Software Test Management Tool

A dissertation submitted for the Degree of Master of Information Technology

J.B.A Gemunu Priyadarshana

University of Colombo School of Computing

2017

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: J B A Gemunu Priyadarshana

Registration Number: 2013/MIT/064

Index Number: 13550643

Signature: Date: 19/06/2017

This is to certify that this thesis is based on the work of

Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name:

Signature: Date:

Abstract

When it comes to test management support tools, there are a large variety of options to choose from. Some are open source and some are commercial. No doubt all these products offer some benefit, but why is the functionality so fragmented? For example, one of the commercial application QAComplete needs TestComplete to run automated tests. Is it simply badly organized product strategies brought on by a series of separate tool acquisitions? Perhaps each tool is designed for slightly different needs, but should we really be required to buy them all and then frequently jump between different tools? – No. Software support tools are supposed to make testing easier.
Testing activities need to be recorded in a single application despite it is manual or automated tests, and should have a track of activities performed by QA personals. Even though there is high number of open-source solutions out there for this mater, not single open-source solution would enable to run both manual and automated tests via single GUI. The very first consideration is therefore, is to look for a single solution that offers a wide variety of test support capability and secondly it should be able to share its data across other third-party applications like JIRA via API.

This Test Management Tool will serve as a web application that can be access over the internet to ease all QA activities. PHP is used as the programming language with CodeIgniter Web Framework. Also, MySQL is used as the database. The outcome of this Test Management Tool is to centralize all QA activities, starting from daily statuses to test case execution.

Table of Contents

List of Figures

List of Tables

List of Acronyms

BDT	Behavior Driven Testing
CDN	Content Delivery Networks
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
IE	Internet Explorer
NA	Not Applicable
PHP	PHP: Hypertext Preprocessor
QA	Quality Assurance

Chapter 1: Introduction

This chapter describes the problem which this project tries to address, motivation to come up with a solution, objectives and the scope of the proposed solution. This solution can be used by any client who is in to Software Quality Assurance.

1.1 Problem Statement

Let’s look at following two test management tools – QA Complete and TestRail. Both these commercial tools have very good features. But the main down side is that this tool does not provide test automation facility out of the box.

For example in order to execute automated tests using QA complete, client needs to buy TestComplete which is also made by the same company. So the customer has to pay if they want to continue with the same setup.

On the other hand TestRail does not have such mechanism to run automated test, but they have a rich API which allows updating TestRail. E.g. One could use the tear down step to update corresponding test case status (Pass, Fail, Blocked etc..) using the TestRail API. The API supports many languages E.g. Java, Python etc.

The proposed project tries to address above mentioned problems in the existing systems and further try to enhance and introduce new functionalities via web application.

1.2 Motivation

People involved in a software project wants to know when testing is completed and the testing progress. To be able to answer that question, it is important that test execution be tracked effectively. So, this is accomplished by collecting test data, or metrics, showing the progress of testing. The metrics helps to identify when corrections must be made to assure success. Additionally, using these metrics the testing team can predict a release date for the application.

QA engineers must document the details of a defect and the steps necessary to recreate it, or reference a test procedure if its steps expose the problem, to help the development team pursue its defect-correction activities. To achieve this, an adequate defect-tracking tool for the system environment is required. In most cases these issues are tracked in different types of issue tracking products. These products are not just focus on bug tracking – provides bug tracking, issue tracking, and project management functions as well. One such example is JIRA. Even though this is a good proprietary solution for defect tracking and project management, it does not have the built in capability of test management. There are third-party plug-ins like Zephyr for JIRA; but the cost and lack of support for automation is a key concern. For example to automate using Zephyr – JIRA, once need to purchase 2 licenses; one for Zephyr for JIRA and one for ZAPI the api for automation. Though this is a very costly approach, still there is no GUI to manage automated tests.

