Day 2 Task

MVP Research Task:

Within a team you will be working alongside developers who will have done their

own testing of their code. They will have done this using two different types of

testing, component (unit) and component integration (CIT). Investigate both testing

types focusing on:

**a) What each one is**

*Unit Testing* – Unit testing is a type of testing which focuses on testing individual components of a software product. Each unit can be of different needs such as functions, or methods.

*Component Integration Testing* – Testing which tests the interactions between the components. Used to ensure interface compatibility with data types and formats as expected. Ensure the components once combined produce desired outcome for variety of conditions.

**b) When you would use each one**

*Unit Testing* – In Test Driven Development, For individual isolated tests, frequently and early in the development lifecycle, after a section of code is completed before integrations

*Component Integration Testing* – After component or unit testing, for Negative cases, exception handling, confirming communication protocols between components are correctly implemented.

**c) The advantages of each type**

*Unit Testing* – Early issue detection in development process, Reduced time for later testing, Allows for refactoring/changes to be implemented safely, better documentation, reduced uncertainty

*Component Integration Testing* – Early detection of integration issues, allows for smoother transition to higher level testing phases, reduces risk of system failures, helps in coordination/collaboration

**d) The disadvantages of each type**

*Unit Testing* – Complex systems can take a lot of time for creation and maintenance of tests, Developer dependant, Complex units can be difficult to isolate units, Not sufficient for integration tests due to isolation, Automation tends to be relied upon for these cases,

*Component Integration Testing* – Complex systems with many components can be hard to manage all interactions, can cause loss of time for creation and maintenance of tests, if a test fails it can be hard to pinpoint which component caused the issue

You looked at different types of functional testing within the morning lesson. We

would like you to research a few more. Write a summary about each of the below,

focusing on what they are, when you would use them and the advantages and

disadvantages.

**a) Sanity testing** - Sanity Testing/Surface Testing is a type of testing performed after smoke testing on stable builds to verify that minor changes have not caused any new defects, this tests can also be performed after defect fixes. Its primary objective is to ensure the core functionality of the software works as expected. This type of testing can be completed in both manual and automated testing, depending on circumstances.

Some advantages to this type of testing are quick issue detection and rejection of faulty builds, focused verification of functionalities and dependencies, time and cost effective. Likewise some disadvantages to this can be primarily focuses on the core functionality and can miss broader issues, complex systems will be harder to verify dependencies in limited scope, change of requirements for each iteration can cause missed priorities.

**b) Smoke testing** – Smoke Testing/Build Verification Testing is a type of testing performed at the beginning of the development process to ensure the most critical functions are working correctly. This helps to ensure that the build is stable enough to perform more extensive testing. Manual and Automated testing can be utilised for this type of testing. This testing can be applied at software build. But also at system level testing to ensure the main parts and modules of the software are operating in unison, and further can be implemented at integration testing level, to verify the integrated modules or components interact with one another and communicate correctly.

Some advantages of smoke testing include ensuring that future builds are built upon bug free software, and lessening the risks that come with adding code to existing builds, flexibility of manual and automation testing, ensures the build is stable, smoke testing is generally easy and fast to perform. Some disadvantages are that like many different types of tests it does not cover full functionality and is only certain areas covered, errors can be missed or still occur after smoke testing, there is usually a minimum number of tests and as such much can be missed, and if manually completed can be time consuming,

**c) Exploratory testing** – Exploratory testing is a more hands on type of testing where QA/Testers are given a bit more freedom from following set test cases and are actively exploring the application, much like a real user would and relies on the tester to uncover defects that are not easily covered. Exploratory testing is often used when a tester needs to quickly learn the software and provide feedback, further it can be used to find new test scenarios that might not have been found during requirement building, it can be effectively used for finding edge cases.

Some advantages of exploratory testing are that it allows for an analysis of the software as though in a live scenario, the testers creativity and individuality will give different results, further helps identify any user or GUI issues, provides insight outside of scripted tests. Some disadvantages are, there will tend to be a lack of documentation from these types of tests, testers may be biased in expectations due to personal preference, critical features may be missed during testing.

**d) Risk based testing** – Risk based testing is a type of testing which is based on the probability of risk, it involves assessing the risk based on complexity, criticality, frequency of use and possible expected defects areas. Using risk based testing, teams identify and assess these risks and follow a testing process to ensure the critical risks are tested early in the lifecycle. Risk based testing is most often implemented when there are resource, budget or time constraints, during security testing in cloud computing environments, continuously changing project or when testing for projects with high risk factors.

Some advantages of risk based testing are optimisation of resource usage, early detection of high risk defects, customer centric focus, fast adapting to changing requirements, focuses on high risk areas, helps prioritise test executions due to risk impact being quantified. Some advantages are high risk focus can cause potential lack of coverage, requires constant reassessment and realignment, not the most effective for some new applications, requires in depth knowledge of the domain.

**e) API testing** – API testing is a type of testing that focuses on the testing of individual API methods and the interactions between different API’s, as well as ensuring API’s are working as expected. There are several types of API tests covering functionality, security and performance. These can be tested manually or through automation. This type of testing is generally performed at the integration level after unit testing has been completed but before user interface testing begins.

