**Software Test Strategies for ALM**

**Version 1.0 - April-2017**

Executive Summary

The key objectives of the document is to help CUB teams understand the basics of software testing processes. Provide simple yet robust software testing approach which is easy to implement and at the same time delivers best-in-class results.

Contents

[1. Software Test Strategy 5](#_Toc479147683)

[1.1 Test Strategy – An Introduction 5](#_Toc479147684)

[2. Test Strategies for Web/Mobile/Mainframe/Desktop Applications 7](#_Toc479147685)

[2.1 Test Strategy for Web Based Applications 8](#_Toc479147686)

[2.1.1 System Testing 8](#_Toc479147687)

[2.1.1.1 Smoke Testing 8](#_Toc479147688)

[2.1.1.2 Regression Testing 8](#_Toc479147689)

[2.1.1.3 Compatibility Testing 9](#_Toc479147690)

[2.1.1.4 Functional Testing 9](#_Toc479147691)

[2.1.1.5 Performance Testing 10](#_Toc479147692)

[2.1.1.6 Security Testing 10](#_Toc479147693)

[2.1.1.7 Automation Testing For Web Based Applications 11](#_Toc479147694)

[2.1.2 Integration Testing 11](#_Toc479147695)

[2.2 Test Strategy for Mainframe Applications 11](#_Toc479147696)

[2.2.1 Classification of Mainframe Testing 11](#_Toc479147697)

[2.2.1.1 Batch Job Testing 11](#_Toc479147698)

[2.2.1.2 Online Testing 11](#_Toc479147699)

[2.2.2 Mainframe Testing Methodology 12](#_Toc479147700)

[2.2.2.1 System Testing 12](#_Toc479147701)

[2.2.2.2 Shakedown/Smoke Testing 12](#_Toc479147702)

[2.2.2.3 Regression Testing 12](#_Toc479147703)

[2.2.2.4 Automation Testing For Mainframe Applications 12](#_Toc479147704)

[2.2.2.5 Functional Testing 12](#_Toc479147705)

[2.2.2.6 Integration Testing 13](#_Toc479147706)

[2.3 Test Strategy for Mobile Applications 13](#_Toc479147707)

[2.3.1 Functional Testing 14](#_Toc479147708)

[2.3.2 User Interface Testing 14](#_Toc479147709)

[2.3.3 Usability Testing 15](#_Toc479147710)

[2.3.4 Compatibility Testing 15](#_Toc479147711)

[2.3.5 Performance Testing 15](#_Toc479147712)

[2.3.6 Security Testing 15](#_Toc479147713)

[2.3.7 Interruption Testing 15](#_Toc479147714)

[2.3.8 Automation Testing 16](#_Toc479147715)

[2.4 Test Strategy for Desktop Applications 16](#_Toc479147716)

[2.4.1 Smoke and Sanity Testing 16](#_Toc479147717)

[2.4.2 GUI Testing 16](#_Toc479147718)

[2.4.3 Functional Testing 18](#_Toc479147719)

[2.4.4 Compatibility Testing 19](#_Toc479147720)

[2.4.5 Performance Testing 19](#_Toc479147721)

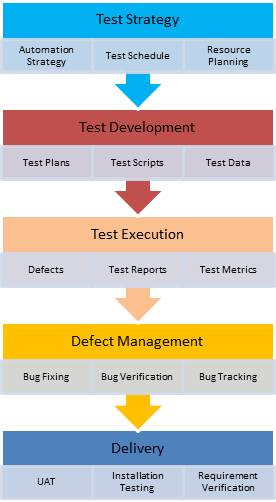
[2.4.6 Automation Testing 19](#_Toc479147722)

# Software Test Strategy

# Test Strategy – An Introduction

Test strategy is a plan for defining the testing approach, and it answers to questions like what you want to get done and how you are going to accomplish it. It is most important document for any QA team in software testing, and effectively writing this document is a skill that every tester develops with experience.