1.3 Objectives

The main aim of the project is to develop a user friendly web application to facilitate test management solution as well as to provide management information for effective decision making. Further:

- Increase productivity by 10% by removing manual excel maintains by introducing a GUI to Create, maintain and execute manual test cases.
- Simplify test execution providing a GUI to execute automated tests via web interface.
- Live monitor project health via test results dashboard to display both manual and automated test results.
- Centralize QA activities to measure productivity by displaying both manual and automated test results in a single web application.
- Reduce time to switching application by allowing to work in different projects at the same time.
- Improve User Experience by making both mobile and desktop friendly test management solution.
- Export data in to csv format for further needs.
- Export data in to pdf format.
- Make the testers life easier by providing a central place for all day to day activities.
- Daily stats to measure the productivity of the tester.
- Accurate estimates based on project and user daily stats.
- Communicate with third-party tools using an API.

1.4 Scope

The project is concerned with implementing a software solution to facilitate software test management solution with test automation features. Implementation will be web based software. Primary function of the software will be create, manage and execute test cases using a web application. Primary stakeholders are test automation team and management of the organization.

The application is not restricted to a single project and can be used across multiple projects within the organization.

Users of the software will be grouped as follows with different authorization levels: Administrator, Tester and Registered User.
Registered user can be any user who may browse the dashboard. Registered user can only view dashboards in their respective project. These are typically development people/ managers who are interested in project dashboards.

Testers can login to the system and create test suites and add test cases for those test suites. Once test case is approved/ reviewed by peers they will be executed in test cycles. Testers can add can export test data in to csv.

User creation will consist of images of the user and other user details. Not only users can create manual test cases but also they could execute automated test cases using the web application. Admin user can make a user active / inactive from the user management page.

A single test case is divided in to three sections – Organize, Design and Execute. Test details will be saved as per the execution cycle, such that historical data will be available with time and these data can be used for test estimations in future projects.

The daily status section is designed with five sections by identifying tasks that QA will do on daily basis. Summary section, Test Design, Automated, Executed and Issues Identified are the four sections. Summary section includes daily tasks they carried out and tomorrows plan and their blocking issues. The other three sections contain number of test cases based on the test severity – High, Medium, and Low. The last section contain number of issues identified divided in to Showstopper, High, Medium, Low.

The Dashboard section displays the number of test cases with the execution statuses in Graphical manner. Further the individual activity/productivity will be displayed based on the daily status stats.

1.5 Outline of the Dissertation

Chapter 1 describes the problem which the Software Test Management Tool project addresses. Chapter describes the problem statement, motivation, objectives and scope.

Chapter 2 describes the domain of the business problem, which is Software Quality Assurance. Chapter describes the study of business background, existing solutions and their feature comparison.

Chapter 3 describes the analysis and design of the implementation of the project. Chapter describes fact gathering techniques, use cases, functional/ nonfunctional features, design for the solution.

Chapter 4 describes implantation of the solution. Chapter describes implementation decisions and justifications, technologies and frameworks adopted.

Chapter 5 describes testing and evaluation of the solution. Chapter describes test plan, test cases, performance testing and usability testing.

Chapter 6 concludes the project work. Chapter describes results, challenges faced; lessons learnt and identified areas for improvement.

Chapter 2: Background

This chapter describes the background of software test management tools and their usages.

2.1 Introduction

There are many test management solutions available and each of them have their own advantages and disadvantages.
Test Rail [1]: One of the key advantages is that TestRail makes it very easy for estimating QA effort in sprints. It’s also very useful for organizing test efforts on the fly once test suites are created and organized. The JIRA plugin for TestRail is working and the ability to create test cases off of user stories is quite useful. Further UI: very intuitive and easy to digest and does not have to watch tutorials how to use the tool. Ability to log bug right from the test rail makes it easy since no need to open JIRA for that.

2.2 Existing Solutions

Following is an analysis of currently available test management solutions.
TestRail Test Management solution:
Test rail is a effective TCM (Test Case Management) tool, that helps to organize and manage the work of the QA team. However, these built in tools weren’t always provide the solution out of the box. One such example is the test automation. It comes with a HTTP-based API to integrate with the automated test results. By making use of the TestRail API, users can report test results in to test rail. Additionally, it integrates with many issue tracking tools that makes requirements from external systems to be linked to test cases in TestRail; bugs can also be created in the external systems and links can be established to the corresponding test case.

QAComplete Test Management Solution:
QAComplete is a powerful, flexible test management tool that helps users easily manage requirements, tests and defects all in one place. The tool is easy to use, and provides a central hub to manage and report on all of your tests – manual, Selenium, TestComplete, SoapUI. To make automation work TestComplete needs to be purchased.
It is customizable enough to fit into any development process, from Waterfall to Agile, and integrates tightly with the project management and workflow tools such as Jira, Bugzilla, Visual Studio and more.

