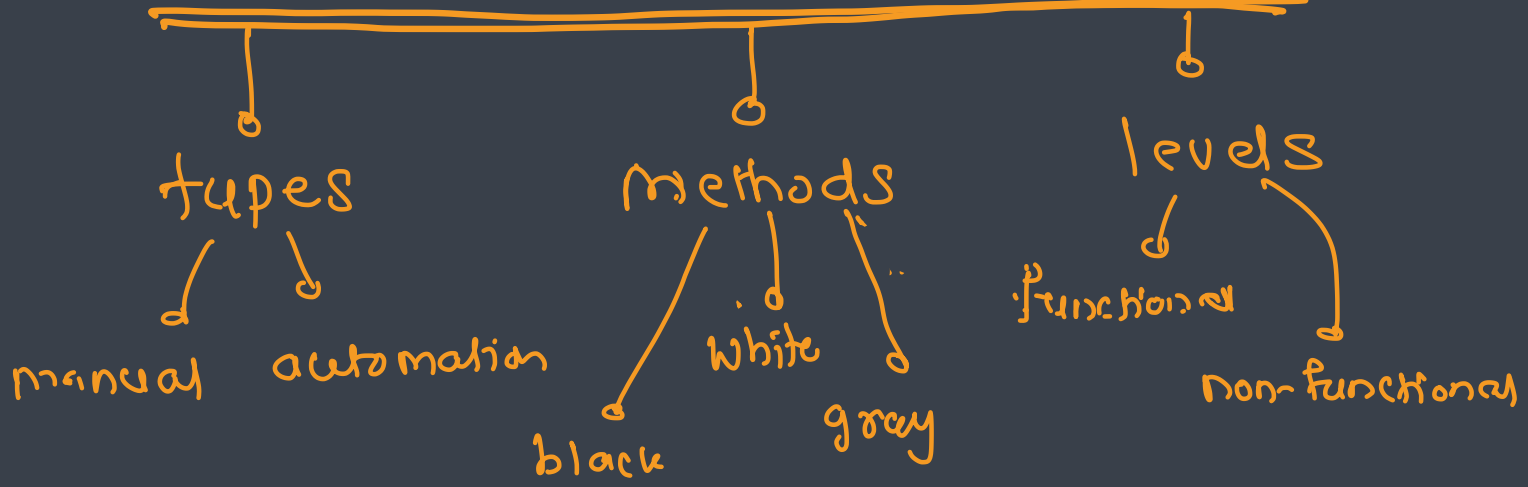
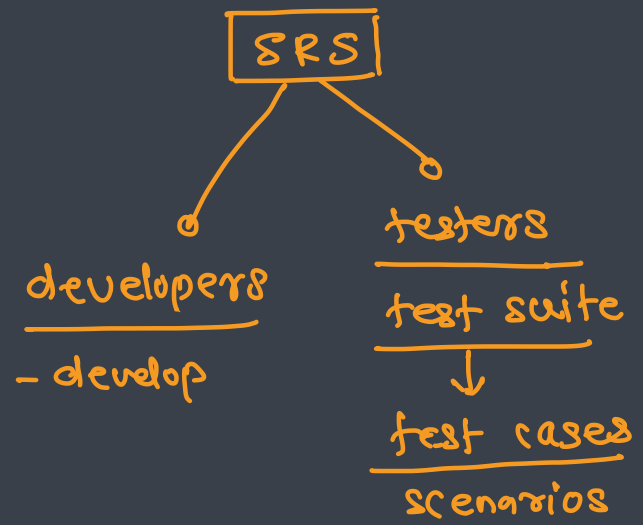
# Application Testing



# What is testing?

- process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not
- executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements
- A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item
- Testing will be done by
  - Software Developer
  - Software Tester
  - Project Lead/Manager
  - End User



# When to Start Testing?

- During the requirement gathering phase, the analysis and verification of requirements are also considered as testing
- Reviewing the design in the design phase with the intent to improve the design is also considered as testing
- Testing performed by a developer on completion of the code is also categorized as testing

→ planning → if all the requirements are captured and analysed
→ defining → if all the requirement are captured in SRS
→ designing → if architecture will be feasible, or are the screens correct
→ building → if the requirements are correctly implement → unit testing
→ testing → if the app has all the requirements from SRS → tester
→ deployment → if the app is working in the respective env