Test planning activities guides team to define test coverage and testing scope. It also aids test managers to get clear picture of the project at any instance. The possibility of missing any test activity is very low when there is a proper test strategy in place. Testing strategy plan must be conversed with the entire team so that the team will be consistent on approach and responsibilities



*Test Strategy in STLC*

# Test Strategies for Web/Mobile/Mainframe/ Desktop Applications

# Test Strategy for Web Based Applications

## System Testing

## Smoke Testing

The following features of web application should be tested in Smoke Testing:

* Test for – all the links in web pages, database connection, forms used for submitting or getting information from user in the web pages, Cookie testing etc.
* Test the outgoing links from all the pages to specific domain under test.
* Test all internal links.
* Test links jumping on the same pages.
* Test links used to send email to admin or other users from web pages.
* Test to check if there are any orphan pages.
* Basic testing on critical features

## Regression Testing

Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects in previously working modules of software/application. This type of testing is typically associated either as a ‘*challenge*’ or ‘*unimportant*’ by developers; but a good tester always enjoys breaking the software to scour out each and every fault from it.Below are few important points which needs to be considered while selecting regression scripts:

1. **Include the test cases which verify critical/core features of the application:**Prior to designing of the test cases figure out all critical scenarios/features of application. Ensure that, test case cover all functionality mentioned in requirement document. One can make use of traceability matrix to make sure that no requirement is left untested
2. **Include the test cases which have frequent defects:**Some areas in product are so error prone that they usually fail with a small change in code. We can keep track of failing test cases due to such areas throughout the product cycle and cover them in regression test suite.
3. **Prioritize the test cases for regression testing:**Prioritize the test cases depending on business impact, critical & frequently used functionalities. It is always helpful if some analysis is done to find out what test cases are relevant and what are not. It is a good approach to plan and act for regression testing from the beginning of project before the test cycles. One of the ideas is to classify the test cases into various Priorities based on importance and customer usage
4. **Categorize the selected test cases:**Regression testing becomes very difficult when the application scope is very huge and there are continuous increments or patches to the system. In such cases selective tests needs to be executed in order to save testing cost and time. Categorizing test cases makes this work easier. We can categorize them as
   * Reusable Test Cases:  
     It includes test cases which can be repetitively used in succeeding regression cycles. This can be automated so that set of test cases can be easily executed on new build.
   * Obsolete Test Cases:  
     These test cases are defect specific and can’t be used in succeeding cycles. Smart way is to use them is when relative defect occurs

## Compatibility Testing

A web application that behaves precisely as desired in one web browser might have some issues when run in another web browser – issues that could keep the vital functions of the application from working all together. There are many client components that can yield different results based on the web browser they are run in, such as Applets, JavaScript, Flash, AJAX requests, and the list goes on. For anyone who desires to have a professional looking website or application, Cross Browser Testing is essential. Not only are web browsers on personal computers (whether desktop or laptop) a concern here, but with the ever progressing advancement of tablets and phones with their web access capabilities, Cross Browser Testing of web applications must extend to mobile web browsers as well.

A detailed analysis should be done on the AUT (Application under test) itself to determine what parts of the application or if all of it has to undergo this. It is advisable that all of it be tested on multiple browsers, but again costs and time have to be considered. A good strategy is to perform 100% testing on one browser per platform and for the other just test the most critical/widely used functionality.

## Functional Testing

All software are designed and developed to meet and satisfy certain functional requirements. A functional requirement may be technical, business, or process based. Functional Testing is the process by which expected behavior of an application can be tested.

Web Functional testing is a process to test Web applications to detect if your web application is functionally correct. Web Functional testing involves carrying set of tasks and comparing the result of same with the expected output and ability to repeat same set of tasks multiple times with different data input and same level of accuracy. Implementing functional testing for your application early in the software development cycle speeds up development, improves quality and reduces risks towards the end of the cycle. Web Functional Testing can be performed both manually with a human tester or could be performed automatically with use of a software program.

***Note:*** All the new features and test cases as per test strategy need to be executed.

## Performance Testing

Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload. Performance testing is commonly conducted to accomplish the following:

* Evaluate against performance criteria
* Compare performance characteristics of multiple systems or system configurations
* Find the source of performance problems
* Support system tuning

Each application should have a suite of test cases(updated for every major release) which should be executed and the key application performance metrics should be noted and compared with previous release. If any significant change is noticed, the application tuning/refactoring should be considered. Automation tools such as MS Load Test or JMeter should be used for performance testing

## Security Testing

As more and more vital data is stored in web applications and the number of transactions on the web increases, proper security testing of web applications is becoming very important. Security testing is the process that determines that **confidential data stays confidential** (i.e. it is not exposed to individuals/ entities for which it is not meant) and users can perform only those tasks that they are authorized to perform (e.g. a user should not be able to deny the functionality of the web site to other users, a user should not be able to change the functionality of the web application in an unintended way etc.). Security(**Vulnerability**) Testing should cover the following security features of the application.

