CHAPTER 1

INTRODUCTION

1.1 Company Profile :

Source Code Technology

Source Code Technologyis a leader in Software Development and empowers IT individuals with competitive advantage.Source Code Technology is an Indian Software Development Company. A rapidly growing software company with a team of experienced intellectuals working in various technologies. We are committed to the qualitative, efficiency, innovativeness and timeliness of our deliverables with high focus on maximum customer satisfaction. Source Code Technologyis a high end full service IT solution Company based in India. Today we are comprised of a team of programming technicians, designers, and marketing executives- selectively chosen to lead our clients in their IT solutions.

Source Code Technologyhas grown from strength to strength in both our Business and Software Solutions arena. From our IT Consulting as well as Custom Application Development, Web Development and E-commerce all of which help our customers with their diverse yet demanding needs. We are geared towards generating business value to the companies by providing expertise personnel and software services.

Our Services

Source Code Technology provides various Software Development services.We, at Source Code Technology strongly believe that technology is a true business enhancer and you should not implement technology for the sake of it. That's why; we help you make best use of information technology.

1.2 About the System under Test :

Existing System:

DRUG procurement is an important part of medical management in hospitals for keeping appropriate drug inventory levels to support medical activities. However, in existing approaches of drug procurement planning, drug requirements are mainly estimated based on past experiences, which often leads to shortage of some drugs and overstocking of others the latter can greatly increase the cost and cause waste of inventory and human resources, while the former can significantly decrease the ability of medical services.

1. The consumption method, which uses data on medicine consumption, gives in many instances the most accurate prediction of future needs.

2. The morbidity method quantifies the theoretical quantity needed for the treatment of specific diseases.

3. In emergency situation not provide the location of medicals which is near by the user.

LIMITATIONS OF EXISTING SYSTEM:

- Lack of security of data.
- Time consuming.
- Consumes large volume of paper work.
- Manual work.

Need for the System:

1. **Planned approach towards working: -**The working in the organization will be well planned and organized. The data will bestored properly in data stores, which help in retrieval of information as well as its storage.

2. **Accuracy: -** The level of accuracy in the proposed system will behigher. All operation would be done correctly and it ensures that whatever information is coming from the centre is accurate.

3. **Reliability:-**The reliability of the proposed system will be high due to the above stated reasons. The reason for the increased reliability of the system is that now there would be proper storage of information.

4. **No Redundancy:-** In the proposed system almost care would be that no information is repeated anywhere, in storage or otherwise. Thiswould assure economic use of storage space and consistency in thedata stored.

1.3 Scope of Work :

1) To design more sophisticated prediction models and feature extraction techniques and extend our proposed system to predict other clinical risks.

2) Some other future possibilities are for the discovery of the development more effective treatment protocols and for the development of personalized medicine.

1.4 Operating Environment - Hardware and Software :

1.2 Operational Environments

Hardware Specifications (Client Side)

RAM	Minimum 1 GB and above
Hard Disk	Minimum 20 GB and above
Processor	Pentium-IV and above

Hardware Specifications (Server Side)

RAM	Minimum 2GB and above
Hard Disk	Minimum 80 GB and above
Processor	Pentium-IV and above

Software Specifications (Client Side)

Operating System	Windows XP/Later
Web Browser	Internet Explorer-6/Later Mozilla Firefox

Software Specifications (Server Side)

Operating System	Windows 7
Web Browser	Internet Explorer-6/Later Mozilla Firefox
Technology	Core JAVA
Testing Platform	Manual + Automation (Selenium)
Database	My SQL
Development Tool(Editor)	Eclipse
Server	Tomcat
Supporting Technology	JSP, JavaScript, CSS

1.5 Detailed Description of Tool(s) Used in Testing :

Automation for Web Applications: Selenium

Selenium testing is applied for this application. It is tested Many, perhaps most, software applications today are written as web-based applications to be run in an Internet browser. The effectiveness of testing these applications varies widely among companies and organizations. In an era of highly interactive and responsive software processes where many organizations are using some form of agile methodology, test automation is frequently becoming a requirement for software projects. Test automation is often the answer. Test automation means using a software tool to run repeatable tests against the application to be tested. For regression testing this provides that responsiveness.

Test automation has specific advantages for improving the long-term efficiency of a software team's testing processes. Test automation supports:

- Frequent regression testing
- Rapid feedback to developers
- Virtually unlimited iterations of test case execution
- Support for Agile and extreme development methodologies
- Disciplined documentation of test cases
- Customized defect reporting
- Finding defects missed by manual testing

To Automate or Not to Automate for Our Application?

Is automation always advantageous? When should one decide to automate test cases?

It is not always advantageous to automate test cases. There are times when manual testing may be more appropriate. For instance, if the application's user interface will change considerably in the near future, then any automation might need to be rewritten anyway. Also, sometimes there simply is not enough time to build test automation. For the short term, manual testing may be more effective. If an application has a very tight deadline, there is currently no test automation available, and it's imperative that the testing get done within that time frame, then manual testing is the best solution.

Introducing Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

Selenium's Tool Suite

Selenium is composed of multiple software tools. Each has a specific role.

Selenium 2 (aka. Selenium WebDriver)

Selenium 2 is the future direction of the project and the newest addition to the Selenium toolkit. This brand new automation tool provides all sorts of awesome features, including a more cohesive and object oriented API as well as an answer to the limitations of the old implementation.

Selenium 1 (aka. Selenium RC or Remote Control)

As you can read in Brief History of The Selenium Project, Selenium RC was the main Selenium project for a long time, before the Web Driver/Selenium merge brought up Selenium 2, the newest and more powerful tool.

Selenium IDE

Selenium IDE (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.

Selenium-Grid

Selenium-Grid allows the Selenium RC solution to scale for large test suites and for test suites that must be run in multiple environments. Selenium Grid allows you to run your tests in parallel, that is, different tests can be run at the same time on different remote machines.

Supported Browsers and Platforms

In Selenium 2.0, the supported browsers vary depending on whether you are using Selenium-WebDriver or Selenium-RC.

Selenium-WebDriver

Selenium-WebDriver supports the following browsers along with the operating systems these browsers are compatible with.

- Google Chrome
- Internet Explorer 7, 8, 9, 10, and 11 on appropriate combinations of Vista, Windows 7, Windows 8, and Windows 8.1. As of April 15 2014, IE 6 is no longer supported. The driver supports running 32-bit and 64-bit versions of the browser where applicable
- Firefox: latest ESR, previous ESR, current release, one previous release
- Safari
- Opera
- HtmlUnit
- phantomjs
- Android (with Selendroid or appium)
- iOS (with ios-driver or appium)

Selenium 1.0 and Selenium-RC.

This is the old, support platform for Selenium 1.0. It should still apply to the Selenium 2.0 release of Selenium-RC.

Browser	**Selenium IDE**	**Selenium 1 (RC)**	**Operating Systems**
Firefox 3.x	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
Firefox 3	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
Firefox 2	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
IE 8	Test execution only via Selenium RC*	Start browser, run tests	Windows
IE 7	Test execution only via Selenium RC*	Start browser, run tests	Windows
IE 6	Test execution only via Selenium RC*	Start browser, run tests	Windows
Safari 4	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac
Safari 3	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac
Safari 2	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac

CHAPTER 2

TEST PLANNING

2.1 Objectives of Testing:

Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- According to user location show the nearest medical to the user in any emergency situation.

- To examine current standards, methods, and uses for big data to develop a morbidities prediction system to assist providers in establishing higher standards of care, suggesting particular drugs for curing the disease and a more personalized medical care plan for the patient.

2.2 Software Requirement Specification

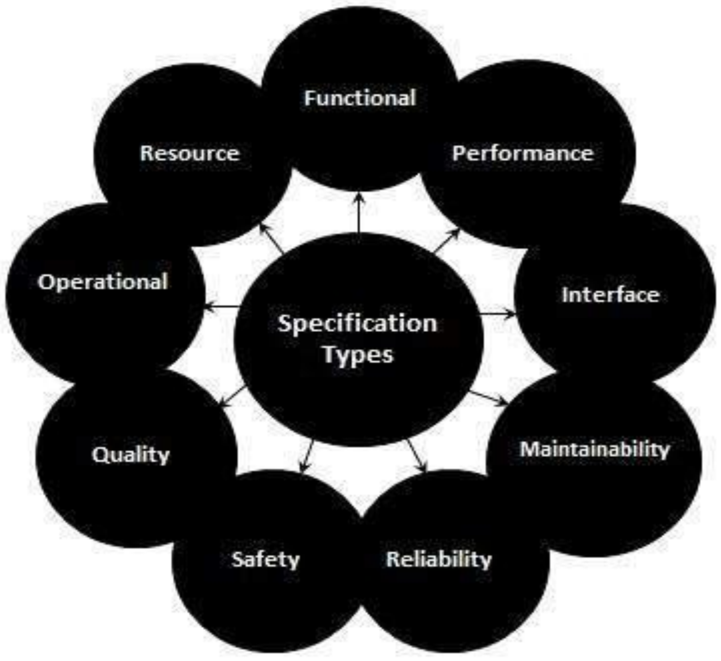
A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.



Deriving Test cases:

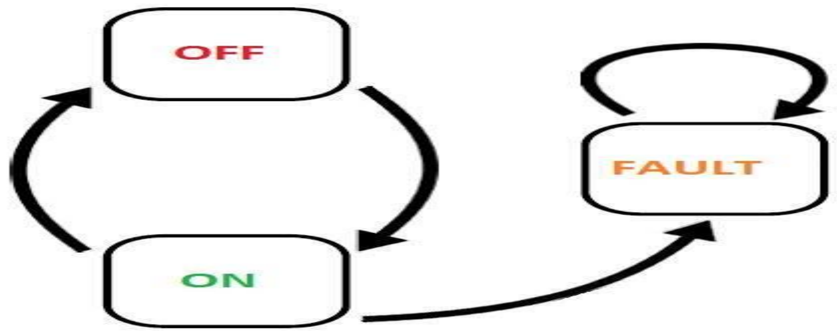
- Understand the various state and transition and mark each valid and invalid state
- Defining a sequence of an event that leads to an allowed test ending state
- Each one of those visited state and traversed transition should be noted down
- Steps 2 and 3 should be repeated until all states have been visited and all transitions travers

Advantages:

- Allows testers to familiarize with the software design and enables them to design tests effectively.
- It also enables testers to cover the unplanned or invalid states.

Example:

A System's transition is represented as shown in the below diagram:



2.3 Master Test Plan (IEEE format)

Introduction

This document is a high-level overview defining our testing strategy for the Sorted Binary Tree application. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. We will utilize testing criteria under the white box, black box, and system-testing paradigm. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process we will be applying the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation.

Team Interaction

The following describes the level of team interaction necessary to have a successful product.

- The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.

- Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing.

Test Objective:

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to mange our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

Testing Process:

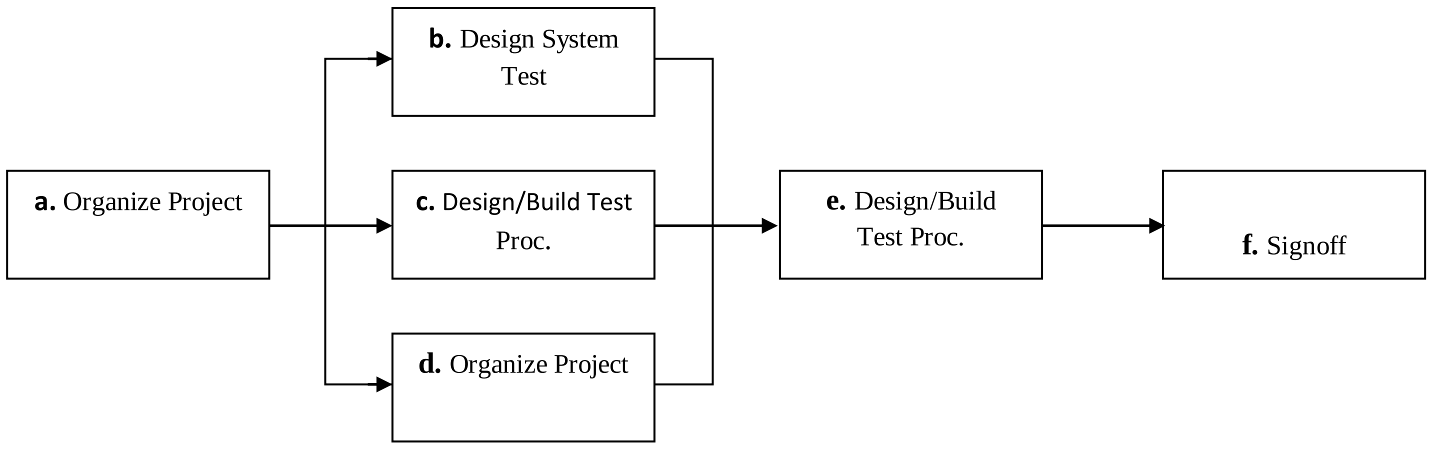

b. Design System Test
a. Organize Project
c. Design/Build Test Proc.
e. Design/Build Test Proc.
f. Signoff
d. Organize Project

Figure 1: Test Process Flow

The diagram above outlines the Test Process approach that will be followed.

a. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

b. **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

c. **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

d. **Build Test Environment** includes requesting/building hardware, software and data set-ups.

e. **Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

f. **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

2.4 Review of Test Basis (prototype testing)

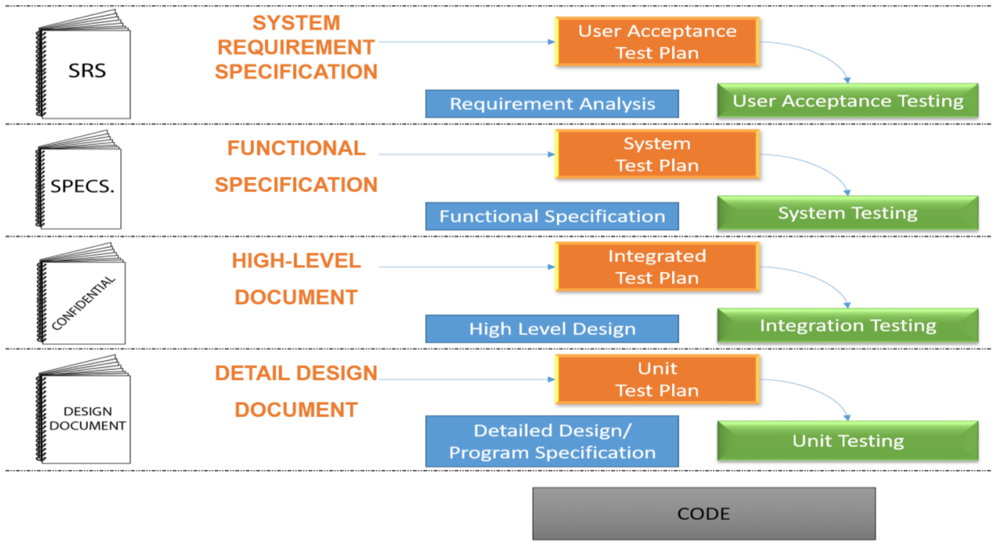
What is Test Basis in Software Testing?

