SOFTWARE TESTING ASSIGNMENT MODULE 2
Manual testing

1. What is Exploratory Testing?
   - ➔ Exploratory testing is a concurrent process where Test design, execution and logging happen simultaneously. Testing is often not recorded makes use of experience, heuristics and test patterns. Testing is based on a test charter that may include Scope of the testing (in and out). The focus of exploratory testing is more on testing as a “thinking” activity. A brief description of how tests will be performed Expected problems Is carried out in time boxed intervals.

2. What is traceability matrix?
   - ➔ A traceability matrix is **a** document that details the technical requirements for a given test scenario and its current state. It helps the testing team understand the level of testing that is done for a given product. The traceability process itself is used to review the test cases that were defined for any requirement.

3. What is Boundary value testing?
   - ➔ Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges. Boundary value analysis is a method which refines equivalence partitioning. Boundary value analysis generates test cases that highlight errors better than equivalence partitioning. The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes. At those points when input values change from valid to invalid errors are most likely to occur. Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design.

4. What is Equivalence partitioning testing?
   - ➔ Aim is to treat groups of inputs as equivalent and to select one representative input to test them all EP can be used for all Levels of Testing.
   - ➔ Equivalence partitioning is the process of defining the optimum number of tests by: Reviewing documents such as the Functional Design Specification and Detailed Design Specification, and identifying each input condition within a function.
   - ➔ Selecting input data that is representative of all other data that would likely invoke the same process for that particular condition.

5.What is Integration testing?

- ➔ Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
- ➔ The purpose of this level of testing is to expose faults in the interaction between integrated units.
- ➔ Test drivers and test stubs are used to assist in Integration Testing.
- ➔ Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- ➔ Integration testing is done by a specific integration tester or test team. Components may be code modules, operating systems, hardware and even complete systems.

6. What determines the level of risk?
   - ➔ The probability that an adverse event will occur.
   - ➔ The likelihood of an adverse event and the impact of the event.
   - ➔ The cost of dealing with an adverse event if it occurs.
   - ➔ The amount of testing planned before release of a system.

7. What is Alpha testing?
   - ➔ Alpha testing is always performed by the developers at the software development site.
   - ➔ Sometimes it is also performed by Independent Testing Team.
   - ➔ Alpha Testing is not open to the market and public It is conducted for the software application and project.
   - ➔ It is always performed in Virtual Environment. It is always performed within the organization.
   - ➔ It is the form of Acceptance Testing. Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
   - ➔ It comes under the category of both White Box Testing and Black Box Testing.

8. What is beta testing?
   - ➔ Beta testing is always performed by the customers at their own site.
   - ➔ It is not performed by Independent Testing Team.
   - ➔ Beta Testing is always open to the market and public.
   - ➔ It is usually conducted for software product. It is performed in Real Time Environment.
   - ➔ It is always performed outside the organization.
   - ➔ It is also the form of Acceptance Testing.
   - ➔ Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
   - ➔ It is only a kind of Black Box Testing.

9. What is component testing?
   - ➔ Component(Unit) – A minimal software item that can be tested in isolation.
   - ➔ Component testing is also known as unit testing.
   - ➔ It means “A unit is the smallest testable part of software.”
   - ➔ Component Testing – The testing of individual software components.

- ➔ Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
- ➔ Unit testing is the first level of testing and is performed prior to Integration Testing. Sometimes known as Unit Testing,
- ➔ Module Testing or Program Testing Component can be tested in isolation – stubs/drivers may be employed Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing. Functional and Non-Functional testing.

10. What is functional system testing?
    - ➔ Functional System Testing : A requirement that specifies a function that a system or system component must perform.
    - ➔ A Requirement may exist as a text document and/or a model.
    - ➔ There is two types of Test Approach Requirement Based Functional Testing Process Based Testing.

11. What is Non-Functional Testing?
    - ➔ Non-Functional Testing: Testing the attributes of a component or system that do not relate to functionality. e.g. reliability, efficiency, usability, interoperability, maintainability and portability.
    - ➔ May be performed at all Test levels (not just Non Functional Systems Testing).
    - ➔ Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g. performance test scaling.
    - ➔ Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.

12. What is GUI Testing?
    - ➔ 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 bars – tool bar, menu bar, dialog boxes and windows etc.

13. What is Adhoc testing?
    - ➔ Adhoc testing is an informal testing type with an aim to break the system. It does not follow any test design techniques to create test cases.
    - ➔ In fact is does not create test cases altogether! This testing is primarily performed if the knowledge of testers in the system under test is very high. Testers randomly test the application without any test cases or any business requirement document.
    - ➔ Adhoc 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 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.

