprehensive testing process being adopted.

You are required initially to identify and discuss this fundamental testing process that is widely used irrespective of SDLC model adopted.

Additionally, within this fundamental testing process, identify and discuss comprehensively the software testing approaches that are widely regarded as being key to a successful testing strategy. Highlight how different testing techniques are used within these aforementioned approaches in the pursuit of testing excellence.

**Criteria Assessed**

* 1. **1.2 2.1 2.3 2.4 2.5**

The software development life cycle refers to a methodology with a clearly defined process for creating high-quality software in the shortest time and at the lowest cost. The SDLC methodology focuses several phases the first being planning; As the first phase the leaders evaluate the terms of the project, including calculating labour and material costs, creating a timetable with target goals, and creating the projects’ teams and leadership structure as well as getting feedback from stakeholders to define the scope and purpose of the application. The second phase is requirements in this phase functional and non-functional requirements are defined for the application, functional requirements are the things that the application is expected to do like a booking system being able to book something, and non-functional these are things like accessibility, scalability and appearance. The third phase is Design and prototyping This phase models the way the software application will work this includes building the architecture, the user interface, and programming some functionality, this prototype is then given to the stakeholders to get some hands-on experience so they can give feedback and readjust the requirements based on the prototype the next phase is the actual Software Development during this stage development starts, Every developer sticks to the agreed blueprint and develops the application. After software development the code testing phase takes place, this phase is where the software is tested for defects and deficiencies, when these have been identified the issues are then fixed by the development team. After the previous phases are complete Software deployment occurs this is the final of the main phases within SDLC during this stage the goal is to deploy the software to the production environment so users can start using the product.

There are several methodology’s that the SDLC model can be applied to including Waterfall, spiral, and the agile. The most common SDLC model is implemented on top of agile and is called agile SDLC. This model utilizes the agile approach of having sprints in which tasks are completed and at the end of each sprint features and bugs that need to be addressed are taken from the backlog and added to the next sprint during these sprints you would pass through the phases in SDLC from requirements to deployment until the project has been completed.

Arguably the most important phases during the agile SDLC is the software testing phase in this phase testers partake in processes to ensure the quality of the product being developed typically the first type of testing used is unit testing; Unit testing tests individual use cases so essentially individual methods in classes are tested. The objective during UNIT testing is to get the code coverage to 100% so that it is assured that every single function works. Unit tests are typically written by the developer and are considered white box testing which is a testing type where the tester writes tests based on their knowledge of the internal structure of the program.

After unit tests are completed, it is important to next use integration testing. This type of testing groups modules together to see if they work together because although unit testing ensures that individual methods work it does not test if it all works together as intended there are multiple types of integration testing the main type follows the natural control flow hierarchy top to bottom this would take you from the start of the application to the end testing each of the features using black box testing which is when you test without knowing the internal structure of what you are actually testing these tests are typically performed by a tester and it is harder to detect defects than in unit testing making is potentially more costly.

System testing is the final phase in the testing process this process validates the complete and fully integrated software product using 2 different stages called alpha and beta testing the purpose of this is to evaluate the end-to-end system specifications this is done through forms of acceptance testing which is a quality assurance process that determines to what degree an application meets end user’s approval this testing checks for stability and checks for flaws, Acceptance testing encompasses various types. The main two types used during this phase is alpha acceptance testing and beta acceptance testing. Alpha testing takes place before the product is released to the customers for beta testing and then deployment. Alpha acceptance testing is done by developers within the company to identify defects and bugs for the application, during this phase both black-box and white box testing are performed. This is typically where the bulk of the testing process takes place before release. After this testing is complete beta testing takes place where the product is released to the customer base to see if the product has any bugs that could not be found by the developers and also to ensure that the product fulfils their requirements of the customer this is entirely done through black box testing as the customers have no idea about the internal components, they can only use the application. Once this in complete if there is any issue that need to be addressed the team can go back a few steps in the software development lifecycle and address these issues before finally deploying the application.

During this entire testing phase, it is very important that the team is fluid and has good communication so that testing is as efficient as possible as when the QA team gets better information about issues and features that need testing they can start testing sooner and as they find defects they can communicate and resolve those defects more quickly with the developers, better communication also provides a better-shared understanding across all the stakeholders of the risks, issues, and impediments to the project and finally at the end because of the good communication a better quality product can be delivered. Another important part of the testing phase is having good project management; Project management provides a clear description of tasks for a team and adds abstraction for testers so they can focus on specific features and bugs that need testing and not get overwhelmed by the complexity of the entire system, good project management means that work will be divided well across the team making jobs easy to manage and ensuring no one gets burned out. project managers also provide the scope for testing so there is no ambiguity ensuring there is no confusion among the stakeholders of the project, Good Project management ensure quality and cost efficiency and can communicate updates among stakeholders. To improve communication among stakeholders in a project it is common practice to make documentation that is easy to understand for example during the software testing phase documents like the testing plan which covers information like what features/modules need tested and how it’s going to be tested and the expected outputs so that when it comes to testing it is simple. Then when tests are been completed a test log should be made showing the results of the tests so that it is easy to see what need to be fixed as screenshots of issues are added to the test report so it can be added to the backlog for the developers to fix these issues. these documents not only allow for testing to be simpler for the testers and developers but also allows for the non-technical stakeholders to easily understand the issues and what’s being tested and how.