Test Basis is defined as the source for creation of test Cases. It can be the Application itself or the requirement documents like SRS (Software Requirement Specification), BRS (Business Requirement Specification), etc. This tutorial explains "Test-Basis". with the help of a case study Consider a scenario, where the client sends a request to add a functionality to Flight Reservation to allow sending an order via email. He also specifies the GUI fields and buttons he wants. Even though, the application is yet to be developed, try and develop a few test cases for this requirement. A few test cases among the many you could have thought of are listed below

- Check the response when valid Email ID is entered, and send is pressed
- Check the response when invalid Email ID are entered and send is pressed
- Check the response when Email ID is empty and send is pressed

You may have also realized that to create test cases you need to look at something to base your test. This is nothing but Test Basis. This test basis could be the actual **Application Under Test(AUT)**, or maybe even by experience but most of the times, like, in this case, would be based on documents.

In fact, this is what happens during the different phases of V- Model.



.

Where test plans are created using the corresponding documents, and once the code is ready, it is ready for testing.

What is Test Basis?

Test basis is defined as the source of information or the document that is needed to write test cases and also for test analysis.

Test basis should be well defined and adequately structured so that one can easily identify test conditions from which test cases can be derived.

Typical Test Basis:

- Requirement document
- Test Plan
- Codes Repository
- Business Requirement

What is Prototype Testing?

Prototype Testing is conducted with the intent of finding defects before the website goes live. Online Prototype Testing allows seamlessly to collect quantitative, qualitative, and behavioral data while evaluating the user experience.

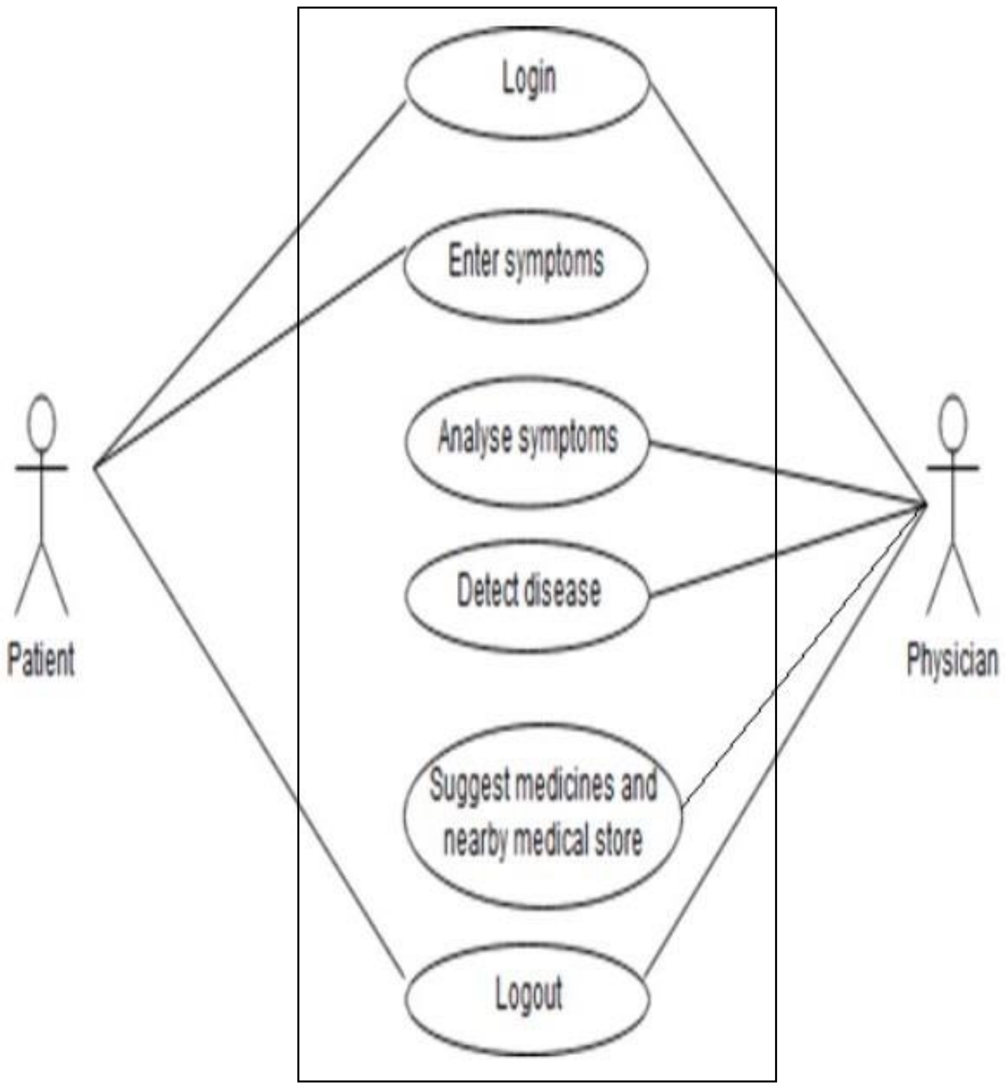
Characteristics of Prototype Testing:

- To evaluate new designs prior to the actual go live to ensure that the designs are clear, easy to use and meet users requirements.
- Is best when iterative testing is built into the development process, so that changes can be easily made often to ensure that major issues do not arise well before going live.
- Provides confirmation about the new design direction, branding and messaging is going in the right direction.

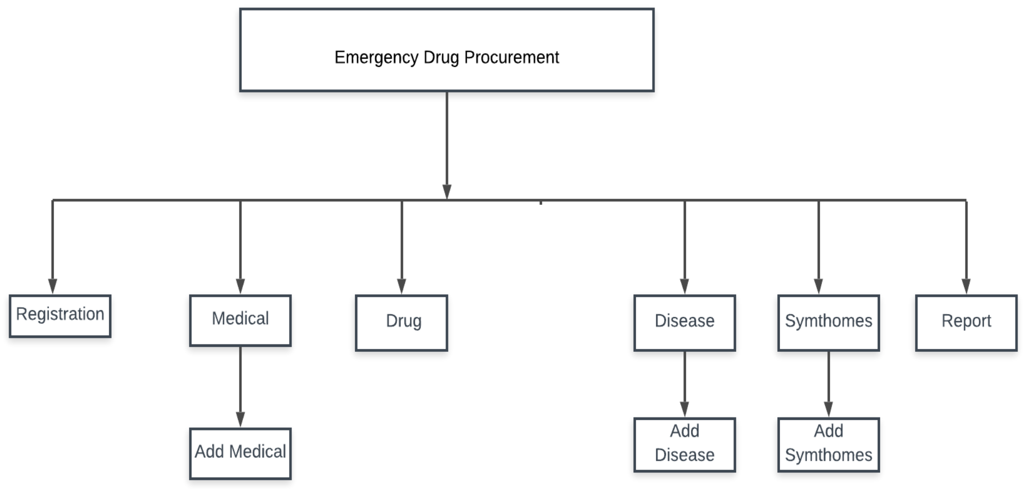
CHAPTER 3

TEST ANALYSIS & DESIGN

3.1 **Use Case Diagram:**



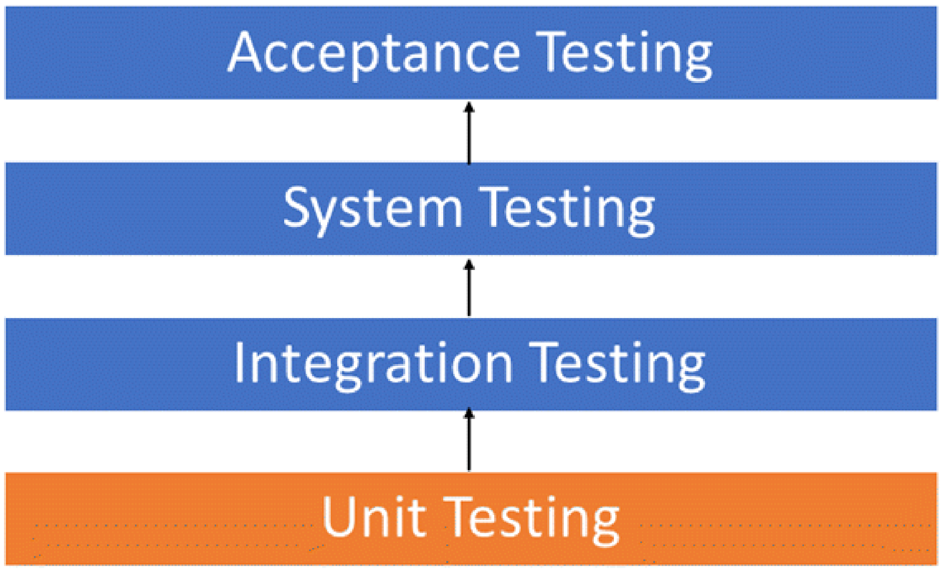
3.2 Module Hierarchy Diagram:



3.3 Unit Test Plan :

Unit Testing

It is the testing of individual software units of the application .it is done after the complexion of an individual unit before integration. Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.



Unit Testing Benefits

- Unit testing increases confidence in changing maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.
- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

Unit Testing Tips

- Find a tool/framework for your language.
- Do not create test cases for everything. Instead, focus on the tests that impact the behaviour of the system.
- Isolate the development environment from the test environment.
- Use test data that is close to that of production.
- Aim at covering all paths through the unit. Pay particular attention to loop conditions.
- Make sure you are using a version control system to keep track of your test scripts.
- In addition to writing cases to verify the behaviour, write cases to ensure the performance of the code.
- Perform unit tests continuously and frequently.

Unit Test Plan:

Sr.	**Requirements**	**Typical Components**	**Detailed Description**
1)	**Introduction**	a) Test Strategy and Approach	
		b) Test Scope	
		c) Test Assumptions	
2)	**Walkthrough (Static Testing)**	a) Defects Discovered and Corrected	
		b) Improvement Ideas	
		c)Structured Programming Compliance	
		d) Language Standards	
		e)Development Documentation Standards	
3)	**Test Cases (Dynamic Testing)**	a) Input Test Data	
		b) Initial Conditions	
		c) Expected Results	
		d) Test Log Status	

4)	**Environment Requirements**	a) Test Strategy and Approach	
		b) Platform	
		c) Libraries	
		d) Tools	
		e) Test Procedures	
		f) Status Reporting	

3.4 Test Harness :

What is Test Harness?

Test harness enables the automation of tests. It refers to the system test drivers and other supporting tools that requires to execute tests. It provides stubs and drivers which are small programs that interact with the software under test.

There are two context where Test Harness is used

1. **Automation testing:** It contains the test scripts, parameters necessary to run these scripts and gather results to analyze it

2. **Integration testing:** It is used to put together two units of code or module that interact with each other to check whether or not the combined behavior is as expected or not.

3.5 Development of Test Scenario as per requirements

What is a Test Scenario?

A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility. As a tester, you may put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

What is Scenario Testing?

Scenario Testing is a variant of Software Testing where Scenarios are Used for Testing. Scenarios help in an Easier Way of Testing of the more complicated Systems

How to create a Test Scenario

As a tester, you can follow these five steps to create Test Scenarios-

- **Step 1**: Read the Requirement Documents like BRS, SRS, FRS, of the System Under Test (SUT). You could also refer uses cases, books, manual, etc. of the application to be tested.

- **Step 2**: For each requirement, figure out possible users actions and objectives. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker's mindset.

- **Step 3:** After reading the Requirements Document and doing your due Analysis, list out different test scenarios that verify each feature of the software.

- **Step 4:** Once you have listed all possible Test Scenarios, a Traceability Matrix is created to verify that each & every requirement has a corresponding Test Scenario

- **Step 5:** The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other Stakeholders in the project.

3.6 Requirement Traceability Matrix (Horizontal traceability)

What is Traceability Matrix?(TM)

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

What is RTM (Requirement Traceability Matrix)?

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or software development team and their traceability in a single document delivered at the conclusion of the life-cycle.

Requirement Traceability Matrix – Parameters include

- Requirement ID
- Risks
- Requirement Type and Description
- Trace to design specification
- Unit test cases
- Integration test cases
- System test cases
- User acceptance test cases
- Trace to test script

How to create Requirement Traceability Matrix

On the basis of **Business Requirement Document (BRD)** and **Technical Requirement Document (TRD)**, testers start writing test cases.

Note: QA teams do not document the BRD and TRD. Also some companies use **Function Requirement Documents (FRD)** which are similar to Technical Requirement Document but the process of creating Traceability Matrix remains the same.

Let's Go Ahead and create RTM Testing

Step 1: Our Test Case is Verify Login, when correct ID and Password is entered, it should login successfully"

Step 2: Identify the Technical Requirement that this test case is verifying. For our test case, the technical requirement is T94 is being verified.

Step 3: Note this Technical Requirement in the Test Case.

Step 4: Identify the Business Requirement for which this TR (Technical Requirement-T94) is defined

Step 5: Note the BR (Business Requirement) in Test Case

Step 6: Do above for all Test Cases. Later Extract the First 3 Columns from your Test Suite. RTM in testing is Ready!

3.7 Test Case Design

3.8 Test Data Generation

What is Test Data Generation? Why test data should be created before test execution?

Depending on your testing environment you may need to CREATE Test Data (Most of the times) or at least identify a suitable test data for your test cases (is the test data is already created).

Typically test data is created in-sync with the test case it is intended to be used for.

Test Data can be generated -

- Manually
- Mass copy of data from production to testing environment
- Mass copy of test data from legacy client systems
- Automated Test Data Generation Tools

1] Test case For Admin Login Page:

Project Name: Emergency Drug Procurement Planning Based on Big-Data Drive Morbidity Prediction.

