SQA DOCUMENT

Version 1.0

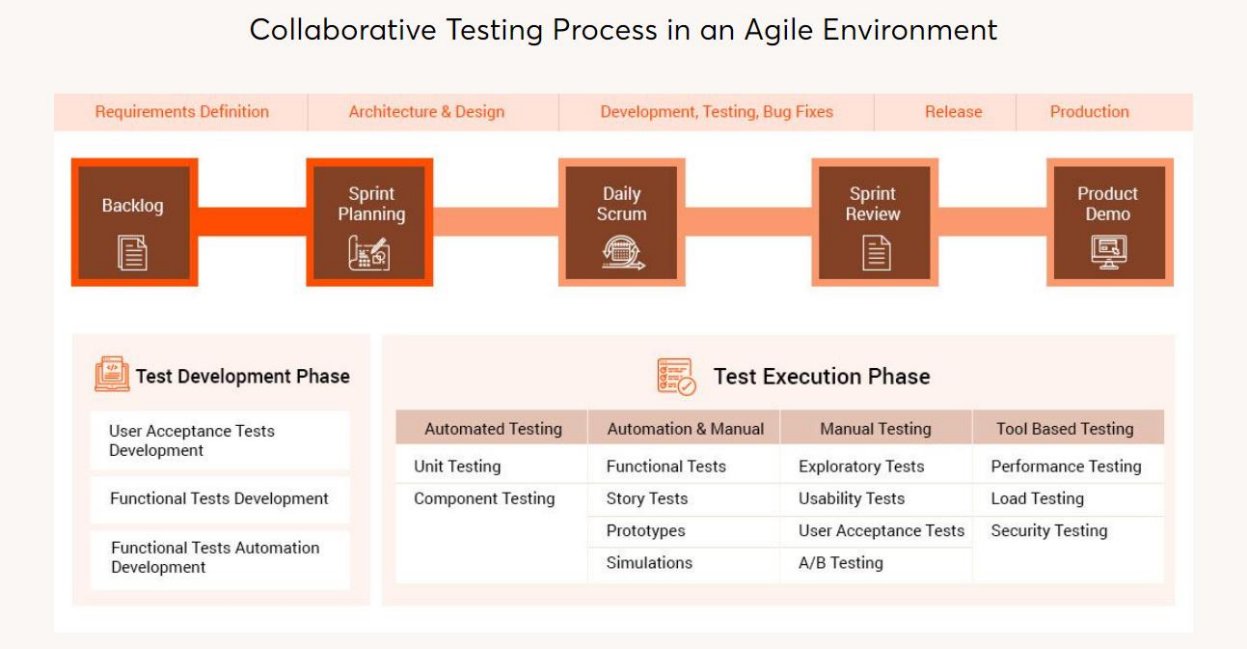
IDEOFUZION PVT LTD

Version 1.0

Contents

1. Software Quality Assurance

It is a process that assures that all software engineering processes, methods, activities, and work items are monitored and comply with the defined standards. These defined standards could be one or a combination of any like ISO 9000, CMMI (Capability Maturity Model Integration) model, ISO15504, etc.
SQA incorporates all software development processes starting from defining requirements to coding until release. Its prime goal is to ensure quality.



2. What is manual testing

Manual testing is the process in which QA analysts execute tests one-by-one in an individual manner. The purpose of manual testing is to catch bugs and feature issues before a software application goes live.

3. Automation testing

Automation testing is the process in which testers utilize tools and scripts to automate testing efforts.

4. Quality Control

Quality control is a subset of QA. In QC, teams ensure that the developed product meets the organization's quality standards. Defects in a software product, such as UI glitches, design imperfections, accessibility issues or security gaps, can cause irreparable damage to a brand's reputation. Through a systematic QC process, the organization can correct products to ensure that they meet business requirements and customer expectations.

5. Software Testing

According to ANSI/IEEE 1059 standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

6. Importance of Software testing

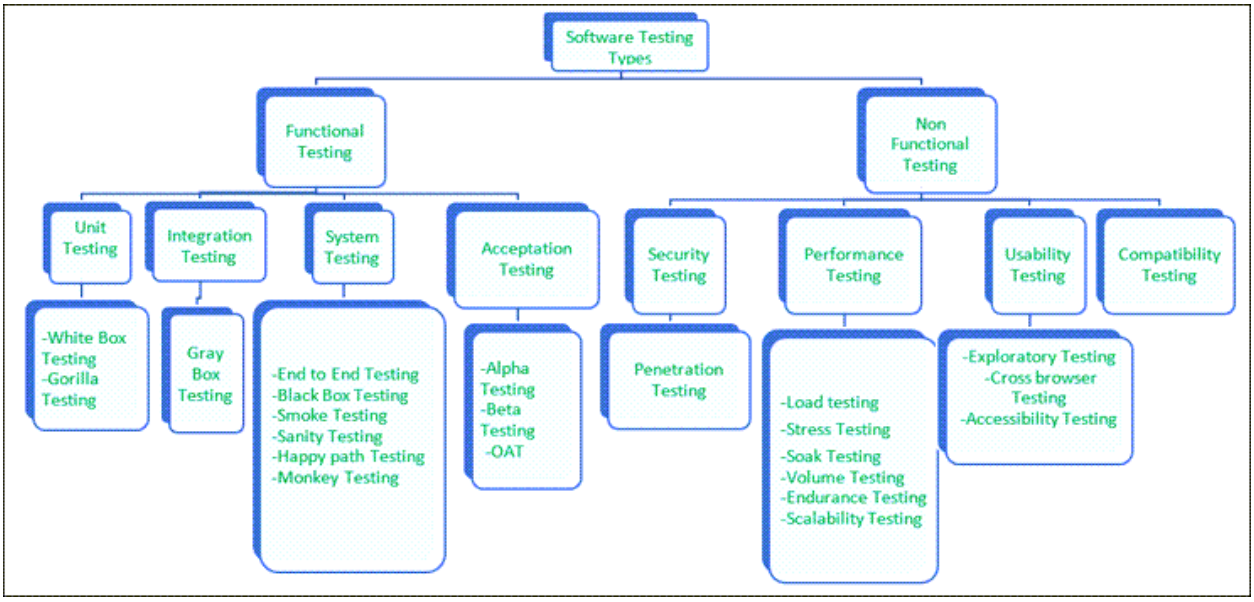
Human errors can cause a defect or failure at any stage of the software development life cycle. The results are classified as trivial or catastrophic, depending on the consequences of the error. The requirement of rigorous testing and their associated documentation during the software development life cycle arises because of the below reasons:

- To identify defects
- To reduce flaws in the component or system
- Increase the overall quality of the system
- The testing is important since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software.
- It makes the software more reliable and easy to use.
- Thoroughly tested software ensures reliable and high-performance software operation.

Testing helps developers and testers to compare actual and expected results in order to improve quality. If the software production happens without testing it, it could be useless or sometimes dangerous for customers. So, a tester should wear a unique hat that protects the reliability of the software and makes it safe to use in real-life scenarios.

A mistake in coding is called **Error**, error found by tester is called **Defect**, defect accepted by development team then it is called **Bug**, build does not meet the requirements then it is **Failure**.

7. Types of Software testing



The types of software testing.

1) White box

2) Black box

3) Grey Box

4) Alpha Testing

5) Beta Testing

7.1What is White Box Testing?

White Box Testing is also called as Glass Box, Clear Box, and Structural Testing. It is based on applications internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually was done at the unit level.

7.2What is Black Box Testing?

Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure. This can be applied to every level of software testing such as Unit, Integration, System and Acceptance Testing.

7.3What is Grey Box Testing?

Grey box is the combination of both White Box and Black Box Testing. It is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known

7.4What is Alpha Testing?

Alpha testing is done by the in-house developers (who developed the software) and testers. Sometimes alpha testing is done by the client or outsourcing team with the presence of developers or testers.

It has two phases:

- In the first phase of alpha testing, the software is tested by in-house developers. They use debugger software. The goal is to catch bugs quickly.
- In the second phase of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.

7.5What is Beta Testing?

Beta Testing is performed by “real users” of the software application in “real environment” and it can be considered as a form of external User Acceptance Testing. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.

7.6What is Functional Testing?

Functional testing is a type of software testing that validates the software system against the functional requirements/specifications. Verify that each function of the software application behaves as specified in the requirement document. Testing all the functionalities by providing appropriate input to verify whether the actual output is matching the expected output or not. It falls within the scope of black box testing and the testers need not concern about the source code of the application.

7.7What is Unit/Module Testing?

Unit Testing is also called as Module Testing or Component Testing. It is done to check whether the individual unit or module of the source code is working properly. It is done by the developers in the developer's environment.

7.8What is Integration Testing?

Integration Testing is the process of testing the interface between the two software units. Integration testing is done by three ways.

