ASSIGNMENT

Module 2 (Manual Based)

Submitted by:

Anumol.G

BASIC:

B1. What is software testing?

Software testing is a process used to identify correctness, completeness and quality of developed computer software.

B2. What is Exploratory testing?

Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution.

B3. What is Traceability matrix?

A Traceability matrix is a document that details the technical requirements for a given test scenario and its current state.

B4. What is Boundary value testing?

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range.

B5. What is Equivalence partitioning testing?

Equivalence partitioning or equivalence class partitioning is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which testcases can be derived. In principle ,testcases are designs to cover each partition at least once.

B6. What is Integration testing?

Integration Testing - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

B7. What determines the level of risks?

Risk can be defined as the combination of the probability of an event occurring and the consequences if that event does occur.

B8. What is Alpha testing?

Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.

B9. What is beta testing?

Beta Testing is always performed at the time when software product and project are marketed.

B10. What is component testing?

Component (Unit): A minimal software item that can be tested in isolation. It means "A unit is the smallest testable part of software".

B11. What is functional system testing?

Testing based on an analysis of the specification of the functionality of a component or system. 'Specification' – E.g. Requirements specification, Use Cases, Functional specification or maybe undocumented

B12. What is 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

B13. What are the benefits of Independent Testing?

This degree of independence avoids author bias and is often more effective at finding defects and failures

• There is several level of independence in software testing which is listed here from the lowest level of independence to the highest:

- Tests by the person who wrote the item.
- Tests by another person within the same team, like another programmer.
- Tests by the person from some different group such as an independent test team.
- Tests by a person from a different organization or company, such as outsourced testing or certification by an external body.

B14. What is an equivalence partition (also known as an equivalence class)?

This is a software testing technique which divides the input date into many partitions. Values from each partition must be tested at least once. Partitions with valid values are used for Positive Testing. While, partitions with invalid values are used for negative testing.

B15. What is negative and positive testing?

Negative Testing can be performed on the system by providing invalid data as input.

Positive Testing can be performed on the system by providing the valid data input.

B16. What are the Experience-based testing techniques?

Experience based techniques are non- structured and do not rely on specification documents. This makes them unable to be measured in terms of coverage. In experience-based techniques, people's knowledge, skills and background are of prime importance to the test conditions and test cases.

B17. How much testing is enough?

No testing is enough, but we can maximize the test coverage by using a smart test approach. Smart testing optimizes the design verification process for maximum possible coverage ,given the product cycle time while keeping costs at or below the defined target.

B18. What is V-Model?

The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model.

B19. What is maintenance testing?

Maintenance testing: Testing the changes to an operational system or the impact of a changed environment to an operational system.

B20. What is test coverage?

Test coverage is defined as metric in software resting that measures the amount of testing performed by a set of test.

B21. 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.

B22. 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. Adhoc testing can be achieved with the testing technique called Error Guessing.

B23. What is bug zila?

Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep tra ck of outstanding problems with their product. It is written in Perl and uses MYSQL database.

B24. To Create HLR on this below Use Case.

1) UML diagram Online shopping

Functionality id	**Functionality name**
1	Check view item by customer
2	Check authentication after view item
3	Check identity provider provide items
4	Check process for make purchase by customer
5	Check include view item into make purchase
6	Check checkout process after make purchase
7	Check authentication after complete checkout
8	Check identity provide after complete checkout
9	Check credit payment services after complete checkout
10	Check pay pal after complete checkout
11	Check request for login by customer
12	Check authentication after login process

2)ATM

Functionality id	**Functionality name**
1	Check Withdraw cash by customer
2	Check bank process after withdraw cash
3	Check process for transfer funds by customer
4	Check bank after complete the transfer funds
5	Check refill machine by maintenance person

Intermediate

I1. 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.

Branch Condition testing

• Branch Condition Testing requires that the True and False of each Boolean

operand is tested (Boolean Operands in this example: If A > 30 and B >= 5)

• Branch Condition Combination testing

• Branch Condition Combination Coverage would require all combinations

of Boolean operands to be evaluated

• Modified Condition Decision testing

• Modified Condition Decision Coverage requires test cases to

show that each Boolean operand can independently affect the outcome of the decision

• Dataflow testing

• Data flow testing aims to execute sub-paths from points where each variable in a component is defined to points where it is referenced.

• Linear Code Sequence And Jump (LCSAJ) testing

- LCSAJ testing requires a model of the source code which identifies control flow jumps (where control flow does not pass to a sequential statement).

I2. 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.

- There are four specification-based or black-box technique:
    - Equivalence partitioning
    - Boundary value analysis
    - Decision tables
    - State transition testing
    - Use-case Testing
    - Other Black Box Testing
    - Syntax or Pattern Testing

I3. What are the different test levels?

Software Testing Levels

- Component Testing (Unit Testing): A unit is the smallest testable part of software.
- Integration Testing: Integration Testing is a level of the software testing process where individual units are combined and tested as a group
- Component Integration Testing: Testing performed to expose defects in the interfaces and interaction between integrated components
- System Integration Testing: It tests the interactions between different systems and may be done after system testing.
- System testing: The process of testing an integrated system to verify that it meets specified requirements

• User Acceptance Testing (UAT): User Acceptance Testing (Also known as Beta Testing) is performed by the end users of the software. They can be the customers themselves or the customers' customers.

I4. What is the difference between UAT (User Acceptance Testing) and System testing?

Acceptance testing	**System testing**
Focused on a validation type testing.	Validates just the software system as per the requirements specifications.
The goal of acceptance testing is to establish confidence in the system	It checks system functionalities and features
This approach is also often used for functional system test	Functional and Non-Functional Testing will be considered for testing
The usability of the component may be done during component testing	It's executed after integration testing
Black Box Testing method is used in Acceptance Testing.	Both Manual and Automation can be performed for system testing

I5. Mention the difference between Regression Testing and Retesting?

- **Retesting** means testing the functionality or bug again to ensure the code is fixed. If it is not fixed, defect needs to be re-opened. If fixed, defect is closed.

• **Regression testing** means testing your software application when itundergoes a code change to ensure that the new code has not affected other parts of the software.

I6. What is Latent defect?

Latent defects is a popular term in the dictionary of software testing which are hidden undetected flaws in software which cannot be identified by any user until some special set of operation are performed. These flaws can be detected only when a specific task is performed in an unusual circumstances.

I7. What is test management review and why it is important?