14. What is load testing?
    - ➔ Load testing - Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.
    - ➔ Load testing is a kind of performance testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.
    - ➔ This testing usually identifies – The maximum operating capacity of an application.
    - ➔ Determine whether current infrastructure is sufficient to run the application.
    - ➔ Sustainability of application with respect to peak user load Number of concurrent users that an application can support, and scalability to allow more users to access it.
    - ➔ It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.

15. What is stress Testing?
    - ➔ Stress testing - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
    - ➔ Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
    - ➔ It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
    - ➔ Stress Testing is done to make sure that the system would not crash under crunch situations.
    - ➔ Stress testing is also known as endurance testing.

16. What is white box testing and list the types of white box testing?
    - ➔ White Box Testing: Testing based on an analysis of the internal structure of the component or system.
    - ➔ Structure-based testing technique is also known as 'white-box' or 'glass-box' testing technique because here the testers require knowledge of how the software is implemented, how it works.
    - ➔ In white-box testing the tester is concentrating on how the software does it. For example, a structural technique may be concerned with exercising loops in the software.
    - ➔ Types of white box testing:-
    - ➔ Statement coverage
    - ➔ Decision coverage
    - ➔ Condition coverage
17. What is black box testing? What are the different black box testing techniques?

- ➔ Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
- ➔ Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
- ➔ The testers have 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, not how it does it.
- ➔ Specification-based techniques are appropriate at all levels of testing (component testing through to acceptance testing) where a specification exists.
- ➔ For example, when performing system or acceptance testing, the requirements specification or functional specification may form the basis of the tests.
- ➔ The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.

18. Mention what are the categories of defects?
    - ➔ At ScienceSoft, we usually single out the following key types of defects:
    - ➔ Functional defects. Functional bugs can be revealed during smoke, system, integration, regression, and user acceptance testing
    - ➔ Performance defects
    - ➔ Usability defects
    - ➔ Security defects.

19. Mention what bigbang testing is?
    - ➔ In Big Bang testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
    - ➔ Big Bang testing has the advantage that everything is finished before integration testing starts.
    - ➔ The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
    - ➔ Here all component are integrated together at once, and then tested.

20. What is the purpose of exit criteria?
    - ➔ Purpose of exit criteria is to define when we STOP testing either at the:
    - ➔ End of all testing – i.e. product Go Live End of phase of testing (e.g. hand over from System Test to UAT).

21.When should "Regression Testing" be performed?
- ➔ when the system is stable and the system or the environment changes.
- ➔ when testing bug-fix releases as part of the maintenance phase It should be applied at all Test Levels.

- ➔ It should be considered complete when agreed completion criteria for regression testing have been met.
- ➔ Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation.
- ➔ Change in requirements and code is modified according to the requirement.
- ➔ New feature is added to the software
- ➔ Defect fixing
- ➔ Performance issue fix

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

1. Testing shows presence of Defects
2. Exhaustive Testing is Impossible!
3. Early Testing
4. Defect Clustering
5. The Pesticide Paradox
6. Testing is Context Dependent
7. Absence of error fallacy

- ➔ Testing shows presence of Defects:-
- ➔ Testing can show that defects are present, but cannot prove that there are no defects. 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. However Testing cannot prove that there are no defects present.

- ➔ Exhaustive Testing is Impossible! :-
- ➔ Testing everything including all combinations of inputs and preconditions is not possible. So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts. For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests. This is very unlikely that the project timescales would allow for this number of tests. So, accessing and managing risk is one of the most important activities and reason for testing in any project. We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions). That is we must Prioritise our testing effort using a Risk Based Approach.

- ➔ Early Testing:-
- ➔ Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives. Testing activities should start as early as possible in the development life cycle These activities should be focused on defined objectives – outlined in the Test Strategy Remember from our Definition of Testing, that Testing doesn't start once the code has been written!

- ➔ Defect Clustering:-

- ➔ A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures. Defects are not evenly spread in a system They are ‘clustered’ In other words, most defects found during testing are usually confined to a small number of modules Similarly, most operational failures of a system are usually confined to a small number of modules An important consideration in test prioritisation!

- ➔ Pesticide Paradox:-
- ➔ If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects. Testing identifies bugs, and programmers respond to fix them As bugs are eliminated by the programmers, the software improves As software improves the effectiveness of previous tests erodes.

- ➔ Testing is Context Dependent:-
- ➔ Testing is basically context dependent. Testing is done differently in different contexts
- ➔ Different kinds of sites are tested differently. For example
- ➔ Safety – critical software is tested differently from an e-commerce site. Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software 1 to 3 failures per KLOC typical for industrial software 0.01 failures per KLOC for NASA Shuttle code! Also different industries impose different testing standards.