- Big Bang Approach
- Top-Down Approach
- Bottom-Up Approach.

7.9What is System Testing?

Testing the fully integrated application to evaluate the system's compliance with its specified requirements is called System Testing End to End testing. Verifying the completed system to ensure that the application works as intended or not. System testing is carried out by specialist testers or independent testers. System testing should investigate both functional and non-functional requirements of the testing.

7.10 What is Smoke Testing?

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as “Day 0” check. It is done at the “build level”. It helps not to waste the testing time to simply testing the whole application when the key features don’t work or the key bugs have not been fixed yet.

7.11 What is Sanity Testing?

Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper. It is also called as a subset of Regression testing. It is done at the “release level”. We perform sanity testing when we don’t have enough time for regression testing.

7.12 What is Regression Testing?

Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

Usually, we do regression testing in the following cases:

1. New functionalities are added to the application
2. Change Requirement (In organizations, we call it as CR)
3. Defect Fixing
4. Performance Issue Fix
5. Environment change (E.g., Updating the DB from MySQL to Oracle)

7.13 What is Retesting Testing?

Retesting is done to make sure that the tests cases which failed in last execution are passed after the defects are fixed. Retesting is carried out based on the defect fixes. In Retesting, the cases which are failed earlier can be included to check if the functionality failure in an earlier build.

7.14 What is Exploratory Testing?

Usually, this process will be carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements.

7.15 What is Monkey Testing?

Perform abnormal action on the application deliberately in order to verify the stability of the application.

7.16 What is Big Bang Approach?

Combining all the modules once and verifying the functionality after completion of Individual module testing.
Top down and bottom up are carried out by using dummy modules known as Stubs and Drivers. These Stubs and Drivers are used to stand-in for missing components to simulate data communication between modules.

7.17 What is Top-Down Approach?

Testing takes place from top to bottom. High-level modules are tested first and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended. Stubs are used as a temporary module if a module is not ready for integration testing.

7.18 What is Bottom-Up Approach?

It is a reciprocate of the Top-Down Approach. Testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules and finally integrating the high-level modules to a low level to ensure the system is working as intended. Drivers are used as a temporary module for integration testing.

7.19 What is User Acceptance Testing / UAT?

It is also known as pre-production testing. This is done by the end users along with the testers to validate the functionality of the application. After successful acceptance testing. Formal testing conducted to determine whether an application is developed as per the requirement. It allows the customer to accept or reject the application. Types of acceptance testing are Alpha, Beta & Gamma.

7.20 What is Positive and Negative Testing?

Positive Testing: It is to determine what system supposed to do. It helps to check whether the application is justifying the requirements or not.
Negative Testing: It is to determine what system not supposed to do. It helps to find the defects from the software.

7.21 What is Non-Functional Testing?

In simple words, how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility etc., Main focus is to improve the user experience on how fast the system responds to a request.

7.22 What is Performance Testing?

This type of testing determines or validates the speed, scalability, and/or stability characteristics of the system or application under test. Performance is concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the project or product.

7.23 What is Load Testing?

It is to verify that the system/application can handle the expected number of transactions and to verify the system/application behavior under both normal and peak load conditions.

7.24 What is Volume Testing?

It is to verify that the system/application can handle a large amount of data.

7.25 What is Stress Testing?

It is to verify the behavior of the system once the load increases more than its design expectations.

7.26 What is Scalability Testing?

Scalability testing is a type of non-functional testing. It is to determine how the application under test scales with increasing workload.

7.27 What is Concurrency Testing?

Concurrency testing means accessing the application at the same time by multiple users to ensure the stability of the system. This is mainly used to identify deadlock issues.

7.28 What is GUI Testing?

Graphical User Interface Testing is to test the interface between the application and the end user.

7.29 What is Recovery Testing?

Recovery testing is performed in order to determine how quickly the system can recover after the system crash or hardware failure. It comes under the type of non-functional testing.

7.30 What is Installation Testing?

It is to check whether the application is successfully installed and it is working as expected after installation.

7.31 What is Compatibility Testing?

It is to deploy and check whether the application is working as expected in a different combination of environmental components.

7.32 What is Usability Testing?

To verify whether the application is user-friendly or not and was comfortably used by an end user or not. The main focus in this testing is to check whether the end user can understand and operate the application easily or not. An application should be self-exploratory and must not require training to operate it.

7.33 What is User Testing? How Does User Testing Work?

User Testing is the only effective way in existence to find out your app or website will work or not. When we put the business into perspective. How does user testing work?

So you have a business idea! But will it work? Any idea if it is not desirable by its prospective users is worthless. The same goes for the software world. Any software or application idea if it is not desirable by its users is of no use.t is hence very important, to get your idea tested before converting it into reality. And here is where you will require.

When is user testing done?

This testing is done after you get an idea and before you commence working on the software/ application. The input to your testing is your idea and the output or result is its user's feedback on your idea.

Who does user testing?

User testing tests the user's perspective on the application idea. The main reviewers in such kind of testing are the prospective users.

The testers and the idea owners create a survey and the users take that survey. Hence the main participants of this testing are the users and the idea owners/testers.

What is the purpose of user testing?

The purpose of this testing is to evaluate whether the idea you are planning to work upon is

desirable by its prospective users or not. It is a good idea to convert your idea into an app-only if people are ready to use it if and only if they think they need such a solution to make their lives easier.

What factors are considered while performing user testing?

Customer needs, their Perception, and their Demand are the major factors that are considered while performing this testing.

What are the types of User Testing

Usability testing

A kind of user testing where real users embark on a journey to figure out the ease of access of the app/software. The main goal behind user testing is to measure the human interaction on a particular software.

User scenarios can be verified using this technique and the success percentage can be measured prior to release.

Tree Testing

A tree-like sitemap will be given to users and they will be asked to navigate the software without any distraction.

Remote Usability Testing

There will be a centralized platform that will record user interaction by users located in various parts of the world. Localization testing is the perk of this type of testing.

A/B Testing

It's one of the most effective ways of user testing. If there is a huge update for the software, for instance, a website is planning to launch a new version of its web app. Through A/B testing the company can release two versions of the web app and seek feedback from users and compare

it to get a better perspective. The purpose of A/B testing is to study user behavior and compare it with the feasibility of update or the production of software.

How is user testing conducted?

User testing can be conducted through in-person discussion, coffee shop discussion, Google hangout, Type form, Virtually at Skype, Google forms, Survey Monkey, social media polls, social media discussion, online discussion forums, etc.

What to ask during user testing?

The main concern in this testing is what questions to ask your users while conducting the survey or this testing. Some of the basic questions you can ask are:

How you solve that problem currently?

Have you ever thought of doing it in a better way?

Will you use the proposed solution to accomplish this task?

Will you pay money if asked to use this solution?

Do you think this solution will ease out your work?

How to do user testing done?

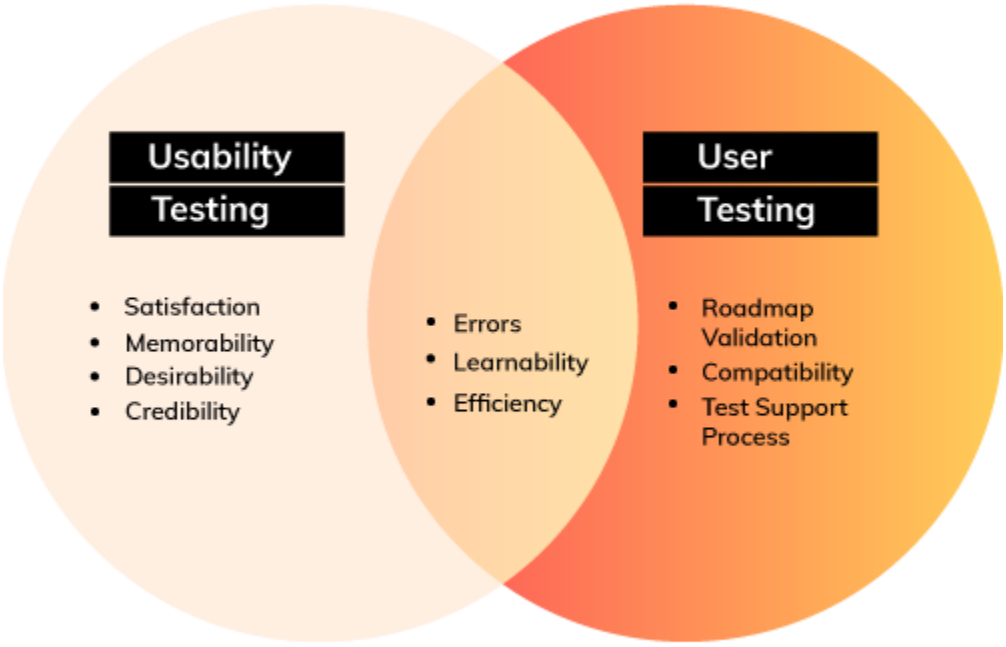
User testing serves as the analysis of the achieve ability of any idea. A questionnaire will be prepared including all the questions that will extensively define the user's perspective to that particular idea. This questionnaire is then presented to the prospective users and their answers will be collected. Based on their answers a decision will be made whether to give the idea a go-ahead or dissolve the plan.

