TOPSTECHNOLOGIES
Training | Outsourcing | Placement | Study Abroad

Diploma in Software Manual Testing

AN ASSIGNMENT ON

Course Title: Software Manual Testing

Module -2 (Manual Testing)

Submitted To:

RAKESH KHARVA

(Technical Trainer)

Tops Technologies Pvt. Ltd.

Submitted By:

KACHHADIYA DHARA

Date of submission: 02 June, 2023

Module–2(Manual Testing)

- What is Exploratory Testing?
- What is traceability matrix?
- What is Boundary value testing?
- What is Equivalence partitioning testing?
- What is Integration testing?
- What determines the level of risk?
- What is Alpha testing?
- What is beta testing?
- What is component testing?
- What is functional system testing?
- What is Non-Functional Testing?
- What is GUI Testing?
- What is Adhoc testing?
- What is load testing?
- What is stress Testing?
- What is white box testing and list the types of white box testing?
- What is black box testing? What are the different black box testing techniques?

- Mention what are the categories of defects?
- Mention what big bang testing is?
- What is the purpose of exit criteria?
- When should "Regression Testing" be performed?
- What is 7 key principles? Explain in detail?
- Difference between QA v/s QC v/s Tester
- Difference between Smoke and Sanity?
- Difference between verification and Validation
- Explain types of Performance testing.
- What is Error, Defect, Bug and failure?
- Difference between Priority and Severity
- What is Bug Life Cycle?
- Explain the difference between Functional testing and Nonfunctional testing

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

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

.• What is priority?

- What is severity?
- Bug categories are…
- Advantage of Bugzila

.• Difference between priority and severity

- What are the different Methodologies in Agile Development Model?

- Explain the difference between Authorization and Authentication in Web testing what is the common problem faced in Web testing?
- To create HLR & Test Case of Web Based (What's App web, Instagram)
    1. WhatsApp Web : https://web.whatsapp.com/ Write a scenario of only Whatsapp chat messages Write a Scenario of Pen
- Write a Scenario of Pen Stand
- Write a Scenario of Door
- Write a Scenario of ATM
- When to used Usability Testing?
- What is the procedure for GUI Testing?
- Write a scenario of Microwave Owen
- Write a scenario of Coffee vending Machine
- Write a Scenario of Whatsapp payment

What is Exploratory Testing?

Exploratory Testing is a type of software testing where Test cases are not created in advance but tester's check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a "thinking" activity.

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

Exploratory Testing Techniques

- Is not random testing but it is ad-hoc testing with a purpose of find bugs
- Is structured and rigorous
- Is cognitively (thinking) structured as compared to the procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
- Is highly teachable and manageable
- It is not a technique but it is an approach. What actions you perform next is governed by what you are
- doing currently

Pros and Cons of Exploratory Testing

Advantages

 This testing is useful when requirement documents are not available or partially available

 It involves Investigation process which helps find more bugs than normal testing-

 Uncover bugs which are normally ignored by other testing techniques

 Helps to expand the imagination of testers by executing more and more test cases which finally improves productivity as well

 This testing drill down to the smallest part of an application and covers all the requirements

 This testing covers all the types of testing and it covers various scenarios and cases

 Encourages creativity and intuition

 Generation of new ideas during test execution

Disadvantages:

- This testing purely depends on the tester skills
- Limited by domain knowledge of the tester
- Not suitable for Long execution time

When use exploratory testing?

- The testing team has experienced testers
- Early iteration is required
- There is a critical application
- New testers entered into the

What is traceability matrix?

Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability. Traceability can be horizontal through all the test documentation for a Given test level (e.g. system testing, from test conditions through test cases to test scripts) or it can be vertical through the layers of Development documentation (e.g. from requirements to components).

How to Create a RTM

You can create a requirements test matrix (RTM) in Microsoft Excel. Or you can use specialized tools to accelerate the process.

There are three basic steps — no matter which tool you use.

1. Define your goals.
2. Establish your artifacts (and their relationships).
3. Fill in the traceability matrix

Types Of Traceability Matrix

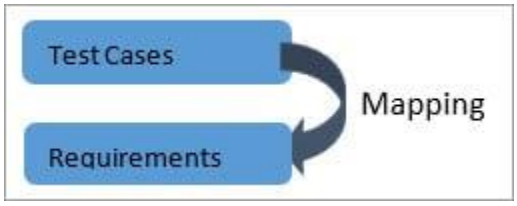
Forward Traceability:

In 'Forward Traceability' Requirements to the Test cases. It ensures that the project progresses as per the desired direction and that every requirement is tested thoroughly.

