Manual Testing Step by Step Tutorial

1) Software Development Life Cycle and SDLC Model

i) Requirement Gathering
ii) Analysis
iii) Design
iv) Coding / Development
v) Testing
vi) Deployment & Maintenance

SDLC Models

i) Waterfall Model
ii) V Model

2) Software Test Levels

i) Unit Testing
ii) Integration Testing
iii) System Testing
iv) Acceptance Testing

3) Software Test Types

i) Functional testing
ii) Non-functional testing
iii) Structural testing
iv) Change related testing

4) Software Test Design Techniques

i) Static Test Design Techniques

a) Reviews (Manual Examination)

1) Informal Review
2) Walk through
3) Technical Review
4) Inspection

b) Static Analysis (Automated Analysis)

ii) Dynamic Test Design Techniques

a) Specification based or Black box Techniques
1) Equivalence Partitioning (EP)
2) Boundary Value Analysis (BVA)
3) Decision Table Testing
4) State Transition Testing
5) Use Case Testing Etc…

b) Structure based or White box Techniques
1) Statement Coverage
2) Decision Coverage
3) Condition Coverage
4) Multi Condition Coverage
5) LCSAJ etc…

c) Experience based Techniques
1) Error Guessing
2) Exploratory Testing Etc…

5) Software Test Life Cycle

i) Requirement Analysis
ii) Test Planning
iii) Test Design & Development
iv) Test Environment Setup
v) Test Execution
vi) Test Cycle Closure

Manual Testing Step Step Videos

1) Software Development Life Cycle and SDLC Model

Software Development Life Cycle

> Software Development Life Cycle is a systematic approach to develop software. It is a Process followed by Software Developers and Software Testing is an integral part of Software Development, so it is also important for Software Testers...

> Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

> ISO/IEC 12207 is an international standard for software life-cycle processes. It defines all the tasks required for developing and maintaining software.

Phases of Software Development Life Cycle,

These phases may vary form one organization to another, but purpose is almost all same, that is "Develop and Maintain Quality Software",

i) Requirement Gathering
ii) Analysis
iii) Design
iv) Coding / Development
v) Testing
vi) Deployment & Maintenance

Note: It is General Software Development Life Cycle, we have various SDLC Models in the IT Industry, Waterfall Model, V Model, Spiral Model and Agile Development Models etc...,

Software Development process varies from one SDLC Model to another.

i) Requirement Gathering

> Requirement Gathering is the most important phase in software development life cycle, Business Analyst collects the requirements from the Customer/Client as per the clients business needs and documents the requirements in the Business Requirement Specification and provides the same to Development Team.

Note: Document name may vary from one Organization to another, Some examples are Customer Requirement Specification (CRS), Business Requirement Document (BRD) etc…

> Suppose Our Planned Software is not intended for a single customer and the software product for multiple customers then Business Analyst or Business Team collects Requirements from the Market and also evaluate Other similar products in the Market

> Key Role in this phase is Business Analyst and Outcome of the phase is "Business Requirement Specification"

ii) Analysis

> Once the Requirement Gathering is done the next step is to define and document the product requirements and get them approved by the customer. This is done through SRS (Software Requirement Specification) document. SRS consists of all the product requirements to be designed and developed during the project life cycle.

> Key people involved in this phase are Project Manager, Business Analyst and Senior members of the Team. The outcome of this phase is Software Requirement Specification.

iii) Design

> In Design phase Senior Developers and Architects, they give the architecture of the software product to be developed. It has two steps one is HLD (High Level Design) or Global Design and another is LLD (Low Level Design) or Detailed Design,

> High Level Design (HLD) is the overall system design, covers the system architecture and database design. It describes the relation between various modules and functions of the system.

> Low Level Design (LLD) is the detailed system design, covers how each and every feature in the product should work and how every component should work.

> The outcome of this phase is High Level Document and Low Level Document which works as an input to the next phase Coding...

iv) Coding / Development

> Developers (seniors, juniors and fresher) involved in this phase, this is the phase where we start building the software and start writing the code for the product.

> The outcome of this phase is Source Code Document (SCD) and the developed product.

v) Testing

> Once the software is complete then it is deployed in the testing environment. The testing team starts testing (either test the software manually or using automated test tools depends on process defined in STLC)

> Testing is done to verify that the entire application works according to the customer requirement.

> During this phase, Testing team may find defects which they communicate to developers, the development team fixes the defect and send back to Testing for a re-test. This process continues until the software is Stable, and working according to the business needs of that system.

vi) Deployment & Maintenance

> After successful testing, the product is delivered (deployed to the customer for their use), Deployment is done by the Deployment/Implementation engineers and Once when the customers start using the developed system then the actual problems will come up and needs to be solved from time to time.

