Software Development Models

What is SDLC?

SDLC stands for “Software Development Life Cycle”. It describes the various phases involved in the software development process. The software development life cycle starts with the requirement gathering phase and after that based on the requirements, design specifications are created. The high and low level designs created in the design-specification phase lead to the implementation phase. In the implementation phase, coding is done. This phase leads to a software product that is required to be tested in order to validate the business requirements of the product. After successful testing(product with no high priority bugs), the software is deployed to the client.

The different phases of Software Development Life Cycle are-

- Requirement Gathering
- Design Specification
- Coding/Implementation
- Testing
- Deployment
- Maintenance

Now, let's see the different phases of SDLC in detail.

Requirement Gathering

Requirement gathering is one of the most critical phase of SDLC. This phase marks as the basis of whole software development process. All the business requirements are gathered from the client in this phase. A proper document is made which tells the purpose and the guidelines for the other phases of the life cycle. For example- if we want to make a website for a restaurant. The requirement analysis phase will answer the questions like-

- What type of website is needed – static or dynamic?
- What kind of functionalities are needed by the client?
- Is ordering online facility required?
- Is online payment functionality required? etc

Design Specification

A software design or we can say a layout is prepared in this phase according to the requirements specified in the previous step. In this phase, the requirements are broken down into multiple modules like login module, signup module, menu options on other modules etc. So this design is considered as the input for the next implementation phase.

Implementation

In this phase, the actual development gets started. The developer writes codes using different programming languages depending upon the need of the product. The main stakeholders in this phase are the development team.

Testing

After the completion of the development phase testing begins. Here testers test the software and provide appropriate feedback to the developing team. The tester checks that whether the software developed fulfills the desired requirements of the client that are described in the requirement phase or not. Functional and non-functional testing is performed here before delivering it.

Deployment

After the testing gets completed these product developed gets live and is handed over to the client. Now the client can publish it online and can decide about customer’s access.

Maintenance

In this phase, the maintenance of the software product is taken care of like making changes to the software that are required for the intended functionality of the product over a period of time.

There are various models in software development life cycle depending on the requirement, budget, criticality and various other factors. Some of the widely used SDLC models are-

- Waterfall model
- Iterative model
- Incremental model
- Spiral model
- Agile model

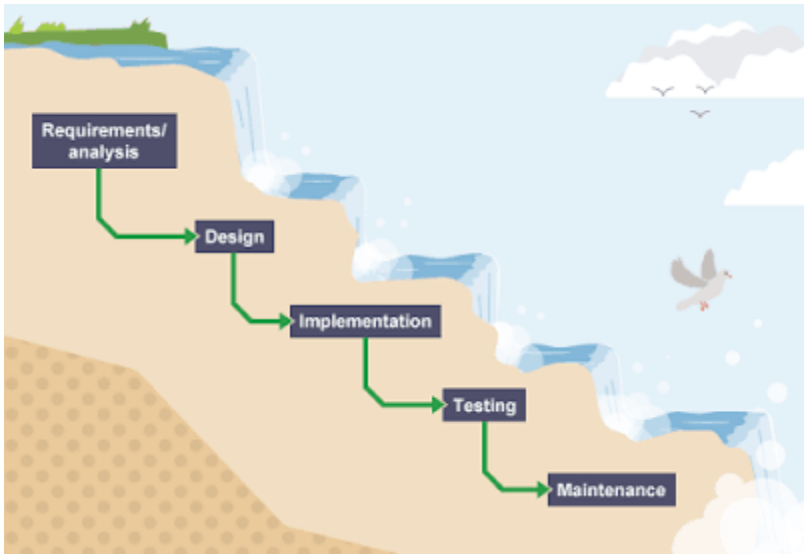
Irrespective of the SDLC model we use, all the models have the above phases in different parts of its lifecycle.

Waterfall Model

Waterfall model is one of the initial models introduced in Software Development Life Cycle. So, it is also known as the "Traditional Model". It is called waterfall model because as the water once falls from the mountain cannot be returned back, in the same way once a phase of development cycle is completed we cannot go back to that phase again. The development life cycle traverses different phases and follows a linear sequence. So, this model is also known as Linear Sequential Model.

It is easy to use and simple to understand model which includes the following phases-

- Requirement Gathering & Analysis
- System Design
- Implementation
- Testing
- Deployment
- Maintenance



In this model, each phase must be completed before the next phase can begin as the outcome of previous phase will act as the input for the next phase.

Requirement Gathering & Analysis

All the possible requirements of the system to be developed are captured in this phase. Once the requirements for the software are gathered it gets freezed means it cannot be changed. Here the requirement feasibility test is done to ensure whether the requirements are testable. Requirement understanding document is delivered to the next stage which describes all the requirements of the client.

System Design

After the completion of requirement gathering phase comes the system Design phase in which all the requirement specifications are studied and a design is created which has both the hardware as well as software requirements. Designs are documented here and high level design document(HLDD) and low level design document(LLDD) are delivered to the next stage which keeps the record of which module should be linked up with which.

Implementation

The outcome of system design phase becomes input of this phase. In this the system is first developed in small programs called units which are integrated in next phase.

Testing

All the units that are developed in the implementation phase are integrated into a system in this phase after the testing of the each unit. After the complete Integration the entire system is tested for any faults or failures.

Deployment

After the functional and the non-functional testing is performed the product is released in the market.

Maintenance

In this phase, the maintenance of the software product is taken care of like making changes to the software that are required for the intended functionality of the product over a period of time.

Advantages of Waterfall Model

- It is simple to understand and follow.
- There are specific deliverables of each phase.
- Tasks are arranged easily as all the stages are clearly defined.
- It is perfectly suitable for short projects where all the requirements are predefined and understood clearly.

Disadvantages of Waterfall Model

- There is uncertainty and high amount of risk, hence not preferred for complex projects.
- Working model cannot be produced until last phase.
- Once you come to the next phase you cannot go back to the previous phase to recheck it.
- Progress cannot be measured within stages.

Application of Waterfall Model

The Waterfall model is This model is applicable for following types of projects-

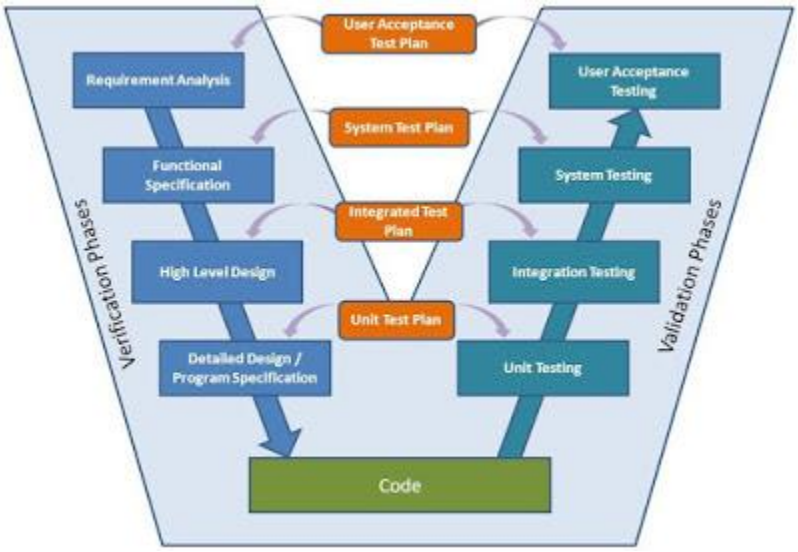
- Short term or smaller projects.
- Projects where requirements are perfectly documented and are clearly understood.
- Projects where frequent changes are not required in the product.

V Model

SDLC

Software Development Life Cycle refers to all the activities that are performed during software development including- requirement analysis, designing, implementation, testing, deployment and maintenance phases. In this tutorial, we will refer to one of the most widely used SLDC model - V model.

V Model



V model is also known as Verification and Validation model. In V model, the testing phase goes in parallel with the development phase. Thus, the testing phase starts right at the beginning of SDLC.

As we can see in the above diagram, the test activities start in parallel with the development activities e.g. during the requirement analysis phase, acceptance-test cases are prepared by the testing team; during system design phase, the system test case are prepared. Similarly, for each phase of development a corresponding QA activity is performed. Later, when the deliverable gets ready, the QA artifacts are used to conduct the testing. Along with that, each phase of development phase is verified before moving to the next phase.

Advantages of V Model

- Each phase of development is tested before moving to next phase, hence there is higher rate of success.
- It avoids defect leakage to the later phases as each phase is verified explicitely.
- The model has clear and defined steps. So, it is easier to implement.
- It is suitable for smaller projects where requirements are fixed.

Disadvantages of V Model

- The testing team starts in parallel with development. Hence, the overall budget and resource usage increases.
- Change in requirement are difficult to incorporate.
- The working model of the software is only available in the later phases of the development.
- It is not suitable for complex and large applications because of its rigid process.

Agile Methodology

What is Agile Methodology?

Agile methodology of software development is based on iterative and incremental approach. In agile, we have small incremental builds presented in multiple iterations to the end user and other stakeholders for their feedback. Based on the feedback, changes are incorporated in the next iterations of the build on the basis of their priority.
In agile, the different cross functional teams work together in the different iterations of the build with emphasis on more face-to-face interaction and less formal documentation.

Basically, agile is a methodology and there are different implementations following this methodology. Following are some of the most widey used agile implementations -

- **Scrum** - Scrum is one of the most popular implementation of agile where different roles like - product owner, scrum master and team members are assigned to different participants of software development. Daily scrum meeting are organised for the updates and a build is delivered in a two to three week cycle called sprint.

- **Extreme Programming (XP)** - Extreme programming is an agile implementation in which frequent customer feedback and changes are incorporated with focus on quality software. Quality of software is maintained by following the coding practices like pair programming(code reviews, unit testing etc.) to the extreme level. Hence, the name extreme programming.
- **Kanban** - Kanban is a development approach in which the tasks are organized in a Kanban board, wherein we can track the progress of the work, helping in decision making.
- **Crystal Clear** - Crystal clear development like other agile methodologies focusses on frequent delivery and feedback. It is a lightweight approach based on the fact that customization of process and practices is required to meet the project specific requirements.
- **Lean Software Development** - Lean software developement methodology is based on seven lean principles - eliminate waste(like any code not adding value), amplify learning, decide late, deliver fast, empower team, build integrity and See the Whole(see the product as a whole).

Features of Agile

Following are some of the features of Agile methodology-

1. Agile allows frequent deliverables to the end user.
2. Customer feedbacks and changes are embraced and incorporated in the iterations based on their priority.
3. It emphasis on colloborative work of cross functional team.
4. It focuses on more interaction and face-to-face communications.
5. It promotes regular review of the whole development process and fine tuning if required.

Advantages of Agile

Following are the advantages of Agile methodology-

1. Agile is very much suited for projects where requirements and the end product is not very clear.
2. It promotes customer satisfaction as their feedbacks and changes are embraced.

3. It reduces risk factor as early deliverables are made visible to the end users.
4. Exhaustive planning is not required at the beginning of the development process.
5. It is easy to manage with minimal rules and more flexibility.
6. Dividing the project into incremental deliverable builds leads to more focus on quality of the product.

Disadvantages of Agile

Following are the disadvantages of Agile methodology-

1. As it is highly customer-centric, so it can pose problem when the customer does not have clear understanding of the product and process.
2. Lack of formal documentation and designing leads to very high dependency on the individuals for training and other tasks.
3. For complex projects, the resource requirement and effort are difficult to estimate.
4. Frequent deliverable, feedback and collaboration can be very demanding for some customers.
5. Because of the ever-evolving features, there is always a risk of ever-lasting project.

Manual Testing

In this post, we will provide an overview of Software Testing. We will answer some common queries related to software testing and will be discussing **What, Why, Who, When and How** of testing.

What is Software Testing?

Software testing is the process of evaluating a system with the intend of finding bugs. It is performed to check if the system satisfies its specified requirements. Testing measures the overall quality of the system in terms of its correctness, completeness, usability, performance and other functional and non-functional attributes.

Why is Testing required?

Software Testing as a separate activity in SDLC, is required because-

- Testing provides an assurance to the stakeholders that product works as intended.
- Avoidable defects leaked to the end user/customer without proper testing adds bad reputation to the development company.
- Separate testing phase adds a confidence factor to the stakeholders regarding quality of the software developed.
- Defects detected earlier phase of SDLC results into lesser cost and resource utilisation of correction.
- Saves development time by detecting issues in earlier phase of development.
- Testing team adds another dimension to the software development by providing a different view point to the product development process.

Who does Testing?

Software Testing is/can be done by all technical and non technical people associated with the software. Testing in its various phases is done by-

- Developer - Developer does the unit testing of the software and ensure that the individual methods work currently
- Tester - Testers are the face of the software testing. A tester verifies the functionality, usability of the application as functional tester, a tester checks the performance of the application as a Performance tester, a tester automates the manual-functional test cases and creates test scripts as an automation tester
- Test Managers/Lead/Architects - Define the test strategy and test plan
- End users - A group of end users do the User Acceptance Testing (UAT) of the application to make sure the software can work in the real world.

When do we start Software Testing?

Based on the selection of different Software Development Life Cycle Model for the software project, testing phase gets started in the different phases. There is a software myth that testing is done only when some part of software is built but testing can(should) be started even before a single line of code is written. It can be done in parallel with development phase.

How is Software Testing done?

Software testing can be done both manually as well as using automation tools. Manual effort includes verification of requirement and design; development of test strategy and plan; preparation of test case and then the execution of tests. Automation effort includes preparation of test scripts for UI automation, back-end automation, performance test script preparation and use of other automation tools.

Why is testing necessary?

Hello friends! in this post, we will study about the need of software testing. Human beings are prone to mistakes because of in-attention, incorrect assumptions, carelessness or inadequate knowledge of the system. This very nature of humans make software vulnerable to bugs, defects and errors (we will get to know these terms in detail in later posts)

Let's now briefly see why we need testing in software context-

- Testing is important as it uncovers a defect before it is delivered to customer ensuring quality of software.
- So that the defects or bugs can be identified in early stages of development; as later the stage in which bug is identified, more is the cost to rectify it.
- It makes software more reliable and user-friendly to operate.
- An untested software not only makes software error prone, it also costs the customer business failure too like in case of Microsoft's MP3 player - Zune's crash.
- Software issue can cost lives too e.g. in case of Therac 25 - many people died due to concurrent programming errors wherein patients were given radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.
- Well tested software provide efficient resource utilization resulting in low cost.
- A thoroughly tested software ensures reliable and high performance functioning of the software.

What is Software Quality?

Software quality is the conformance of a software system to its requirements. In software perspective, quality is defined by two factors - Validation and Verification. Validation checks if the process used during software development is correct or not whereas in verification the product is evaluated to check if its meets the specifications. We will deal with verification and validation in detail, in our next post.

Attributes of Software Quality

- **Correctness** - Correctness measures the software quality for the conformance of the software to its requirements

- **Reliability** - Checks if the software performs its functions without any failure within the expected conditions
- **Robustness** - Robustness is the ability of the software to not crash when provided with unexpected input
- **Usability** - Usability is the ease of operating the software
- **Completeness** - Completeness is the extent tp which the software system meets its specifications
- **Maintainable** - Maintainability is the measure of the amount of effort required for software maintenance after it has shipped to end user
- **Portability** - Ability of the software to be transformed from one platform or infrastructure to other
- **Efficiency** - Efficiency is the measure of resources required for the functioning of the software

Verification and Validation

What is Verification?

Verification is a process of evaluating the intermediary work products of a software development lifecycle to check if we are in the right track of creating the final product.

Now the question here is : What are the intermediary products? Well, These can include the documents which are produced during the development phases like, requirements specification, design documents, data base table design, ER diagrams, test cases, traceability matrix etc. We sometimes tend to neglect the importance of reviewing these documents but we should understand that reviewing itself can find out many hidden anomalies when if found or fixed in the later phase of development cycle, can be very costly.
In other words we can also state that verification is a process to evaluate the mediator products of software to check whether the products satisfy the conditions imposed during the beginning of the phase.

What is Validation?

Validation is the process of evaluating the final product to check whether the software meets the business needs. In simple words the test execution which we do in our day to day life are actually the validation activity which includes smoke testing, functional testing, regression testing, systems testing etc…

Verification	Validation
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

When to stop testing?

This question - "When to stop testing" or "how much testing is enough" is very tricky to answer as we can never be sure that the system is 100% bug-free. But still there are some markers that help us in determining the closure of the testing phase of software development life cycle.

- **Sufficient pass percentage** - Depending on the system, testing can be stopped when an agreed upon test case pass percentage is reached.
- **After successful test case execution** - Testing phase can be stopped when one complete cycle of test cases is executed after the last known bug fix.
- **On meeting deadline** - Testing can be stopped after deadlines get met with no high priority issues left in system.

- **Mean Time Between failure (MTBF)**- MTBF is the time interval between to inherent failures. Based on stakeholders decisions, if the MTBF is quite large one can stop the testing phase
- **Based on Code coverage value** - Testing phase can be stopped when the automated code coverage reaches a certain acceptable value.

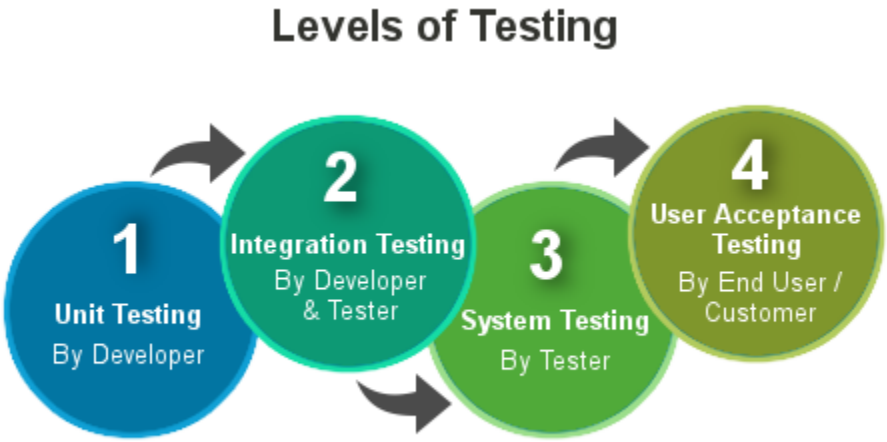
What is Software Testing Life Cycle?

Testing a software is not a single activity wherein we just validate the built product, instead it comprises of a set of activities performed throughout the application lifecycle. Software testing life cycle or STLC refers to all these activities performed during the testing of a software product.