# When to Stop Testing?

- Testing Deadlines
- Completion of test case execution
- Completion of functional and code coverage to a certain point
- Bug rate falls below a certain level and no high-priority bugs are identified
- Management decision

# Verification vs Validation

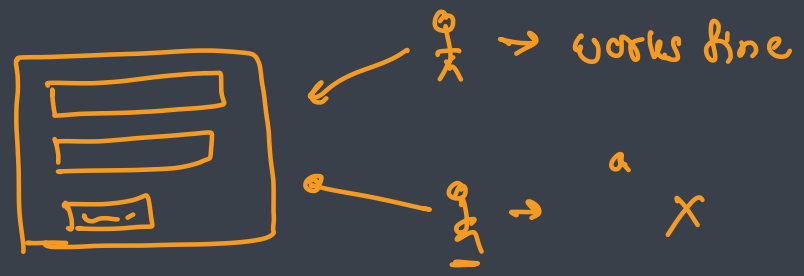
developer

- Addresses the concern: "Are you building it right?"
- Ensures that the software system meets all the functionality
- Takes place first and includes the checking for documentation, code
- Done by developers
- It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software
- It is an objective process and no subjective decision should be needed to verify a software

code analyzer

tester

- Addresses the concern: "Are you building the right thing?"
- Ensures that the functionalities meet the intended behaviour
- Validation occurs after verification and mainly involves the checking of the overall product
- Done by testers
- It has dynamic activities, as it includes executing the software against the requirements
- It is a subjective process and involves subjective decisions on how well a software works



# Quality Assurance vs Quality Control

- QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements
- Focuses on processes and ` rather than conducting actual testing on the system
- Process-oriented activities
- Preventive activities
- It is a subset of Software Test Life Cycle (STLC)

policy makers

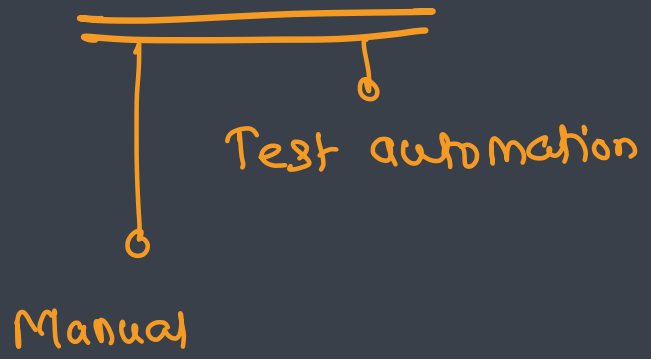
- QC includes activities that ensure the verification of a developed software with respect to documented requirements
- Focuses on actual testing by executing the software with an aim to identify bug/defect through implementation of procedures and process
- Product-oriented activities
- It is a corrective process
- QC can be considered as the subset of Quality Assurance

policy followers

# Checking the testing process

- **Audit** → Internal
  - It is a systematic process to determine how the actual testing process is conducted within an organization or a team
  - It is an independent examination of processes involved during the testing of a software
  - It is a review of documented processes that organizations implement and follow
  - Types: Legal Compliance Audit, Internal Audit, and System Audit
- **Inspection** → External
  - Inspection is a formal evaluation technique in which software requirements, designs, or codes are examined in detail
  - Conducted by a person or a group other than the author to detect faults, violations of development standards, and other problems

# Types



# Manual Testing

- Manual testing includes testing a software manually, i.e., without using any automated tool or any script
- The tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug
- There are different stages for manual testing
  - unit testing
  - integration testing
  - system testing
  - user acceptance testing (UAT)
- Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing

# Automation Testing

Selenium
Cypress
→ web UI

- Also known as Test Automation
- Tester writes scripts and uses another software to test the product
- Involves automation of a manual process
- Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly
- Used to test
  - Regression
  - Performance
  - Stress point
- It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing

# What to Automate?

- It is not possible to automate everything in a software
- The areas at which a user can make transactions such as
  - the login form or registration forms → UI
  - any area where large number of users can access the software simultaneously → Scalability
  - all GUI items
  - connections with databases
  - field validations

# When to Automate?

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users
- Stable software with respect to manual testing
- Availability of time
- Skills

# How to Automate?

Test Driven Development (TDD)

