Manual Testing Class 2 Contents:  
  
Phase 1- Software Testing Concepts  
---------------------------------------------------------  
7 Principal of Software Testing   
Product & Project  
QA/Testing Activities  
Software Testing Types  
  
  
Phase 2 - Software Testing Project  
------------------------------------------------------  
Project Introduction  
Understanding Requirement (BRS, Mockup,  
Wireframe)  
Understanding on Test Scenario & Test Case  
Create Test Scenarios  
Create Test cases from BRS Document  
Test Execution  
--------------------------  
  
**7 Principles of Software Testing**  
The seven basic Software Testing Principles that every Software tester and QA professional should know.  
  
7 Principles of Software Testing:  
1. Testing shows presence of defects  
  
There are several domains available in the market like Banking, Insurance, Medical, Travel, Advertisement etc and each domain has a number of applications. Also for each domain, their applications have different requirements, functions, different testing purpose, risk, techniques etc.  
Different domains are tested differently, thus testing is purely based on the context of the domain or application.  
For Example, testing a banking application is different than testing any e-commerce or advertising application. The risk associated with each type of application is different, thus it is not effective to use the same method, technique, and testing type to test all types of application.  
2. Exhaustive testing is not possible  
  
It is not possible to test all the functionalities with all valid and invalid combinations of input data during actual testing. Instead of this approach, testing of a few combinations is considered based on priority using different techniques.  
Exhaustive testing will take unlimited efforts and most of those efforts are ineffective. Also, the project timelines will not allow testing of so many number of combinations. Hence it is recommended to test input data using different methods like Equivalence Partitioning and Boundary Value Analysis.  
3. 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  
4. Defect clustering  
  
During testing, it may happen that most of the defects found are related to a small number of modules. There might be multiple reasons for this like the modules may be complex, coding related to such modules may be complicated etc.  
This is the Pareto Principle of software testing where 80% of the problems are found in 20% of the modules.   
By experience, we can identify such risky modules.  
5. Pesticide paradox  
Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.  
6. Testing is context dependent  
The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the Android application.  
7. Absence of errors fallacy  
  
If the software is tested fully and if no defects are found before release, then we can say that the software is 99% defect free. But what if this software is tested against wrong requirements? In such cases, even finding defects and fixing them on time would not help as testing is performed on wrong requirements which are not as per needs of the end user.  
  
For Example, suppose the application is related to an e-commerce site and the requirements against “Shopping Cart or Shopping Basket” functionality which is wrongly interpreted and tested. Here, even finding more defects does not help to move the application into the next phase or in the production environment.  
  
  
**Product & Project:**  
if Software application is developed for specific customer based on the requirements then it is called Project.  
  
Example: banking Application  
  
if Software application is developed for multiple customer based on the market requirements then it is called Product.  
  
Example: MS Excel, Word, Google product (Google Map)  
  
**QA/Testing Activities:**  
  
Understanding the requirement functional specifications of the applications  
Identifying required test scenarios'  
Designing test cases to validate application  
Setting up test environment  
Execute test case to validate requirement  
Log Test Result (How many test cases pass or fail)  
Defect reporting and tracking  
Retesting fixed defect which is find out during testing  
Perform various types of testing in the application  
Report to respective line manager regarding the assigned task status  
Participate on regular team meeting (scrum meeting)  
Creating automation script  
provides recommendation on whether or not the application is ready for production  
  
**Software Testing Types:**  
White Box Testing & Black Box Testing  
Positive Testing   
Negative Testing  
Exploratory Testing  
Ad hoc testing  
Re-Testing  
Regression Testing   
Smoke Testing  
Sanity Testing  
Smoke Testing   
System Testing  
Integration Testing   
Usability Testing  
  
  
**Project Introduction:**  
 **Understanding Requirement (BRS, Mockup,  
Wireframe):**  
  
**What is a Test Scenario?**  
**Test Scenario gives the idea of what we have to test.**  
Test Scenario is like a high-level test case.  
**Test Scenarios are derived from Requirements/ User Stories**.  
**Test cases are derived from Test Scenarios .  
Test scenario is ‘What functionality is to be tested’**  
  
**What is test Tase?**  
 **Test Cases are ‘ How to test the functionality ’.**  
  
A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement.   
  
The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.  
  
A Test Case is a set of actions executed to verify a particular feature or functionality of your software application.   
  
A test case is a defined format for software testing required to check if a particular application/software is working or not. A test case consists of a certain set of conditions that need to be checked to test an application or software.  
  
  
**Need to focus on below's:**  
Test Scenarios have multiple test cases.  
Test case may or may not be associated to multiple Test scenarios.  
  
