# **Software Testing Assignment**

## **Module-2(Manual Testing)**

**Q**. What is Exploratory Testing?

**A:** Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution. It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests. It is highly effective approach to testing software that gives testers the freedom to adapt and experiment on the fly based of their observations of the system and users behaviors. It allows unlocking the full potential of testing team and achieving better results by catching issues that rigid and traditional testing approaches might miss.

**Q**. What is traceability matrix?

**A:** A Traceability Matrix is a document that maps or rather traces the relationship between two baseline documents. Here, one of the documents has the requirement specifications, whereas the other one has test cases. RTM stands for Requirements Traceability Matrix. RTM maps all the requirements with the test cases. By using this document one can verify test cases cover all functionality of the application as per the requirements of the customer.

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

**A:** Boundary Value Testing is based on testing the boundary values of valid and invalid partitions. The behavior at the edge of the equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects. It is also called Boundary Value Analysis and it is a method which refines equivalence partitioning.

1. What is Equivalence partitioning testing?

**A:** **Equivalence Partitioning Testing** is also known as Equivalence class partitioning (ECP). It is a [software testing](https://www.geeksforgeeks.org/software-testing-basics/) technique or [black-box testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/) that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed. 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.

1. What is Integration testing?

**A: Integration testing** is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed. There are two levels of integration testing which are as below

* + Component Integration Testing
  + System Integration Testing

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

**A:** Risks should be prioritized according to their level, which is obtained by assessing the likelihood of the event occurring and the impact of that event. Then, the residual level should be determined by considering the management response to the risk.

1. What is Alpha testing?

**A:** Alpha testing is the first end-to-end testing of a product to ensure it meets the business requirements and functions correctly. It is typically performed by internal employees and conducted in lab/stage environment. An alpha test ensures the product really works and does everything it’s supposed to do. Sometimes it is also performed by Independent Testing Team. Alpha Testing is not open to the market and public. It comes under the category of both White Box Testing and Black Box Testing.

**Q.** What is beta testing?

**A:** Beta testing is the final round of testing before releasing a product to a wide audience. The objective is to uncover as many bugs or usability issues as possible in this controlled setting. It is an opportunity for real users to use a product in a production environment to uncover any bugs or issues before a general release. Beta Testing is always open to the market and public. It is performed in Real Time Environment. It is only a kind of Black Box Testing.

**Q.** What is component testing?

**A:** Component testing, also known as unit testing or module testing, is a level of software testing that focuses on verifying the individual components or units of a system. A component refers to a self-contained module or a group of related functions within the software. Component testing is performed before integration testing and after unit testing. In SDLC, during the development phase, developers will do [unit testing](https://www.browserstack.com/guide/unit-testing-a-detailed-guide) on that component and release a build to the QA team for testing that particular component alone. The entry criteria for component testing is the unit test completion. Then testers will do rigorous testing and validate the functionality against the requirements and report if any bugs are identified.

1. What is functional system testing?

**A:** Functional system testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. It validates web or mobile applications against pre-determined specifications and requirements. The goal is to ensure that the software performs as expected by both developers and end-users.

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

**A:** Non-functional testing is a type of software testing that verifies non-functional aspects of the product, such as performance, stability, and usability. It verifies how well the product performs. Non-Functional testing assesses application properties that aren’t critical to functionality but contribute to the end-users experience, such as reliability, efficiency, usability, stability and portability. Non-functional testing should be performed after functional testing.

1. What is GUI Testing?

**A:** GUI stands for Graphical User Interface. It is the process of ensuring proper functionality of the graphical user interface (GUI) for a specific application. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

**Q.** What is Adhoc testing?

**A:** Adhoc testing is a type of software testing the is performed informally and randomly after the formal testing is completed to find any loophole in the system. For this reason, it is also known as Random or Monkey Testing. Adhoc testing is not performed in a structured way so it is not based on any methodological approach. It is a type of Unstructured Software Testing. It requires no documentation, no test design, and no test case. Adhoc testing can be achieved with the testing techniques 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.

1. What is load testing?

**A:** Load testing is performed to determine a systems behavior under both normal and anticipated peak load conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks (causing the traffic to slow down or stop) and determine which element is causing degradation. Load testing checks how the system reacts when load is gradually increased. It does not break the system.