Phases of STLC

- **Requirement Analysis -** In this phase the requirements documents are analysed and validated. Along with that the scope of testing is defined.
- **Test Planning and Control -** Test planning is one of the most important activities in test process. It involves defining the test specifications in order to achieve the project requirements. Whereas, test Control includes continuous monitoring of test progress with the set plan and escalating any deviation to the concerned stake holders.
- **Test Analysis and Design -** This phase involves analyzing and reviewing requirement documents, risk analysis reports and other design specifications. Apart from this, it also involves setting up of test infrastructure, creation of high level test cases and creation of traceability matrix.
- **Test Case Development -** This phase involves the actual test case creation. It also involves specification of test data and automated test scripts creation.
- **Test Environment Setup -** This phase involves creation of a test environment closely simulating the real world environment.
- **Test Execution -** - This phase involves manual and automated test case execution. During test case execution any deviation from the expected result leads to creation of defects in a defect management tool or manual logging of bugs in an excel sheet.
- **Exit Criteria Evaluation and Reporting -** This phase involves analyzing the test execution result against the specified exit criteria and creation of test summary report.
- **Test Closure -** This phase marks the formal closure of testing. It involves checking if all the project deliverable are delivered, archiving the testware, test environment and documenting the learning.

Different levels of Software Testing



Software testing can be performed at different levels of software development process. Performing testing activities at multiple levels help in early identification of bugs and better quality of software product. In this tutorial we will be studying the

different levels of testing namely - Unit Testing, Integration Testing, System Testing and Acceptance Testing.

Now, we will describe the different testing level of testing in brief and in the next tutorials we will explain each level individually, providing example and detailed explanation.

Unit Testing

- Unit Testing is the first level of testing usually performed by the developers.
- In unit testing a module or component is tested in isolation.
- As the testing is limited to a particular module or component, exhaustive testing is possible.
- Advantage - Error can be detected at early stage saving time and money to fix it.
- Limitation - Integration issues are not detected in this stage, modules may work perfectly on isolation but can have issues in interfacing between the modules.

Integration Testing

- Integration testing is the testing of a group of related modules.
- It aims at finding interfacing issues between the modules.
- It can be done in two ways - bottom-up integration and top-down integration (third type called Sandwich Integration is also used). We will refer to each approach in our Integration Testing Tutorial.

System Testing

- System Testing is the level of testing where the complete integrated application is tested as a whole.
- It aims at determining if the application conforms to its business requirements.
- System testing is carried out in an environment which is very similar to the production environment.

Acceptance Testing

- Acceptance testing is the final and one of the most important levels of testing on successful completion of which the application is released to production.
- It aims at ensuring that the product meets the specified business requirements within the defined standard of quality.
- There are two kinds of acceptance testing- alpha and beta testing. When acceptance testing is carried out by end users in developer's site it is known as alpha testing. User acceptance testing done by end users at end-user's site is called beta testing.

Test Design Techniques

What are the different test design techniques?

Test design techniques are standards of test designing which allow creation of systematic and widely accepted test cases. These techniques are based on the different scentific models and over the years experiences of many QA professional.

The test design techniques can be broadly categorized into two parts - "Static test design technique" and "Dynamic test design technique".

Static Test Design Techniques

The Static test design techniques are the testing techniques which involve testing without executing the code or the software application. So, basically static testing deals with Quality Assurance, involving reviewing and auditing of code and other design documents. The various static test design techniques can be further divided into two parts - "Static testing performed manually" and "Static testing using tools".

1. Manual Static Design Techniques

- **Walk through** - A Walk-through is step by step presentation of different requirement and design documents by their authors with the intent of finding defects or any missing pieces in the documents.
- **Informal reviews** - As the name suggest, an informal review done by an individual without any process or documentation.
- **Technical reviews** - A technical review involves reviewing the technical approach used during the development process. It is more of a peer review activity and less formal as compared to audit and inspection.
- **Audit** - An audit is a formal evaluation of the compliance of the different processes and artifacts with standards and regulations. It is generally performed by an external or independent team or person.
- **Inspection** - An inspection is a formal and documented process of reviewing the different documents by experts or trained professional.
- **Management review** - It is a review performed on the different management documents like project management plans, test plans, risk management plans etc.

2. Static Design Techniques Using Tools

- **Static analysis of code** - The static analysis techniques for the source code evaluation using tools are -
  - Control flow analysis - The control flow analysis requires analysis of all possible control flows or paths in the code.
  - Data flow analysis - The data flow analysis requires analysis of data in the application and its different states.
- **Compliance to coding standard** - This evaluates the compliance of the code with the different coding standards.
- **Analysis of code metrics** - The tool used for static analysis is required to evaluate the different metrics like lines of code, complexity, code coverage etc.

Dynamic Test Design Techniques

Dynamic test design techniques involves testing by running the system under test. In this technique, the tester provide input data to the application and execute it, in order to verify its different functional and non-functional requirements.

- **Specification based** - Specification based test design techniques are also referred to as **blackbox testing**. These involve testing based on the specification of the system under test without knowing its internal architecture. The different types of specification based test design or black box testing techniques are - "Equivalence partitioning", "Boundary value analysis", "Decision tables", "Cause-effect graph", "State transition testing" and "Use case testing".
- **Structure based** - Structure based test design techniques are also referred to as **white box testing**. In this techniques the knowledge of code or internal architecture of the system is required to carry out the testing. The various kinds of testing structure based or white testing techniques are - "Statement testing", "Decision testing/branch testing", "Condition testing", "Multiple condition testing", "Condition determination testing" and "Path testing".
- **Experienced based** - The experienced based techniques as the name suggest does not require any systematic and exhaustive testing. These are completely based on the experience or intution of the tester. Two most common forms of experienced based testing are - adhoc testing and exploratory testing.

In our next tutorials, we will be studying about the different test design techniques in detail.

Specification based Test design Techniques

Black Box Testing Techniques

What is specification based testing?

Specification based testing is also referred to as black-box testing. It involves performing testing based on the specification of the system under test. The knowledege of the internal architecture and coding is not required in specification based testing.

Specification based testing techniques

The different types of specification based test design techniques or black box testing techniques are-

- **Equivalence partitioning** - In equivalence class partitioning we group the input data into logical groups or equivalence classes. All the data items lying in an equivalence class are assumed to have same behavior when passed as input to the

application. E.g. for an application finding square of numbers, we can different equivalence classes like - all whoe positive numbers, negative numbers, decimal numbers, negative decimal numbers etc.

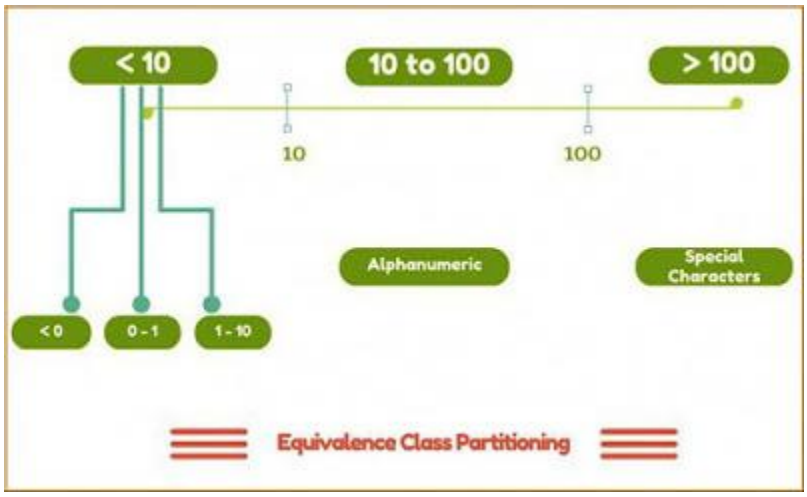
- **Boundary value analysis** - In boundary value analysis the boundary values of the equivalence partitioning classes are taken as input to the application. E.g. for equivalence classes limiting input between 0 and 100, the boundary values would be 0 and 100.
- **Decision tables** - Decision tables testing is used to test application's behaviour based on different combination of input values. A decision table has different set of input combination and their corresponding expected outcome on each row.
- **Cause-effect graph** - A cause-effect graph testing is carried out using graphical representation of input i.e. cause and output i.e. effect. We can find the coverage of cause effect graphs based on the percentage of combinations of inputs tested out of the total possible combinations.
- **State transition testing** - The state transition testing is based on state machine model. In this technique, we test the application by graphically representing the transition between the different states of the application based on the different events and actions.
- **Use case testing** - Use case testing is carried out using use cases. In this technique, we test the application using use-cases, representing the interaction of the application with the different actors.

What is equivalence class partitioning?

Equivalence class partitioning is a black-box testing technique (or specification based testing technique) in which we group the input data into logical groups or equivalence classes. All the data items lying in an equivalence class are assumed to be processed in the same way by the application when passed as input.

Example

Consider an example of an application that accepts a numeric number as input with value between 10 to 100 and finds its square. Now, we can create the following equivalence classes of the input-



- 10 to 100 - A valid equivalence class with acceptable numeric numbers
- Greater than 100 - An invalid equivalence class containing integer values greater than 100
- Less than 0 - An invalid equivalence class with negative numbers
- Between 0 to 1 - An invalid equivalence class with decimal numbers
- Alphanumeric characters - An invalid equivalence class with alphanumeric characters for checking the robustness of application
- Special characters - An invalid equivalence class with special characters

Advantages of equivalence class partitioning

1. Having equivalence classes greatly **reduces the number of test cases without compromising the test coverage or quality of testing.**
2. It helps in reducing the overall test execution time due to reduced set of test data.
3. It is highly used in cases where exhaustive testing is not possible but at the same time test good coverage needs to be maintained.

Disadvantages of equivalence class partitioning

1. The identification of equivalence classes relies heavily on the expertise of the tester. Having incorrectly identified equivalence classes leads to higher risk of defect leakage and less test coverage.
2. The equivalence classes needs to be partitioned just to the right amount and groups as having too large partitions leads to risk of missing defects. Whereas, partitioning into more groups of smaller sizes leads to redundant tests.

What is boundary value analysis?

Boundary value analysis is a black-box testing technique, closely associated with equivalence class partitioning. In this technique, we analyse the behaviour of the application with test data residing at the boundary values of the equivalence classes.

Example

Considering the same example we used in equivalence partitioning tutorial - an application that accepts a numeric number as input with value between 10 to 100. We identified different equivalence classes, out of which one of the valid equivalence classes was - integer with values ranging from 10 to 100. Now, for this equivalence class our boundary values would be {10, 100}. Similarly, we can find the values at the edges of the equivalence classes to create set of test data for the boundary value analysis.

Advantages of boundary value analysis

1. It is easier and faster to find defects as the density of defects at boundaries is more.
2. The overall test execution time reduces as the number of test data greatly reduces.

Disadvantages of boundary value analysis

1. The success of the testing using boundary value analysis depends on the equivalence classes identified, which further depends on the expertise of the tester and his knowledge of application. Hence, incorrect identification of equivalence classes leads to incorrect boundary value testing.
2. For application with open boundaries or application not having one dimensional boundaries are not suitable for boundary value analysis. In those cases, other black-box techniques like "Domain Analysis" are used.

Structure Based Test Design Techniques

White Box Testing Techniques

What is structure based testing?

Structure based testing is also referred to as white-box testing. In this technique, the knowledge of the code and internal architecture of the system is required to carry out the testing.

Structure based testing techniques

The different types of structure based test design or the white box testing techniques are-

- **Statement testing** - Statement testing is a white box testing approach in which test scripts are designed to execute code statements. The statement coverage is the measure of the percentage of statements of code executed by the test scripts out of the total code statements in the application. The statement coverage is the least preferred metric for checking test coverage.

- **Decision testing/branch testing** - Decision testing or branch testing is a white box testing approach in which test coverage is measured by the percentage of decision points(e.g. if-else conditions) executed out of the total decision points in the application.
- **Condition testing** - Testing the condition outcomes(TRUE or FALSE). So, getting 100% condition coverage required exercising each condition for both TRUE and FALSE results using test scripts(For n conditions we will have 2n test scripts).
- **Multiple condition testing** - Testing the different combinations of condition outcomes. Hence for 100% coverage we will have 2^n test scripts. This is very exhaustive and very difficult to achieve 100% coverage.
- **Condition determination testing** - It is an optimised way of multiple condition testing in which the combinations which doesn't affect the outcomes are discarded.
- **Path testing** - Testing the independent paths in the system(paths are executable statements from entry to exit points).

What is Test Strategy?

A Test Strategy is a **high-level document describing the way testing will be carried out**. It is presented by the project manager to all the stakeholders in the testing process. It can have a scope of an **entire organization or a particular project**.

Test Strategy Approaches

The the different apporaches to test strategy are-

1. Analytical Approach - Based on the risk analysis
2. Model-based Approach - Based on the various statistical models
3. Consultative Approach - Based on the consultation with technology or domain experts
4. Methodical Approach - Based on the experience
5. Heuristic Approach - Based on the exploratory techniques
6. Standard-compliant Approach - Based on the industry standards and processes

Test Strategy Document Template

A test strategy document can contain the following fields-

- **Test Strategy Id** - An identifier of the test strategy document and its various versions.
- **Introduction** - A brief introduction to the purpose and scope of the document.
- **Standards to use** - The different standards or set of guidelines to be followed.
- **Risks and Mitigations** - The different risks associated with in testing and their mitigation strategies.
- **Entry Criteria** - The set of pre-requisite that must be performed before testing can start.
- **Exit Criteria** - The criteria defining when the testing can be stopped.

- **Test design techniques** - The test design techniques to be used like - equivalence partitioning, boundary value analysis etc.
- **Test environment** - The test environment specifications.
- **Configuration management of testware** - Specification of the right version of testware for testing.
- **Test process improvement** - The approaches to use for improving the test process.
- **Approvals** - The persons approving the test strategy document.

What is a Test Plan?

A Test Plan is a formal document derived from requirement documents (Software Requirement Specification, Use Case documents etc.), describing in detail the scope of testing and the different activities performed in testing. It is generally prepared by a test manager and approved by the different stakeholders of the application.

Features of a Test Plan

A Test Plan needs to address the following-

- Overall scope of testing
- Risk Analysis
- Test estimate
- Resource Requirement
- Tools used
- Scheduling, review and analysis of test design activities
- Creation of test cases and test data
- Identification of test monitoring and test control activities

Test Plan Template

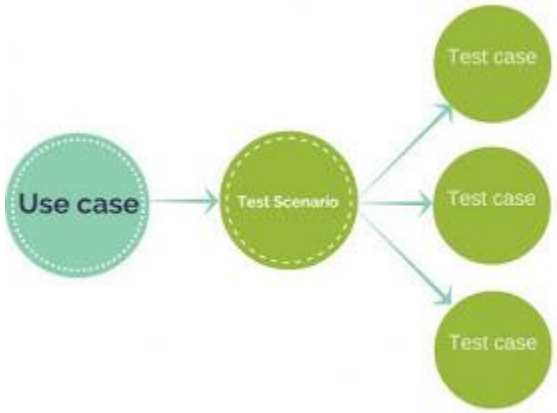
A test plan contains the following fields-

- **Test Plan Id** - A unique identifier of the test plan and its different associated versions.
- **Introduction** - A brief introduction to the application under test.
- **Purpose and scope** - The overall purpose and scope of testing.
- **Testing Strategy** - Describes the testing approach in the software development life cycle.
- **Features to be tested** - Describes the features of the application to be tested.
- **Features not to be tested** - Describes the feature not in the scope of testing.
- **Test deliverables** - The different types of test artifacts to be delivered like test cases, test results etc.
- **Environmental needs** - Any environment specific need of project.
- **Responsibilities** - The resources involved in the testing process and their responsibilities.
- **Training needs** - The project specific training requirements of different resources.
- **Schedule** - The estimate or schedule of the test deliverables.
- **Risks and Mitigations** - The different risks associated with the project and their mitigation strategies.
- **Tools** - List of tools used (if any).

- **Approvals** - The persons approving the test plan.

What is a Test Scenario?

A Test Scenario is generally a one line statement describing a feature of application to be tested. It is used for end to end testing of a feature and is generally derived from the use cases. A single test scenario can cover one or more test cases i.e. it has a one to many relationship with test cases.



What is Scenario testing?

Scenario testing is a type of testing carried out using scenarios derived from the use cases. Using scenario testing, complex application-logic can be tested using easy to evaluate test scenarios. Some advantages of test scenarios are -

- Test scenarios can serve as basis for lower level test case creation.
- Testing using test scenarios can be carried out relatively faster than the one using test cases.
- Saves a lot of time, better with projects having time constraints.

What is a Test Case?

A test case is a set of conditions for evaluating a particular feature of a software product to determine its compliance with the business requirements. A test case has pre-requisites, input values and expected results in a documented form which cover the different test scenarios.

A test case can have following attributes-

1. TestCaseId - A unique identifier of the test case.
2. Test Summary - One liner summary of the test case.
3. Description - Detailed description of the test case.
4. Prerequisite or pre-condition - A set of prerequisites that must be followed before executing the test steps.
5. Test Steps - Detailed steps for performing the test case.
6. Expected result - The expected result in order to pass the test.
7. Actual result - The actual result after executing the test steps.

8. Test Result - Pass/Fail status of the test execution.
9. Automation Status - Identifier of automation - whether the application is automated or not.
10. Date - The test execution date.
11. Executed by - Name of the person executing the test case.

How to write good test cases?

As we know that a test case is a set of conditions for evaluating a software product to determine its compliance with the business requirements. Having an ill-formed test cases can lead to severe defect leakage, which can cost both time and money. So, writing effective test cases is an utmost requirement for the success of any software product.

Now, let's see how we can write effective test cases-

1. Test design technique - Follow a test design technique best suited for your organization or project specific needs like - boundary value analysis, equivalence class partitioning etc. This ensures that well researched standards and practices are implemented during test case creation.
2. Clear and concise tests - The test case summary, description, test steps, expected results etc should be written in a clear and concise way, easily understandable by the different stakeholders in testing.
3. Uniform nomenclature - In order to maintain consistency across the different test cases a uniform nomenclature and set of standards should be followed while writing the test cases.
4. Fundamental/Atomic Test cases - Create test cases as fundamental as possible, testing a single unit of functionality without merging or overlapping multiple testable parts.
5. Leave no scope of ambiguity - Write test case with clear set of instruction e.g. instead of writing "Open homepage", write - "Open homepage - http://www.{homepageURL}.com in the browser and press enter".
6. No Assumptions - While writing test cases do not assume any functionality, pre-requisite or state of the application. Instead, bound the whole test case creation activity to the requirement documents - SRS, Use-case documents etc.
7. Avoid redundancy - Don't repeat the test cases, this leads to wastage of both time and resources. This can be achieved by well planned and categorized test cases.
8. Traceable tests - Use traceability matrix to ensure that 100% of the application's feature in the scope of testing are covered in the test cases.
9. Ensure that different aspects of software are covered - Ensure that apart from the functionality, the different aspects of software are tested like performance, usability, robustness etc are covered in the test case by creating performance test cases and benchmarks, usability test cases, negative test cases etc.
10. Test data - The test data used in testing should be as diversed and as close to real time usage as possible. Having diverse test data can more reliable test cases.

What is a defect?

