MANUAL TESTING

Software Testing Concepts - Session #01

What is Software

Software is collection of programs to help us to perform a task.

Types of Software

Application software
System Software
Programming software

What is Software Testing

Software testing is a part of software development process where product is verified and validated to ensure

1) software or application is bug-free
2) meets the customer requirement
3) timely delivery within budget
4) Maintainable product

Why Software Testing is important?

Cost-Effective
Security
Product quality
Customer Satisfaction

Error, Bug/Defect & Failure

Reasons of Software Bugs

1) Lack of communication between stakeholders, development and testing team.
2) Frequent change in requirement/ unclear requirement
3) Software complexity
4) Programming error/source code error
5) Unskilled testing team
6) Time Pressure

MANUAL TESTING

Software Testing Concepts - Session #02

What is Software Development Life Cycle (SDLC) or Application Development Life Cycle

The software development life cycle (SDLC) is a process used in software development to plan, write, and modify software.
SDLC is a systematic process for building software that ensures the quality and correctness of the software built.
It Provides a framework for a standard set of activities and deliverables.

Phases of SDLC

1) Requirement Gathering & Analysis
2) Design
3) Development/implementation / coding
4) Testing
5) Deployement
6) Maintenance

SDLC Models

1) Waterfall model
2) Spiral Model
3) V-Model

Waterfall model

Waterfall model illustrates the software development process in a linear sequential flow. This means that any phase in the development begins only if the previous phase is complete. Therefore, phases donot overlap.

Advantage

a) simple & easy to understand
b) Each phase has specific deliverable and review process which makes it easy to manage.
c) Works well with smaller projects where requirements are very well understood.
d) Clearly defined stages
e) Result of each phases are well documented.

DisAdv

Cannot accommodate changing requirements. Thefore , not suitable for project where there is risk of requirement change.
It is not good for complex projects

No working software is produced until late during the life cycle. Testing will start only after coding is complete.
There is high amount of risk and uncertainity.

If there is defect in requirement phase it will be continued in the later phases also.

Spiral Model

Spiral Model is Iterative in nature and It is also called Version Control Model.
It allows incremental releases of the product or incremental refinement through each iteration around the spiral.
It overcomes limitation/drawback of waterfall model.
It is best suited for large project or where there is dependency in modules.

Advantages

It allows requirement changes
Suitable for large and complicated projects
The development can be distributed into smaller parts.
It allows better risk analysis
Cost effective due to good risk management

Disadvantages

Not suitable for small projects
The success of the project depends on the risk analysis phase
There is no testing in requirement and design phase.

V-Model/ Verification and Validation Model(V&V Model)

Static Testing - Testing project related document is called static testing.

Review
Inspection
Walkthrough

Dynamic Testing

Whitebox testing
Functional Testing
Integration Testing
System Testing
User Acceptance Testing

Verification

Verification is to check whether we are building the right product or not.
Verification in Software Testing is a process of checking documents, design in order to check if the software has been built according to the requirements or not.
The verification process involves activities like reviews, walk-throughs and inspection. Static testing is performed in Verification.

Validation in Software Engineering is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not.
Validation mean checking whether the software product is right or not.
Dynamic testing is performed for validation.

Advantages:

Testing starts in early stages of product development.

Test team will be ready with the test cases by the time developers release the software which in turns saves a lot of time

Testing is involved in every stage of product development.

Less or no rework.

Disadvantages:

Initial investment is more because test team & development team involves right from the early stage.

Whenever there is change in requirement, the same procedure continues. It leads more documentation work.

Why SDLC

Here, are prime reasons why SDLC is important for developing a software system.

It offers a basis for project planning, scheduling, and estimating
Provides a framework for a standard set of activities and deliverables
It is a mechanism for project tracking and control
Increases visibility of project planning to all involved stakeholders of the development process
Increased and enhance development speed
Improved client relations
Helps you to decrease project risk and project management plan overhead

MANUAL TESTING

Software Testing Concepts - Session #03

Difference Between Technical Review , Walkthrough & Inspection

Participants of Review Process:

The Moderator : Leads the review process. His role is to determinse the type of review, scheduling meeting, distribute documents to other particpants.

The Author: He is writer of the "document under review" . His job is to understand improvement.

The Scribber/writter/recorder: He is responsible to record each defect found and any suggestions given in the meeting for process improvement.

The Reviewer: to check defects and further improvement.