Backward Traceability:

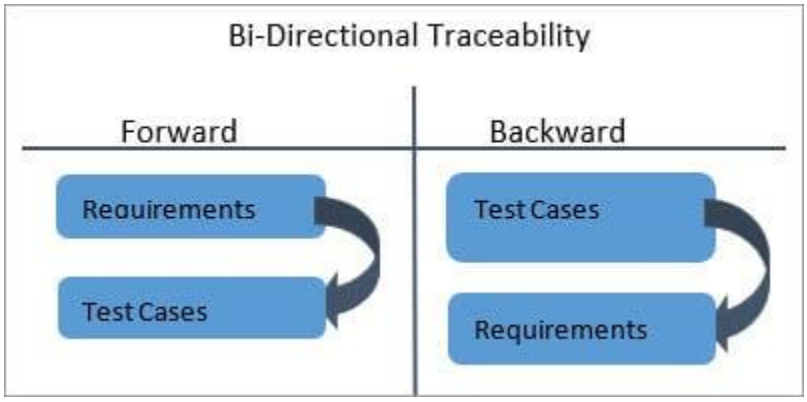
The Test Cases are mapped with the Requirements in 'Backward Traceability'. Its main purpose is to ensure that

the current product being developed is on the right track. It also helps to determine that no extra unspecified functionalities are added and thus the scope of the project is affected.



Bi-Directional Traceability:

(Forward + Backward): A Good Traceability matrix has references from test cases to requirements and vice versa (requirements to test cases). This is referred to as 'Bi-Directional' Traceability. It ensures that all the Test cases can be traced to requirements and each and every requirement specified has accurate and valid Test cases for them.



Examples Of RTM

1) Business Requirement

BR1: Writing emails option should be available.

Test Scenario (technical specification) for BR1

TS1: Compose mail option is provided

Test Cases:

Test Case 1 **(TS1.TC1)**: Compose mail option is enabled and works successfully.

Test Case 2 **(TS1.TC2)**: Compose mail option is disabled.

2) Defects

After executing the test cases if any defects are found that too can be listed and mapped with the business requirements, test scenarios and test cases.

For Example, If TS1.TC1 fails i.e. Compose mail option though enabled does not work properly then a defect can be logged. Suppose the defect ID auto-generated or manually assigned number is D01, then this can be mapped with BR1, TS1, and TS1.TC1 numbers.

- **Pros of Traceability Matrix**

  - Make obvious to the client that the software is being developed as per The requirements.
  - To make sure that all requirements included in the test cases
  - To make sure that developers are not creating features that no one has Requested
  - Easy to identify the missing functionalities.

- **Cons of Traceability Matrix**

- No traceability or Incomplete Traceability Results into:
- Poor or unknown test coverage, more defects found in production
- It will lead to miss some bugs in earlier test cycles which may arise in
- later test cycles. Then a lot of discussions arguments with other teams and managers before release

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

Boundary Value Analysis (B.V.A.)

Explain the Test Case of Boundary Value Testing

The idea regarding this is cleared from the above statements. Now, it is time to test what are the possible causes for Boundary value testing.

Test Case Number: 1

Let us assume a test case that takes the value of age from 21 to 65.

BOUNDARY VALUE TEST CASE		
INVALID TEST CASE **(Min Value – 1)**	**VALID TEST CASES** **(Min, +Min, Max, -Max)**	**INVALID TEST CASE** **(Max Value + 1)**

20	**21, 22, 65, 64**	**66**

From the above table, we can view the following inputs that are given.

- The minimum boundary value is given as 21.
- The maximum boundary value is given as 65.
- The valid inputs for testing purposes are 21, 22, 64 and 65.
- The invalid inputs for test cases are 20 and 66.

Test Case Scenarios

Input: Enter the value of age as 20 (21-1)

Output: Invalid

2. Input: Enter the value of age as 21

Output: Valid

3. Input: Enter the value of age as 22 (21+1)

Output: Valid

4. Input: Enter the value of age as 65

Output: Valid

5. Input: Enter the value of age as 64 (65-1)

Output: Valid

6. Input: Enter the value of age as 66 (65+1)

Output: Invalid

What is Equivalence partitioning testing?