Does user testing really pay?

There is an on-demand user testing platform called usertesting.com which helps products acquire real-user feedback.

Basically how it works is that you have to visit the website and register yourself as a user tester you get to test. As per the information provided on the website you ger to earn $10 for every

20-minute. However, the dubious part is whether you get enough chances to test the website or will there be any geographical disparity to not?



User Testing	Usability Testing
Do users need your app?	Can users use your app?
Done after you have an idea	Done after you have you're your design ready and continues till the end before the app is live

A discussion or a survey is best suited for this testing	You have to have a design ready to get usability testing

7.34 What is Shift left Testing? And why is it so relevant?

- **What is Shift Left Testing?**

To reduce the number of bugs during and after software development and improve the quality of the product, Shift Left Testing or Early Testing is implemented.

It is a method to push testing towards the early stage of software development like requirements defects, complicated designing, and so on.

By doing so, you uncover and solve the issues in an early testing phase before they become major.

Don't confuse the term Shift Left here with literally shifting to left.

It's about shifting to automated testing from manual testing to detecting bugs early and at all stages and often testing, diminishing the extra manual work.

Why do you need shift left testing?

In a traditional Software Development Life Cycle, the testing phase numbers last in the process, just before the launch of the product.

This results in encountering serious issues may be in your design, longer buffer time, clickable buttons not working as required, and more like this.

This requires extensive efforts under experience manpower supervision that would cost you a fortune.

To sum up, start testing early throughout the development process and test often to reduce risk, cost, and time.

Using the strategy, you keep improving your software's quality before it goes under the last testing phases.

Advantages and benefits of shift left testing

Potential smaller bugs with major risks that are detected earlier cost much less to fix and remove.

Automation is essential to reduce human errors and lets the testers focus on multiple inspiring tasks without affecting the quality.

l Precaution is better than a cure, and not just medically.

By reducing the unexpected threats from the potential malfunction with the Shift Left Testing strategy, you increase development process cycle efficiency.

How to Implement Shift Left Testing? / How do you shift a test to the left?

Implement a successful shift left testing strategy by following the below given important steps:

Plan & Analysis- Include Testers from the beginning to carefully **analyze the requirements, design plan**, and calculate the estimated budget.

Include Developers & Testers– Build a professional team of QAs and developers throughout the designing and developing phase of the project and increase the actual testing phase efficiency.

Unified Testing Strategy– This allows you to identify and analyze the dependencies on the environment, code reviews, automation, and test data; and helps define clear responsibilities to each team member.

Risk-Based Analysis– It is implemented to determine the impact and chances of failure for each test scenario. The testers must prioritize the test cases and discuss with the developers the likely-to-be failed aspects and their impacts on the overall development cycle.

Introduce Test Automation– As Shift Left Testing involves testing often and throughout the process, embrace the test automation tools to speed up the development lifecycle, increase efficiency, fewer bugs, and generate quick feedback. It ensures better code coverage and maintains the product's quality.

Different Types of the Shift Left Testing

There are four different types of Shift Left Testing that provide different results in different scenarios.

1) Traditional Shift Left Testing

To understand the traditional shift testing, we must first understand the traditional SDLC V-Model, which emphasizes, acceptance and system-level testing, on testing from lower down on the right side of the V-model.

Whereas Traditional Shift-Left Testing emphasizes unit testing and integration testing, this is done using API testing and automated testing tools.

2) Incremental Shift Left Testing

This is a suitable approach for large software companies and projects with complex designs and development.

The tasks are broken into smaller segments that are built on each other with an increment.

After each increment is delivered to a customer and operation team, then the developing and testing incrementally shifts to the left, which helps testers to run test case scenarios on each individual bit making it easier to identify and fix the potential risks.

3) DevOps/Agile Shift Left Testing

This kind of software testing approach is practiced to run continuous tests on the number of sprints via an evolutionary development life cycle, like DNA.

It is mainly done for development testing only once when the system is operational.

4) Model-based Shift Left Testing

The shift-left testing starts at the early stage of the development cycle, so the bugs and threats are fixed long before the software development life cycle would start.

It leads to missing some critical issues regarding the requirement gathering phase, which gets introduced after development cycle completion and can contain serious threats and errors.

Model-based testing shifts to the left by testing requirements, building, and designing models are fulfilling the needs of the desired product.

Factors to Keep In Mind for a Successful Early Testing

There are lots of factors because of which your shift-left testing strategies can fail like testers or developers are not involved from the early stage, not able to analyze the right testing environment, and so on.

Not testing enough!

To reduce the impact of bugs and the likeliness of the failures of the testing, you must ensure to run testing continuously.

This approach helps testers to identify minor and major issues earlier.

Practicing the same development process

The SDLC process needs to adapt and accommodate the earlier testing environment.

For instance, if you want to perform user interface testing at an early stage, you need to change and develop the system requirements that support the earlier testing environment.

Still Stuck on Manual testing!

The larger the system, the more is it at risk and more testing.

Testers cannot keep up with the instant update, release, customization, and integration manually and maintain the product's quality at the same time.

Hence, shift to automated testing and tackle every challenge with more testing capacity and accuracy. Some of the tools you can use are Selenium, Leap work, etc.

There is something called shift right as well! What's the difference between shift left and shift right?

Shift right is a bit of a shocking concept actually. To speed up the development process the entire testing process will be sifted to post-development.

The reason behind such a drastic shift is to gain user insight regarding the issues and correct it so that higher UX gain can be achieved

A/B testing can be performed easily

The stability of the back-end architecture can be examined in detail

Issues with the app can be traced out in the early stages of deployment real-world performance insights can be gained.

7.35 What is Cross Browser Testing?

Cross browser testing validates the proper working of your application over different browsers. It verifies that your website works steadily and as per requirements. It can be used both for web and mobile applications.

What kinds of applications should be tested for cross browser testing?

Applications that are Customer facing are more desirable to be executed for cross-browser testing. But finally, the question arises that most of the applications are meant for end-users only.

Why is cross browser testing needed?

Cross browser testing as discussed earlier is done to test the application's compatibility with multiple browsers. Some of the other common reasons for executing cross-browser testing are:

- To find defects in the applications and be able to rectify them.
- To enhance user-experience across all browsers.
- To increase the efficiency of the app
- To find any probable drawbacks
- To test if the app looks the same across various browsers.
- To validate if all the functionalities of the app work fine on various browsers.
- To verify if it fulfills the client's requirements across all the browsers.

Who Performs This Testing?

The answer to this question might end in a single word –"Testers". But it is not exactly the case. The testers play an important role in cross browser testing, but business analysts, clients; stakeholders have an equally important role in deciding upon the number and names of the browsers, the testing has to be conducted on.

Analyzing the statistics of the users using different browsers, they pick up a few browsers that are more frequently used by the application users. It is next to impossible for the testers to perform cross browser testing on 1000's of browsers available in the market.

The analyst team and the clients make a list of the most frequently used browsers by the application end-users and testers then perform testing over it.

- How to Perform Cross Browser Testing?
- Talking about the most important part – performing the cross-browser testing, the very first thing that has to be decided is whether to perform manual testing or automation testing.

  Manual testing can be performed for cross-browser testing, but considering the multiple machines, multiple OS, Multiple browsers, manual testing can lead to many problems, and various challenges. Hence automation testing is a preferred method of testing for cross browser testing.

Manual Method

When executing cross browser testing manually, it is very difficult to test the application on the number of diverse browsers. In manual testing, the app can be tested over a few limited browsers only and hence delimiting the efficiency of the app.

Also, manual testing for cross-browser testing is both costly and time-consuming.

Automated Method

Cross-browser testing requires running the same test suit over various browsers. This is a recurrent task, and can result in errors if done manually and is also very time-consuming and costly when done manually.

Hence, cross browser testing is generally executed using automation testing. Cross browser testing is more cost and time effective when done using automated testing tools.

A lot many tools are available that assists in cross-browser testing.

Recommended Tools

1. LambdaTest

LambdaTest is used to test web app on over 2000+ combinations of different browser and operating system. It is a cloud-based cross-browser testing platform.

2. Cross Browser Testing

Cross browser testing has a vast competency to execute manual, visual, and Selenium tests in the cloud on over 2050+ real desktop and mobile browsers.

3. Experitest

Experitest allows executing a parallel test on different browsers and mobile devices.

4. Selenium

Selenium with an easy transaction of browser can help test the web applications easily in parallel. It is a well-known automated testing tool.