Technical Review : It is lead by trained moderator and technical reviewers will read the document before meeting for better understanding and to suggest improvement to author.
Conducted on these documents to ensure correctness and completeness of documents.

Requirement review
Design review
Code review
Test plan
test case

Walkthrough :

It is a informal review. In this author reads the document or code and discuss with the peers so that they note out the defects ans suggestions. It is not preplanned and it can be done whenever it is required.

Inspection :

It is most formal review type, in which an author requests the services of a moderator, scribe, reviewers in a formal meeting. The moderator books the room, sends out the material. The reviewers read the material before the meeting. During the meeting, the reviewers take turns reading the work artifact out loud. The scribe takes notes of issues the reviewers discovered in advance and during the meeting. A formal follow-up is carried out by the moderator after inspection.

Quality Assurance (QA) Vs Quality Control (QC)

SDLC Phases

Requirement Analysis & Gathering Phase
Design
Coding
Testing
Deployment
Maintenance

Quality Assurance involves in process-oriented activities. It ensures the prevention of defects in the process used to make Software Applications. So the defects don’t arise when the Software Application is being developed.

QA aims to prevent defects. QA is responsibe for full SDLC life cycle.

Quality Control involves in product-oriented activities. It executes the program or code to identify the defects in the Software Application.

QC Aims to identify and fix defects. QC is responsible for software testing.

QE - Quality Engineers

MANUAL TESTING

Software Testing Concepts - Session #04

Levels Of Sofware Testing

SDLC Phases

Requirement Analysis & Gathering phase
Design
Coding
Testing
Deployment
Maintenance

There are mainly 4 levels of software testing

Unit Testing: checks if sofware component are fulfilling functionalities or not. It is done by developers in development environment.

Integration Testing: checks the data flow from one modeule to another module. Integration means combining. For example , In this testing phase difference modues are combined and tested as a group to make sure that integrated system is ready for system testing.

Approaches of Integration Testing

1) Top down integration - In Top-Down Integration Testing, the high-level modules are integrated and tested first. i.e Testing from the main module to the submodule. In this approach stubs are used as a dummy module for the missing module if submodule is under development or not available. Stubs are called by the module under test

2) Bottom up integration : In Bottom Up Integration Testing, the low-level modules are integrated and tested first i.e Testing from sub-module to the main module. In this approach drivers are used as a temporary module for mission module if module is under development or not available. Drivers , it calls the module to be tested.

Different Modules	Module functionality
Module-1	Login page of the web application
Module-2	Home-page of the web application
Module-3	Print Setup
Module-4	Log out page

System Testing: System testing is performed on integrated system. It involved load , performace testing, reliability and security testing. It allows checking system's compliance as per the requirement. It focus on

Functional testing
non functional testing
UserInterface testing
Usability testing

It is end-to-end testing where the testing environment is parallel to the production environment. In the third level of software testing, we will test the application as a whole system.

User Acceptance Testing (UAT): Acceptance testing is conducted by the customer/stakeholder/users/domain expert for their satisfaction before accepting the final product. This is fourth level of testing.

Types of UAT

Alpha Testing:

Beta Testing:

MANUAL TESTING

Software Testing Concepts - Session #05

System Testing

System testing is 3rd level of testing which focuses on verifying the system as a whole meets specified requirement.
This is performed after Integration testing and before User Acceptance testing.
It is performed by the testing team independent of development team.
It is black box testing.
This testing is performed in testing environment parallel to production environment.

System Testing Process

Test environment setup
Test plan creation
Test case preparation
test data preparation
Test case execution
Defect report preparation
Retest & Regression testing
signoff

Types of system testing

1) Graphical User Interface (GUI) Testing
2) Functional Testing
3) Usability testing
4) Non - functional testing

GUI testing

GUI testing is software testing type that checks the Graphical user Interface of the Software to ensure the functionalities of the software works as per requirement specification.

Checklist for GUI Test

Check for -

1. All the GUI elements for size, position, width, length, and acceptance of characters or numbers.
2. Error Messages & Warning message are displayed correctly
3. For Clear demarcation of different sections on screen
4. Font used in an application is readable
5. Alignment of the text is proper
6. Color of the font and warning messages is aesthetically pleasing
7. Images have good clarity and they are properly aligned
8. Positioning of GUI elements for different screen resolution.
9. Execution of the required functionality of the application using the GUI
10. Test the color of the hyperlink
11. check for spelling
12. Test the scrollbar according to the size of the page.
13. Test the headings whether it is properly aligned or not.