**Q.** What is stress Testing?

**A:** Stress testing is software testing used to determine how an application or system behaves when under extreme levels of stress. This testing puts the system or application through scenarios designed to push it beyond its normal limits. Stress testing tries to break the system by testing with overwhelming data or resources. Stress testing can be used to test new features and upgrades before they are released into production. This helps ensure that any changes will not cause unexpected errors or performance issues in the live environment. Stress testing can also help identify potential security risks by exposing any vulnerability in the system or application before they become an issue in production.

1. What is white box testing and list the types of white box testing?

**A:** White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level. It is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

Types of White Box Testing techniques are listed below: -

* + Unit Testing
  + Loop Testing
  + Path Testing
  + Static Analysis
  + HtmlUnit
  + Security Testing
  + JUnit
  + Performance Testing
  + System Integration
  + Mutation Testing
  + White box Penetration Testing
  + Condition Coverage
  + Path Coverage
  + Conditional Testing
  + Memory Leak Tests
  + Structural Testing
  + Branch Coverage
  + Statement Coverage
  + Control Flow Testing
  + CppUnit
  + Regression Testing
  + Dynamic Analysis
  + NUnit

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

**A:** Black box testing is a testing technique in which the internal workings of the software are not known to the tester. The tester only focuses on the input and output of the software. It is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements. Here, the tester has no knowledge of how the system or component is structured inside the box. In black-box testing, the tester is concentrating on what the software does, and not on how it does..

Different Black Box Testing Techniques are listed below: -

* + State Transition Testing
  + Decision Table Testing
  + Regression Testing
  + Path Testing
  + All-Pair Testing
  + Dependency on Requirements
  + Selenium
  + Boundary Value Analysis
  + Functional Testing
  + Comparison Testing
  + Use Case
  + Appium
  + LoadRunner
  + Equivalence Partitioning
  + Error Guessing
  + Prioritize Test Cases
  + JUnit
  + Black Box Testing
  + Non-Functional Testing

1. Mention what are the categories of defects?

**A:** Quality control professionals typically classify quality defects into three main categories: Minor, Major, and Critical. The nature and severity of a defect determine which of the three categories it belongs to. The defect is the occurrence of variance between the expected result and the actual result.

It can be further classified as below:

* + Database defect
  + Critical Functionality defect
  + Functionality defect
  + Security defect
  + User Interface defect

**Q.** Mention what bigbang testing is?

**A:** Big Bang Testing is a type of Integration Testing. It is a testing approach where all components or modules are integrated and tested as a single unit. This is done after all modules have been completed and before any system-level testing is performed.

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

**A:** Exit criteria are conditions that signify the completion of the testing phase. They help determine when it’s time to wrap up testing and move forward in the software development lifecycle. It provides a sense of closure and help evaluate the overall quality of the software.

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

**A:** Regression testing should be done to ensure that existing functionality works as expected after making changes to code, such as adding new features, fixing bugs of previously tested program, or improving performance. Regression testing is performed before each new instance of the product is deployed, guaranteeing that the program works perfectly in each setting. For instance, we need to make sure the product continues to perform properly in a production environment before we release it.

1. What is 7 key principles? Explain in detail?

**A:** Software testing has 7 key principles which are as follows:

1. Testing Shows Presence of Defects

2. Exhaustive Testing is Impossible

3. Early Testing

4. Defect Clustering

5. Pesticide Paradox

6. Testing is Context Dependent

7. Absence-of-Errors Fallacy

**Testing Shows Presence of Defects:** Testing can reveal the presence of defects, but it cannot prove their absence. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness. We test to find Faults. As we find more defects, the probability of undiscovered defects remaining in a system reduces.

**Exhaustive Testing is impossible:** It’s impossible to test every possible input combination and scenario. So, testing efforts should be focused on areas with the highest probability of finding defects.

**Early Testing:** Testing should start as early as possible in the software development lifecycle to defect defects early and reduce the cost of fixing them.

**Defect Clustering:** A small number of modules typically contain the majority of defects. So, focusing testing efforts on these high-risk areas is important.

**Pesticide Paradox:** If the same tests are repeated over and over again, eventually, they will no longer find new defects. Test cases need to be regularly reviewed and updated to find new defects.

**Testing is Context Dependent:** The effectiveness of testing techniques and methods depends on the context of the project, including factors such as the development methodology, technology, and domain.

**Absence-of-Errors Fallacy**: Finding and fixing defects does not guarantee that the software is defect-free or meets the customer’s need and expectations. Testing should focus on both finding defects and ensuring that the software meets requirements and user expectation.

