KASPER
ANALYTICS

kasperanalytics.com

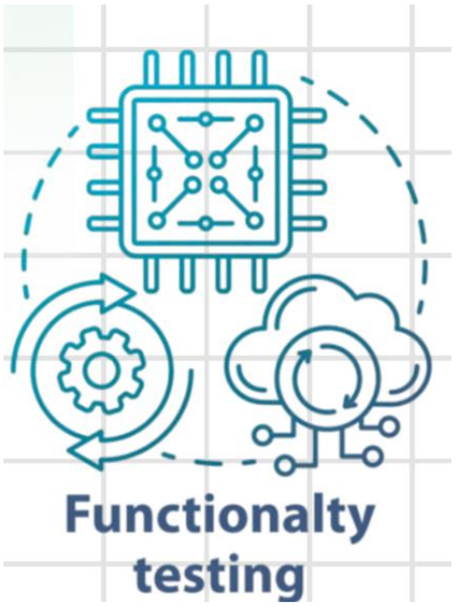
+918130877931

FUNCTIONAL TESTING

hr@kasperanalytics.com

kasper-analytics

Functional Testing Types



- Unit Testing
- Integration Testing
- Regression Testing
- Sanity Testing
- Smoke Testing
- End-to-End Testing
- User Acceptance Testing
- White Box Testing
- Black Box Testing
- Interface Testing

Unit Testing

- Unit testing is a type of software testing which is done on an individual unit or component to test its corrections.
- Typically, Unit testing is done by the developer during the application development phase. Each unit in unit testing can be viewed as a method, function, procedure, or object.
- Developers often use test automation tools such as NUnit, Xunit, JUnit, Chai, Jest for the test execution.

Integration Testing

- Integration testing is a type of software testing where two or more modules of an application are logically grouped and tested as a whole. This type of testing focuses on finding the defect in interface, communication, and data flow among modules. A Top-down or Bottom-up approach is used while integrating modules into the whole system.

Integration Testing Examples

- This type of testing is done by integrating modules of a system or between systems. For example, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

Regression Testing

- Any new change or feature added to the software can wreck its existing functionalities. Regression testing is performed every time alterations are made to check for the software's stability and functionalities. Due to its work-intensive nature, regression testing is often automated.
- Example: A food delivery app added a function to help users add multiple promotions on top of each other. A regression test needs to be done to make sure the checkout and payment process is not affected.

Sanity Testing

- Similar to regression testing, sanity testing is conducted for a new build with minor bug fixes, or new code added. If rejected in the sanity testing phase, the build will not proceed to further testing. While regression testing checks the entire system after alterations, sanity testing targets specific areas that are affected by the new code or bug fixes only.
- Example: On an e-commerce webpage, users cannot add a particular product to their cart even when the stock is available. After the issue is fixed, sanity testing is performed to ensure that the “add to cart” function is indeed working.

Smoke Testing

- When a new build is completed, it is handed to the QAs for smoke testing. In this phase, only the most critical and core functionalities are tested to ensure that they yield the intended results. As an early-stage acceptance test, smoke testing adds a verification layer to determine whether or not the new build can proceed to the next stage or needs re-work.

Smoke Testing Example

- Example: A utility company built an app with the function to report outages in customers’ homes. This function reports the address and other relevant information as well as notifies the homeowner when a dispatcher is on the way to help. Smoke testing will validate this feature on a fundamental level to ensure that when an outage is reported, the correct information is sent so a dispatcher can be there on time.

End to End Testing

- It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.
- For example, a tester is testing a pet insurance website. End-to-end testing involves testing buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users' accounts, updating user address information, and receiving order confirmation emails and policy documents.

User Acceptance Testing

- User acceptance testing (UAT), also called application testing or end-user testing, is a phase of software development in which the software is tested in the real world by its intended audience.
- User acceptance testing validates the testing done at the end of the development cycle. It is typically completed after unit testing, quality assurance, system testing and integration testing. The software may undergo other testing phases and be completely functional but might still not meet its requirements if it is not well received by its intended users.

White Box Testing

- White Box Testing, also known as Clear Box Testing or Glass Box Testing, is a software testing technique where the internal structure, design, and implementation of the software are known to the tester. This approach involves testing the software's code, algorithms, logic, and data flow to ensure that the internal operations are functioning as expected.

Key Aspects of White Box Testing

- Code Coverage: Ensures that all paths, branches, and lines of code are tested.
- Control Flow Testing: Verifies the flow of control through the program's code.
- Data Flow Testing: Examines the flow of data through the software.
- Unit Testing: Involves testing individual components or units of code

Black Box Testing

- Black Box Testing is a software testing method where the tester evaluates the functionality of an application without knowing its internal code, structure, or implementation details. The primary focus is on the inputs and outputs of the software system to ensure it behaves as expected, based on requirements and specifications.

Key Characteristics of Black Box Testing

- No Knowledge of Internal Code: Testers do not need to understand the software's internal workings.
- Focus on Functionality: Testing is based on user interactions, inputs, and expected outputs.
- Test Cases Derived from Specifications: Test cases are created based on functional requirements and user stories.
- Used for Various Levels of Testing: Applicable in unit testing, integration testing, system testing, and acceptance testing.

Interface Testing

- Interface Testing is a type of software testing that focuses on verifying the interactions between different software components or systems. The main objective is to ensure that the interfaces between these components work correctly and efficiently, allowing them to communicate and exchange data as intended.

Key Aspects of Interface Testing

Communication Protocols:

- Testing the communication methods between components, such as APIs, web services, or data transfer protocols.
- Ensuring that data is transmitted correctly and securely across the interface.

Data Validation:

- Verifying that the data sent and received through the interface is accurate, complete, and properly formatted.
- Checking for any data corruption, loss, or unexpected changes during transmission.

Error Handling:

- Ensuring that the interface can handle errors gracefully, such as incorrect data formats, communication failures, or unexpected inputs.
- Verifying that appropriate error messages or codes are returned when something goes wrong.

Performance:

- Measuring the efficiency and speed of communication between components through the interface.
- Ensuring that the interface can handle the expected load and respond within acceptable time limits.

Security:

- Verifying that the interface does not expose any vulnerabilities that could be exploited by malicious actors.
- Ensuring that sensitive data is encrypted or protected during transmission.

KASPER
ANALYTICS

Need any experience/support on

Hands-on Live Project

and Live Frameworks.

Please
Connect

Contact Number - 8130877931

Email - hr@kasperanalytics.com

LinkedIn - www.linkedin.com/company/kasper-analytics/