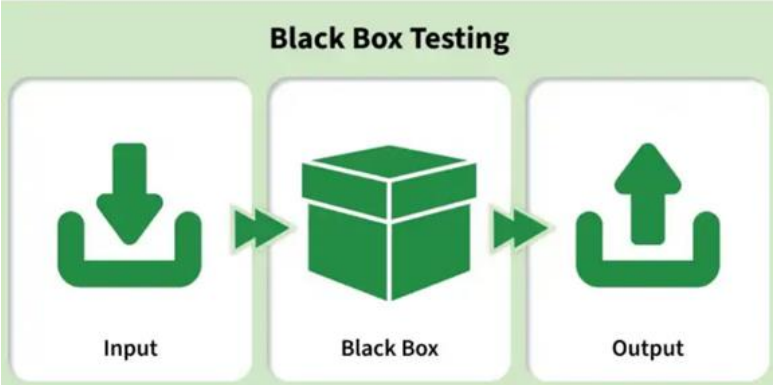
UNIT II – LEVELS OF TESTING

Black Box Testing: Requirements based Testing – Positive and Negative Testing – Boundary Value Analysis – Decision Tables – State Based Testing – Compatibility Testing – User Documentation Testing – Domain Testing. Levels of Testing: System and Acceptance Testing.

2.1 Black Box Testing

Black Box Testing is a software testing technique that verifies the functionality of an application without examining its internal code or implementation. Test cases are designed based on requirements and specifications, focusing on validating inputs and expected outputs. The tester evaluates whether the system behaves correctly according to user requirements and can perform this testing at various levels, including functional, system, and acceptance testing.

- Evaluates software functionality without knowledge of internal code.
- Test cases are derived from requirements and specifications.
- Applicable at functional, system, and acceptance testing levels.



Black Box Testing Architecture:

Black Box testing Architecture describes how testing is performed without access to the application's internal code. The tester validates the system by providing inputs and checking outputs against expected results based on requirements.

- The tester interacts only with the system's inputs and outputs.
- Internal code and implementation details remain hidden.
- Testing is based on requirements, specifications, and user scenarios.

Building Blocks of Black Box testing Architecture

- Input Layer: Contains user inputs, test data, and external requests provided to the system.
- System Under Test (SUT): The application being tested, where processing and business logic are performed.

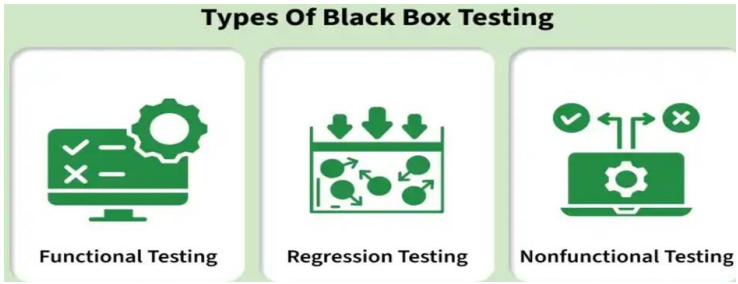
- Output Layer: Produces the system's responses, results, messages, or screen changes.
- Expected Result (Test Oracle): The predefined correct output used to verify actual results.
- Test Case Repository: A collection of test cases, test scenarios, and test data used during testing.

Key Principles of Black Box Testing

- Input-Output Validation: Testers provide inputs and check whether the outputs match the expected results without knowing the internal code.
- Requirements-Driven Design: Test cases are created from requirements and user needs to ensure proper functionality and coverage.

Types Of Black Box Testing

Testing an application without knowing its internal code or structure following are the various.



Functional Testing

Functional Testing is a testing method used to confirm that all software functions operate correctly according to the specified requirements. It focuses on validating the system's behavior by providing inputs and comparing the actual outputs with the expected results, without considering the internal code structure.

Regression Testing

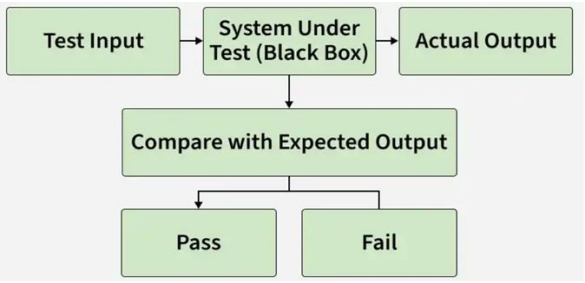
Regression Testing is performed to ensure that modifications, updates, or bug fixes in the software have not caused new defects or affected the existing functionality of the application.

Non-Functional Testing

Non-Functional Testing is used to evaluate the quality attributes of software, such as performance, usability, security, scalability, reliability, and stability, to ensure the system meets non-functional requirements before deployment.

Workflow of Black Box Testing:

The Black Box Testing workflow includes the following step by step process:



Black Box Testing Process

- Provide Test Input: The tester enters valid, invalid, and boundary inputs such as usernames, passwords, or form data.
- System Processing: The application processes the inputs while its internal code and logic remain hidden.
- Generate Actual Output: The system produces a response, such as a success message or an error message.
- Validate Results: The actual output is compared with the expected output defined in the requirements.
- Determine Test Status: The test is marked as Pass if the outputs match; otherwise, it is marked as Fail.

Example: Login Functionality

- Valid Credentials: Enter correct username and password → Login is successful → Pass
- Invalid Credentials: Enter incorrect password → Appropriate error message is displayed → Pass
- Empty Fields: Leave username or password blank → Validation message is shown → Pass (if handled correctly)
- Tools Used for Black Box Testing
- Playwright: Automation tool for testing web applications across multiple browsers.
- Selenium 4: Open-source tool used for automating web application testing.
- Katalon Studio: Testing tool for web, mobile, and API applications with built-in reporting features.
- Cypress: Fast testing framework designed for modern web applications.
- Testim: AI-based automation tool that improves test stability and reduces maintenance.
- Mabl: AI-powered test automation platform used to enhance test reliability.

