**Software Testing Assignment:2**

• What is Exploratory Testing?

**Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers 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 is a concurrent process where

• Test design, execution and logging happen simultaneously

• Testing is often not recorded

• Makes use of experience, heuristics and test patterns

• Testing is based on a test charter that may include

• Scope of the testing (in and out)

• The focus of exploratory testing is more on testing as a “thinking” activity.

• A brief description of how tests will be performed

• Expected problems

• Is carried out in time boxed intervals • More structured than Error guessing.

• What is traceability matrix?

-To protect against changes you should be able to trace back from every system component to the original requirement that caused its presence. A software process should help you keeping the virtual table up-to-date. Simple technique may be quite valuable (naming convention).

Traceability Matrix (also known as Requirement Traceability Matrix - RTM).

• What is Boundary value testing?

-Testing software to make sure that it functions properly before its release can help organizations improve the overall quality of their software and reduce the prevalence of larger errors. Boundary testing is one software testing technique organizations often use, and it involves evaluating data based on its boundary values, or its two opposite ends, such as its minimums and maximums. Learning more about this technique can help you check software errors more efficiently. In this article, we discuss what boundary testing is, why it's important and how it works and list a few examples to demonstrate how to use it.

• What is Equivalence partitioning testing?

Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other. The equivalence partitions are derived from requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.

• What is Integration testing?

• 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:

• Bing 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.

• 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.

• What is Alpha testing?

First of all test newly developed hardware or software in a laboratory setting. When the first round of bugs has been fixed, the product goes into beta test with actual users. For custom software, the customer may be invited into the vendor's facilities for an alpha test to ensure the client's vision has been interpreted properly by the developer.

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.

• What is beta testing?

Test of new or revised hardware or software that is performed by users at their facilities under normal operating conditions. Beta testing follows alpha testing. Vendors of packaged software often offer their customers the opportunity of beta testing new releases.

• What is component testing?

- Unit testing in Extreme Programming involves the extensive use of testing frameworks. A unit test framework is used in order to create automated unit tests. Unit testing frameworks are not unique to extreme programming, but they are essential to it. Below we look at some of what extreme programming brings to the world of unit testing:

-Tests are written before the code

- Rely heavily on testing frameworks

- All classes in the applications are tested

- Quick and easy integration is made possible.

• What is functional system testing?

-Functional System Testing: A requirement that specifies a function thata system or system component must perform

- A Requirement may exist as a text document and/or a model

- There is two types of Test Approach

- Requirement Based Functional Testing

- Process Based Testing

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 techniques

May support this with white-box techniques (e.g., menu structures, web page navigation) Risk based approach.

• What is Non-Functional Testing?

Non-Functional Testing: Testing the attributes of a componentor system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability

- May be performed at all Test levels (not just Non Functional SystemsTesting)

- Measuring the characteristics of the system/software that can bequantified on a

varying scale- e.g. performance test scaling

- Non-functional testing includes, but is not limited to, performancetesting,

load testing, stress testing, usability testing, maintainabilitytesting, reliability

testing and portability testing.

- It is the testing of “how” the system works. Non-functional testing maybe

performed at all test levels.

- The term non-functional testing describes the tests required to measure

characteristicsofsystemsand softwarethat can bequantifiedonavarying

scale, such as response times for performancetesting.

- To address this issue, performance testing is carried out to check & fine tune

system response times. The goal of performance testing isto reduce response

time to an acceptable level

- Hence load testing is carried out to check systems performance atdifferent

loads i.e. numberof users accesing the system.

• What is GUI Testing?

* GUI means **Graphical User Interface (GUI)**

-The **Command-line interface**is used when we need to type text, and at the same time computer responds to that command.

-On the other hand, the **Graphical user interface** is used to interrelate along with the computer by using the images rather than text.

-We have some of the critical GUI elements that can be used for communication between the user and application.

• What is Adhoc testing?

- Adhoc testing is an informal testing type with an aim to break the system.

- It does not follow any test design techniques to create test cases.

- In fact is does not create test cases altogether!

- This testing is primarily performed if the knowledge of testers in the system under test is

very high.

