Module–2 (Manual Testing)

1) What is Exploratory Testing?

Exploratory testing is a style of software testing that emphasizes the personal Freedom and responsibility of the individual tester to continually optimize the quality of her work by testing test related learning, test design, test execution, And test result.
When you do exploratory testing just focus on that do not do just click click testing.

When to do
This testing is suited for specific testing scenarios, such as when someone needs to learn about a product or application quickly and provide rapid feedback.

Benefit
- Less preparation is needed
- Important bugs are found quickly
- The approach tends to be more intellectually stimulating than execution of scripted tests.

2) What is traceability matrix?

A traceability matrix is a document used in software development and project management to ensure that all requirements are linked to their corresponding deliverables, such as design document, test cases, and code modules.
This helps verify that all requirements are addressed and fulfilled though the project life cycle.

Requirement id	Design	Code module	Test case
R1	DD1	CM1	TC1
R2	DD2	CM2	TC2
R3	DD3	CM3	TC3

The traceability matrix helps project managers, developers, and tester ensure completeness, traceability, and accountability throughout the project lifecycle.

3) What is Boundary value testing?

Boundary value testing is a type of software testing technique that focuses on the values at the edges of input domains.
In the boundary value testing you must test less than one of minimum value and more than one to maximum value
Like if you want to test 1 to 100 than you must check only 0, 1,2 and 100,101,102.
In this test 0 and 102 should be fail than this test is pass.
If you test with this type so you can save time.
It is relatively straightforward to design and implement.

4) What is Equivalence partitioning testing?

Equivalence partitioning is a software testing technique used to reduce the number of test cases by dividing the input data of a software unit into partitions of equivalent data from which test cases can be derived. In this approach, the idea is that if one test case in partition passes, the others in the same partition are assumed to pass as well, because they are functionally equivalent in terms of how the system processes them.
Like if you want to test 1 to 1000 test case.

Equivalence	Representative	Result
<1	0	Fail
1-200	105	Pass
201-400	210	Pass
401-600	413	Pass
601-800	624	Pass
801-1000	933	Pass
>1000	1002	Fail

5) What is Integration testing?

Integration testing is a part of STLC.
Testing multiple software modules or components together.
Unite are combined, now test.
Like Login, Signup, add to card payment modules to combine that and you get Complete
E-commerce.
<u>There are 2 types of integration testing</u>
1) component integration testing
2) System integration testing

6) What determines the level of risk?

— Complexity of the software

More modules and interfaces increase the risk due to the higher potential for integration issues.

Complex algorithms are harder to test and more prone to defects.

External decencies on third party libraries, APIs, or systems can introduce additional risk.

—Project size and scope

Larger projects typically have more moving parts, increasing the chance of defects and integration issues.

Frequent changes in project scope can introduce new risks and make it harder to manage existing risk.

—Technology stack

Using new or less understood technologies can increase risk due to the learning curve and potential for unforeseen issues.

Integrating with or updating legacy systems can be risky due to outdated technology and lack of documentation.

—Team experience and skills

The experience and expertise of the development and testing teams play a critical role in mitigating risks.

High turnover rates can lead to a loss of knowledge and increase the risk of defects.

—Development process

Well-defined and mature development processes reduce risk by ensuring consistency and thoroughness.

Comprehensive testing, including unit, integration, system, and acceptance testing, helps identify and mitigate risk early.

—Requirements quality

Clear well-defined, and stable requirements reduce the risk of misunderstandings and scope changes.

Frequently changing requirements can introduce new risks and complicate project management.

7) What is Alpha testing?

Alpha testing is a type of software testing performed to identify bugs before releasing the product to real users or public. It is conducted at the end of development phase and is typically the first phase of testing where the software is tested.

Alpha testing is always performed by the developers at the software development site.
It is always performed in virtual environment.

8) What is beta testing?

Beta testing is a type of software testing that involves releasing the software to a limited number of external users outside the development team or company.
This phase comes after alpha testing and is typically the final testing stage before the software's official release.
The main purpose of beta testing is to indicate the software in a real-world environment and gather feedback from actual users to ensure the product meets user needs and function correctly in diverse scenarios.

9) What is component testing?

Component testing, also known as unit testing or module testing, is a type of software testing where individual components or units of the software are tested in isolation.
The goal of component testing is to validate that each component function correctly on its own, ensuring that it behaves as expected and meets its design and requirements specification.
Unite testing is the first level of testing and is performed prior to integration testing.
Unite testing is performed by using the white box testing method.
Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing.