Usability Testing Or User Experience Testing (UX Testing)

In Usability Testing, we check the application for user friendliness, efficiency and accuracy."

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

Context Sensetive Help : Online help or application embedded help for improve user experience.

MANUAL TESTING

Software Testing Concepts - Session #06

Functional Testing

Functional Testing is a type of Software Testing whereby the system is tested against the functional requirements.
Functions/features are tested by providing appropriate input and examining the output. The actual results are then compared with expected results.
Functional testing ensures that the requirements are properly satisfied by the application.

Testers follow the following steps:

verification/Analysis of the requirement specification in the software application.
Create Test Plan
Design the test case.
Make traceability matrix is to trace the requirement with its corresponding test scenarios and test cases.
Execute the test case design
Analysis of the coverage to examine the covered testing area of the application.
    Finding the area of a requirement not implemented by a set of test cases
    Helps to create additional test cases to increase coverage
    Identifying meaningless test cases that do not increase coverage
Defect management should do to manage defect resolving.

1) Database Testing

Database Testing is used to validate the functional requirements of a database from the end-user’s perspective.
The main goal of functional database testing is to test whether the transactions and operations performed by the end-users which are related to the database works as expected or not.

SQL (Structure Query Language) : It is a programming used to define and manipulate the databse.

DDL (Data Definition Language)
DML (Data Manipulation Language) - Insert, Delete , Query , Update

Database testing involves checking :

Schema / Mapping Testing
Stored Procedures and Views Testing
Trigger Testing
Tables and Column testing
Database Server Check

Calculation Testing
Error Handling Testing
Verify HyderLinks in case of Web application testing
Cookie Testing
Session Testing

MANUAL TESTING

Software Testing Concepts - Session #07

Non Functional Testing

Non-functional testing verifies the attributes of the system/software such as performance, load, stress, scalability, security, compatibility, etc., Also it focuses on improving the user experience on how fast the system responds to a request.
It checks the attributes such as memory leaks, performance, or robustness of the system.
It covers all the areas that are not covered in functional testing.

It is performed once the functional testing is complete.
This is Black box testing technique as it doesnot require knowledge of the internal system i.e it doesnot require knowledge of the code for the tester.

Non-Functional Testing Types

Performance Testing:

Evaluate the overall performance / speed of the system. It validates that the system mees the expected response time.

Load Testing: Slowly/gradually increase the load on the application/system and check the response time / speed of the application.

Stress Testing: Sudden increase the load on the application/system and check the response time /speed of the application.

Volume testing: We evaluate how much data application is able to handle or we can it evaluates the behavior of the application when large amount of data is passed.

Security Testing: Evaluates how secure is our application , to ensure there is no loophole in the application leads to thread or data loss.

This includes testing of authentication and authorization.

authentication - who you are
authorization/access - what you can do

Recovery Testing: Checks that application terminates gracefully in case of failure and data is recovered.

Compatibility Testing: We check whether the application is compatible with difference environment like web browser, hardware platform, databases, operating system, newtoworks , different version , configuration etc... In this we ensure that application works without any issue in different environment.

Forward Compatibility : check the behavior and compatibility of the hardware or software with newer version.

Backward compatibility :This testing is performed to check the behaviour and compatibility of the hardware or software with their older versions.

Instability Testing: checks if the software installs and uninstalls correctly.

Sanitation Testing/Garbage Testing .During this testing tester are finding extra functionality in build w. r. to Customer requirement

Software Testing Concepts - Session #08

Regression Testing

It is used to authenticate a code change in the software does not impact the existing functionality of the product. Regression testing is making sure that the product works fine with new functionality, bug fixes, or any change in the existing feature.

Types of Regression Testing

1) Unit Regression : testing only the changed feature is called the Unit Regression Testing.

2) Partial/Regional Regression: test the modification along with the impact area or regions . To find the impacted areas , impact analysis meeting is conducted and impact list is prepared.

3) Complete Regression : Test the modification along with the remaining areas.

Re-Testing

Re-testing Testing means testing the functionality or bug again to ensure that bug is fixed. If not fixed, defects need not be re-opened. If fixed, the defect closed. This is to ensure that the bugs of previous build is fixed or not.

