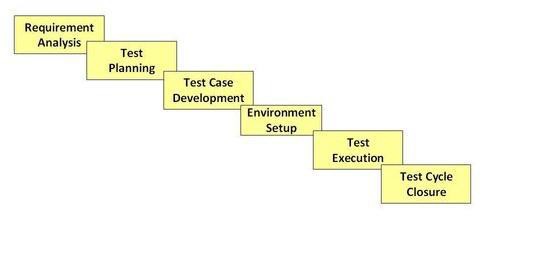
**Software Testing Life Cycle**

Software Testing Life Cycle (STLC) identifies, what test activities to be performed to complete those tasks. It used to find whether specified requirements are working as expected. These activities are conduct by testing team in sequential order to improve the quality of the software. STLC differs from organizations.

Software testing life cycle has the following phases:

* Requirement Analysis
* Test Plan
* Test Case Development
* Test Environment
* Test Execution
* Test Closure.



# Requirement Analysis:

During this phase, QA team goes through the requirement document in order to identify the testable requirements. If requirement is not clear or understandable, then QA team follow up with Business Analyst, project Manager, System architect etc. to better understand of requirement. Once we are done with understanding the complete requirement, QA team will document the acceptance criteria and approved by the customer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Activities** | **Exit Criteria** | **Deliverables** |
| Requirement document  Acceptance criteria  Application architectural will be | Prepare the list, which we clarified from stack holders.  Identify types of testing and techniques can be performed. | RTM approval from the test lead.  Test automation feasibility report is signed off by the client | RTM document  Automation feasibility report |

|  |  |  |  |
| --- | --- | --- | --- |
| well defined. | Check for the automation feasibility for project  Prioritizing the feature and need focus testing  Prepare RTM (Requirement Traceability Matrix). |  |  |

# Test Planning:

After completion of requirement analysis, we will do test plan. In this phase, senior QA Manager or QA lead will prepare the **test plan**. As per the document, we can estimate the effort and cost of the project.

# Test Plan:

Test plan is a strategic document, which contains relevant information that describes how to perform testing on an application in an effective, efficient and optimized way.

# Introduction:

* **Objective**:

It describes the test plan preparation i.e., why are we preparing test plan.

# Reference Document:

The list of all the documents that are referred, while preparing the test plan.

# Ex:

* Project Plan, SRS Document.
* Basically Project Plan is developed by Project Manager.

# SCOPE OF TESTING:

* + **Features to be tested:**

The list of all the features that are within the scope and planned for testing.

# Features not to be Tested:

The list of all the features that are not planned for testing. Usually based on the following scenarios it happens.

* + - Out of Scope Features.
    - Low Risk Features.
    - Features that are planned to be incorporated in future.
    - Features that are skipped based on the time Constraints.’

# Testing Methodologies:

Depending upon the application, we decided what type of testing should conducted to the various features of the application. We must ensure that what type of testing is define and described in the testing methodologies, so that everybody can understand in the team.

# Ex:

We have to test **www.amazon.com**; we do the following types of testing.

|  |  |  |
| --- | --- | --- |
| Smoke testing | Functional testing | Integration testing |
| System testing | Adhoc testing | Compatibility testing |
| Regression testing | Globalization testing | Accessibility testing |
| Usability testing | Performance testing |  |

For standalone application, like **PhotoShop**, we do the following types of testing.

|  |  |  |
| --- | --- | --- |
| Smoke testing | Functional testing | Integration testing |
| System testing | Adhoc testing | Compatibility testing |
| Regression testing | Globalization testing | Accessibility testing |
| Usability testing | Reliability testing | Recovery testing |

Installation / Uninstallation testing

# Approach:

It describes how you test the application in future.

1. By writing high level scenarios
2. By writing flow graphs

# By writing high level scenarios

For ex, we are testing [**www.gmail.com**](http://www.gmail.com/)

I) Login to Gmail – send a mail and check whether it is in Sent Items page.

1. Delete mail – select mail and click on delete button and check whether it was deleted.
2. …..

It explain the approach to be taken to test the product. Only for the critical features, we will write a few very high level scenarios. We do not cover all scenarios here. That is the job of the respective Test Engineers for whom the features have been allocated.

1. **By writing flow graphs**

**Gmail Mail**

**Login**

**Compose Sent Items Inbox Read mail**

**Reply Reply all Forward**

**Logout**

We write flow graphs because of the following advantages,

* 1. Merging is easy
  2. Coverage is easy

Flow graphs are written, because writing high-level scenarios is time consuming.