10) What is functional system testing?

When you check user behavior that's call functional testing.
Functional system testing is a type of software testing that evaluates the behavior of a complete and integrated software system against its specified functional requirements. Unlike component testing, which focuses on individual components or unite, functional system testing examines the system to ensure that it meets user expectations and performs its intended functions correctly.

Functional system testing, also known as system testing, is a type of software testing that evaluates the system to ensure that it meets specified functional requirements. Unlike component testing which focuses on testing individual

components in isolation, system testing verifies the behavior of the entire system in an integrated environment.

11) What is Non-Functional Testing?

Non-fictional testing refers to the process of evaluating, assessing, or examining real-world scenarios, information, or actual phenomena. This type of testing is used in various filed, including education, software development, research, and quality assurance.
Here are some examples of non-fictional testing in different contexts:
—Educational testing
Exams like the SAT, ACT Or state assessments that measure students' knowledge and skill based on information and real- world scenarios.
—Software testing
Ensuring that software functions as intended with real world data and use cases.
Assessing how real users interact with the software to identify any usability issues.
—Quality assurance
Evaluating product in real-world condition to ensure they meet safety and performance standards.
Testing services to ensure they perform as expected under actual operating condition.
—Research and development
Testing new medications or treatments on real patients to evaluate their effectiveness and safety.
Conducting experiments in natural setting to observe phenomena in their actual context.
—Engineering
Applying stress to materials or structures to determine their durability and performance under real-world conditions.

12) What is GUI Testing?

Graphic user interface testing is the process of ensuring proper functionality of graphical user interface for a specific application.
GUI testing is a process of testing a software application's user interface to ensure it meets its specifications and provides a good user experience.

This type of testing focuses on how the application interacts with the user through graphical elements such as button, text, filed, icons and other interactive components.
Selenium is an open-source GUI testing tool that support web applications.
Types of GUI testing
1) manual testing
2) Automated testing

13) What is Ad-hoc testing?

Ad-hoc testing: unstirred testing where testers randomly check the GUI components without predefined test cases.
Ad-hoc testing is a software testing technique performed without any specific test plan or predefined set of steps.
Example: A tester testing a massaging app might send random messages to themselves and others without any set sequence or order.

14) What is load testing?

Load testing examines how the system behaves during normal and high loads and determines if a system, piece of software, or computing device can handle high loads given a high demand of end-users SDETs test the system for sufficient capacity to support expected traffic.
Load testing is a type of performance testing that simulates a real- world load on any software, application, or website.
Without it, your application could fail miserably in real world conditions.
It measures the speed or capacity of the system or component through transaction response time.
You can use tools like Web load, Load view, and Load runner.

15) What is stress Testing?

Stress testing is a type of performance testing used to evaluate how a system behaves under extreme conditions, such as high user load, heavy data processing, or other stressful scenarios.
The goal is to determine the system's robustness, stability, and reliability under this condition and to identify any breaking points or performance bottlenecks.
Types of stress testing
1) load testing
2) Spike testing

3) Configuration testing
4) Endurance testing (soak testing)

16) What is white box testing and list the types of white box testing?

White box testing is a form of application testing that provides the tester with complete knowledge of the application being tested, including access to source code and design documents.
White box testing also known as clear box testing, glass box testing, transparent box testing and structural testing is a method of testing software that involves looking at the internal structure, design, and implementation of the software being tested.
This type of testing is based on the knowledge of the internal logic of an application's code, which differentiated it from black box testing, where the tester only interacts with the software's user interface without any knowledge of the underlying code.
Types of white box testing
Unite testing
Integration testing

17) What is black box testing? What are the different black box testing techniques?

Black box testing is a system which can be viewed in terms of its inputs and outputs without any knowledge of its internal software parts.
Example: The smart phone app link to a gps device that measures and records vehicle speed, location, distance traveled, driving frequency, and time of day the car is in motion.

Techniques of black box testing
1) equivalence partitioning
2) Boundary value analysis
3) Decision tables
4) State transition testing
5) Use-case testing
6) Other black box testing
   Syntax or pattern testing

18) Mention what are the categories of defects?