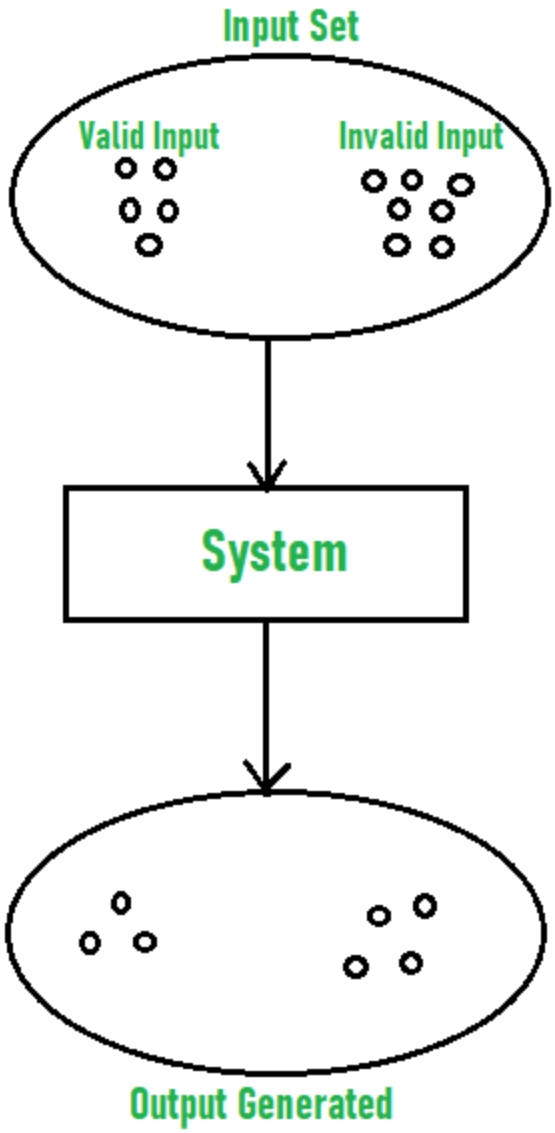
Equivalence Partitioning Method:

is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

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

Guidelines for Equivalence Partitioning:

- If the range condition is given as an input, then one valid and two invalid equivalence classes are defined.
- If a specific value is given as input, then one valid and two invalid equivalence classes are defined.
- If a member of set is given as an input, then one valid and one invalid equivalence class is defined.
- If Boolean no. is given as an input condition, then one valid and one invalid equivalence class is defined.


Input Set
Valid Input
Invalid Input
System
Output Generated

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.

There are 2 levels of Integration Testing

Component Integration Testing

System Integration Testing

Component Integration Testing

Component Integration Testing: Testing performed to expose defects in the Interfaces and interaction between integrated components Usually formal (records of test design and execution are kept)

All individual components should be integration tested prior to system testing It tests the interactions between software components and is done

After component testing. The following testing techniques are appropriate for Integration Testing:

Functional Testing using Black Box Testing techniques against the

Interfacing requirements for the component under test

Non-functional Testing (where appropriate, for performance or reliability

Testing of the component interfaces, for example)

System Integration Testing

It tests the interactions between different systems and may be done after system testing. It verifies the proper execution of software components and proper interfacing between components within the solution.

The objective of SIT Testing is to validate that all software module Dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.

Integration Testing Methods

During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. For example, whether the cap fits into the body or not. Any of Black Box Testing, White Box Testing, and Gray Box Testing methods can be used. Normally, the method depends on your definition of 'unit'.

- **There is two types methods of Integration Testing:**

Big Bang Integration Testing

Incremental Integration Testing

Top down Approach

Bottom up Approach

When is Integration Testing performed?

Integration Testing is performed after Unit Testing and before System Testing.

Who performs Integration Testing?

Either Developers themselves or independent Testers perform Integration Testing.

Big Bang Integration Testing

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.

Here all component are integrated together at once, and then Tested.

Big Bang Integration Testing

- **Advantages:**
- Convenient for small systems.
- **Disadvantages:**
- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
- Since the integration testing can commence only after “all” the modules are designed, testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority

Top down Approach

Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu).

Components or systems are substituted by stubs. In Top to down approach, testing takes place from top to down following the control flow of the software system. Takes help of stubs for testing.

Top down Approach

- **Advantages:**
- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.
- **Disadvantages:**
  - Needs many Stubs.
  - Modules at lower level are tested inadequately.

Bottom up Approach

Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing Bottom Up Approach

- **Advantages:**
- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach
- **Disadvantages:**
  - Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
  - Early prototype is not possible

Stub and Driver Approach

Stubs and Drivers are the dummy programs in Integration testing used to facilitate the software testing activity. These programs act as substitutes for the missing models in the testing. They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.

Stub: Is called by the Module under Test.

Driver: Calls the Module to be tested.

Continuous Integration Approach

Continuous Integration is a software development method where team members integrate their work at least once a day.

In this method, every integration is checked by an automated build to detect, errors. This concept was first introduced over two decades ago to avoid “integration hell,” which happens when integration is put off till the end of a project. In Continuous Integration after a code commit, the software is built and tested immediately.

In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If the deployment is a success, the code is pushed to Production.

This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

- **Entry and Exit 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

❖ **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. Limitations
- Any condition not specified in integration tests, apart from the confirmation of the execution of the design items is usually not tested.

What determines the level of risk?