# Risk & Mitigation Plan:

**Risk:**

It is a sudden occurrence of chances of getting loss, If the assumptions fail in the project, risks are involved.

# Mitigation Plan:

This plan used to reduce the risks in the project. At least to reduce the percentage from 100% to 20 %.

# Ex:

**Risk over load:**

System can handle only 1000 users at one time, if more than 1000 users want to access the system, it may denial of service.

# Mitigation Plan:

An extra system to be maintained, so that it could share the users with the main system.

# Roles and Responsibilities:



Test Manager

Test Lead

Test Lead

Test Lead

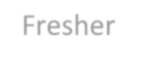
Test Lead



Senior Test Engineer



Junior Test Engineer



Fresher

Usually test manager writes the test plan, if it is a big project.

If there are small projects, then Test Manager allocates each project to each Test lead. The Test lead writes the test plan for the project, which Test Manager is allocated.

# Test Manager:

* Writes or reviews test plan
* Interacts with customer, development team and management
* Sign off release note
* Handle issues and escalations

# Test Lead:

* Writes or reviews test plan
* Interacts with development team and customers
* Allocates work to test engineers and ensure that they are completing the work within the schedule time.
* Consolidate reports sent by Test Engineers and communicate it to development team, customers(if it is a time & material project) and management

# Test Engineer 1

* Review test plan
* Write test cases for trend analysis
* Asset survey
* Write traceability matrix
* Review test cases written for modules
* Execute test cases written for trend analysis, asset survey, registration (old module developed in previous release. Adding trend analysis and asset survey has affected. Old module has been affected. So do regression testing)
* Perform compatibility testing using Internet Explorer, Mozilla Firefox and Google Chrome in Windows XP and Windows Vista
* Prepare test execution report and communicate it to Test lead.

# Test Engineer 2

* Set up and install the product
* Identify test cases to be automated
* Automate identified test cases using QTP
* Execute and maintain automation scripts

# Schedules:

It describes, when exactly each activity should start and end. The main aim of schedule is to mention the exact date for every activity.

* System study – Date (Estimated date).
* Write test cases – Date (Estimated date).
* Execute test cases – Date (This date is given by development team).
* Release date – Date (This date is given by Client).

For each activity, there will be a starting date and ending date. Specific date will be announce for every build and type of testing for each build.

# Defect Tracking:

It mainly deals with how to track the defects and how to communicate with the development while testing the application and sending the defects reports.

# Test Environment:

**Hardware:**

**Server**: sun starcat15k

From these server, testing team take the application for testing

# Client server:

* Machines with following configurations,
* Processor: Intel i5.
* RAM: 4GB.

# Software:

**Server:**

* OS: Linux.
* Web Server: Tom Cat.
* Application Server: Web sphere.
* Database Server: Oracle (or) MS – SQL Server.

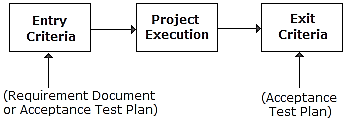
The test engineers to test the application use these serves.

# Client:

OS: Windows XP, 7,8, 10,11.

Browsers: internet Explorer, Mozilla Fire Fox, Google Chrome.

# Entry and Exit criteria:



**Entry criteria:**

* It ensures that the proper environment is in place to support the entire system test process. Item must meet in entry criteria
* All test hardware platforms must have been successfully installed, configured and functioning properly.
* All standard software tools including testing tools must have been successfully installed and functioning properly.
* All documentation and design of the architecture must made available.
* All personnel involved in the system test effort must trained in tools to be used during testing process.
* A separate QA environment (with its own web server, database and Application server instance)

must be available.

* Proper test data is available.

# Exit criteria:

* It ensures that the project application has been satisfactorily completed, before exiting the system test stage and clarifying the application as complete.
* Application must provide the required services.
* Ensure all application documentation has been completed and is up to date.
* 100% of all Priority 1 and priority 2 bugs must be resolved.

# Test automation:

* Which feature to be automated?
* Which feature not to be automated?
* Which is the automation tool we are planning to use?
* What is the automation framework you are planning to use?

**Test Metrics**:

A Metric is a quantitative measure of the degree to which a system component, or process possesses a given attribute*.* It used to measure the quality of the project.

* It used to measure the test coverage.
* It used to measure the defect density.
* It used to measure the defect removal efficiency.