A defect or a bug is an error in a program that causes the application to perform in an unintended manner, deviating from its requirements. Based on the urgency of fixing the defect, we can classify them on a scale of P0 to P3, with P0 defect having the most urgency to fix. Also, the defects can be classified based on their criticality or the impact to the functionality. Depending on the organisation, we can have different levels of defect severity ranging from minor to critical or show stopper.
To report a bug we have different Defect Management Tools like - Jira, Mantis, Bugzilla etc. Next, we will see the different components of a Defect Report.

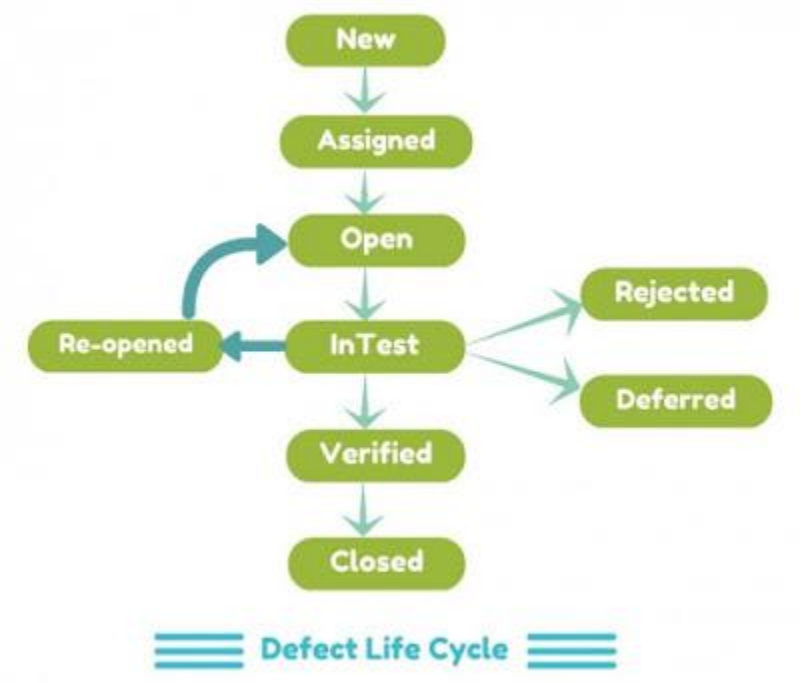
Defect Reporting Template

- **DefectId** - A unique identifier of the defect.

- **Summary** - A one line summary of the defect, more like a defect title.
- **Description** - A detailed description of the defect.
- **Build Version** - Version of the build or release in which defect is found.
- **Steps to reproduce** - The steps to reproduce the defect.
- **Expected Behaviour** - The expected behaviour from which the application is deviating because of the defect.
- **Actual Behaviour** - The current erroneous state of the application w.r.t. the defect.
- **Priority** - Based on the urgency of the defect, this field can be set on a scale of P0 to P3.
- **Severity** - Based on the criticality of the defect, this field can be set to minor, medium, major or show stopper.
- **Reported By** - Name of the QA, reporting the defect.
- **Reported On** - The date on which the defect was raised.
- **Assigned To** - The person to whom the defect is assigned in its current state. It can be the developer fixing the defect, the QA for verification of the fixed defect or the manager approving the defect.
- **Current Status** - The current status of defect (one of the states of the defect life cycle).
- **Environment** - The environment on which the defect is found - release, staging, production etc.

What is a defect life cycle?

A defect life cycle is the movement of a bug or defect in different stages of its lifetime, right from the beginning when it is first identified till the time is marked as verified and closed. Depending on the defect management tool used and the organisation, we can have different states as well different nomenclature for the states in the defect life cycle.



- **New** - A bug or defect when detected is in New state

- **Assigned** - The newly detected bug when assigned to the corresponding developer is in Assigned state
- **Open** - When the developer works on the bug, the bug lies in Open state
- **Rejected/Not a bug** - A bug lies in rejected state in case the developer feels the bug is not genuine
- **Deferred** - A deferred bug is one, fix of which is deferred for some time(for the next releases) based on urgency and criticality of the bug
- **Fixed/InTest** - When a bug is resolved by the developer it is marked as fixed and assigned to the tester
- **Reopened** - If the tester is not satisfied with issue resolution the bug is moved to Reopened state
- **Verified** - After the Test phase if the tester feels bug is resolved, it is marked as verified
- **Closed** - After the bug is verified, it is moved to Closed status.

Test Scripts, Test Cases, and Test Scenarios: Understanding the Difference

If you were asked to write a test case, would you know what to do? What about a test script, or a test scenario? The first step is learning what these terms mean. Each of these terms implies a different level of detail and is used for a different purpose. Once a tester knows what each of these terms mean, they can figure out how to use them to describe the testing work that is done on a daily basis.

Test Scripts

This story begins with the most detailed way to document testing, the test script. When people talk about test scripts, they usually mean a line-by-line description of all the actions and data needed to perform a test. A script typically has 'steps' that try to fully describe how to use the program — which buttons to press, and in which order — to carry out a particular action in the program. These scripts also include specific results that are expected for each step, such as observing a change in the UI. An example step might be "Click the 'X' button," with an example result of "The window closes."

When a tester first starts a new job, they might not know much about the product, the business domain, or even software testing. Scripts can help bridge that gap. If the tester carefully follows the directions — enter the string 'abc', click the submit button, make sure the form submitted and the value was saved — the test idea will be covered enough to consider it 'tested'.

There are a few drawbacks to consider before going all-in with detailed scripts. Active software projects change often — pages get redesigned, user experience changes, and new functionality is added. To be effective over time, testers have to make a continuous effort to update the scripts to match the new product. This can take time away from testing. Another drawback is that scripted tests are often designed to test one specific thing repeatedly, using the same steps and the same data each time the test is executed. This means that if there are bugs that lie outside the directions given in the test script, they will not be found unless the tester strays from the script. Scripted tests do not always encourage testers to use the creativity and technical skill required to find hidden bugs.

Test Cases

The second most detailed way of documenting testing work is to use test cases. Test cases describe a specific idea that is to be tested, without detailing the exact steps to be taken or data to be used. For example, a test

case might say “Test that discount codes can be applied on top of a sale price.” This doesn’t mention how to apply the code or whether there are multiple ways to apply the code. The actual testing that will cover this test case may vary from time to time. Will the tester use a link to apply a discount, or enter a code, or have a customer service rep apply the discount, or will they feel compelled to test every way to add a discount that they can think of? Test cases give flexibility to the tester to decide exactly how they want to complete the test.

This flexibility from test cases is both good and bad. Flexibility is beneficial when the tester is familiar with testing and familiar with the software under test and the current set of risks in the software. If the tester clearly understands what has already been tested, what has changed recently in the program, and how users typically use the program, they will choose an approach in their testing that will exercise both the most common user paths, and the less common paths that are most likely to reveal bugs.

On the other hand, if the tester does not have a good understanding of how the program is used, the recent risks to the program, and how to evaluate those risks as a tester, they may not have the information or skill they need to assess the actions required to reveal important bugs.

Test Scenarios

The least detailed type of documentation is the test scenario. A test scenario is a description of an objective a user might face when using the program. An example might be “Test that the user can successfully log out by closing the program.” Typically, a test scenario will require testing in a few different ways to ensure the scenario has been satisfactorily covered. Just based on that light description, the tester might choose to close the program through the menu option, kill it through the task manager, turn the computer off, or see what happens when the program runs out of memory and crashes. Since test scenarios offer little information about how to complete the testing, they offer the maximum amount of flexibility to the tester responsible for them.

Like test cases, the flexibility that comes with using test scenarios creates similar benefits and drawbacks. Testing skill and domain knowledge make it easier for the tester to break test scenarios down into the relevant test ideas, select the approach that makes most sense, and perform tests that find important problems. This work is fun and challenging for a skilled tester, but it may be difficult or impossible for a novice unless they are able to collaborate with others to get the needed skill and perspective.

5 Major Mobile App Testing Challenges & Solutions to Check Out

As compared to traditional desktop and web application testing, **mobile app testing** is more complex; thus, it is important for mobile app developers and testers to go through different challenges during the app testing process.

In the recent times, mobile apps have made a huge space in the lives of humans and have grown much faster than we expected. However, an increase in the app usage has been followed by **rapid mobile app development** that gave birth to a need for proper testing. One of the main reasons behind this is to provide better user experience and app performance.

But there are some of the challenges that a mobile app tester will face while testing an application, so he/she needs to overcome the challenges and develop a competing mobile application. Before we jump into the different challenges of it, we should understand the exact need of it.

Importance of Mobile App Testing

As per the latest report, 39% of mobile users are not happy with the online experience that they get on a daily basis. One another report suggests that 46% of these unsatisfied users will never come back after experiencing a poor experience.

According to these two reports, there is a huge need of improvement. Once you act on right time, you will have approximately 3 to 5 people worldwide, who prefer to purchase smartphone so that they can make them your application user by 2020. These numbers have formed a need to learn why you should enhance the mobile app experience.

Let's Give an Instant Look at the Major Challenges of Mobile App Testing

Alertness

It is important that a mobile app tester stays updated with the **latest and current innovative testing approaches.** However, it can begin the process of mobile app and comprise everything till the data functionalities with the environment.

You should comprehend that the mobile functionalities can modify as per the contexts with different data that ultimately invite a completely new scenario for the testing process. Apart from this, the changing field of smartphones complexes the situation. Therefore, the app tester also needs to stay updated with the same.

Device Differences

Because of the compatibility issues, mobile app testing is quite difficult as the mobile app can be deployed across different devices, which have varied operating systems like iOS, Android, etc., manufactures (Samsung, Apple, etc.) and more. Moreover, the team can't be 100% sure about the tested application that if the app works well with a given device, it will run efficiently on other devices.

Creating User Experience

In the success of the mobile application, user experience plays a very important role. However, the process of developing an app needs different experts and professionals, including designers, developers, and analysts that are on the top on the list.

Though every single of them does their work efficiently, it lands on the tester to appraise the mobile application. However, he needs to go carefully through each aspect. One of the major challenges here is to comprehend the target users, market, and the competitors.

Therefore, you should keep these all together with the brand's value and app tester needs to conduct important tests. Even the smallest mistake can result in poor user experience, but it can be solved with the right communication between the team and by some research.

Usability

As we all know that mobile device screens are quite small, and there is always a huge data that we want to present than possible to fit the screen. It is challenging for you to keep the interaction clean and easy for the user, and showing all the important information.

However, readability and font size are also considered as the complex factors of usability. While testing mobile apps, it is essential to give attention on the size of click areas and ensure that all texts are readable without lenses.

Compatibility With the Network

The connectivity of the network can't be defined based on beliefs. A user with 4G/3G connectivity doesn't necessarily have sound connectivity. However, there are some good chances that your application users might be dependent on a 2G network.

Thus, it is important for a mobile app tester to cross check that the app works smoothly and wonderfully on different networks. In order to do this, the tester needs to ensure he executes the tests on all different network types.

So, these are the five major challenges of mobile app testing that every mobile app tester face while testing an app, but it is must they overcome these challenges and make the app more efficient. However, it would be better to hire a professional mobile app tester company that has a qualified team of quality assurance.

Manual Testing Interview Questions

Ques.1.What is Software Testing?

Ans. Software testing is the process of evaluating a system to check if it satisfies its business requirements. It measures the overall quality of the system in terms of attributes like correctness, completeness, usability, performance etc. Basically, it is used for ensuring the quality of software to the stakeholders of the application.

Ques.2. Why is testing required?

Ans. We need software testing for following reasons-

1. Testing provides an assurance to the stakeholders that product works as intended.
2. Avoidable defects leaked to the end user/customer without proper testing adds bad reputation to the development company.
3. Defects detected earlier phase of SDLC results into lesser cost and resource utilisation of correction.
4. Saves development time by detecting issues in earlier phase of development.
5. Testing team adds another dimension to the software development by providing a different view point to the product development process.

Ques.3. When should we stop testing?

Ans. Testing can be stopped when one or more of the following conditions are met-

1. After test case execution - Testing phase can be stopped when one complete cycle of test cases is executed after the last known bug fix with agreed upon value of pass-percentage.
2. Once the testing deadline is met - Testing can be stopped after deadlines get met with no high priority issues left in system.
3. Based on Mean Time Between failure (MTBF)- MTBF is the time interval between two inherent failures. Based on stakeholders decisions, if the MTBF is quite large one can stop the testing phase.
4. Based on code coverage value - Testing phase can be stopped when the automated code coverage reaches a specific threshold value with sufficient pass-percentage and no critical bug.

Ques.4. What is Quality Assurance?

Ans. Quality assurance is a process driven approach which checks if the process of developing the product is correct and conforming to all the standards. It is considered as a preventive measure as it identifies the

weakness in the process to build a software. It involves activities like document review, test cases review, walkthroughs, inspection etc.

Ques.5. What is Quality Control?
Ans. Quality control is product driven approach which checks that the developed product conforms to all the specified requirements. It is considered as a corrective measure as it tests the built product to find the defects. It involves different types of testing like functional testing, performance testing, usability testing etc.

Ques.6. What is the difference between Verification and Validation?
Ans. Following are the major differences between verification and validation-

Verification	Validation
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

Ques.7. What is SDLC?
Ans. Software Development Life Cycle refers to all the activities that are performed during software development, including - requirement analysis, designing, implementation, testing, deployment and maintenance phases.

Ques.8. Explain STLC - Software Testing life cycle.
Software testing life cycle refers to all the activities performed during testing of a software product. The phases include-

- Requirement analyses and validation - In this phase the requirements documents are analysed and validated and scope of testing is defined.
- Test planning - In this phase test plan strategy is defined, estimation of test effort is defined along with automation strategy and tool selection is done.
- Test Design and analysis - In this phase test cases are designed, test data is prepared and automation scripts are implemented.
- Test environment setup - A test environment closely simulating the real world environment is prepared.
- Test execution - The test cases are prepared, bugs are reported and retested once resolved.
- Test closure and reporting - A test closure report is prepared having the final test results summary, learnings and test metrics.

Ques.9. What are the different types of testing?
Testing can broadly be defined into two types-

- **Functional testing** - Functional testing involves validating the functional specifications of the system.
- **Non Functional testing** - Non functional testing includes testing the non-functional requirements of the system like performance, security, scalability, portability, endurance etc.

Going by the way the testing is done, it can be categorized as-

- **Black box testing** - In black box testing, the tester need not have any knowledge of the internal architecture or implementation of the system. The tester interact with the system through the interface providing input and validating the received output.
- **White box testing** - In white box testing, the tester analyses the internal architecture of the system as well as the quality of source code on different parameters like code optimization, code coverage, code re usability etc.
- **Gray box testing** - In grey box testing, the tester has partial access to the internal architecture of the system e.g. the tester may have access to the design documents or database structure. This information helps tester to test the application better.

Ques.10. What is a test bed?
Ans. A test bed is a test environment used for testing an application. A test bed configuration can consist of the hardware and software requirement of the application under test including - operating system, hardware configurations, software configurations, tomcat, database etc.

Ques.11. What is a test plan?
Ans. A test plan is a formal document describing the scope of testing, the approach to be used, resources required and time estimate of carrying out the testing process. It is derived from the requirement documents(Software Requirement Specifications).

Ques.12. What is a test scenario?
Ans. A test scenario is derived from a use case. It is used for end end to end testing of a feature of an application. A single test scenario can cater multiple test cases. The scenario testing is particularly useful when there is time constraint while testing.

Ques.13. What is a test case?
Ans. A test case is used to test the conformance of an application with its requirement specifications. It is a set of conditions with pre-requisites, input values and expected results in a documented form.

Ques.14. What are some attributes of a test case?
Ans. A test case can have following attributes-

1. TestCaseId - A unique identifier of the test case.
2. Test Summary - Oneliner summary of the test case.
3. Description - Detailed description of the test case.
4. Prerequisite or pre-condition - A set of prerequisites that must be followed before executing the test steps.
5. Test Steps - Detailed steps for performing the test case.
6. Expected result - The expected result in order to pass the test.
7. Actual result - The actual result after executing the test steps.
8. Test Result - Pass/Fail status of the test execution.
9. Automation Status - Identifier of automation - whether the application is automated or not.
10. Date - The test execution date.
11. Executed by - Name of the person executing the test case.

Ques.15. What is a test script?
Ans. A test script is an automated test case written in any programming or scripting langauge. These are basically a set of instructions to evaluate the functioning of an application.

Ques.16. What is a bug?
Ans. A bug is a fault in a software product **detected at the time of testing**, causing it to function in an unanticipated manner.

Ques.17. What is a defect?
Ans. A defect is non-conformance with the requirement of the product **detected in production** (after the product goes live).

Ques.18. What are some defect reporting attributes?
Ans. Some of the attributes of a Defect report are-

- DefectId - A unique identifier of the defect.
- Defect Summary - A one line summary of the defect, more like a defect title.

- Defect Description - A detailed description of the defect.
- Steps to reproduce - The steps to reproduce the defect.
- Expected Result - The expected behaviour from which the application is deviating because of the defect.
- Actual Result- The current erroneous state of the application w.r.t. the defect.
- Defect Severity - Based on the criticality of the defect, this field can be set to minor, medium, major or show stopper.
- Priority - Based on the urgency of the defect, this field can be set on a scale of P0 to P3.

Ques.19. What are some of the bug or defect management tools?
Ans. Some of the most widely used Defect Management tools are - Jira, Bugzilla, Redmine, Mantis, Quality Center etc.

Ques.20. What is defect density?
Ans. Defect density is the measure of density of the defects in the system. It can be calculated by dividing number of defect identified by the total number of line of code(or methods or classes) in the application or program.

Ques.21. What is defect priority?
Ans. A defect priority is the urgency of the fixing the defect. Normally the defect priority is set on a scale of P0 to P3 with P0 defect having the most urgency to fix.

Ques.22. What is defect severity?
Ans. Defect severity is the severity of the defect impacting the functionality. Based on the organisation we can different levels of defect severity ranging from minor to critical or show stopper.

Ques.23. Give an example of Low priority-Low severity, Low priority-High severity, High priority-Low severity, High priority-High severity defects.
Ans.

1. **Low priority-Low severity** - A spelling mistake in a page not frequently navigated by users.
2. **Low priority-High severity** - Application crashing in some very corner case.
3. **High priority-Low severity** - Slight change in logo color or spelling mistake in company name.
4. **High priority-High severity** - Issue with login functionality.

Ques.24. What is a blocker?
Ans. A blocker is a bug of high priority and high severity. It prevents or blocks testing of some other major portion of the application as well.

Ques.25. What is a critical bug?
Ans. A critical bug is a bug that impacts a major functionality of the application and the application cannot be delivered without fixing the bug. It is different from blocker bug as it doesn't affect or blocks the testing of other part of the application.

