EXPERIMENT NO.:07

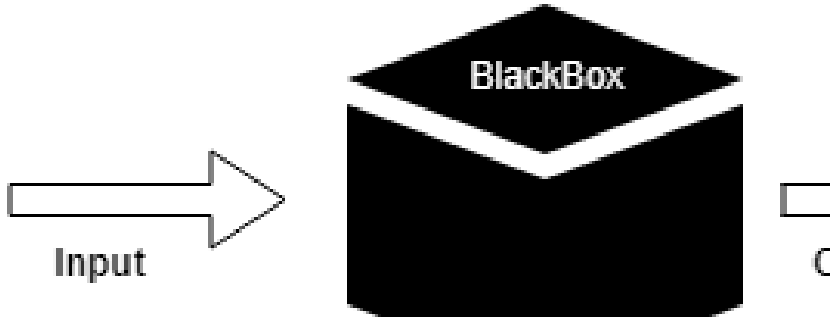
Date of Performance:

Date of Submission:

Aim: Write test cases for black box testing

Software Used: Selenium/GitHub/Jira

Theory:- Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.
In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

White Box Testing

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

The developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

Test procedure

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality.

It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique.

Test cases

Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications. For the testing, the test designer selects both positive test scenario by taking valid input values and adverse test scenario by taking invalid input values to determine the correct output. Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team; there is not any involvement of the development team of software.

Techniques Used in Black Box Testing

Decision Table Technique	A structured method for representing complex business rules, where different conditions lead to various outcomes. This technique helps visualize scenarios, ensuring all possible combinations of inputs are considered in testing. By organizing rules into a table format, testers can easily identify and verify the expected behavior of the system.
Boundary Value Technique	This technique emphasizes testing at the edges of input ranges where errors are most likely to occur. It involves selecting test cases at and just outside the boundaries of valid input ranges to ensure that the system can handle extreme values appropriately. Boundary testing helps uncover issues that may not be apparent in typical input values.
State Transition Technique	This method focuses on the different states an application can be in and how it transitions between these states based on user actions or system events. State transition testing helps verify that the application behaves correctly under various conditions and maintains the correct state after transitions. By modeling states and transitions, testers can identify potential issues in state management.

All-pair Testing Technique	All-pair testing, or pairwise testing, is a technique that focuses on testing all possible pairs of input parameters. It reduces the number of test cases while ensuring adequate coverage by identifying defects caused by the interactions between input parameters. This method is particularly useful when the number of input combinations is large, as it allows testers to efficiently find bugs that arise from specific combinations of inputs.
Cause-Effect Technique	This technique identifies the relationship between causes (input conditions) and effects (expected outputs), allowing testers to derive test cases based on business rules. By mapping out these cause-effect relationships, you can ensure that the application behaves as expected under various input conditions. This approach helps uncover logical errors and ensures that all rules are tested.
Equivalence Partitioning Technique	Equivalence partitioning divides input data into valid and invalid partitions to reduce the total number of test cases while still achieving adequate coverage. By selecting representative values from each partition, testers can ensure that the system behaves correctly across different categories of input without needing to test every possible value. This method helps identify defects in specific input ranges efficiently.
Error Guessing Technique	This technique relies on the tester's intuition and experience to identify areas where defects are likely to occur. It often complements other testing methods by focusing on known problematic areas or previous issues. Testers leverage their knowledge of similar applications or user behaviors to create targeted tests that explore high-risk scenarios.
Use Case Technique	Use case testing focuses on real-world scenarios based on user interactions and workflows. It helps validate that the application meets user requirements and behaves as expected throughout typical user journeys. By creating test cases that align with actual use cases, testers can ensure comprehensive coverage of critical functionalities and identify any gaps in user experience.

Conclusion: Thus we have written test cases for black box testing .

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	