# Deliverables:

It is the output from the testing team. It contains what we will deliver to the customer at the end of the project. It has the following sections,

* + 1. Test Plan
    2. Test Cases
    3. Test Scripts
    4. Traceability Matrix
    5. Defect Report
    6. Test Execution Report
    7. Test Metrics
    8. Release Note

# Release notes:

Release notes is a document, which prepared by test manager during release of the project. The release note includes.

* + - * List of pending bugs.
      * List of features added, modified, or deleted.
      * Platforms (browsers, os, hardware) which the project is tested.
      * Version of the software etc.

# TEST CASE DEVELOPMENT:

Testers will involve in this phase and they will understand the SRS and identify the scenarios and writes / design the test cases, and create test data/scripts for testing if required. After writing test cases, it is

reviewed by peer members and test lead.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Activities** | **Exit Criteria** | **Deliverables** |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement document  Test plan document  Automation feasibility report | Create test cases  Create test data/test scripts.  Verification of test cases | Review test cases and approved  Review test data/ scripts and approved | Test cases Test data  Test Automation Scripts. |

# Sample Test Case Template:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project Name:yahoo | | | | | | | | | | |
| Module Name:login | | | | | | | | | | |
| Author: xxxx | | | | | | | | | | |
| Requireme nt Id | Test Case Id | Category Type | Pre- requisi te | Descriptio n /  Test Steps | Expecte d Value | Test Data | Actua l Value | Resul t | Build No | Priority |
| Yahoo\_logi n\_01 | Logi n1 |  |  |  |  |  |  |  |  |  |

**TEST ENVIRONMENT**:

Test environment is the environment where the testing team will test the application. Test environment is prepared by understanding system architecture, hardware, software. Testing team is not involved in preparing test environment, based on company developers or customer will prepare the test environment. Testing team conducts smoke test to check whether the basic functionalities are working fine or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Activities** | **Exit Criteria** | **Deliverables** |
| Test plan document  Smoke test cases are available  Test data is available | Based on the requirements, list the components in software environments  Creation of test data  Setup test environment Smoke test execution | Test environment is implemented as per the check list.  Smoke test reviews and signed off | Smoke test results  Test data is created  Setup environment is ready |

# TEST EXECUTION:

In this phase, testers execute test cases, which were prepared earlier. Testers find bugs and reports to development team, and they will fix the bug and resend to testing team and testers do retest to verify it

is free to live. Testers will observe the actual behavior of the application. In this phase, the Test Engineer will compare the expected values with actual values and if both are matching, they will decide the result as pass otherwise fail. If at all, the test Cases not executed, then they will decide result as blocked.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Activities** | **Exit Criteria** | **Deliverables** |
| Test plan Test cases  Test environment | Execution of test cases Reporting test results Defect tracking Retesting  Closure of defects. | Test cases executes successfully  Defect tracking to closure | Test execution report Bug report  Updated test cases and results |

# Bug Tracking:

It is a process in which the defects are Identified, Isolated and managed. Bug (or) Error (or) Defect = Problem

* + - * If a problem is identified by Developer, then it is called as Error.
      * If a problem is identified by Test Engineer, then it is called as Defect.
      * If a Defect is identified by Test Engineer and is accepted by Developer then it is called as Bug.
      * If Customer finds any problems at the enduser side while using the application , then we call it as Failure.

# Bug/Defect Life cycle:

**A diagram of a software process

Description automatically generated**

**Defect status**: It indicates presence status of the defect in development process. Different defect statues:

* + - * Open
      * Assign
      * Fixed
      * Closed
      * Reopen
      * Rejected
      * cannot be fix
      * Duplicate
      * Postpone
      * Not reproducible
      * Request for enhancement

# Explanation for Bug Life Cycle:

* + - * As soon as build comes as a test engineers execute the test cases
      * We are executing test cases means we are going to catch bugs.
      * After catching bug prepare report in project management tool like (Ex: QC, jira Etc.)
      * Now bug life cycle begins:

# Open:

* + Prepares report on project management tool.
  + Status changes as **New.**
  + Send it to development lead.

# Assign:

* + Development lead read the report understand the bug.
  + In project management tool status changes **New** to **assign**.
  + Identify the right developer assign the task to developer.

# Fix:

* + Developer read the report and understand the bug.
  + Go throw the code and fix the bug.
  + In project management tool status changes to **assign** to **fix**.