> Fixing the issues found by the customer comes in the maintenance phase. 100% testing is not possible – because, the way testers test the product is different from the way customers use the product. Maintenance should be done as per SLA (Service Level Agreement)

Software Development Life Cycle Models

> There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models".

i) Waterfall Model

> Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Phases of Waterfall Model:

a) Requirements Gathering:

> This first step is also the most important, because it involves gathering information about what the customer needs and defining, in the clearest possible terms, the problem that the product is expected to solve.

b) Software Requirements:

> In this phase Business Requirements are converted as Software Requirements.

c) Design:

> In this phase Global and Detailed design can be produced based on Software Requirements.

d) Coding:

> This step consists of actually constructing the product as per the design specification(s) developed in the previous step. Typically, this step is performed by a development team consisting of programmers, interface designers and other specialists, using tools such as compilers, debuggers, interpreters and media editors. The output of this step is one or more product components, built according to a pre-defined coding standard and debugged, tested and integrated to satisfy the system architecture requirements.

e) Testing:

> In this stage, System will be tested by testers, if the find any mismatch they report defects. Developers /Programmers fix the defects and then testers close defects by performing confirmation testing (Regression Testing).

f) Release & Maintenance:

> Release team (consists of a few developers, testers, and tech-support people etc…) install software in Customer environment and they consider below factors;

Correct & Complete installation, User Management, Services Management, Coexistence with other software, Handling of Input & Output devices, and Handling of secondary storage devices

Etc…

Maintenance team process Customer issues based on service agreements.

Advantages of Waterfall Model:

a) Simple and easy to use

b) Easy to manage due to the rigidity of the model- each phase has specific deliverables and a review process.

c) Phases are processed and completed one at a time.

d) Works well for smaller projects where requirements are very well understood.

Disadvantages of Waterfall Model:

a) No working software is produced until late during the life cycle

b) High amount of risk and uncertainty

c) Poor model for complex and object oriented projects.

d) Poor model for Long and ongoing projects

e) Poor Model where requirements are at a moderate to high risk of changing.

---

ii) V Model

> It is Verification & Validation model, known as V Model, in this model all development phases can be integrated with Testing phases.

> The V-model illustrates how testing activities can be integrated into each phase of the software development life cycle.

> V Model was inaugurated in order to avoid drawbacks in Waterfall model and its main focus on multiple stages of testing.

> Multiple stages of Testing avoids defects multiplication.

Development Phases Integration with Testing Phases

a) User Requirements Vs Acceptance Testing

> Business Analyst category people gather requirements and the document the requirements, after documentation Reviews, Meetings like verification will take place in order get correct & Complete Requirements.

End Users conduct Acceptance Testing using Business / User Requirements.

b) Software Requirements Vs System Testing

> Development Manager/Tech Manager converts User Requirements as Software Requirements and Reviews, Meetings like verification methods will be performed on Software Requirements, after Verification Project manager provides Approval.

> Independent testers generate test cases from Software Requirements in order to perform System Testing

c) Global Design Vs Integration Testing

> System Architect / senior developer creates Global design, Informal Review/ Walk through / Technical Review / Inspection like Verification methods will be applied on Design documents.

> Developers perform Integration Testing based on Software Global Design.

d) Detailed Design Vs Unit / Component Testing

> Developers perform Unit /Component Testing based on Software Detailed Design.

Advantages of V Model:

a) Tester role will take place in the requirement phase it self

b) Multiple stages of Testing available so that Defects multiplication can be reduced.

c) Can be used for any type of requirements

d) Due to Multiple stages of Testing and Multiple teams involvement Quality can be improved.

e) The V Model Supports wide range of development methodologies such as Structured and Object oriented systems development.

f) The V Model supports tailoring.

Disadvantages of V Model:

a) It an expensive model than Waterfall model, needs lot of resources, budget and time.

b) Co-ordination and Maintenance are difficult.

c) Adoption of changes in Requirements and Adding New Requirements at middle of the process are difficult.

d) It needs an established process for proper implementation.

---

2) Software Test Levels

There are mainly four testing levels are:

i) Unit Testing
ii) Integration Testing
iii) System Testing
iv) Acceptance Testing

i) Unit Testing

> In Unit Testing level individual units/ components of a software are tested. The purpose is to validate that each unit of the software works as designed.

> Unit Testing can be done Manually as well as automated

> Unit Test Tools – NUnit, JUnit, TestNG etc…

> Developers conduct Unit Testing using White Box Test Design Techniques…

ii) Integration Testing

> In Integration Testing Level, individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

Popular Integration Test Tools:
FitNesse, Rational Integration Tester, Protractor, TESSY and Jasmine etc…

> Independent Testers conduct this level of Testing,

iii) System Testing

> In System Testing level a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified software requirements.

> System Testing can be done manually and automated...

Popular Testing Tools are:

Selenium, UFT, RFT (Functional Testing)
LoadRunner, JMeter, RPT (Performance Testing)
SoapUI – Web Services Testing
Appium – Mobile Testing etc…

> Independent Testers conduct System Testing using Block Box Test Design Techniques...

iv) Acceptance Testing

> In Acceptance Testing level a software system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements

Here, Acceptance basically two types,

1) Internal Acceptance Testing (Alpha Testing): It is conducted by members of the development organization who are not directly involved in the project, Usually, it is the members of Product Management, Sales and/or Customer Support people.

2) External Acceptance Testing / User Acceptance Testing (Beta Testing) is conducted by the end users of the software.

Note: Acceptance Testing environment and System Testing environment almost all same, but Unit Test Environment and integration Test environment are different.