Defects in software can be categorized in various ways depending on their nature, impact, and origin.
Here are some common categories of defects:
Functional defects:
When the software does not behave as expected according to the requirement or specifications.
Mistakes in the logic or algorithms that incorrect outcomes.
Performance defects:
The software does not perform operations within acceptable time limits.
Problem with the volume of data the software can process in a given time.
Usability defects:
Problem with layout, design, or functionality of the user interface.
Issues with the flow or accessibility of different parts of the software.
Security defects:
Weaknesses that could be exploited by attackers to gain unauthorizes access or perform malicious actions.
Problem with verifying the identity of users.
Compatibility defects:
Problem when the software does not run correctly on different operating systems or environments.
Issues that arise when the software behave differently across various web browsers.
Maintain defects:
Code that is difficult to understand, modify, or extend.
Inadequate or incorrect documentation that hampers the maintenance process.

19) Mention what big bang testing is?

Big bang testing is a software testing approach where all or most of the developed modules are combined and tested simultaneously as a complete system.
Unlike incremental testing approaches, where integration testing is performed step by step, big bang testing waits until the last possible moment to integrate all modules together.
This method is typically used in smaller systems or when there is a lack of time for proper testing due to tight schedules.
When to use big bang testing:

Small systems: suitable for small, simple projects where the risk of integration issues is low.
Tight schedules: used when there is not enough time to perform incremental integration and testing.
Proof of concept: when the goal is to quickly demonstrate the feasibility of the entire system.

20) What is the purpose of exit criteria?

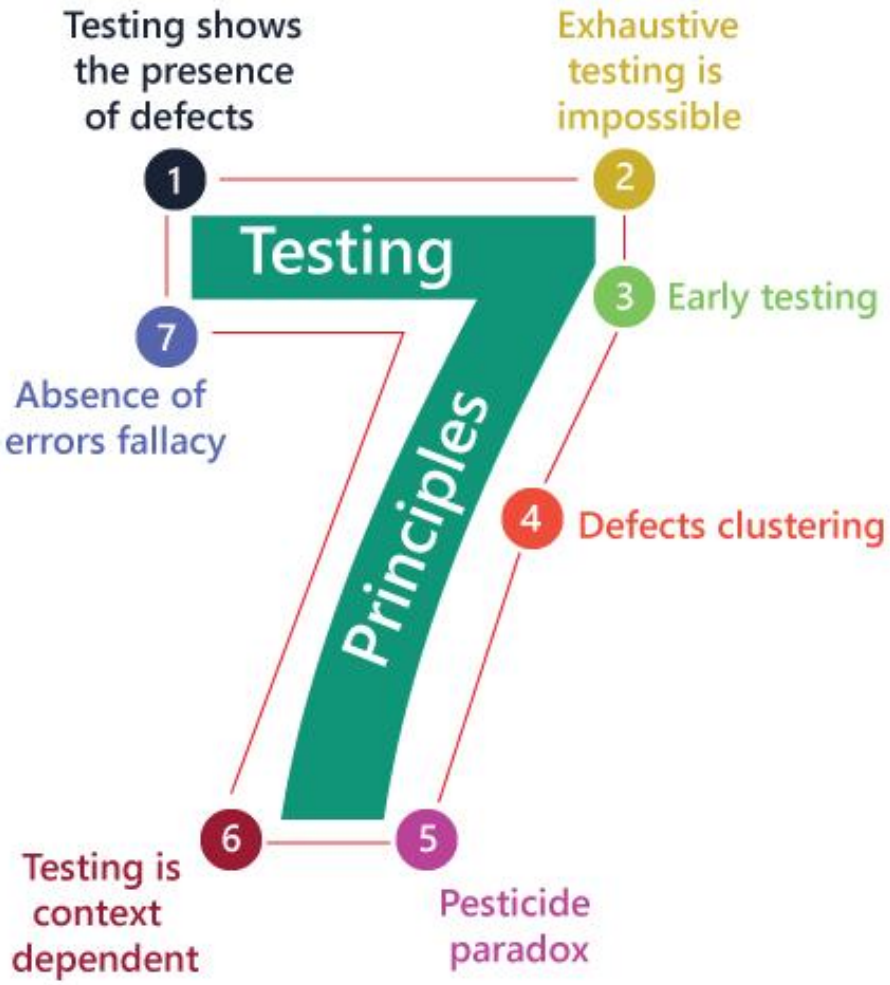
Exit criteria are essential elements in project management, software development, and various other filed to determine when a phase, milestone, or project can be considered complete.
Here are the primary purposes of exit criteria:

1) Ensures that the work completed meets predefined quality standards. Exit criteria often include requirements that must be met for a product to be considered of acceptable quality.
2) Help in identifying and mitigating risk by ensuring that all critical tasks and checks are completed before moving on to the next phase.
3) When time and budget are done.
4) When manager or boss say stop.
5) When you test again and again, and you find very small bugs.
6) When you are sure the application, a software is customer friendly.
7) When you confident about software is done according to customer desire.
8) All defects have been fixed.

