Assignment 2:-

**Q - What is Exploratory Testing?**

A – It is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.

Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

**Q – What is traceability matrix?**

A – **It** is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

**Q - What is Boundary value testing?**

A - Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects.

It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

**Q - What is Equivalence partitioning testing?**

A - In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.

**Q – What is Integration testing?**

A - Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as ‘I & T’ (Integration and Testing), ‘String Testing’ and sometimes ‘Thread Testing’.

**Q - What determines the level of risk?**

A - The likelihood of an adverse event and the impact of the event.

**Q - What is Alpha testing?**

A - Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of acceptance testing. The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

**Q - What is beta testing?**

A - Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of User Acceptance Testing. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers.

**Q - What is component testing?**

A - Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It’s also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.

It’s one of most frequent black box testing types which is performed by QA Team.

**Q - What is functional system testing?**

A - a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

**Q - What is Non-Functional Testing?**

A - Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

**Q - What is GUI Testing?**

A - GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.

GUI is what the user sees. Say if you visit guru99.com what you will see say homepage it is the GUI (graphical user interface) of the site. A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

**Q - What is Adhoc testing?**

A - Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called Error Guessing. Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

**Q - What is load testing?**

A - Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

**Q - What is stress Testing?**

A - Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn’t crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions

**Q - What is white box testing and list the types of white box testing?**

A - a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Black box testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term “White Box” was used because of the see-through box concept. The clear box or White Box name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings.

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below

Unit Testing: It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.

Testing for Memory Leaks: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Following are important WhiteBox Testing Techniques:

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage

**Q - What is black box testing? What are the different black box testing techniques?**

A - a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Types of Black Box Testing

There are many types of Black Box Testing are as below:-

Functional testing – This black box testing type is related to the functional requirements of a system; it is done by software testers.

Non-functional testing – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.

Regression testing – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

Equivalence Class Testing: It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.

Boundary Value Testing: Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.

Decision Table Testing: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

**Q - Mention what are the categories of defects?**

A –

Functional defects

Functional bugs can be revealed during smoke, system, integration, regression, and user acceptance testing. A feasible share of test automation during functional testing is the key to making the process cost- and time-effective.

Performance defects

Performance bugs are found during stress, load, stability, and scalability testing. This kind of testing is fully automated. The most popular performance testing tools are Apache JMeter and LoadRunner.

Usability defects

Usability defects are revealed during usability testing, UX audit, or UX research. ScienceSoft applies expert-based (e.g., heuristic evaluation and cognitive walkthrough) and user-based (interviews and surveys of the TA, executing scenarios by members of the TA) techniques to promptly detect usability issues.

Security defects

Detecting security defects requires vulnerability assessment, penetration testing, security code review, and more. Software compliance assessment (e.g., for HIPAA, PCI DSS, GDPR) can be considered a part of security testing as well. Security testing can be both manual and automated, depending on the needs of each specific project.

**Q - Mention what big bang testing is?**

A - Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

**Q - What is the purpose of exit criteria?**

A - Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

Examples of Exit Criteria:

* Verify if All tests planned have been run.
* Verify if the level of requirement coverage has been met.
* Verify if there are NO Critical or high severity defects that are left outstanding.
* Verify if all high risk areas are completely tested.
* Verify if software development activities are completed within the projected cost.
* Verify if software development activities are completed within the projected timelines.

**Q – When should "Regression Testing" be performed?**

A - There is a need for regression testing whenever the code is changed, and you need to determine whether the modified code will affect other parts of the software application. Moreover, regression testing is needed when a new feature is added to the software application. Regression tests may also be performed when a functional or performance defect/issue is fixed.

**Q - What is 7 key principles? Explain in detail?**

A –

1. Testing shows the presence of defects: The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn’t talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.
2. Exhaustive testing is not possible: It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.
3. Early Testing: To find the defect in the software, early test activity shall be started. The defect detected in the early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.
4. Defect clustering: In a project, a small number of modules can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.
5. Pesticide paradox: Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.
6. Testing is context-dependent: The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the Android application.
7. Absence of errors fallacy: If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil all the customer requirements.