Test management process of managing the tests. A test management is also performed using tools to manage both types of tests, automated and manual, that have been previously specified by a test procedure.

Test management tools allow automatic generation of the requirement test matrix which is an indication of functional coverage of the application under test. It often has multifunctional capabilities such as test ware management, test scheduling, the logging of results, test tracking, incident management and test reporting.

It has a clear set of roles and responsibilities for improving the quality of product. It helps the development and maintenance of product metrics during the course of project.

I8. In manual testing what are stubs and drivers?

Stubs and Drivers do not implement the entire programming logic

of the software module but just simulate data communication with the calling module.

- Stub: Is called by the Module under Test.
- Driver: Calls the Module to be tested.

I9. What is the step you would follow once you find the defect?

- Dynamic Testing
- This testing technique needs computer for testing.
- It is done during Validation process.
- The software is tested by executing it on computer. Ex: Unit testing, integration testing, and system testing.
- Static Testing
- Static testing is the testing of the software work products manually, or with a set of tools, but they are not executed

.• It starts early in the Life cycle and so it is done during the verification process.

- It does not need computer as the testing of program is done without executing the program. For example: reviewing, walk through, inspection, etc.

• Most static testing techniques can be used to ‘test’ any form of document including source code, design documents and models, functional specifications and requirement specifications.

I10. Mention what are the categories of defects?

• **Data Quality/Database Defects**: Deals with improper handling of data in the database.

• Examples: • Values not deleted/inserted into the database properly

• Improper/wrong/null values inserted in place of the actual values

• **Critical Functionality Defects**: The occurrence of these bugs hampers the crucial functionality of the application. Examples: - Exceptions

• **Functionality Defects**: These defects affect the functionality of the application. • Examples: • All JavaScript errors

• Buttons like Save, Delete, Cancel not performing their intended functions

• A missing functionality (or) a feature not functioning the way it is intended to
• Continuous execution of loops

• **Security Defects**: Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.

• Examples: • Authentication: Accepting an invalid username/password

• Authorization: Accessibility to pages though permission not given

• **User Interface Defects**: As the name suggests, the bugs deal with problems related to UI are usually considered less severe.

• Examples: • Improper error/warning/UI messages

• Spelling mistakes

• Alignment problems

I11. Mention what the difference between a "defect" and a "failure" in software testing is?

Defects occur because human beings are fallible

• Also because of:

• time pressure

- complex code
- complex infrastructure
- changed technologies
- and/or many system interactions
- A Defect may result in a Failure
- A Failure is a ‘Deviation of the component or system from its expected delivery, service or result’
- Failures can be caused by environmental conditions as well • E.g. radiation, magnetism, electronic fields
- Pollution can cause faults in firmware or influence the execution of software by changing hardware conditions.

I12. Mention what bottom-up testing is?

BottomUp Integration Testing: In bottomup integration testing, module at the lowest level are developed first and other modules which go towards the 'main' program are integrated and tested one at a time. It is usually performed by the testing teams

I13. Mention what Top-down testing is?

Testing technique that involves starting at the stop of a system hierarchy at the user interface and using stubs to test from the top down until the entire system has been implemented. It is conducted by the testing teams.

I14. Mention what big bang testing is?

In Big Bang integration 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.

I15. Mention what the purpose behind doing end-to-end testing is?

Modern software systems are complex and are interconnected with multiple sub-systems A sub-system may be different from the current system or may be owned by another organization. If any one of the sub-system fails, the whole software system could collapse. This is major risk and can be avoided by End-to-End testing. End-to-End testing verifies the complete system flow. It increase test coverage of various sub-systems. It helps detect issues with sub-systems and increases confidence in the overall software product.

I16. What is risk-based testing?

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 preconditions). That is we must Prioritise our testing effort using a Risk Based Approach

I17. What is the purpose of exit criteria?

Exit Criteria:

- Successful Testing of Integrated Application.
- Executed Test Cases are documented
- All High prioritized bugs fixed and closed
- Technical documents to be submitted followed by release Notes.

I18. What is the purpose of entry criteria?

- Entry Criteria:
  - Unit Tested Components/Modules
  - All High prioritized bugs fixed and closed
  - All Modules to be code completed and integrated successfully.
  - Integration test Plan, test case, scenarios to be signed off and documented.
  - Required Test Environment to be set up for Integration testing

I19. When is used Decision table testing?

Decision Table Testing is a software testing methodology used to test system behaviour for various input combinations. In this systematic approach, the several input combinations and their corresponding system behaviour are represented in tabular form.

Table based technique where

- Inputs to the system are recorded
- Outputs to the system are defined

I20. As part of which test process do you determine the exit criteria?

Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the "enough testing". These criteria vary from project to project and are known as exit criteria.

- Test execution is assessed against the objectives defined in Test Planning
- This should be done for each Test Level (i.e. test stage)
- A group of test activities that are organized and managed together.
- Exit criteria come into picture, when:
- Maximum test cases are executed with certain pass percentage.
- Bug rate falls below certain level.
- When achieved the deadlines.

I21. Mention what the difference between Pilot and Beta testing is?

Pilot testing	**Beta testing**
It is performed before the launch of product in the market	It is performed after the launch of the product
Feed back comes from some selected users	Feedback comes from customer directly
It is done before beta testing	It is done after pilot testing
It requires an environment or lab	It doesn't require an environment or lab
This testing is done exactly between the UAT and production	This testing is exactly after production

I22. What is random/monkey testing? When is it used?

- Monkey Testing
- Randomly test the product or application without test cases with a goal to break the system.

It is mainly used when there is lack of time or resource as these tests are inexpensive. There are two major types of monkey tests Smart test and dumb test. In smart testing, the inputs are generated from the expected usage statistics whereas in dumb testing, the input is not on the basis of the usage statistic

I23. Types of Adhoc Testing?

There are different types of Adhoc testing and they are listed as below:

- Buddy Testing

- Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after unit testing completion.

- Pair testing : Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing.

- Monkey Testing: Randomly test the product or application without test cases with a goal to break the system.

I24. Consider the following techniques. Which are static and which are dynamic techniques? Static Testing: Static testing techniques involve examination of the project's documentation, software and other information about the software products without executing them.

Dynamic testing: Testing that involves the execution of the software of a component or system.

Static testing techniques rely on the manual examination (reviews) and automated analysis (static analysis) of the code or other project documentation. Dynamic testing techniques are boundary value analysis and equivalence Partitioning.

I25. What are the phases of a formal review?