---

3) Software Test Types

> A test type is a group of test activities aimed at testing specific characteristics of a software system, or a part of a system, based on specific test objectives. Such objectives may include:

> Evaluating functional quality characteristics, such as completeness, correctness, and appropriateness

> Evaluating non-functional quality characteristics, such as reliability, performance efficiency, security, compatibility, and usability

> Evaluating whether the structure or architecture of the component or system is correct, complete, and as specified

> Evaluating the effects of changes, such as confirming that defects have been fixed (confirmation testing) and looking for unintended changes in behavior resulting from software or environment changes (regression testing)

i) Functional Testing

> Functional testing of a system involves tests that evaluate functions that the system should perform. Functional requirements may be described in work products such as business requirements specifications, epics, user stories, use cases, or functional specifications, or they may be undocumented. The functions are "what" the system should do.

> Functional tests should be performed at all test levels. Functional testing considers the behavior of the software

ii) Non-functional Testing

> Non-functional testing of a system evaluates characteristics of systems and software such as usability, performance efficiency or security. Non-functional testing is the testing of "how well" the system behaves. non-functional testing can and often should be performed at all test levels, and done as early as possible. The late discovery of non-functional defects can be extremely dangerous to the success of a project.

iii) Structural Testing

> Structural Testing derives tests based on the system's internal structure or implementation. Internal structure may include code, architecture, work flows, and/or data flows within the system.

iv) Change-related Testing

> When changes are made to a system, either to correct a defect or because of new or changing functionality, testing should be done to confirm that the

changes have corrected the defect or implemented the functionality correctly, and have not caused any unforeseen adverse consequences.

> Confirmation Testing: After a defect is fixed, the software may be tested with all test cases that failed due to the defect, which should be re-executed on the new software version.

> Regression Testing: It is possible that a change made in one part of the code, whether a fix or another type of change, may accidentally affect the behavior of other parts of the code, whether within the same component, in other components of the same system, or even in other systems.

> Changes may include changes to the environment, such as a new version of an operating system or database management system. Such unintended side-effects are called regressions.

> Regression testing involves running tests to detect such unintended side-effects. Confirmation testing and regression testing are performed at all test levels.

> Regression test suites are run many times and generally evolve slowly, so regression testing is a strong candidate for automation. Automation of these tests should start early in the project.

---

4) Software Test Design Techniques

> What is Technique?

An Efficient way of doing or achieving something.

> What is Test Design Technique?

A test design technique is used to select a good set of tests from the all possible tests for a given system.

> Why we need to use Test Design Techniques?

Exhaustive Testing is not possible, so we need to use Test Design Techniques in order to reduce the size of the input.

Exhaustive Testing is a Test approach in which the test suite comprises all combination of input values and preconditions.

Exhaustive Testing is not recommendable due to Time and Budget considerations.

Categories of Test Design Techniques?

There are two main categories of Test Design Techniques, They are:

i) Static Techniques
ii) Dynamic Techniques

i) Static Techniques

> Testing of the software documents manually or with a set of tools but without executing the Software.

Two types of static testing techniques

a) Reviews (Manual Examination)

b) Static Analysis (Automated Analysis)
————————————-
a) Reviews
Types of Reviews

1) Informal Review
2) Walkthrough
3) Technical Review
4) Inspection

b) Static Analysis
Static analysis tools are typically used by developers, Compilers offer some support for Static analysis,

ii) Dynamic Test Design Techniques

> The software is tested by executing it on computer.

Categories of Dynamic Test Design Techniques

a) Specification based or Black box Techniques
1) Equivalence Partitioning (EP)
2) Boundary Value Analysis (BVA)
3) Decision Table Testing
4) State Transition Testing
5) Use Case Testing Etc…

b) Structure based or White box Techniques
1) Statement Testing and coverage
2) Decision Testing and Coverage
3) Condition Testing,
4) Multi Condition Testing
5) LCSAJ etc…

c) Experience based Techniques
a) Error Guessing
b) Exploratory Testing
————————————————————
2a) Black box Test Design Techniques
1) Equivalence Partitioning (EP)
2) Boundary Value Analysis (BVA)
3) Decision Table Testing
4) State Transition Testing
5) Use Case Testing Etc…

1) Equivalence Partitioning (EP)

• It can be applied at any level of testing (Unit, Integration, System and Acceptance Testing)

• In Equivalence Partitioning, inputs to the Software are divided into groups that are expected to exhibit similar behavior.

• Equivalence Partitions/Classes can be found for both valid data and invalid data.

Example 1 (Data Range):

Tickets field in a Reservation system accepts 1 to 10 Tickets only.

Partition 1	Partition 2	Partition 3
-Ne to 00	0 1 to 10	11 to 99 or above
(Invalid)	(Valid)	(Invalid)

Example 2 (Data Type):

Customer Identification Number field in a CRM system accepts only numbers.

Partition 1 Partition 2 Partition 3 Partition 4
Alpha bytes Numbers Special Characters Alpha-numeric
(Invalid) (Valid) (Invalid) (Invalid)

Example 3 (Data Size)

Phone Number filed accepts 10 digits number only

Partition 1 Partition 2 Partition 3
Below 10 10 Above 10
(Invalid) (Valid) (Invalid)

Example 4 (Others)

A Payment management system accepts credit card payments only

Partition 1 Partition 2 Partition 3
Credit card Net Banking Cash on Delivery
(Valid) (Invalid) (Invalid)

2) Boundary Value Analysis (BVA)

• The maximum and minimum values of a partition are its boundary values.

• Behavior at edge of each equivalence partition is more likely to be incorrect than behavior within the partition.

• Boundary value analysis can be applied at all Test levels(Unit, Integration, System and Acceptance Testing).