- ➔ Absence of Errors Fallacy:-
- ➔ If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help. If we build a system and, in doing so, find and fix defects .... It doesn’t make it a good system Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations.

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

Quality Assurance	Quality Control	Tester
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 some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.

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.
process oriented activities.	Product oriented activities.	Product oriented activities.
Preventive activities.	It is a corrective process.	It is a preventive process.
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.

24. Difference between Smoke and Sanity?

➔

SMOKE TESTING	SANITY TESTING
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 ‘’Stability’’ of the system in order 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 is usually performed by testers.
Smoke testing is usually documented or scripted.	Sanity testing is usually not documented and not scripted.
This is subset of Regression testing.	This is subset of Acceptance testing.

Smoke testing is like general health check up.	Sanity testing is like specialized health check up.

25. Difference between verification and Validation.

➔

Verification	**Validation**
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

26. Explain types of Performance testing.

- ➔ Load testing
- ➔ Stress testing
- ➔ Endurance testing
- ➔ Spike testing
- ➔ Volume testing
- ➔ Scalability testing

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

- ➔ Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.
- ➔ Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.
- ➔ Error: An incorrect step, process, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner. See: bug, defect, error, exception.
- ➔ Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

28. Difference between Priority and Severity.

Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.

Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.
Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.
Who Decides the Defect	The testing engineer basically decides a defect’s severity level.	The product manager basically decides a defect’s priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

29. What is Bug Life Cycle?

➔ “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”

➔ The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’.

➔ When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.

➔ The process by which the defect moves through the life cycle is depicted next slide.

30. Explain the difference between Functional testing and NonFunctional testing.

➔

Parameters	**Functional**	**Non-functional testing**

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 behavior 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	Examples of Non-functional Testing Types
	● Unit testing ● Smoke testing ● User Acceptance ● Integration Testing ● Regression testing ● White box testing ● Black box testing ● Sanity testing	● Performance Testing ● Volume Testing ● Installation Testing ● Usability Testing ● Load Testing ● Stress Testing ● Security Testing ● Compatibility Testing ● MigrationTesting ● Penetration testing

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

Parameters	SDLC	STLC
Definition	SDLC, or software development life cycle, relates mainly to software development and includes all phases of software development, including testing.	In essence, STLC is related to software testing, meaning that it is a software testing process that entails several phases.
Relationship	As a whole, it covers the entire life cycle of the software and can be considered the predecessor.	Since it is part of SDLC and only involves testing, it is considered a child or successor.

Focus/Goal	SDLC aims to manage the entire process of software development from start to finish and to deliver a quality product that meets customer needs.	The focus is solely on test development and helps to make the testing process more sophisticated, consistent, and useful.
Performed	Phases of the SDLC are completed before those of the STLC.	Phases of STLC are carried out after the phases of SDLC.
Requirement Gathering	Business Analysts and Product Analysts collect requirements and prepare a Development Plan during the Requirements collection phase of the SDLC.	The QA (Quality Assurance) team will analyze requirement documents such as functional and non-functional requirements, and then prepare a System Test Plan as part of the Requirement Analysis phase of the STLC.