Prepared Date:-1-02-2019. Prepared By: PritamPatil

Module Name: Login. Reviewed Date:-7-02-2019

Project Code: - Drug Reviewed By:-Somesh Patil

Total no of test Cases:-04

Total no of test Cases Passed:-04

Total no of test Cases failed:-00

Total no of test Cases executed:-04

Total no of test Cases pending:-00

Test Case ID	Test Case Procedure	Input Data	Expected Output	Actual Output	Test Status
Drug-LG-01	Checking the functionality of Admin LOGIN Button	1.Enter valid Usernames in textbox 2. Enter valid Password in textbox 3. Click on Admin LOGIN Button	Admin Welcome page should be displayed	Admin Welcome page displayed	Pass

Drug-LG-02	Checking the functionality of Admin LOGIN Button	1.Enter invalid User Name in text box 2. Enter valid Password in password textbox 3. Click on Admin LOGIN Button	Admin Welcome page should not be displayed	Admin Welcome page is not displayed	Pass
Drug-LG-03	Checking the functionality of Admin LOGIN Button	1.Enter valid User name in User name textbox 2. Enter invalid Password in password textbox 3. Click on Admin LOGIN Button	Admin Welcome page should not be displayed	Admin Welcome page is not displayed	Pass

Drug-LG-04	Checking the functionality of Admin LOGIN Button	1.Enter invalid User name in User name textbox 2. Enter invalid Password in password textbox 3. Click on Admin login Button	Admin Welcome page should not be displayed	Admin Welcome page is not displayed	Pass

2] Test case For Add Distributor Page:

Project Name:Emergency Drug Procurement Planning Based on Big-Data Drive Morbidity Prediction.

Prepared Date:-8-02-2019. Prepared By:-: PritamPatil

Module Name: Patient Registration. Reviewed Date:-15-02-2019

Project Code: - Drug Reviewed By:-: Somesh Patil

Total no of test Cases:-04

Total no of test Cases Passed:-04

Total no of test Cases failed:-00

Total no of test Cases executed:-04

Total no of test Cases pending:-00

Test Case ID	**Test Case Procedure**	**Input Data**	**Expected Output**	**Actual Output**	**Test Status**
Drug-AT-01	Checking the functionality of Registration Button	1.Enter valid First name, last name 2. Enter valid Contact Number and Email. 3.Select valid Blood Group 3. Click on Registration	Patient Registration Successfully should be displayed	Patient Registration Successfully displayed	Pass

		Button			
Drug - AT-02	Checking the functionality of Registration Button	1.Enter invalid First name, last name 2. Enter valid Mobile no and email id. 3.Select valid Blood Group 3. Click on Registration Button	Patient Registration Successfully should not be displayed	Patient not Registration Successfully displayed	Pass
Drug - AT-03	Checking the functionality of Registration Button	1.Enter valid First name, last name 2. Enter invalid Contact Number and Email. 3.Select valid Blood Group 3. Click on Registration	Patient Registration Successfully should not be displayed	Patient not Registration Successfully displayed	Pass

		Button			
Drug-AT-04	Checking the functionality of Registration Button	1.Enter valid First name, last name 2. Enter valid Contact Number and Email. 3.Select invalid Blood Group	Patient Registration Successfully should not be displayed	Patient not Registration Successfully displayed	Pass

3.9 Test Execution:

What is Test Execution?

Test Execution is an extension of Software Testing Life Cycle and is said to be the most important and “happening” part of Software Testing Life Cycle (STLC) and the entire software development. It is in short describes as the process of executing the code and comparing the expected and actual results. Test Executions starts when the entry criteria has been satisfied, i.e. the test manager ensures that the process of test executions starts only when the entry criteria is satisfied in order to avoid any unnecessary defects and delays in testing. Furthermore, it synchronizes the inputs with the application under test and measures the timing of tests. However, before elaborating more on Test Execution, it is important to know about Software Testing Life Cycle (STLC) to get a better understanding of Test Execution cycles.

Guidelines for Test Execution:

- The Build being deployed to the quality assurance environment is the most important part of the test execution cycle.
- Test execution is done in Quality Assurance (QA) environment.
- Test team size is not constant since the beginning of the project.
- Test execution happens in at least two cycles.
- Test execution phase consists Executing the test scripts + test script maintenance (correct gaps in the scripts) + reporting (defects, status, metrics, etc.)
- Exploratory tests are carried out once the build is ready for testing.

Test Execution Tools:

Also known as ‘test running tool’, Test Execution Tool is basically a tool used to run tests. These tools require a scripting/programming language in order to be executed, which creates a great advantage as they can repeat actions in loops for various data values, can take different routes depending on the outcome of a test and also can be called from other scripts giving structure to sets of tests.

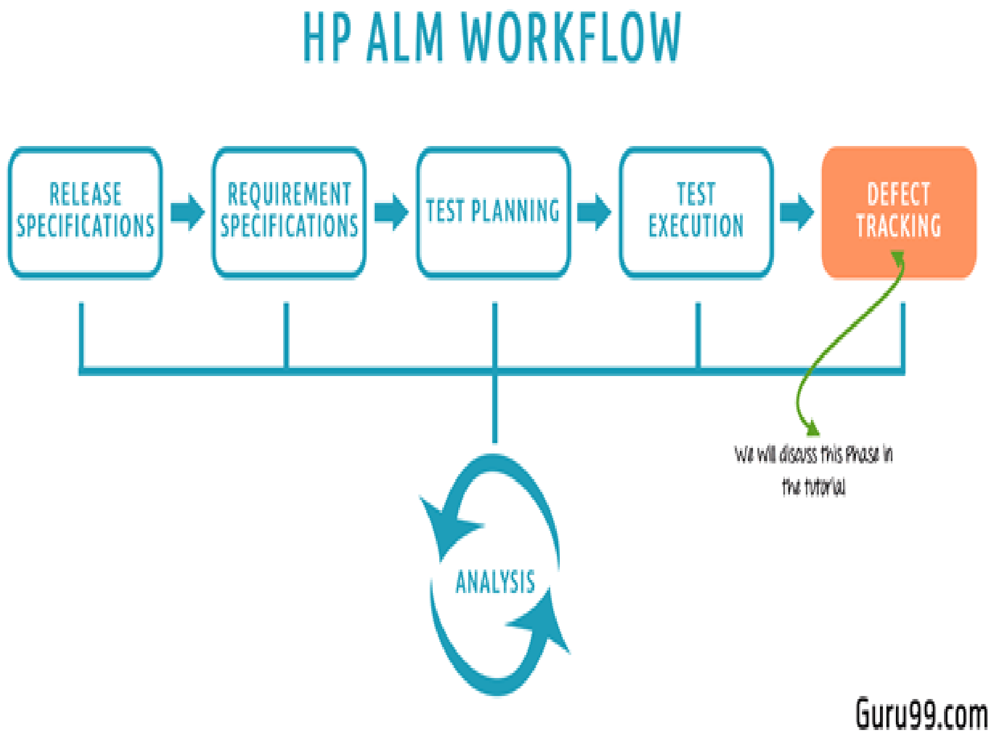
Characteristics of Test Execution Tools:

- Captures tests inputs while tests are executed manually.
- Stores expected results in the form of a screen or an object for comparison when the tests are executed again.
- It executes tests from stored scripts and optionally data files accessed by the script.
- Provides a dynamic comparison of screens, elements, links, controls, objects and values while running the tests.
- It performs post execution comparison.
- Keeps a log of results of the completed tests cases.
- Synchronizes inputs with the application under test.
- Measures the timing of tests.

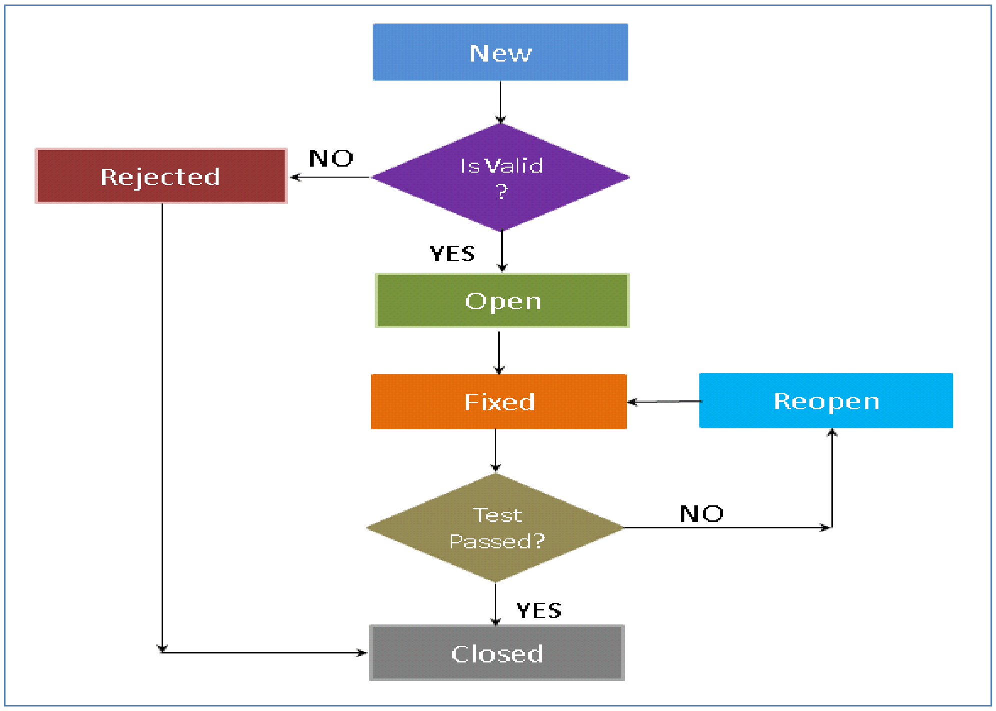
3.10 Defect Screens (Snapshot):

Defect Management Life Cycle in HP ALM (Quality Centre)

- A Defect is logged during the test execution, when expected result and actual result don't match with each other.
- Defect module in HP ALM not only helps users to post the defects but also enables them to track and gives the overall quality of the release at any stage of the development process.



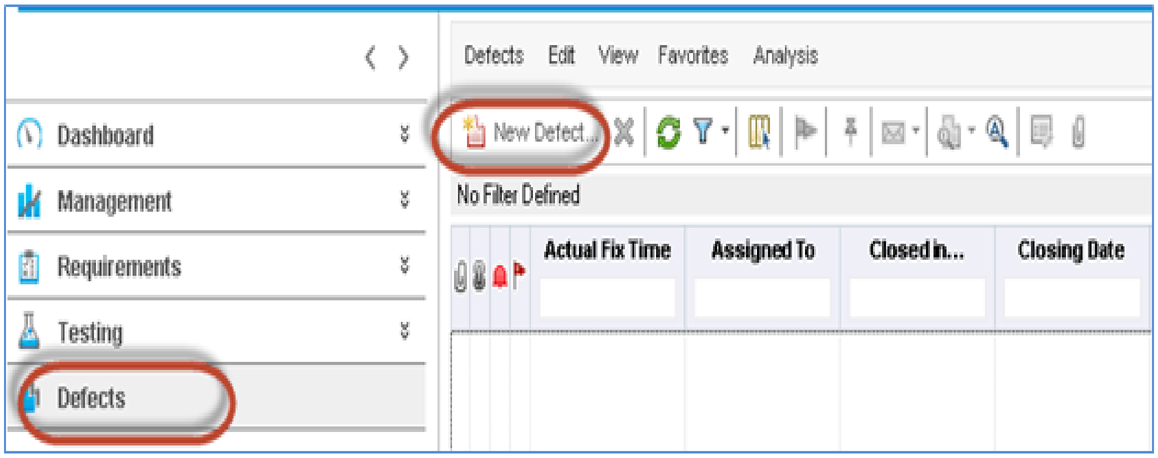
Default Defect Life Cycle in ALM:



Status	Explanation
New	When a defect is posted, the default status is 'New'
Open	When the defect is accepted by developers it is moved to 'Open' Status
Rejected	When the defect is rejected by developers it is moved to 'Rejected' Status
Fixed	When the defect is fixed by developers it is moved to 'Fixed' Status.
Reopen	If the testing has failed, the defect is moved to 'Reopen' status
Closed	If the testing has passed, the defect is moved to 'Closed' Status.

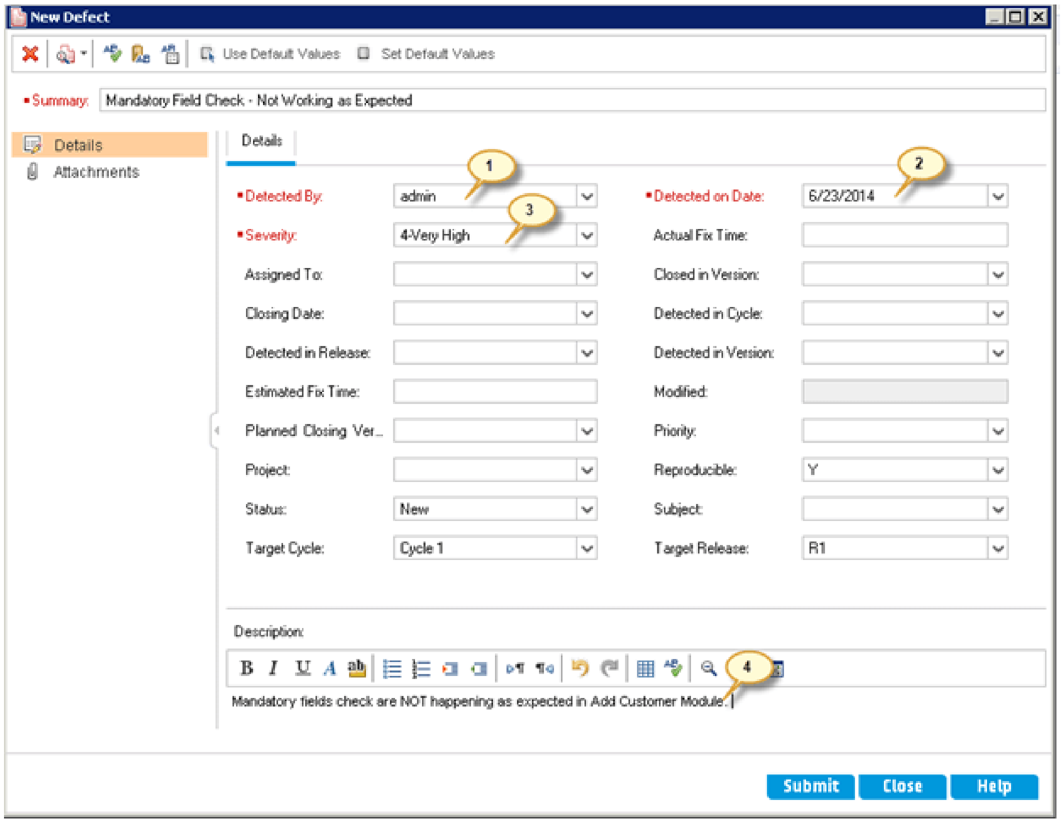
How to Create a New Defect

Step 1) Navigate to defects Tab in Quality Center and Click on "New Defect" button.