5. BrowserStack

This cloud-based web and mobile testing allow on-demand browsers, operating systems, and real mobile devices testing.

6. Browserling

Browserling is a live interactive for testing effortlessly. Browserling offers speedy access to all the common browsers and popular operating systems.

7.36 What is Security Testing?

Security Testing is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders. The purpose of Security Tests is to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue, repute at the hands of the employees or outsiders of the Organization.

Why Security Testing is Important?

The main goal of **Security Testing** is to identify the threats in the system and measure its potential vulnerabilities, so the threats can be encountered and the system does not stop functioning or cannot be exploited. It also helps in detecting all possible security risks in the system and helps developers to fix the problems through coding.

Types of Security testing:

- **Vulnerability Scanning**: This is done through automated software to scan a system against known vulnerability signatures.
- **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.
- **Penetration testing**: This kind of testing simulates an attack from a malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
- **Risk Assessment:** This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.
- **Security Auditing:** This is an internal inspection of Applications and Operating systems for security flaws. An audit can also be done via line by line inspection of code
- **Ethical hacking:** It's hacking an Organization Software systems. Unlike malicious hackers, who steal for their own gains, the intent is to expose security flaws in the system.
- **Posture Assessment:** This combines Security scanning, Ethical Hacking and Risk Assessments to show an overall security posture of an organization.

7.37 Security Testing Tool

1. Acunetix

Intuitive and easy to use, Acunetix by Invicti helps small to medium-sized organizations ensure their web applications are secure from costly data breaches. It does so by detecting a wide range of web security issues and helping security and development professionals act fast to resolve them.

Features:

Advanced scanning for 7,000+ web vulnerabilities, including OWASP Top 10 such as SQLi and XSS

Automated web asset discovery for identifying abandoned or forgotten websites

Advanced crawler for the most complex web applications, incl. multi-form and password-protected areas.

Combined interactive and dynamic application security testing to discover vulnerabilities other tools miss
Proof of exploit provided for many types of vulnerabilities
DevOps automation through integrations with popular issue tracking and CI/CD tools
Compliance reporting for regulatory standards, such as PCI DSS, NIST, HIPAA, ISO 27001, and more.

2. Intruder

Intruder is a powerful, automated penetration testing tool that discovers security weaknesses across your IT environment. Offering industry-leading security checks, continuous monitoring and an easy-to-use platform, Intruder keeps businesses of all sizes safe from hackers.

Features:
Best-in-class threat coverage with over **10,000 security checks**
Checks for configuration weaknesses, missing patches, application weaknesses (such as SQL injection & cross-site scripting) and more.
Automatic analysis and prioritization of scan results.
Intuitive interface, quick to set-up and run your first scans.
Proactive security monitoring for the latest vulnerabilities.
AWS, Azure and Google Cloud connectors.
API integration with your CI/CD pipeline.

3. Owasp

The Open Web Application Security Project (OWASP) is a worldwide non-profit organization focused on improving the security of software. The project has multiple tools to pen test various software environments and protocols. Flagship tools of the project include
Zed Attack Proxy (ZAP – an integrated penetration testing tool)
OWASP Dependency Check (it scans for project dependencies and checks against know vulnerabilities)
OWASP Web Testing Environment Project (collection of security tools and documentation)

4. WireShark

Wireshark is a network analysis tool previously known as Ethereal. It captures packet in real time and display them in human readable format. Basically, it is a network packet analyzer- which provides the minute details about your network protocols, decryption, packet information, etc. It is an open source and can be used on Linux, Windows, OS X, Solaris, NetBSD,

FreeBSD and many other systems. The information that is retrieved via this tool can be viewed through a GUI or the TTY mode TShark Utility.

5. W3af

w3af is a web application attack and audit framework. It has three types of plugins; discovery, audit and attack that communicate with each other for any vulnerabilities in site, for example a discovery plugin in w3af looks for different url's to test for vulnerabilities and forward it to the audit plugin which then uses these URL's to search for vulnerabilities.

7.38 What is Adhoc Testing?

Ad-hoc testing is quite opposite to the formal testing. It is an informal testing type. In Adhoc testing, testers randomly test the application without following any documents and test design techniques. This testing is primarily performed if the knowledge of testers in the application under test is very high. Testers randomly test the application without any test cases or any business requirement document.

7.39 What is Bucket Testing?

Bucket or Split testing is a method to compare two versions of an application against each other to determine which one performs better.

7.40 What is Defect Cascading in Software Testing?

Defect cascading in Software testing means triggering of other defects in an application.
When a defect is not identified or goes unnoticed while testing, it invokes other defects.
It leads to multiple defects in the later stages and results in an increase in a number of defects in the application.
For example, if there is a defect in an accounting system related to negative taxation then the negative taxation defect affects the ledger which in turn affects other reports such as Balance Sheet, Profit & Loss etc.,

7.41 What is Walk Through?

A walkthrough is an informal meeting conducts to learn, gain understanding, and find defects. The author leads the meeting and clarifies the queries raised by the peers in the meeting.

7.42 What is Inspection?

Inspection is a formal meeting lead by a trained moderator, certainly not by the author.
The document under inspection is prepared and checked thoroughly by the reviewers before the meeting. In the inspection meeting, the defects found are logged and shared with the author for appropriate actions. Post inspection, a formal follow-up process is used to ensure a timely and corrective action.

7.43 What is performance testing?

Performance Testing is a type of software testing that ensures software applications to perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity and stability under a particular workload. Performance Testing is the process of analyzing the quality and capability of a product. It is a testing method performed to determine the system performance in terms of speed, reliability and stability under varying workload. Performance testing is also known as ***Perf Testing***.

Performance Testing Attributes:

- **Speed:**

It determines whether the software product responds rapidly.

- **Scalability:**

It determines amount of load the software product can handle at a time.

- **Stability:**

It determines whether the software product is stable in case of varying workloads.

- **Reliability:**

It determines whether the software product is secure or not.

Types of Performance Testing:

- **Load testing:**

It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in market.

- **Stress testing:**

It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.

- **Endurance testing:**

It is performed to ensure the software can handle the expected load over a long period of time.

- **Spike testing:**

It tests the product's reaction to sudden large spikes in the load generated by users.

- **Volume testing:**

In volume testing large number of data is saved in a database and the overall software system's behavior is observed. The objective is to check product's performance under varying database volumes.

- **Scalability testing:**

In scalability testing, software application's effectiveness is determined in scaling up to support an increase in user load. It helps in planning capacity addition to your software system.

8. What is a Defect?

The variation between the actual results and expected results is known as a defect. If a developer finds an issue and corrects it by himself in the development phase, then it's called a defect.

9. What is a Bug?

If testers find any mismatch in the application/system in testing phase, then they call it as Bug.

10. What is an Error?

We can't compile or run a program due to a coding mistake in a program. If a developer Unable to successfully compile or run a program, then they call it as an error.

11. What is a Failure?

Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue, then that particular issue is called as a failure.

12. What is Bug Severity?

Bug/Defect severity can be defined as the impact of the bug on customer's business. It can be Critical, Major or Minor. In simple words, how much effect will be there on the system because of a particular defect.

13. What is Bug Priority?

Defect priority can be defined as how soon the defect should be fixed. It gives the order in which a defect should be resolved. Developers decide which defect they should take up next based on the priority. It can be High, Medium or Low. Most of the times the priority status is set based on the customer requirement.

Examples of Bug Severity and Bug Priority

High Priority & High Severity: Submit button is not working on a login page and customers are unable to login to the application

Low Priority & High Severity: key feature failed but there's no impact on customer business, e.g., calculation fault in yearly report which end user won't use on daily basis.

High Priority & Low Severity: Spelling mistake of a company name on the homepage

Low Priority & Low Severity: FAQ page takes a long time to load

14. What is a Critical Bug?

A critical bug is a show stopper which means a large piece of functionality or major system component is completely broken and there is no workaround to move further.

For example, Due to a bug in one module, we cannot test the other modules because that blocker bug has blocked other modules. Bugs which affect the customers' business are considered as critical.

Example:

1. "Sign In" button is not working on Gmail App and Gmail users are blocked to login to their accounts.

2. An error message pops up when a customer clicks on transfer money button in a Banking website.

15. What are entry criteria?

Entry criteria is a set of conditions that permits a task to perform, or in absence of any of these conditions, the task cannot be performed.

- The requirement document should be available.
- Complete understanding of the application flow is required.
- The Test Plan Document should be ready.

16. What is exit criteria?

Exit criteria is a set of expectations; this should be met before concluding the STLC phase.