- **Logging phase**: In this phase the issues and the defects that have been identified during the preparation step are logged page by page. The logging is basically done by the author or by a scribe. Scribe is a separate person to do the logging and is especially useful for the formal review types such as an inspection.

- **Discussion phase:** If any issue needs discussion then the item is logged and then handled in the discussion phase. As chairman of the discussion meeting, the moderator takes care of the people issues and prevents discussion from getting too personal and calls for a break to cool down the heated discussion. The outcome of the discussions is documented for the future reference.

• **Decision phase**: At the end of the meeting a decision on the document under review has to be made by the participants, sometimes based on formal exit criteria. Exit criteria are the average number of critical and/or major defects found per page (for example no more than three critical/major defects per page). If the number of defects found per page is more than a certain level then the document must be reviewed again, after it has been reworked.

I26. When should "Regression Testing" be performed?

Regression testing is performed when changes are made to the existing functionality of the software or if there is a bug fix in the software.

I27. What type of review requires formal entry and exit criteria, including metrics? The formal reviews the moderator performs the entry check and also defines the formal exit criteria. The entry check is done to ensure that the reviewer's time is not wasted on a document that is not ready for review. • After doing the entry check if the document is found to have very little defect then it's ready to go for the reviews.

• So, the entry criteria are to check that whether the document is ready to enter the formal review process or not. Hence the entry criteria for any document to go for the reviews are:

• The documents should not reveal a large number of major defects.

• The documents to be reviewed should be with line numbers.

• The documents should be cleaned up by running any automated checks that apply.

• The author should feel confident about the quality of the document so that he can join the review team with that document

I28. When should testing be stopped?

All the high priority bugs are fixed.

• The rate at which bugs are found is too small.

• The testing budget is exhausted.

• The project duration is completed.

• The risk in the project is under acceptable limit.

• Practically, we feel that the decision of stopping testing is based on the level of the risk acceptable to the management. The risk can be measured by Risk analysis but for small duration / low budget / low resources project, risk can be deduced by simply: –

• Measuring Test Coverage.

• Number of test cycles.

• Number of high priority bugs.

I29. Faults found should be originally documented by whom?

By Testers

I30. Why is incremental integration preferred over "big bang" integration?

In Incremental integration testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects. This approach is known as incremental integration testing. On the contrary, big bang is one other integration testing technique, where all the modules are integrated in one shot

I31. What is the purpose of test design technique?

A test design technique basically helps us to select a good set of tests from the total number of all possible tests for a given system. There are many different types of software testing technique, each with its own strengths and weaknesses. Each individual technique is good at finding particular types of defect and relatively poor at finding other types. • For example, a technique that explores the upper and lower limits of a single input range is more likely to find boundary value defects than defects associated with combinations of inputs. Similarly, testing performed at different stages in the software development life cycle will find different types of defects; component testing is more likely to find coding logic defects than system design defects.

I32. What is 7 key principles? Explain in detail?

• Testing shows presence of Defects

• Exhaustive Testing is Impossible!

• Early Testing • Defect Clustering

• The Pesticide Paradox

• Testing is Context Dependent

• Absence of Errors 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.

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.

Early Testing •Testing 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

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'

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 is Context Dependent • Testing is basically context dependent. Testing is done differently in different contexts • Different kinds of sites are tested differently.

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

I33. Difference between QA v/s QC v/s Tester

Quality Assurance	Quality Control	Testing

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
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

I34. Difference between Smoke and Sanity?

Smoke	**Sanity**
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is a subset of Regression testing	Sanity testing is a subset of Acceptance testing
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
testing Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

I35. 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 doesn’t include the execution of the code	It includes the execution of the code
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

I36. Explain types of Performance testing.

Types of Performance Testing

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

Load Testing: Testing technique that puts demand on a system or device and measures its response. It is usually conducted by the performance engineers.

Stress Testing: Testing technique which evaluates a system or component at or beyond the limits of its specified requirements. It is usually conducted by the performance engineer.

Endurance Testing: Type of testing which checks for memory leaks or other problems that may occur with prolonged execution. It is usually performed by performance engineers.

Volume Testing: Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner. It is usually conducted by the performance engineer. Scalability Testing: Part of the battery of non-functional tests which tests a software application for measuring its capability to scale up -

be it the user load supported, the number of transactions, the data volume etc. It is conducted by the performance engineer.

I37. What is Error, Defect, Bug and failure?

Bug – It is found in the development environment before the product is shipped to the respective customer.

Error – It is the Deviation from actual and the expected value.

Defect – It is found in the product itself after it is shipped to the respective customer

Failure - A Failure is a 'Deviation of the component or system from its expected delivery, service or result

I38. Difference between Priority and Severity?

Priority	**Severity**
Priority is a parameter to decide the order in which defects should be fixed	Severity is a parameter denotes the impact defect on the software
Priority means how fast defect has to be fixed	Severity means how severe defect is affecting the functionality
Priority is related to scheduling to resolve the problem. Product manager decides the priorities of defects	Severity is related to the quality standard.Testing engineer decides the severity level of the defect
Its value is subjective	Its value is objective
Its value changes from time to time	Its value doesn't change from time to time
Priority of 3 types: low, high, medium	5types: Critical, Major, Moderate, Minor, cosmetic

I39. 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.

I40. Explain the difference between Functional testing and NonFunctional testing

Functional	**Non – Functional**
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
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Easy to do manual testing	Tough to do manual 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.

I41. Define Key elements of Bug zila.

Key features of Bugzilla include

• Advanced search capabilities

• E-mail Notifications

• Modify/file Bugs by e-mail

• Time tracking

• Strong security

• Customization

• Localization

Advanced

A1. What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

Software Development Life Cycle (SDLC) is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support.

SDLC Phases

Requirements Collection/Gathering	Establish Customer Needs
Analysis	Model And Specify the requirements- “What”
Design	Model And Specify a Solution – “Why”
Implementation	Implementation Construct a Solution In Software
Testing	Validate the solution against the requirements
Maintenance	Repair defects and adapt the solution to the new requirements

Software Testing Life Cycle (STLC) is a sequence of different activities performed during the software testing process

Stages of STLC Fundamental Test Process (STLC)

- Test Planning and Controlling
- Test Analysis and Design
- Test Implementation and Execution
- Evaluating Exit Criteria and Reporting
- Test Closure Activities.