Advantages of Black Box Testing

- No Code Knowledge Required: Testers can perform testing without understanding the source code.
- Suitable for Large Systems: Effective for testing large and complex applications.
- User Perspective Testing: Validates software behavior from the end-user's viewpoint.
- Easy Test Case Design: Test cases are simple to create and maintain.
- Requirement Validation: Helps identify missing, incorrect, or unclear requirements.

Disadvantages of Black Box Testing

- Repeated Testing: Similar test cases may be executed multiple times.
- Dependent on Requirements: Test cases cannot be created effectively if requirements are incomplete.
- Difficult with Large Input Sets: Testing multiple input combinations can be challenging.
- Root Cause Not Identified: Does not help determine the exact cause of defects.
- Internal Logic Not Tested: Intermediate processing and code paths remain unverified.
- Time-Consuming: Testing large amounts of data can require significant time and effort.

Difference Between Black Box and White Box Testing

Sr.No	Black Box Testing	White Box Testing
1	The internal working of the software is not known to the testers.	The internal working of the software is known to the testers.
2	It only deals with the inputs fed to the software and the outputs generated from them.	It deals with the independent code snippets, algorithms, methods, functions etc.
3	It only verifies the software functionalities and if they are working as per the user requirements.	It verifies the internal working of the software.
4	It does not require technical skills and programming knowledge of the testers.	It requires technical skills and programming knowledge of the testers.
5	It uses methods like equivalence partitioning, boundary value analysis, error guessing etc.	It uses methods like statement coverage, boundary value analysis, data flow, control flow etc.
6	It mainly focuses on the functional periphery of the software.	It mainly focuses on the unit, integration, and system level testing of the software.

2.1 Requirements-Based Testing

Requirements-Based Testing is a testing technique used to verify that the software is developed according to the specified requirements and behaves as expected. Test cases are created directly from the requirements to ensure that all features and functions work correctly.

- Verifies that the software meets the given requirements and specifications.
- Ensures that each requirement is properly implemented and tested.
- Test cases are designed based on requirement documents.
- Helps identify missing, incorrect, or incomplete functionality.
- Ensures the software satisfies user needs and expected outcomes.

Types of Requirements in Software Development

- **Functional Requirements:** Describe the specific functions, features, and operations that the software must perform.
- **Non-Functional Requirements:** Define quality attributes such as performance, security, reliability, and usability.
- **Business Requirements:** Specify the business objectives and expected benefits that the software should achieve.
- **User Requirements:** Represent the needs, expectations, and preferences of end users.
- **System Requirements:** Define the technical and environmental conditions required for the software to operate effectively.
- **Stakeholder Requirements:** Capture the expectations and needs of all stakeholders involved in the project.
- **Interface Requirements:** Specify how the software communicates and exchanges data with external systems or applications.

Principles of Requirement-Based Testing

- Traceability: Establishes clear links between requirements and test cases for easy tracking.
- Early Engagement: Involves testing activities early to validate and clarify requirements.
- Validation and Verification: Ensures the software meets specified requirements and functions correctly.
- Adaptability: Allows testing to accommodate requirement changes without affecting objectives.
- Coverage: Ensures all identified requirements are thoroughly tested.
- Documentation: Maintains accurate records of requirements, test cases, and test results.

Advantages of Requirements-Based Testing

- **Ensures Requirement Compliance:** Verifies that the software is developed according to customer requirements.
- **Early Defect Detection:** Identifies requirement-related defects in the early stages of development.
- **Improves Traceability:** Maintains clear links between requirements, test cases, and defects.
- **Supports Better Resource Allocation:** Helps prioritize testing efforts on high-risk requirements

Disadvantages of Requirements-Based Testing

- **Depends on Requirement Quality:** Incomplete or incorrect requirements can lead to ineffective test cases.
- **Limited Testing Scope:** Important scenarios may be missed if they are not clearly specified in the requirements.
- **Time-Consuming Documentation:** Requires extensive documentation and traceability of test cases.
- **Difficult to Manage Changes:** Requirement changes can increase testing effort and resource usage.
- **Affected by Ambiguity:** Unclear requirements can negatively impact the entire testing process.

Requirements Traceability Matrix (RTM)

A **Requirements Traceability Matrix (RTM)** is a document that links **requirements with test cases**. It ensures that all requirements are tested properly and helps improve software quality.

- Links requirements with test cases
- Tracks testing progress
- Ensures all requirements are tested
- Reduces missed or untested requirements

Components of Requirements Traceability Matrix

- **Requirement ID:** It is a unique identifier assigned to each requirement to easily track and manage it throughout the testing process.
- **Test Case ID:** It is a unique identifier of the test cases created to validate a specific requirement.
- **Test Scenario:** It describes the high-level condition or situation that needs to be tested before writing detailed test cases.

- **Test Case Id :** is the unique number/code used to identify the test case that verifies a particular requirement.
- **Test Execution Status:** It shows the current result of test cases such as pass, fail, or not executed.
- **Defect ID:** It is the unique identifier of a bug linked to a failed test case for tracking and resolution purposes.

Example:

Req. ID	Requirement	Priority	Test Case ID	Test Case	Expected Result	Test Result
R01		High	TC01			
R02		Medium	TC02			
R03		High	TC03			
R04		Low	TC04			