MANUAL TESTING

Software Testing Concepts - Session #09

Smoke Testing/Build Verification Testing:

Smoke Testing is performed after software build to find out the critical /basic functionalities of the software is working find. This is to check whether software is testable or not.

Sanity Testing

Aims to quickly evaluate whether the basic functionality of a new software build is working correctly or not.

Smoke Testing	Sanity Testing
It is performed on initial unstable builds & verifies whether the software is testable or not.	It is performed on stable build (post regression) to verify the basic feature / high level feature of the software is working or not
It is part of basic testing	It is part of regression testing.
It is performed by QA team but can be performed by developer also	It is always performed by QA team.
This testing is a normal health check-up of the application build before taking it to test in-depth.	verify whether the requirements are met or not. High level features are verified.

MANUAL TESTING

Software Testing Concepts - Session #10

Adhoc Testing

It is a informal testing process. It is performed after formal testing.
There is no documentation.
There is no test design
There is not test cases.
There is not requirement spec.
Mostly negative test scenarios are tested with a aim to break the system.
In this we check application without any sequence or procedure randomly and find some issues.
The effectiveness of adhoc testing depends upon the capability of the tester and tester in depth knowledge about the system.
It helps to simulate unusual behavior and hard-find and hard to reproduce bugs/defects.

Exploratory Testing

In this testing, first application under test is explored, flow of application is understood , then test case is designed and test cases are executed.
This testing is performed when requirement is missing.
This is also informal testing process.
It involves test design and control and notes are taken
progress is also tracked.
It helps to study the product and agument the document and also research the bug.

Monkey Testing

This test is also random in nature, therefore test cases are not used in monkey testing.
Monkey testing can be performed by an individual who does not have a good knowledge of the application.
Tester test randomly by clicking on random objects and entering the random and invalid data to check is the application give an error or not.

MANUAL TESTING

Software Testing Concepts - Session #11

Positive Testing

In positive testing, tester always checks application with valid set of input.
Tester checks whether an application behaves as exptected with the positive input.

Negative Testing

tester checks an application with invalid set of input.
checks whether an application behaves as exptected with the negative input.
This is to test the application that does not do anything that it is not suppose to do.

End To End Testing (E2E Testing)

In E2E testing, tester checks the flow of application from start to end.

In E2E testing, tester create an environment identical to the one that will be used by real users. Then tester tests all actions that user might perform on the application.

1) **Requirement:** Password text box field shall allow users to enter passwords between 8 and 15 alphanumeric characters.

Various positive and negative test scenarios of above requirement.

Positive Test Scenarios	**Negative Test Scenarios**
The Password text box should allow 8 characters input.	The password text box should throw an error or should not accept when less than 8 characters are entered.
The Password text box should allow 15 characters of input.	The password text box should throw an error or should not accept when more than 15 characters are entered.
Any values between 8 and 15 characters long should be accepted by the Password text box.	The password text box should not accept special characters as input
It should accept any combination of letters and numbers in Password text box.	The password text box should not accept a combination of numbers only or a combination of letters only.

<u>2) Requirement:</u> A text box field shall allow users to enter alphabets between 6-20 characters.

Various positive and negative test scenarios of above requirement.

Positive Test Scenarios	**Negative Test Scenarios**
Text box accepts 6 characters.	Text box shall not accept less than 6 characters
Text box accepts upto 20-character length	Text box shall not accept more than 20 characters
Text box accepts any value between 6-20 character length	Text box shall not accept special characters
Text box accepts all alphabets	Text box shall not accept numbers

MANUAL TESTING

Software Testing Concepts - Session #12

Test Design Techniques

Help you design better test cases. It helps to prepare test data.

Reduce the test data
Reduce the number of test cases
Increasing test coverage i.e to cover each and every area of the feature

Types Of Test Case Design Technique

Equivalence Class Partitioning (ECP)
Boundary Value Analysis (BVA)
Decision Table
State Transition
Error Guessing

Equivalence Class Partitioning (ECP)

We divide the test conditions in classes or groups and from each group we check only one condition assuming all the conditions in the group work in the same manner. It reduces no. of test data and saves time.

Example 1: we have to test a field which accepts age 18-35

Valid Age Input is 18 to 35

```
--- to 17(invalid) - 16
18 to 35 (valid) - 25

36 to --- (invalid)-55
```

Example 2: we have to test a field which accepts mobile number of 10 digit

Equivalance partition

```
less than 10 digit (invalid): 98715431
10 digit (valid):            9868543124
more than 10 digit (invalid):986854312498
```

Boundary Value Analysis (BVA)

We check the boundaries of the input.