Step 2) The "New Defect" Dialog Open up. Fill in the following mandatory information.

- Enter Detected by Field
- Enter the Detected-on Date – By Default current date would be picked up
- Set the severity level of the defect.
- User can also enter other information and enter a brief description about the defect



Summary: Mandatory Field Check - Not Working as Expected

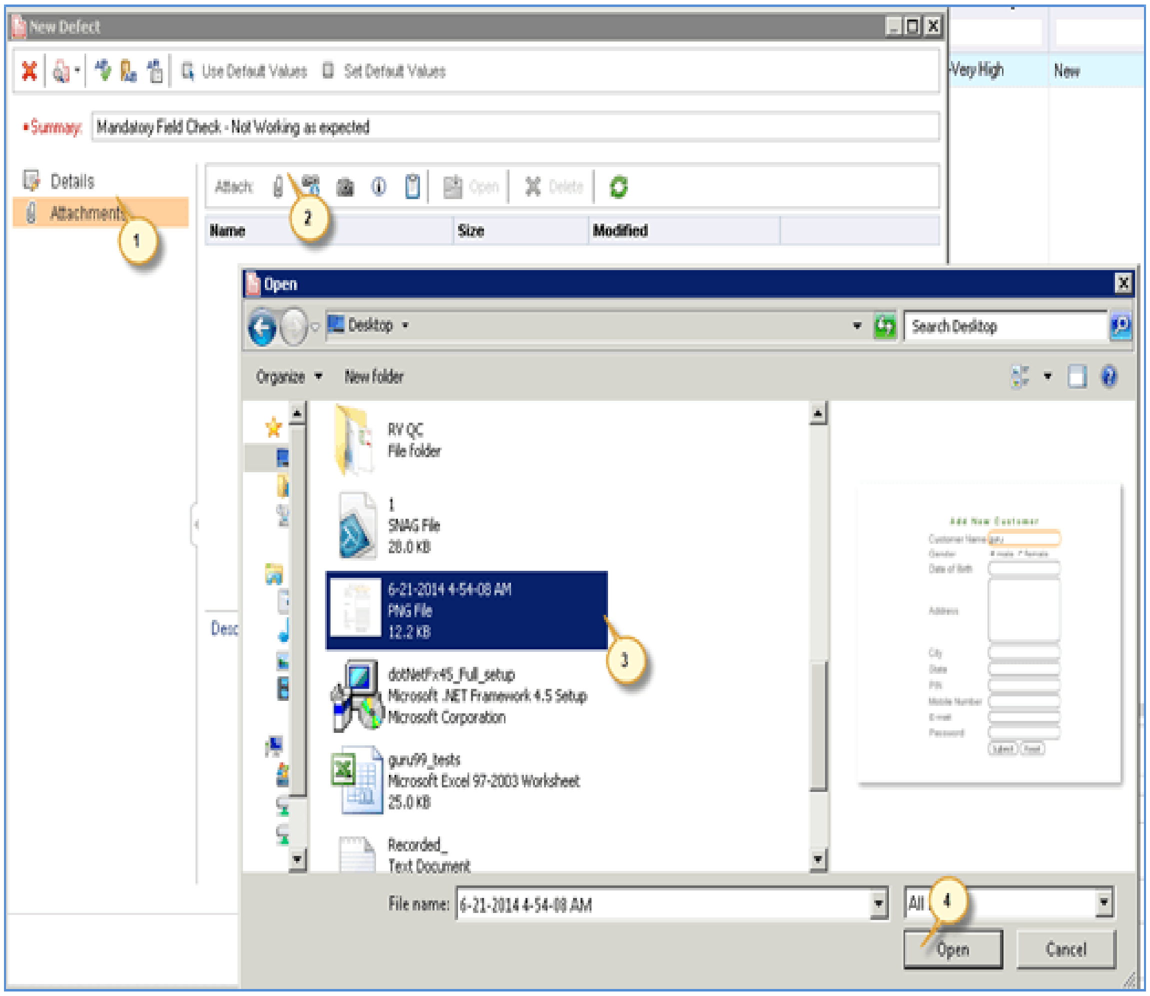
Detected By:	admin	Detected on Date:	6/23/2014
Severity:	4-Very High	Actual Fix Time:	
Assigned To:		Closed in Version:	
Closing Date:		Detected in Cycle:	
Detected in Release:		Detected in Version:	
Estimated Fix Time:		Modified:	
Planned Closing Ver...		Priority:	
Project:		Reproducible:	Y
Status:	New	Subject:	
Target Cycle:	Cycle 1	Target Release:	R1

Description:

Mandatory fields check are NOT happening as expected in Add Customer Module.

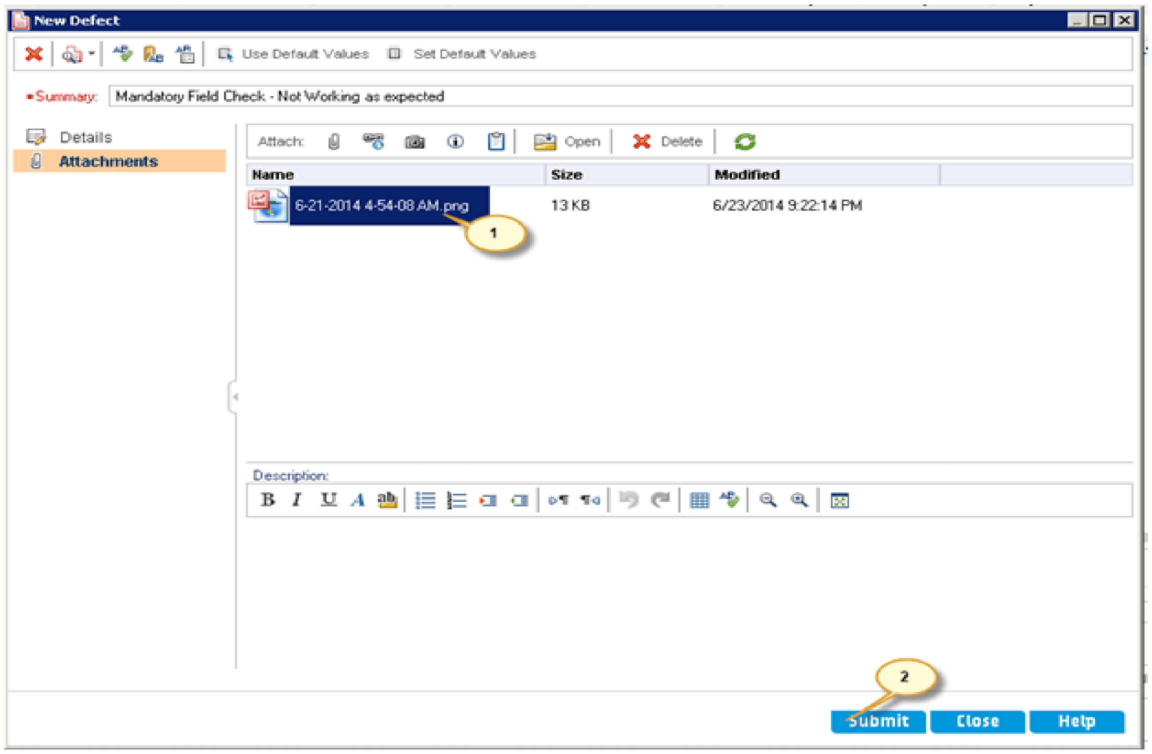
Step 3) Tester can also attach screenshots/other relevant files associated with the defect using 'attachments' tab.

1. Click 'Attachments' Tab
2. Click 'Attachments' Button
3. Select a File from the File explorer dialog.
4. Click 'Open'



Step 4) Upon clicking 'Open' we will be able to see that the file is attached under the attachment section.

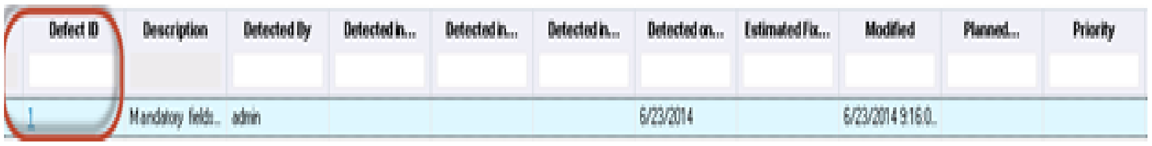
1. The Selected file has been uploaded
2. Click 'Submit' to post a defect after which it generates a defect ID.



Summary: Mandatory Field Check - Not Working as expected

Name	Size	Modified
6-21-2014 4-54-08 AM.png	13 KB	6/23/2014 9:22:14 PM

Step 5) The defect is posted, the same be accessed in Defects Tab as shown below. You can also notice that the defect ID is generated upon posting the defect.



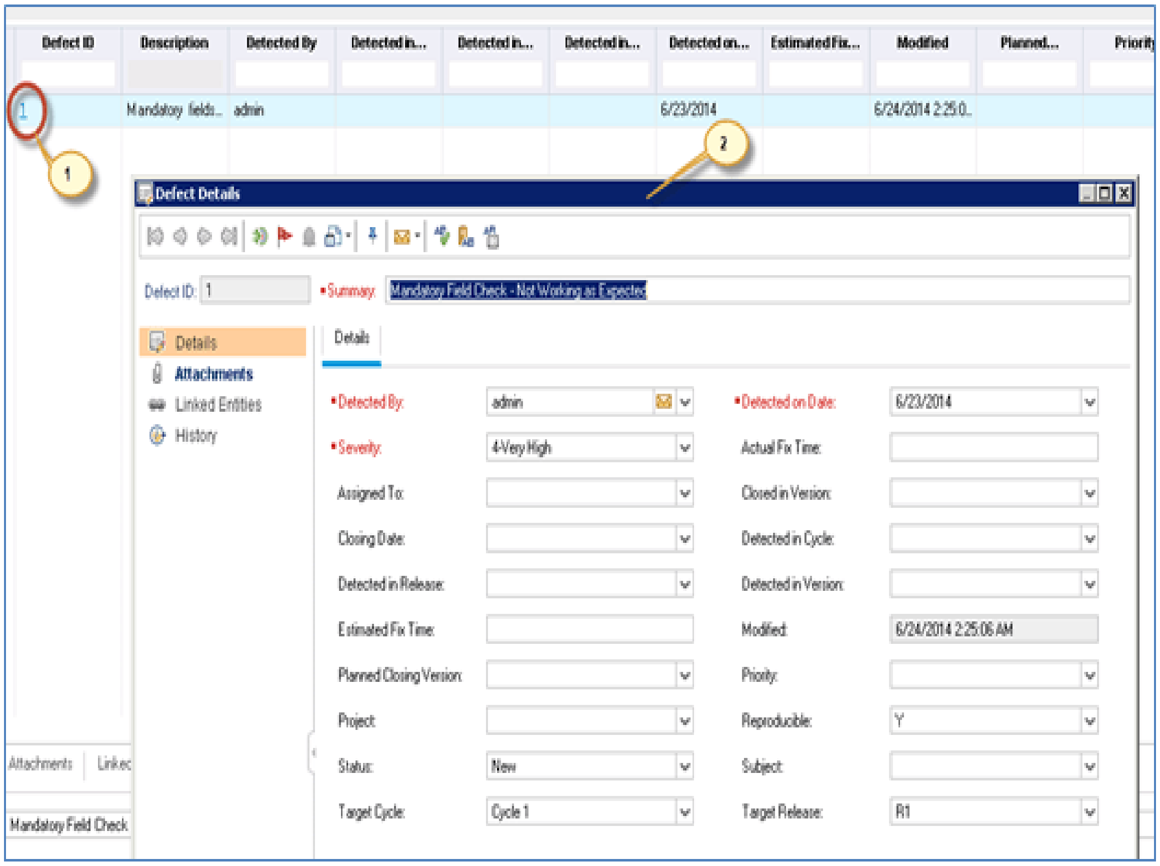
Defect ID	Description	Detected By	Detected in...	Detected in...	Detected in...	Detected on...	Estimated Fix...	Modified	Planned...	Priority
1	Mandatory fields...	admin				6/23/2014		6/23/2014 9:16:0...		

How to Link Defect to a Requirement

Users can link a defect with other defects or link a defect with requirements. By Linking defects and requirement we can generate coverage analysis graph and traceability matrix.

Step 1) After creating the defect, testers can map the linked requirements against it. To do the same,

1. Click on 'Defect ID'
2. The defect details dialog opens as shown below.



Defect ID	Description	Detected By	Detected in...	Detected in...	Detected in...	Detected on...	Estimated Fix...	Modified	Planned...	Priority
1	Mandatory fields...	admin				6/23/2014		6/24/2014 2:25:0...		