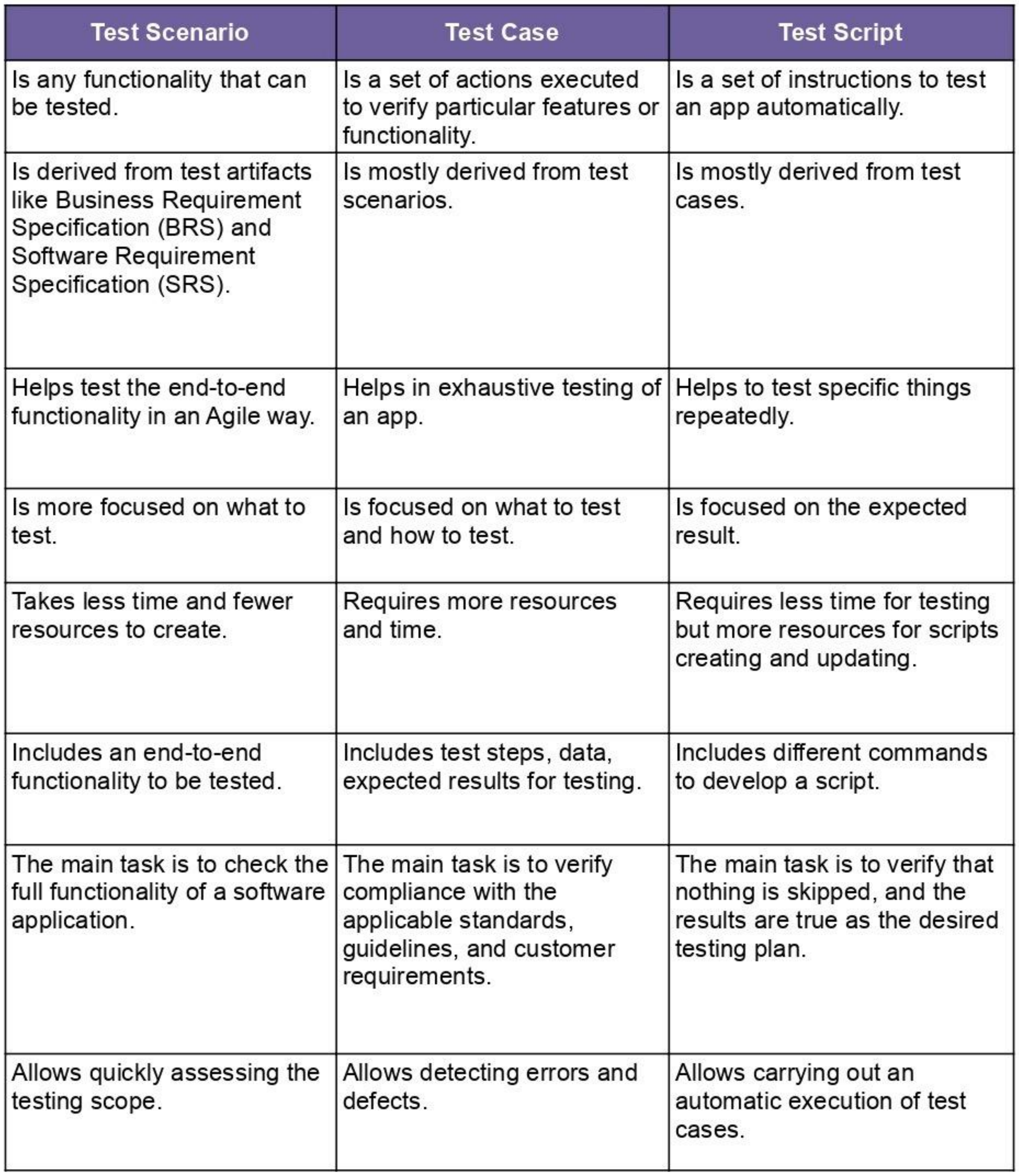
Different phases	The SDLC includes the following phases:Requirements Collection/PlanningAnalysis/Defining Designing the software programming or Coding (Building the Software)TestingDeployment/InstallationMaintenance	The STLC includes the following phases:Requirement AnalysisTest PlanningTest DevelopmentTest Environment SetupTest ExecutionTest Closure
Objective	Throughout the SDLC process, the intent is to overcome any hurdle on the way to successful software development.	Testing is only intended to find any weaknesses or pitfalls in the system.
Design phase	SDLC involves planning and designing software based on the requirements of the development team.	STLC involves the planning of tests by the testing team (Test Architect or Test Lead).

Coding phase	In this phase, programmers start writing code according to the designed document in any programming language to build the system from scratch.	Test cases and test scripts are developed by the testing team (quality assurance team) to verify the product’s quality. They prepare the test environment and execute the tests.
Environment Setup	As soon as the development team has written the code, they set up a test environment to validate it.	The testers ensure the test environment is prepared based on the prerequisites and conduct smoke tests to determine if the environment is stable enough for testing the product.
Testing Phase	The objective of this phase is to test the software. Among the activities included in this testing are Unit, Integration, System, Retest & Regression testing, etc., and the development team also	System integration testing is then conducted based on the test cases. All bugs and errors are reported, retested, and fixed. The product is also subject to regression tests and is signed off

	participates in fixing reported bugs.	as soon as it meets the exit criteria.
Deployment/Product Release	As soon as the application has been approved by the various testing teams, it is deployed in a production environment for real end-users.	After the product has been deployed, smoke testing and sanity testing take place in the production environment, and the testing team prepares test reports and an analysis matrix for analyzing the product.
Maintenance	If necessary, post-deployment support, enhancement, and update are included.	The QA team runs regression tests to check deployed maintenance code. The team maintains test cases and automated scripts to make sure that tests are updated.
Members Required	Throughout the SDLC, more people (developers) are needed.	The QA team runs regression tests to check deployed maintenance code. The team maintains test cases and automated scripts

		to make sure that tests are updated.
Output	The end result of SDLC is the creation of reusable software systems.	STLC results in a tested software system.