During the software testing its essential that the correct testing techniques are used as this can make or break the actual effectiveness of testing. There are four main categories which testing techniques can be categorized into these are structural, functional dynamic and static techniques all of these have one thing in common in that they all are used to ensure adequate testing is done however they do have their differences: Structural testing tests to uncover errors during the coding of a program it looks at both the results and the process which the results are gotten it is used during every phase where design requirements and algorithms are discussed the main objective is to ensure that the functionality of the program is working find and the product is technically good enough to implement in the real environment structural testing is sometimes called white box. Some examples of structural testing include stress testing this is when tests are done to determine the robustness of software by testing beyond the limits of normal operation, this type of testing is important for applications that are considered mission critical but can be used for all types of software, other examples are execution testing where code is executed, and you compare the actual outputs to what is expected. There are several more examples like operations, recovery, compliance and security testing. Functional testing is similar in purpose to structural testing in that it wants to ensure reliability of the software however functional testing only concerns itself with the processing not the actual results some examples of structural testing include unit testing which I already talked about, Another example is component testing also known as module testing, this is where individual parts of the system are tested similarly to unit testing the main difference between unit and component testing is that unit testing is done by the developer in a white-box format to verify that program modules execute, while component testing is done by testers in a black-box format to validate individual objects or parts of the software. Smoke testing is another this is a type of acceptance test where there is an initial check that new software builds and its critical functionality stable. If the smoke tests pass, the build can undergo further testing. Smoke testing, also called build verification testing. If the smoke tests don’t pass additional development is needed before moving on. Every test done falls under dynamic testing or static testing, Dynamic testing is a method of accessing the feasibility of a software program by giving input and examining the output. The dynamic method requires that the code be compiled and run dynamic testing can be classified as either black or white box some examples of dynamic testing are unit testing, integration testing system testing, acceptance testing and performance testing. While static testing will analyse the code, requirement documents and design documents examples of how static testing can be performed include informal reviews where co-workers review documents and provide informal comments and technical/peer reviews where the technical specifications are reviewed by peers to detect any errors.

Overall, the software development lifecycle is a fundamental framework used in almost all modern software projects which provides clear standards of how to develop a good software product from beginning to end.

**Task 2**

A test plan is the foundation of every testing effort. It helps set out how the software will be checked and what tests will be performed. You are required to explain the key features of a test plan and how it is successfully managed based on various documentation produced.

A key goal of a test plan is the number of test cases designed based on a test condition. Identifying the relevant number of key test cases and test data is key to achieving this goal.

Initially investigate and discuss the key testing techniques used to acquire relevant test data.

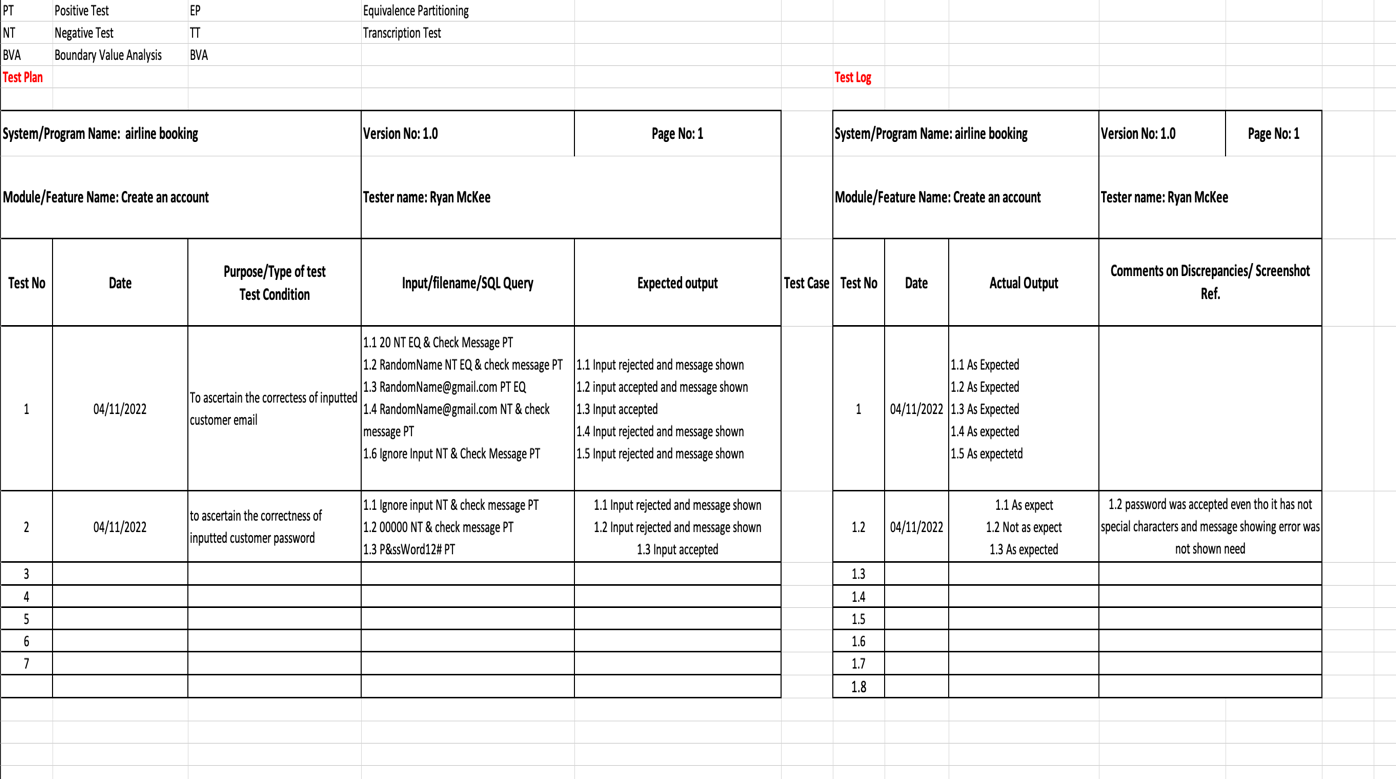
Finally, demonstrate these techniques providing relevant practical examples of their significance.