# Close:

* + After fixing the bug, deploy the fixed functionality in QA server.
  + Test Engineers re-test the bug to verify the bug got fixed or not.
  + Bug is fixed then status changes to **fix** - **close.**

# Reopen:

* + Bug is not fixed, then status change to **fix** - **reopen.**

# Rejected:

Whenever the developers feel it is not at all a defect then they will set the status as “Rejected”. Whenever the defect is rejected, the testers will once again check it, if at all they feel it is a defect then they will “Reopen it” otherwise “Close” it.

* + Because misunderstood of requirement.
  + Because improper installation of the product.
  + This cycle will goes on until the bug fix.

# Cannot be fixing:

Bug accepted by the development team, but development team takes time to fix the bug due to following reasons:

* + If cost of fixing bug is more than the cost of bug.
  + May be Critical bug.
  + Technology itself not supported.
  + If major/minor bug in the core feature might fix the bug, but it will impact on all other features.

# Duplicate:

* + Defect is already sent by another test engineer.
  + Because of common feature.
  + Someone else tested your module bug send it to development lead you may find bug later then we will get duplicate status.

# Postpone:

* + Developers accepted the bug, but they want to fix it later.
  + Because of they do not have a sufficient time to fix.
  + Because of customer want to changes in requirement.

# Not reproducible:

* + Because improper defect report preparing.
  + Because of using incorrect data formats, Developers are using different OS, Browsers and Testers are using some other OS, Browsers.
  + Because of inconsistent defect.

# Request for Enhancement:

Problem occurs, but it is not part of requirement, than it is called Request for enhancement.

**Severity**: severity of a bug is based on how much does it impact of the system. It reflects on how bad the bug is for the system. QA will decide the severity level.

**Priority**: How fast we should resolve the bug, should we fix it now? How difficult is it to resolve? It is done by test lead and project manager.

* + **High priority & Low severity**: G mail logo is not correct.
  + **Low priority & High severity**: Suppose one application, which generates some banking related reports, monthly, quarterly, yearly, weekly. Mistake in calculating yearly statement. It can be

fixed in next release.

* + **High Priority & High severity**: Mistake in calculating weekly report, so we should fix it in a time and release.
  + **Low priority & Low severity**: Spelling mistake.

# Defect Report Template:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project Name: | | | | | | | | | | | | | | | |
| Module Name: | | | | | | | | | | | | | | | |
| Author: | | | | | | | | | | | | | | | |
| Def | T | Issue | Reprod | Detected | | | | Defect | | Def | Fixed | | | | En |
| ect | es | Descri | ucible |  | | | |  | | ect |  | | | | cl |
| Id | t | ption | Steps |  | | | |  | | Sta |  | | | | os |
|  | C |  |  |  | | | |  | | tus |  | | | | ur |
|  | as |  |  |  | | | |  | |  |  | | | | e |
|  | e |  |  |  | | | |  | |  |  | | | |  |
|  | Id |  |  |  | | | |  | |  |  | | | |  |
|  |  |  |  | B  y | Dat  e | Buil  d | Versi  on | Severit  y | Priorit  y |  | By | Date | Buil  d | versio  n |  |

* + **Defect Id**:

Ids are provided for defects.

# Test Case Id:

Test Case Id based on which defect is found.

# Issue Description:

What exactly the defect is, will be clearly described.

# Reproducible Steps:

The list of all the steps followed by the Test Engineer to identify that defect.

# Detected By:

The name of the Test Engineer who has detected that defect.

# Detected Date:

The date on which the defect is identified.

# Detected Build:

The build in which the defect is identified.

# Detected Version:

The version in which the defect is identified.

Version will be changed (i.e., the number) only when there are some changes to the application.

# Suggestion:

Whenever the problems are related, to value of the application, then such type of problems are treated as suggestion.

# TEST CLOSURE:

This is the final activity done during the testing process, where the Test Lead will prepare the test summary report. Test Summary Report contains the following information:

* + Number of cycle of execution.
  + Number of Test Cases executed in each cycle.
  + Duration of each cycle.
  + Number of defects found in each cycle and etc.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Activities** | **Exit Criteria** | **Deliverables** |
| Test cases execution is completed  Defect report Execution report | Evaluate test coverage and software quality  Prepare test closure report  Analyze defect distribution by type and severity  Project documentation for future purpose. | Test closure is signed off  After the testclosure is signed off, then the product is released. | Report of test closure. Defect report |