Example 1:
Partition 1 Partition 2 Partition 3

0 1 to 10 11 to 99 or above
(Invalid) (Valid) (Invalid)

Minimum/maximum 0
Minimum 1
Maximum 10
Minimum 11
Maximum 99
————————————-
Example 3 (Data Size)

Phone Number filed accepts 10 digits number only

Partition 1 Partition 2 Partition 3
Below 10 10 Above 10
(Invalid) (Valid) (Invalid)

Minimum -9
Minimum and Maximum – 10
Maximum -11
————————————-

Example: User Id field accepts 10 to 20 characters

Partition 1 Partition 2 Partition 3
Below 10 10 to 20 21 to 99

Minimum -1
Maximum – 9

Minimum – 10
Maximum – 20

Minimum – 21
Maximum -99

3) Decision Table Testing

• The decision tables are good way to capture system requirements that contain logical conditions.

• It may be applied for all situations when the action of the software depends on logical decisions.

BSRB (Govt) System Job eligibility criteria,

Age should be in between 21 and 35

Conditions:
i) For SC or ST Candidates 5 Years age relaxation
ii) For BC Candidates 5 Years age relaxation
iii) PHC Candidates 5 Years age relaxation

Category Age Valid/Invalid
————————————————-
OC 20 Invalid
OC 21 Valid
OC 35 Valid

OC 36 Valid
BC 36 Valid
BC 39 Invalid
SC 39 Valid
PHC 39 Valid
ST 40 Valid
————————————————————
Banking System interest rates For fixed deposits.
1 to 2 years 7%
2 to 3 Years 8%
3 to 5 Years 10%

Condition:

For Senior citizens 0.5% extra for all ranges

Age Period Interest Rate
————————————
25 1year 7%
35 2.5 8%
56 4 10%
66 4 10.5%

4) State Transition Testing

• In State transition Testing Test cases are designed to execute valid and invalid state transitions.

• A System (Application Under Test) may exhibit a different response on current conditions or previous history.

Example: Internet Banking System Fund Transfer operation

Initial Balance: 45000

Transaction Amount Transaction Result
1 20000 Successful (Pass)
2 20000 Successful (Pass)
3 20000 Unsuccessful (Pass)

5) Use Case Testing

• In Use Case Testing Test Cases are designed to execute User Scenarios or Business Scenarios.

• A Use Case describes interactions between actors, including users and the system.

• A Use case usually has a mainstream scenario and sometimes alternative scenarios.

Example:

Business Scenario: ATM Cash Withdrawal operation

Mainstream Scenario:

1)
User: Inserts ATM Card

System: Asks for PIN

2)
User: Enters PIN

System: Validates PIN and asks to select language

3)
User: Selects Language

System: Asks to select Account Type

4)
User: Selects Account Type

System: Asks to enter Amount

5)
User: Enters Amount

System: Releases Money
————————————————–
Alternatives

2a) Suppose if user enters invalid Pin

System: Shows error message and asks to enter correct PIN
User: Enters Correct PIN
—————————
4a) Suppose if user selects incorrect Account Type

System: Shows error and asks to select correct Account Type
User: Select correct account type

5a) If User enters incorrect amount (More than the balance amount or more than the day limit)

System: Shows Error message and asks to enter correct amount
User: Enters correct amount

---

5) Software Testing Life Cycle or Software Testing Process

> Software Testing Life Cycle (STLC) identifies what test activities to carry out and when to accomplish those test activities. Even though testing differs between Organizations, there is a testing life cycle.

> Just like Software Developers follow the Software Development Life Cycle (SDLC), Software Testers also follow the Software Testing Life Cycle.

> Software Test Process is not a single activity, it consists of many different activities which are executed to achieve a good quality product.

There are different phases in STLC which are given below:

i) Requirement Analysis
ii) Test Planning
iii) Test Design & Development
iv) Test Environment Setup
v) Test Execution
vi) Test Cycle Closure

We have Entry and Exit Criteria for all levels in the Software Testing Life Cycle…

Entry Criteria: Entry Criteria gives the prerequisite items that must be completed.

Exit Criteria: Exit Criteria defines the items that must be completed.

i) Requirement Analysis

> In Requirement Analysis phase, test team studies the requirements and identify the testable requirements.

Entry Criteria: Requirements Document available (both functional and non functional), Application Architectural document or Product should be available…

Exit Criteria: RTM should be signed off and The customer should sign off on the test automation feasibility

Activities in this phase:
a) Identify types of tests to be performed.
b) Risk Analysis
c) Prepare Requirement Traceability Matrix (RTM).
d) Identify Test environment details
e) Automation feasibility analysis (if required).

Deliverables:
a) Requirement Traceability Matrix (RTM)
b) Automation feasibility report(Optional)

ii) Test Planning

> In this phase the Test Manager or Test Lead prepares the Test Plan and Test strategy documents.

Entry Criteria: Requirements Document, Requirement Traceability Matrix (RTM) and Automation Feasibility Report

Exit Criteria: Approved Test Plan document, Test Strategy document and Effort estimation document

Activities in this phase:
a) Selection of Testing Approach
b) Test Estimation
c) Team Formation
d) Preparation of Test Plan, Test strategy documents
e) Configuration Management Planning
f) Resource planning
g) Test Tool Selection (if required)
h) Training Requirement

Deliverables:
a) Test Plan, Test Strategy document.
b) Test estimation document.
Etc..

iii) Test Design & Development

> In Test design phase, testers prepare test scenarios, test cases/test scripts and test data based on the Requirement Document/s and Test Plan.