1. Difference between QA v/s QC v/s Tester

**A:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Quality Assurance | Quality Control | Testing |
| 1 | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in same cases) requirements. | Activities which ensure the identification of bugs/error/defects in the software. |
| 2 | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| 3 | Process oriented activities. | Product oriented activities. | Product oriented activities. |
| 4 | Preventive activities | It is a corrective process. | It is a preventive process. |
| 5 | It is a subset of software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

1. Difference between Smoke and Sanity?

**A:**

|  |  |  |
| --- | --- | --- |
| No | Smoke | Sanity |
| 1 | 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. |
| 2 | The objective of this testing is to verify “stability” of the system in order to with more rigorous testing. | The objective of the testing is to verify the “rationality” of the system in order proceed to proceed with more rigorous testing. |
| 3 | This testing is performed by the developers or testers. | Sanity testing is usually performed by testers. |
| 4 | Smoke testing is usually documented or scripted. | Sanity testing is usually not documented and is unscripted. |
| 5 | Smoke testing is a subset of Acceptance testing. | Sanity testing is a subset of Regression testing. |
| 6 | Smoke testing exercise only the particular component of the entire system. | Sanity testing exercise the entire system from end to end. |
| 7 | Smoke testing is like General Health Check Up. | Sanity testing is like specialized health check-up. |

1. Difference between verification and Validation

**A:**

|  |  |  |
| --- | --- | --- |
| Criteria | Verification | Validation |
| Definition | The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for the phase. | The process of evaluating software during of at end of the development process to determine whether it satisfies business requirements. |
| Objective | To ensure that the product is being built according to the requirements and design specification. In other words, to ensure that work products meet their specified requirements. | To ensure that the product actually meets the user’s needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfils its intended use when placed in its intended environment. |
| Question | Are we building the product right? | Are we building the right product? |
| Evaluating Items | Plans, Requirements Specs, Design Specs, code, Test Cases | The actual product/software |
| Activities | * Reviews * Walkthroughs * Inspections | * Testing |

1. Explain types of Performance testing.

**A:** Performance Testing types are as follows:

* + Load testing
  + Stress testing
  + Volume testing
  + Spike testing
  + Soak testing
  + Configuration testing
  + Load Runner
  + Compatibility testing
  + Scalability testing
  + Reliability testing
  + Breakpoint testing

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

**A:**

**Error:** A mistake in coding is called error. A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

**Defect:** Error found by tester is called defect. The variation between the actual results and expected results is known as defect. Commonly refers to several troubles with the software products, with its external behaviour or with its internal features.

**Bug:** Defect accepted by development team is called bug/anomaly. A fault in a program which causes the program to perform in an unintended or unanticipated manner. Anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

**Failure:** When a defect reaches the end user is called failure. The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

1. Difference between Priority and Severity

**A:**

|  |  |  |
| --- | --- | --- |
| Features | Severity | Priority |
| Definition | Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| Purpose | Severity means how severe the defect is affecting the functionality. | Priority means how fast the defect has to be fixed. |
| Relation | Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| Categories | Severity is divided into 4 categories:   * Critical * Major * Medium * Low | Priority is divided into 3 categories:   * Low * Medium * High |
| Who decides defects? | The testing engineer decides the severity level of the defect. | The product manager decides the priorities of defects. |
| Value | Its value is objective. | Its value is subjective. |
| Value change | Its value doesn’t change from time to time. | Its value changes from time to time. |
| Association | It is associated with functionality or standards. | It is associated with scheduling. |
| Indication | It indicates the seriousness of the bug in the product functionality. | It indicates how soon the bug should be fixed. |
| Driving factor | It is driven by functionality | It is driven by business value. |
| Based on | It is based on the technical aspect of the product. | It is based on the customer’s requirements. |

1. What is Bug Life Cycle?

**A:** A bug life cycle (also known as defect life cycle) refers to the journey of a defect in which a defect/bug passes throughout its lifetime. It entails all stages ranging from the detection of a new defect/bug to the closing of the particular defect/bug by a tester. A defect/bug life cycle is the sequence of steps a bug or defect goes through from its identification to its resolution in software development. This life cycle standardizes the bug management process, ensuring teams can manage and resolve them more effectively.