**Q - Difference between QA v/s QC v/s Tester**

A -

|  |  |  |  |
| --- | --- | --- | --- |
|  | **QA** | **QC** | **Tester** |
| **Purpose** | Setting up adequate processes, introducing the standards of quality to prevent the errors and flaws in the product. | Making sure that the product meets the requirements and specifications before it is released. | Detecting and solving software errors and flaws. |
| **Focus** | Processes | Whole product | Source code & Design |
| **What** | Prevention | Verification | Detection |
| **Who** | The team including the stakeholders | The team | Test engineers & Developers |
| **When** | Throughout the process | Before the release | At the testing stage or along with the development process |

**Q - Difference between Smoke and Sanity?**

A –

|  |  |
| --- | --- |
| **Smoke** | **Sanity** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of Regression Testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

**Q - Difference between Verification and Validation**

A –

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| The verifying process includes checking documents, design, code, and program | It is a dynamic mechanism of testing and validating the actual product |
| It does not involve executing the code | It always involves executing the code |
| Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc. | It uses methods like Black Box Testing, White Box Testing, and non-functional testing |
| Whether the software conforms to specification is checked | It checks whether the software meets the requirements and expectations of a customer |
| It finds bugs early in the development cycle | It can find bugs that the verification process can not catch |
| Target is application and software architecture, specification, complete design, high level, and database design etc. | Target is an actual product |
| QA team does verification and make sure that the software is as per the requirement in the SRS document. | With the involvement of testing team validation is executed on software code. |
| It comes before validation | It comes after verification |

**Q - Explain types of Performance testing.**

A –

**Load testing** – checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.

**Stress testing** – involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.

**Endurance testing** – is done to make sure the software can handle the expected load over a long period of time.

**Spike testing** – tests the software’s reaction to sudden large spikes in the load generated by users.

**Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.

**Scalability testing** – The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

**Q – What is Error, Defect, Bug and failure?**

A - **Error**

An error is a mistake made by human that leads to discrepancy between the actual and the expected result.

**Defect**

A defect is a problem in the functioning of a software system during testing. ISTQB defines a defect as “A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g., an incorrect statement or data definition.”

**Bug**

A bug is a flaw in a software system that causes the system to behave in an unintended manner.

**Failure**

A failure is the inability of a software system to perform its operations within the specified performance benchmark.

**Q - Difference between Priority and Severity**

|  |  |
| --- | --- |
| **Priority** | **Severity** |
| Priority is the order in which the developer should resolve a defect | Severity is the degree of impact that a defect has on the operation of the product. |
| Priority is categorized into three types: low, medium and high | Severity is categorized into five types: critical, major, moderate, minor and cosmetic. |
| Priority is associated with scheduling while | Severity is associated with functionality or standards. |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality. |
| Priority of defects is decided in consultation with the manager/client | Severity levels of the defects are determined by the QA engineer. |
| Priority is driven by business value | Severity is driven by functionality. |
| Priority value is subjective and can change over a period of time depending on the change in the project situation | Severity value is objective and less likely to change. |
| High Priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application while High | Severity and low priority status indicates defect have to be fixed but not on immediate bases. |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product. |

**Q - What is Bug Life Cycle?**

A - Defect Life Cycle or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

**Q - Explain the difference between Functional testing and NonFunctional testing**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Functional** | **Non-functional** |
| **Execution** | It is performed before non-functional testing. | It is performed after the functional testing. |
| **Focus area** | It is based on customer’s requirements. | It focusses on customer’s expectation. |
| **Requirement** | It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| **Usage** | Helps to validate the behaviour of the application. | Helps to validate the performance of the application. |
| **Objective** | Carried out to validate software actions. | It is done to validate the performance of the software. |
| **Requirements** | Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| **Manual testing** | Functional testing is easy to execute by manual testing. | It’s very hard to perform non-functional testing manually. |
| **Functionality** | It describes what the product does. | It describes how the product works. |
| **Example Test Case** | Check login functionality. | The dashboard should load in 2 seconds. |
| **Testing Types** | Examples of Functional Testing Types   * Unit testing * Smoke testing * User Acceptance * Integration Testing * Regression testing | Examples of Non-functional Testing Types   * Performance Testing * Volume Testing * Scalability * Usability Testing * Load Testing * Stress Testing |