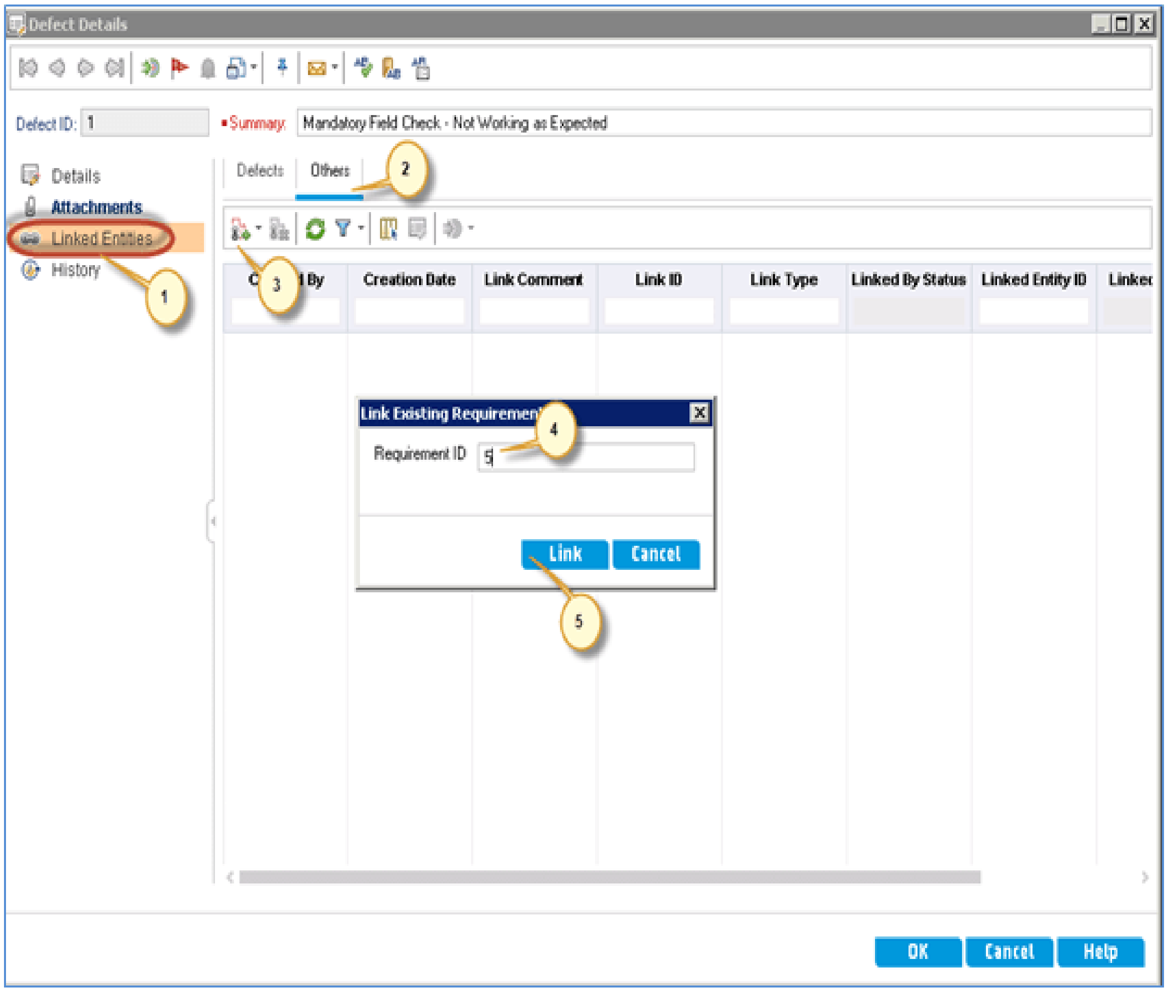
Mandatory Field Check

Defect ID: 1 *Summary: Mandatory Field Check - Not Working as Expected

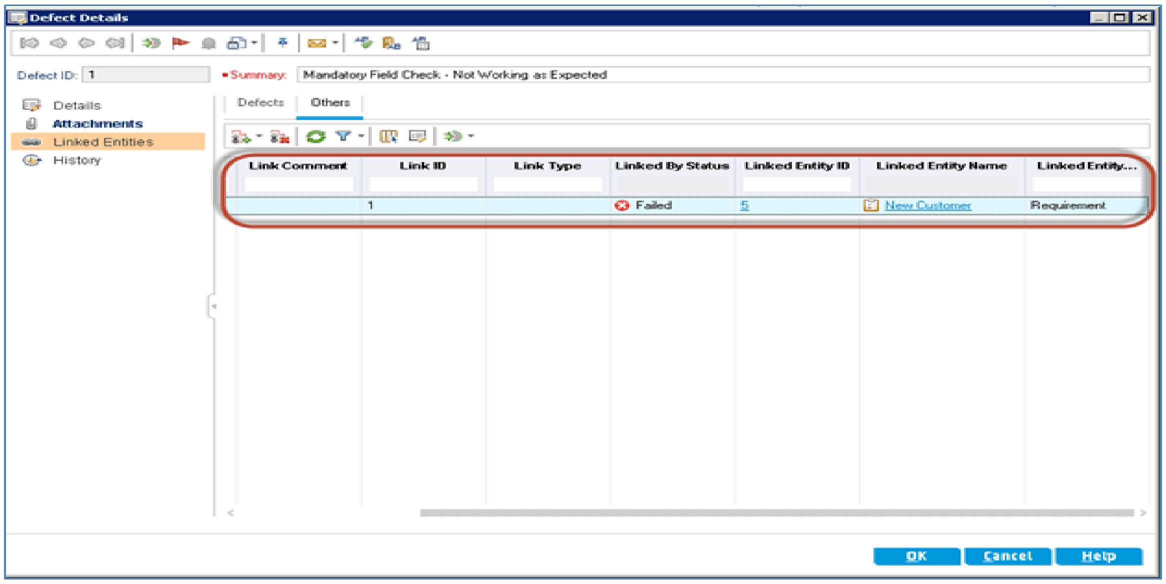
*Detected By:	admin	*Detected on Date:	6/23/2014
*Severity:	4-Very High	Actual Fix Time:	
Assigned To:		Closed in Version:	
Closing Date:		Detected in Cycle:	
Detected in Release:		Detected in Version:	
Estimated Fix Time:		Modified:	6/24/2014 2:25:06 AM
Planned Closing Version:		Priority:	
Project:		Reproducible:	Y
Status:	New	Subject:	
Target Cycle:	Cycle 1	Target Release:	R1

Step 2) To Link entities,

1. Navigate to 'Linked Entities'
2. Click 'Others' for linking requirements against this defect.
3. Click 'Link' button and choose 'by Id'(we can also select based on a requirement name)
4. Enter the Requirement ID against which this defect has to be mapped.
5. Click 'Link' Button



Step 3) After clicking link button the defect details window displayed back to the user with the added link as shown below.

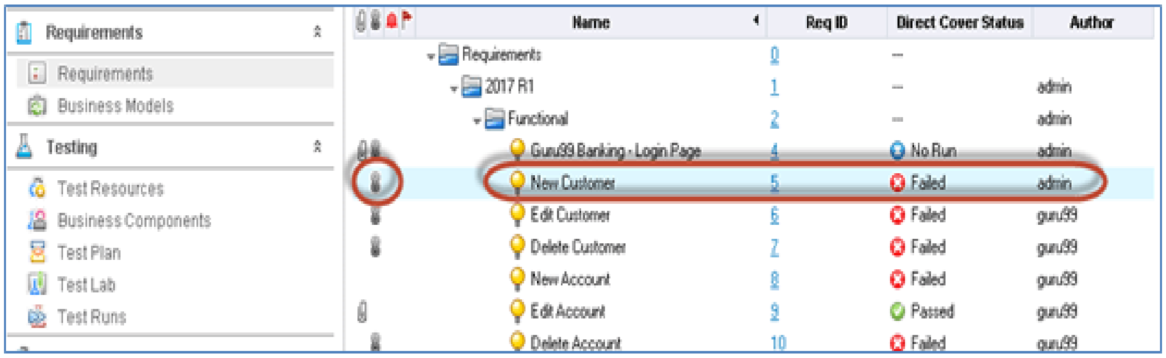


Defect ID: 1

*Summary: Mandatory Field Check - Not Working as Expected

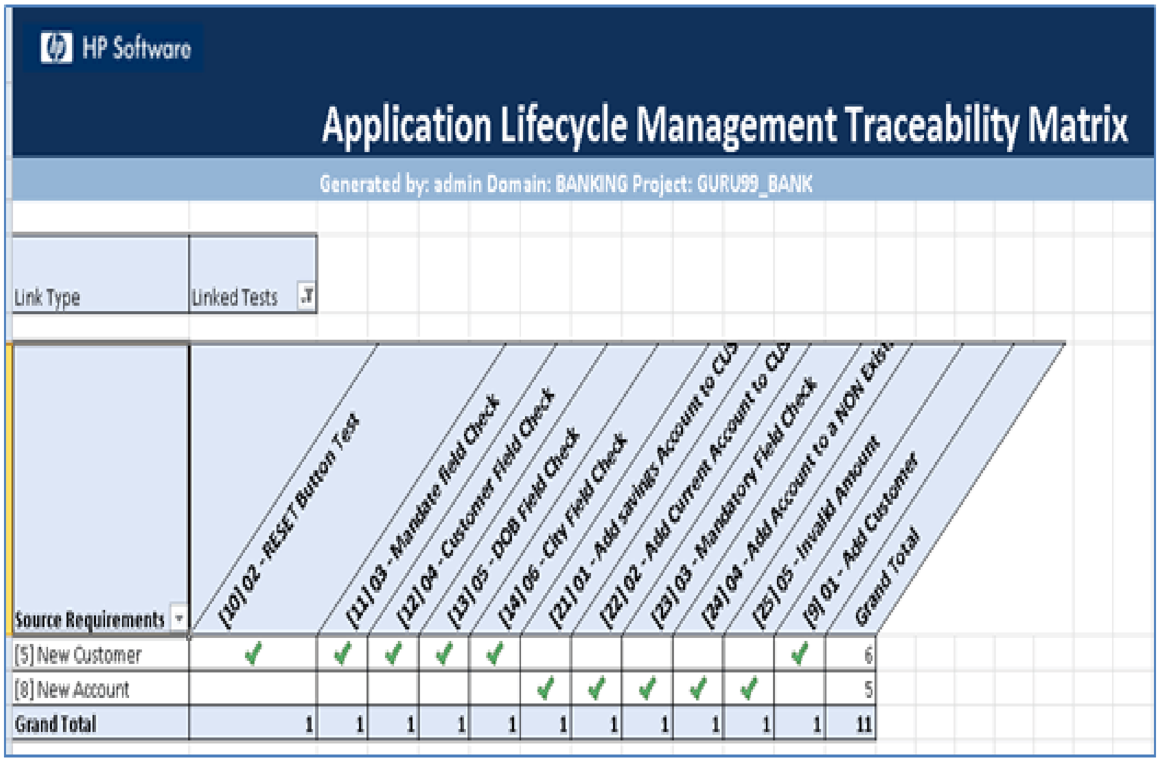
Link Comment	Link ID	Link Type	Linked By Status	Linked Entity ID	Linked Entity Name	Linked Entity...
	1		Failed	5	New Customer	Requirement

Step 4) Once the requirement is linked against a defect, the requirement displays with the link symbol against it as shown below.



Name	Req ID	Direct Cover Status	Author
Requirements	0	--	
2017 R1	1	--	admin
Functional	2	--	admin
Guru99 Banking - Login Page	4	No Run	admin
New Customer	5	Failed	admin
Edit Customer	6	Failed	guru99
Delete Customer	7	Failed	guru99
New Account	8	Failed	guru99
Edit Account	9	Passed	guru99
Delete Account	10	Failed	guru99

Step 5) Once the requirement is linked against a defect, the requirement Traceability Matrix can be generated. To generate the Traceability Matrix navigate to view menu of 'Requirements' and select 'Traceability matrix'. The generated Traceability Matrix would be generated as shown below.



Application Lifecycle Management Traceability Matrix

Generated by: admin Domain: BANKING Project: GURU99_BANK

Link Type	Linked Tests

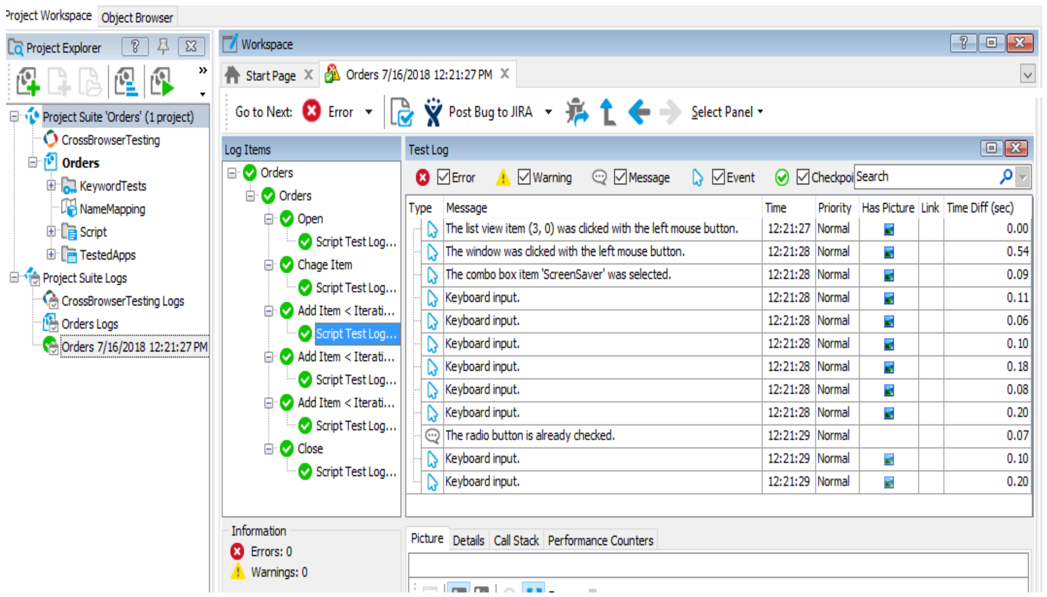
Source Requirements	[10] 02 - RESET Button Test	[11] 03 - Mandate field Check	[12] 04 - Customer Field Check	[13] 05 - DOB Field Check	[14] 06 - City Field Check	[21] 01 - Add savings Account to CUS	[22] 02 - Add Current Account to CUS	[23] 03 - Mandatory Field Check	[24] 04 - Add Account to a NON Exist	[25] 05 - Invalid Amount	[9] 01 - Add Customer	Grand Total
[5] New Customer	✓	✓	✓	✓	✓						✓	6
[8] New Account						✓	✓	✓	✓	✓		5
Grand Total	1	1	1	1	1	1	1	1	1	1	1	11

3.11 Test Logs

Test Complete generates a full-detailed log of all actions it performs during the test run. The test log provides the overall test run summary, indicates passed and failed tests and contains detailed information about each test operation, including the reasons of failed operations.

About Test Log

By default, the test log is automatically opened at the end of the test run session. Additionally, if the Show log on pause option is enabled, Test Complete displays the temporary test log when the test is paused. You can also open logs of previous test runs from the Project Explorer panel, which lists all available logs for the currently opened project suite and its projects.



Type	Message	Time	Priority	Has Picture	Link	Time Diff (sec)
	The list view item (3, 0) was clicked with the left mouse button.	12:21:27	Normal	▣		0.00
	The window was clicked with the left mouse button.	12:21:28	Normal	▣		0.54
	The combo box item 'ScreenSaver' was selected.	12:21:28	Normal	▣		0.09
	Keyboard input.	12:21:28	Normal	▣		0.11
	Keyboard input.	12:21:28	Normal	▣		0.06
	Keyboard input.	12:21:28	Normal	▣		0.10
	Keyboard input.	12:21:28	Normal	▣		0.18
	Keyboard input.	12:21:28	Normal	▣		0.08
	Keyboard input.	12:21:28	Normal	▣		0.20
	The radio button is already checked.	12:21:29	Normal			0.07
	Keyboard input.	12:21:29	Normal	▣		0.10
	Keyboard input.	12:21:29	Normal	▣		0.20

Posting Messages, Images and Files to the Log

In addition to information generated by Test Complete, you can post custom entries (messages, images and so on) to the test log. For more information, see Posting Messages, Images and Files to the Log.