1. URL Manipulation
2. SQL Injection
3. Cross Site Scripting
4. Spoofing
5. Password Cracking

## Automation Testing For Web Based Applications

Tools Such as Coded UI and Selenium/HP UFT should be used for automating the above mentioned testing other than functional testing.

## Integration Testing

For a standalone web application, there could be no upstream/ downstream applications. In that case, functional testing should be carried in integration testing. How ever, if there is data flow between applications(upstream/downstream), then the data should be setup in upstream application and use that data for testing across all downstream applications. With Data driven framework, the test cases automated for System testing can be used for regression testing in integration testing.

# Test Strategy for Mainframe Applications

## Classification of Mainframe Testing

## Batch Job Testing

* Testing process involves executions of batch jobs for the functionality implemented in the current release.
* The test result extracted from the output files and the database are verified and recorded.
* Ex: It can be paper enrollment or enrollment on a third party website. The Offline data (also called as batch) will be entered into the company database through batch jobs. An input flat file is prepared as per the prescribed data format and fed to the sequence of batch jobs.

## Online Testing

* Online Testing refers to testing of CICS screens which is similar to testing of the web page.
* The functionality of the existing screens could be changed, or new screens could be added.
* Various applications can have enquiry screens and update screens. The functionality of these screens needs to be checked as part of the online testing.
* Ex: It is done on the member enrollment screen. Just like a web page the database is validated with data entered through the screens

## Mainframe Testing Methodology

## System Testing

## Shakedown/Smoke Testing

The main focus in this stage is to validate whether the code deployed is in the right test environment. It also ensures that there are no critical issues with the code.

## Regression Testing

Regression Testing is a common phase in any type of testing project. This testing in Mainframes ensures that batch jobs and the online screens which do not directly interact with the system under test (or do not come in the scope of requirements) are not affected by the current project release.

In order to have effective regression testing, a particular set of test cases should be shortlisted depending on their complexity and a regression bed (Test cases repository) should be created. This set should be updated whenever there is a new functionality rolled out into the release.

## Automation Testing For Mainframe Applications

HP UFT can be used for automating above mentioned testing for mainframe applications.

## Functional Testing

Below are the types of testing done as part of System Testing:

**Batch Testing** – This testing should be done by validating the test results on output files and data changes done by the batch jobs under testing scope and recording of them.

**Online Testing** – This testing will be done on the front end of the mainframe application. Here the application is tested for correct entry field like an insurance plan, interest on the plan, etc.

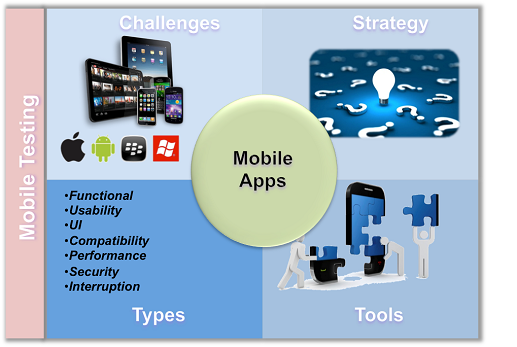
**Online-Batch Integration testing** – This testing will be done on the systems having batch processes and online application. The data flow and interaction between the online screens and the batch jobs is validated.(Example for this type of testing – Consider an update on Plan details like increase of interest rate. The change of interest is done on an update screen, and the balance details on the affected accounts will be modified only by a nightly batch job. Testing in this case will be done by validating the Plan details screen and the batch job run for updating all the accounts).

## Integration Testing

For a Mainframe applications, data flow between applications(upstream/downstream), then the data should be setup in upstream application and use that data for testing across all downstream applications. With Data driven framework, the test cases automated for system testing can be used for regression testing in integration testing.

# Test Strategy for Mobile Applications

It’s an era of mobile savvy users where thousands of mobile apps are booming with wide-ranging flavours of technologies. Today building app is not just about writing a good code, the success of app is largely driven by user experience. A successful app should have an aesthetically pleasing UI and should deliver best user experience on all devices and various form factors, of course there are other important factors to be taken into consideration as well.



The following are key factors to be considered before starting Mobile Application Testing:

* OS Version– Testing of a mobile app should be done on all major versions of mobile OS
* Compatibility Factor– A mobile applications may behave in a different manner on smartphone, tablet, and any other mobile devices, so the mobile application should be tested on all devices with which it is compatible

## Functional Testing

Functional testing should mainly cover the following testing:

* All the business requirements and functional features of the application
* Application workflow
* All error handling and error messages
* Other Languages and validations.