21) When should "Regression Testing" be performed?

1) regression testing should be performed after any code modifications, bug fixes, or enhancements are made to the software.
2) When the system is stable and the system or the environment changes.
3) It's essential to conduct regression testing before releasing a new version or update of the software to ensure that no new defects have been introduced.
4) Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation.
5) Even if no immediate changes have been made, regression testing, often scheduled as part of regular release cycles, helps maintain software quality and stability.
6) When new feature is added to the software.
7) Defect fixing.

22) What is 7 key principles? Explain in detail?



Caption

1. Testing shows presence of Defects

Testing can show that defects are present' but it cannot prove that there are no defect. In other word, testing reduces that probability of undiscovered defects remaining in the software but is not a guarantee of a defect- free product.

2. Exhaustive Testing is Impossible!

It is impossible to test all possible inputs and scenarios. Instead, risk analysis and prioritization should focus on the most important and impactful tests.

3. Early Testing

Testing activities should start as early as possible in the software development lifecycle. The earlier defects are found, the cheaper they are to fix. This principle is often referred to as shift left testing.

4. Defect Clustering

A small number of modules usually contain the most defects. This is often referred to as the Pareto principle or the 80/20 rule, which suggests that 80% of the problem are found in 20% of the modules.

5. The Pesticide Paradox

If the same tests are repeated, eventually these tests will no longer find any new defects. To overcome this, the tests need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software.

6. Testing is Context Dependent

The approach to testing will depend on the context of the application. Different types of application require different testing approaches and methodologies.

Ex. Web application, mobile apps, desktop software.

7. Absence of Errors Fallacy

Just because a system is tested thoroughly, and no defects are found does not mean the system is usable. It must also meet the user's needs and requirements. Testing should ensure that the system not only functions correctly but also deliver value to the users.

23) Difference between QA v/s QC v/s Tester

QA	QC	Tester
QA focuses on improving the processes used to	QC focuses on identifying defects in the final product.	Testers focus on executing tests on the product to identify
QA activities include process definition and implementation, audits, training and process improvement.	QC activities include inspection, testing and reviewing the product to ensure it meets the specified requirements.	Testers write and execute test cases, report defects, and verify fixes.

QA is process oriented. It involves the entire software development life cycle, ensuring that every stage meets the defined standards.	QC is product - oriented. It involves verifying that the final product is defect-free and meets the quality standards.	Testers are involved in both QC and A activities, though primarily in QC.
QA ensures that the development process is followed correctly and that it results in a quality product.	QC ensures that the product is of the required quality and that any defects are identified and corrected before the product is	Testers are responsible for finding and documenting defects in the software.
Implementing a coding standards guideline, performing regular process audits, and conducting training sessions for the	Conducting functional testing, performance testing, and cod reviews.	Creating and executing test cases based on requirements, using automated testing tools to perform

24) Difference between Smoke and Sanity?

Smoke	Sanity
Smoke testing is a preliminary testing process to check whether the critical functionalities of a software build are working correctly.	It aims to verify that the changes or fixes did not introduce new issues.
Broad and shallow. Smoke testing covers all the major functionalities but does not delve deeply into any	Narrow and deep. Sanity testing focuses on a limited set of functionalities or areas affected by
Conducted on initial builds or after major changes to ensure basic functionality works before proceeding with more detailed testing.	Conducted after receiving a stable build that has passes smoke testing or after minor changes and bug fixes to confirm that the specific issues have been resolved.

Ensures that the build is not broken and that the application is ready for more rigorous testing.	Ensures that the particular functionality or bug fixes work as intended without affecting other
Verifying that the application launches, logging in , and basic navigation between major modules.	Verifying that a bug fix in the login module did not break the login functionality or checking a new feature to ensure it works correctly.