Test Log’s Toolbar

For more information on actions available via the toolbar of the Test Log window, see the Test Log's Toolbar topic.

Test Log Panels

Test log information is displayed in several panels and tabbed pages. For more information on available test log panels, refer to the Test Log Pages and Panels section.

Working With Test Results

There are many operations that you can perform in the Test Log panel to analyze test results and collaborate with your colleagues. For example, you can filter test results to display only the information you are interested in or you can view only error messages. You can also print test results, export them to a file, send them by e-mail or submit them to an issue-tracking system.

3.12 Defect Report

Defect report is a document that identifies and describes a defect detected by a tester. The purpose of a defect report is to state the problem as clearly as possible so that developers can replicate the defect easily and fix it.

Defect Report:

ID	Drug
Project	Emergency Drug Procurement Planning Based on Big-Data Drive Morbidity Prediction
Product	Drug Procurement
Release Version	3.2
Module	Patient Registration
Detected Build Version	3.2
Summary	Not Detected any defect.
Description	There were no known defects in the Module, Further there were no new defects found in the testing phase, hence there was nothing to report
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.
Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Severity	Severity of the Defect. (See Defect Severity)
Defect Priority	Priority of the Defect. (See Defect Priority)
Reported By	The name of the person who reported the defect.
Assigned To	The name of the person that is assigned to analyze/fix the defect.
Status	The status of the defect. (See Defect Life Cycle)
Fixed Build Version	Build version of the product where the defect was fixed.

3.13 Defect Matrices:

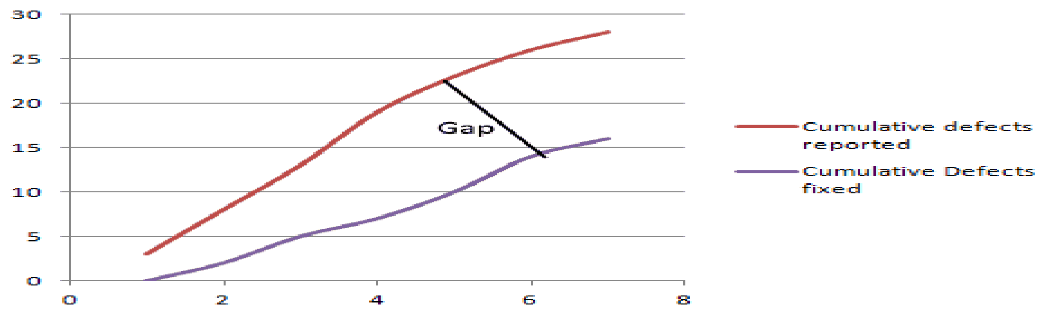
Defect removal efficiency is the extent to which the development team is able to handle and remove the valid defects reported by the test team.

To calculate Defect Gap, get a count of total defects submitted to the Development team and the total number of defects that were fixed by the end of the cycle. Calculate a quick percentage using the formula,

$$\text{Defect Gap \%} = \left(\frac{\textit{Total No.of defect Fixed}}{\textit{Total No.of valid defects reported}}\right) \times 100$$

Example: In a test cycle if the QA team reported 100 defects out of which 20 were invalid(not bugs, duplicates, etc.) and if the development team has resolved 65 of them, the defect gap % is: (65/100-20)X100= 81%(approximately)

When the data is collected over a period of time, Defect Gap Analysis can also be plotted as a graph as below:



A large gap shows that the development process needs changing.

Defect Density

Defect Density is defined as the number of defects per size of the software or application area of the software.

$$\text{Defect Density} = \frac{\textit{Total Number of defects}}{\textit{Total number of modules}}$$

If the total number of defects at the end of a test cycle is 30 and they all originated from 6 modules, the defect density is 5.

Defect Age

Defect Age is a measure that helps us track the average time it takes for the development team to start fixing the defect and resolve it. Defect Age is usually measured in the unit days, but for teams of rapid deployment models that release weekly or daily, projects, it this should be measured in hours.

For teams with efficient development and testing processes, a low defect age signals a faster turnaround for bug fixes.

Defect Age= Difference in Time created, and Time resolved

3.14 Test Summary Report

Test Summary Report Template

(IEEE 829-1998)

Test Summary Report Identifier

Some type of unique company generated number to identify this summary report, its level and the level of software that it is related to. Preferably the report level will be the same as the related software level. The number may also identify whether the summary report is for the entire project or a specific level of testing. This is to assist in coordinating software and testware versions within configuration management.

Summary

Identify all relevant support materials so that the reader of the report knows which version and release of the project/software is being reported on. It may be particularly important to identify the specific version of an external package used in the testing, especially if a new release occurred during the test cycle and was not included. The version/release information should match the information contained in the configuration management system and may include the following elements.

Test Items

This should match the item definitions from the appropriate level test plan that this report is covering. Any variance from the items specified in the test plan should be identified. Elements from the features sections of the test plan (both included and excluded) can also be included here or in a separate reference section.

Environment

The environment and any variances for that identified in the test plan should be verified here to ensure that the correct test setup was used. This will help avoid confusion when the product is released to production and will ensure that the test environment matches the destination platform.

References

Any documents that support this report and their location within the configuration management system.

Variances

Document any changes or deviations from those areas agreed on in thereference documents, especially in areas that may cause concern to the group accepting the test results.

Summary of Results

Report on the overall status of the incidents. Focus should be on trends and patterns in the process and not on specific individuals or teams. Avoid pure numbers, as numbers due not really provide insight as to the nature and cause of problems. The focus should be on costs, impacts, and trends; including any positivetrends. This is where you begin to set the stage for the evaluation of the test process, the quality of the testing and the software quality and can include areas such as:

- By Severity and Priority
- By cost/failure impact

Evaluation

Based on the evaluation of the testing as documented in sections three (3) through (6) assess the quality of the software. This should be an objectiveassessment of the failure likelihood andoverall quality in terms of the criteria specified in the appropriate level test plan. Each item identified in Section 2 under test items should be covered in the evaluation.

Include Good news as well. If the application tested out with high quality in some areas (even though others were no so good) be sure to state that as well.

- Limitations
- Incomplete or partial functions/feature
- Dropped features (due to requirements change or defects)
- Failure Likelihood
- High or Medium risk areas
- Good quality areas or features

Summary of Activities

Cover the planned activities and the changes to those plans especially in areas where the amount of actual effort greatly exceeded the planned effort. Include the reasons for the variances and the possible impact on the testing staff.

CHAPTER 4

USER (TESTER) MANUAL

4.1 Test Procedure Specification

A test procedure specification is a document that specifies a sequence of actions for the execution of a test. The test procedures test the implementation of the requirement. Test procedure specification development can begin after the test cases and design are completed and approved.

A test procedure specification includes the following elements:

- **Test Procedure Specification Identifier.** This is a unique identifier for a test procedure.
- **Purpose.** Describes what the procedure is for.
- **Special Requirements.** List of prerequisite procedures, special test skills, and environmental needs.
- **Procedure Steps.** Includes a list of the steps. IEEEStd 829-1998 describes the following key words, as applicable, that should be used in describing procedure steps:

    - **Log**. Special methods or formats for logging results and observations.
    - **Setup**. Preparation for execution of the procedure.
    - **Start**. How to begin execution of the procedure.
    - **Proceed**. Actions necessary during program execution.
    - **Measure**. How test measurements (e.g., response times) are made.
    - **Shutdown**. How to suspend testing in the face of an unscheduled event.
    - **Restart**. Where and how to restart any test step after a shut down of the test.
    - **Stop**. How to bring test execution to an orderly halt.
    - **Wrapup**. How to restore the test environment to its original state.
    - **Contingencies**. What to do in the case of an anomalous event.

4.2RTM(Vertical Traceability)

What is Traceability Matrix? (TM)

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

What is RTM (Requirement Traceability Matrix)?

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or software development team and their traceability in a single document delivered at the conclusion of the life-cycle.

Why RTM is Important?

The main agenda of every tester should be to understand the client's requirement and make sure that the output product should be defect-free. To achieve this goal, every QA should understand the requirement thoroughly and create positive and negative test cases.

Which Parameters to include in Requirement Traceability Matrix?

- Requirement ID
- Requirement Type and Description
- Test Cases with Status

4.3 Configuration Management (Test Item Transmitted Report)

TheConfiguration Management is a process of establishing and maintaining a product's performance, functional and physical attributes with its requirements, design, and functionalities through its life.

Configuration Management is a change control process. It helps in managing & controlling the versions of software and hardware configurations. It is used primarily when software requirements change.

Goal of Configuration Management

- Establish system and project product integrity.
- Maintain this integrity throughout the life cycle.

Activities in Configuration Management

- Configuration item identification
- Change Control
- Status Accounting
- Audits

Configuration Items

It is a decision on identification and control of configuration items. An element of Configuration Management, consisting of the evaluation, coordination, approval or disapproval and implementation of changes to Configuration items after formal establishment of their Configuration identification.

- Software requirements
- Design
- Code
- Test Cases
- Test Scripts
- Test Plan
- Project Plan

Change Control

Change Control is Board including numbers of people which take important decisions regarding the Configuration Management of Projects.

- Project Manager
- User representative
- Funding Manager
- Contractor's Project Manager
- Configuration Manager
- Quality Manager
- System Engineer

Change Control Process

- Change request submitted by the client.
- Impact Assessment performed by BA
- Recommendation prepared by BA
- Report submitted to CCB

CHAPTER 5

CONCLUSION

5. Conclusion:

- Big data analytics has the potential to transform the way healthcare providers use sophisticated technologies to gain in sight from their clinical and other data repositories and make informed decisions.

- In the future we'll see the rapid, wide spread implementation and use of big data analytics across the healthcare organization and the healthcare industry.

- As big data analytics becomes more mainstream, issues such as guaranteeing privacy, safeguarding security, establishing standards and governance, and continually improving the tools and technologies will garner attention.

- Big data analytics and applications in healthcare are at a nascent stage of development, but rapid advances in platforms and tools can accelerate their maturing process.

CHAPTER 6

BIBLIOGRAPHY

6. Bibliography

[1] J. D. Quick, “Applying management science in developing countries: ABC analysis to plan public drug procurement,” Socio-Eco. Plan. Sci., vol. 16, no. 1, pp. 39–50, 1982.

[2] M. Duggan and F. M. Scott Morton, “The distortionary effects of government procurement: Evidence from medicaid prescription drug purchasing,” Quar. J. Eco., vol. 121, no. 1, pp. 1–30, 2006.

[3] Y. Shu and R. Tong, “Common problems of hospital drugs procurement and its solutions,” China Pharm., vol. 22, no. 33, pp. 3114–3115, 2011.

[4] P. V. Singh, A. Tatambhotla, and R. R. Kalvakuntla, “Replicating Tamil Nadu’s drug procurement model,” Eco. Polit. Weekly, vol. 47, no. 39, pp. 26–29, 2012.

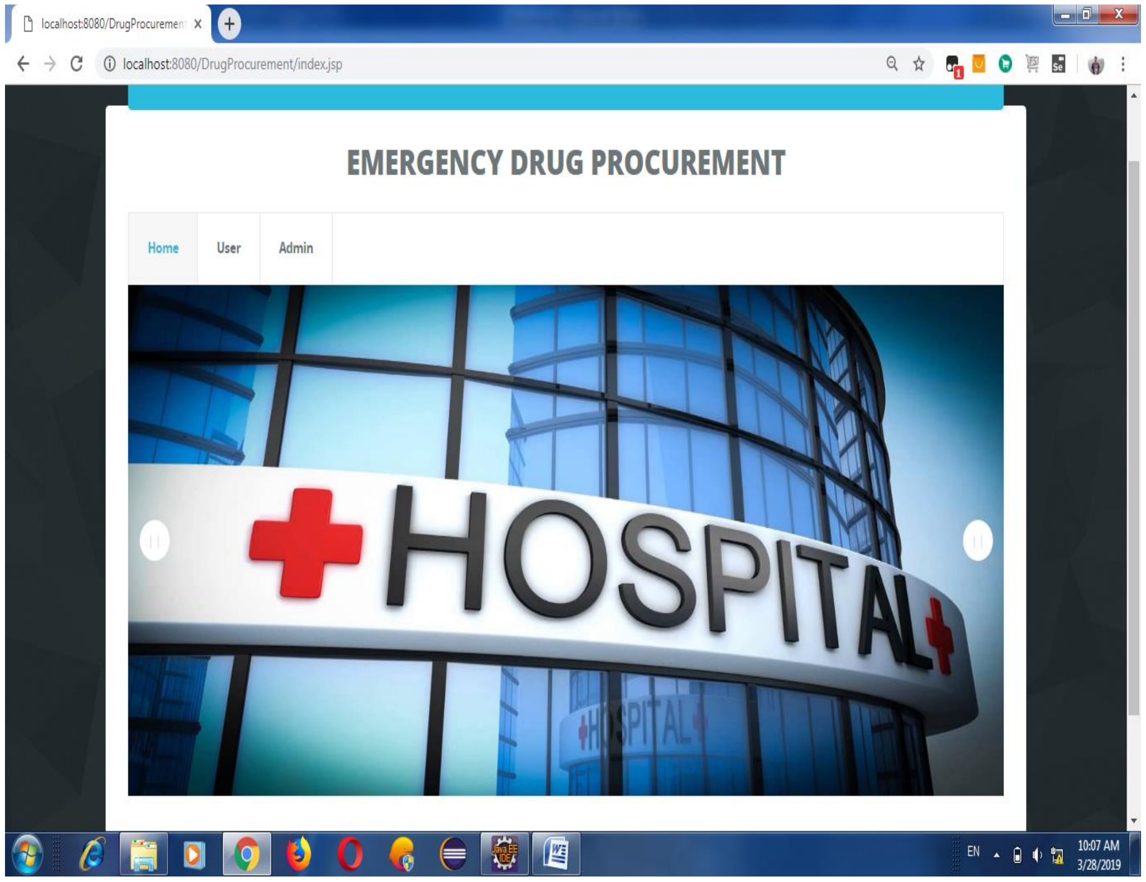
[5] A. L. Kjos, N. T. Binh, C. Robertson, and J. Rovers, “A drug procurement, storage and distribution model in public hospitals in a developing country,” Res. Social Admin. Pharm., vol. 12, no. 3, pp. 371–383, 2016.