1. Explain the difference between Functional testing and Non-Functional testing

**A:**

|  |  |
| --- | --- |
| FUNCTIONAL TESTING | NON-FUNCTIONAL TESTING |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. | Non-Functional testing checks the performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first | Non-functional testing should be performed after functional testing. |
| Manual testing or automation tools can be used for functional testing. | Using tools will be effective for this testing |
| Business requirements are the inputs to functional testing | Performance parameters like speed, scalability are inputs to non-functional testing. |
| Functional testing describes what the product does | Non-functional testing describes how good the product works |
| Easy to do manual testing | Tough to do manual testing |
| Types of Functional testing are   * Unit Testing * Smoke Testing * Sanity Testing * Integration Testing * White Box Testing * Black Box Testing * User Acceptance Testing * Regression Testing | Types of Non-Functional testing are   * Performance Testing * Load Testing * Volume Testing * Stress Testing * Security Testing * Installation Testing * Penetration Testing * Compatibility Testing * Migration Testing |

1. To create HLR & Test Case of

**A:**

(Instagram, Facebook) only first page

* HLR – Instagram, Facebook <https://github.com/KashyapTrivedi89/TestingProject/blob/main/HLR%20For%20Assignment%20Module2.xlsx>
* Instagram – Test Cases <https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestCase%20For%20Assignment%20Module%202.xlsx>
* Facebook – Test Cases <https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestCase%20For%20Assignment%20Module%202.xlsx>

1. What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

**A:**

|  |  |
| --- | --- |
| SDLC | STLC |
| SDLC is mainly related to software development. | STLC is mainly related to software testing |
| Besides development other phases like testing is also included. | It focuses only on testing the software. |
| SDLC involves total six phases or steps. | STLC involves only five phases or steps. |
| In SDLC, a greater number of members (developers) are required for the whole process. | In STLC, a smaller number of members (testers) are needed. |
| In SDLC, development team makes the plans and designs based on the requirements. | In STLC, testing team (Test Lead or Test Architect) makes the plans and designs. |
| Goal of SDLC is to complete successful development of software. | Goal of STLC is to complete successful testing of software. |
| It helps in developing good quality software. | It helps in making the software defects free. |
| SDLC phases are completed before the STLC phases. | STLC phases are performed after SDLC phases. |
| Post deployment of reusable software systems is the end result of SDLC, | Regression tests are run by QA team to check deployed maintenance code and maintain test cases and automated scripts. |
| Creation of reusable software systems is the end result of SDLC. | A tested software system is the end result of STLC. |

1. What is the difference between test scenarios, test cases, and test script?

**A:**

|  |  |  |
| --- | --- | --- |
| Test Scenarios | Test Cases | Test Script |
| Test scenario is the functionality that can be tested. | Test case is the set of actions to be executed to verify particular functionality. | It is set of instructions to test an app automatically using programming |
| It is derived from the business requirements and software requirements. | It is mostly derived from the Test scenarios. | It is mostly derived from the test cases. |
| Helps to test to end to end functionality in agile way. | Helps in exhaustive testing of a system | Helps to test specific thing repeatedly |
| It is mostly focused on what to test | It is focused on what to test and how to test | It is focused on the expected result |
| It includes an end to end functionality to be tested | It includes a test steps, data, expected result for testing etc. | It includes different commands to develop test script |
| Allows the quickly assessing the testing scope | Allows the detecting the errors and defects | Allows carrying out an automatic execution of test cases. |
| The main task is to check the full functionality of a software application | The main task is to verify compliance with the standards, guidelines, and customer requirements. | The main task is to verify that nothing is skipped and results are true as the desired testing plan. |

1. Explain what Test Plan is? What is the information that should be covered?

**A:** Test Plan is a detailed document that outlines the Objectives, Strategies, Timeline, Goals, Estimation, Deadlines, and Resources needed for the successful completion of a project. It provides a framework that is designed by QA managers to provide clarity about the necessary tests that you need to verify to ensure the proper functioning of the software. This includes defining test objectives, test approach, test tools, test environment, test schedules and team responsibilities and composition. However, before the right test approach and other planning details can be defined, a larger view of the organizational and project objectives must be defined first.

1. What are the different Methodologies in Agile Development Model?

**A:** Given below are some different methodologies in Agile Development Model:

**Crystal Agile Methodology:**

The Crystal Agile Software Development Methodology places a strong emphasis on fostering effective communication and collaboration among team members, as well as taking into account the human elements that are crucial for a successful development process.

**Dynamic Systems Development Method (DSDM):**

DSDM Methodology is tailored for projects with moderate to high uncertainty where requirements are prone to change frequently. Its clear-cut roles and responsibilities focus on delivering working software in short time frames.