-Testers randomly test the application without any test cases or any business requirement

document.

- Adhoc Testing does not follow any structured way of testing and it is randomly done on any

part of application.

- Main aim of this testing is to find defects by random checking.

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

- Error guessing can be done by the people having enough experience on the system to

“guess” the most likely source of errors.

• What is load testing?

In software testing, load testing is an integral part of performance testing under.

Load testing is testing where we check an application's performance by applying some load, which is either less than or equal to the desired load.

Here, load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.

Load testing will help to detect the maximum operating capacity of an application and any **blockages** or bottlenecks.

It governs how the software application performs while being accessed by several users at the same time.

The load testing is mainly used to test the **Client/Server's performance and applications that are web-based.**

In other words, we can say the **load testing** is used to find whether the organization used for compering the application is necessary or not, and the performance of the application is maintained when it is at the maximum of its user load.

Generally, load testing is used to signify how many concurrent users handle the application and the application's scale in terms of **hardware, network capacity** etc.

• What is stress Testing?

-Stress testing - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

-Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.

-It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.

- Stress Testing is done to make sure that the system would not crash under crunch situations.

- Stress testing is also known as endurance testing.

- Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding

capacity.

- Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.

- It also checks whether system demonstrates effective error management under extreme conditions.

- The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad.

• 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.

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.

-type of white box testing.

### 1. Path Testing

**2. Loop Testing**

**3. Conditional Testing**

**4. Unit Testing**

**5. Mutation Testing**

**6. Integration Testing**

**7. Penetration Testing**

**8. Testing based on Memory Perspective**

**9. Test Performance of the Program**

• What is black box testing? What are the different black box testing techniques?

Black box testing is a software testing approach in which the functionality of the SUT (Software Under Test) is tested without regard for its implementation specifics, internal route knowledge, or internal code structure.

* This type of testing is entirely dependent on the software's specs and needs.
* The outputs and inputs of the software system, rather than the program's underlying knowledge, are the focus of black-box testing.
* The "black box" is the system that undergoes this form of testing, and it may be any program, such as a database, website, or operating system.

The methodologies used during Black box testing for a software program are as follows.

### Boundary Value Analysis (BVA)

It is one of the most important and helpful Black box testing techniques for equivalence partitioning. BVA may be used to evaluate any program that has a boundary or extreme values.

Rather than concentrating on the range of input values, this approach is capable of discovering problems in the limitations of input values. Edge or extreme output values are likewise dealt with via boundary value analysis.

### Partitioning by Equivalence Class

Black box testing is a method for writing test cases that is frequently utilized. It may be beneficial for condensing a large number of potential inputs into a smaller number of more effective ones.

* It is accomplished by categorizing inputs into classes and assigning a value to each class.
* It's used when you need to do a lot of testing and you don't want your inputs to be redundant.

### Testing for State Transitions

This method generally examines a system's status, outputs, and inputs throughout a period of time.

* It tests for behavioral changes of a system in a certain state or another state while keeping the same inputs, depending on the kind of software being tested.
* The test cases for this approach are built by examining the order in which the inputs transition and state or events occur.
* The predicted output values and all states will be traversed over the whole collection of test cases.

• Mention what are the categories of defects?

A software bug arises when the expected result don't match with the actual results. It can also be error, flaw, failure, or fault in a computer program. Most bugs arise from mistakes and errors made by developers, architects

-Arithmetic Defects

-Logical Defects

-Syntax Defects

-Multithreading Defects

-Interface Defects

-Performance Defects

• Mention what bigbang testing is?

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

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

modulesare designed,testingteamwillhavelesstimefor

execution in the testingphase.

-Since all modules are tested at once, high risk critical modules

are not isolated and tested on priority. Peripheral modules which deal with userinterfaces are also not isolated and tested on priority.

• 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.

## **Examples of Exit Criteria:**

* Verify if All tests planned have been run.
* Verify if the level of requirement coverage has been met.
* Verify if there are NO Critical or high severity defects that are left outstanding.
* Verify if all high risk areas are completely tested.
* Verify if software development activities are completed within the projected cost.
* Verify if software development activities are completed within the projected timelines.