Ques.26. Explain bug lifecycle or the different states of a bug.
Ans. A bug goes through the following phases in software development-

- New - A bug or defect when detected is in New state
- Assigned - The newly detected bug when assigned to the corresponding developer is in Assigned state
- Open - When the developer works on the bug, the bug lies in Open state
- Rejected/Not a bug - A bug lies in rejected state in case the developer feels the bug is not genuine
- Deferred - A deferred bug is one, fix of which is deferred for some time(for the next releases) based on urgency and criticality of the bug
- Fixed - When a bug is resolved by the developer it is marked as fixed
- Test - When fixed the bug is assigned to the tester and during this time the bug is marked as in Test
- Reopened - If the tester is not satisfied with issue resolution the bug is moved to Reopened state
- Verified - After the Test phase if the tester feels bug is resolved, it is marked as verified
- Closed - After the bug is verified, it is moved to Closed status.

Ques.27. What are the different test design techniques?
Ans. Test design techniques are different standards of test designing which allow systematic and widely accepted test cases. The different test design techniques can be categorized as static test design technique and dynamic test design technique.

1. Static Test Design Techniques - The test design techniques which involves testing without executing the code. The various static test design techniques can be further divided into two parts manual and using tool-

- Manual static design techniques-
  - Walk through
  - Informal reviews
  - Technical reviews
  - Audit
  - Inspection
  - Management review
- Static design techniques using tool-
  - Static analysis of code - It includes analysis of the different paths and flows in the application and different states of the test data.
  - Compliance to coding standard - This evaluates the compliance of the code with the different coding standards.
  - Analysis of code metrics - The tool used for static analysis is required to evaluate the different metrics like lines of code, complexity, code coverage etc.

2. Dynamic Test Design Techniques - Dynamic test design techniques involves testing by running the system under test.

- Specification based - Specification based test design techniques are also referred to as blackbox testing. These involve testing based on the specification of the system under test without knowing its internal architecture.
- Structure based - Structure based test design techniques are also referred to as white box testing. In this techniques the knowledge of code or internal architecture of the system is required to carry out the testing.
- Experienced based - The experienced based techniques are completely based on the experience or intution of the tester. Two most common forms of experienced based testing are - adhoc testing and exploratory testing.

Ques.28. Explain the different types of specification based test design technique?

Ans. Specification based test design techniques are also referred to as blackbox testing. It involves testing based on the specification of the system under test without knowing its internal architecture. The different types of specification based test design or black box testing techniques are-

- Equivalence partitioning - Grouping test data into logical groups or equivalence classes with the assumpation that any all the data items lying in the classes will have same effect on the application.
- Boundary value analysis - Testing using the boundary values of the equivalence classes taken as the test input.
- Decision tables - Testing using decision tables showing application's behaviour based on different combination of input values.
- Cause-effect graph - Testing using graphical representation of input i.e. cause and output i.e. effect is used for test designing.
- State transition testing - Testing based on state machine model.
- Use case testing - Testing carried out using use cases.

Ques.29. Explain equivalence class partitioning.
Ans. Equivalence class partitioning is a specification based black box testing techniques. In equivalence class partitioning, set of input data that defines different test conditions are partitioned into logically similar groups such that using even a single test data from the group for testing can be considered as similar to using all the other data in that group. E.g. for testing a Square program(program that prints the square of a number- the equivalence classes can be-
Set of Negative numbers, whole numbers, decimal numbers, set of large numbers etc.

Ques.30. What is boundary value analysis?
Ans. Boundary value analysis is a software testing technique for designing test cases wherein the boundary values of the classes of the equivalence class partitioning are taken as input to the test cases e.g. if the test data lies in the range of 0-100, the boundary value analysis will include test data - 0,1, 99, 100.

Ques.31. What is decision table testing?
Ans. Decision table testing is a type of specification based test design technique or black box testing technique in which testing is carried out using decision tables showing application's behaviour based on different combination of input values. Decision tables are particularly helpful in designing test cases for complex business scenarios involving verification of application with multiple combinations of input.

Ques.32. What is a cause effect graph?
Ans. A cause effect graph testing is black box test design technique in which graphical representation of input i.e. cause and output i.e. effect is used for test designing. This technique uses different notations representing AND, OR, NOT etc relations between the input conditions leading to output.

Ques.33. What is state transition testing?
Ans. State transition testing is a black box test design technique based on state machine model. State transition testing is based on the concept that a system can be defined as a collection of multiple states and the transition from one state to other happens because of some event.

Ques.34. What is use case testing?
Ans. A use case testing is a black box testing approach in which testing is carried out using use cases. A use case scenario is seen as interaction between the application and actors(users). These use cases are used for depicting requirements and hence can also serve as basis for acceptance testing.

Ques.35. What is structure based testing?
Ans. Structure based test design techniques are also referred to as white box testing. In this techniques the knowledge of code or internal architecture of the system is required to carry out the testing. The various kinds of testing structure based or white testing techniques are-

- Statement testing - Test scripts are designed to execute code statements and coverage is the measure of line of code or statements executed by test scripts.
- Decision testing/branch testing - Measure of the percentage of decision points(e.g. if-else conditions) executed out of the total decision points in the application.
- Condition testing- Testing the condition outcomes(TRUE or FALSE). So, getting 100% condition coverage required exercising each condition for both TRUE and FALSE results using test scripts(For n conditions we will have 2n test scripts).
- Multiple condition testing - Testing the different combinations of condition outcomes. Hence for 100% coverage we will have 2^n test scripts. This is very exhaustive and very difficult to achieve 100% coverage.
- Condition determination testing - It is an optimized way of multiple condition testing in which the combinations which doesn't affect the outcomes are discarded.

- Path testing - Testing the independent paths in the system(paths are executable statements from entry to exit points).

Ques.36. What is Statement testing and statement coverage in white box testing?
Ans. Statement testing is a white box testing approach in which test scripts are designed to execute code statements.
Statement coverage is the measure of the percentage of statements of code executed by the test scripts out of the total code statements in the application.
The statement coverage is the least preferred metric for checking test coverage.

Ques.37. What is decision testing or branch testing?
Ans. Decision testing or branch testing is a white box testing approach in which test coverage is measured by the percentage of decision points(e.g. if-else conditions) executed out of the total decision points in the application.

Ques.38. What are the different levels of the testing?
Ans. Testing can be performed at different levels during the development process. Performing testing activities at multiple levels help in early identification of bugs. The different levels of testing are -

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

Ques.39. What is unit testing?
Ans. Unit testing is the first level of testing and it involves testing of individual modules of the software. It is usually performed by developers.

Ques.40. What is integration testing?
Ans. Integration testing is performed after unit testing. In integration testing we test the group of related modules. It aims at finding interfacing issues between the modules.

Ques.41. What are the different types of integration testing?
Ans. The different type of integration testing are-

1. Big bang Integration Testing - In big bang integration testing, testing starts only after all the modules are integrated.
2. Top-down Integration Testing - In top down integration, testing/integration starts from top modules to lower level modules.
3. Bottom-up Integration Testing - In bottom up integration, testing starts from lower level modules to higher level module up in the heirarchy.
4. Hybrid Integration Testing - Hybrid integration testing is the combination of both Top-down and bottom up integration testing. In this approach the integration starts from middle layer and testing is carried out in both the direction

Ques.42. What is stub?
Ans. In case of Top-down integration many a times lower level modules are not developed while beginning testing/integration with top level modules. In those cases Stubs or dummy modules are used that simulate the working of modules by providing hardcoded or expected output based on the input values.

Ques.43. What is driver?
Ans. In case of Bottom up integration drivers are used to simulate the working of top level modules in order to test the related modules lower in the heirarchy.

Ques.44. What is a test harness? Why do we need a test harness?
Ans. A test harness is a collection of test scripts and test data usually associated with unit and integration testing. It involves stubs and drivers that are required for testing software modules and integrated components.

Ques.45. What is system testing?
Ans. System testing is the level of testing where the complete software is tested as a whole. The conformance of the application with its business requirements is checked in system testing.

Ques.46. What is acceptance testing?
Ans. Acceptance testing is a testing performed by the potential end user or

customers to check if the software conforms to the business requirements and can be accepted for use.

Ques.47. What is alpha testing?
Ans. Alpha testing is a type of acceptance testing that is performed end users at the developer site.

Ques.48. What is beta testing?
Ans. Beta testing is the testing done by end users at end user's site. It allows users to provide direct input about the software to the development company.

Ques.49. What is adhoc testing?
Ans. Adhoc testing is an unstructured way of testing that is performed without any formal documentation or proper planning.

Ques.50. What is monkey testing?
Ans. Monkey testing is a type of testing that is performed randomly without any predefined test cases or test inputs.

Ques.51. How is monkey testing different from adhoc testing?
Ans. In case of adhoc testing although there are no predefined or documented test cases still testers have the understanding of the application. While in case of monkey testing testers doesn't have any understanding of the application.

Ques.52. What is exploratory testing?
Ans. Exploratory testing is a type of testing in which new test case are added and updated while exploring the system or executing test cases. Unlike scripted testing, test design and execution goes parallely in exploratory testing.

Ques.53. What is performance testing?
Ans. Performance testing is a type of non-functional testing in which the performance of the system is evaluated under expected or higher load. The various performance parameters evaluated during performance testing are - response time, reliability, resource usage, scalabilty etc.

Ques.54. What is load testing?
Ans. Load testing is a type of performance testing which aims at finding application's performance under expected workload. During load testing we evaluate the response time, throughput, error rate etc parameters of the application.

Ques.55. What is stress testing?
Ans. Stress testing is a type of performance testing in which application's behaviuor is monitored under higher workload tehn expected. Stress testing is done to find memory leaks, robustness of the application as it is subjected to high workload.

Ques.56. What is volume testing?
Ans. Volume testing is a type of performance testing in which the performance of application is evaluated with large amount of data. It checks the scalability of the application and helps in identification of bottleneck with high volume of data.

Ques.57. What is endurance testing or Soak testing?
Ans. Endurance testing is a type of performance testing which aims at finding issues like memory leaks when an application is subjected to load test for a long period of time.

Ques.58. What is spike testing?
Ans. Endurance testing is a type of performance testing in which the application's performance is measured while suddenly increasing the number of active users during the load test.

Ques.59. What is usability testing?
Ans. Usability testing is the type of testing that aims at determining the extent to which the application is easy to understand and use.

Ques.60. What is Accessibility testing?
Ans. Accessibility is the type of testing which aims at determining the ease of use or operation of the application specifically by with disabilities.

Ques.61. What is compatibity testing?
Ans. Testing software to see how compatible the software is with a particular environment - Operating system, platform or hardware.

Ques.62. What is configuration testing?
Ans. Configuration testing is the type of testing used to evaluate the configurational requirements of the software along with effect of changing the required configuration.

Ques.63. What is localisation testing?
Ans. Localisation testing is a type of testing in which we evaluate the application's customization(localized version of application) to a particular culture or locale. Generally the content of the application is checked for updation(e.g. content language).

Ques.64. What is globalisation testing?
Ans. Globalisation testing is a type of testing in which application is evaluated for its functioning across the world.

Ques.65. What is negative testing?
Ans. Negative testing is a type of testing in which the application's robustness(graceful exiting or error reporting) is evaluated when provided with invalid input or test data.

Ques.66. What is security testing?
Ans. Security testing is a type of testing which aims at evaluating the integrity, authentication, authorization, availabilty, confidentiality and non-repudation of the application under test.

Ques.67. What is penetration testing?
Ans. Penetration testing or pen testing is a type of security testing in which application is evaluated(safely exploited) for different kinds of vulnerabilities that any hacker could expolit.

Ques.68. What is robustness testing?
Ans. Robustness testing is a type of testing that is performed to find the

robustness of the application i.e. the ability of the system to behave gracefully in case of erroneous test steps and test input.

Ques.69. What is A/B testing?
A/B testing is a type of testing in which the two variants of the software product are exposed to the end users and on analysing the user behaviour on each variant the better variant is chosen and used thereafter.

Ques.70. What is concurrency testing?
Ans. Concurrency testing is a multi-user testing in which an application is evaluated by analyzing application's behaviour with concurent users acccessing the same functioanity.

Ques.71. What is all pair testing?
Ans. All pair testing is a type of testing in which the application is tested with all possible combination of the values of input parameters.

Ques.72. What is failover testing?
Ans. Failover testing is a type of testing that is used to verify application's ability to allocate more resources(more servers) in case of failure and transfering of the processing part to back-up system.

Ques.73. What is fuzz testing?
Ans. Fuzz testing is a type of testing in which large amount of random data is provided as input to the application in order to find security loopholes and other issues in the application.

Ques.74. What is UI testing?
Ans. UI or user interface testing is a type of testing that aims at finding Graphical User Interface defects in the application and checks that the GUI conforms to the specifications.

Ques.75. What is risk analysis?
Ans. Risk analysis is the analysis of the risk identified and assigning an appropriate risk level to it based on its impact over the application.

Ques.76. What is the difference between regression and retesting?
Ans. Regression testing is testing the application to verify that a new code change doesn't affect the other parts of the application. Whereas, in retesting we verify if the fixed issue is resolved or not.

Ques.77. What is the difference between blackbox and whitebox testing?
Ans. Blackbox testing is a type of testing in which internal architecture of the code is not required for testing. It is usaually applicable for system and acceptance testing.
Whereas whitebox testing requires internal design and implementation knowledege of the application being tested. It is usually applicable for Unit and Integration testing.

Ques.78. What is the difference between smoke and sanity testing?
Ans. The difference between smoke and sanity testing is-

- Smoke testing is a type of testing in which the all major functionalities of the application are tested before carrying out exhaustive testing. Whereas sanity testing is subset of regression testing which is carried out when there is some minor fix in application in a new build.
- In smoke testing shallow-wide testing is carried out while in sanity narrow-deep testing (for a particular fucntionality) is done.
- The smoke tests are usually documented or are automated. Whereas the sanity tests are generally not documented or unscripted.

Ques.79. What is code coverage?
Ans. Code coverage is the measure of the amount of code covered by the test scripts. It gives the idea of the part of the application covered by the test suite.

Ques.80. What is cyclomatic complexity?
Ans. Cyclomatic complexity is the measure of the number of independent paths in an application or program. This metric provides an indication of the amount of effort required to test complete functionality. It can be defined by the expression -
L – N + 2P, where:
L is the number of edges in the graph
N is the number of node
P is the number of disconnected parts

Ques.81. What is dynamic testing?
Ans. Testing performed by executing or running the application under test either manually or using automation.

Ques.82. What is an exit criteria?
Ans. An exit criteria is a formal set of conditions that specify the agreed upon features or state of application in order to mark the completion of the process or product.

Ques.83. What is traceability matrix?
Ans. In software testing a traceability matrix is a table that relates the high level requirements with detailed requirements, test plans or test cases in order to determine the completeness of the relationship.

Ques.84. What is pilot testing?
Ans. Pilot testing is a testing carried out as a trial by limited number of users evaluate the system and provide their feedback before the complete deployment is carried out.

Ques.85. What is backend testing?
Ans. Backend testing is a type of testing that invloves testing the backend of the system which comprises of testing the databases and the APIs in the application.

Ques.86. What are some advantages of automation testing?
Ans. Some advantages of automation testing are-

1. Test execution using automation is fast and saves considerable amount of time.
2. Carefully written test scripts remove the chance of human error during testing.
3. Tests execution can be scheduled for nightly run using CI tools like Jenkins which can also be configured to provide daily test results to relevant stakeholders.
4. Automation testing is very less resource intensive. Once the tests are automated, test execution requires almost no time of QAs. Saving Qa bandwidth for other explratory tasks.

Ques.87. What are some disadvantages of automation testing?
Ans. Some advantages of automation testing are-

1. It requries skilled automation testing experts to write test scritps.
2. Additional effort to write scripts is required upfront.
3. Automation scripts are limited to verification of the tests that are coded. These tests may miss some error that is very glaring and easily identifiable to human(manual QA).
4. Even with some minor change in application, script updation and maintenance is required.

Ques.88. What is mutation testing?
Ans. Mutation testing is a type of white box testing in which the source code of the application is mutated to cause some defect in its working. After that the test scripts are executed to check for their correctness by verifying the failures caused the mutant code.

Ques.89. What should be the psychology testing?

The two main stakeholders in software development life cycle - Testers and Developers have different mindsets while approaching an application. Testers tend to have a more stringent approach of examining the software. Most of the time they are looking to "break the application". Whereas, developers have the mindset to "make the application work".
ISTQB has defined certain psychological factors that influence the success of testing-

- Independence - Testers should enjoy a certain degree of independence while testing the application rather than following a straight path. This can be achieved by different apporaches like off-shoring the QA process, getting test strategies and other test plans by someone not following any sort of bias.
- Testing is often looked upon as destructive activity as it aims at finding flaws in system. But QA personnel should present testing as required and integral part, presenting it as constructive activity in overall sofware development lifecycle by mitigating the risks at early stages.
- More often than not, tester and developers are at the opposite end of spectrum. Testers need to have good interpersonal skills to communicate their findings without induging in any sort of tussle with the developers.

- All the communication and discussions should be focussed on facts and figures(risk analysis, priority setting etc). Emphasizing that the collaborative work of developers and testers will lead to better software.
- Empathy is one characterstic that definitely helps in testing. Testers empathizing with developers and other project stakeholders will lead to better relation between the two. Also, empathizing with end users will lead to a software with better usability of the product.
- Reference- ISTQB Foundation Level Syllabus - The Certified Tester Foundation Level in Software Testing

Ques.90. What is the difference between Testing and debugging?
Ans. Testing is the primarily performed by testing team in order to find the defects in the system. Whereas, debugging is an activity performed by development team. In debugging the cause of defect is located and fixed. Thus removing the defect and preventing any future occurrence of the defect as well.
Other difference between the two is - testing can be done without any internal knowledge of software architecture. Whereas debugging requires knowledge of the software architecture and coding.

Ques.91. Explain Agile methodology?
Ans. Agile methodology of software development is based on interative and increamental approach. In this model the application is broken down into smaller build on which different cross functional team work together providing rapid delivery along with adapting to changing needs at the same time.

Ques.92. What is scrum?
Ans. A scrum is a process for implementing Agile methodology. In scrum, time is divided into sprints and on completion of sprints, a deliverable is shipped.

Ques.98. What are the different roles in scrum?
Ans. The different roles in scrum are -

1. Product Owner - The product owner owns the whole development of the product, assign tasks to the team and act as an interface between the scrum team(development team) and the stakeholders.
2. Scrum Master - The scrum master monitors that scrum rules get followed in the team and conducts scrum meeting.
3. Scrum Team - A scrum team participate in the scrum meetings and perform the tasks assigned.

Ques.93. What is a scrum meeting?
Ans. A scrum meeting is daily held meeting in scrum process. This meeting is conducted by scrum master and update of previous day's work along with next day's task and context is defined in scrum.

Ques.94. Explain TDD (Test Driven Development).
Ans. Test Driven Development is a software development methodology in which the development of the software is driven by test cases created for the functionality to be implemented. In TDD first the test cases are created and then code to pass the tests is written. Later the code is refactored as per the standards.

Introduction to Selenium

The name **Selenium** might evoke the chemical element to some, but to developers and testers it brings to mind something quite different. They would probably think of a familiar open source software testing tool that has gained a lot of popularity in the decade of its existence. Many turn to Selenium testing solution as a first resource for testing web applications. Here's a quick look at Selenium and its components, and an attempt to understand why it is such an ideal choice for many.