Example 1: we have to test a field which accepts age 18-35

```
min=1 8 (valid)
min-1=      17(invalid)
min+1=      19(valid)

max   = 35(Valid)
max +1      = 36(invalid)
max-1 = 34 (valid)
```

ECP & BVA techniques are used in input domain testing

MANUAL TESTING

Software Testing Concepts - Session #13

Decision Table / cause-effect table

This technique is appropriate for preparing test data to test functionalities which has logical relationship.

Example 1: login page validation. Allows user to login only when user name and password are correctly entered.

	Test Case-1	Test Case-2	Test Case-3	Test Case-4
User name - Condition1	T	T	F	F
password - Condition2	T	F	T	F
Home page - Action 1	Execute			
Show Msg “Invalid user credential” - Action 2		Execute	Execute	Execute

Example 2 : Transferring money online to an account which is already added and approved.

	Test Case-1	Test Case-2	Test Case-3	Test Case-4	Test case - 5
Account Approved	T	T	T	T	F
OPT Matched	T	T	F	F	X
Sufficient Money in account	T	F	T	F	X
Transfer money	Execute				
Show msg “Insufficient balance”		Execute			
Block the transaction			Execute	Execute	X

State Transition Testing Technique

This technique is used when features of a system are represented as states which transform into one another.

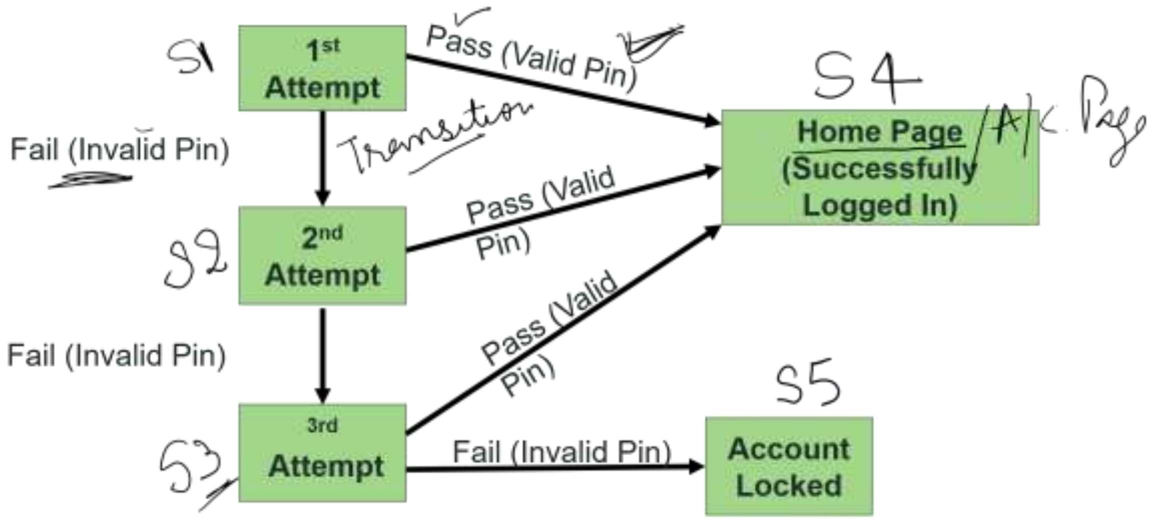
Example:

ATM system feature, where if the user enters the **invalid password three times the account will be locked.**

If the user enters a **valid password** in any of the **first three attempts** the user will be **logged in successfully.**

If the user enters the **invalid password** in the **first or second try**, the user will be asked to re-enter the password. And finally, if the user enters incorrect **password 3rd time**, the **account will be blocked.**

There are two main ways to represent or design state transition, State transition diagram, and state transition table.