• When should "Regression Testing" be performed?

Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

- If the test is re-run and passes you cannot necessarily say the fault has been resolved because.

-You also need to ensure that the modifications have not caused unintended side-effects elsewhere and that the modified system still meets its requirements – Regression Testing

-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.

• What is 7 key principles? Explain in detail?

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy.

1)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.

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

2)Early Testing

-Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

-Testing activities should start as early as possible in the development life cycle

-These activities should be focused on defined objectives – outlined in the Test Strategy.

3)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

-In other words, most defects found during testing are usually confined to a small number of modules

-Similarly, most operational failures of a system are usually confined to a small number of modules.

## 4) Testing shows a presence of defects

Hence, testing principle states that – Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software 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.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

## 5) Absence of Error – fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user’s needs & requirements.

## 6) Early Testing

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

## 7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software’s are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

• Difference between QA v/s QC v/s Tester

|  |  |  |
| --- | --- | --- |
| QA | QC | Tester |
| Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| Process oriented activities | Product oriented activities. | Product oriented activities. |
| 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 testin |
| Preventive activities | It is a corrective process. | It is a preventive process. |
| It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

• Difference between Smoke and Sanity?

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of regression testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

• Difference between verification and Validation

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does *not* include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| It comes before validation. | It comes after verification. |
| Verification is about process, standard and guideline. | Validation is about the product. |

• Explain types of Performance testing.

Types of Performance Testing

1.Load testing

2.Stress testing

3.Endurance testing

4.Spike testing

5.Volume testing

6. Scalability testing.

### Load Testing

Load testing is sort of the simplest form of performance testing. You induce a normal or expected workload to a system under test and observe it. You can use load tests to determine general system behavior, latency and throughput. In general load tests are used to verify your quality criteria.

### Stress Testing

Stress testing is basically a load test, but we are applying a higher-than-expected workload and see how the system behaves under serious stress and when exceeding the design limits. You want to learn when your system breaks and how it starts to fail when being in a serious traffic situation.

**Endurance testing**

is done to make sure the software can handle the expected load over a long period of time.

**Volume testing**

Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.

**Spike testing**

tests the software’s reaction to sudden large spikes in the load generated by users.

### Scalability Testing

With scalability testing you are changing the perspective to answer the question: How effective can I grow? You can run a series of stress tests and gather data on how effective you really are.

Using stress tests in a series where you steadily increase the system’s resources, you can easily tell if your system can translate this into additional capacity.

• 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. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

• Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

• Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

• Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

• Difference between Priority and Severity

|  |  |
| --- | --- |
| **Priority** | **Severity** |
| Priority is a team that defines how fast we need to fix a defect | Severity is a team that denotes how severely a defect can affect the functionality of the software. |
| Priority is basically a parameter that decides the order in which we should fix need to fix a defect. | Severity is bacsically a parameter that denote the total impact of a given defect on any software. |
| Priority relates to the scheduling of defect to resolve them in software. | Severity reletes to the standards of quality. |
| The value of priority is subjective. | The value of Severity in objective. |
| The value of Priority change from time to time | The value of Severity changes continually from time to time |
| The product manager basically decides a defect’s priority level. | The testing engineer basically decides a defect’s severity level. |

• 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.

NEW

ASSIGNED

DUBLICATE REJECTED DEFFERED NOT A BUG

OPEN

FIXED

REOPENED

PENDING RETEST

REDSFSD

RETEST

VERIFIED

CLOSED

• Explain the difference between Functional testing and Non Functional testing

|  |  |
| --- | --- |
| **Functional Testing** | **Non-functional Testing** |
| It verifies the operations and actions of an application. | it verifies the behavior of an application. |
| It is based on requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behavior of the application. | It helps to improve the performance of the application. |
| It helps to improve the performance of the application. | It is hard to execute non-functional testing manually. |
| It tests what the product does. | It describes how the product does. |
| Functional testing is based on the business requirement. | Non-functional testing is based on the performance requirement. |

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

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

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

1)Test scenarios ;

The least detailed type of documentation is the test scenario. A test scenario is a description of an objective a user might face when using the program. An example might be “Test that the user can successfully log out by closing the program.” Typically, a test scenario will require testing in a few different ways to ensure the scenario has been satisfactorily covered. Just based on that light description, the tester might choose to close the program through the menu option, kill it through the task manager, turn the computer off, or see what happens when the program runs out of memory and crashes. Since test scenarios offer little information about how to complete the testing, they offer the maximum amount of flexibility to the tester responsible for them.

