EXPERIMENT NO.:07

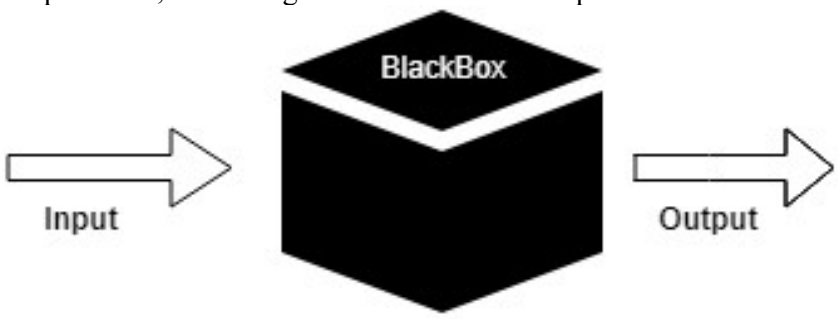
Date of Performance:

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.



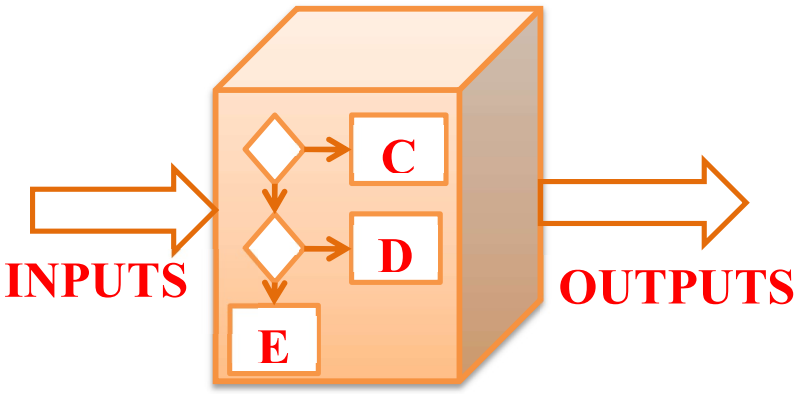
Generic steps of black box testing

- 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	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
Boundary Value Technique	Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.

State Transition Technique	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
Cause-Effect Technique	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result. It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

TEST CASE ID	**TEST CASE TITLE**	**TEST SCENARIO**	**INPUT DATA**	**EXPECTED RESULT**
TC-001	Validate Service Selection	Test selecting different services	Select "Ironing" from the service menu	Service is correctly selected, and price is displayed.
TC-002	Check Pickup Scheduling	Verify pickup scheduling options	Schedule pickup for tomorrow at 10 AM	Pickup is scheduled with a confirmation message.
TC-003	Validate Payment Processing	Ensure payment process works	Pay using UPI or card	Payment is processed successfully, and receipt is generated.

TC-004	Track Order Status	Test tracking order throughout	Track an existing order with ID	The app displays current status (e.g., "In Progress").
TC-005	Test Notifications	Check SMS/Email notifications	Complete an order	User receives order completion notification via SMS/Email.
TC-006	UI Consistency	Ensure interface consistency	Access app in different screen sizes	Layout remains consistent with no visual issues.
TC-007	Validate Error Handling	Test form validation	Submit an incomplete service form	Error message displays instructing the user to fill required fields.
TC-008	Performance Check	Assess response times	Load app during peak hours	The app responds within acceptable time limits (e.g., < 5 seconds).
TC-009	Cross-Platform Testing	Check app on different devices	Access app on Android and iOS	App functions correctly on both Android and iOS devices.
TC-010	Usability Feedback	Gather user feedback on app usage	Conduct user survey after usage	Positive feedback regarding ease of use and navigation.

Conclusion: Thus studied black box testing for Web Translating Tool.

Sign and Remark:

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

EXPERIMENT NO.:08

Date of Performance:

Date of Submission:

Aim: Write test cases for white box testing

Software Used: **: Selenium/GitHub/Jira**

Theory:-

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.