**Featured-Driven Development (FDD):**

FDD approach is implemented by utilizing a series of techniques, like creating feature lists, conducting model evaluations, and implementing a design-by-feature method, to meet its goal. It is effective in ensuring that the end product is delivered on time and that it aligns with the requirements of the customer.

**Scrum:**

Scrum methodology serves as a framework for tackling complex projects and ensuring their successful completion. It is led by a Scrum Master, who oversees the process, and a Product Owner, who establishes the priorities. The Development Team, accountable for delivering the software, is another key player.

**Extreme Programming (XP):**

Extreme Programming uses specific practices like pair programming, continuous integration, and test-driven development to achieve these goals. It is ideal for projects that have high levels of uncertainty and require frequent changes, as it allow for quick adaption to new requirements and feedback.

**Lean Development:**

Lean Development is rooted in the principles of lean manufacturing and aims to streamline the process by identifying and removing unnecessary steps and activities.

**Unified Process:**

Unified Process is a methodology that can be tailored to the specific needs to any given project. It combines elements of both Waterfall and Agile methodologies, allowing for an iterative and incremental approach to development. It is characterized by a series of iterations, each of which results in a working product increment, allowing for continuous improvement and the delivery of value to the customer.

1. Explain the difference between Authorization and Authentication in Web testing.

**A:**

|  |  |
| --- | --- |
| Authentication | Authorization |
| In the [authentication](https://www.geeksforgeeks.org/authentication-in-computer-network/) process, the identity of users is checked for providing the access to the system. | While in [authorization](https://www.geeksforgeeks.org/what-is-aaa-authentication-authorization-and-accounting/) process, a the person’s or user’s authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user’s login details. | While it needs the user’s privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines w**hat permission does the user have?** |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| Popular Authentication Techniques-   * Password-Based Authentication * Password less Authentication * 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) * Social authentication | Popular Authorization Techniques-   * Role-Based Access Controls (RBAC) * [JSON web token (JWT) Authorization](https://www.geeksforgeeks.org/json-web-token-jwt/) * SAML Authorization * OpenID Authorization * OAuth 2.0 Authorization |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. |

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

**A:** Common problems faced in Web Testing are Security Vulnerabilities, Compatibility Issues, Performance Issues and Usability Concerns. These common problems are further explained below:

**Security vulnerabilities:** Web applications are often targeting for malicious attacks, so security testing is crucial to identify and address vulnerabilities.

**Compatibility issues:** Web applications need to work across various browsers, devices, and operating systems. Compatibility testing ensures that the application functions correctly and displays properly across different environments.

**Performance issues:** Poor performance, slow loading time, and high server response time can negatively impact user experience and lead to decreased engagement and customer dissatisfaction. Performance testing helps identify and address bottlenecks, scalability issues, and other performance-related problems.

**Usability concerns:** Usability testing evaluates the user interface, navigation, and overall user experience of the web application to ensure it is intuitive, user-friendly, and meets the needs of its target audience.

**Q.** To create HLR & Testcase of Web Based (WhatsAppweb, Instagram) 1. WhatsAppWeb: <https://web.whatsapp.com/>

**A:**

HLR – WhatsAppWeb<https://github.com/KashyapTrivedi89/TestingProject/blob/main/HLR%20For%20Assignment%20Module2.xlsx>

TEST CASE – WhatsAppWeb<https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestCase%20For%20Assignment%20Module%202.xlsx>

TEST CASE – Instagram

<https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestCase%20For%20Assignment%20Module%202.xlsx>

**Q.** To create HLR and Testcase on this Link. <https://artoftesting.com/>

**A:**

HLR:

<https://github.com/KashyapTrivedi89/TestingProject/blob/main/HLR%20For%20Assignment%20Module2.xlsx>

TEST CASE:

<https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestCase%20For%20Assignment%20Module%202.xlsx>

**Q.** Write test scenario’s

<https://github.com/KashyapTrivedi89/TestingProject/blob/main/TestScenario%20For%20Assignment%20Module%202.docx>

**Q.** When to used Usability Testing?

**A:** Usability Testing is used before any Design Decisions are made. This process can be as simple as watching someone interact with the current version of the product. If is building something from scratch, one can test how their audience uses a competing product. Usability Testing is conducted before putting any design resources to work. It identifies specific areas where testing and validation can enhance the concept.

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

**A:** Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of barstool bar, menu bar, dialog boxes and windows etc. It is important to verify all the images used throughout the application are displayed correctly and not blurry or cut out in any device.