State Transition Diagram

Test Case	TC1	TC2	TC3	TC4	TC5	TC6
State	S1	S1	S2	S2	S3	S3
Input	Correct Pin	Incorrect Pin	Correct Pin	Incorrect Pin	Correct Pin	Incorrect Pin
Output	Home Page	2nd Attempt	Home Page	3rd Attempt	Home Page	Account Locked
Finished State	S4	S2	S4	S3	S4	S5

State Transition Table

Convert above State Transition to formal test case

TC ID	Description	Steps	Expected Result
TC1	Validate that the system is able to do the transition from 1st attempt to Home Page on correct pin input	1. Open the application 2. input correct pin	User shall be directed to Home Page
TC2	Validate that the system is able to do the transition from 1st attempt to 2nd attempt on incorrect pin input	1. Open the application 2. input incorrect pin	User shall be directed to 2nd attempt
TC3			
TC4			
TC5			
TC6			

MANUAL TESTING

Software Testing Concepts - Session #15

Error Guessing Technique

Error guessing is a software testing technique in which the tester uses their experience of prior testing & intuition to guess the types of errors that might occur in a system, and then tests for those specific errors.

It is not a replacement for formal testing methods and should not be relied upon as the sole testing approach.

The error guessing is best used to supplement other testing techniques and methods to identify defects that may have been missed by other approaches.

Example

Suppose a tester is testing a web application that allows users to create and manage accounts.
The tester might use error guessing to identify potential errors that could occur during the account creation process.

Based on their experience with similar systems, the tester might guess that the following errors could occur:

Invalid email address format:

The system may not properly validate email addresses, allowing users to enter invalid formats that could cause problems later on.
Enter an invalid email address format (e.g., missing the "@" symbol) and attempt to create an account. Verify that the system displays an error message and does not allow the account to be created.

Password too weak-

The system may not enforce password complexity requirements, allowing users to create weak passwords that could easily guessed or hacked.

Create an account with a weak password (e.g., "password" or "123456"). Verify that the system displays an error message and does not allow the account create.

Duplicate usernames-

The system may not properly check for duplicate usernames, allowing users to create multiple accounts with the same username.

Attempt to create multiple accounts with the same username. Verify that the system displays an error message and does not allow the duplicate account create.

User profile not properly saved-

The system may not properly save user profile information, causing user data lost or corrupted.

Create a user profile and verify that the information properly saved and can access later.

MANUAL TESTING

Software Testing Concepts - Session #16

Software Development Life Cycle (SDLC)

Phases of SDLC

1) Requirement Gathering & Analysis
2) Design
3) Development/implementation / coding
4) Testing
5) Deployement
6) Maintenance

Software Testing Life Cycle (STLC)

Phases of STLC

1) Requirement Analysis
2) Test Planning
3) Test Design
4) Test Execution
5) Defect Reporting & Tracking
6) Closure/ Sign Off

Sample Defect Report Template

1) Defect ID: Unique No or Name

2) Description: Summary of the defect

3) Feature: Module / Function / Service, in these module TE found the defect

4) Test Case Name: Corresponding failing test condition

5) Reproducible (Yes / No): Yes -> Every time defect appears during test execution
No -> Rarely defect appears

6) If Yes, attach the test procedure:

7) If No, attach snapshot & strong reasons:

8) Status: New / Reopen

9) Severity: Seriousness of defect w.r.t functionality (high / medium / low)

10) Priority: Importance of the defect w.r.t customers (high / medium / low)

11) Reported bug: Name of Test Engineer

12) Reported on: Date of submission

13) Assign to: Name of the responsible person in development team -> PM

14) Build Version ID: In which build, Test Engineer fount the defect

15) Suggested fix (Optional): Tester tries to produce a suggestion to solve this defect

16) Fixed by: PM or Team Lead

17) Resolved by: Developer name

18) Resolved on: Date of solving By Developers

19) Resolution type: check out in next page

20) Approved by: Signature of Project Manager (PM)

Defect Age: The time gap between “reported on” & “resolved on”

MANUAL TESTING

Software Testing Concepts - Session #17

Test Plan

Test plan is like a blue print of how the testing activity is going to take place in the project.

Test Plan Template

1) Introduction
2) Scope
3) Test Strategy (Approach)
4) Test Environment
5) Staffing and Training Needs
6) Test Schedule and Estimation
7) Test Deliverables
      Test deliverable before testing phase - test plan doc, test case doc, test design spec
      Test deliverable during testing   - test script, simulators,test data, error logs & execution logs
      Test deliveables after testing the testing cycle is over - Test report, Defect report, Release note..

8) Exit Criteria
9) Suspension and Resumption Criteria
10) Responsibilities
11) Risk and Contingencies
12) Assumptions
13) Review and Approvals