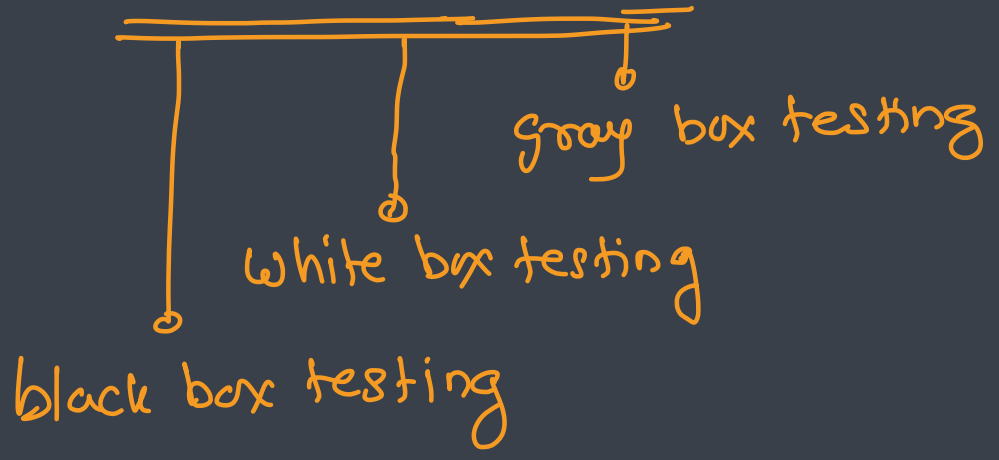
- Identifying areas within a software for automation
  - ① GUI
  - ② dB
- Selection of appropriate tool for test automation → cypress / selenium / Jmeter / LoadRunner
- Writing test scripts → python / JS
- Development of test suits → collection of test cases
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issues

# Software Testing Tools

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LoadRunner
- Visual Studio Test Professional
- WATIR

Jest / Jasmine

# Methods



# Black-Box Testing : cheapest

- The technique of testing without having any knowledge of the interior workings of the application
- The tester is oblivious to the system architecture and does not have access to the source code
- Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon
- Advantages
  - Well suited and efficient for large code segments
  - Code access is not required : outsourcing
  - Clearly separates user's perspective from the developer's perspective through visibly defined roles
  - Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems
- Disadvantages
  - Limited coverage, since only a selected number of test scenarios is actually performed → no testing of architecture, design, algorithms etc
  - Inefficient testing, due to the fact that the tester only has limited knowledge about an application
  - Blind coverage, since the tester cannot target specific code segments or errorprone areas → algorithms / logic

# White-Box Testing

- Detailed investigation of internal logic and structure of the code
- Is also called **glass testing** or **open-box testing**
- A tester needs to know the internal workings of the code
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately
- Advantages
  - As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively
  - It helps in optimizing the code
  - Extra lines of code can be removed which can bring in hidden defects
  - Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing
- Disadvantages
  - Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
  - Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested
  - It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools

# testing data

→ developer : abc@gmail.com / test

→ tester :
- a@ / test
- a@b / test
- abc@gmail.com/test
- —"— / test213
- —"— / Test*123@12

# Grey-Box Testing

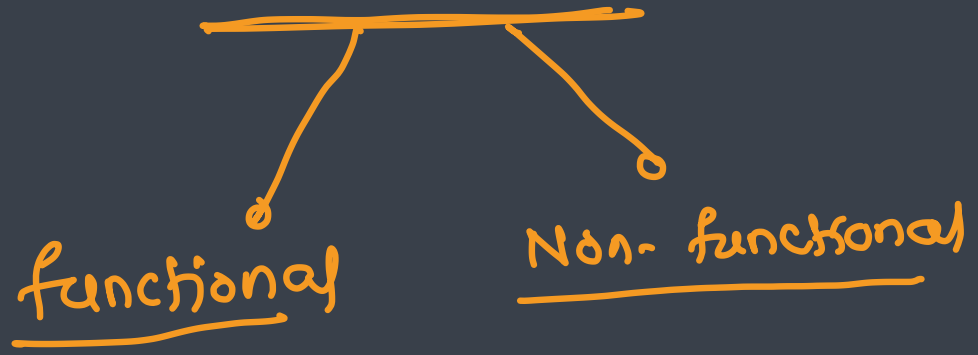
- A technique to test the application with having a limited knowledge of the internal workings of an application
- Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database
- Advantages
  - Offers combined benefits of black-box and white-box testing wherever possible
  - Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications
  - Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling
  - The test is done from the point of view of the user and not the designer
- Disadvantages
  - Since the access to source code is not available, the ability to go over the code and test coverage is limited
  - The tests can be redundant if the software designer has already run a test case
  - Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested

IDL

# Black vs Grey vs White

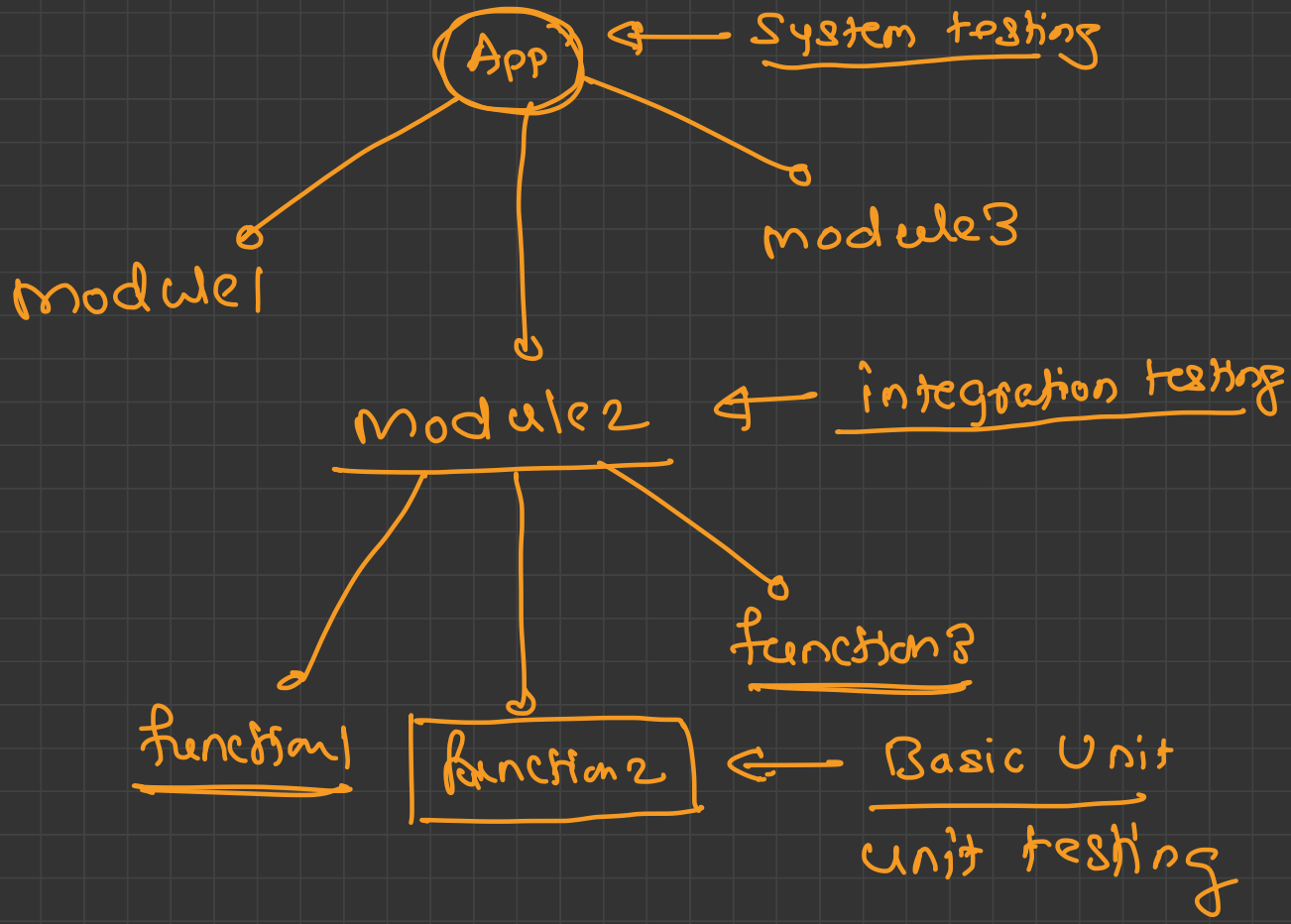
| Black-Box Testing | Grey-Box Testing | White-Box Testing |
|---|---|---|
| The internal workings need not be known | Requires limited knowledge of the internal workings | Requires full knowledge of the internal workings |
| Also known as closed-box testing, data-driven testing, or functional testing | Also known as translucent testing | Also known as clear-box testing, structural testing, or code-based testing |
| Performed by end-users and also by testers and developers | Performed by end-users and also by testers and developers | Normally done by testers and developers |
| Testing is based on external expectations | Testing is done on the basis of high-level database diagrams and DFDs | Tester can design test data accordingly |
| It is exhaustive and the least time-consuming | Partly time-consuming and exhaustive | The most exhaustive and time-consuming type of testing |
| Not suited for algorithm testing | Not suited for algorithm testing | Suited for algorithm testing |
| Only be done by trial-and-error method | Data domains and internal boundaries can be tested | Data domains and internal boundaries can be better tested |