Entry Criteria: Requirements Document/s, RTM and Test Plan, Automation analysis report

Exit Criteria: Reviews Test cases, Test Scripts (if automation) and Test data.

Activities in this phase:
a) Derive Test Scenarios
b) Test Case Documentation
c) Review Test Cases
d) Update RTM – Map Test Cases to Requirements in RTM
e) Creation of Test Scripts if required
f) Collect Test Data
Etc…

Deliverables:
a) Test cases
b) Test scripts (for automation if required)
c) Test Data

iv) Test Environment Setup

> It is a combination of hardware and software environment on which the tests will be executed.

> Test Environment supports test execution with software, hardware and network configured. Test environment configuration must mimic the production environment.

> Readiness of the test environment can be validated by smoke testing performed by the Testing team.

Entry Criteria: System design document/s, Architectural document of the application and Environment set-up checklist. Provision of Test Plan, readiness of Smoke Test cases and preparation of test data.

Exit criteria: Test environment should be ready and smoke testing should be performed successfully.

Activities:
a) Setup Test Environment and Test Data
b) Verify Test Environment by Conducting Smoke Tests

Deliverables:
a) Test Environment ready with test data set up
b) Smoke Test Results.

v) Test Execution

> In Test Execution phase the test cases are executed in the testing environment, while execution of the test cases the Testing team may find bugs which will be reported, bugs are fixed by the developer and they are retested by the Testing Team.

Entry Criteria: Test Plan document, Test cases, Test data, Test Environment

Exit Criteria: Test case execution report. Defect report, RTM

Activities:
a) Execution of Test Cases
b) Document test results, and log defects for failed cases
c) Update RTM – Map defects to test cases in RTM
d) Retest the Defect fixes
e) Track the Defects to Closure

Deliverables:
a) Test execution Report
b) Updated test cases with results
c) Completed RTM with execution status
d) Opened and Closed Bug Report/s

vi) Test Cycle Closure

> Testing team will meet, discuss and analyze testing artifacts and evaluate Test cycle completion criteria. Identify strategies that have to be implemented in future and taking lessons from the current test cycle.

Entry Criteria: Test case Execution report and Opened and closed Defect Reports

Exit Criteria: Test Closure Report signed off by client, Test Metrics

Activities:
a) Evaluate Test Cycle completion criteria

b) Prepare test metrics
c) Documentation of the learning from the project
d) Prepare Test closure report

Deliverables:
a) Test Closure report
b) Test metrics

Note: This Software Testing Life Cycle or Software Test Process is for conducting Software
Testing at System Testing Level and It is Manual Testing Process…

System Test Plan Documentation

I) Test Planning Phase

Input or References:

Requirement Specifications
Project Plan Document
Test Strategy Document
Global design document
Low Level Design document
Development and Test process standards
Corporate standards and guidelines

Tasks:

Understanding & Analyzing the Requirements
Risk Analysis
Test Strategy Implementation
Test Estimations (in terms of time, budget, resources, and scope of the project)
Team formation
Test Plan documentation
Configuration management planning
Creating Traceability matrix
Defining test environment setup

Output:
Test Plan document

Test Plan Template

1) Test Plan ID:

Some type of unique company generated number to identify this test plan.

2) Introduction:

Describe the purpose of the Plan, possibly identifying the level of the plan (System Test Plan etc.). This is essentially the executive summary part of the plan.

3) Test Items:

These are things you intend to test within the scope of this test plan.

4) References:

List all documents that support this test plan. Refer to the actual version/release number of the document as stored in the configuration management system.

5) Features to be Tested:

This is a listing of what is to be tested from the Users viewpoint of what the system does. This is not a technical description of the software, but a User's view of the functions.

6) Features not to be Tested:

This is a listing of what is NOT to be tested from both the Users viewpoint of what the system does and a configuration management/version control view. This is not a technical description of the software, but a User's view of the functions.

7) Test Approach:

This is your overall test strategy for this test plan; it should be appropriate to the level of the plan (master, acceptance, etc.) and should be in agreement with all higher and lower levels of plans. Overall rules and processes should be identified.

8) Entry Criteria:

It describes when to start Testing

9) Exit Criteria:

It describes when to stop testing

10) Suspension Criteria:

It describes when to stop testing temporarily.

11) Roles & Responsibilities

Team Lead or Test Lead and Team members Roles and Responsibilities.

12) Schedule:

Schedule for all Test activities in this Software Test Process.

13) Training:

> Training on the application/system (Domain Training)
> Training for any test tools to be used.

14) Test Environment / Lab:

It describes Require Hardware and software for setting-up Test Environment or Test Lab.

15) Test Deliverables:

Lists out that what is to be delivered as part of this plan?

16) Approvals

Who can approve the process as complete and allow the project to proceed to the next level.

17) Glossary:

Define terms and acronyms used in the document, it can be used to understood the terms used in this plan.

Software Test Plan Documentation

A Sample Test Plan Document for Internet Banking Application

1) Test Plan ID: IBS_ST_TP_001

2) Introduction:

It is System Test Plan for Interment Banking System, internet web application, provides access to Account holders and guest users from any ware in the world. It has two interfaces one is Admin interface another is User interface. Admin can be accesses by Bank authorized users, user interface can be accessed by Bank account holders and guest users.