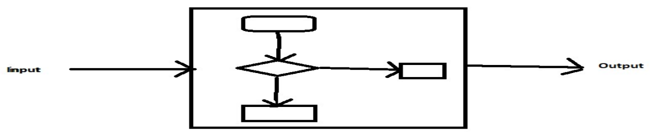
Difference between SDLC and STLC

SDLC	STLC
Related to software development	Related to software testing
Besides development other phases like testing is also included	In focuses only on testing the software
SDLC involves total six phases or steps.	STLC involves only five phases or steps

In SDLC, more number of members are required for the whole process.	In STLC, less number of testers are needed
In SDLC, the development team makes the plans and designs	In STLC, testing team 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 STLC phases	STLC phases are performed after SDLC phases

A2. In white box testing, what do you verify?

White box testing technique is used by both developers as well as testers. It helps them understand which line of code is actually executed and which is not. This may indicate that there is either missing logic or a typo.



White box testing involves the testing by looking at the internal structure of the code & when you completely aware of the internal structure of the code then you can run your test cases & check whether the system meet requirements mentioned in the specification document. Based on derived test cases the user exercised the test cases by giving the input to the system & checking for expected outputs with actual output.

In the White box testing following steps are executed to test the software code:

- Basically verify the security holes in the code.

- Verify the broken or incomplete paths in the code.
- Verify the flow of structure mention in the specification document
- Verify the Expected outputs
- Verify the all conditional loops in the code to check the complete functionality of the application.
- Verify the line by line or Section by Section in the code & cover the 100% testing.

Above steps can be executed at the each steps of the STLC i.e. Unit Testing, Integration & System testing.

A3. What is the difference between static and dynamic testing?

Static Testing	**Dynamic Testing**
Testing done without executing the program	Testing done by executing the program
This testing does verification process	Dynamic testing does validation process
Static testing is about prevention of defects	Dynamic testing is about finding and fixing the defects
Static testing gives assessment of code and documentation	Dynamic testing gives bugs/bottlenecks in the software system
Static testing involves checklist and process to be followed	Dynamic testing involves test cases for execution
This testing can be performed before compilation	Dynamic testing is performed after compilation
Static testing covers the structural and statement coverage testing	Dynamic testing covers the executable file of the code
Cost of finding defects and fixing is less	Cost of finding and fixing defects is high
Return on investment will be high as this process involved at early stage	Return on investment will be low as this process involves after the development phase
More reviews comments are highly recommended for good quality	More defects are highly recommended for good quality.
Requires loads of meetings	Comparatively requires lesser meeting

A4. What Test Plans consists of?

A document describing the scope, approach, resources and schedule of intended test activities apply to the software under test.

Test design, scope, test strategies, approach are various details that Test plan document consists of.

1. Test case identifier
2. Scope
3. Features to be tested
4. Features not to be tested
5. Test strategy & Test approach
6. Test deliverables
7. Responsibilities
8. Staffing and training
9. Risk and Contingencies

A5. What is the difference between test scenarios, test cases, and test script?

Test Scenario • A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility.

• Test Scenario is 'What to be tested'

• Test scenario is nothing but test procedure.

• The scenarios are derived from use cases.

• Test Scenario represents a series of actions that are associated together.

• Scenario is thread of operations

Test Case • Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

• Test Case is 'How to be tested'

• Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.

• Test cases are derived (or written) from test scenario.

• Test Case represents a single (low level) action by the user.

• Test cases are set of input and output given to the System.

• Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing.

Test Script • A set of sequential instruction that detail how to execute a core business function

• One script is written to explain how to simulate each business scenario

• Written to a level of detail for which someone else (other than the script writer) would be able to easily execute

• Identifies the test condition that is being satisfied for each step, if applicable

• Identified the input/test data that should be entered for each transaction

• Identifies the expected results for each step, if applicable

• Should demonstrate how the system can support the HCA warehouse business processes

• A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected.

• There are various means for executing test scripts.

• Manual Testing

• Automation Testing

A6. What are the two parameters which can be useful to know the quality of test execution?

we can use two parameters * **Defect reject ratio** * **Defect leakage ratio** These two ratios are two ways to determine the quality of test execution

- **Defect rejection ratio:** (No. of defects rejected/ total no. of defects raised) X 100
- **Defect leakage ratio:** (No. of defect missed/total defects of software) X 100

A7. What all things you should consider before selecting automation tools for the AUT?

Before selecting an automation tool, it is highly advised to understand the technology that the application is built on and the test requirements. For instance, one may want to find out whether the application testing requires only functional or both functional and performance testing

A8. How will you conduct Risk Analysis?

To perform a risk analysis is useful for almost any kind of decision-making process — no matter your professional role or sector. When you analyze risk, you can develop soft skills such as critical thinking and problem-solving.

- Risk analysis helps identify potential problems that could arise during a project or process. You can analyze risk to: Reduce the impact of a negative event.
- Evaluate whether there are more benefits to a project than risks before initiation.
- Plan the company’s response to emergencies or other adverse events.
- Eliminate risks during a process.

Risk analysis is a useful tool to use in the decision-making process. It allows you to identify the potential benefits and detriments of each option, evaluate the likelihood of problems occurring and decide whether to move forward considering such risks. Once you have identified potential risks, you can determine how to manage them and even develop a comprehensive preventative plan

Perform risk analysis:

1.Identify the risk

2.Define level of uncertainty

3.Estimate the impact of uncertainty

4.Complete the risk analysis model

5.Analyse the result

6.Implement the solution

A9. What are the categories of debugging?

In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Debugging Process: Steps involved in debugging are:

- Problem identification and report preparation.
- Assigning the report to software engineer to the defect to verify that it is genuine.
- Defect Analysis using modeling, documentations, finding and testing candidate flaws, etc.
- Defect Resolution by making required changes to the system.
- Validation of corrections

A10. Explain what Test Plan is? What is the information that should be covered in Test Plan

• A document describing the scope, approach, resources and schedule of intended test activities

• Determining the scope and risks, and identifying the objectives of testing.

• Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.

• Integrating and coordinating the testing activities into the software life cycle activities:

• acquisition, supply, development, operation and maintenance.

• Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, and how the test results will be evaluated?

• Scheduling test analysis and design activities.

• Scheduling test implementation, execution and evaluation.