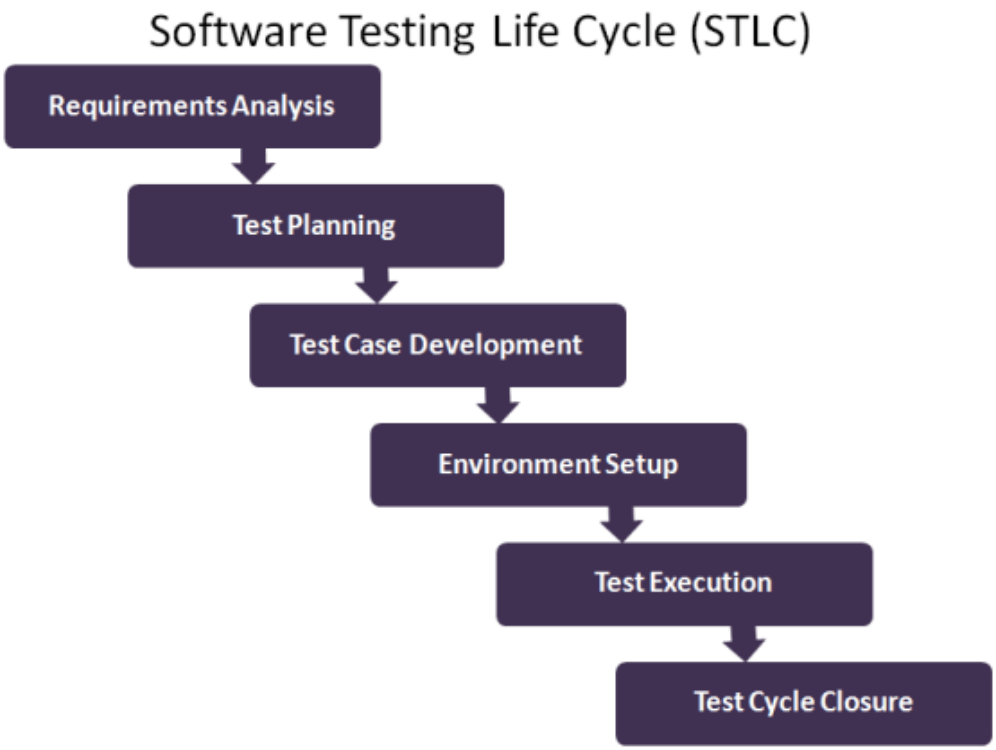
- Test Cases should be written and reviewed.
- Test Data should be identified and ready.
- Test automation script should be ready if applicable.

17. STLC Life cycle

STLC is a testing process which is executed in a sequence, in order to meet the quality goals

It is not a single activity but it consists of many different activities which are executed to achieve a good quality product. STLC is followed by the testing team.

There are 6 phases in the Software Testing Life Cycle or STLC life cycle. The entry criteria must be fulfilled before each phase can start. The exit criteria should be fulfilled before exiting a phase. Every phase has one or more deliverables that are produced at the end of the phase The phases are executed in a sequence.



17.1 Requirement Analysis

This is the very first phase of Software testing Life cycle (STLC). In this phase testing team goes through the Requirement document with both Functional and non-functional details in order to identify the testable requirements.

Activities to be done in Requirement analysis phase are given below:

- Analyzing the System Requirement specifications from the testing point of view
- Preparation of RTM that is Requirement Traceability Matrix
- Identifying the testing techniques and testing types
- Prioritizing the feature which need focused testing

- Analyzing the Automation feasibility
- Identifying the details about the testing environment where actual testing will be done

Deliverables (Outcome) of Requirement analysis phase are:

17.1.1 Requirement Traceability Matrix (RTM)

https://docs.google.com/spreadsheets/d/1twNAz01PPBQRMNfZ7NlaqJ8RKUvLBf3d/edit?usp=sharing&ouid=113144539660973376722&rtpof=true&sd=true

Examples Of RTM

#1) Business Requirement

BR1: Writing emails option should be available.

Test Scenario (technical specification) for BR1

TS1: Compose mail option is provided.

Test Cases:

Test Case 1 (TS1.TC1): Compose mail option is enabled and works successfully.

Test Case 2 (TS1.TC2): Compose mail option is disabled.

#2) Defects

After executing the test cases if any defects are found that too can be listed and mapped with the business requirements, test scenarios and test cases.

For Example, If TS1.TC1 fails i.e. Compose mail option though enabled does not work properly then a defect can be logged. Suppose the defect ID auto-generated or manually assigned number is D01, then this can be mapped with BR1, TS1, and TS1.TC1 numbers.

Thus all Requirements can be represented in a table format.

Business Requirement #	Test Scenario #	Test Case #	Defects #
BR1	TS1	TS1.TC1 TS1.TC2	D01
BR2	TS2	TS2.TC1 TS2,TC2 TS2.TC3	D02 D03
BR3	TS3	TS1.TC1 TS2.TC1 TS3.TC1 TS3.TC2	NIL

Advantage of the RTM:

Advantage of Requirement Traceability Matrix

- It confirms 100% test coverage
- It highlights any requirements missing or document inconsistencies
- It shows the overall defects or execution status with a focus on business requirements
- It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases.

17.1.2Automation feasibility report

Before we begin automating the manual test cases, we must first determine whether we can proceed with the automated test cases. In automation testing, feasibility analysis refers to a checklist that we use to determine if we should automate the test cases or not. Let's look at an example to help you grasp it better:

Without seeing the test cases, an automation team was assigned 500 test cases to automate and a time estimate of 30 days. When they began automating it, they ran into problems with item recognition, functional flow, and a variety of other challenges. As a result, the delivery of the scripts is delayed. If you don't want to be in the same tumultuous position, you must have a solid automated procedure in place. The feasibility analysis should be the initial stage in this procedure.

If you don't want to be in the same tumultuous position, you must have a solid automated procedure in place. The feasibility analysis should be the initial stage in this procedure.

This checklist consists of various factors upon which automation can be decided.

17.1.2.1 Functional Knowledge of Application

It is critical to understand the application's functional flow before beginning automation testing. The automation team need proper knowledge transfer (KT). Manual testers or developers can provide this knowledge transfer. This will make it easy for automated testers to partition test cases according to functionality and do feasibility studies.

17.1.2.2 Status of development of the Application

The development of the application should be frozen before the automation of the application begins, or else it will require more effort to adjust the script every time in the same cycle, causing the entire automation life cycle for that release to be delayed. In a nutshell, automation is only applicable to stable applications.

17.1.2.3 Required Access/Privileges for the application

Automation tool is able to identify the application in form of objects or not. This is yet another crucial need that must be met before we move on with automation. The program is divided into the number of items for UI automation. Only if the proposed automation tool can detect these things can we proceed with the automation; otherwise, automation with that tool is not possible. The only option left is to use another automation tool to identify the application's objects.

17.1.2.4 Percentage of the automation that can be achievable

When considering the automation of a certain application, we must first determine the minimal proportion of automation that is permissible. In most organizations, this figure hovers around 70%.

https://docs.google.com/spreadsheets/d/13MWEjn4tNqKgowBBbNjESl5vfb2xJgZb/edit?usp=sharing&ouid=113144539660973376722&rtpof=true&sd=true

17.2 Test Planning

Test Planning phase starts soon after the completion of the Requirement Analysis phase. In this phase the QA manager or QA Lead will prepare the Test Plan and Test strategy documents. As per these documents they will also come up with the testing effort estimations.

Activities to be done in Test Planning phase are given below:

- Estimation of testing effort
- Selection of Testing Approach
- Preparation of Test Plan, Test strategy documents
    - **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation and deliverables, and resources required for testing. Test Plan helps us determine the effort needed to validate the quality of the application under test.
- Resource planning and assigning roles and responsibility to them
- Selection of Testing tool

Deliverables (Outcome) of Test Planning phase are:

- Test Plan document
- Test Strategy document

    - Test Strategy is a high-level document (static document) and usually developed by project manager. It is a document which captures the approach on how we go about testing the product and achieve the goals. It is normally derived from the Business Requirement Specification (BRS). Documents like Test strategy doc is project-based document it can change according to project domain and requirements.
- Best suited Testing Approach
- Number of Resources, skill required and their roles and responsibilities
- Testing tool to be used

17.3 Test Case Development

In this phase the QA team write test cases. They also write scripts for automation if required. Verification of both the test cases and test scripts are done by peers. Creation of Test Data is done in this phase. Test data is the data that is used by the testers to run the test cases. Whilst running the test cases, testers need to enter some input data. To do so, testers prepare test data. It can be prepared manually and also by using tools.

Activities to be done in Test Case Development phase are given below:

- Creation of test cases
    - Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results.
- Creation of test scripts if required
- Verification of test cases and automation scripts
- Creation of Test Data in testing environment

Deliverables (Outcome) of Test Case Development phase are:

- Test cases
- Test scripts (for automation if required)

- Test Data

17.4 Test Environment setup

This phase includes the setup or installation process of software and hardware which is required for testing the application. In this phase the integration of the third party application is also carried out if required in the project.