25) Difference between verification and Validation

Verification	Validation
Verification is the process of checking whether the product meet the specified requirements at various stages of development.	Validation is the process of evaluating the final product to ensure it meets the business needs and requirements.
Ensure that the product's design and implementation conform to the requirements and specifications.	Ensure that the product fulfills the intended use and user needs.
Detect errors in the early stages of the development process.	Identify any discrepancies between the final product and user
Methods is reviews, inspections, walkthroughs, static analysis.	Method is testing, simulation, prototyping, user feedback and
Conducting static analysis to find potential error in the code before	Performing system tests to verify that all components work together

26) Explain types of Performance testing.

Performance testing is a type of non-functional testing aimed at determining how a system performs in terms of responsiveness and stability under a particular workload.
Types if performance testing:

Load testing
Evaluates the system's performance under expected user loads.
Ensure that the system can handle expected usage levels without significant degradation in performance.
Like, simulating 500 concurrent users on an e-commerce website to ensure it handles the traffic smoothly.

Stress testing
Tests the system beyond its normal operational capacity to see how it behaves under extreme conditions.
Identify the breaking point of the system and how it recovers from failures.
Like, increasing the number of concurrent users to 1000 to see if the e-commerce website crashes and how it recovers.

Endurance testing
Evaluates the system's performance over an extended period.
Ensure that the system can handle sustained use without degradation.
Like, running a system for 24 hours with a constant load to check for memory leaks or other long-term issues.

Spike testing
Tests the system's response to sudden, significant increases in load.
Assess the system's ability to handle unexpected and sharp spikes in user activity.
Like, simulating a sudden increase from 100 to 1000 users in a few seconds to see how system copes with the sudden surge.

Volume testing
Evaluates how well a system performs when subjected to a large amount of data.
Ensure that the system can handle a significant volume of data without crashing and slowdown.

Scalability testing
Evaluates how well the system scales with increasing loads.
Ensure that the system can handle increased loads by adding resources.

Like, testing how the addition of more servers affects the performance of a web application.

27) What is Error, Defect, Bug and failure?

Error
A mistake in coding is called error. An error is a human action that produces an incorrect result. It's a mistake made by a developer while writing code or designing a system. Errors can lead to defects in the software.

Defect
Error found by tester is called defect. A defect, also known as a fault or issue is an anomaly in the software that causes it to behave incorrectly, produce incorrect results, or fail to meet specified requirements.
When a defective is present in the software, it means that the software does not conform to its requirements.
Bug
Defect accept by development team then it is called bug. A bug is a specific instance of a defect in the software. The term bug is often used colloquially to refer to any kind of software problem, though technically it specifically denotes a defect that results in a failure when the program is executed.

Failure
Software build does not meet the customer requirements then it is called failure. A failure occurs when the software does not perform its intended function and produces incorrect result or behaves unexpectedly.

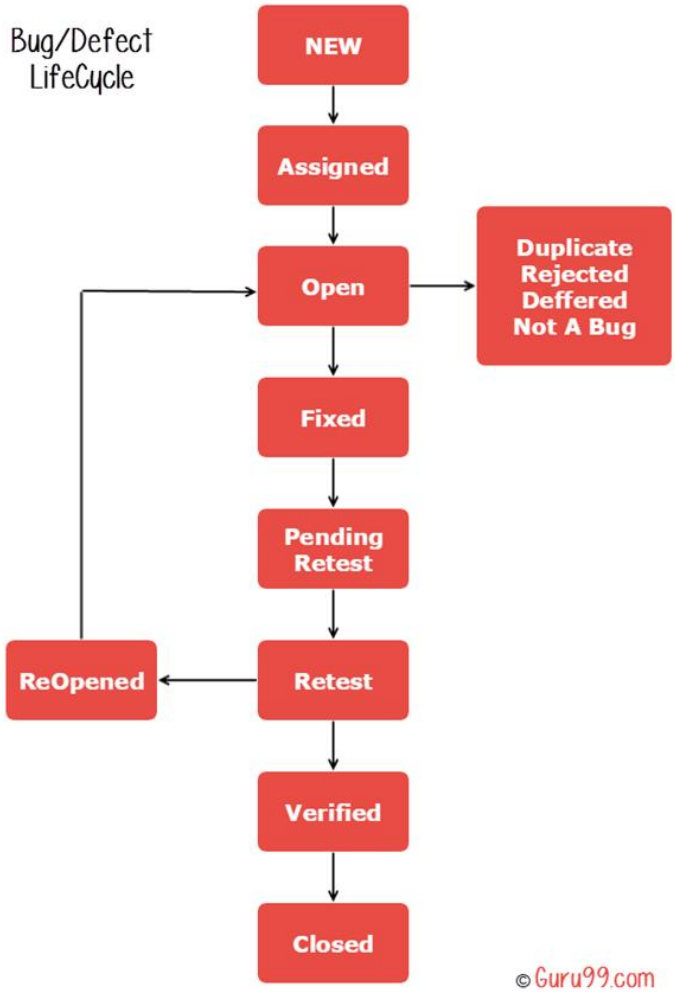
28) Difference between Priority and Severity