ANNEXURE 1

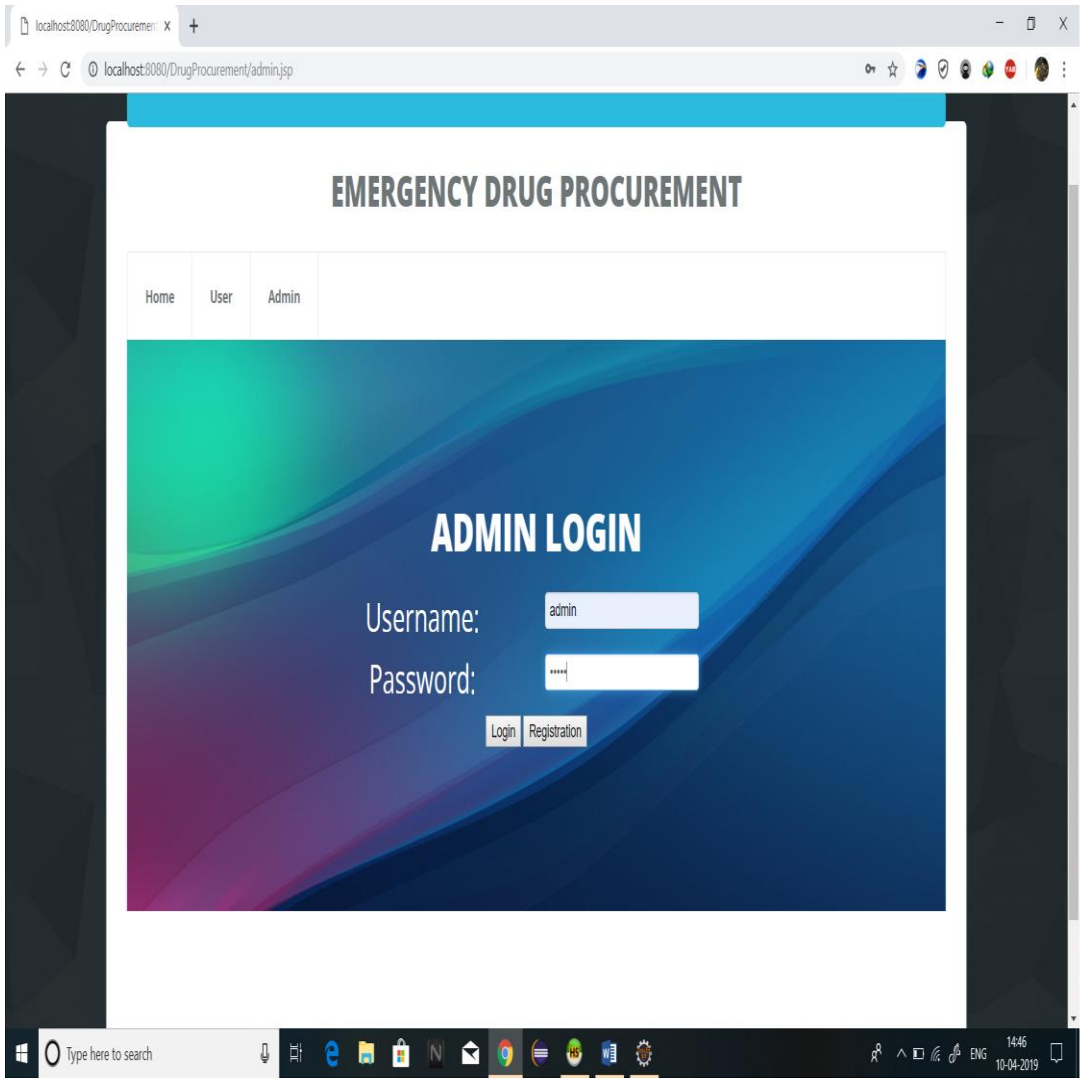
USER INTERFACE SCREEN

ANNEXURE 1: USER INTERFACE SCREENS used for testing

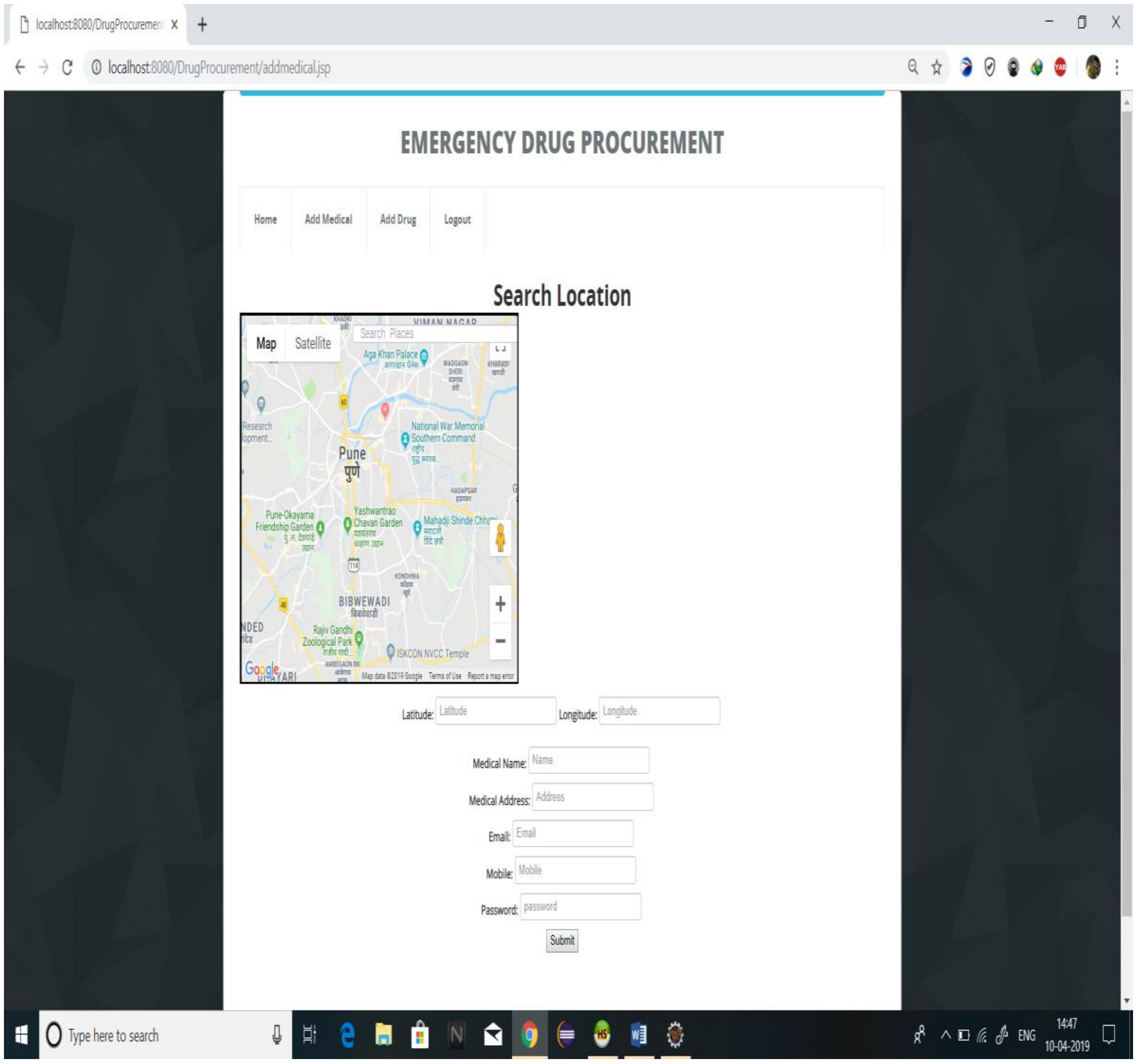
Home Page:


localhost:8080/DrugProcurement
localhost:8080/DrugProcurement/index.jsp
EMERGENCY DRUG PROCUREMENT
Home
User
Admin
HOSPITAL
EN
10:07 AM
3/28/2019

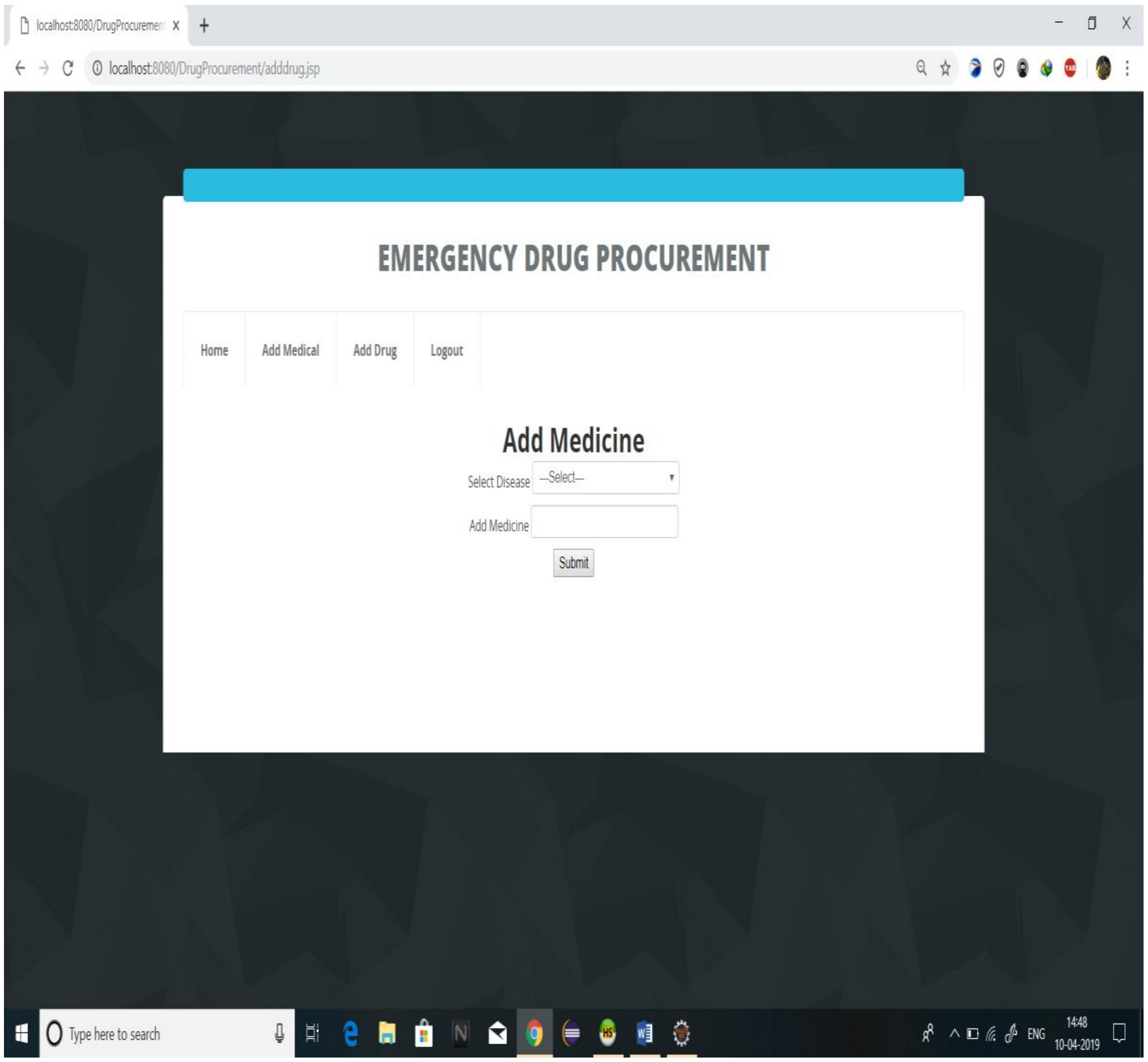
Admin Login :


localhost:8080/DrugProcurement/admin.jsp
EMERGENCY DRUG PROCUREMENT
Home
User
Admin
ADMIN LOGIN
Username:
admin
Password:
Login
Registration
Type here to search

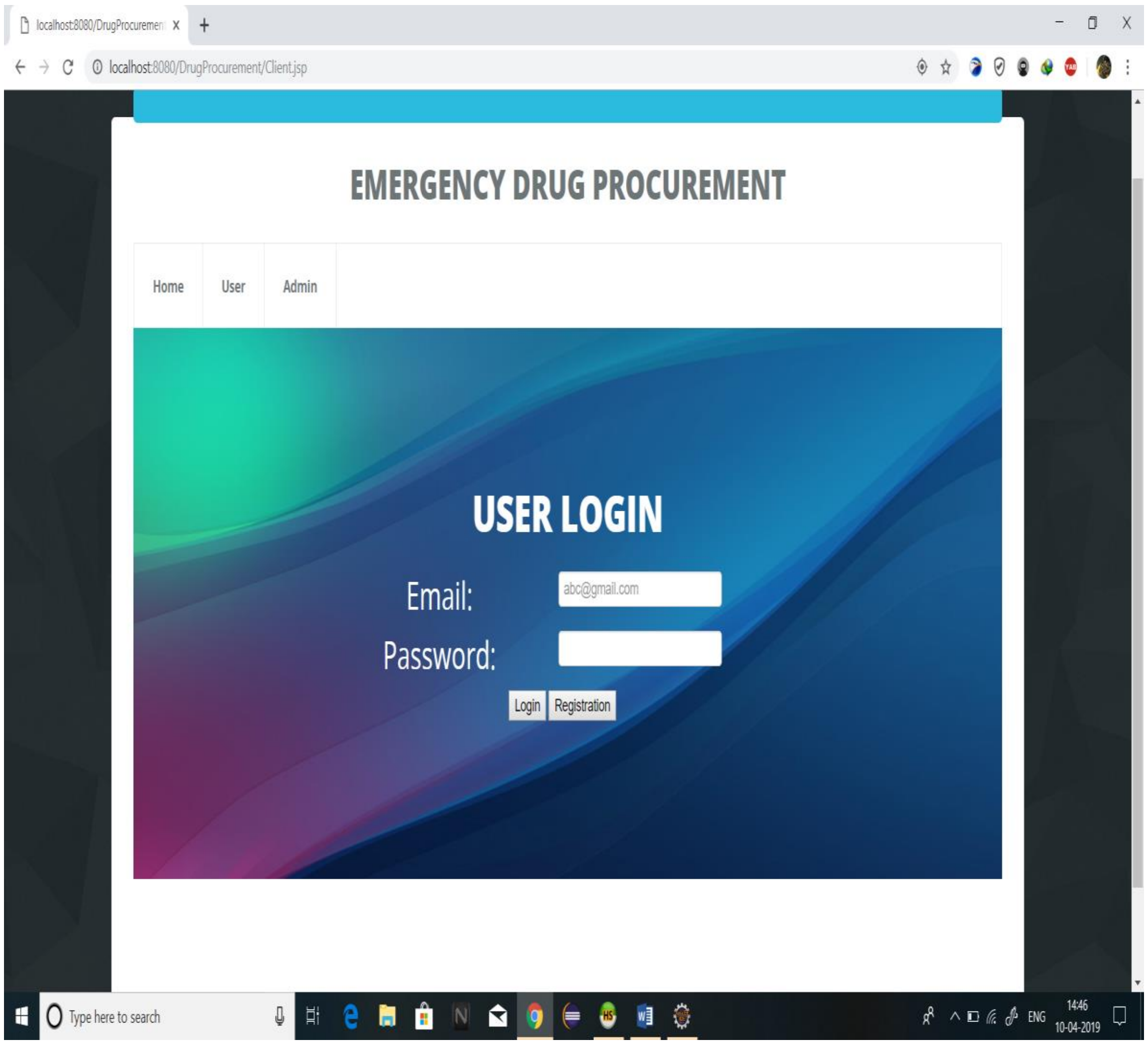
Add Medical :


localhost:8080/DrugProcurement/addmedical.jsp
EMERGENCY DRUG PROCUREMENT
Home
Add Medical
Add Drug
Logout
Search Location
Map
Satellite
Search Places
Pune
Latitude:
Longitude:
Medical Name:
Medical Address:
Email:
Mobile:
Password:
Submit
Type here to search
14:47
10-04-2019