**Q - What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

**A -**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **STLC** | **SDLC** |
| **Origin** | **Testing Life Cycle** | **Development Life Cycle** |
| Objective | The only objective of the STLC phase is testing. | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. |
| Req. Gathering | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan | * In SDLC the business analyst * gathers the requirements and   create Development Plan |
| High & Low Level Design | In STLC, the test analyst creates the Integration Test Plan | In SDLC, the development team creates the high and low-level design plans |
| Coding | The testing team prepares the test environment and executes them | * The real code is developed,   and actual work takes place as per the design documents. |
| Maintenance | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. | SDLC phase also includes post-deployment supports and updates. |

**Q - What is the difference between test scenarios, test cases, and test script?**

A - Difference between test scenarios and test cases is that

**Test Scenarios:** A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.

**Test Cases:** It is a document that contains the steps that has to be executed, it has been planned earlier.

**Test Script:** It is written in a programming language and it's a short program used to test part of functionality of the software system. In other words a written set of steps that should be performed manually.

**Q - Explain what Test Plan is? What is the information that should be covered.**

* **A – Test Plan :-** A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”

* **Scope:** Details the objectives of the particular project. Also, it details user scenarios to be used in tests. If necessary, the scope can specify what scenarios or issues the project will not cover.
* **Schedule**: Details start dates and deadlines for testers to deliver results.
* Resource Allocation: Details which tester will work on which test.
* **Environment**: Details the nature, configuration, and availability of the test environment.
* **Tools**: Details what tools are to be used for testing, bug reporting, and other relevant activities.
* **Defect Management**: Details how bugs will be reported, to whom and what each bug report needs to be accompanied by. For example, should bugs be reported with screenshots, text logs, or videos of their occurrence in the code?
* **Risk** **Management**: Details what risks may occur during software testing, and what risks the software itself may suffer if released without sufficient testing.
* **Exit** **Parameters**: Details when testing activities must stop. This part describes the results that are expected from the QA operations, giving testers a benchmark to compare actual results to.

**Q - What is priority?**

**A -** Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.

Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

**Q - What is severity?**

A - Defect Severity is defined as the degree of impact that a defect has on the operation of the product

**Q - Bug categories are…**

A -

* **Functional Bugs**

Functional bugs are associated with the functionality of a specific software component.

In simple terms, any component in an app or website that doesn’t function as intended is a functional bug.

* **Logical Bugs**
* A logical bug disrupts the intended workflow of software and causes it to behave incorrectly. These bugs can result in unexpected software behavior and even sudden crashes. Logical bugs primarily take place due to poorly written code or misinterpretation of business logic.
* **Workflow Bugs**

Workflow bugs are associated with the user journey (navigation) of a software application. Let’s consider an example of a website where a user needs to fill up a form regarding their medical history.

* **Unit Level Bugs**

Unit level bugs are very common, and they are typically easier to fix. Once the initial modules of software components are developed, developers perform unit testing to ensure that the small batches of code are functioning as expected. Here’s where developers encounter various bugs that get overlooked in the coding stages.

* **System-Level Integration Bugs**

System-level integration bugs primarily pop up when two or more units of code written by different developers fail to interact with each other. These bugs primarily occur due to inconsistencies or incompatibility between two or more components. Such bugs are difficult to track and fix as developers need to examine a larger chunk of code. They are also time-consuming to replicate.

* **Out of Bound Bugs**

Out of Bound Bugs show up when the system user interacts with the UI in an unintended manner. These bugs occur when an end-user enters a value or a parameter outside the limits of unintended use — for example, entering a significantly larger or a smaller number or entering an input value of an undefined data type. These bugs often pop up in form validations during functional testing of web or mobile apps.

**Q - Advantage of Bugzila .**

**A -**

* it is an open-source widely used bug tracker;
* it is easy in usage and its user interface is understandable for people without technical knowledge;
* it easily integrates with test management instruments;
* it integrates with an e-mailing system;

it automates documentation.