Features of Selenium

Selenium began its story in 2004, when Jason Huggins developed it at Thoughtworks. It later moved on to become a collaborative project, also becoming open source along the way. Here are some of its key features that make it attractive to many testers:

Platform Independent: Whether you are a Windows, Linux or Mac user, Selenium will deploy seamlessly for your needs. According to the Selenium documentation, each new release is tested on Windows versions 7, 8 and 8.1. Windows XP was also supported while in use. Other versions are supported as well but tests are not run regularly using these.

Language Independent: Testers can choose to code in a range of languages when using Selenium for their testing purposes. These include **C#, Haskell, Java, JavaScript, Objective-C, Perl, PHP, Python, and Ruby**. This also enables the use of many frameworks using these languages including **JUnit and TestNG.**

Use of Multiple Browsers: Depending on the OS that you use in your testing environment, you might work with a variety of browsers including Firefox, Internet Explorer, Safari, Opera and Chrome. Many versions of these browsers are supported by Selenium, and the details of which can be found here.oin our webinar on **Selenium WebDriver Implementation using TestNG**

Why Selenium

The need for an automated testing tool like Selenium comes in due to the advantages of using automation for most testing environments. While intense testing has to be an integral phase for any product, the ease and speed of performing this testing could be a big differentiator in the product lifecycle.

Reduces manual errors: Performing the same set of cases over and over is a sure way of boredom and errors creeping in. In addition, it would later be tough to trace back and find the origin of the error. Using a tool that helps in automated testing is the ideal method to have a well-executed repetitive set of tests.

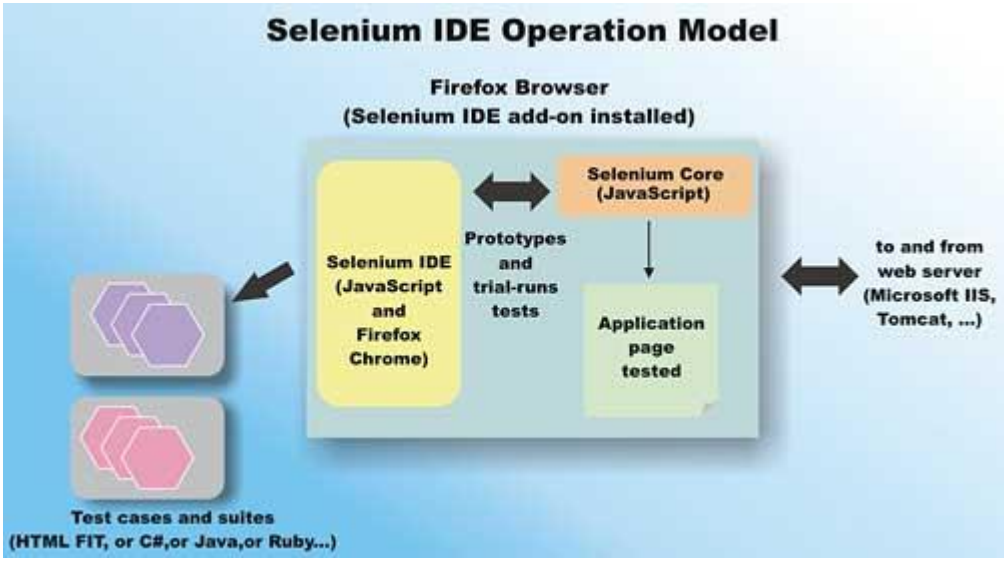
Increases speed of execution: Selenium can execute a big list of test cases much faster than any human tester can attempt to complete them. It can also help with parallel execution and quick repetition of test cases, saving a lot of time and money to the organization in the long run.

Open Source: Other than the list of features already mentioned, the open source nature of this testing tool is also a big reason to choose it. This is because the large number of users and the communities involved in developing and supporting Selenium could be a great resource to the amateur developer.

Selenium IDE

If your aim is to perform quick testing operations such as reproducing error or bug scenarios or to do exploratory testing, then Selenium IDE is the part of the Selenium testing solution that you should be looking at in detail. It is available as an extension or plug-in to the Firefox browser and includes all the functionality that is required for fast recording and playback of your test cases in the environment that you need.

Some of the advantages provided by Selenium IDE include:



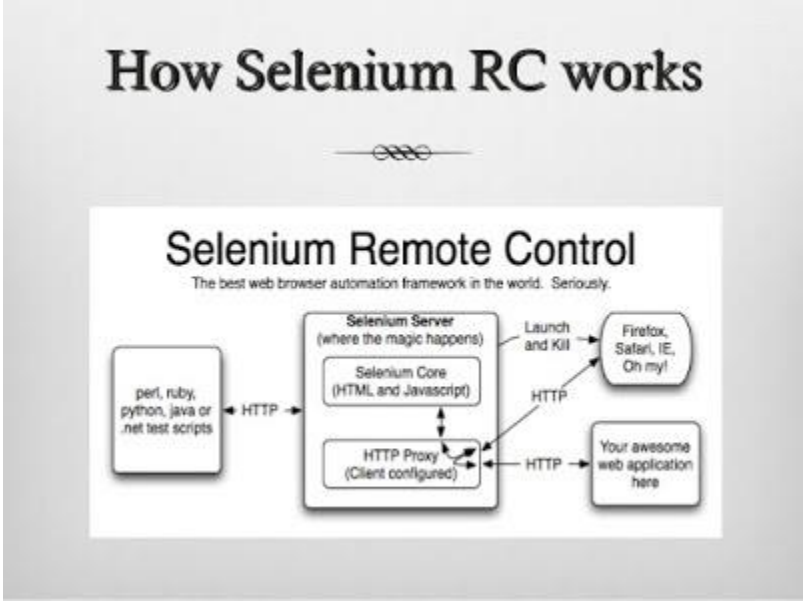
- Easy recording and playback of test cases with a very short learning curve for a new user.

- Possibility of editing test cases for experienced users with the use of commands in Selenium's own test language, Selenese. An autocomplete feature for all common commands is included in the functionality.
- Saving of tests as HTML, Ruby and many other formats.
- An extensive set of functions including intelligent field selection, walkthrough feature for tests, debugging including setting breakpoints and customization through the use of plugins.

The Selenium IDE environment allows easy editing of test cases, so that even a new user could start learning the syntax by studying the available list of commands and trying them out. As you continue to use the Selenium IDE environment, it would in fact attempt to predict the command and parameters that you might need for the UI element that is selected on screen.

Selenium RC

Selenium RC Server: Selenium RC is jar file which provides the functionality to launch the selenium test scripts on different – different browsers; it gets all command from selenium client and executes the scripts .By default the server is run at 4444.



For running the selenium scripts ,Selenium RC sever is run through the command prompt,, navigate the path where selenium rc server jar file is present and run the command "java –jar seleniumserver.jar "

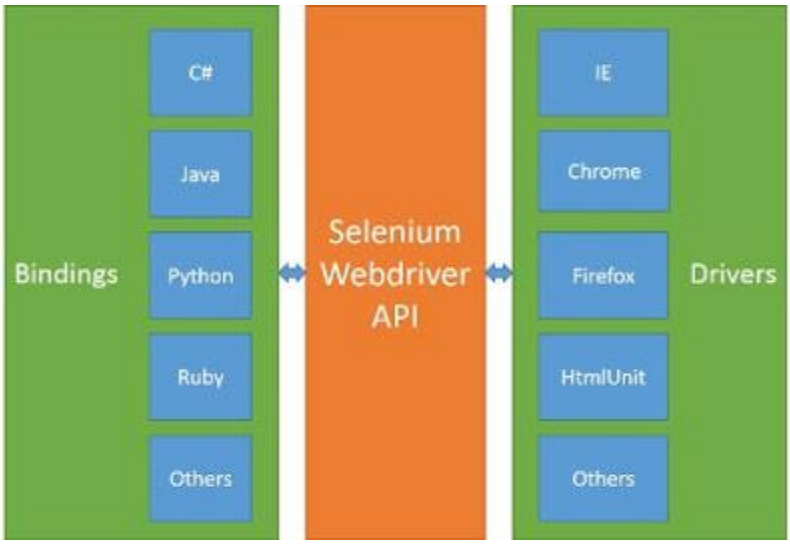
Selenium Client Server: Selenium client is nothing just the jar files or say language through which enhancement of the scripts is done and send the request to selenium rc server. For running the scripts, selenium rc is run through the command prompt.

Selenium Grid

Selenium Grid: Selenium Grid is most important aspect of the selenium. With the help of this, parallel execution can be done on different – different platform with different browser. It saves the time and increases the scope of the testing and provides the quality software product.

Selenium Webdriver

Selenium 2.0: Selenium 2.0 is the updated version of Selenium 1.0, it is also known as the Selenium WebDriver. It is completely based on the object oriented API so supposed to be more robust and can support in better way for the web based testing. ,Selenium IDE & Selenium Grid are working in similar way as implemented in selenium 1.0 however there is no need to run the selenium server for running the scripts. Selenium server is required when have to perform parallel execution through selenium grid or have to connect the remote machine.



WebDriver API makes the direct call to launch the browser using browser's native call. For every browser, browser's driver is present. Below is list for the same for Java.

Internet Explorer: WebDriver driver = new InternetExplorerDriver();
Firefox: FirefoxProfile profile = new FirefoxProfile();
Chrome : WebDriver driver = new ChromeDriver();
Opera : WebDriver driver = new OperaDriver();

Selenium WebDriver API also supports the Android and IOS operating system; It has separate API to interact with it.

Reporting: Reporting is very important aspect for testing, In Selenium, there is predefined framework like Junit, TestNG which generates the HTML reports and there are so many API through which customized report is generated. Selenium can integrate with test management tool like QC and Test link so report can be imported from these tools also.

Automation Framework: Automation framework is strategy for the automation, it provides the structured way for the automation for obtaining the quality work with quantity and effectiveness .There are many predefined framework in Selenium like Junit, TestNG, Nunit etc. however implementation of hybrid framework concept with any predefined framework make selenium more effective.

Keyword driven approach for selenium web driver is more effective because there are maximum 30 to 40 keywords which can be converted into the reusable function and can be used for testing any web based application.

Challenges and workaround: Selenium is for testing the web based applications so while doing web based testing, many times user has to interact with window component or window popup so selenium cannot automate that however Autoit is third party open source tool which can be used. At the run time, Objects are not identified with default ID or X-path than can use different xpath for the same object or different property like CSS, DOM, and Name etc.

Conclusion: Selenium is better option for the automation of the web based application. It can increase the scope of testing and provide more confidence to the customer. As well as that Regression and Smoke testing time will be condensed. I hope you enjoyed my short introduction to Selenium and that you found it useful.

Selenium Interview Questions

Prepare for selenium interview with our comprehensive list of over 100 interview questions. These interview questions are designed for both beginners and professionals. We will start with fairly simple questions and move to the more advanced level as the post progresses.

Ques.1. What is Selenium?

Ans. Selenium is a robust test automation suite that is used for automating web based applications. It supports multiple browsers, programming languages and platforms.

Ques.2. What are different forms of selenium?

Ans. Selenium comes in four forms-

1. *Selenium WebDriver* - Selenium WebDriver is used to automate web applications using browser's native methods.
2. *Selenium IDE* - A firefox plugin that works on record and play back principle.
3. *Selenium RC* - Selenium Remote Control(RC) is officially deprecated by selenium and it used to work on javascript to automate the web applications.
4. *Selenium Grid* - Allows selenium tests to run in parallel across multiple machines.

Ques.3. What are some advantages of selenium?

Ans. Following are the advantages of selenium-

1. Selenium is open source and free to use without any licensing cost.
2. It supports multiple languages like Java, ruby, python etc.
3. It supports multi browser testing.
4. It has good amount of resources and helping community over the internet.

5. Using selenium IDE component, non-programmers can also write automation scripts
6. Using selenium grid component, distributed testing can be carried out on remote machines possible.

Ques.4. What are some limitations of selenium?

Ans. Following are the limitations of selenium-

1. We cannot test desktop application using selenium.
2. We cannot test web services using selenium.
3. For creating robust scripts in selenium webdriver, programming langauge knowledge is required.
4. We have to rely on external libraries and tools forperforming tasks like - logging(log4J), testing framework-(testNG, JUnit), reading from external files(POI for excels) etc.

Ques.5. Which all browsers are supported by selenium webdriver?

Ans. Some commonly used browsers supported by selenium are-

1. Google Chrome - ChromeDriver
2. Firefox - FireFoxDriver
3. Internet Explorer - InternetExplorerDriver
4. Safari - SafariDriver
5. HtmlUnit (Headless browser) - HtmlUnitDriver
6. Android - Selendroid/Appium
7. IOS - ios-driver/Appium

Ques.6. Can we test APIs or web services using selenium webdriver?

Ans. No selenium webdriver uses browser's native method to automate the web applications. Since web services are headless, so we cannot automate web services using selenium webdriver.

Ques.7. What are the testing type supported by Selenium WebDriver?

Ans. Selenium webdriver can be used for performing automated functional and regression testing.

Ques.8. What are various ways of locating an element in selenium?

Ans. The different locators in selenium are-

1. Id

2. XPath
3. cssSelector
4. className
5. tagName
6. name
7. linkText
8. partialLinkText

Ques.9. What is an XPath?

Ans. Xpath or XML path is a query language for selecting nodes from XML documents. XPath is one of the locators supported by selenium webdriver.

Ques.10. What is an absolute XPath?

Ans. An absolute XPath is a way of locating an element using an XML expression beginning from root node i.e. html node in case of web pages. The main disadvantage of absolute xpath is that even with slightest change in the UI or any element the whole absolute XPath fails.
Example - html/body/div/div[2]/div/div/div/div[1]/div/input

Ques.11. What is a relative XPath?

Ans. A relative XPath is a way of locating an element using an XML expression beginning from anywhere in the HTML document. There are different ways of creating relative XPaths which are used for creating robust XPaths (unaffected by changes in other UI elements).
Example - //input[@id='username']

Ques.12. What is the difference between single slash(/) and double slash(//) in XPath?

Ans. In XPath a single slash is used for creating XPaths with absolute paths beginning from root node.
Whereas double slash is used for creating relative XPaths.

Ques.13. How can we inspect the web element attributes in order to use them in different locators?

Ans. Using Firebug or developer tools we can inspect the specific web elements.
Firebug is a plugin of firefox that provides various development tools for debugging applications. From automation perspective, firebug is used

specifically for inspecting web-elements in order to use their attributes like id, class, name etc. in different locators.

Ques.14. How can we locate an element by only partially matching its attributes value in Xpath?

Ans. Using contains() method we can locate an element by partially matching its attribute's value. This is particularly helpful in the scenarios where the attributes have dynamic values with certain constant part.

```
xPath expression = //*[contains(@name,'user')]
```

The above statement will match the all the values of name attribute containing the word 'user' in them.

Ques.15. How can we locate elements using their text in XPath?

Ans. Using the text() method -

```
xPathExpression = //*[text()='username']
```

Ques.16. How can we move to parent of an element using XPath?

Ans. Using '..' expression in XPath we can move to parent of an element.

Ques.17. How can we move to nth child element using XPath?

Ans. There are two ways of navigating to the nth element using XPath-

- Using square brackets with index position- Example - div[2] will find the second div element.
- Using position()- Example - div[position()=3] will find the third div element.

Ques.18. What is the syntax of finding elements by class using CSS Selector?

Ans. By .className we can select all the element belonging to a particluar class e.g. '.red' will select all elements having class 'red'.

Ques.19. What is the syntax of finding elements by id using CSS Selector?

Ans. By #idValue we can select all the element belonging to a particluar class e.g. '#userId' will select the element having id - userId.

Ques.20. How can we select elements by their attribute value using CSS Selector?

Ans. Using [attribute=value] we can select all the element belonging to a particluar class e.g. '[type=small]' will select the element having attribute type of value 'small'.

Ques.21. How can we move to nth child element using css selector?

Ans. Using :nth-child(n) we can move to the nth child element e.g. div:nth-child(2) will locate 2nd div element of its parent.

Ques.22. What is fundamental difference between XPath and css selector?

Ans. The fundamental difference between XPath and css selector is using XPaths we traverse up in the document i.e. we can move to parent elements. Whereas using CSS selector we can only move downwards in the document.

Ques.23. How can we launch different browsers in selenium webdriver?

Ans. By creating an instance of driver of a particular browser-

```
WebDriver driver = new FirefoxDriver();
```

Ques.24. What is the use of driver.get("URL") and driver.navigate().to("URL") command? Is there any difference between the two?

Ans. Both *driver.get("URL")* and *driver.navigate().to("URL")* commands are used to navigate to a URL passed as parameter. There is no difference between the two commands.

Ques.25. How can we type text in a textbox element using selenium?

Ans. Using sendKeys() method we can type text in a textbox-

```
WebElement searchTextBox = driver.findElement(By.id("search"));
searchTextBox.sendKeys("searchTerm");
```

Ques.26. How can we clear a text written in a textbox?

Ans. Using clear() method we can delete the text written in a textbox.

```
driver.findElement(By.id("elementLocator")).clear();
```

Ques.27. How to check a checkBox in selenium?

Ans. The same click() method used for clicking buttons or radio buttons can be used for checking checkbox as well.

Ques.28. How can we submit a form in selenium?

Ans. Using submit() method we can submit a form in selenium.

```
driver.findElement(By.id("form1")).submit();
```

Also, the click() method can be used for the same purpose.

Ques.29. Explain the difference between close and quit command.

Ans. *driver.close()* - Used to close the current browser having focus *driver.quit()* - Used to close all the browser instances

Ques.30. How to switch between multiple windows in selenium?

Ans. Selenium has *driver.getWindowHandles()* and *driver.switchTo().window("{windowHandleName}")* commands to work with multiple windows. The getWindowHandles() command returns a list of ids corresponding to each window and on passing a particular window handle to driver.switchTo().window("{windowHandleName}") command we can switch control/focus to that particular window.

```
for (String windowHandle : driver.getWindowHandles()) {
 driver.switchTo().window(handle);
}
```

Ques.31. What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in selenium?

Ans. driver.getWindowHandle() returns a handle of the current page (a unique identifier)
Whereas driver.getWindowHandles() returns a set of handles of the all the pages available.

Ques.32. How can we move to a particular frame in selenium?

Ans. The *driver.switchTo()* commands can be used for switching to frames.

```
driver.switchTo().frame("{frameIndex/frameId/frameName}");
```

For locating a frame we can either use the index (starting from 0), its name or Id.

Ques.33. Can we move back and forward in browser using selenium?

Ans. Yes, using *driver.navigate().back()* and *driver.navigate().forward()* commands we can move backward and forward in a browser.

Ques.34. Is there a way to refresh browser using selenium?