• Assigning resources for the different activities defined

• Defining the amount, level of detail, structure and templates for the test documentation. Test Plan & Strategy

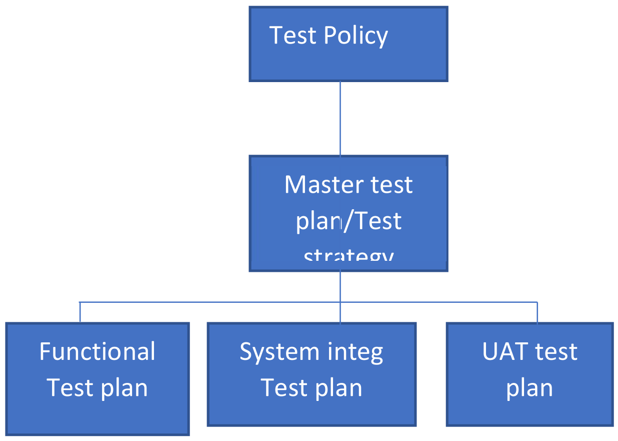
• All projects require a set of plans and strategies which define how the testing will be conducted.

• There are number of levels at which these are defined:

Defines how the organisation will conduct testing

Defines how the project will conduct testing

Defines how each level of testing will be conduct



Test Planning Factors

• Factors which affect test planning

• The organization's test policy

• Scope of the testing being performed

• Testing objectives

• Project Risks – e.g. business, technical, people

• Constraints – e.g. business imposed, financial, contractual etc

• Criticality (e.g. system/component level)

• Testability • Availability of resources

• Test plans are continuously refined

• As more information becomes available

• As new risks arise or others are mitigated

• Not set in concrete, but changes must be carefully managed

Test Planning Activities

• Approach: Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.

• Integrating and coordinating the testing activities into the software life cycle activities: acquisition, supply, development, operation and maintenance.

• Making decisions about:

• what to test • Who do testing? i.e. what roles will perform the test activities

• when and how the test activities should be done and when they should be stopped (exit criteria – see next slides) • how the test results will be evaluated

• Assigning resources for the different tasks defined.

• Test ware: Defining the amount, level of detail, structure and templates for the test documentation.

• Selecting metrics for monitoring and controlling test preparation and execution, defect resolution and risk issues.

• Process: Setting the level of detail for test procedures in order to provide enough information to support reproducible test preparation and execution.

A11. How can you eliminate the product risk in your project?

Risk – 'A factor that could result in future negative consequences; usually expressed as impact and likelihood'

Types of Risk • A Risk could be any future event with a negative consequence .You need to identify the risks associated with your project

• Risks are of two types

• Project Risks

• Product Risk

Product risks would be Flight Reservation system not installing in test environment • Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing

A12. What is the common risk that leads to project failure?

Risk – 'A factor that could result in future negative consequences; usually expressed as impact and likelihood'

Types of Risk

• A Risk could be any future event with a negative consequence .You need to identify the risks associated with your project

• Risks are of two types

• Project Risks

• Product Risk

Types of Risk Examples

Project risk is Senior Team Member leaving the project abruptly.

• Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10 .

• But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place

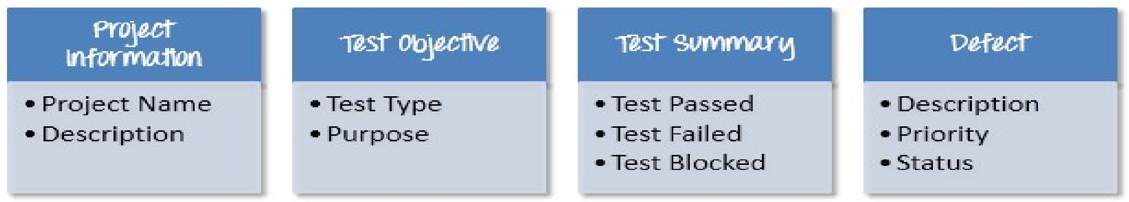
The common risk that leads to a project failure are

- Not having enough human resource
- Testing environment may not be setup properly
- Limited budget
- Limited time.

A13. What does a typical test report contain? What are the benefits of test reports?

Test Report is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the testing is performed. Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.
For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed.

Test Report Example



A test report contains the following things:

Project Information

All information of the project such as the project name, product name, and version should be described in the test report.

Test Objective

Test Report should include the objective of each round of testing, such as Unit Test, Performance Test, System Test

Test summary

This section includes the summary of testing activity in general. Information detailed here includes

- The number of test cases executed
- The numbers of test cases pass
- The numbers of test cases fail
- Pass percentage
- Fail percentage
- Comments

Defect

One of the most important information in Test Report is defect. The report should contain following information

- Total number of bugs
- Status of bugs (open, closed, responding)
- Number of bugs open, resolved, closed

The benefits of test reports are:

Current status of project and quality of product are informed

If required, stakeholder and customer can take corrective action

A final document helps to decide whether the product is ready for release

A14. Explain which test cases are written first black boxes or white boxes?

Normally black box test cases are written first and white box test cases later. In order to write black box test cases we need the requirement document and, design or project plan. All these documents are easily available at the initial start of the project. White box test cases cannot be started in the initial phase of the project because they need more architecture clarity which is not available at the start of the project. So normally white box test cases are written after black box test cases are written.

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.

• What a system does, rather than HOW it does it

• Typically used at System Test phase, although can be useful throughout the test lifecycle

• The tester is oblivious to the system architecture and does not have access to the source code. • Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

• **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. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.

• Structure-based techniques are also used in system and acceptance testing, but the structures are different. For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing. Testing based upon the structure of the code. Typically undertaken at Component and Component Integration Test phases by development teams. White box testing is the detailed investigation of internal logic and structure of the code.

• White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.

• The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

A15. Mention what the different types of test coverage techniques are?

• Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and is known as test coverage.

• The basic coverage measure is where the 'coverage item' is whatever we have been able to count and see whether a test has exercised or used this item.

$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

Types of Coverage

• The different types of coverage are:

• Statement coverage

• Decision coverage

• Condition coverage

Statement/Segment Coverage • The statement coverage is also known as line coverage or segment coverage. • The statement coverage covers only the true conditions. • Through statement coverage we can identify the statements executed and where the code is not executed because of blockage. • In this process each and every line of code needs to be checked and executed. • Aim is to display that all executable statements have been run at least once • The statement coverage can be calculated as shown below:

$$\text{Statement coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100\%$$

Decision/Branch Coverage

• Decision coverage also known as branch coverage or all-edges coverage.

• It covers both the true and false conditions unlikely the statement coverage.
• A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.

• Aim is to demonstrate that all Decisions have been run at least once

• Decision and Branch Test coverage for a piece of code is often the same, but not always

• With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF.