**Criteria Assessed**

**2.2 4.3**

A software test plan is a document that sets out the scope, approach and schedule of intended testing activities. The test plan may also list the resources the software tester needs to function effectively. The primary objective for a test plan is to produce documentation that describes how the tester will verify that a system works as intended. The document should describe what needs to be tested, how it will be tested, and who’s responsible for doing so. By writing up a test plan, all team members can work in conjunction with one another following the SMART objectives (aka. Specific, measurable achievable relevant and time bound goals). The test plan usually includes the following information like the overall objective of the testing effort. A detailed outline of how testing will be conducted (the test approach). The features applications or components to be tested and detailed scheduling and resource allocation plans for testers and developers throughout all stages of testing.

There are several different sections in a test plan these are: the master test plan this is the document that contains testing information like what the application is, who created it, the test version, when it was created, the objectives, scope, approach and what the focus of a software testing effort should be. This Document also shows all the different features that need to be tested on the project. The next documents are the test plans, each of these documents focus on individual features defined in the master test plan. Within each of these documents there is information about the feature. What the test version is, what application the feature is a part of and who the tester is. Within the testing plan there is several different test conditions these are a set of information describing what should be verified within the project then the test condition is a piece of documentation within the test plan that shows several test data, predictions and expected results then finally from that. when the tests are carried out the test data is produced this data is the data that has been identified after the test is completed.



Above is an example of a test plan for the create account feature of an airline booking application, In the test plan it shows information at the top of the table about the application like the system name, the feature being tested the tester name and the version number in this case which is 1.0. Underneath that there are two test cases one of ascertain the correctness of the customer email entered and another to test correctness of password entered. Within the tests above for the test data there is letters beside them e.g., PT, NT, BVA, TT, EQ, these letters represent the type of test being executed. In order to black box testing to have as much code coverage as possible it is important that the inputted data used in each of the test cases cover as much user entry possibilities to ensure that the features in the application work totally as expect. In black box testing there are many different types of tests but for the test plan above I used a few of them these include positive testing represented by PT where the input is expected to be accepted, Negative testing/ error testing where the input is expected to be rejected this is represented by the alias NT, Boundary value analysis represented by BVA which tests around specific boundary value for example where a specific field only accepts values between 0-99 you can test boundary values (-1, 0, 99, 100), Transitive testing which is when the test is used to prepare for any possibility of what the user might do this is represented by TT and finally Equivalence partitioning which is when a tester divides possible inputs into groups or partitions and test only one example input from each group.

These tests used in conjunction for example in the airline booking application test plan can ensure an error free application is produced.

References:

During this assessment referred to:

* Level 4 diploma in software testing course Details.pdf
* <https://www.datprof.com/solutions/what-is-test-data/>
* techtarget.com/searchsoftwarequality/definition/static-testing

**Assignment Criteria**

|  |  |
| --- | --- |
| **1.1** | Explain the stages of the software development lifecycle |
| **1.2** | Explain the importance of communication skills, project management skills, document creation and associated processes and procedures and how they are applied. |
| **2.1** | Explain the stages of system testing including alpha, beta, and acceptance testing |
| **2.2** | Explain the contents of a software test plan and how it should be managed |
| **2.3** | Explain the purpose of unit, integration and system testing of software |
| **2.4** | Critically compare the application of unit, integration and system testing of software |
| **2.5** | Critically compare functional and structural testing techniques and dynamic and static testing |
| **4.3** | Explain the various methods used to obtain data to test computer programs |