The propose of the system (Application) is to provide bank information and services online (through Internet), Bank account holders can get Banking services from any ware, without visiting the Bank branches.

3) Test Items:

Admin Interface:

Master Data

User Management

Reports
etc…

User Interface

Information

Personal Banking

Corporate Banking

Business

Etc…

4) References:

Requirements

Project Plan

Test Strategy

—
Use cases (If available)

High Level Design doc

Low Level design docs

Process guide line doc

Prototypes

5) Features to be Tested:

a) Admin Interface:
i) Master Data

1) Add new branch, Edit Branch /Delete Branch
2) Add new ATM
3) Add new loan type
4) Add new account type
5) Add new deposit type
….
ii) User Management

1) Create new user
2) Edit user
3) Delete user
etc…..
iii) Reports
1) Branch wise report
2) User wise report
3) Day, month, yearly reports
4) Service wise report (only loans, only new account. fixed deposits)b)

User Interface:
i) Information

1) Branch locators
2) ATM locators
3) Loans information

4) Bank history
5) Bank financial details
6) Fixed deposits information
7) Calculators
etc…

ii) Personal Banking
1) Login
2) Balance enquiry
3) Bill payment (utilities, subscriptions)
4) Fund transfer (transfer to same bank, others banks)
5) Statement generation (mini statement, detailed report)
etc…

iii) Corporate Banking
1) Add user, Edit user, Delete user
2) Balance enquiry
3) Money transfer
4) Payroll
5) Reports
etc..

6) Features not to be tested:

NA

7) Entry Criteria:

a) Test Design
Team formation, Responsibilities, Schedule,
Requirements, Test Case Template etc…

Training on Domain, on Automation tools

b) Test Execution:
Readiness of Test Lab
Readiness of AUT
Requirements
Test Case docs
Test Data
Defect Report Template
Etc…

8) Exit Criteria:

All possible test cases executed

Maximum defects fixed, Final Regression performed successfully

Confidence on Test process

Time Limitations

Budget Limitations

9) Suspension Criteria:

Show-Stopper bug found

Supplier issues

Vast changes in requirements

If resolving defects are more

10) Roles & Responsibilities

Sno Name Role Responsibilities Remarks
1 Kareemulla SK Test Lead Test Planning, guidance, Monitoring and Test control
2 Venkat Rao P Sr. Tester Test Data Collection, Generating Test Scenarios.
3 Swapna DK Tester Test Case Documentation, Test execution, defect reporting and tracking for Admin module.
4 Srinivas V Tester Test Case Documentation, Test execution, defect reporting and tracking for Personal Banking module.
5 Suneetha B Tester Test Case Documentation, Test execution, defect reporting and tracking for Corporate Banking module.
6

11) Schedule:

Sno Task Days Duration Remarks
1) Understanding & Analyzing Requirements. 5 2nd July to 6th July
2) Review Meeting 01 9th July
3) Generating Test Scenarios. 10 11th July to 22nd July
4) Reviews 02 25th July to 26th July
5) Test Case Documentation 40 29th July to 12th August
6) Reviews 04 14th August to 18th August

7) Test Data collection 06 20th August to 26th August
8) Reviews 01 28th August
9) Verifying Test Environment setup 01 29th August
10) Create Test batches 02 30th, 31st August
11) Sanity Testing 01 3rd September
12) Comprehensive Testing 25 4th September to 2nd October
13) Sanity Testing 01 3rd October
14) Selecting Test cases 02 4th and 5th October
15) Regression Testing 05 8th October to12 th October
16) Sanity Testing 01 15th October
17) Selecting Test cases 01 16th October
18) Regression Testing Cycle 2 04 17th October to 22nd October
. . . .
. . . .
. . . .
28) Final Regression 08 19th November to 28th November
29) Evaluating exit criteria 01 or 02 29th, 30th of November
30) Collecting all artifacts 02 3rd, 4th December
31) Test Summary Report 01 5th December
Note: Regression Testing depends on Application and strength of Development team.

12) Training

Training Program on Banking Domain

Test Automation Training using Selenium Tool

13) Risks & Mitigation

Team member's issues

Vendor issues

Time

Budget

14) Test Environment / Lab

Application Type: Web Application, Internet and Public

Server side:

Windows Server 2016

UNIX server

Ms Exchange Server

a) Web Server, b) EDP, 3) Data storage

Bugzilla Tool

VSS

MS Office

Selenium Tool etc….

Browser MS Edge

Client side:

Windows 10

VSS

MS Office

Selenium

Etc…

AUT Environment

.NET (C#, VC++, ADO)

IIS -web server

COM+ -App server

SQL Server 2017 for Database server

---

15) Test Deliverables

Test Plan,

Review Reports,

RTM

Test Scenario docs

Test Case docs

Test data

Opened, Closed Defect Reports

Test Summary Report

16) Approvals

SNO Task/s Author /Role Date & Signature
1) Test Plan Documentation Kareemulla SK (Test Lead)
2) Review Hari Prasad (QA Analist)
3) Approval Vinod Rao (Project Manager)

17) Glossary

AUT -Application Under Test
.
.
.
PIN -Project Initiation Note
.
.
.
SRS- Software Requirements Specification
.
.
.