Ans. There a multiple ways to refresh a page in selenium-

- Using *driver.navigate().refresh()* command
- Using *sendKeys(Keys.F5)* on any textbox on the webpage
- Using driver.get("URL") on the current URL or using *driver.getCurrentUrl()*
- Using driver.navigate().to("URL") on the current URL or *driver.navigate().to(driver.getCurrentUrl());*

Ques.35. How can we maximize browser window in selenium?

Ans. We can maximize browser window in selenium using following command-

```
driver.manage().window().maximize();
```

Ques.36. How can we fetch a text written over an element?

Ans. Using *getText()* method we can fetch the text over an element.

```
String text = driver.findElement("elementLocator").getText();
```

Ques.37. How can we find the value of different attributes like name, class, value of an element?

Ans. Using *getAttribute("{attributeName}")* method we can find the value of different attrbutes of an element e.g.-

```
String valueAttribute =
driver.findElement(By.id("elementLocator")).getAttribute("value");
```

Ques.38. How to delete cookies in selenium?

Ans. Using deleteAllCookies() method-

```
driver.manage().deleteAllCookies();
```

Ques.39. What is an implicit wait in selenium?

Ans. An implicit wait is a type of wait which waits for a specified time while locating an element before throwing NoSuchElementException. As by default selenium tries to find elements immediately when required without any wait. So, it is good to use implicit wait. This wait is applied to all the elements of the current driver instance.

```
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

Ques.40. What is an explicit wait in selenium?

Ans. An explicit wait is a type of wait which is applied to a particular web element untill the expected condition specified is met.

```
WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("elemen
tId")));
```

Ques.41. What are some expected conditions that can be used in Explicit waits?

Ans. Some of the commonly used expected conditions of an element that can be used with expicit waits are-

- elementToBeClickable(WebElement element or By locator)
- stalenessOf(WebElement element)
- visibilityOf(WebElement element)
- visibilityOfElementLocated(By locator)
- invisibilityOfElementLocated(By locator)
- attributeContains(WebElement element, String attribute, String value)
- alertIsPresent()
- titleContains(String title)
- titleIs(String title)
- textToBePresentInElementLocated(By, String)

Ques.42. What is fluent wait in selenium?

Ans. A fluent wait is a type of wait in which we can also specify polling interval(intervals after which driver will try to find the element) along with the maximum timeout value.

```
Wait wait = new FluentWait(driver)

    .withTimeout(20, SECONDS)

    .pollingEvery(5, SECONDS)

    .ignoring(NoSuchElementException.class);

  WebElement textBox = wait.until(new Function<webdriver,webElement>() {

    public WebElement apply(WebDriver driver) {

    return driver.findElement(By.id("textBoxId"));

    }
}
);
```

Ques.43. What are the different keyboard operations that can be performed in selenium?

Ans. The different keyboard operations that can be performed in selenium are-

1. .sendKeys("sequence of characters") - Used for passing charcter sequesnce to an input or textbox element.
2. .pressKey("non-text keys") - Used for keys like control, function keys etc that ae non text.
3. .releaseKey("non-text keys") - Used in conjuntion with keypress event to simulate releasing a key from keyboard event.

Ques.44. What are the different mouse actions that can be performed?

Ans. The different mouse evenets supported in selenium are

1. click(WebElement element)
2. doubleClick(WebElement element)
3. contextClick(WebElement element)
4. mouseDown(WebElement element)
5. mouseUp(WebElement element)
6. mouseMove(WebElement element)
7. mouseMove(WebElement element, long xOffset, long yOffset)

Ques.45. Write the code to double click an element in selenium?

Ans. Code to double click an element in selenium-

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.doubleClick(element).perform();
```

Ques.46. Write the code to right click an element in selenium?

Code to right click an element in selenium-

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.contextClick(element).perform();
```

Ques.47. How to mouse hover an element in selenium?

Ans. Code to mouse hover over an element in selenium-

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.moveToElement(element).perform();
```

Ques.48. How to fetch the current page URL in selenium?

Ans. Using getCurrentURL() command we can fetch the current page URL-

```
driver.getCurrentUrl();
```

Ques.49. How can we fetch title of the page in selenium?

Ans. Using driver.getTitle(); we can fetch the page title in selenium. This method returns a string containing the title of the webpage.

Ques.50. How can we fetch the page source in selenium?

Ans. Using driver.getPageSource(); we can fetch the page source in selenium. This method returns a string containing the page source.

Ques.51. How to verify tooltip text using selenium?

Ans. Tooltips webelements have an attribute of type 'title'. By fetching the value of 'title' attribute we can verify the tooltip text in selenium.

```
String toolTipText = element.getAttribute("title");
```

Ques.52. How to locate a link using its text in selenium?

Ans. Using linkText() and partialLinkText() we can locate a link. The difference between the two is linkText matches the complete string passed as parameter to the link texts. Whereas partialLinkText matches the string parameter partially with the link texts.

```
WebElement link1 = driver.findElement(By.linkText(“artOfTesting”));
WebElement link2 = driver.findElement(By.partialLinkText(“artOf”));
```

Ques.53. What are DesiredCapabilities in selenium webdriver?

Ans. Desired capabilities are a set of key-value pairs that are used for storing or configuring browser specific properties like its version, platform etc in the browser instances.

Ques.54. How can we find all the links on a web page?

Ans. All the links are of anchor tag 'a'. So by locating elements of tagName 'a' we can find all the links on a webpage.

```
List<WebElement> links = driver.findElements(By.tagName("a"));
```

Ques.55. What are some commonly encountered exceptions in selenium?

Ans. Some of the commonly seen exception in selenium are-

- NoSuchElementException - When no element could be located from the locator provided.
- ElementNotVisibleException - When element is present in the dom but is not visible.
- NoAlertPresentException - When we try to switch to an alert but the targetted alert is not present.
- NoSuchFrameException - When we try to switch to a frame but the targetted frame is not present.
- NoSuchWindowException - When we try to switch to a window but the targetted window is not present.

- UnexpectedAlertPresentException - When an unexpected alert blocks normal interaction of the driver.
- TimeoutException - When a command execution gets timeout.
- InvalidElementStateException - When the state of an element is not appropriate for the desired action.
- NoSuchAttributeException - When we are trying to fetch an attribute's value but the attribute is not correct
- WebDriverException - When there is some issue with driver instance preventing it from getting launched.

Ques.56. How can we capture screenshots in selenium?

Ans. Using getScreenshotAs method of TakesScreenshot interface we can take the screenshots in selenium.

```
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("D:\\testScreenShot.jpg"));
```

Ques.57. How to handle dropdowns in selenium?

Ans. Using Select class-

```
Select countriesDropDown = new Select(driver.findElement(By.id("countries")));
dropdown.selectByVisibleText("India");

//or using index of the option starting from 0
dropdown.selectByIndex(1);

//or using its value attribute
dropdown.selectByValue("Ind");
```

Ques.58. How to check which option in the dropdown is selected?

Ans. Using isSelected() method we can check the state of a dropdown's option.

```
Select countriesDropDown = new Select(driver.findElement(By.id("countries")));
dropdown.selectByVisibleText("India");

//returns true or false value
System.out.println(driver.findElement(By.id("India")).isSelected());
```

Ques.59. How can we check if an element is getting displayed on a web page?

Ans. Using isDisplayed method we can check if an element is getting displayed on a web page.

```
driver.findElement(By locator).isDisplayed();
```

Ques.60. How can we check if an element is enabled for interaction on a web page?

Ans. Using isEnabled method we can check if an element is enabled or not.

```
driver.findElement(By locator).isEnabled();
```

Ques.61. What is the difference between driver.findElement() and driver.findElements() commands?

Ans. The difference between driver.findElement() and driver.findElements() commands is-

- findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.
- Syntax of findElement()- WebElement textbox = driver.findElement(By.id("textBoxLocator")); Syntax of findElements()- List <WebElement> elements = element.findElements(By.id("value"));
- Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements.

Ques.62. Explain the difference between implicit wait and explicit wait.?

Ans. An implicit wait, while finding an element waits for a specified time before throwing NoSuchElementException in case element is not found. The timeout value remains valid throughout the webDriver's instance and for all the elements.

```
driver.manage().timeouts().implicitlyWait(180, TimeUnit.SECONDS);
```

Whereas, Explicit wait is applied to a specified element only-

```
WebDriverWait wait = new WebDriverWait(driver, 5);
```

```
wait.until(ExpectedConditions.presenceOfElementLocated(ElementLocator));
```

It is advisable to use explicit waits over implicit waits because higher timeout value of implicit wait set due to an element that takes time to be visible gets applied to all the elements. Thus increasing overall execution time of the script. On the other hand, we can apply different timeouts to different element in case of explicit waits.

Ques.63. How can we handle window UI elements and window POP ups using selenium?

Ans. Selenium is used for automating Web based application only(or browsers only). For handling window GUI elements we can use AutoIT. AutoIT is a freeware used for automating window GUI. The AutoIt scripts follow simple BASIC lanaguage like syntax and can be easily integrated with selenium tests.

Ques.64. What is Robot API?

Ans. Robot API is used for handling Keyboard or mouse events. It is generally used to upload files to the server in selenium automation.

```
Robot robot = new Robot();

//Simulate enter key action
robot.keyPress(KeyEvent.VK_ENTER);
```

Ques.65. How to do file upload in selenium?

Ans. File upload action can be performed in multiple ways-

1. Using element.sendKeys("path of file") on the webElement of input tag and type file i.e. the elements should be like -

```
2.        <input type="file" name="fileUpload">
```

3. Using Robot API.
4. Using AutoIT API.

Ques.66. How to handle HTTPS website in selenium? or How to accept the SSL untrusted connection?

Ans. Using profiles in firefox we can handle accept the SSL untrusted connection certificate. Profiles are basically set of user preferences stored in a file.

```
FirefoxProfile profile = new FirefoxProfile();
profile.setAcceptUntrustedCertificates(true);
profile.setAssumeUntrustedCertificateIssuer(false);
WebDriver driver = new FirefoxDriver(profile);
```

Ques.67 How to do drag and drop in selenium?

Using Action class, drag and drop can be performed in selenium. Sample code-

```
Actions builder = new Actions(driver);
Action dragAndDrop = builder.clickAndHold(SourceElement)
.moveToElement(TargetElement)
.release(TargetElement)
.build();
dragAndDrop.perform();
```

Ques.68. How to execute javascript in selenium?

Ans. JavaScript can be executed in selenium using JavaScriptExecuter. Sample code for javascript execution-

```
WebDriver driver = new FireFoxDriver();
if (driver instanceof JavascriptExecutor) {
 ((JavascriptExecutor)driver).executeScript("{JavaScript Code}");
}
```

Ques.69. How to handle alerts in selenium?

Ans. In order to accept or dismiss an alert box the alert class is used. This requires first switching to the alert box and than using accept() or dismiss() command as the case may be.

```
Alert alert = driver.switchTo().alert();
//To accept the alert
alert.accept();

Alert alert = driver.switchTo().alert();
//To cancel the alert box
alert.dismiss();
```

Ques.70. What is HtmlUnitDriver?

Ans. HtmlUnitDriver is the fastest WebDriver. Unlike other drivers (FireFoxDriver, ChromeDriver etc), the HtmlUnitDriver is non-GUI, while running no browser gets launched.

Ques.71. How to handle hidden elements in Selenium webDriver?

Ans. Using javaScript executor we can handle hidden elements-
(JavascriptExecutor(driver))
.executeScript("document.getElementsByClassName(ElementLocator).click()
;");

Ques.72. What is Page Object Model or POM?

Ans. Page Object Model(POM) is a design pattern in selenium. A design pattern is a solution or a set of standards that are used for solving commonly occuring software problems.
Now coming to POM - POM helps to create a framework for maintaining selenium scripts. In POM for each page of the application a class is created having the web elements belonging to the page and methods handling the events in that page. The test scripts are maintained in seperate files and the methods of the page object files are called from the test scripts file.

Ques.73. What are the advantages of POM?

Ans. The advantages are POM are-

1. Using POM we can create an Object Repository, a set of web elements in seperate files along with their associated functions. Thereby keeping code clean.
2. For any chnage in UI(or web elements) only page object files are required to be updated leaving test files unchanged.
3. It makes code reusable and maintable.

Ques.74. What is Page Factory?

Ans. Page factory is an implementation of Page Object Model in selenium. It provides @FindBy annotation to find web elements and PageFactory.initElements() method to initialize all web elements defined with @FindBy annotation.

```
public class SamplePage {

    WebDriver driver;
```

```
    @FindBy(id="search")
    WebElement searchTextBox;

    @FindBy(name="searchBtn")
    WebElement searchButton;

    //Constructor
    public samplePage(WebDriver driver){
    this.driver = driver;
    //initElements method to initialize all elements
    PageFactory.initElements(driver, this);
    }

    //Sample method
    public void search(String searchTerm){
    searchTextBox.sendKeys(searchTerm);
    searchButton.click();
    }
}
```

Ques.75. What is an Object repository?

Ans. An object repository is centralized location of all the object or WebElements of the test scripts. In selenium we can create object repository using Page Object Model and Page Factory design patterns.

Ques.76. What is a data driven framework?

Ans. A data driven framework is one in which the test data is put in external files like csv, excel etc separated from test logic written in test script files. The test data drives the test cases, i.e. the test methods run for each set of test data values. TestNG provides inherent support for data driven testing using @dataProvider annotation.

Ques.77. What is a keyword driven framework?

Ans. A keyword driven framework is one in which the actions are associated with keywords and kept in external files e.g. an action of launching a browser will be associated with keyword - launchBrowser(), action to write in a textbox with keyword - writeInTextBox(webElement, textToWrite) etc. The code to perform the action based on a keyword specified in external file is implemented in the framework itself.
In this way the test steps can be written in a file by even a person of non-programming background once all the identified actions are implemented.

Ques.78. What is a hybrid framework?

Ans. A hybrid framework is a combination of one or more frameworks. Normally it is associated with combination of data driven and keyword driven frameworks where both the test data and test actions are kept in external files(in the form of table).

Ques.79. What is selenium Grid?

Ans. Selenium grid is a tool that helps in distributed running of test scripts across different machines having different browsers, browser version, platforms etc in parallel. In selenium grid there is hub that is a central server managing all the distributed machines known as nodes.

Ques.80. What are some advantages of selenium grid?

Ans. The advantages of selenium grid are-

1. It allows running test cases in parallel thereby saving test execution time.
2. Multi browser testing is possible using selenium grid by running the test on machines having different browsers.
3. It is allows multi-platform testing by configuring nodes having different operating systems.

Ques.81. What is a hub in selenium grid?

Ans. A hub is server or a central point in selenium grid that controls the test executions on the different machines.

Ques.82. What is a node in selenium grid?

Ans. Nodes are the machines which are attached to the selenium grid hub and have selenium instances running the test scripts. Unlike hub there can be multiple nodes in selenium grid.

Ques.83. Explain the line of code Webdriver driver = new FirefoxDriver();.

Ans. In the line of code ***Webdriver driver = new FirefoxDriver();*** 'WebDriver' is an interface and we are creating an object of type WebDriver instantiating an object of FirefoxDriver class.

Ques.84 What is the purpose of creating a reference variable- 'driver' of type WebDriver instead of directly creating a FireFoxDriver object or any other driver's reference in the statement Webdriver driver = new FirefoxDriver();?

Ans. By creating a reference variable of type WebDriver we can use the same variable to work with multiple browsers like ChromeDriver, IEDriver etc.

Ques.85. What is testNG?

Ans. TestNG(NG for Next Generation) is a testing framework that can be integrated with selenium or any other automation tool to provide multiple capabilities like assertions, reporting, parallel test execution etc.

Ques.86. What are some advantages of testNG?

Ans. Following are the advantages of testNG-

1. TestNG provides different assertions that helps in checking the expected and actual results.
2. It provides parallel execution of test methods.
3. We can define dependency of one test method over other in TestNG.
4. We can assign priority to test methods in selenium.
5. It allows grouping of test methods into test groups.
6. It allows data driven testing using @DataProvider annotation.
7. It has inherent support for reporting.
8. It has support for parameterizing test cases using @Parameters annotation.

Ques.87. What is the use of testng.xml file?

Ans. testng.xml file is used for configuring the whole test suite. In testng.xml file we can create test suite, create test groups, mark tests for parallel execution, add listeners and pass parameters to test scripts. Later this testng.xml file can be used for triggering the test suite.

Ques.88. How can we pass parameter to test script using testNG?

Ans. Using @Parameter annotaion and 'parameter' tag in testng.xml we can pass paramters to test scripts. Sample testng.xml -

```
<suite name="sampleTestSuite">
   <test name="sampleTest">
  <parameter name="sampleParamName" value="sampleParamValue"/>
  <classes>
   <class name="TestFile" />
  </classes>
   </test>
</suite>
```

Sample test script-

```
public class TestFile {
   @Test
   @Parameters("sampleParamName")
   public void parameterTest(String paramValue) {
  System.out.println("Value of sampleParamName is - " + sampleParamName);
   }
```

Ques.89. How can we create data driven framework using testNG?

Ans. Using @DataProvider we can create a data driven framework in which data is passed to the associated test method and multiple iteration of the test runs for the different test data values passed from the @DataProvider method. The method annotated with @DataProvider annotation return a 2D array of object.

```
//Data provider returning 2D array of 3*2 matrix
 @DataProvider(name = "dataProvider1")
   public Object[][] dataProviderMethod1() {
  return new Object[][] {{"kuldeep","rana"}, {"k1","r1"},{"k2","r2"}};
   }

   //This method is bound to the above data provider returning 2D array of 3*2 matrix
   //The test case will run 3 times with different set of values
   @Test(dataProvider = "dataProvider1")
   public void sampleTest(String s1, String s2) {
  System.out.println(s1 + " " + s2);
   }
```

Ques.90. What is the use of @Listener annotation in TestNG?

Ans. TestNG provides us different kind of listeners using which we can perform some action in case an event has triggered. Usually testNG listeners are used for configuring reports and logging. One of the most widely used lisetner in testNG is ITestListener interface. It has methods like onTestSuccess, onTestFailure, onTestSkipped etc. We need to implement this interface creating a listner class of our own. After that using the *@Listener annotation we can use specify that for a particular test class our customized listener class should be used.*

```
@Listeners(PackageName.CustomizedListenerClassName.class)

public class TestClass {
    WebDriver driver= new FirefoxDriver();@Test
    public void testMethod(){
    //test logic
    }
}
```

Ques.91. What is the use of @Factory annotation in TestNG?

Ans. @Factory annotation helps in dynamic execution of test cases. Using @Factory annotation we can pass parameters to the whole test class at run time. The parameters passed can be used by one or more test methods of that class.
Example - there are two classes TestClass and the TestFactory class. Because of the @Factory annotation the test methods in class TestClass will run twice with the data "k1" and "k2"

```
public class TestClass{
    private String str;

    //Constructor
    public TestClass(String str) {
    this.str = str;
    }

    @Test
    public void TestMethod() {
    System.out.println(str);
    }
}

public class TestFactory{
    //The test methods in class TestClass will run twice with data "k1" and "k2"
    @Factory
    public Object[] factoryMethod() {
    return new Object[] { new TestClass("K1"), new TestClass("k2") };
    }
```

```
}
```

Ques.92. What is difference between @Factory and @DataProvider annotation?