Single test scenarios are never repeatable.  
Single test case may be used for multiple times in different scenarios.  
Brainstorming sessions are required to finalize a Test Scenario.  
Detailed technical knowledge of the software application is required  
Brief documentations required.  
Detailed documentation is required.  
**Assume that we need to test the functionality of a login page of Gmail application.**  
  
Test scenario for the Gmail login page functionality as follows:  
Verify the login functionality  
One Test Scenario can have multiple ‘Test Cases’.   
  
A test scenario, sometimes also called a scenario test, is basically a documentation of a use case.  
  
Example 1: Test Scenario for eCommerce Application  
  
Test Scenario 1: Check the Login Functionality  
  
In order to help us understand the difference Test Scenario and Test Cases, specific test cases for this Test Scenario would be:   
Check system behavior when valid email id and password is entered.   
Check system behavior when email id and password are left blank and Sign in entered.  
  
Test Scenario 2: Check the Search Functionality  
  
**Best Practice for writing good Test Case:**  
1. Test Cases need to be simple and transparent:  
  
Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them.  
Use assertive language like go to the home page, enter data, click on this and so on. This makes the understanding the test steps easy and tests execution faster.  
  
2. Create Test Case with End User in Mind  
  
The ultimate goal of any software project is to create test cases that meet customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective  
  
3. Avoid test case repetition.  
  
Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column  
  
4. Do not Assume  
  
Do not assume functionality and features of your software application while preparing test case. Stick to the Specification Documents.  
  
5. Ensure 100% Coverage  
  
Make sure you write test cases to check all software requirements mentioned in the specification document. Use Traceability Matrix to ensure no functions/conditions is left untested.  
  
6. Test Cases must be identifiable.  
  
Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.  
  
7. Implement Testing Techniques  
  
It’s not possible to check every possible condition in your software application. Software Testing techniques help you select a few test cases with the maximum possibility of finding a defect.

* Boundary Value Analysis (BVA): As the name suggests it’s the technique that defines the testing of boundaries for a specified range of values.
* Equivalence Partition (EP): This technique partitions the range into equal parts/groups that tend to have the same behavior.
* State Transition Technique: This method is used when software behavior changes from one state to another following particular action.
* Error Guessing Technique: This is guessing/anticipating the error that may arise while doing manual testing. This is not a formal method and takes advantages of a tester’s experience with the application

8. Self-cleaning  
  
The test case you create must return the Test Environment to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.  
  
9. Repeatable and self-standing  
  
The test case should generate the same results every time no matter who tests it  
  
10. Peer Review.  
  
After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.  
  
Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results.  
  
Test Case answers “”  
  
Assume that we need to test the functionality of a login page of Gmail application. Test cases for the above login page functionality as follows:  
  
 Enter valid User Name and valid Password  
 Enter valid User Name and invalid Password  
 Enter invalid User Name and valid Password  
 Enter invalid User Name and invalid Password  
  
**Software Testing Types:**  
  
**What is Functional testing in software?**  
It is a type of software testing which is used to verify the **functionality** of the software application, whether the function is working according to the requirement specification. In functional testing, each function tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application.  
  
Main focus on the functionality of the element  
  
**What is functionality?**  
  
functionality is nothing but behavior of the application.  
Functional testing talks about how the application feature should work according to the customer requirement.  
Functional testing can be manual or automated.  
  
**Functional testing involves the following steps:**  
  
Identify function that is to be performed.  
Create input data based on the specifications of function.  
Determine the output based on the specifications of function.  
Execute the test case.  
Compare the actual and expected output.  
  
**Types of functional testing are:**  
Unit Testing  
Smoke Testing  
Sanity Testing  
Integration Testing  
White box testing  
Black Box testing  
User Acceptance testing  
Regression Testing  
Retesting  
Ad-hoc testing  
Grey Box Testing  
  
**Black Box Testing:**  
  
Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.  
  
Black Box Testing is also known as behavioral, opaque-box, closed-box, specification-based or eye-to-eye testing.  
  
The main focus of Black Box Testing is on the functionality of the system as a whole. The term ‘Behavioral Testing’ is also used for Black Box Testing.  
  
The test engineers perform the black box testing.  
  
**Types of Black Box Testing:**  
  
Practically, there are several types of Black Box Testing that are possible, but if we consider a major variant of it then only the below mentioned are the two fundamental ones.  
  
1) Functional Testing  
  
2) Non Functional Testing  
  
**Functional Testing:**  
  
This black box testing type is related to the functional requirements of a system; it is done by software testers.  
  