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



Test Scenario	Test Case	Test Script
Is any functionality that can be tested.	Is a set of actions executed to verify particular features or functionality.	Is a set of instructions to test an app automatically.
Is derived from test artifacts like Business Requirement Specification (BRS) and Software Requirement Specification (SRS).	Is mostly derived from test scenarios.	Is mostly derived from test cases.
Helps test the end-to-end functionality in an Agile way.	Helps in exhaustive testing of an app.	Helps to test specific things repeatedly.
Is more focused on what to test.	Is focused on what to test and how to test.	Is focused on the expected result.
Takes less time and fewer resources to create.	Requires more resources and time.	Requires less time for testing but more resources for scripts creating and updating.
Includes an end-to-end functionality to be tested.	Includes test steps, data, expected results for testing.	Includes different commands to develop a script.
The main task is to check the full functionality of a software application.	The main task is to verify compliance with the applicable standards, guidelines, and customer requirements.	The main task is to verify that nothing is skipped, and the results are true as the desired testing plan.
Allows quickly assessing the testing scope.	Allows detecting errors and defects.	Allows carrying out an automatic execution of test cases.

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

➔ A Test Plan is **a** detailed document that catalogs the test strategies, objectives, schedule, estimations, deadlines, and resources required to complete that project. Think of it as a blueprint for running the tests needed to ensure the software is working correctly – controlled by test managers.

You already know that making a Test Plan is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

34.What is priority?

Priority is defined as the order in which the defects should be resolved. The priority status is usually set by the testing team while raising the defect against the dev team mentioning the timeframe to fix the defect. The Priority status is set based on end users requirement.

35.What is severity?

One can define Severity as the extent to which any given defect can affect/ impact a particular software. Severity is basically a parameter that denotes the impact of any defect and its implication on a software's functionality. In other words, Severity defines the overall impact that any defect can have on a system.

36.Bug categories are..

➔
    Functional errors
    Syntax errors
    Logic errors
    Calculation errors
    Unit-level bugs
    System-level integration bugs
    Out of bounds bugs

37.Advantage of Bugzilla.

- Open source, free bug tracking tool.

- Automatic Duplicate Bug Detection.
- Search option with advanced features.
- File/Modify Bugs By Email.
- Move Bugs Between Installs.
- Multiple Authentication Methods (LDAP, Apache server).
- Time Tracking.
- Automated bug reporting; has an API to interact with system.

38.Difference between priority and severity.

Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.
Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.
Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.

Who Decides the Defect	The testing engineer basically decides a defect’s severity level.	The product manager basically decides a defect’s priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

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

➔ Scrum
➔ Kanban
➔ Extreme Programming (XP)
➔ Lean Development
➔ Crystal
➔ Feature driven development(FDD)

40.Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?

➔

Authentication	Authorization
Authentication verifies who the user is.	Authorization determines what resources a user can access.

Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user.	Authorization works through settings that are implemented and maintained by the organization.
Authentication is the first step of a good identity and access management process.	Authorization always takes place after authentication.
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or changeable by the user.
Example: By verifying their identity, employees can gain access to an HR application that includes their personal pay information, vacation time, and 401K data.	Example: Once their level of access is authorized, employees and HR managers can access different levels of data based on the permissions set by the organization.

The common problem faced in web testing

- Integration. Integration testing exposes problems with interfaces among different program components before deployment. ...
- Interoperability. ...
- Security. ...
- Performance. ...
- Usability. ...
- Quality Testing, Exceptional Services

46. When to used Usablity Testing?

If possible, usability testing can and should be conducted on the current iteration of a product before beginning any new design work, after you've begun the strategy work around a brand new site or app.

47. What is the procedure for GUI Testing?

- Testing the size, position, height, width of the visual elements
- Verifying and testing the error messages are displayed or not
- Testing different sections of the display screen
- Verifying the usability of carousel arrows
- Checking the navigation elements at the top of the page
- Checking the message displayed, frequency and content
- Verifying the functionality of proper filters and ability to retrieve results.
- Checking alignment of radio buttons, drop downs
- Verifying the title of each section and their correctness
- Cross-checking the colors and its synchronization with the theme

48. Write Agile manifesto principles.

Customer satisfaction through early and continuous software delivery – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.

Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.

Frequent delivery of working software – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.

Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned. Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.

Enable face-to-face interactions – Communication is more successful when development teams are co-located. 396

Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.

Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.

Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.

Simplicity – Develop just enough to get the job done for right now.

Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.