## User Interface Testing

The following UI aspects of the application should be tested

* Check all UI element alignments, font color, size, margins from screens.
* Check the UI design guidelines laid out by specific platforms are adhered.
* Check application compatibility with various screen form factors and resolutions

## Usability Testing

Usability testing should cover the ease with which the application can be used by user. The application should be checked for responsiveness, accuracy, screen navigation etc.

## Compatibility Testing

Test the application for compatibility for various platforms, platform versions and devices as per the requirements. The application is also tested for browser compatibility, network compatibility.

## Performance Testing

The following parameters should be checked in performance testing

* Application memory useage
* CPU and battery usage.
* Server connections

## Security Testing

The following features of the application should be tested:

* Authentication & Authorization
* Data protection, encryption/decryption of data while communicating with servers

## Interruption Testing

The following features of the application should be tested:

Interruption testing is used to check app behavior during interruption from notification or other apps like, phone calls, messages, calendar events, camera, alarm, battery notification, other app notifications, voice messages.

## Automation Testing

Mobile testing automations important to improve the testing Using above techniques to achieve maximum test coverage, it is extremely important for QA teams to automate the testing. Automation will also reduce time, cost and help achieve better quality. Automation can be achieved using various tools available in market and on cloud.

# Test Strategy for Desktop Applications

Desktop applications run on personal computers and work stations, a specific environment acts as a baseline for your test plan. We can test the complete application broadly in categories like Graphical User Interface, Functionality, Load,  etc. A desktop application is usually used by a single user at a time and needs to be installed as an exe file hence highlighting the need for installation testing.

## Smoke and Sanity Testing

The main focus in this stage is to validate the following:

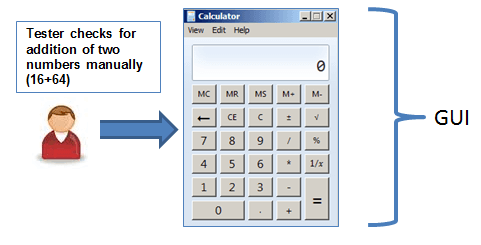
* Installation of the application is successful
* All the features of the release and bug fixes working as expected(validate at high level).
* Ensures that there are no critical issues with the code.

## GUI Testing

GUI testing can be done through three ways:

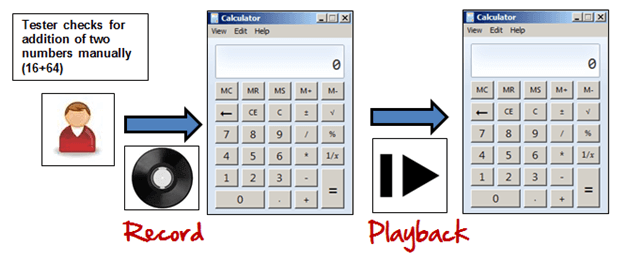
**Manual Based Testing**

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in the business requirements document.

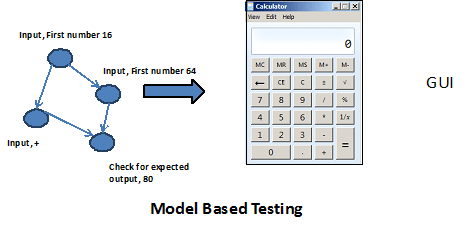


**Record and Replay**

GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured by the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - [QTP](http://www.guru99.com/quick-test-professional-qtp-tutorial.html)/Coded UI.



**Model Based Testing:**



A model is a graphical description of system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements. Following needs to be considered for this model based testing:

* Build the model
* Determine Inputs for the model
* Calculate expected output for the model
* Run the tests
* Compare the actual output with the expected output
* Decision on further action on the model

Some of the modeling techniques from which test cases can be derived:

* Charts - Depicts the state of a system and checks the state after some input.
* Decision Tables - Tables used to determine results for each input applied

Model based testing is an evolving technique for the generating the test cases from the requirements**.** Its main advantage, compared to above two methods, is that it can determine undesirable states that your GUI can attain.

## Functional Testing

Apart from the **business requirements**, the following system requirements should also be tested:

* Installation Testing (Upgrade/Downgrade)
* Testing with multiple user accounts
* Check for broken links
* Warning messages

## Compatibility Testing

* Testing the application on different hardware/operating systems

## Performance Testing

* Continuous use of application for very long period of time. Record the metrics
* Memory useage during initial useage of application and long period of application shouldn’t vary much.

## Automation Testing

Coded UI/HP UFT can be used for automating above mentioned testing for desktop based applications.