White box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, 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.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Test Case ID	Test Case Title	Description	Expected Result
WTC-001	Ironing Path Coverage	Ensure all ironing modes are tested on different fabric types	All fabrics are ironed without damage
WTC-002	Validate Fabric Detection Logic	Validate logic for detecting garment type for optimal temperature settings	Correct temperature set based on fabric type without errors
WTC-003	Conditional Heat Adjustment Testing	Test conditional statements in heat adjustment based on fabric	The correct temperature is displayed based on user input or fabric detection
WTC-004	Input Validation Logic	Check input validation for custom temperature settings	Proper error messages displayed for invalid temperature inputs

WTC-005	Performance Profiling	Assess performance of machinery under high workload	Machinery operates within acceptable limits without overheating
WTC-006	Security Testing	Test for data security vulnerabilities in customer data handling	Customer data remains secure and encrypted
WTC-007	Loop and Iteration Testing	Validate loops in automated ironing routines	Machinery handles multiple iterations without performance degradation
WTC-008	Resource Management	Check energy usage during peak operations	Energy consumption remains within optimal limits without wastage
WTC-009	Exception Handling Verification	Verify exception handling for unavailable machinery or parts	System handles errors gracefully with fallback notifications to users

Conclusion: Thus studied white box testing for Web Translating Tool.

Sign and Remark:

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

EXPERIMENT NO.:09

Date of Performance:

Date of Submission:

Aim: Preparation of Risk Mitigation, Monitoring and Management plan (RMMM.)

Software Used: Ms Word

Theory:-

RMMM Plan :

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis. After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

Risk Mitigation :

It is an activity used to avoid problems (Risk Avoidance).

Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

Risk Monitoring :

It is an activity used for project tracking.

It has the following primary objectives as follows.

To check if predicted risks occur or not.

1. To ensure proper application of risk aversion steps defined for risk.
2. To collect data for future risk analysis.
3. To allocate what problems are caused by which risks throughout the project.

Risk Management and planning :

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

Steps for Risk Management

1. Identify possible risks and recognize what can go wrong

2. Analyse each risk to estimate the probability that it will occur and the impact (i.e., damage) that it will do if it does occur

3. Rank the risks by probability and impact. Impact may be negligible, marginal, critical, and catastrophic.

4. Develop a contingency plan to manage those risks having high probability and high impact

Risk Table

Risks ID	**Risks**	**Category**	**Probability**	**Impact**	**Risk Management**
R001	Delays in pickup/dropoff scheduling	Operational	Medium	2 (Critical)	Implement real-time scheduling with buffer times;Integrate with traffic prediction tools.
R002	Damage to customer clothes	Technical	Low	2 (Critical)	Use advanced fabric sensing technologies; develop SOPs for handling different materials.Develop a standardized
R003	Insufficient ironing quality	Quality	Medium	3 (Marginal)	quality checklist; conduct periodic quality inspections.
R004	System security	Vulnerabilities Security	Low	1(Catastrophic)	Conduct regular security audits; implement data encryption and secure customer information.
R005	Performance degradation of machinery	Performance	Medium	2 (Critical)	Schedule periodic maintenance; set alerts for performance thresholds.
R006	Inaccurate garment identification	Content	Medium	3 (Marginal)	Utilize computer vision technology; involve manual verification for rare cases.
R007	Lack of trained personnel for garment handling	Resource	High	3 (Marginal)	Develop training sessions for personnel; allocate budget for specialized garment handling courses.

R008	Inadequate handling of fragile materials	Technical	Low	2 (Critical)	Use sensors to detect material fragility; perform careful heat control adjustments.
R009	User resistance to automation	Human Factors	Medium	3 (Marginal)	Provide clear information about automation benefits; offer manual options for users.
R010	Insufficient staff training on new technologies	Organizational	Medium	3 (Marginal)	Develop a comprehensive training program; schedule regular knowledge-sharing sessions.

Conclusion: Thus studied Preparation of Risk Mitigation, Monitoring and Management plan (RMMM.)

Sign and Remark:

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