Software Test Case Documentation

> What is Test Case?

A set of input values, execution preconditions, expected results and execution post conditions, developed for a particular objective or test

condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

> Test Case Template

Test Case Template may vary from one company to another, sometimes one project to another in the same company. If you familiar with anyone Test Case then you can easily write test Cases using any type of Test Case Template.

I am going to introduce a Model Test Case Template,

Sample Test Case Template:

1) Test Case Id: a Unique Name to identify the Test Case

2) Test Case Title: Title or Name of the Test Case

3) Module Id: Name of the main module or sub-module being Tested

4) Precondition: Status before the Test Case Execution

5) Test Steps: Steps for Executing the Test Case

6) Test Data: Required Input Data for Executing the Test Case

7) Expected Result: Expected Result as per Requirements

8) Post-condition: Status After Test Case Execution

9) Actual Results: (During & After Test Case Execution)

10) Status: Pass or Fail Status after comparing Expected Results with Actual Results

11) Comments: (Optional)

Test Case Writing

1) Test Case Id: U_TC_001
2) Test Case Title: Gmail Login Functionality
3) Module Id: User
4) Precondition: Valid Gmail Id and Password

5) Test Execution Steps:
i) Launch the Browser and Navigate to www.Gmail.com
ii) Write / Type Valid Gmail ID in the "Edit Box"
iii) Click on "Next" Button
iv) Write / Type Valid Password in the Edit Box
v) Click on "Sign In" Button

6) Test Data:
i) Gmail Id: gcr1977
ii) Password: gcbannureddy

7) Expected Result:
i) Gmail Home Page is Opened
ii) Gmail ID is Entered
iii) Another Web page is opened which gives an option to Enter Password
iv) Password is Entered
v) User is logged in to Gmail.

8) Post-Condition: User Gmail Page opened with all options like Compose, Delete etc…
9) Actual Results: (* During Test Case Execution)
10) Status: (* After Test Case Execution)
11) Comments:

> Our Test Case after the Execution

1) Test Case Id: U_TC_001
2) Test Case Title: Gmail Login Functionality
3) Module Id: User
4) Precondition: Valid Gmail Id and Password
5) Test Case Steps:
i) Launch the Browser and Navigate to www.Gmail.com
ii) Write / Type Valid Gmail ID in the "Edit Box"
iii) Click on "Next" Button
iv) Write / Type Valid Password in the Edit Box
v) Click on "Sign In" Button

6) Test Data:
i) Gmail Id: gcr1977
ii) Password: gcbannureddy

7) Expected Result:
i) Gmail Home Page is Opened
ii) Gmail ID is Entered

iii) Another Web page is opened which gives an option to Enter Password
iv) Password is Entered
v) User is logged in to Gmail.

8) Post-Condition: User Gmail Page opened with all options like Compose, Delete etc…

9) Actual Results:
i) Gmail Home Page Opened
ii) Gmail ID Entered
iii) Another Webpage opened and It has an option to Enter Password
iv) Password Entered
v) User logged in to Gmail.

10) Status:
i) Step 1: Pass
ii) Step 2: Pass
iii) Step 3: Pass
iv) Step 4: Pass
v) Step 5: Pass
Test Case: Pass

11) Comments: Smooth Execution, and Fields also User Friendly

Note 1: You can some fields to this Test Case if required, Ex: Tester's Name, Environment, Date of Creation, Date of Execution etc…

Anyhow Test Case Template may vary form one company to another and one project to another, based scope of the Project usually we can Select Test case Template.

Note: Usually we write Manual Test Cases in Excel File using our Company prescribed format, if we use any Test Tool like, ALM, Jira etc… then they provide Test Case temple and User/Tester can document Test Cases, and one more thing some Test Tools provide options to customize the Test Case Template.

Defect Reporting

Application Life Cycle (ALM)

Development Phase Testing Phase Production Phase

Error Defect Failure

Mistake Bug

Fault

We have 3 phases in Software Application Life Cycle

a) Development Phase

In this phase If developers find any mismatch, they call it as Error or Mistake.

b) Testing Phase

In this phase If Testers find any mismatch, they call it as Defect or Bug or Fault.

c) Production Phase

In this phase If End users find any mismatch, they call it as Failure.

Note: Terminology vary from one phase to another.

Defect Management:

Defect Reporting, Defect Tracking, and Status Tracking is called Defect Management.

Some companies use Manual Process (Excel workbook), and some companies use Tool based process
for Defect Management.

Tools Examples:

Bugzilla / Issue-Tracker / PR-Tracker etc…

Jira, QC

Model Defect Report Template:

1) Defect Id: any unique name for Identifying the Defect (Alphanumeric)

2) Defect Description: Details about the Defect

3) Test Case Id: Corresponding Test Case Id for tracking

4) Tester: Tester's name (who found the Defect)

5) Product Version: Version of the Product on which defect was found

6) Build Version: Version of the Build on which defect was found

7) Priority: Importance of the Defect based on Business /Customer

8) Severity: Importance of the Defect based on Functionality

9) Status: Status of Defect

10) Reproducible or not: Yes / No

If Reproducible:
Steps:

If not Reproducible:
Attachments

11) Reporting to: Corresponding Developer

12) Remarks : Comments (Optional)

Status: Status of Defect

New: Tester provides new status while Reporting (for the first time)

Open: Developer / Dev lead /DTT opens the Defect

Rejected: Developer / Dev lead /DTT rejects if the defect is invalid or defect is duplicate.