As Risk is determined by a combination of Probability and Severity, the main area of the Matrix reveals the Risk Levels. The levels are Low, Medium, High, and Extremely High. To have a low level of risk, we must have a somewhat limited probability and level of severity.

1. Step 1: Identify the Risk. The initial step in the risk management process is to identify the risks that the business is exposed to in its operating environment. ...
2. Step 2: Analyze the Risk.
3. Step 3: Evaluate the Risk or Risk Assessment.
4. Step 4: Treat the Risk.
5. Step 5: Monitor and Review the Risk.

What is Alpha Testing?

- It 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.

What is beta testing?

- It 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.
- Beta Testing is always performed at the time when software product and project are marketed.
- It is always performed at the user's premises in the absence of the development team.
- Beta testing can be considered "pre-release" testing.
- Pilot Testing is testing to product on real world as well as collect data on the use of product in the classroom.

What is component testing?

Component testing is a form of closed-box testing, meaning that the test evaluates the behavior of the program without considering the details of the underlying code. Component testing is done on the section of code in its entirety, after the development has been completed.

Component testing takes longer to conduct than unit testing, because a component is made up of multiple units of code. Although it can be time-consuming, it is still very much necessary. Sometimes the individual units work on their own, but start having problems when you use them together.

Component testing examines use cases, so it could be considered a form of **end-to-end (E2E) testing**. End-to-end testing and component testing replicate real-life scenarios and test the system against those scenarios from a user's perspective.

Implementing component testing is beneficial because it can reveal bugs and errors that users might encounter. It is always better to catch and fix these errors before users know about them.

There are two types of component testing: component testing in small and component testing in large.

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**

1. Requirement Based Functional Testing
2. Process Based Testing

Functional System Testing Functionality as below:

- Accuracy Provision of right or agreed results or effect
- Interoperability Ability to interact with specified systems
- Compliance Adhere to applicable standards, conventions, regulations Or laws
- Auditability Ability to provide adequate and accurate audit data
- Suitability Presence and appropriateness of functions for specified

- **Requirement Based Testing**

  - Testing against requirements and specifications
  - Test procedures and cases derived from
  - Detailed user requirements
    - System requirements functional specification
    - User documentation/instructions
    - High level System design
  - Starts by using the most appropriate black-box testing technique May support this with white-box techniques (e.g. menu structures, web page navigation)
  - Risk based approach

Business Process Based Testing:

- **Test procedures and cases derived from**:

- Expected user profiles
- Business scenarios
- Use cases

- Testing should reflect the business environment and processes in which the system will operate.
- Therefore, test cases should be based on real business processes.

Functional Testing:

Functional 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. '**Function'** – what the system does Functional testing verifies that each function of the software Application operates in conformance with the requirement Specification.

Functional Testing Examples

Web Based Testing:

Are you able to login to a system after entering correct credentials?

Does your payment gateway prompt an error message when you enter incorrect card number?

Does your "add a customer" screen adds a customer to your records' successfully?

Desktop Based Testing:

Verifies Installation testing, Check for broken lines,

Testing with different client accounts, Theme change and Print,

Check for broken links. Warning messages. Resolution change effect on the application

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

It is the testing of “how” the system works. Non-functional testing may Be performed at all test levels.

The term non-functional testing describes the tests required to Measure characteristics of systems and software that can be Quantified on a varying scale, such as response times for performance Testing.

To address this issue, performance testing is carried out to check &Fine tune system response times. The goal of performance testing is

To reduce response time to an acceptable level Hence load testing is carried out to check systems performance at Different loads i.e. number of users accessing the system

Types of Nonfunctional testing are

· Performance Testing

· Load Testing

· Volume Testing

· Stress Testing

· Security Testing

· Installation Testing

· Penetration Testing

· Compatibility Testing

· Migration Testing

Non - Functional Testing Examples

Web Based Testing:

Identify the software processes that directly influence the overall performance of the system.

In website number of user/customer will increase, how the website will handled to every customer/user.

Desktop Based Testing:

Numerous other such GUI test cases, the desktop application tester must view

Guarantee that error messages are instructive and helpful for the client Memory, and different other issues

What is GUI Testing?

Graphical User Interface:

A graphics-based operating system interface that uses icons, menus and a mouse (to click on the icon or pull down the menus) to manage interaction with the system. Developed by Xerox, the GUI was popularized by the Apple Macintosh in the 1980s.

It is a friendly visual environment that allows the user to perform any action without having to have programming knowledge. An example of the GUI are the Windows, Mac os or Android environments, thanks to which commands can be sent through gestures or mouse movements, without the need to enter any code.

- **Approach of GUI Testing**

MANUAL BASED TESTING:

Under this approach, graphical screens are checked manually by tester in conformance with the requirements stated in business requirement document.