• The decision coverage can be calculated as shown.

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

Condition Coverage

• This is closely related to decision coverage but has better sensitivity to the control flow.

• However, full condition coverage does not guarantee full decision coverage.

• Condition coverage reports the true or false outcome of each condition.

• Condition coverage measures the conditions independently of each other.

A16. Mention what the basic components of defect report format are?

Defect Reporting

• Defect reports are among the most important deliverables to come out of test. They are as important as the test plan and will have more impact on the quality of the product than most other deliverables from the test. Effective defect reports will:

• Reduce the number of defects returned from development

• Improve the speed of getting defect fixes

• Improve the credibility of test

• Enhance teamwork between test and development

Defect Report Fields

Provide enough detail while reporting the bug keeping in mind the people who will use it – test lead, developer, project manager, other testers, new testers assigned etc.

• This means that the report you will write should be concise, straight and clear.

• Following are the details your report should contain:

• Bug Title • Bug identifier (number, ID, etc.)

• The application name or identifier and version / Test Case Id or Description

• The function, module, feature, object, screen, etc. where the bug occurred

• Environment (OS, Browser and its version)

• Bug Type or Category/Severity/Priority

• Bug status (Open, Pending, Fixed, Closed, Re-Open)

• Test case name/number/identifier

• Bug description

• Steps to Reproduce • Actual Result

• Tester Comments Defect Report Fields

• Bug Category: Security, Database, Functionality (Critical/General), UI

• Bug Severity: Severity with which the bug affects the application – Very High, High, Medium, Low, Very Low

• Bug Priority: Recommended priority to be given for a fix of this bug – P0, P1, P2, P3, P4, P5 (P0-Highest, P5-Lowest)

WHAT DOES THE TESTER DO WHEN THE DEFECT IS FIXED?

• Once the reported defect is fixed, the tester needs to re-test to confirm the fix. This is usually done by executing the possible scenarios where the bug can occur. Once retesting is completed, the fix can be confirmed and the bug can be closed. This marks the end of the bug life cycle.

A17. Mention what the basic components of Test case format are?

- Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

- Test Case is 'How to be tested'

- Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.

- Test cases are derived (or written) from test scenario.

- Test Case represents a single (low level) action by the user.

- Test cases are set of input and output given to the System. Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing.

The main components which are always available and included in every test case:

- Test case ID • Product Module ID

- Product version (Optional)

- Revision history (Optional)

- Purpose/ Test Case Description

- Assumptions (Optional)

- Pre-Conditions(Optional)

- Test Steps

- Expected Outcome/Result

- Actual Outcome/Result

- Post Conditions(Pass/Fail)

- The Step # Identifies the task sequence in the script

- Action/Input Data
- The Action Steps details the task to be performed
- Write action steps in terms that support execution (action oriented)
- Level of detail should reflect core business function, not system navigation
- The Input Data describes what needs to be entered for a step (if applicable), It's called Test Data.
- Include all search criteria (Ex. First name, last name)
- Expected Results
- The Expected Results documents what results are anticipated for this step
- Include detail needed for business process (Ex. Required fields, external system interfaces)
- Actual Results • Where the "script executor" enters the result that occurred from this step
- Be specific • Developing test material can be split into two distinct stages:
- Defining "what" needs to be tested
- Defining "how" the system should be tested
- This process can vary from organisation to organisation, can be very formal or very informal with little documentation
- The more formal, the more repeatable the tests, but it does depend on the context of the testing being carried out
- The process of identifying test conditions and designing tests consists of the following steps:
- Identify and Defining Test Conditions
- Specifying Test Cases
- Specifying Test Procedures

- Developing a Test Execution Schedule

A18. Explain in a testing project what testing activities would you automate?

- Tests that should be automated: Business critical paths - the features or user flows that if they fail, cause a considerable damage to the business.
- Tests that need to be run against every build/release of the application, such as smoke test, sanity test and regression test.
- Tests that need to run against multiple configurations — different OS & Browser combinations.
- Tests that execute the same workflow but use different data for its inputs for each test run e.g. data-driven.
- Tests that involve inputting large volumes of data, such as filling up very long forms.
- Tests that can be used for performance testing, like stress and load tests.
- Tests that take a long time to perform and may need to be run during breaks or overnight.
- Tests during which images must be captured to prove that the application behaved as expected, or to check that a multitude of web pages looks the same on multiple browsers.

A19. Explain in a testing project what testing activities would not to automate?

Test that should be not automated:

- Tests that you will only run only once. The only exception to this rule is that if you want to execute a test with a very large set of data, even if it's only once, then it makes sense to automate it.
- User experience tests for usability (tests that require a user to respond as to how easy the app is to use).
- Tests that need to be run ASAP. Usually, a new feature which is developed requires a quick feedback so testing it manually at first
- Tests that require ad hoc/random testing based on domain knowledge/expertise - Exploratory Testing.
- Intermittent tests. Tests without predictable results cause more noise that value. To get the best value out of automation the tests must produce

predictable and reliable results in order to produce pass and fail conditions.

- Tests that require visual confirmation, however, we can capture page images during automated testing and then have a manual check of the images.
- Test that cannot be 100% automated should not be automated at all, unless doing so will save a considerable amount of time.

A20. What is the difference between Testing Techniques and Testing Tools?

Testing Techniques

Software testing techniques are the ways employed to test the application under test against the functional or non-functional requirements gathered from business. Each testing technique helps to find a specific type of defect.

- Testing techniques are methods or ways or approaches towards testing a particular product.
- Testing techniques are the methods approaches guidelines of how to do testing like functional, security testing etc.

Testing Tools

Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis

- Testing tools are some tools which will help us perform the testing in a required testing technique.
- Testing tools are something which helps us in executing testing techniques like firebug, selenium etc
- Testing tools would be defined based on the testing technique selected.

A21. What are the different Methodologies in Agile Development Model?

- There are various methodologies present in agile testing and those are listed below:
- Scrum
- eXtreme Programming
- Below listed methodologies are used less frequently

• Dynamic System Development Method (DSDM)

• This is an Iterative and incremental approach that emphasizes on the continuous user involvement.

• Test Driven Development (TDD)

• This is a technique which has short iterations where new test cases covering the desired improvement or new functionality are written first.

• Feature Driven Development

• This is an iterative and incremental software development process and this can aim depends on the features.

• XBreed • Agile enterprise previously known as Xbreed .It is agile way of managing, architecting and monitoring the enterprise.