# Levels



# Functional Testing

- This is a type of black-box testing that is based on the specifications of the software that is to be tested
- The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for
- Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements
- Includes
  - Unit Testing
  - Integration Testing
  - System Testing
  - Regression Testing
  - Acceptance Testing
  - Alpha Testing
  - Beta Testing


App
System testing
module1
module2
module3
integration testing
function1
function2
function3
Basic Unit
unit testing

# Functional Testing – Steps

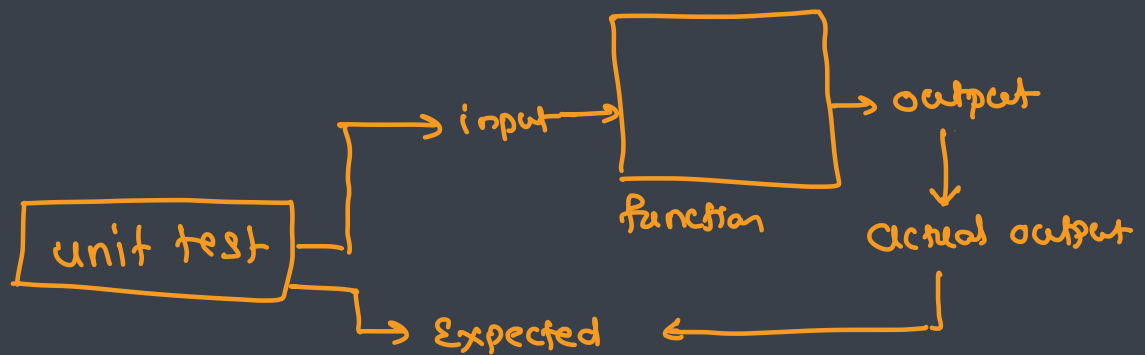
- The determination of the functionality that the intended application is meant to perform
- The creation of test data based on the specifications of the application
- The output based on the test data and the specifications of the application
- The writing of test scenarios and the execution of test cases
- The comparison of actual and expected results based on the executed test cases

# Unit Testing

unit - function

TDD

- Is performed by developers before the setup is handed over to the testing team to formally execute the test cases
- Unit testing is performed by the respective developers on the individual units of source code assigned areas
- The developers use test data that is different from the test data of the quality assurance team
- The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality
- Limitation
  - Testing cannot catch each and every bug in an application
  - It is impossible to evaluate every execution path in every software application
  - There is a limit to the number of scenarios and test data that a developer can use to verify a source code
  - After having exhausted all the options, there is no choice but to stop unit testing and merge the code segment with other units



```
function add (p1, p2) {
    return p1 + p2
}
```

add (10, '20') ⇒ "1020"

Jest / Jasmine

```
const result = add (10,20)
```

actual result

[ p1 = 10, p2 = 20, result must be 30 ]

Expected result

```
if (result == = 30) {
    console.log ("success")
} else {
    console.log ("failure")
}
```

unit test case

# Integration Testing : module testing

- Integration testing is defined as the testing of combined parts of an application to determine if they function correctly
- Integration testing can be done in two ways
  - Bottom-up integration testing
    - This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds
  - Top-down integration testing
    - In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter
- bottom-up testing is usually done first, followed by top-down testing