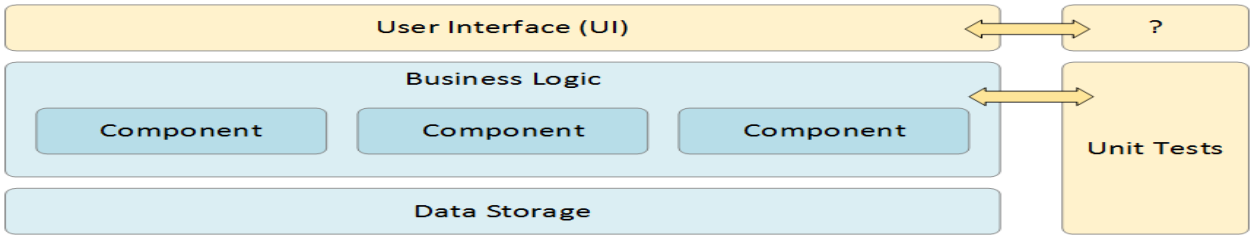
RECORD AND REPLAY:

GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the

Recorded test steps are executed on the Application under Test. Example of **such tools - QTP.**

MODEL BASED TESTING:

A model is a graphical description sof system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.



GUI Testing Examples:

Web Based Testing & Desktop Based Testing:

The scrollbar should be enabled only when necessary.Font size, style, and color for headline, description text, labels, infield data, and grid info should be standard as specified in SRS.

The description text box should be multi-lined.

Enough space should be provided between field labels, columns, rows, error messages, etc.

- **Mobile Based Testing:**

If mobile is in every orientation mode so display image, video properly.

Every app will display in responsive type.

Alignment should be applied properly of every field.

- **Game Based Testing :**

Game infra designs will showing properly

Game points or score will display proper with its background color.

Game sound manage with its background effect

What is Adhoc testing?

Adhoc Testing also known as “Error Guessing” The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.

Using experience

to postulate errors.

Use Error Guessing to Complement Test Design Techniques.

Types of Adhoc Testing

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

1. **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.

2. **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.

3. **Monkey Testing:**

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

What is load testing?

Load testing –

It's 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.

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.

Why Load Testing?

Load testing gives confidence in the system & its reliability and performance.

Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.

Goals of Load Testing

Loading testing identifies the following problems before moving the application to market or Production:

- Response time for each transaction
- Performance of System components under various loads
- Performance of Database components under different loads

Pre-Requisites for Load Testing:

The chief metric for load testing is response time. Before you begin load testing, you must determine -

Whether the response time is already measured and compared – Quantitative

Whether the response time is applicable to the business process – Relevant

Load Testing

Tools Load runner

Web Load

Astra Load Test

Review's Web Load

Pros and Cons of Load Testing

- **Pros:**
  - Performance bottlenecks identification before production
  - Improves the scalability of the system
  - Minimize risk related to system down time

- Reduced costs of failure Increase customer satisfaction

❖ **Cons:**

- Need programming knowledge to use load testing tools.
- Tools can be expensive as pricing depends on the number of virtual users supported.

What is StressTesting?

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 done to make sure that the system would not crash under crunch situations.

Stress testing is also known as endurance testing.

Need For Stress Testing

During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.

When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.

Goal of Stress Testing

Testers should not lose this security related data while doing stress testing.

The main purpose of stress testing is to make sure that the system recovers after failure which is called as recoverability.

Types of Stress Testing

- Application Stress Testing
- Transactional StressTesting
- Systemic Stress Testing
- Exploratory Stress Testing

Stress testing Tools

Stress Testing, Neo Load, App Perfect

Metrics for Stress Testing:

Measuring Scalability & Performance

- Pages per Second: Measures how many pages have been requested / Second,
- Throughput: Basic Metric – Response data size/Second,
- Rounds: Number of times test scenarios has been planned Versus Number of times client has executed

Application Response

- Hit time: Average time to retrieve an image or a page,
- Time to the first byte: Time taken to return the first byte of data or information,
- Page Time: Time taken to retrieve all the information in a page

Failures

- Failed Connections: Number of failed connections refused by the client,
- Failed Rounds: Number of rounds it gets failed,
- Failed Hits: Number of failed attempts done by the system

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.

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.

White Box Testing Examples

Desktop Based Testing:

When we debug the code when we writing

Mobile Based Testing:

The Android SDK and related plug-in for Eclipse Android devices enabled for development and debugging, with appropriate USB drivers as necessary

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.

Through statement coverage we can identify the statements executed and where the code is not executed because of blockage.

The statement coverage covers only the true conditions.