Ans. @Factory method creates instances of test class and run all the test methods in that class with different set of data. Whereas, @DataProvider is bound to individual test methods and run the specific methods multiple times.

Ques.93. How can we make one test method dependent on other using TestNG?

Ans. Using dependsOnMethods parameter inside @Test annotation in testNG we can make one test method run only after successful execution of dependent test method.

```
@Test(dependsOnMethods = { "preTests" })
```

Ques.94. How can we set priority of test cases in TestNG?

Ans. Using priority parameter in @Test annotation in TestNG we can define priority of test cases. The default priority of test when not specified is integer value 0. Example-

```
@Test(priority=1)
```

Ques.95. What are commonly used TestNG annotations?

Ans. The commonly used TestNG annotations are-

- @Test- @Test annotation marks a method as Test method.
- @BeforeSuite- The annotated method will run only once before all tests in this suite have run.
- @AfterSuite-The annotated method will run only once after all tests in this suite have run.
- @BeforeClass-The annotated method will run only once before the first test method in the current class is invoked.
- @AfterClass-The annotated method will run only once after all the test methods in the current class have been run.

- @BeforeTest-The annotated method will run before any test method belonging to the classes inside the <test> tag is run.
- @AfterTest-The annotated method will run after all the test methods belonging to the classes inside the <test> tag have run.

Ques.96. What are some common assertions provided by testNG?

Ans. Some of the common assertions provided by testNG are-

1. assertEquals(String actual, String expected, String message) - (and other overloaded data type in parameters)
2. assertNotEquals(double data1, double data2, String message) - (and other overloaded data type in parameters)
3. assertFalse(boolean condition, String message)
4. assertTrue(boolean condition, String message)
5. assertNotNull(Object object)
6. fail(boolean condition, String message)
7. true(String message)

Ques.97. How can we run test cases in parallel using TestNG?

Ans. In order to run the tests in parallel just add these two key value pairs in suite-

- parallel="{methods/tests/classes}"
- thread-count="{number of thread you want to run simultaneously}".

```
<suite name="ArtOfTestingTestSuite" parallel="methods" thread-count="5">
```

Check Running Selenium Tests in parallel for details.

Ques.98. Name an API used for reading and writing data to excel files.

Ans. Apache POI API and JXL(Java Excel API) can be used for reading, writing and updating excel files.

Ques.99. Name an API used for logging in Java.

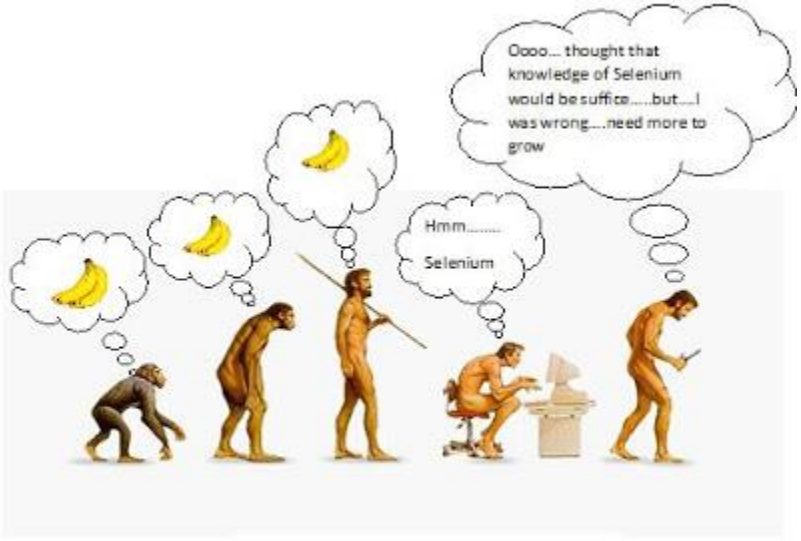
Ans. Log4j is an open source API widely used for logging in Java. It supports multiple levels of logging like - ALL, DEBUG, INFO, WARN, ERROR, TRACE and FATAL.

Ques.100. What is the use of logging in automation?

Ans. Logging helps in debugging the tests when required and also provides a storage of test's runtime behaviour.

Introduction to Appium
Appium tutorial – An Introduction

Developing a mobile app is just the beginning. To make sure your app is running fine over 3 billion strong Smartphone carrying audience using all types of devices, mobile operating systems, and new and old versions of everything. That takes a lot of mobile testing. The only way to cover it all is with automation testing and in this introductory Appium tutorial post, we will introduce to Appium, Design concepts in Appium and Appium Architecture.



It's no surprise, then, that developing, testing, and releasing quality mobile apps quickly and efficiently is a high priority for most organizations—as it should be! Fortunately, mobile development tools are evolving to meet these needs, with modern development teams having access to better test automation than ever before. Say Hello to Appium, the most popular automation testing tool for mobile software testing.

1 Hello Appium

Appium, an open-source tool for automating mobile applications on iOS and Android platforms has been around in one form or another since 2012. Since its early days, Appium has grown and matured significantly.Go ahead and get started.

In January 2013, Sauce Labs decided to fully back Appium and provide more developer power. Since then, Appium has been given various awards and is now the most popular open-source cross-platform mobile automation framework.

Stability of the tool has improved, bugs prioritized and fixed, and features have been added. Sauce Labs has now increased the number of developers it donates to working on Appium, but the entire community is involved in guiding the project and contributing to it, and project governance happens in the open, on public mailing lists and GitHub's issue tracker.

Appium allows native, hybrid and web application testing and supports automation test on physical devices as well as on emulator or simulator both.

2 Design Concepts of Appium

Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

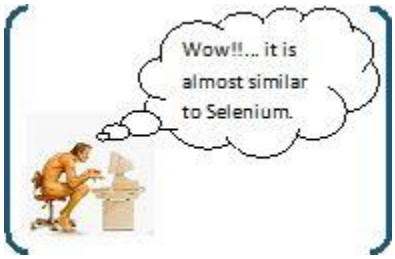
1. **You shouldn't have to recompile your app or modify it in any way.** Say if you have to modify your app to automate it, you are at risk. Because you tested one version of the app (modified one) and you are submitting another version to app stores. Appium allows you to test the same app (without any modifications) that you will be submitting to app stores.
2. **You shouldn't be locked into a specific language or framework to write and run your tests.** If a tool limits you to write tests in a particular language then you might use it or NOT, based on many factors, including but not limited to language familiarity, developers support, online help etc. Appium gives the flexibility to write automated test in almost any language (Java, C#, Ruby, Python etc.)
3. **A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.** Selenium has become a standard for web automation. So if you are already familiar with Selenium, you know Appium
4. **A mobile automation framework should be open source, in spirit and practice as well as in name!**. This is pretty self-explanatory.

So how does the structure of the Appium project live out the philosophy that we discussed above?

- Requirement #1 is met by using vendor-provided automation frameworks under the hood. That way, one don't need to compile in any Appium-specific or third-party code or frameworks to your app. This means **you're testing the same app you're shipping**. The vendor-provided frameworks used in Appium are:

    1. iOS: Apple'sUIAutomation
    2. Android 4.2+: Google'sUiAutomator
    3. Android 2.3+: Google'sInstrumentation. (Instrumentation support is provided by bundling a separate project, Selendroid)

- Requirement #2 is met by wrapping the vendor-provided frameworks in one API, theWebDriver WebDriver (aka "Selenium WebDriver") specifies a client-server protocol (known as the JSON Wire Protocol). Given this client-server architecture, a client written in any language can be used to send the appropriate HTTP requests to the server. There are already clients written in every popular programming language that you can find at http://appium.io/downloads. This also means that you're free to use whatever test runner and test framework you want; the client libraries are simply HTTP clients and can be mixed into your code any way you please. In other words, Appium & WebDriver clients are not technically "test frameworks" — they are "automation libraries". You can manage your test environment any way you like!



- Requirement #3 is met in the same way: WebDriver has become the de facto standard for automating web browsers. Appium developers have extended the protocol with extra API methods useful for mobile automation.
- Requirement #4 is self-explanatory.

3 Internal Architecture

Appium at its core is a webserver written in Node.js that exposes a REST API. The Appium server does the following.

- Receives connections from a client and creates a session
- Listens to the commands
- Executes the commands (on real devices or emulators)
- Responds back with a HTTP response representing the results of the executed commands

As per the figure, the full cycle will be something like

1. You write your tests using one of Appium client libraries (which supports Appium's extensions to the Webdriver protocol)
2. Your tests calls the Webdriver API
3. The Webdriver sends the request in form of JSon via http request to the Appium server. (Your Webdriver scripts and Appium server can be on same machine or different machines, depending how you want it to be)
4. The Appium server, under the hood invokes vendor specific mechanisms to execute the test commands
5. The client (devices or emulators) responds back to Appium server
6. Appium server logs the results in console.

That's pretty much it about Appium architecture. There are a couple of additional concepts (like session and desired capabilities), we will explain them when we actually start setting up Appium and do hands on.

4 Appium's Pros and Cons:

Atlast, I want to pin point few of the pros and cons of using Appium as Mobile Apps Automation Testing Tool.

Pros:

- The beauty of Appium is that, all the complexities are under the hood of Appium server and for an automation developer the programming language and the whole experience would remain same irrespective of the platform he is automating (iOS or Android).
- The other benefits of Appium is that it opens the door to cross-platform mobile testing which means the same test would work on multiple platforms.
- Unlike other tools Appium doesn't require you to include some extra agents in your app to make it automation friendly. It believes in the philosophy of testing the same app which we are going to submit in the app store.
- It is developed and supported by Sauce Labs and it is getting picked really fast with in the WebDriver community for mobile automation.
- It can automate Web, Hybrid and Native mobile applications.

Cons:

- Scaling up is an important consideration with Continuous Integration and Appium comes across as a great tool to fulfill this expectation. The reason for this is a technical limitation, in iOS we can only run one instance on Instruments per Mac OS so we can only run our iOS scripts on one device per mac machine. So if we want to run our tests on multiple iOS devices at the same time then we would need to arrange the same number of Mac machines, which would be costly affair. But this limitation can be resolved if we execute our scripts in Sauce Lab's mobile cloud which at present supports running scripts on multiple iOS simulators at the same time.
- Appium uses UIAutomator for Android automation which only supports Android SDK Platform, API 16 or higher so to support the older APIs they have used another open source library called Selendroid. So I would not say it as a limitation but it is definitely an overhead on the configuration side.

Friday, 4 August 2017

Appium interview question session 1

1.What are the Advantages of using Appium?

- It allows you to write tests against multiple mobile platforms using the same API.
- You can write and run your tests using any language or test framework.
- It is an open-source tool that you can easily contribute to.

2. Do I need Appium?

The answer to such a question is always: “It depends on what you need!”. So the actual question becomes: “Which conditions make Appium suitable for me?”. The most important assumption is that you are developing apps (pretty obvious I know). If you are developing an app for a specific platform (and have no intention of supporting others in future), Appium is not really required and this is basically the answer you are looking for. Appium becomes meaningful only when you have apps targeting more than one platform (Windows, Android or iOS to cite some). Appium becomes essential if you have a webview-based app (necessarily) targeting many platforms out there.

3.How difficult is it to set up a working environment?

The assumption is that Appium comes with a not-so-tiny documentation, so users are not really left alone. However it is not so straightforward to set up Appium to work on a Windows or Mac machine (did not try on Unix so far). In my experience, instead of installing the GUI-based application, it is much better to install the command-line application (which is released more often). Also beware [sudo], as Appium will surely bite you back late in time if you installed it as a [superuser] (this is probably the clearest point in the documentation)

4.What is Appium’s strongest point?

Appium is based on Selenium which is an HTTP protocol by Google designed to automate browsers. The idea is actually very nice as automating an app (especially a webview-based one) is not so different (in terms of required APIs) from automating a browser. Appium is also designed to encourage a 2-tier architecture: a machine runs the test written in one language ([csharp], [ruby], [javascript] are only a few among the many supported ones) and another one (the test server) actually executes it. Furthermore the WebDriver protocol targets scalability (because based on HTTP), this makes Appium very scalable as well; remember that you will need to **write your test once,** Appium will be in charge of executing it on more platforms.

5.What is Appium’s weakest point?

Open source software is great, however it naturally comes with some downsides which nobody is to be blamed for: **unreliability** is probably one of the most undeniable. My test suites need to run many tests every day and those tests must be stable (it means they should fail only when the product, or the test has a defect). However it took me much time to
build layers on top of Appium to make my tests stable. This is especially true when it comes to synchronizing your tests with the automation: your test issues a command which translates into an automation command by Appium causing an interaction on the device; a command returns when after it has been issued, not when the interaction is actually over! For this reason you will probably find yourself adding delays to your tests (if you are doing things in the wrong/most-straightforward way) or developing synchronization layers on top of your test APIs (the good approach).

6.What is the Appium Philosophy?

- R1. Test the same app you submit to the marketplace
- R2. Write your tests in any language, using any framework
- R3. Use a standard automation specification and API
- R4. Build a large and thriving open-source community effort

7.Why do the Appium clients exist?

We have the Appium clients for 3 reasons:

- 1) There wasn't time to go through a full commit and release cycle for Selenium once we'd set a release date for 1.0
- 2) Some of the things that Appium does, and which its users really find useful, are never going to be an official part of the new mobile spec. We want a way to make these extensions available
- 3) There are some behaviors whose state is as yet unknown. They might make it into the spec and get deleted from the clients, or they might be in category #2

Ultimately, the only reason for the clients will be #2. And even that is actually evidence that we are conforming to the WebDriver spec (by implementing the extension strategy it recommends) rather than departing from it. The Appium clients are the easiest and cleanest way to use Appium.

8.Explain what is Appium?

Appium is a freely distributed open source mobile application UI testing framework.

9.What are main Advantages of using Appium on Sauce Labs?

- You save the time it takes to set up the Appium server locally.
- You don't have to install/configure the mobile emulators/simulators in your local environment.
- You don't have to make any modifications to the source code of your application.
- You can start scaling your tests instantly.

10.Which language should I use to write my tests?

This is probably the best thing about Appium: you can write your tests in **any language.** Since Appium is nothing more than an HTTP server, a test which needs to be interfaced with Appium can simply use HTTP libraries to create HTTP sessions. You just need to know the Selenium protocol in order to compose the right commands and that's it!
However, as you can imagine, there are already some libraries doing this for the most common languages and development frameworks out there: C#, [dotnet], [java], Ruby, [python] and Javascript are just few examples; and they all are open source

11.What type of tests are suitable for Appium?
When it comes to testing, especially webview-based apps, there are a lot of scenarios that can be tested also depending on the feature coverage you want to ensure. Appium is pretty handy for testing scenarios that users will go through when using your app. But if you need to test more than UX simple interactions, then Appium will become a limitation. Think about features like keyboarding. It is not so easy when complex touch/keyboard mixed scenarios are involved, the probability of a false failure is high; do not misunderstand me on this: I am not saying it is impossible to do, just not so easy as you might think!
Another little nightmare with Appium is **exchanging data.** When your test needs to exchange data with your app (especially in the incoming direction), you will need to play some tricks. So always consider that sending and receiving information is not that straightforward. It is not Appium's fault, the WebDriver specification was designed for automating stuff, not exchanging data!

12.Can Appium be used for all my tests?
This is an implied question in this question. The answer is **No** (in general). As I said before Appium is not suitable for all types of tests you might want to write (this depends on the functionalities you need to cover). There are some scenarios that can be difficult to test and some of them are so platform specific that you will need to write some suites just for Android or iOS for example. Remember that you can always get to do something no matter how hard it is, so you can test all your difficult scenarios using Appium, but always keep in mind one question: is it worth the time and the pain? Having Appium testing some scenarios leaving a few tests to other approaches is fine too! World is not black and white!

13.List out the Appium abilities?
Appium abilities are

- Test Web
- Provides cross-platform for Native and Hybrid mobile automation
- Support JSON wire protocol
- It does not require recompilation of App
- Support automation test on physical device as well as similar or emulator both
- It has no dependency on mobile device

14.List out the pre-requisite to use APPIUM?
Pre-requisite to use APPIUM is

- ANDROID SDK
- JDK
- TestNG
- Eclipse
- Selenium Server JAR
- Webdriver Language Binding Library
- APPIUM for Windows
- APK App Info On Google Play
- Js

15.What is Appium's most considerable limitation?
Hand down my chin starting thinking and mumbling. If I had to provide one single thing you should be aware of about Appium before starting using it, it would surely be: **multiple session handling.** Since Appium is a server, it serves HTTP requests; you might have two different computers running a test each against the same Appium server: what happens? As for now, Appium does not support this scenario and the second test will be aborted. This is a considerable limitation, because no queuing system comes with Appium. If you need to support multiple sessions, you will need to implement this feature by yourself.

16.How active is Appium?
Appium is available on GitHub and there you can find all you need. The Appium team is responsible for developing many different subsystems revolving around Appium (like APIs for different languages), thus I

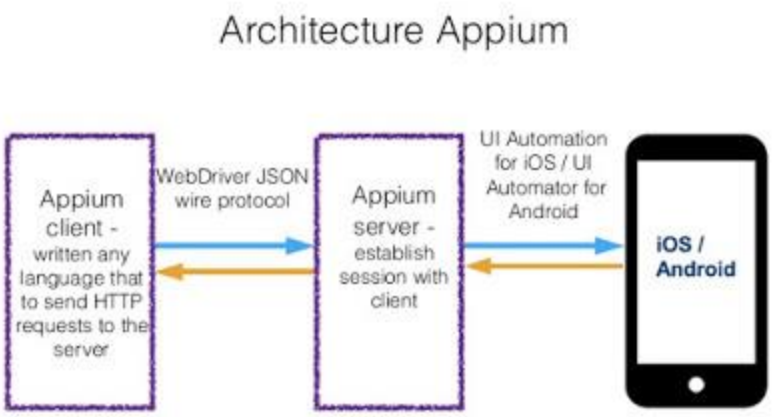
can tell you that this product is alive and very active. The team is also pretty well responsive and once you open an issue you will find a reply after no more than 36 hours (this ETA comes by my personal experience). The community around Appium is also pretty large and growing every month.

17.What about performance?
Appium is not a huge application and **requires very little memory.** Its architecture is actually pretty simple and light as Appium acts like a proxy between your test machine and each platform automation toolkit. Once up and running, Appium will listen to HTTP requests from your tests; when a new session is created, a component in Appium's Node.js code called _proxy_ will forward these Selenium commands to active platform drivers. In the case of Android for example, Appium will forward incoming commands to the [chromedriver] (90% of cases, Appium will not even change commands while routing them), this happens because ChromeDriver supports WebDriver and Selenium. For this reason Appium will not allocate much memory itself, you will see a lot of memory being allocated by other processes like [adb], ChromeDriver or the iOS automation toolkit (called by Appium while testing and automating).