• Crystal • Crystal is an adaptive technique mainly used for software development methodologies.

Scrum

• SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment. Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members). It consists of three roles and their responsibilities are explained as follows:

• Scrum Master: Master is responsible for setting up the team, sprint meeting and removes obstacles to progress

• Product owner: The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration

• Scrum Team: Team manages its own work and organizes the work to complete the sprint or cycle

eXtreme Programing

• This is a light weight agile testing methodology in which development and testing happen in parallel. Business requirements are gathered in terms of stories.

• All those stories are stored in a place called parking lot.

• In this type of methodology, releases are based on the shorter cycles called Iterations with span of 14 days' time period.

• Each iteration include phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

A22. Who is responsible for document all the issues problems and open point that were identified during the review meeting?

SCRIBE

The review meeting consists of three phases:
• Logging phase:
• In this phase the issues and the defects that have been identified during the preparation step are logged page by page.
• The logging is basically done by the author or by a scribe.
• Scribe is a separate person to do the logging and is especially useful for the formal review types such as an inspection.
• Every defects and it's severity should be logged in any of the three severity classes given below:
• Critical: The defects will cause downstream damage.
• Major: The defects could cause a downstream damage.
• Minor: The defects are highly unlikely to cause the downstream damage.
• During the logging phase the moderator focuses on logging as many defects as possible within a certain time frame and tries to keep a good logging rate (number of defects logged per minute).
• In formal review meeting the good logging rate should be between one and two defects logged per minute.

• Discussion phase:
• If any issue needs discussion then the item is logged and then handled in the discussion phase.
• As chairman of the discussion meeting, the moderator takes care of the people

issues
and prevents discussion from getting too personal and calls for a break to cool down
the heated discussion.

- The outcome of the discussions is documented for the future reference.
- Decision phase:
- At the end of the meeting a decision on the document under review has to be made
by the participants, sometimes based on formal exit criteria.
- Exit criteria are the average number of critical and/or major defects found per page
(for example no more than three critical/major defects per page).
- If the number of defects found per page is more than a certain level then the document must be reviewed again, after it has been reworked.

A23. Explain Agile Methodologies?

- There are various methodologies present in agile testing and those are listed below:
- Scrum
- eXtreme Programming
- Below listed methodologies are used less frequently
- Dynamic System Development Method (DSDM)
- This is an Iterative and incremental approach that emphasizes on the continuous user involvement.
- Test Driven Development (TDD)
- This is a technique which has short iterations where new test cases covering the desired improvement or new functionality are written first.
- Feature Driven Development
- This is an iterative and incremental software development process and this can aim depends on the features.
- XBreed
- Agile enterprise previously known as Xbreed .It is agile way of managing, architecting and monitoring the enterprise.
- Crystal
- Crystal is an adaptive technique mainly used for software development methodologies

A24. List out Test strategies for Mobile Device Application

• Manual and Automated Testing

• Database Testing

• Compatibility Testing

• Functional Testing

• Interoperability Testing

• Mobile Analytics Testing

• Power Consumption Testing

• Usability Testing

• Security Testing

• Mobile Device Emulators

• Manual Testing

• Test scripts are executed by a tester using an actual device under various scenarios • The most labor intensive and time consuming technique but necessary for select, critical test cases

• Automated Testing

• Test scripts are executed using emulator(s) and performance testing tool(s) eggPlant is an example of a broad based QA automation testing tool that emulates mobile devices and automates the testing; eggPlant can be downloaded for the Windows or Mac platforms; interfaces with devices, device emulators and other tools to consolidate/complete the testing process

• Database Testing

• Check for data integrity and errors while editing, deleting and modifying the forms and all other DB related information • Executed manually without use of tools

• Compatibility Testing

• Assures the application works as intended with the selected device, operating system, screen size, display and internal hardware • Can be tested with automation, the following are example tools that simulate different devices, operating systems, screens, etc.

• Functional Testing

• Includes the testing of controls, storage media handling options, and other operational aspects • Functionality testing for the mobile application is black-box testing and assures that the application functions per the business specifications • Executed manually without use of tools

• Interoperability Testing

• Includes the testing of different functionalities within the device,

• Usability Testing

• Used to verify mobile interface (UI), navigation, and application, as well as consistency, and soberness

• Executed manually without use of tools

• Mobile Analytics Testing

• Test the correct implementation of analytics: verify page and link tags, redirects, page source and user attributes as well as data capture • Can be tested with automation, the following are example tools

• Performance Testing is used to load and stress test the mobile application and database servers; Empirixe Tester to preform load and stress testing, eLoadExpert simulates concurrent users

• Test performance under realistic conditions of wireless traffic and maximum user load

• Security Testing • Security issues include: sensitive data that may remain on handsets, or passwords that are displayed on the screen • Testing with automation is not feasible, what is required to be tested is the behavior of the actual handset and security designs vary among handsets – each device must be individually tested

• Power Consumption/ Battery Life Testing

• Testing uncovers defects related to battery drainage caused by the application (device settings can also drain battery life and may make it difficult to determine the cause of the drain), the following are examples of different methods to test power consumption: • iPhone, iPod & iPad settings are adjusted; screen brightness, minimize use of location services, turn off push notifications, turn off other downloaded applications, fetch new data less frequently and turn off push mail; run the application to determine the rate it took for the battery life to decrease (testing executed manually without any testing tools)

A25. What is Test Plan and contents available in a Test Plan?

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

1. is the main document often called as master test plan or a It project test plan and usually developed during the early phase of the project. Test plan templates will be different within different companies. However some common contents of plan are mentioned below.

- Introduction: Brief introduction of the test strategies, process and methodologies.
- Objective : For example, Application Under Test conforms to functional and non-functional requirements.
- Features to be tested : What is inscope and what is out of scope.
- Testing approach or test strategy :For eg, exploratory testing, functional testing, regression testing, user interface testing, component testing, integration testing.
- Test Environment: Hardware or software requirements or additional tools required.
- Roles and responsibilities: Information about testers,managers and test leads.
- Schedule :How many man hours are required.
- Deliverables :

1. Test Plan
2. Test Cases
3. Requirement Traceability Matrix
4. Bug Reports
5. Test Strategy
6. Test Metrics

- Risks and contingency

A26. Tell some examples of Bug Severity and Bug Priority?

Defect Priority

• Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.

• **For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.**

• Priority can be of following types:

• Low: The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.

• Medium: The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.