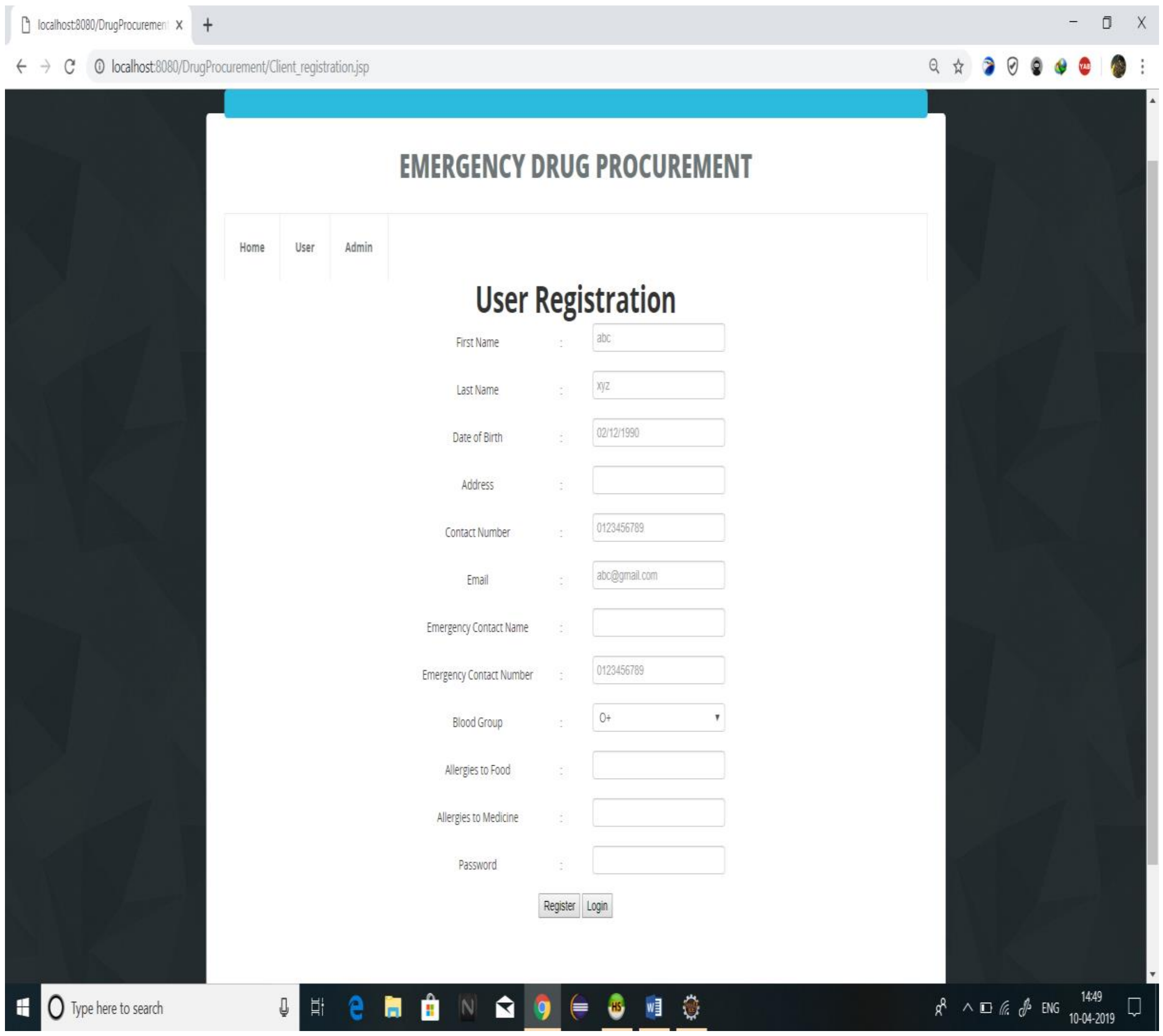
Add Medicine :



User Login :


localhost:8080/DrugProcurement/Client.jsp
EMERGENCY DRUG PROCUREMENT
Home
User
Admin
USER LOGIN
Email:
abc@gmail.com
Password:
Login
Registration
Type here to search
14:46
10-04-2019

User Registration :


localhost:8080/DrugProcurement/Client_registration.jsp

EMERGENCY DRUG PROCUREMENT

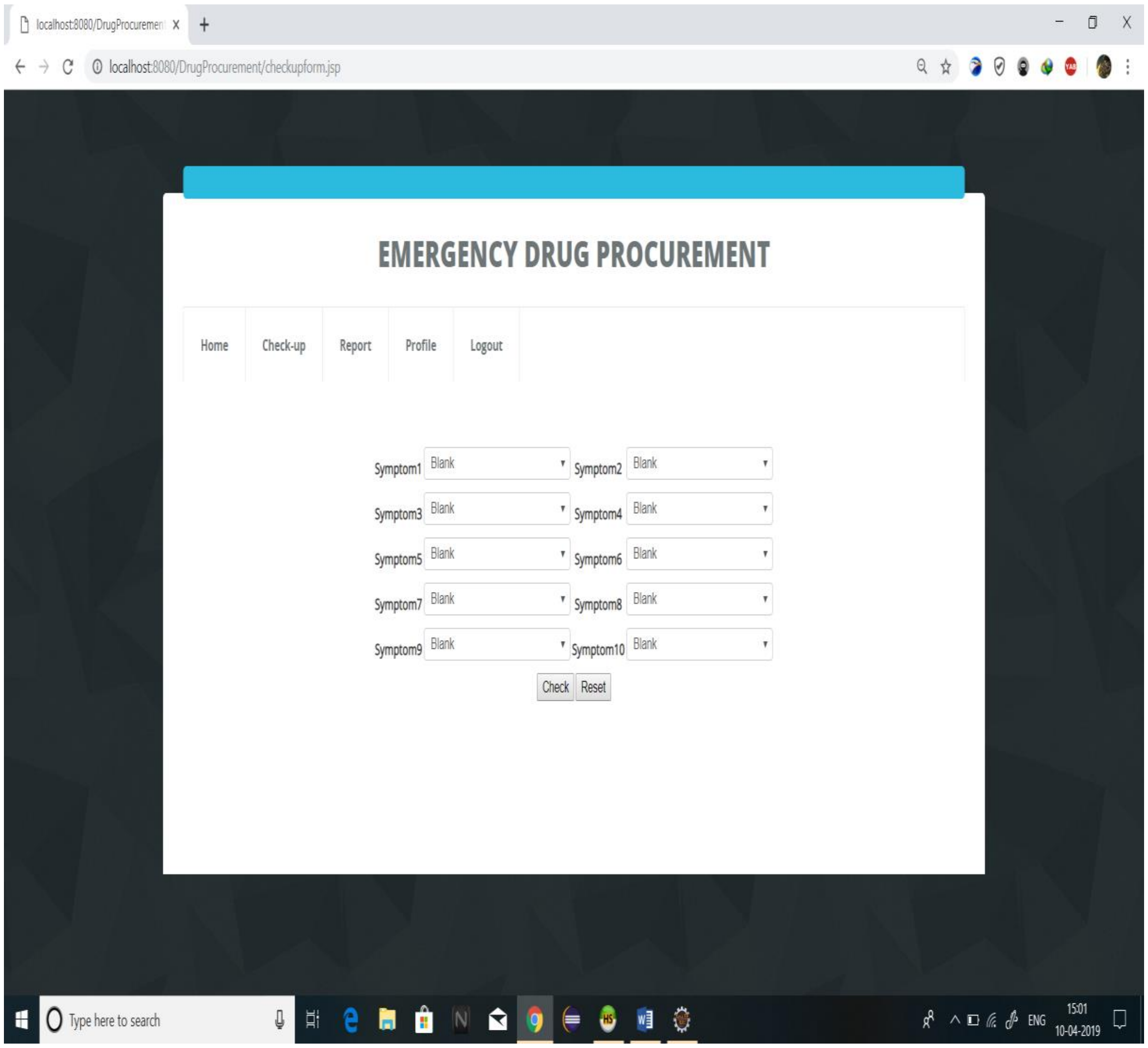
Home User Admin

User Registration

First Name	:	abc
Last Name	:	xyz
Date of Birth	:	02/12/1990
Address	:	
Contact Number	:	0123456789
Email	:	abc@gmail.com
Emergency Contact Name	:	
Emergency Contact Number	:	0123456789
Blood Group	:	O+
Allergies to Food	:	
Allergies to Medicine	:	
Password	:	

Register Login

Check up Form :



localhost:8080/DrugProcurement/checkupform.jsp

EMERGENCY DRUG PROCUREMENT

Home Check-up Report Profile Logout

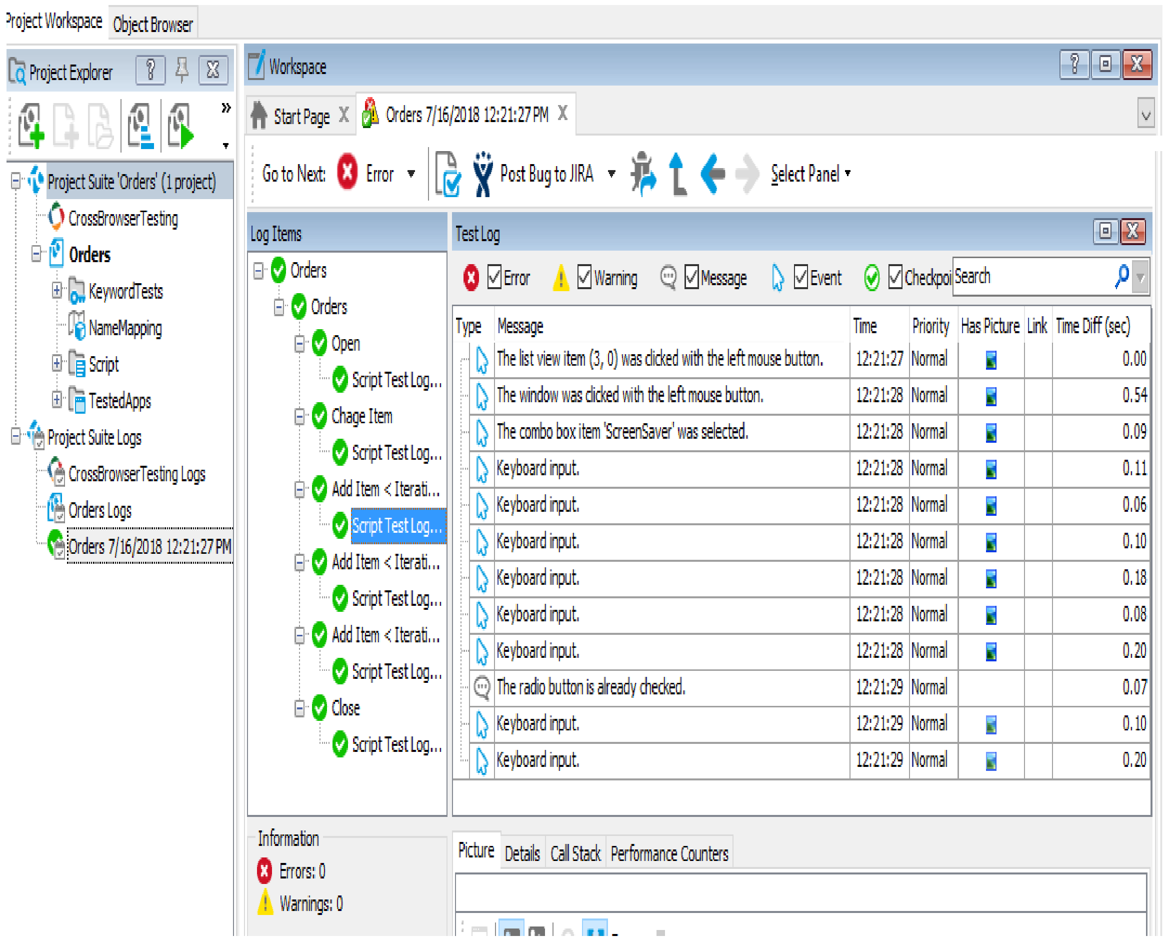
Symptom1	Blank	Symptom2	Blank
Symptom3	Blank	Symptom4	Blank
Symptom5	Blank	Symptom6	Blank
Symptom7	Blank	Symptom8	Blank
Symptom9	Blank	Symptom10	Blank

Check Reset

ANNEXURE 2

TEST LOG

Test Log:



Type	Message	Time	Priority	Has Picture	Link	Time Diff (sec)
	The list view item (3, 0) was clicked with the left mouse button.	12:21:27	Normal	▪		0.00
	The window was clicked with the left mouse button.	12:21:28	Normal	▪		0.54
	The combo box item 'ScreenSaver' was selected.	12:21:28	Normal	▪		0.09
	Keyboard input.	12:21:28	Normal	▪		0.11
	Keyboard input.	12:21:28	Normal	▪		0.06
	Keyboard input.	12:21:28	Normal	▪		0.10
	Keyboard input.	12:21:28	Normal	▪		0.18
	Keyboard input.	12:21:28	Normal	▪		0.08
	Keyboard input.	12:21:28	Normal	▪		0.20
	The radio button is already checked.	12:21:29	Normal			0.07
	Keyboard input.	12:21:29	Normal	▪		0.10
	Keyboard input.	12:21:29	Normal	▪		0.20