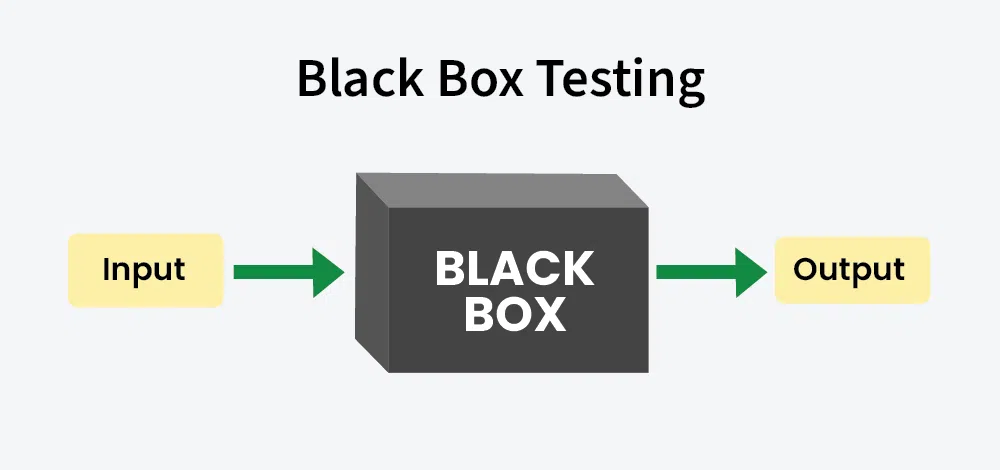
What is Black Box Testing?

Black-Box testing is a type of software testing in which the tester is not concerned with the softwar’s internal knowledge or implementation details about rather focuses on validating the functionality based on the provided specifications or requirements.



Types of Black Box Testing:

1. Functional Testing.
2. Non-funcational Testing.

What is Functional Testing?

Functional testing is a type of software testing that ensures each function of the application works as intended, based on the requirements and specifications. It focuses on validating the functionality of the software without examining the source code. Test inputs are provided, and the actual outputs are compared with the expected results. This testing covers areas like the user interface, APIs, database, security, and overall functionality, and it can be done either manually or through automation.

Types of Functional Testing:

1. **Unit Testing**: Tests individual components or units of the software to ensure they work as expected in isolation.
2. **Integration Testing**: Verifies that different modules or components of the software work together as expected when integrated.
3. **System Testing**: Tests the complete and integrated software system to ensure it meets the specified requirements.
4. **Smoke Testing**: A basic test to check if the application’s critical features are working after a new build or update.
5. **Sanity Testing**: A quick check to verify if a specific functionality or bug fix is working as intended, usually after minor changes.
6. **Regression Testing**: Ensures that new code changes do not negatively impact the existing functionality of the software.
7. **User Acceptance Testing (UAT)**: Confirms that the software meets the business requirements and is acceptable to the end users.
8. **Alpha Testing**: Internal testing conducted by the development team to identify bugs before releasing the software to external testers.
9. **Beta Testing**: Conducted by a limited group of users outside the development team to gather feedback and find issues before the final release.
10. **Interface Testing**: Validates that the interactions between different software modules or systems work correctly, including external systems or APIs.
11. **Ad-hoc Testing**: Unstructured testing where the tester checks the software without following any specific test cases or plans, often based on intuition.
12. **Exploratory Testing**: Testers explore the software, often in an unscripted manner, to discover defects by using their knowledge and experience.

What is Non-functional Testing?

Non-functional testing is a type of software testing that evaluates the non-functional aspects of an application, such as performance, scalability, security, usability, and reliability, rather than its functionality. It ensures that the software meets the specified non-functional requirements, such as handling load, performing efficiently, and providing a good user experience.

Types of Non-functional Testing:

1. **Performance Testing**: Evaluates how the software performs under various conditions, including response time, throughput, and resource usage.
2. **Load Testing**: Assesses the system’s ability to handle a specific load or number of users concurrently.
3. **Stress Testing**: Tests the system under extreme conditions to see how it handles stress or fails under overload.
4. **Scalability Testing**: Determines how well the application can scale up (handle increased load) or scale out (handle growth in capacity).
5. **Usability Testing**: Evaluates the user interface and overall user experience to ensure the software is intuitive and easy to use.
6. **Compatibility Testing**: Verifies that the software works across different environments, such as different browsers, operating systems, devices, or network configurations.
7. **Security Testing**: Assesses the software’s security mechanisms, identifying vulnerabilities and ensuring the software is protected from malicious attacks.
8. **Recovery Testing**: Checks how well the software can recover from crashes, hardware failures, or other catastrophic events.
9. **Reliability Testing**: Measures how dependable the software is over time under normal usage.
10. **Installability Testing**: Ensures the software can be installed and configured correctly in the target environment.
11. **Localization Testing**: Verifies the software works correctly in different languages, regions, or cultural settings.
12. **Compliance Testing**: Ensures the software adheres to legal, regulatory, and other compliance standards.