After setting up the required software and hardware the installation of build is tested. Once the installation of build is successful and complete then the Test Data is generated.

After the creation of Test data the Smoke testing is executed on the build in order to check whether the basic functionalities are working fine or not. This phase can be done in parallel with the Test Case Development phase.

Activities to be done in Test Environment Setup phase are given below:

- As per the Requirement and Architecture document the list of required software and hardware is prepared
- Setting up of test environment
- Creation of test data
- Installation of build and execution of Smoke testing on it

Deliverables (Outcome) of Test Environment Setup phase are:

- Test Environment setup is ready
- Test Data is created
- Results of Smoke testing

17.5 Test Execution

Before starting the Test Execution phase the Test Environment setup should be ready. In Test Execution phase the test cases are executed in the testing environment.

While execution of the test cases the QA team may find bugs which will be reported against that test case. This bug is fixed by the developer and is retested by the QA.

Activities to be done in Test Execution phase are given below:

- Execution of Test Cases
- Reporting test results
- Logging defects for the failed test cases
- Verification and retesting of the defect
- Closure of defects

Deliverables (Outcome) of Test Execution phase are:

- Test execution Report
- Updated test cases with results
- Bug Report

17.6 Test Cycle Closure

In order to start the Test Cycle Closure activity the Test Execution phase should be completed. In Test Cycle phase the QA team will meet and discuss about the testing artifacts. Test Closure is the note prepared before test team formally completes the testing process. This note contains the total no. of test cases, total no. of test cases executed, total no. of defects found, total no. of defects fixed, total no. of bugs not fixed, total no of bugs rejected etc.

The whole intent of this discussion is to learn lessons from the bad practices. This will help in future projects.

Activities to be done in Test Cycle Closure phase are given below:

- To evaluate the test completion on the basis of Test Coverage and Software Quality
- Documentation of the learning from the project
- Analyzing the test results to find out the distribution of severe defects

- Test Closure Report preparation

Deliverables (Outcome) of Test Cycle Closure phase are:

- Report of Test Closure

18. What is SDLC?

The Software Development Lifecycle is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality Software's that meets customer expectations. The software development should be complete in the pre-defined time frame and cost.

18.1 Why SDLC?

Here, are prime reasons why SDLC is important for developing a software system.

- It offers a basis for project planning, scheduling, and estimating the project
- Provides a framework for a standard set of activities and deliverables
- Improved client relations, Increases visibility of project planning to all involved stakeholders of the development process
- Increased and enhanced development speed
- Helps you to decrease project risk and project management plan overhead
- It helps in effectively planning before starting the actual development. SDLC allows developers to analyze the requirements.
- It helps in reducing unnecessary costs during development. During the initial phases, developers can estimate the costs and predict costly mistakes

18.2 Limitations of SDLC

- It is difficult to estimate the actual cost of the entire project and the project overruns.
- It may lead to an increase in the cost of software development, especially if the customer requirements are not understood properly.

- It consists of specific requirements and phases that need to be completed, which increase the time taken in software development.
- Sometimes the input of users may be limited.
- The execution of SDLC phases depends on factors, such as customer requirements and the availability of funds.

18.3 SDLC Phases

18.3.1 Phase 1: Requirement gathering and analysis

This is a process with much communication taking place between stakeholders, end users and the project team. Meetings with managers, stake holders and users are held in order to determine the requirements like; who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirement gathering phase. **The QA engineer playing the role to configure the requirements using requirements traceability matrix (RTM).**

18.3.2 Phase 2: Feasibility study

This phase involves analyzing the feasibility of the plans and requirements made in the first stage.

- **Economic feasibility**: Are there enough funds to invest in the development of the software?
- **Legal feasibility**: Is the company able to comply with related regulations?
- **Operational feasibility:** Is it possible to meet the workflow and operational requirements set in the requirement stage?
- **Technical feasibility**: Does the organization have the necessary technology and human resources for the SDLC process?
- **Schedule feasibility:** Can the development process be completed on time?

18.3.3Phase 3: Design

In this phase the software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. **In this phase the QA Engineers comes up with the Test strategy, where they mention what to test, how to test.**

18.3.4Phase 4: Coding

Upon receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of SDLC. **In this phase the QA Engineers comes up with the Test Environment setup and test Case Documentation.**

18.3.5Phase 5: Testing

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. **During this phase all types of like unit testing, integration testing, Smoke Testing, functional testing,**

Sanity Testing, system testing, acceptance testing is done as well as non-functional testing are also done.

18.3.6Phase 6: Deployment

After successful testing the product is delivered / deployed to the customer for their use. As soon as the product is given to the customers, they will first do the beta testing/User Acceptance Testing. If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the bugs are fixed then the final deployment will happen.

18.3.7Phase 7: Maintenance

This phase involves solving issues faced by the customers when they use the software. When an issue is solved by the developers or software engineers, the software is tested to ensure it functions well. The software is then handed back to the customer for use. In the maintenance phase, the software can be enhanced to add other new features. It can also be upgraded to establish a new version of the system.

19. SDLC Models

19.1 Waterfall model

Waterfall model works well for smaller projects where requirements are very well understood. The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases. In this SDLC model, the outcome of one phase acts as the input for the next phase. This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

19.2 Incremental Model

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

19.3 V-Model

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

19.4 Agile Model

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks. In 'Agile Model' after every sprint there is a demo-able feature to the customer. Hence customer can see the features whether they are satisfying their need or not. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.

19.5 Spiral Model

The spiral model is a risk-driven process model. This SDLC model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. This model adopts the best features of the prototyping model and the waterfall model. The spiral

methodology is a combination of rapid prototyping and concurrency in design and development activities.

19.6 Prototyping Model

Prototyping model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved. It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail. It is an iterative, trial and error method which takes place between developer and client.

20. What is Verification & Validation in software testing?

Verification: Are we building the software right?

Validation: Are we building the right software?

Verification in Software Testing is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

Validation in Software Engineering is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

Verification process targets on software architecture, design, database, etc. while Validation process targets the actual software product.

21. What is Risk Factor and it's Types?

In software testing Risks are the possible problems that might endanger the objectives of the project stakeholders. It is the possibility of a negative or undesirable outcome. A risk is Something that has not happened yet and it may never happen; it is a potential problem.

The types of Risk in a Test Project can be broadly categorized as

1. Strategy Risk: This includes Budget, Communication and Management risks
2. Project Definition Risks: This includes Project target, Scope, and requirements risks.
3. Human Resources Risk: This includes Skill, Team members and organization risks.

22. List out Test Deliverables?

1. Test Strategy
2. Test Plan
3. Effort Estimation Report
4. Test Scenarios
5. Test Cases/Scripts
6. Test Data
7. Requirement Traceability Matrix (RTM)
8. Defect Report/Bug Report
9. Test Execution Report
10. Graphs and Metrics
11. Test summary report
12. Test incident report
13. Test closure report
14. Release Note
15. Installation/configuration guide
16. User guide
17. Test status report
18. Weekly status report (Project manager to client)

23. What is Test Coverage?

Test Coverage states which requirements of the customers are to be verified when the testing phase starts. Test Coverage is a term that determines whether the test cases are written and executed to ensure to test the software application completely, in such a way that minimal or NIL defects are reported.

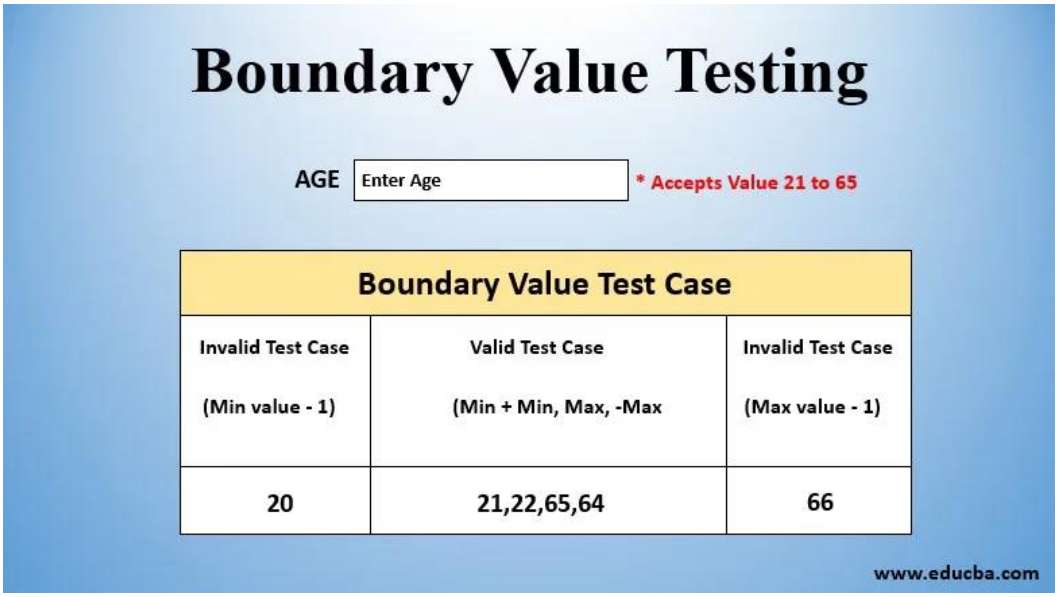
How to achieve Test Coverage?