Priority	Severity
Priority refers to the order in which defects or bugs should be addressed or fixed by developers.	Severity refers to the impact or bug on the functionality of the software.
Does fixing this bug depend on other fixes or components that need to be completed first?	The bug is minor and only affects the appearance of the software without impacting functionality.
It determines the urgency or importance of fixing the issue relative to other issues.	It measures how severe or critical the bug is in terms of its impact on the user experience or the system's

Priority is relative and business	severity is absolute and customer
The bug can be addressed later, as it has minimal impact on functionality or can be easily worked around.	The bug has a relatively minor impact on functionality or can be easily worked around.

29) What is Bug Life Cycle?

New
When a bug is first identified and reported, it is given a 'NEW' status.
At this stage, the bug is recorded in the bug tracking system with details about its environment and steps to reproduce.



Assigned
Once reviewed by the project manager or team lead, the bug is assigned to a developer or team member responsible for fixing it.
The status changes to 'Assigned'.

Open
The assigned developer begins working on the bug.
The status changes to 'open' to indicate that the bug is actively being investigated and fixed.

Fixed
After the developer has made the necessary code changes to resolve the issue, the bug status is updated to 'fixed.'
At this point, the bug is considered resolved from developers' perspective, but it still needs to be verified by testing team.

Pending retest
The bug is fix but test is pending that's called a pending retest.

Retest
The bug is passed to the testing team for verification.
The status changes to ‘test’ or ‘in test’ to indicate the fix is being tested to ensure that the issue is resolved, and no new issues have been introduced
Verified
If the testing team confirms that the bug has been fixed and the software behaves as expected, the status is updated to ‘verified.’
This means the fix has been validated and the bug is considered resolved.

Closed
Once the bug has been verified, the status is changed to closed.
The indicates that the bug is fully resolved, and no further action is required.

Duplicate rejected deferred note a bug
Sometime, a bug may not be critical or may not be feasible to fix immediately due to time constraints, resource limitations, or other priorities.

30) Explain the difference between Functional testing and Nonfunctional testing

Functional testing	Non-Functional testing
Verifies that the software functions according to specified requirements and performs its intended tasks	Evaluates the non-functional aspects of the software, such as performance, usability, reliability, etc.
Validates the actions and operations of the system. Check what the system dose.	Checks the quality attributes of the system. Measures how the system
Types: unite testing, integration testing, system testing, acceptance testing.	Types: performance testing, usability testing, security testing, compatibility testing, reliability
Verifying CRUD operation in a database.	Checking if the software is usable across different web browsers and operating systems.
Based on requirement and specification.	Based on quality attributes and benchmarks.
Tools: selenium, JUnit, test NG, QTP.	Tools: JMeter, LoadRunner,

31) To create HLR & Test-case of

1)(Instagram, Facebook) only first page

HLR	file:///Users/komalshah/Documents/tops career komal shah/EXEL /HLR-Instagram-first-page.xlsx
Test case	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /INSTAGRAM FRIST PAGE TEST CASE .xlsx

2) Facebook Login Page: https://www.facebook.com/

HLR	file:///Users/komalshah/Documents/tops career komal shah/EXEL /HLR-Facebook-first-page.xlsx
Test case	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /FACEBOOK FRIST PAGE

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

STLC	SDLC
STLC is sequence of specific activities conducted during the testing process to ensure software	SDLC is a process used for planning, deploying an information system.