# System Testing

- System testing tests the system as a whole
- Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards
- This type of testing is performed by a specialized testing team (QC)
- Importance
  - is the first step in the SDLC, where the application is tested as a whole
  - The application is tested thoroughly to verify that it meets the functional (SRS) and technical specifications
  - The application is tested in an environment that is very close to the production environment where the application will be deployed
  - System testing enables us to test, verify, and validate both the business requirements as well as the application architecture

# Regression Testing

- The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application
- Importance
  - Minimizes the gaps in testing when an application with changes made has to be tested
  - Testing the new changes to verify that the changes made did not affect any other area of the application
  - Test coverage is increased without compromising timelines
  - Increase speed to market the product

→ Not opening new bugs while fixing older ones

# Acceptance Testing : UAT

- The most important type of testing, as it is conducted by the Quality Assurance Team who will verify whether the application meets the intended specifications and satisfies the client's requirement
- The QA team will have a set of pre-written scenarios and test cases that will be used to test the application
- Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application
- By performing acceptance tests on an application, the testing team will reduce how the application will perform in production.
- There are also legal and contractual requirements for acceptance of the system.

# Alpha Testing



- This test is the first stage of testing and will be performed amongst the teams (developer and QA teams)
- Unit testing, integration testing and system testing when combined together is known as alpha testing
- During this phase, the following aspects will be tested in the application
  - Spelling Mistakes
  - Broken Links
  - The Application will be tested on machines with the lowest specification to test loading times and any latency problems

# Beta Testing

External (Real user) [ production ]

- This test is performed after alpha testing has been successfully performed
- In beta testing, a sample of the intended audience tests the application
- Beta testing is also known as **pre-release testing**
- Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release
- Users will install, run the application and send their feedback to the project team
- Typographical errors, confusing application flow, and even crashes
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users
- The more issues you fix that solve real user problems, the higher the quality of your application will be
- Having a higher-quality application when you release it to the general public will increase customer satisfaction

real end users

# Non-Functional Testing

- Non-functional testing involves testing a software from the requirements which are nonfunctional in nature but important such as performance, security, user interface, etc
- Includes
  - Performance testing
  - Usability Testing
  - Security Testing

# Performance Testing

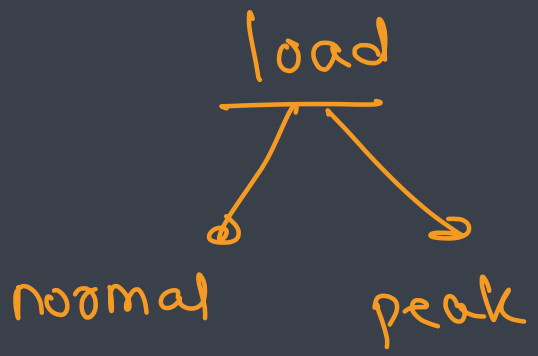
- It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software
- There are different causes that contribute in lowering the performance of a software
  - Network delay
  - Client-side processing
  - Database transaction processing
  - Load balancing between servers
  - Data rendering
- Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects –
  - Speed (i.e. Response Time, data rendering and accessing)
  - Capacity
  - Stability
  - Scalability
- Performance testing can be either qualitative or quantitative and can be divided into
  - Load testing
  - Stress testing

# Load Testing

peak load

- It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data
- It can be done at both normal and peak load conditions
- This type of testing identifies the maximum capacity of software and its behavior at peak time
- Most of the time, load testing is performed with the help of automated tools such as Load Runner, AppLoader etc