Fixed: Developer provides fixed status after fixing the defect

Deferred: Developer provides this status due to time etc…

Closed: Tester provides closed status after performing confirmation Testing

Re-open: Tester Re-opens the defect with valid reasons and proofs
---
Note: Defect Reporting Template vary from one company to another

If we use Tool for Defect management, every tool provides their own template.

Defect Reporting Process

Defect Reporting Process vary from one company to another.

a) Small scale Company

Tester -> Developer

b) Medium scale Company

Tester -> Test Lead -> Development Lead -> Developer

c) Large scale Company

Tester -> Test Lead -> DTT -> Development Lead -> Developer

A Sample Defect Report

1) Defect Id: FR_Usr_Df001

2) Defect Description: Agent Name accepting Numeric values

3) Test Case Id: FR_Usr_Tc-004

4) Tester: Kanaka Rao

5) Product Version: 1.0

6) Build Version: 1.0

7) Priority: Medium

8) Severity: High

9) Status: New

10) Reproducible or not: Yes

Steps:
i) Launch the Application
ii) Enter Numeric values into Agent Name field
iii) Enter valid Password
iv) Click on default(OK) button

11) Reporting to: xyz

12) Remarks : Comments (Optional)

Severity:

Severity Levels depends on Company strategy

a) 5 Level Severity

Urgent

Very High

High

Medium

Low
————
b) 3 Level Severity

High

Medium

Low

Priority

Priority Levels depends on Company strategy

a) 5 Level Priority

Urgent

Very High

High

Medium

Low
————
b) 3 Level Priority

High

Medium

Low

Defect Life Cycle

Different Flows of the Defect

a) New -> Opened -> Fixed -> Closed

b) New -> Opened -> Rejected -> Closed

c) New -> Opened -> Fixed -> Re-opened -> Fixed -> Closed

d) New -> Opened -> Deferred

e) New -> Opened -> Rejected -> Re-opened -> Fixed -> Closed

Etc…

Test Summary Report

A test summary report is a Quality work product / Test Document that formally summarizes the
results of all testing.

Test Lead or Test Manager prepares this document at end of the Testing, means in Test Closure
phase (Last phase in STLC/Software Test Process.

Purpose:

To enable Project management and Customer to know the status of testing status of the
project and Application Quality Level.

Advantages:

All stake holders of the Project able to get project test status, Application Quality status
and they can take corrective actions (if required)

Guidelines:

A Test summary report can be generated for every software release
• It should have metrics, charts and table forms, if possible
• It has to summarize all test activities based on Quality work products.
Test Summary Report Template

Introduction:

Test Items:

Reference documents

Target Audience

Test Summary

a) Test Suite Information:
• Number of test suites planned.
• Number and percentage of test suites implemented.
• Number and percentage of test suites executed.

b) Test Case Information:
• Number of test cases planned.
• Number and percentage of test cases implemented.
• Number and percentage of test cases executed.
• Number and percentage of test cases passed.
• Number and percentage of test cases failed (total and by severity).

c) Defect Report Information

Number of defects found in comprehensive Testing

Number of Test Cases selected for Regression Testing Cycle1
Number of defects found in Regression Testing Cycle1

Number of Test Cases selected for Regression Testing Cycle 2
Number of defects found in Regression Testing Cycle 2

Number of Test Cases selected for Regression Testing Cycle 3
Number of defects found in Regression Testing Cycle 3
.
.
.
Number of Test Cases selected for Final Regression.
Number of opened defects in this release.

Approvals

A Sample Test Summary Report

Introduction:

It is for Internet Banking System Application Version 2.0, IBS (Internet Banking System) has
Information, Personal banking, Corporate Banking, Master Data, User Management and Reports
modules already. Now in this release some features added to Personal Banking Module and One
New module "Small Business" added.

Test Items:

Personal Banking
New Business
Reports

Reference documents:

Test Plan, Test Case documents, Opened and Closed Defect Reports, Metrics docs, Review
Reports.

Target Audience:

Project Manager, Release Team, Maintenance Team and Customer (End Users).

Test Summary

a) Test Suite Information:

- Test suites planned: 42
- Test suites implemented: 39
- Test suites executed: 38

b) Test Case Information:

- Test cases planned: 983
- Test cases implemented: 978
- Test cases executed: 967
- Test cases passed: 940 (Final)
- Test cases failed: 14 (Final)
- Test Cases pending: 3

c) Defect Report Information

Defects found in comprehensive Testing: 148
Test Cases selected for Regression Testing Cycle1: 320
Defects found in Regression Testing Cycle1: 96
Test Cases selected for Regression Testing Cycle 2: 213
Defects found in Regression Testing Cycle 2: 42
Test Cases selected for Regression Testing Cycle 3: 107
Defects found in Regression Testing Cycle 3: 26
.
.
.
Test Cases selected for Final Regression: 512
Number of opened defects in this release: 14

Approvals

Name Role Responsibility Date & Signature
abcd Test Lead Test Summary 20-06-2018
Report preparation
xyz QA Analyst Review 25-06-2018
pqr Project Manager Approval 29-06-2018
* On 29-06-2012 Test Summary Report base lined (Finalized).
****