QAComplete is a QA testing software solution that provides a centralized zone for all testing activity. After a test is completed, the program stores it inside a library and allows users to pull it up later for future review and drill down into details at a granular level.

Users can then decide to implement further tests and suggest new testing activities. The system additionally has tools specifically for testing mobile applications, such as time budgeting, industry-specific guideline enforcements and support for native, hybrid and web apps. Lastly, QAComplete can automatically create defect reports as users run into defects during testing.

QAComplete Key Features

- SaaS or on-premise QA testing software solution
- Provides a centralized zone for all product testing
- Stores all tests in a library
- Users can drill down to a granular level within each test report
- Provides suggestions for further testing and new testing routines
- Has tools for mobile app testing
- Can create defect reports if users have problems during testing

2.3 Feature Comparison

Table 2.1 tabulates the feature comparison of discussed existing solutions.

Feature	QAComplete	TestRail	This System
User Creation	✓	✓	✓
Project Allocation	✓	✓	✓
Test Automation	-	-	✓
API	-	✓	✓
JIRA Plug-in	✓	✓	-
Daily Status	-	-	✓
SMS Notifications	-	-	✓
Mobile friendly UI	-	-	✓
Export to CSV	✓	✓	✓
Export to PDF	✓	✓	✓
Import from CSV	✓	✓	-

Table 2.1 Feature comparison of existing solutions

Above feature comparison shows that each system facilitates users to perform basic tasks from user creation to tests management. Each of them lacks process to execute automated tests.

Chapter 3: Analysis and Design

This chapter describes the analysis of the problem domain; identified stakeholders, methods used for requirements gathering, and identified functional requirements and non-functional requirements.

3.1 Analysis

This sub chapter describes the analysis of the problem domain; identified stakeholders, methods used for requirements gathering, and identified functional requirements and non-functional requirements.

3.1.1 Fact Gathering Techniques

Requirements were gathered using multiple approaches for better understanding. Approaches used were as follows.

- Discussions and Interviews

Interviews were arranged with individuals that are familiar with the context. Further clarifications were made through discussions with QA personals. After the discussion with different levels of QA people, discovered that their requirements are different based on the designation.

- Studying similar systems

Studying similar systems such as Test Rail and QA Complete helped to identify requirements which are not considered by existing systems. Also studying similar systems helped to implement an operational prototype within a brief time period.

- Practicing the manual steps

Practicing the manual QA process helped to understand the possible improvements for existing systems. From requirement gathering to maintains, QA activities were identified.

- Using web resources

Reading about similar test management applications helped to accelerate the requirements capturing process. For example Test Rail API document helped to gather vital information and how to structure the data base as wel.s

3.1.2 Requirements Analysis

During the requirement analysis process actors of the system was identified. Then the use cases were identified and documented. There after the necessary diagrams were drawn and functional and nonfunctional requirements are listed.

3.1.2.1 Actors of the System

Following actors were identified during the analysis.

- Registered user – Users looking to view dashboards
- Tester - Users who create and manage tests
- Admin User – Create, Manage users and projects and assign them accordingly

3.1.2.2 Use Cases

Figure 3.1 illustrates the use case diagram for the system.