Phases: requirement analysis, test planning, test case development, test environment set up, test execution, test cycle closure.	Phases: requirement gathering, planning, design, coding, testing, maintenance.
To identify and fix defects in the software.	To deliver high-quality software that meets or exceeds customer
Specification to the testing phases within the overall software development process.	Overall software development, from conception to deployment and maintenance.
Focuses solely on testing phase within the SDLC.	Encompasses the entire lifecycle of software development.

33) What is the difference between test scenarios, test cases, and test script?

Test case	Test script	Test scenarios
A test case is a detailed document that outlines a specific condition or set of conditions under which a tester will determine whether a software application or system is working	A test script is a set of instructions written in a programming or scripting language that is executed by automated testing tools.	A test scenario is a high-level description of a functionality or feature to be tested.
To provide a comprehensive guide for testing specific aspects of the software ensuring consistency	To automate repetitive testing tasks, increase, efficiency and ensure consistency in test execution.	To outline the various way a user might interact with the system, ensuring that all functional paths and
Highly detailed and specific, focusing on individual conditions and action.	Detailed, either in code or instructions focusing on the steps to be executed.	High level and conceptual, focusing on end-to-end functionality or user

To verify specific functionalities or conditions of the	To automate the execution of test cases or perform detailed	To ensure comprehensive coverage of use cases
Used by tester to execute specific test manually.	Used by automated testing tools or testers to execute tests.	Used in the planning phase to ensure all functionalities and paths are covered by

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

A test plan is a document that outline the strategy, resources, scope of intended test activities. A comprehensive test plan covers various aspects to ensure that all critical areas are addressed.

Here's a detailed outline of the information typically include in a test plan.
Introduction: define the objectives of the test plan and the intended outcomes.
Describe the extent and boundaries of the testing activities.
Provide explanations for specific terms and abbreviations used in the document.
Test objectives: clearly state the goals of the testing efforts, such as verifying functionality, ensuring performance, or validating security.
Test scope: detail the features, functionalities, and systems that will be tested.
Specify what will not be tested to manage exceptions.
Test items: List all the components, modules, or systems that are subject to testing, including software, hardware, and interfaces.
Assumptions and Dependencies: Document any assumptions made that could impact testing.
Highlight dependencies such as test environments, third-party services, or specific hardware that might affect testing schedules or outcomes.
Test strategy: Define the various levels of testing (e.g., unit**, i**ntegration, system, acceptance)

Describe the types of testing to be conducted (e.g., functional, regression, performance, usability).
Explain the methodologies and approaches used to design the tests (e.g., boundary value analysis, equivalence partitioning).
Test environment: Provide detailed specifications of the environments required for testing, including hardware, software, network configurations, and tools.
Test schedule: Outline the timeline for testing activities, including start and end dates, key milestones, and deadlines.
Test resources: Identify the team members involved in the testing process, including their roles and responsibilities.
Identify the team members involved in the testing process, including their roles and responsibilities
Test deliverables: Enumerate the documents and artifacts that will be produced during the testing process, such as: test cases, test script, test data, defect reports, test summary reports.
Entry and exit criteria: define the conditions that must be met before testing can begin.
Specify the conditions that must be met before testing can be considered complete.

35) Difference between priority and severity

Priority	Severity
Priority is usually set by the project managers or product owners and reflects the business impact.	Severity is typically determined by the testing team and reflects the technical impact of the defect.
The defect should be fixed as soon as possible because it has a high impact on the business or project.	The defect causes complete failure of the software or a major part of it, preventing any further testing or use
e.g. defect that affect a major release or critical customer	E.g. application crashes, critical, data loss
Measures the urgency of fixing the defect from a business or project	Measures the impact on the system or application.
Priority is usually determined by project managers, product owners, or stakeholders based on business	Severity is typically determined by the testing team based on the technical assessment.

36) What are the different Methodologies in Agile Development Model?

Agile methodologies	Agile development model
Agile methodologies are specific approaches or practices that implement the principles and values of the agile development model.	The agile development model refers to the overarching framework and philosophy that emphasize iterative development, flexibility, collaboration, and customer
These methodologies provide structured frameworks and processes to guide teams in	It is a high-level concept that encapsulates the principles and value outlined in the agile manifesto.
Agile methodologies prescribe specific ways of working, often with tools and ceremonies.	Agile development model can be adapted and customized to fit various contexts and project needs.
Agile methodologies offer concrete methods and processes to apply agile principles in practice.	Agile development model provides the philosophical foundation and overall approach to agile
Agile methodologies specific implementations of the agile model with defined practices, roles, and	Agile development model broad framework encompassing principles, values, and general practices.