18. Which approach is the best? Testing on real devices or simulators/emulators?
This is a tough question because both options offer different levels of testability and flexibility when testing. There are also many problems associated with each. So my answer will be again: "It depends on your needs!".
Running test on a device is, always in my opinion, the best solution because it offers a testing environment completely aligned with the running environment: tests run on those devices where your apps will be used once published on stores. However devices must be connected to the Appium server via USB at least, and this is not always a very nice thing. ADB has a known issue for which a device disconnects after a while (even though it remained plugged all the time): because of this your tests might fail after a while and Appium will report that a device could not be found! I had to write a component which resets ADB after some time so that devices will not disconnect.

19.Tests on emulators or simulators?
On the other hand emulators/simulators will never disconnect from Appium. They also offer nice options like the ability of choosing the orientation or other hardware-related configurations. However your tests will run much slower (sadly, my tests ran 3 times slower) and do expect some crazy behavior from the Android emulator which sometimes shuts down unexpectedly. Another problem is that emulators tend to allocate a lot of memory.

20.What platforms are supported?
Appium currently supports **Android** and **iOS**, no support for Windows unfortunately.



21. Do I need a server machine to run tests on Appium?
No! Appium promotes a 2-tier architecture where a test machine connects to a test server running Appium and automating the whole thing. However this configuration is not mandatory, you can have Appium running on the same machine where your test runs. Instead of connecting to a remote host, your test will connect to Appium using the loopback address.

22.List out the limitations of using Appium?

- Appium does not support testing of Android Version lower than 4.2
- Limited support for hybrid app testing. E.g., not possible to test the switching action of application from the web app to native and vice-versa
- No support to run Appium Inspector on Microsoft Windows

23.How can I test Android tablets?
The best way to test on different Android emulators screen sizes is by using the different Android Emulator Skins . For instance, if you use our Platforms Configurator you'll see the available skins for the different Android versions (e.g Google Nexus 7 HD, LG Nexus 4, Samsung Galaxy Nexus, Samsung Galaxy S3, etc). Some of these skins are tablets, for example the Google Nexus 7C is a tablet which has a very large resolution and very high density.

24.How can I run manual tests for my mobile native app or mobile hybrid app?
Sauce Labs doesn't support manual tests for mobile native app or mobile hybrid app tests.

25.What type of keyboard and buttons do the Android emulators have?
Android Emulators have software buttons and a hardware keyboard. In a regular Android emulator the device buttons are software buttons displayed on the right size of the emulator. For the Android emulators with different skins (e.g Google Nexus 7 HD, LG Nexus 4, Samsung Galaxy Nexus, Samsung Galaxy S3, etc) the device buttons are also software buttons that are overplayed on top of the skin. For instance, if you hover the mouse around the edges of any of our Android emulators with an specified skin, a hover icon will appear and you should be able to find whatever buttons actually exist on the device that the skinned emulator is trying to emulate (e.g power button along the top, volume buttons along the edge, back/home buttons right below the screen, etc).

26.Explain how to find DOM element or xPath in a mobile application?

To find the DOM element use “UIAutomateviewer” to find DOM element for Android application.

27.Explain the design concept of Appium?

- Appium is an “HTTP Server” written using Node.js platform and drives iOS and Android session using Webdriver JSON wire protocol. Hence, before initializing the Appium Server, Node.js must be pre-installed on the system
- When Appium is downloaded and installed, then a server is setup on our machine that exposes a REST API
- It receives connection and command request from the client and execute that command on mobile devices (Android / iOS)
- It responds back with HTTP responses. Again, to execute this request, it uses the mobile test automation frameworks to drive the user interface of the apps. Framework like
- Apple Instruments for iOS (Instruments are available only in Xcode 3.0 or later with OS X v10.5 and later)
- Google UIAutomator for Android API level 16 or higher
- Selendroid for Android API level 15 or less
- **What language does Appium support?**
- Appium support any language that support HTTP request like Java, JavaScript with Node.js, Python, Ruby, PHP, Perl, etc.

28.Explain the pros and cons of Appium?

Pros:

- For programmer irrespective of the platform, he is automating (Android or iOS) all the complexities will remain under single Appium server
- It opens the door to cross-platform mobile testing which means the same test would work on multiple platforms
- Appium does not require extra components in your App to make it automation friendly
- It can automate Hybrid, Web and Native mobile applications

Cons:

- Running scripts on multiple iOS simulators at the same time is possible with Appium
- It uses UIAutomator for Android Automation which supports only Android SDK platform, API 16 or higher and to support the older API’s they have used another open source library called Selendroid

29.I already have platform-specific tests for my app, what should I do to migrate to Appium?

Unfortunately there is not a magic formula to translate your tests into Selenium tests. If you developed a test framework on different layers and observed good programming principles, you should be able to act on some components in your tests in order to migrate your suites to Appium. Your current tests are going to be easy to migrate if they are already using an automation framework or something close to a command-based interaction. Truth being told, you will probably need to write your tests from the beginning, what you can do is actually reusing your existing components

30.How much time does it take to write a test in Appium?

Of course it depends by the test. If your test simply runs a scenario, it will take as many commands as the number of interactions needed to be performed (thus very few lines). If you are trying to exchange data, then your test will take more time for sure and the test will also become difficult to read.

31.Any tips or tricks to speed up my test writing activity or my migration process?

Here is **one piece of advice.** Since your tests will mostly consist in automation tasks (if this condition is not met, you might want to reconsider using Appium), make interactions reusable! Do not write the same sub-scenarios twice in your tests, make a diagram of what your scenarios are and split them in sub activities; you will get a graph where some nodes are reachable from more than one node. So make those tasks parametric and call them in your tests! This will make your test writing experience better even when you need to migrate from existing tests (hopefully you already did this activity for your existing suites).

32.What test frameworks are supported by Appium?
Appium does not support test frameworks because there is no need to support them! You can use Appium with **all test frameworks** you want. NUnit and .NET Unit Test Framework are just a few examples; you will write your tests using one of the drivers for Appium; thus your tests will interface with Appium just in terms of an external dependency. Use whatever test framework you want!

33.Can I interact with my apps using Javascript while I am testing with Appium?
Yes! Selenium has commands to execute Javascript instructions on yourapp from your tests. Basically you can send a JS script from your test to your app; when the commands runs on Appium, the server will send the script to your app wrapped into an anonymous function to be executed.

34.Is it Returning the values?
However your Javascript interaction can get more advanced as your script can return a value which will be delivered to your test when the HTTP response is sent back by Appium once your Javascript has finished running. However this scenario comes with a limitation: your Javascript can send back only primitive types (integers, strings), not complex objects. The limitation can be overtaken by passing objects as JSON strings or by modifying Appium's or Selenium's code to support specific objects.

35.How can I exchange data between my test and the app I am testing?
Appium, actually the WebDriver specification, is not made for exchanging data with your app, it is made to automate it. For this reason, you will probably be surprised in finding data exchange not so easy. Actually it is not impossible to exchange data with your app , however it will require you to build more layers of testability.

36.What data exchange is?
When I say "data exchange" I am not referring to scenarios like getting or setting the value of a textbox. I am also not referring to getting or setting the value of an element's attribute. All these things are easy to achieve in Appium as Selenium provides commands just for those. By "data exchange" I mean exchanging information hosted by complex objects stored in different parts of your webview-based app like the window object. Consider when you dispatch and capture events, your app can possibly do many things and the ways data flows can be handled are many. Some objects might also have a state and the state machine behind some scenarios in your app can be large and articulated. For all these reasons you might experience problems when testing.

37.What are Testability layers?
In order to make things better, as a developer, what you can do is adding testability layers to your app. The logic behind this approach is simply having some test-related objects in your app which are activated only when your tests run. I learned about this strategy from one of my colleagues *Lukasz* and such a technique can be really powerful. Enable your testability layers when testing in order to make data exchange easy.

38.Is it Exchanging data through Javascript?
Selenium provides commands do execute Javascript on the app, it is also possible to execute functions and have them return data (only basic types). If you exchange JSON strings it should be fine as JSON.stringify(str) will turn your JSON string into an object on the app side, while on the test side (depending on the language you are using), you can rely on hundreds of libraries to parse the string you receive.

39.What are the most difficult scenarios to test with Appium?

Appium is not suitable for all types of tests. There is a particular scenario that will make your tests more difficult to write: **data exchange**. I already said it but I will repeat the same thing because it is very important: Appium and WebDriver are designed to automate stuff… not to exchange data with them. So what if we need to exchange information with our app during tests? Should we give up on Appium and write our tests manually for each platform? I am not saying this, but there are cases where you should consider this option (not nice I know, but if the effort of writing tests for Appium is higher than the benefits, than just throw Appium away).
Appium is very nice because it will let you write tests once for all platofrms instead of writing as many tests as the numbers of platforms you need to support. So if you need to exchange data with your app while testing it and this data flow is the same for all platforms, then you should probably keep on using Appium and find a way to write a layer on top of it to handle data. Depending on your needs this might take time, but, in my experience, it is really worth it.

40. I don't want to set up a whole infrastructure for my tests and I don't want to spend money on HW. Can Appium help me?
If you think about it, what really is required from you is writing tests. Then the fact that you must deploy an Appium server somewhere is something more. If you want to skip this part, you can rely on some web services that already deployed a whole architecture of Appium servers for your tests. Most of them are online labs and they support Selenium and Appium

41.I need to debug Appium, is it difficult?
No really! Appium is a Node.js application, so it is Javascript in the essence. Depending on what you have to debug, you will probably need to go deeper in your debugging experience, however there are some key points where setting a breakpoint is always worth: the *proxy* component is worth a mention. In appium/lib/server/proxy.js you can set a breakpoint in function doProxy(req,res), that will be hit everytime commands are sent to platform-specific components to be translated into automation

42. I build my apps with Cordova, is it supported by Appium?
Cordova is a very famous system that enables you to develop webview-based apps for all platforms in short time. Appium does not explicitely say that Cordova is supported, even though they do it implicitely as some examples using apps built with Cordova are provided on Appium's website. So the answer is that **Cordova should not be a problem.** Why am I being so shy about it? Because anything can happen and it actually happened to me!

Cordova and Appium are two different projects that are growing up separately and independently, of course a mutual acknowledgement is present, but both teams do not really talk to each other when pushing features. So problems can occur (I am currently dealing with a problem concerning Cordova's new version which is causing my tests to fail).

43.Explain what is APPIUM INSPECTOR?

Similar to Selenium IDE record and Playback tool, Appium has an "Inspector" to record and playback. It records and plays native application behavior by inspecting DOM and generates the test scripts in any desired language. However, Appium Inspector does not support Windows and use UIAutomator viewer in its option.

44. What are the basic commands that I can use in the Selenium protocol?

Google's Selenium provides a collection of commands to automate your app. With those commands you can basically do the following:

- Locate web elements in your webview-based app's pages by using their ids or class names.
- Raise events on located elements like Click().
- Type inside textboxes.
- Get or set located element's attributes.
- Execute some Javascript code.
- Change the context in order to test the native part of your app, or the webview. If your app uses more webviews, you can switch the context to the webview you desire. If your webview has frames or iframes inside, you can change context to one of them.
- Detect alert boxes and dismiss or accept them. Be careful about this functionality, I experienced some problems.

45. I want to run my tests in a multi threaded environment, any problems with that?

Yes! You need some special care when using Appium in a multithreaded environment. The problem does not really rely on the fact of using threads in your tests: you can use them but you must ensure that no more than one test runs at the same time against the same Appium server. As I mentioned, Appium does not support multiple sessions, and unless you implemented an additional layer on top of it to handle this case, some tests might fail.

46.Mention what are the basic requirement for writing Appium tests?

For writing Appium tests you require,

- **Driver Client**: Appium drives mobile applications as though it were a user. Using a client library you write your Appium tests which wrap your test steps and sends to the Appium server over HTTP.
- **Appium Session**: You have to first initialize a session, as such Appium test takes place in the session. Once the Automation is done for one session, it can be ended and wait for another session
- **Desired Capabilities**: To initialize an Appium session you need to define certain parameters known as "desired capabilities" like PlatformName, PlatformVersion, Device Name and so on. It specifies the kind of automation one requires from the Appium server.
- **Driver Commands**: You can write your test steps using a large and expressive vocabulary of commands.

47.How can I run Android tests without Appium?

For older versions of Android Appium might not be supported. For instance, Appium is only supported in Android versions 4.4 or later for Mobile Web Application tests, and Android versions 2.3, 4.0 and later for Mobile Native Application and Mobile Hybrid Application tests.

For those versions in which Appium is not supported you can request an emulator driven by Webdriver + Selendroid. All you need to do is use our Platforms Configurator and select Selenium for the API instead of Appium.

In the Sauce Labs test you will notice that the top of the emulator says "AndroidDriver Webview App". In addition, you will notice that you will get a "Selenium Log" tab which has the output of the Selendroid driver.

With an emulator driven by Webdriver + Selendroid you will be able to test Mobile Web Application only. You should be able to select any Android emulator version from 4.0 to the latest version and any Android emulator skin (e.g “deviceName”:”Samsung Galaxy Tab 3 Emulator”).

48.How can I run iOS tests without Appium?
For older versions of iOS Appium might not be supported. For instance, Appium is supported in iOS versions 6.1 and later. For earlier versions of iOS the tool or driver used to drive your mobile applications automated test is called iWebdriver.
To obtain a simulator driven by iWebdriver use our Platforms Configurator and select Selenium for the API instead of Appium. With an emulator driven by iWebdriver you will be able to test Mobile Web Application only. In addition, in the Sauce Labs test you will notice a “Selenium Log” tab which has the output of iWebdriver.

49.What mobile web browsers can I automate in the Android emulator?
Currently the only browser that can be automated in our Android emulators is the stock browser (i.e Browser). The Android stock browser is an Android flavor of ‘chromium’ which presumably implies that its behavior is closer to that of Google Chrome.

50.Explain what is Appium?
Appium is a freely distributed open source mobile application UI Testing framework.

51.List out the Appium abilities?
Appium abilities are

- Test Web
- Provides cross-platform for Native and Hybrid mobile automation
- Support JSON wire protocol
- It does not require recompilation of App
- Support automation test on physical device as well as similar or emulator both
- It has no dependency on mobile device

52. List out the pre-requisite to use APPIUM?
Pre-requisite to use APPIUM is

- ANDROID SDK
- JDK
- TestNG
- Eclipse
- Selenium Server JAR
- Webdriver Language Binding Library
- APPIUM for Windows
- APK App Info On Google Play
- js

53.List out the limitations of using Appium?

- Appium does not support testing of Android Version lower than 4.2
- Limited support for hybrid app testing. E.g., not possible to test the switching action of application from the web app to native and vice-versa

- No support to run Appium Inspector on Microsoft Windows

54.Explain how to find DOM element or xPath in a mobile application?

To find the DOM element use “UIAutomateviewer” to find DOM element for Android application.

55.Explain the design concept of Appium?

- Appium is an “HTTP Server” written using Node.js platform and drives iOS and Android session using Webdriver JSON wire protocol. Hence, before initializing the Appium Server, Node.js must be pre-installed on the system
- When Appium is downloaded and installed, then a server is setup on our machine that exposes a REST API
- It receives connection and command request from the client and execute that command on mobile devices (Android / iOS)
- It responds back with HTTP responses. Again, to execute this request, it uses the mobile test automation frameworks to drive the user interface of the apps. Framework like
  - Apple Instruments for iOS (Instruments are available only in Xcode 3.0 or later with OS X v10.5 and later)
  - Google UIAutomator for Android API level 16 or higher
  - Selendroid for Android API level 15 or less

55.What language does Appium support?

Appium support any language that support HTTP request like Java, JavaScript with Node.js, Python, Ruby, PHP, Perl, etc.

56.Explain the pros and cons of Appium?

Pros:

- For programmer irrespective of the platform, he is automating (Android or iOS) all the complexities will remain under single Appium server
- It opens the door to cross-platform mobile testing which means the same test would work on multiple platforms
- Appium does not require extra components in your App to make it automation friendly
- It can automate Hybrid, Web and Native mobile applications

Cons:

- Running scripts on multiple iOS simulators at the same time is possible with Appium
- It uses UIAutomator for Android Automation which supports only Android SDK platform, API 16 or higher and to support the older API’s they have used another open source library called Selendroid

57.Explain what is APPIUM INSPECTOR?

Similar to Selenium IDE record and Playback tool, Appium has an “Inspector” to record and playback. It records and plays native application behavior by inspecting DOM and generates the test scripts in any desired language. However, Appium Inspector does not support Windows and use UIAutomator viewer in its option.

58.Mention what are the basic requirement for writing Appium tests?

For writing Appium tests you require,

- **Driver Client**: Appium drives mobile applications as though it were a user. Using a client library you write your Appium tests which wrap your test steps and sends to the Appium server over HTTP.
- **Appium Session**: You have to first initialize a session, as such Appium test takes place in the session. Once the Automation is done for one session, it can be ended and wait for another session
- **Desired Capabilities**: To initialize an Appium session you need to define certain parameters known as “desired capabilities” like PlatformName, PlatformVersion, Device Name and so on. It specifies the kind of automation one requires from the Appium server.
- **Driver Commands**: You can write your test steps using a large and expressive vocabulary of commands.

59.Mention what are the possible errors one might encounter using Appium?
The possible errors one might face in Appium includes

- Error 1: The following desired capabilities are needed but not provided: Device Name, platformName
- Error 2: Could not find adb. Please set the ANDROID_HOME environment variable with the Android SDK root directory path
- Error 3: openqa.selenium.SessionNotCreatedException: A new session could not be created
- Error 4: How to find DOM element or XPath in a mobile application?

60.Do you need a server machine to run tests on Appium?
No, you don't need server machine to run tests on Appium. Appium facilitates a 2-tier architecture where a test machine connects to a test server running Appium and automating the whole thing. You can have Appium running on the same machine where your test runs.

61.Is it possible to interact with my apps using Javascript while I am testing with Appium?
Yes, it is possible to interact with App while using Javascript. When the commands run on Appium, the server will send the script to your app wrapped into an anonymous function to be executed.

62.Mention what are the most difficult scenarios to test with Appium?
The most difficult scenario to test with Appium is data exchange.

63.While using Appium can I run my tests in a multithreaded environment?
Yes, you can run the test in a multithreaded environment but you have to ensure that no more than one test runs at the same time against the same Appium server.

64.In Android, do you need an app's .apk to automate using Appium or you also need app in my workspace?
In Android, you only need .apk file to automate using Appium.

65.Explain what is Appium package master? How to create package?
Appium package master is a set of tools manage and create appium packages. For example to create package you can use the code
using es7/babe1
Gulp create-package –n <package-name>
#regular es5
Gulp create-package —nobabe1 –n <package-name>
The package will be generated in the out/<package-name>

66.Explain how test frameworks are supported by Appium?
Appium does not support test framework as such there is no need to support them. Appium can be used with any frameworks you want