EXPERIMENT NO.:07

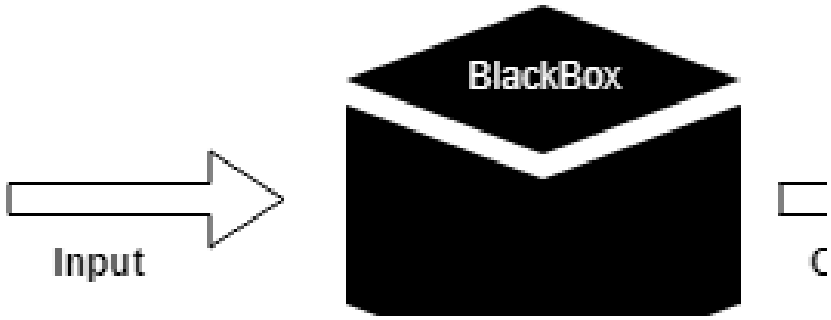
Date of Performance:

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	This technique focuses on defining different user types (tourists and locals) and their actions (viewing attractions or booking events). For example, when a tourist views attractions, the website should display a list of local attractions. However, if a user tries to book an event without registering, the system should prompt them to register first.
Boundary Value Technique	This approach tests input limits, ensuring the website correctly handles edge cases. For instance, testing the "Event Title" field by inputting the maximum character limit of 50 should succeed, while 51 characters should fail. Similarly, checking the minimum characters in the "Attraction Description" field reveals that 1 character is valid, while 0 is not.
State Transition Technique	This technique examines how the website transitions between different states based on user actions. For example, when a user clicks "Events" from the home page, they should be taken to the Events Page. If they click on an event, they should see the Event Details Page, confirming the proper flow of navigation.
All-pair Testing Technique	This method tests various combinations of parameters to ensure comprehensive coverage. By considering user type (local vs. tourist), action (view vs. book), and device type (mobile vs. desktop), we can verify that both locals and tourists receive the correct output, regardless of the device they use.

Cause-Effect Technique	This technique maps causes to expected outcomes. For example, if a user selects a local event, the website should display the event details. If a user attempts to book an event without an account, they should be redirected to the registration page, ensuring logical consistency in functionality.
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 maps causes to expected outcomes. For example, if a user selects a local event, the website should display the event details. If a user attempts to book an event without an account, they should be redirected to the registration page, ensuring logical consistency in functionality.
Use Case Technique	This approach is centered around real-world user scenarios. For example, when a user wants to view local attractions, they should navigate from the home page to the attractions section, which should display a comprehensive list. Similarly, for event registration, users should be able to complete the booking process and receive a confirmation message.

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

Sign and Remark:

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