Jmeter



# Stress Testing

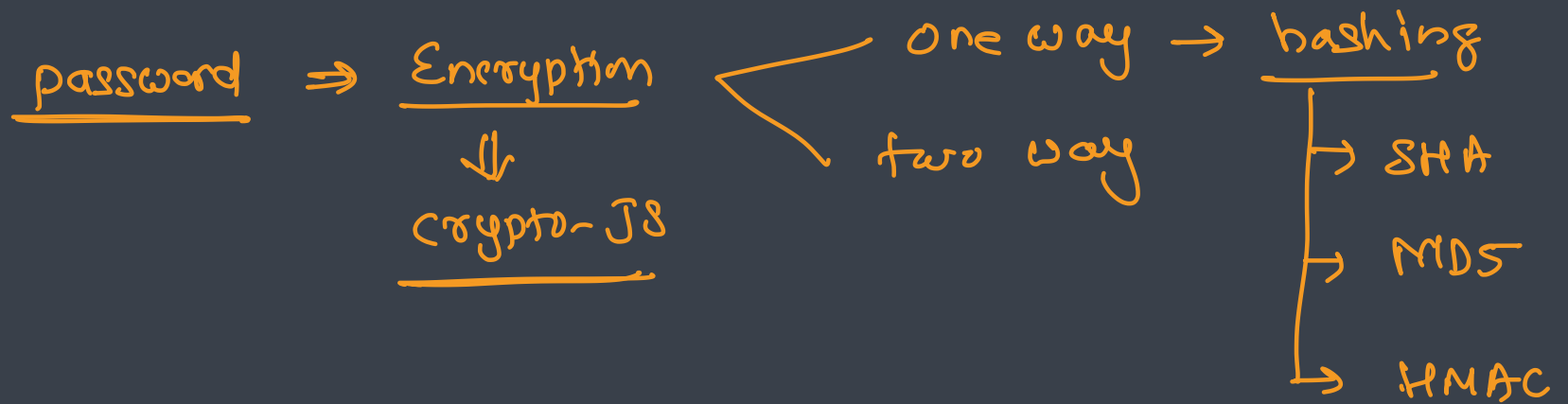
- Stress testing includes testing the behavior of a software under abnormal conditions
- For example, it may include taking away some resources or applying a load beyond the actual load limit
- This testing can be performed by testing different scenarios such as
  - Shutdown or restart of network ports randomly
  - Turning the database on or off
  - Running different processes that consume resources such as CPU, memory, server, etc.

# Usability Testing

- Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation
- Can be defined as
  - efficiency of use
  - learn-ability
  - memory-ability
  - errors/safety
  - Satisfaction
- UI testing can be considered as a sub-part of usability testing.

# Security Testing

- Testing a software in order to identify any flaws and gaps from security and vulnerability point of view.
- Involves
  - Confidentiality
  - Integrity
  - Authentication
  - Availability
  - Authorization
  - Non-repudiation
  - Input checking and validation
  - SQL insertion attacks



# Portability Testing

- Portability testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well
- Following are the strategies that can be used for portability testing
  - Transferring an installed software from one computer to another.
  - Building executable to run the software on different platforms
- Portability testing can be considered as one of the sub-parts of system testing
- Computer hardware, operating systems, and browsers are the major focus of portability testing

# Selenium

# Overview

- Selenium is an open-source and a portable automated software testing tool for testing web application
- It has capabilities to operate across different browsers and operating systems
- Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently
- Tools
  - Selenium IDE
  - Selenium RC (Remote Control)
  - Selenium WebDriver
  - Selenium Grid

# Advantages

- Selenium is an open-source tool
- Can be extended for various technologies that expose DOM
- Has capabilities to execute scripts across different browsers
- Can execute scripts on various operating systems
- Supports mobile devices
- Executes tests within the browser, so focus is NOT required while script execution is in progress
- Can execute tests in parallel with the use of Selenium Grids

# Disadvantages

- Supports only web based applications
- No feature such as Object Repository/Recovery Scenario
- Cannot access controls within the browser
- No default test report generation
- For parameterization, users has to rely on the programming language

# WebDriver

- WebDriver is a tool for automating testing web applications
- It is popularly known as Selenium 2.0
- Interacts directly with the browser and uses the browser's engine to control it
- It is faster, as it interacts directly with the browser
- It can support the headless execution

# Locators

- Element Locators help Selenium to identify the HTML element the command refers to
- Types
  - **identifier** = id Select the element with the specified "id" attribute and if there is no match, select the first element whose @name attribute is id.
  - **id** = id Select the element with the specified "id" attribute.
  - **name** = name Select the first element with the specified "name" attribute
  - **dom** = javascriptExpression Selenium finds an element by evaluating the specified string that allows us to traverse through the HTML Document Object Model using JavaScript. Users cannot return a value but can evaluate as an expression in the block
  - **xpath** = xpathExpression Locate an element using an XPath expression.
  - **link** = textPattern Select the link element (within anchor tags) which contains text matching the specified pattern
  - **css** = cssSelectorSyntax Select the element using css selector