The statement coverage can be calculated as shown below:

$$\text{Statement coverage} = \frac{\text{Numbers of statement exercises}}{\text{Total numbers of Statements}} * 100$$

Statement/Segment Coverage

ADVANTAGE:

- It verifies what the written code is expected to do and not to do
- It measures the quality of code written
- It checks the flow of different paths in the program and it also ensure that whether those path are tested or not.

DISADVANTAGE:

- It cannot test the false conditions.
- It does not report that whether the loop reaches its termination condition.
- It does not understand the logical operators.

Statement Coverage Example

Assume a software program has 70 lines of code. 40 statements are executed during the program run. Now we will calculate the Statement Coverage metric using the above formula.

Statement Counters based on the data:

Number of Covered statements = 40

Number of Missed statements = 30

Total number of statements = 70

Statement Coverage = (Covered statements/Total Statements)* 100;

SC = (40/70) * 100

= 57.14 %

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.

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

$$\text{Decision coverage} = \frac{\text{Numbers of coverage outcomes exercised}}{\text{Total Numbers of Decision outcomes}} * 100$$

A decision is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL, JUMP, GO TO), or a CASE statement, where there are two or more outcomes from the statement.

ADVANTAGES:

- To validate that all the branches in the code are reached
- To ensure that no branches lead to any abnormality of the program's

operation

- It eliminate problems that occur with statement coverage testing

DISADVANTAGES:

- This metric ignores branches within Boolean expressions which occur

Due to short-circuit operators.

NOTE:

Branch Coverage Testing >= Statement Coverage Testing

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

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.

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.

Advantages

- Well suited and efficient for large code segments. Code Access not required.
- Clearly separates user's perspective from the developer's
- Perspective through visibly defined roles.

Disadvantage

- Limited Coverage since only a selected number of test scenarios are actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Black Box Testing Examples: Desktop Based Testing :
- Resolution change effect on the application Installation Testing (Upgrade/Downgrade

Techniques of Black Box testing there are four specification-based or black-box techniques:

a) Equivalence partitioning
b) Boundary value analysis
c) Decision tables
d) State transition testing
e) Use-case Testing
f) Other Black Box Testing
g) Syntax or Pattern Testing

a)Equivalence Partitioning(E.P.)

The numbers fall into a partition where each would have the same, or equivalent, result i.e. an Equivalence Partition (EP) or Equivalence Class EP says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that:

If one value finds a bug, the others probably will too If one doesn't find a bug, the others probably won't either.

Boundary Value Analysis (B.V.A.)

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.

Decision Table:

The other two specification-based software testing techniques, decision tables and state transition testing are more focused on business logic or business rules.

A decision table is a good way to deal with combinations of things (e.g. inputs).

Table based technique where:

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

State Transaction Testing:

State transition testing is used where some aspect of the system can be described in what is called a 'finite state machine'. This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the 'machine'. This is the model on which the system and the tests are based.

Any system where you get a different output for the same input, depending on what has happened before, is a finite state system.

A finite state system is often shown as a state diagram.(next slide given example)

Switch Coverage of State Transaction:

Switch Coverage is a method of determining the number tests based on the number of "hops" between transitions. Sometimes known as Chow

These hops can be used to determine the VALID tests

What is the purpose of exit criteria?

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution. Exit criterion should be part of test plan and decided in the planning stage.
What is the purpose of exit criteria in Istqb?

The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing.

The commonly considered exit criteria for terminating or concluding the process of testing are:

- Deadlines meet or budget depleted.
- Execution and updating of all test cases.
- Desired and sufficient coverage of the requirements and functionalities under the test.
- All the identified defects are corrected and closed.

When should "Regression Testing" be performed?

Regression testing should be carried out:

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

Difference between QA v/s QC v/s Tester

Quality Assurance	Quality Control	Tester
1. Activities which ensure the implementation of processes, procedures and standards in con text 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.
2 .Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3. Process oriented activities.	Product oriented activities.	Product oriented activities.
Preventive activities.	It is a corrective process.	It is a preventive process.
5.It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

Difference between Smoke and Sanity:

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

Difference between verification and Validation:

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

Explain the difference between Functional testing and Nonfunctional testing

Functional Testing	Non Functional Testing
1.It is executed to analyze the functionality of components of an application as per the client requirements	It executed to performance, reliability, scalability, and other non-functional aspects of application.
2.It is executed in the early stage of development	It is generally performed after functional testing
3.can be performed both manually and with automation tools	Requires automation tools for effective testing
4.Focuses on user requirements	Focuses on user expectations
5.Determines what the products is capable of	Determines how effectively the product works
6.Business requirements are the inputs of functional testing	Parameters like speed, scalability are the inputs of non- functional
7.Examples of Functional Testing Unit Testing, White box Testing, smoke Testing, Sanity Testing, Usability Testing, Regression Testing	Examples of Non-Functional Testing performance Testing, load Testing, Stress Testing, Security Testing, Installation Testing, Cross Browser Compatibility Testing.