Some advantages of API testing are, early issue detection and resolution, rapid iterations allow to validate code changes before production, faster speed and coverage of testing, helps identify and secure security vulnerabilities, confirmation of integration and communication. Some disadvantages are, API testing can be complex when testing APIs integrated with other systems or multiple at once, limited visibility of how the API is interacting with other components, time consuming for creation, execution and maintenance of tests, limited test coverage.

Testing can fall into ‘Black box’ or ‘White box’ testing.

**a) Investigate each of these types and write a summary on each, focusing on what**

**they are, when you would use them and the advantages and disadvantages.**

*White Box Testing* – White box testing is a type of testing that focuses on the code and internal structure of the product being tested. The tester will have access to the source code and utilise this knowledge to design test cases that can verify the correctness at the code level. This testing is used within unit or integration testing most commonly and is one of the more time-consuming types.

Some advantages to white box testing are that it leads to code optimisation by identifying errors in the code, it concentrates on using logic to evaluate the code which allows for more accurate release records, allows for automation of unit tests due to understanding of the code, allows for further reflection of the implementation of features. Some disadvantages are, due to the more thorough testing it can be more time and cost expensive to conduct, can only test what is currently fully implemented, automated test cases could become redundant due to code base changing regularly.

*Black Box Testing* – Black box testing is a type of testing that focuses on validating the functionality based on the specifications or requirements and does not consider the internal structure, design or product implementation. The black box only evaluates the external behaviour of the system. This testing is used within System testing or Acceptance testing and is one of the least time consuming.

Some advantages of black box testing are, testers can pay attention to only the functionality and access different information, allowing for experiences as though a user, less time consuming and allows for no need of programming knowledge. Some disadvantages are that with it focusing only on the functionality it can miss errors in the inner workings or structure, may create redundant test cases due to lack of knowledge needed, complex issues can be missed.

**b) Look at the testing types you have learnt today, both in the lesson and in your**

**research, choose 4 and decide if they class as Black or White box testing.**

**Justify your answer**

Boundary Value Testing

Unit Testing

Equivalence Partition Testing

Sanity Testing

*White Box Testing:*

*Unit Testing* – Involves testing individual units or components of the software in isolation, including functions and methods using the knowledge of the code source.

*Black Box Testing:*

*Boundary Value Testing* – Due to the fact it is checking the functionality primarily and validating the information taken to give the expected functional result.

*Equivalent Partition Testing* – Due to the fact it is checking functionality and validating the information given for a functional result.

*Sanity Testing* – Used when there are minor fixes to code, ensures the core functionalities are working as expected, as such it seems to be functionality prioritised.

Extension:

1) Imagine you are a tester in a team working on a payroll system. One feature

calculates income tax based on annual salary. The tax bands are as follows:

• £0 – £12,570: 0% (Personal Allowance)

• £12,571 – £50,270: 20% (Basic Rate)

• £50,271 – £125,140: 40% (Higher Rate)

• £125,141 and above: 45% (Additional Rate)

TO NOTE: Negative values are not accepted as inputs within this code base.

**a) Explain why Boundary Value Testing is particularly important when testing tax**

**band logic.**

Boundary Value Testing is important in tax band logic, because if the boundaries are not clearly set and if the numbers do not correlate to the correct rate, there would be larger or smaller tax rates being charged. In the case this was someone who should be on the lowest rate (personal allowance) and were right on the highest boundary, but this placed them into the Basic rate incorrectly, they would be charged 20% of their small income. Likewise, someone on the Additional Rate could be charged too little if they are right on the lowest boundary but incorrectly implemented to think it should be Higher Rate.

**b) Identify the key boundary values for the tax bands.**

0, 12’570, 50’270, 125’140, 125’141

**c) Design test cases that cover each boundary using Boundary Value Testing.**

*Test Case* *Input* *Expected Result*

PA Just Below Lowest -1 Invalid Input Message

PA Lowest Boundary 0 Personal Allowance

PA Just Above Lowest 1 Personal Allowance

PA Just Below Highest 12569 Personal Allowance

PA Highest Boundary 12570 Personal Allowance

BR Lowest Boundary 12571 Basic Rate

BR Just Above Lowest 12572 Basic Rate

BR Just Below Highest 50269 Basic Rate

BR Highest Boundary 50270 Basic Rate

HR Lowest Boundary 50271 Higher Rate

HR Just Above Lowest 50272 Higher Rate

HR Just Below Highest 125139 Higher Rate

HR Highest Boundary 125140 Higher Rate

AR Lowest Boundary 125141 Additional Rate

AR Just Above Lowest 125142 Additional Rate

**d) For the £0 boundary, -1 would be the lower, which isn’t a salary and hence now**

**a valid input. How would we deal with this?**

Due to negative numbers not being accepted as inputs, an error message would be expected to appear, prompting the user to use a valid input or valid number/salary. This would be tested with negative/positive testing.