**Q - Difference between priority and severity**

A - Repeat Question.

**Q - What are the different Methodologies in Agile Development Model?**

* **Kanban**
* **Scrum**
* **Extreme Programming (XP)**
* **Crystal**
* **Dynamic Systems Development Method (DSDM)**
* **Feature Driven Development (FDD)**
* **Lean Software Development**
* **Scaled Agile Framework (SAFe)**

**Q - Explain the difference between Authorization and Authentication in Web testing.**

**A -**

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| Authentication is the process of identifying a user to provide access to a system. | Authorization is the process of giving permission to access the resources. |
| In this, the user or client and server are verified. | In this, it is verified that if the user is allowed through the defined policies and rules. |
| It is usually performed before the authorization. | It is usually done once the user is successfully authenticated. |
| It requires the login details of the user, such as user name & password, etc. | It requires the user's privilege or security level. |
| Data is provided through the Token Ids. | Data is provided through the access tokens. |
| Example: Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software. | Example: After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles. |
| Authentication credentials can be partially changed by the user as per the requirement. | Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it. |

**Q - What are the common problems faced in Web testing?**

**A -**

* Cross Browser Compatibility
* Responsiveness
* Cross-Device Compatibility
* Integration Testing
* Security
* Performance Testing
* Application Getting Slow
* Usability Testing
* Entry and Exit Points
* Checking the Standards and Compliance
* Firewalls
* Accessibility Testing
* Project Deadline
* User Experience
* Web Service Requests
* User Input Validation

**Q – Write a scenario of only Whatsapp chat messages**

**A -**

* Check the Chat window that contains the entire chat list.
* Check the Chat window displays the contact numbers whose numbers are not saved on mobile.
* Check the Chat window displayed with all contacts with DP or without DP
* Check the Chat window is displayed on the group chat list.
* Check the Chat window displays the last updated chatting time.
* Check the user can send and receive GIFs in the individual chat boxes.
* Check the user can send and receive Stickers in the individual chatboxes.
* Check the user can delete text, video, audio, locations, and documents in the individual chatboxes.
* Check the user can send recorded voice mail in an individual chatbox.
* Check the user can delete the entire chat history in the individual chatbox.
* Check the Chat window displays the name of all contacts on the chat window.
* Check the clicking on one Chat contact then a new window should open with history.
* Check the user can see all delivered and received messages.
* Check the user can see the read or send time of messages.
* Check the user can send and receive text messages in the individual chatbox.
* Check the user can send and receive documents in the individual chatbox.
* Check the user can send and receive photos in an individual chatbox.
* Check the user can send and receive videos in an individual chatbox.
* Check the user is able to voice call in the individual chat box.
* Check the user is able to mute the individuals in the individual chat boxes.
* Check the user is able to change the wallpaper.
* Check the users have options like Report, Block, Clear Chat, Export Chat, and Add Shortcut.
* Check the user can send and receive audio in an individual chat box.
* Check the user can send and receive emotions icons in the individual chat boxes.
* Check the user can send and receive Contacts in the individual chat boxes.
* Check the user can send and receive Location in the individual chatbox.
* Check the user is able to see contact details in the individual chat box.
* Check the user is able to share images, links, and documents from media in the individual chatboxes.
* Check the user is able to search specific chat history using the search option in the individual chatbox.
* Check the user is able to video call in the individual chat box.

**Q - When to used Usablity Testing?**

A – Usability Testing also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user’s ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.

This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

**Q - What is the procedure for GUI Testing?**

A –

* Check all the GUI elements for size, position, width, length, and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
* Check you can execute the intended functionality of the application using the GUI
* Check Error Messages are displayed correctly
* Check for Clear demarcation of different sections on screen
* Check Font used in an application is readable
* Check the alignment of the text is proper
* Check the Colour of the font and warning messages is aesthetically pleasing
* Check that the images have good clarity
* Check that the images are properly aligned
* Check the positioning of GUI elements for different screen resolution.