Few major types of Functional Testing are:

* Smoke Testing
* Sanity Testing
* Integration Testing
* System Testing
* Regression Testing
* User Acceptance Testing

**Non Functional Testing**  
This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.  
there are even several non-functional aspects that are required to be tested to improve the quality and performance of the application.  
  
Few major types of Non-Functional Testing include:

* Usability Testing
* Load Testing
* Performance Testing
* Compatibility Testing
* Stress Testing
* Scalability Testing

**What is White Box Testing?**  
  
White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.  
  
Developers do white box testing  
  
White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.  
  
Types of White Box Testing

* Unit Testing
* Static and Dynamic Analysis
* Memory Leaks Testing
* Statement Coverage
* Branch Coverage
* Mutation Testing
* Security Testing

Usability Testing  
  
Usability Testing also known as User Experience(UX) Testing, is a testing method for measuring how easy and user-friendly a software application is.  
  
Check how easily the end user are able to understand and operate the application.   
  
Checking the user-friendliness, efficiency, and accuracy of the application is known as Usability Testing.  
usability testing is performed from an end-user viewpoint to verify if the system is efficiently working or not.  
 Usability testing mainly focuses on user’s ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.  
  
This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.  
  
it is also known as User Experience (UX) Testing  
  
It can be implemented in the Designing phase of the software development life cycle (SDLC) in order to helps us getting more clarity of the user's needs.  
  
In Usability Testing, the user-friendliness can be described with the help of the following characteristics:

* Easy to understand
* Easy to access
* Look and feel
* Faster to Access
* Effective Navigation
* Good Error Handling

Positive Testing  
  
Positive Testing is a type of testing which is performed on a software application by providing the valid data sets as an input. It checks whether the software application behaves as expected with positive inputs or not. Positive testing is performed in order to check whether the software application does exactly what it is expected to do.  
  
Example attached  
  
**Negative Testing**  
  
Negative Testing is a testing method performed on the software application by providing invalid or improper data sets as input. It checks whether the software application behaves as expected with the negative or unwanted user inputs. The purpose of negative testing is to ensure that the software application does not crash and remains stable with invalid data inputs.  
  
**What is exploratory testing?**  
  
We have to explore the application, understand, completely and test it.  
Understand the application, identify all possible scenarios, document it then use it for testing.  
We do exploratory testing when the application ready but there is no requirement  
  
**Drawback:**  
  
**Might misunderstand any features as a bug since no requirement**  
**timen consuming**  
**If there is nay bug in appplciation, we will never know about it.**  
If requirement does not exist, then we do one round of 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**  
.  
So, for this first, we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application, this approach is known as exploratory testing.  
  
**When we use exploratory testing?**  
  
We will use this testing for the following aspects:

* When the requirement is missing
* Early iteration is required
* The testing team has the experienced testers when we have a critical application, and new testers entered into the team.

**How to perform exploratory testing?**  
  
To perform exploratory testing, first, we will start using the application and understand the requirement of the application from the person who has a good product knowledge such as senior test engineer, and developers.  
Then we will explore the application and write the necessary document, and this document is sent to the domain expert, and they will go through the document and we can test the application based on our knowledge, and taking the help of the competitive product, which is already launched in the market.  
  
**Ad hoc Testing**  
  
**testing application randomly without any test cases or any business requirement document**  
  
**- To find the corner scenario's**  
Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage.  
  
Test Engineer should have knowledge of application even though we doesn't have requirement  /test cases.  
  
Ad hoc testing is done randomly and it is usually an **unplanned activity**which does not follow any documentation and test design techniques to create test cases.  
  
Ad hoc testing is also known as Monkey testing and Gorilla testing.  
  
When execute Ad hoc Testing?  
  
Ad hoc testing can be performed when there is limited time to do elaborative testing. Usually ad hoc testing is performed after the formal test execution. And if time permits, ad hoc testing can be done on the system. Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

* No Documentation.
* No Test cases.
* No Test Design.

Best practices of Ad hoc testing  
  
Good business knowledge  
  
It is done after formal testing.  
  
We go for Ad hoc testing when all types of testing are performed. If the time permits then we will check all the negative scenarios during ad hoc testing.

* When there limited time in hand to test the system.
* When there is no clear test cases to test the product.
* When formal testing is completed.
* When the development is mostly complete.

**Retesting**  
Retesting is testing of a particular bug after it has been fixed. Usually tester raises the bug when they find it while testing the product or its component. This bug is assigned to a developer and he fixes it. Post fixing the bug is assigned to the tester for its verification. This testing is known as retesting.

* Retesting is done by replicating the same scenario with same data in new build.
* In retesting those test cases are included which were failed earlier.
* Retesting ensures that the issue has been fixed and is working as expected.

What is Smoke Testing?  
  
Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”