• High: The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

• Critical: Extremely urgent, resolve immediately

• Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.

• **For example: If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.**

• Severity can be of following types:

• Critical: The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
• Severity can be of following types:

- Major (High): The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.

- Moderate (Medium): The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.

- Minor (Low): The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.

- Cosmetic: The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

A27. How is 'Build' different from 'Release'?

The main difference between Build and Release in Software Testing is that Build is a version of the software the development team hands over to the testing team for testing purpose while Release is the software the testing team hands over to the customer .

After developing the software module, developers convert the source codes into a standalone form or an executable code. Then the development team hands over the build to the testing team to perform testing. Build is in the testing phase; it may have already undergone testing or not. Software testing team checks this build. If it consists of multiple bugs and if it does not meet the requirements, then the software testing team rejects that build. Builds occur prior to the release, and they are generated more frequently.

Release

The release is the final application after completing development and testing. After testing the build, the testing team certifies that software and delivers it to the customer. It is possible for one release to have several builds. Therefore, it is the software delivered to the customer after completing the development and testing phases. Moreover, the release is based on builds, and it can have several build

A27.Difference between Build and Release?

Build refers to the standalone software artifact generated after converting the source code to an executable code that can be run on a computer. A release, on the other hand, is the distribution of the final version of an application. Thus, these definitions explain the fundamental difference between Build and Release.

A28. Explain the difference between Authorization and Authentication in Web testing.

Definition of Authorization

Authorization technique is used to determine the permissions that are granted to an authenticated user. In simple words, it checks whether the user is permitted to access the particular resources or not. Authorization occurs after authentication, where the user's identity is assured prior then the access list for the user is determined by looking up the entries stored in the tables and databases.

Example: For example, a user X wants to access a particular file from the server. The user will send a request to the server. The server will verify the user identity. Then, it finds the corresponding privileges the authenticated user have or whether he/she is allowed to access that particular file or not. In the following case, the access rights could include viewing, modifying or deleting the file if the user has authority to perform the following operations.

Definition of Authentication

Authentication mechanism determines the user's identity before revealing the sensitive information. It is very crucial for the system or interfaces where the user's priority is to protect the confidential information. In the process, the user makes a provable claim about individual identity (his or her) or an entity's identity.

The credentials or claim could be a username, password, fingerprint etc. The authentication and non-repudiation, kind of issues are handled in the application layer. The inefficient authentication mechanism could significantly affect the availability of the service.

Example: For example, there is a sender A sending an electronic document to the receiver B over the internet. How does the system will identify that the sender A has sent a message dedicated to the receiver B. An intruder C may intercept, modify and replay the document in order trick or steal the information this type of attack is called **fabrication**.

A29. What are the common problems faced in Web testing?

Web testing is a software testing practice to test website or application

for potential bugs. It's a complete testing of web-based applications before making live.

A web-based system needs to be checked completely from end-to-end before it goes live for end users.

By performing website testing, an organization can make sure that the web-based system is functioning properly and can be accepted by real-time users.

Web Testing Checklists

1) Functionality Testing
2) Usability testing

3) Interface testing
4) Compatibility testing
5) Performance testing
6) Security testing

Functionality Testing

Test for – all the links in web pages, database connection, forms used for submitting or getting information from the user in the web pages, Cookie testing, etc.

Usability testing is the process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction.

• Ease of learning
• Navigation
• Subjective user satisfaction
• General appearance

Interface Testing

In web testing, the server-side interface should be tested. This can be done by verifying that the communication is done properly. Compatibility of the server with software, hardware, network, and the database should be tested.

The main interfaces are:

- Web server and application server interface
- Application server and Database server interface.

The compatibility of your website is a very important testing aspect.

See which compatibility test to be executed:

- Browser compatibility
- Operating system compatibility
- Mobile browsing

Performance Testing

The web application should sustain a heavy load. Web performance testing should include:

- Web Load Testing
- Web Stress Testing

Test application performance at different internet connection speeds.

Security testing

The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities.
 It also helps in detecting all possible security risks in the system and help developers in fixing these problems through coding.

A30. What are the test case formats in case of a Static website and Dynamic website?

A simple static website will display the same content for all visitors who are visiting the website at different times. It is also known as an informational website. On a static website, only developers can make changes that too in code only. This type of website will not have any major functionalities and it purely depends on UI design.

Testing a simple static website is very easy, you have to consider only a few things while testing

It is the type where the user can update and change their website content regularly. From here I am going to use the word "web application testing" instead of dynamic website testing. The web application is a *combination of front-end and back-end programming*.

The front-end will be HTML and CSS whereas back-end uses programming languages like PHP, Javascript, and ASP etc. With this backend, user/client can add or change the content on the website.

Testing a web application is not easy than testing a static website but not much difficult than testing an e-commerce website. Functionality testing is the most important thing to be performed while testing a web application. The web application may contain much-complicated functionality so tester needs to be very careful while testing.

There are two different types of web applications are there, one is no action will be carried out by the user in front-end (i.e. only back-end changes will reflect in front-end) the other is end-user will work in front-end itself (**for example** login, signup, newsletter subscription, and other similar actions). So testing should be done according to it.

Points to Remember:

The points which I mentioned in static website testing are to be included while testing a web application also. In addition to that, the following things are to be noted.

- In GUI section, ***tooltip is compulsory*** for all fields and buttons, field alignment (spacing) should be done properly, disabled field/ buttons should be greyed out, fields/ buttons should be in standard format as in SRS, error message should be displayed if something goes wrong, pop-up message should only display at the center of the web page, drop-down menu should not be truncated.

Tab shortcut key should work in all fields and more.

- In functionality section, if your web application is having login or sign up functionality then check the ***mandatory field validation***, form validation (i.e. number fields should accept only numbers, not alphabets), character restriction on fields (i.e. only these many characters can be entered).

Special characters and negative numbers restriction on fields, testing the email functionality, testing the document upload (i.e. only ***specified document type can be uploaded***), timeout functionality, sorting functionality, javascript is working on compatible browsers etc should be tested.

- When coming to back-end functionality section, test image uploading for broken images, text entering in the fields is working or not. Back-end update should ***reflect on front-end***, ***database testing*** (i.e. whether you can add new fields or deleting unwanted fields) all these things are to be performed.

Performance is not much necessary for a web application (dynamic website) since it has very less content. If you need you can do with the tools with which you are familiar. Pick-up some standard online performance tool, if you want to do simple performance testing.