38) To create HLR & Test-case of Web Based (WhatsApp web)

Web: https://web.whatsapp.com/

HLR (WhatsApp web)	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /HLR (WHAT'SAPP
Test case (WhatsApp web)	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /whatsapp web TEST

39) To create HLR and Test-case on this Link. https://artoftesting.com/

HLR	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /HLR (ART OF
Test case	file:///Users/komalshah/Documents/tops career komal shah/THE ART OF TESTING TESTCASE .xlsx

40) Write a scenario of only WhatsApp chat messages

Scenario of WhatsApp chat	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /Whatsapp Chet

41) Write a Scenario of Pen

Scenario of pen	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /pen_scenario .xlsx

42) Write a Scenario of Pen Stand

Scenario of pen stand	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /pen stand_scenario .xlsx

43) Write a Scenario of Door

44) Write a Scenario of ATM

Scenario of ATM	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /Atm_scenario .xlsx

45) When to used Usability Testing?

Usability testing should be used in the following scenarios

1) Early design: To gather user feedback on prototypes or wireframes before development.
2) Pre-release: Before launching a new product or feature to ensure it met user expectation.
3) Post-release: To identify usability issues in live products and gather data for future improvements.
4) Major updates: When significant changes are made to the UI or UX to ensure new designs enhance user experience.
5) Comparative analysis: To compare different design alternatives and select the most user-friendly option.
6) Compliance testing: To ensure the product meets accessibility and usability standards.
7) User-centered design process thought the development cycle to continuously validate design decisions with real users.

46) What is the procedure for GUI Testing?

GUI testing ensures that the user interface meets the specified requirements, in user-friendly, and is free of defects.

Here 's a step-by-step procedure for GUI testing.

1) Preparation: Requirement analysis — understand the UI requirements and design specification. review design document. Ensure the necessary software and hardware are available and configured.
2) Test planning: Test case design —create detailed test cases that cover all UI elements. This includes button, menus, toolbar and other graphical element. prepare the data needed for testing. Identify different user scenarios to be tested, including typical user paths, and error conditions.
3) Test execution: Manual testing— check layout, functionality testing, input validation, navigation testing, usability testing. Automation testing—select automation tool (selenium, QTP, test complete), write test script, run automated tests.

4) Defect reporting: Log defect — document any issues found during testing in a defect tracking system. Prioritize defects — Assign priorities to defect based on their impact on the user experience and application functionality.
5) Regression testing: Retest fixed defects — once defects are fixed, retest the affected areas to ensure the issues are resolved. Regression suite execution —Run a regression test suite to ensure new changes have not introduced new defects in previously working functionality.
6) Performance testing: Load testing — test how the UI performance under various load conditions ensuring it remains responsive. Stress testing — Assess the UI's behavior under extreme condition, such as heavy user traffic or low system resources.
7) Compliance testing: Accessibility testing — verify that the UI complies with accessibility standards to ensure it is usable by people with disabilities. Cross-Browser testing —Ensure the ui work correctly across different web browsers and versions. Cross-Platform testing —Test the application on various operating system and devices to ensure consistent behavior.
8) Final review and reporting: The summary report — prepare a comprehensive report summarizing the testing activities, findings, and overall quality of the GUI. Stakeholder review — share the report with stakeholders for review and approval.

47) Write a scenario of Microwave Owen

Scenario of microwave Owen	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /MICROWAVE

48) Write a scenario of Coffee vending Machine

Scenario of coffee vending machine	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /COFFEE VENDING MACHINE_scenario .xlsx

49) Write a scenario of chair

Scenario of chair	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /CHAIR_Scenarios.xlsx

50)Online shopping to buy product (Flipkart)

Scenario of online shopping to buy product	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /Online shopping

51) Write a Scenario of Wristwatch

Scenario of wristwatch	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /wrist

52) Write a Scenario of Lift (Elevator)

Scenario of lift (elevator)	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /ELEVATOR_scenario.xlsx

53) Write a Scenario of WhatsApp Group (generate group)

Scenario of WhatsApp group	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /whatsapp

54) Write a Scenario of WhatsApp payment

Scenario of WhatsApp payments	file:///Users/komalshah/Documents/tops career komal shah/TOPS PROJECT /