Explain types of Performance testing

Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.

Types of Performance Testing

a) Load testing
b) Stress testing
c) Endurance testing
d) Spike testing
e) Volume testing
f) Scalability testing

Long Load time –Load time is normally the initial time it takes an application to start.

This should generally be kept to a minimum.

Poor response time –Response time is the time it takes from when a user inputs data into the application until the application outputs a response to that input.

Generally this should be very quick.

Poor scalability –A software product suffers from poor scalability when it cannot handle the expected number of users or when it does not accommodate a wide enough range of users.

What is Error, Defect, Bug and failure?

"A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure"

Error:

A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.

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 annul intended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

Fault:

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

Mention what are the categories of defects?

Quality control professionals typically classify quality defects into three main categories: **minor**, **major** and **critical.** The nature and severity of a defect determines in which of the three categories it belongs.

- **Minor defects**

Minor defects are usually small, insignificant issues that **don't affect the function or form of the item**. In most cases, the customer wouldn't even notice a minor defect on a product. And the customer wouldn't likely return an item due to a minor defect alone.

- **Major defects**

Major defects are more serious than minor defects. A product with a major defect departs significantly from the buyer's product specifications. Major defects are those which could adversely **affect the function, performance or appearance** of a product.

- **Critical defects**

Critical defects are the most serious of the three defect types. Critical defects render an item completely unusable and/or could cause harm to the user or someone in the vicinity of the product.

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."

Defect Stages

New: When a new defect is logged and posted for the first time. It is assigned a status as NEW.

Assigned: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team

Open: The developer starts analyzing and works on the defect fix

Fixed: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."

Pending retest: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is "pending retest."

Retest: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test.

Verified: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."

Reopen: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.

Closed: If the bug is no longer exists then tester assigns the status "Closed."

Duplicate: If the defect is repeated twice or the defect corresponds to the same

Rejected: If the developer feels the defect is not a genuine defect then it changes concept of the bug, the status is changed to “duplicate.” the defect to “rejected.”

Deferred: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs

Not a bug: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

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.

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

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

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

Test Scenario	Test cases	Test Script
It is any functionality that can be change	Is a set of action executed to verify particular features or functionality	Is a set of instruction to test to any 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 then 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 result 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

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

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.

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.

What is priority?

Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? 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

What is severity?

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, Major (High), Moderate (Medium), Minor (Low), Cosmetic.

Advantage of Bug zilla:

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

Bug zilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Center, Test link etc.

- 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.
- Integrated email capabilities.
- Detailed permissions system.
- Optimized database structure to enhance performance.
- Robust security.
- Powerful query tool.
- Ideal for small projects.

What are the different Methodologies in Agile Development Model?

What is Agile?

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating.

Agile is based on the iterative-incremental model. In an incremental model, we create the system in increments, where each increment is developed and tested individually.

Agile four manifesto:

- Individuals and interactions, Over processes and tools
- Working software, Over comprehensive documentation
- Customer collaboration, Over contract negotiation
- Responding to change, over following a plan

Agile Principles:

- Customer satisfaction through early and continuous software delivery Accommodate changing requirements throughout the development process Frequent delivery of working software
- Collaboration between the business stakeholders and developers throughout the Project
- Support, trust, and motivate the people involved
- Enable face-to-face interactions
- Working software is the primary measure of progress
- Agile processes to support a consistent development pace
- Attention to technical detail and design enhances agility
- Simplicity
- Self-organizing teams encourage great architectures, requirements, and designs
- Regular reflections on how to become more effective

Scrum

Scrum:

SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment. 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.

Sprint: Sprint is a time-boxed period in which the scrum team needs to finish the set amount of work. Each sprint has a specified timeline, i.e., 2 weeks to 1 month. The scrum team agrees with this timeline during the sprint planning meeting.

Scrum Roles:

Product Owner:

There is a client who wants to develop his software, so he approaches to the company who can develop his software. What does the company do? The Company assigns a role, i.e., Product Owner. Product Owner is the person who communicates with the clients understands their requirements. Product Owner is the responsible person from the company for software development.

Scrum Master

During the sprint, Agile says that the team should meet together once daily. When the team is following scrum means that they are conducting meetings daily for 10 to 15 minutes. This meeting is known as a

scrum meeting. Scrum Master is the person who handles the scrum meeting.

Scrum Team:

The team comprises of persons who work on the project. It can be developers, testers or designers. When we talk about Agile or Scrum then we talk about the team, we do not talk about developers, or testers as an individual. Agile says that developers can work as a tester or testers can work as a developer when the need arises.

Scrum Board

Product Backlog: Product Backlog is a set of activities that need to be done to develop the software.