2)Test cases;

The second most detailed way of documenting testing work is to use test cases. Test cases describe a specific idea that is to be tested, without detailing the exact steps to be taken or data to be used. For example, a test case might say “Test that discount codes can be applied on top of a sale price.” This doesn’t mention how to apply the code or whether there are multiple ways to apply the code. The actual testing that will cover this test case may vary from time to time. Will the tester use a link to apply a discount, or enter a code, or have a customer service rep apply the discount, or will they feel compelled to test every way to add a discount that they can think of? Test cases give flexibility to the tester to decide exactly how they want to complete the test.

3)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.

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

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

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

• What is priority?

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

For example: If the company logo is incorrectly placed in the company's web page then the priority is high but it is of low severity.

* Low **-**This defect can be fixed after the critical ones are fixed.
* Medium **-**The defect should be resolved in the subsequent builds.
* High **-**The defect must be resolved immediately because the defect is affecting the application to a considerable extent and the relevant modules cannot be used until it's fixed.
* Urgent **-**The defect must be resolved immediately because the defect is affecting the application or the product severely and the product cannot be used until it has been fixed.

• What is severity?

The impact of the bug on the application is known as severity.  
It can be a **blocker, critical, major, and minor** for the bug.

• High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system.

• High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.

• High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

• Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report.

• Bug categories are…

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

• Advantage of Bugzila .

* Open source, free bug tracking tool.
* Automatic.
* Search option with advanced features.
* File/Modify Bugs By Email.
* Move Bugs Between Installs.
* Multiple  Methods.
* 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?

Developed in 2001, the agile software development methodologies were based on the most revered agile manifesto that laid down the principles and core practices. Before understanding the agile methodologies definition, it is pertinent to know what agile is.

Agile is a set of techniques followed by a team to administer a project or plan by dividing it into various stages with continuous collaboration with customers. There is constant monitoring at every phase of the software development of the project. The agile methodology’s advantages are that both the development plus testing actions are parallel and synchronized, unlike the conventional waterfall methodology.

### 1) **Kanban**

### 2) **Scrum**

### 3) **Extreme Programming (XP)**

### 4) **Crystal**

### 5) **Dynamic Systems Development Method (DSDM)**

### **6) Feature-Driven Development (FDD)**

### 7) Lean Software Development

### 8) Scaled Agile Framework (SAFe)

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

### 

|  |  |
| --- | --- |
| Authorization | Authentication |
| Authorization determines what resources a user can access. | Authentication verifies who the user is. |
| Authorization works through settings that are implemented and maintained by the organization. | Authentication works through password, one-time pins, biometric information, and other information provided or entered by the user. |
| Authorization always takes place after authentication. | Authentication is the first step of a good identity and access management process. |
| Authorization isn’t visible to or changeable by the user. | Authentication is visible to and partially changeable by the user. |
|  |  |

### • When to used Usablity Testing?

### Aesthetics and design are important. How well a product looks usually determines how well it works.

### • There are many software applications / websites, which miserably fail, once launched, due to following reasons –

### • Where do I click next?

### • Which page needs to be navigated?

### • Which Icon or Jargon represents what?

### • Error messages are not consistent or effectively displayed

### • Session time not sufficient.

### • Usability Testing identifies usability errors in the system early in development cycle and can save a product from failure.

### What is the procedure for GUI Testing?

### • Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.

### • Check you can execute the intended functionality of the application using the GUI

### • Check Error Messages are displayed correctly

### • Check for Clear demarcation of different sections on screen

### • Check Font used in application is readable

### • Check the alignment of the text is proper

### • Check the Color of the font and warning messages is aesthetically pleasing

### • Check that the images have good clarity

### • Check that the images are properly aligned

### • Check the positioning of GUI elements for different screen resolution.

### 