The maximum Test Coverage can be achieved by establishing good 'Requirement Traceability'.

- Mapping all internal defects to the test cases designed
- Mapping all the Customer Reported Defects (CRD) to individual test cases for the future regression test suite.

24. What is Boundary Value Analysis?

Boundary value analysis (BVA) is based on testing the boundary values of valid and invalid partitions. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.



25. Equivalence Class Partitioning

Equivalence Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

It divides the input data of software into different equivalence data classes. You can apply this technique, where there is a range in the input field.

Example 1: Equivalence and Boundary Value

- Let's consider the behavior of Order Pizza Text Box Below
- Pizza values 1 to 10 is considered valid. A success message is shown.
- While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered"

Here is the test condition

1. Any Number greater than 10 entered in the Order Pizza field (let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.

26. What is the difference between a Standalone, Client-Server and Web application?

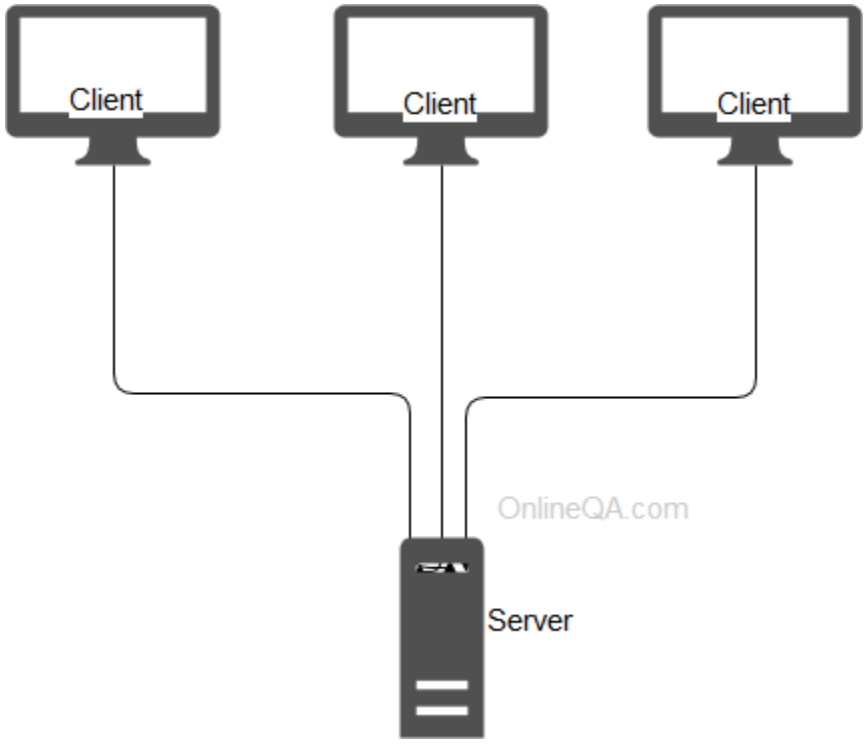
Standalone application:

Software installed in one computer and used by **only one** person.

For ex – Installing s/w of a Calculator, Adobe Photoshop, MS Office, AutoCad. Standalone applications follow one-tier architecture. Presentation, Business, and Database layer are in one system for a single user.

Client-Server Application:

In Client Server application, unlike Standalone Application, part of application is installed on to the client system and the remaining part is installed on to the server machine. Client-server applications follow two-tier architecture. Presentation and Business layer are in a client system and Database layer on another server. Both Client and server interact with the help of network/internet.



Web Application:

A web application is a computer program that uses a web browser to perform a particular function. It is also called a web app. A web application is a client-server program. It means that it has a client-side and a server-side. The term "client" here refers to the program the individual uses to run the application. It is part of the client-server environment, where many computers share information. For example, in the case of a database, the client is the program through which the user enters data. The server is the application that stores the information. Web

server applications follow three-tier or n-tier architecture. The presentation layer is in a client system, a Business layer is in an application server and Database layer is in a Database server. It works both in Intranet and Internet.

27. Can you do System testing at any stage of SDLC?

We can do System Testing only when all the units are in place and working properly. It can only be done before User Acceptance Testing (UAT).

28. When to stop testing? (Or) How do you decide when you have tested enough?

There are many factors involved in the real-time projects to decide when to stop testing.

1. Testing deadlines or release deadlines
2. By reaching the decided pass percentage of test cases
3. The risk in the project is under acceptable limit
4. All the high priority bugs, blockers are fixed
5. When acceptance criteria are met.

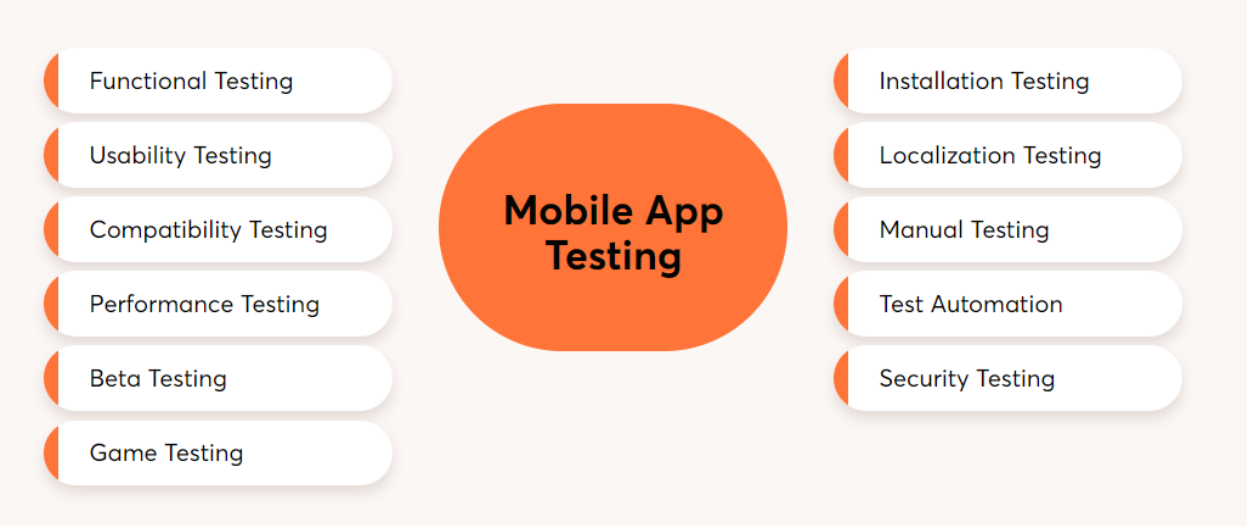
29. What information should be included in a Defect or Bug report?

1. A brief summary of the defect
2. A full description of the defect including steps to reproduce
3. Screenshot attachments if required
4. Date the defect was found and raised
5. Who reported the defect?
6. Severity and/or Priority of the defect
7. Which component is the defect assigned?

8. Current status of Bug.

30. Mobile application testing

Testing involves in mobile app testing:



30.1 Mobile App Testing Parameters

- **Screen sizes**

Thousands of mobile phones are pumped into the market with varied screen size, aspect ratio and pixel densities.

- **Internet strength**

With respect to countries, internet providers and connectivity type internet strength can vary drastically and can affect app performance to a great extent

- **OS versions**

Android versions upgrade usually takes time owing to fragmentation. Meanwhile iOS upgrade takes place swiftly. So our testers check your app on various versions to ensure stable performance on each version.

- **Power consumption**

There is not much innovation happening in battery industry. Owing to that, app optimization check has to be done.

- **Usability**

Owing to the size and shape of mobile phone screens, usability testing indeed a hurdle.

- **Security issues**

Breaches are happening at an alarming rate. To validate the measures adopted by the developers to secure apps is indeed challenging.

- **Contextual issues**

Context can enhance or limit the functionalities of apps. Since apps have varied context with different data, testing process can become complex.

30.2 Functional testing

Testing is done by certifying the requirements like whether the application is working based on the requirements or not.

30.3 Android/IOS UI/Responsiveness testing

This is a user-centric testing of the application. In this test phase, items such as visibility of text in various screens of the app, interactive messages, alignment of elements, the look and feel of the app for different screens, size of fields etc. are tested under this.

