1.Outline some of the difference between White Box Testing and Black Box Testing methods.

=

|  |  |
| --- | --- |
| Black Box Testing | White Box Testing |
| It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| Implementation of code is not needed for black box testing. | Code implementation is necessary for white box testing. |
| It is mostly done by software testers. | It is mostly done by software developers. |
| No knowledge of implementation is needed. | Knowledge of implementation is required. |
| It can be referred to as outer or external software testing. | It is the inner or the internal software testing. |

What typically a white Box Testing should cover in your opinion?

= White box testing should cover a comprehensive range of aspects including code coverage (statement, branch, path, condition, function, loop), code logic and flow, data flow, boundary conditions, performance and optimization, security (vulnerabilities, data validation), integration testing, code review and static analysis, API testing, compatibility, usability, documentation, error handling, concurrency and multithreading, database testing, and reusability and maintainability. The goal is to ensure that the code functions correctly, efficiently, securely, and is maintainable.

What is a good test case?

= A good test case is one that is clear, concise, and effectively verifies the functionality of the software under test. It should be based on requirements, cover different scenarios, and provide maximum coverage. A good test case should include a description, the steps to execute, the expected result, and any necessary preconditions or data setup. It should be easily reproducible, independent, and not redundant. Additionally, a good test case should be prioritized, focusing on critical functionalities and potential risks. Regular maintenance and updates to test cases ensure their relevance and effectiveness throughout the software development lifecycle.

How do you prioritize test cases for execution?

1. Understand the Requirements:

Analyze the requirements thoroughly. Understanding the criticality and importance of each requirement is essential.

Categorize the requirements as critical, high, medium, and low priority.

1. Risk Assessment:

* Identify the high-risk areas where failure would have the most significant impact.
* Risk assessment should include:

Probability of failure

Impact of failure

Coverage of the feature

* Test the high-risk areas first.

1. Business Impact:

* Prioritize based on the business impact of the feature being tested.
* High-impact features should be tested early.
* Features with the potential for revenue generation or customer satisfaction should be given priority.

1. Functional and Technical Complexity:

* Test the functionalities that are more complex or have a higher technical risk.
* Determine the dependencies of each test case and prioritize them based on dependencies.

1. Customer Usage Patterns:

* Understand the customer usage patterns and prioritize test cases accordingly.
* Functions or modules that are more frequently used by customers should be tested early.

1. Time Constraints:

* Consider the time available for testing. If there is a tight deadline, focus on critical areas.
* Use techniques like the Pareto principle (80/20 rule) - focus on the 20% of test cases that will cover 80% of the functionality.

1. Regression Testing:

* Regression testing should be given a high priority to ensure that new changes have not affected existing functionalities.

1. Automatability:

* Test cases that are repetitive and time-consuming are good candidates for automation. Automate these first.

1. Coverage:

* Ensure that the testing process covers all aspects of the application, including positive and negative scenarios, boundary conditions, and failure paths.

1. Peer Review:

* Conduct a peer review to get input from other team members. This can often reveal potential areas that may need higher priority testing.

1. Prioritization Techniques:

* Use prioritization techniques such as MoSCoW (Must have, Should have, Could have, Won't have), pairwise comparison, or ranking.
* Moscow Technique:

Must-have: Critical features or functionalities that are required for the system to work.

Should-have: Important but not critical features.

Could-have: Desirable features that can be left out if necessary.

Won't have: Features that won't be included in the current release.

1. Feedback from Previous Testing Cycles:

* Analyze the results and feedback from the previous testing cycles. If any areas were missed, prioritize those areas first.

1. Continuous Monitoring and Adaptation:

* Priorities may change as the project progresses. Keep monitoring, and adapt the priority list accordingly.

1. Communication:

* Keep stakeholders informed about the test priorities and any changes in the priority list.

1. Documentation:

* Document the reasons for prioritizing certain test cases over others. This can be useful for future reference and for training new team members.

What is the fundamental difference between Verification and Validation?

Following are some of the activities.

Categorize these activities them in terms of verification and validation and also mention the reason why

- Code inspection

- Black Box Testing

- Acceptance Testing

- Requirements specification review

=

|  |  |
| --- | --- |
| Verification | Validation |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does not include the execution of the code. | It does not include the execution of the code. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |

= Now, let's categorize the activities and provide a reason for each:

Verification:

Code inspection:

Categorization: Verification

Reason: Code inspection is a static testing technique used to evaluate the code against coding standards, logic, and other guidelines. It is done to find errors in the code, ensuring that the code is developed as per the specifications. Code inspection falls under verification as it does not involve the actual execution of the code.

Requirements specification review:

Categorization: Verification

Reason: Requirements specification review is a static testing technique to ensure that the requirements are correctly captured, unambiguous, and complete. It verifies whether the documented requirements meet the stakeholder's expectations. It falls under verification as it aims to verify if the product is being built correctly according to the specified requirements.

Validation:

Black Box Testing:

Categorization: Validation

Reason: Black Box Testing is a dynamic testing technique where the functionality of the software is tested without looking at the internal code structure. It is used to validate the actual behavior of the software against the expected behavior. Black Box Testing falls under validation as it verifies whether the software meets the user's requirements.

Acceptance Testing:

Categorization: Validation

Reason: Acceptance Testing is performed to ensure that the software system meets the business requirements and is acceptable for delivery to the end-users. It falls under validation as it validates whether the software is acceptable to the end-users or stakeholders, i.e., it confirms that the product is 'right' for the intended purpose.