Sprint Backlog: Sprint Backlog is a backlog that has taken some of the activities from the product backlog which needs to be completed within this sprint.

Scrum Board: Scrum Board is a board that shows the status of all the activities that need to be done within this sprint. Scrum Board consists of four status:

Open: The 'Open' status means that the tasks which are available in 'Open' are not yet started.

In progress: The 'In progress' status means the developers completed their tasks.

Testing: The 'testing' means that the task is in a testing phase.

Closed: The 'closed' means the task has been completed.

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

Simply put, authentication is the process of verifying who someone is, whereas authorization is the process of verifying what specific applications, files, and data a user has access to. The situation is like that of an airline that needs to determine which people can come on board. The first step is to confirm the identity of a passenger to make sure they are who they say they are. Once a passenger's identity has been determined, the second step is verifying any special services the passenger has access to, whether it's flying first-class or visiting the VIP lounge.

Authentication	Authorization
Authentication verifies who the user is.	Authorization determines what resources user can access.
Authentication work through passwords, one time pin, biometric information, and other information provided or entered by the user.	Authorization work through settings that are implemented and maintained by the organization
Authentication is the first step of a good indent and access management process.	Authorization always take place after Authorization
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or not changeable by the user.
Example: By verifying their identity, employees can again access to a human resource (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 level of data based on the permissions set by the organization.

What are the common problems faced in Web testing?

1. **Insufficient testing for browser compatibility:**

Poor browser compatibility is an obvious turn-off – if users can't see your content clearly and transactional features don't work well, they'll move on.

2. **Failing to conduct thorough functional testing across mobile**

Many websites and apps suffer from functional issues on different mobile devices, because they haven't been tested on all the devices their customers are using.

3. **Failing to conduct thorough functional testing across desktop:**

Yes, it's the same problem but in reverse. Some companies are so carried away with the news that mobile is predominant that they drop the ball on desktop testing. Responsive mobile design has become the default for many development teams. But although desktop usage may be in the minority, crucial segments of your target audience may over-index on desktop. Some users prefer browsing and reading on a larger screen.

4. **Poor data security:**

Data security is a constant challenge for organizations to deal with. Hackers regularly probe company security for weaknesses. New viruses and malware are constantly being developed to exploit unlatched software vulnerabilities. Paying insufficient attention to security in your QA testing is a big risk.

5. **Failing to provide an intuitive experience:**

While it's important that your website's visual design is up-to-date with the latest trends, a Hub Spot report reveals why a large number of consumers view easy navigation as the most important factor in website design.

Common problems that can negatively affect website navigation are:

- Broken links
- Slow loading speeds
- Complicated menu structures

6. Not performing testing frequently enough:

Tick off your pre-launch web testing and you can tick off any future problems right? Of course, this is not the case. You need to repeat website testing regularly to identify any conversion blockers that are costing you revenue, as well as any potential issues that appear over time before they become critical.

7. Leaving digital accessibility to the last minute:

If you don't take digital accessibility seriously when you're creating websites and apps, you're missing out on a share of the £249 billion spent annually by customers with disabilities and impairments that can affect the way they engage with digital products.

8. Releasing new features breaks the existing live system:

After any deployment or release, there's a risk of unintended consequences. Changes can introduce defects or alter the behavior of a flow or specific function.

9. Localization and the global experience:

Launching digital products to a worldwide audience, rather than one geography or in a single language, brings a whole lot more complexity. Customers and users of your software or website may interact in a language different from the one it was written in. Automated translations have come a long way, but they still don't pick up on the subtleties of language.

10. The most common bugs:

Bugs in code and errors in set-up can ruin website or app user experiences. As well as getting in the way of smooth navigation and stopping users completing transactions, they give an impression of carelessness or a lack of attention to detail.

What is the procedure for GUI Testing?

Manual Testing

This approach involves human tester, where each screen is manually checked to validate each functionality by creating and executing test cases. It is a useful approach when part of UI or a feature is ready, the probability of defects is more at the initial stage, and human intervention is required.

Record and Replay Testing

GUI record and replay tools are used to test applications for their user interface. Using such tools, testers run an application and record the user interaction with the app. A script runs to track and save the user actions, including cursor movements, which can be replayed several times to find the issues in the interface.

Model-based testing

In this type of GUI testing, a model is created to understand and evaluate the system's behavior. This approach is useful in creating accurate test cases using system requirements. It is a structured, thorough, measurable form of testing.

There are three essential aspects of model-based GUI testing:

- Automatically generated test cases from the model
- Manually derived test cases from the model
- Model and requirements coverage metrics

Things to consider for model-based testing:

- Create the model
- Determine the information as inputs in the system
- Verifying the expected output
- Execute tests
- Checking and validating actual vs. expected
- Take further action on the model