Most important point of this testing:

- Device Selection (must prefer Real device always)
- Device emulators are cost effective and they come in handy during the initial development phase.
- But, to test the real-life scenarios, physical devices are the must. Both emulators and physical devices are to be used in a balanced manner for an optimized result.

30.4 Compatibility testing

The extension for Android apps is .APK. and for iOS apps is .ipa should be confirm. This testing is done mostly in the form of two matrices of OS Vs app and Device
Model Vs App. Usually, a list of supported OS (and sometimes devices) is provided by the product owner or customer.

30.5 Interface Testing

This testing is done after all the modules of the app are completely developed, tested individually and all the bugs are fixed verified.

30.6 Network Testing

During this testing, request/response to/from the service is tested for various conditions. This test is mainly done to verify the response time in which the activity is performed like refreshing data after sync or loading data after login etc.

30.7 Performance Testing

Performance of the application under some peculiar conditions are checked. Those conditions include:

- Low memory in the device.
- The battery in extremely at a low level.
- Poor/Bad network reception.

30.8 Installation/Uninstallation testing

This is to ensure smooth installation and uninstallation of the application without ending up in errors, partial installation etc.

30.9 Security Testing

Testing of the data flow for encryption and decryption mechanism is to be tested in this phase. Access to stored data is also tested in this phase.

30.10 Field testing

Field testing is done specifically for the mobile data network and not in-house but by going out and using the app as a normal user.

It is basically done to verify the behavior of the app when the phone has a 2G or 3G connection.

Field testing verifies if the app is crashing under slow network connection or if it is taking too long to load the information.

30.11 Interrupt Testing

This is the Offline Scenario Verification. Conditions where the communication breaks in the middle are called as offline conditions.

Some of the conditions where interruptions of a network can be tested are as follows:

- Data cable removal during data transfer process.
- Network outage during the transaction posting phase.
- Network recovery after an outage.
- Battery removal or Power On/Off when it is in the transactional phase.

31. Difference between website and web Application

WEBSITE	WEB APPLICATION
They Are Marketing Focussed And Are Used To Tell People About Your Business Brand.	They Are Basically Web Tools That Perform Certain Functionalities And Are Less Focussed Around Brand Marketing.
They Are Used For Communication Purposes.	They Are Used For Automating Tasks To Achieve Certain Goals.
They Allow Data Visualisation (Reading And Comprehending).	They Permit Data Manipulation.
Websites Don't Require Authentication Services.	They Can Be Called As 'web Portals' Or 'online Stores'.
They Are Less Resource Intensive And Needs Less Processing Power.	They Are More Resource Intensive And Need More Processing Power.
They Are Less Complex And Difficult To Code.	Their General As Well As Coding Complexity Is High.

32. API Testing

32.1 What is an API?

An API (Application Programming Interface) is a software intermediary that enables two applications to communicate with each other. API works as; it takes a request from the source, takes that request to the database, fetches the request data from the database and returns a response to the source. API takes the requests from the user and gives the response without exposing the internal details. API acts as Abstraction.

Example: Amazon API, Google Map API

32.2 What is API testing?

API testing is a type of software testing that involves testing APIs directly. API is a part of integration testing to check whether the API meets expectations in terms of functionality, reliability, performance, and security of applications. Multiple API system can performed API testing. In API testing, our primary focus is on Business Logic Layer of the software architecture.

32.3 What are the types of API testing?

API testing involves the following types of testing:

- Unit Testing
- Functional Testing
- Load Testing
- Runtime/Error Detection
- Security Testing
- UI Testing
- Interoperability and WS compliance Testing
- Penetration Testing
- Fuzz Testing

32.4 What are the protocols used in API Testing?

Protocols used in API testing are:

- HTTP
- REST
- SOAP
- JMS
- UDDI

32.5 What are the advantages of API Testing?

Test for Core Functionality:

API testing provides access to the application without a user interface. The core and code-level of functionalities of the application will be tested and evaluated early before the GUI tests. This will help detect the minor issues which can become bigger during the GUI testing.

Time Effective:

API testing usually is less time consuming than functional GUI testing. The web elements in GUI testing must be polled, which makes the testing process slower. Particularly, API test automation requires less code so it can provide better and faster test coverage compared to GUI test automation. These will result in the cost saving for the testing project.

Language-Independent:

In API testing, data is exchanged using XML or JSON. These transfer modes are completely language-independent, allowing users to select any code language when adopting automation testing services for the project.

Easy Integration with GUI:

API tests enable highly integrable tests, which is particularly useful if you want to perform functional GUI tests after API testing. For instance, simple integration would allow new user accounts to be created within the application before a GUI test started.

32.6 What are the tools used for API Testing?

Tools used for API testing are:

- Parasoft SOAtest
- PostMan
- AlertSite API monitoring

32.7 What are the limits of API usage?

Many APIs have certain limit set up by the provider. Hence, try to estimate our usage and understand how that will impact the overall cost of the offering.

32.8 What are the common tests that performed on API?

Here, are the common tests that performed on API are as:

- Response of the API should be verified based on the request. We will verify that the return value is based on request.
- When API is updating any data structure we should verify the system is authenticating the outcome.
- We will verify whether the API is trigger other event or request another API.
- We will verify the behavior of the API when no value is return.

32.9 What exactly needs to verify in API testing?

In API testing, we send a request to API with the known data and then analysis the response.

1. We will verify the accuracy of the data.
2. Will see the HTTP status code.
3. We will see the response time.
4. Error codes in case API returns any errors.
5. Authorization would be check.
6. Non-Functional testing such as performance testing, security testing.

32.10 What are major challenges faced in API testing?

If you can overcome the challenges in API Testing, you can be confident in the API testing interview too. They are:

- Parameter Selection

- Parameter Combination
- Call sequencing
- Output verification and validation
- Another important challenge is providing input values, which is very difficult as GUI is not available in this case.

32.11 What kinds of bugs that API testing would often find?

- Missing or duplicate functionality
- Fails to handle error conditions gracefully
- Stress
- Reliability
- Security
- Unused flags
- Not implemented errors
- Inconsistent error handling
- Performance
- Multi-threading issues
- Improper errors

32.12 What are API documentation templates that are commonly used?

There are several available API documentation templates help to make the entire process simple and straightforward, such as:

- Swagger

- Miredot
- Slate
- FlatDoc
- API blueprint
- RestDoc
- Web service API specification

32.13 What is REST?

REST (Representational State Transfer) is an architectural style for developing web services which exploit the ubiquity of HTTP protocol and uses HTTP method to define actions. It revolves around resource where every component being a resource that can be accessed through a shared interface using standard HTTP methods.

32.14 What is a RESTFul Web Services?

Mostly, there are two kinds of Web Services which should be remembered in your next API testing interview:

1. SOAP (Simple Object Access Protocol) – an XML-based method to expose web services.
2. Web services developed in the REST style are referred to as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A

   RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation like JSON and a set of HTTP methods.

32.15 What are the differences between SOAP and REST API?

Sr. No.	SOAP API	REST API
1.	**SOAP** stands as Simple Object Access Protocol.	**REST** stands as Representational State Transfer.
2.	**SOAP** is a protocol.	**REST** is an architectural pattern.
3.	**SOAP** can work with XML format. In SOAP all the data passed in XML format.	**REST** permit different data format such as Plain text, HTML, XML, JSON etc. But the most preferred format for transferring data is in JSON.

32.16 What are the major challenges faced during API testing?

The major challenges faced during the API testing are:

- Parameter Selection
- Parameter Combination
- Call sequencing
- Output verification and validation
- A major challenge is providing input values which are very difficult because GUI is not available.

32.17 What are the components of an HTTP request?

An HTTP request have five components. These are:

1. **Action showing HTTP method** like GET, PUT, POST, DELETE.
2. **Uniform Resource Identifier (URI):** URI is the identifier for the resource on the server.
3. **HTTP version:** Indicate the HTTP version like- HTTP V1.1.

4. **Request Header:** Request Header carries metadata for the HTTP request message. Metadata could be a client type, format supported by the client, format of a message body, cache setting etc.
5. **Request Body:** Resource body indicates message content or resource representation.

32.18 What are the most commonly used HTTP methods supported by REST?

- GET is only used to request data from a specified resource. Get requests can be cached and bookmarked. It remains in the browser history and has length restrictions. GET requests should never be used when dealing with sensitive data.
- POST is used to send data to a server to create/update a resource. POST requests are never cached and bookmarked and do not remain in the browser history.
- PUT replaces all current representations of the target resource with the request payload.
- DELETE removes the specified resource.
- OPTIONS is used to describe the communication options for the target resource.
- HEAD asks for a response identical to that of a GET request, but without the response body.

32.19 What is payload in RESTFul Web services?

The “payload” is the data you are interested in transporting. This is differentiated from the things that wrap the data for transport like the HTTP/S Request/Response headers, authentication, etc.