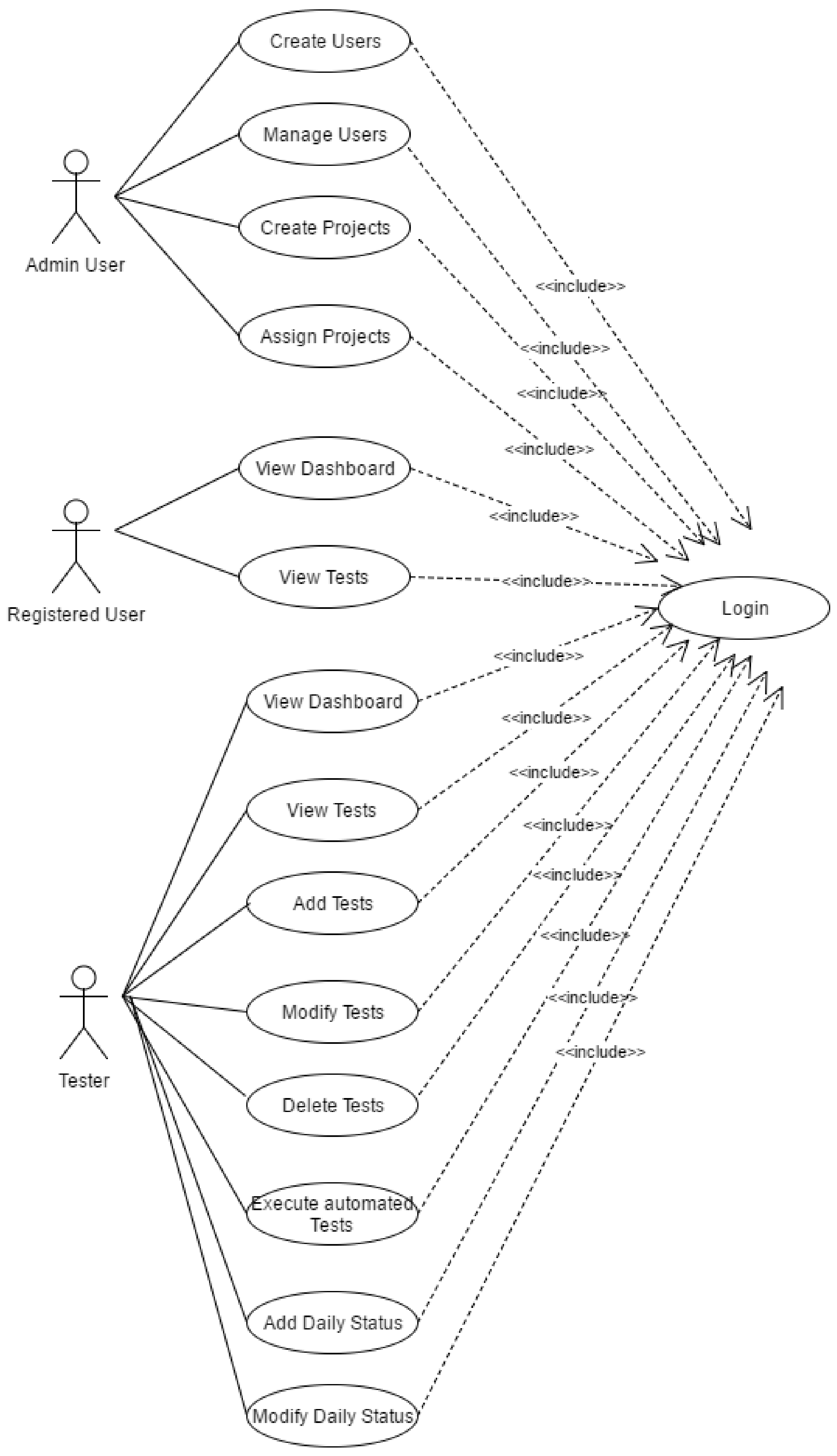


Figure 3.1 Use Case Diagram

Table 3.1 tabulates the use case with respective actor, scope, complexity, priority. Please refer Appendix C for use case scenarios in detail.

Use Case ID	Use Case Name	Primary Actor	Scope	Complexity	Priority
UC-1	Create new users	Admin User	In	Medium	1
UC-2	Manage users	Admin User	In	High	1
UC-3	Create projects	Admin User	In	High	1
UC-4	Manage projects	Admin User	In	Medium	1
UC-5	Assign users	Admin User	In	Medium	1
UC-6	Change password	Registered user	In	Medium	1
UC-7	Login	Registered user	In	Medium	1
UC-8	View dashboard	Registered user	In	Medium	2
UC-9	View tests	Registered user	In	Medium	1
UC-10	Add tests	Tester	In	High	1
UC-11	Search tests	Tester	In	High	1
UC-12	Modify tests	Tester	In	High	1
UC-13	Delete tests	Tester	In	High	1
UC-14	Execute automated tests	Tester	In	High	1
UC-15	Add daily status	Tester	In	High	1
UC-16	Modify daily status	Tester	In	High	1

Table 3.2 Use Case Index

3.1.3 Requirements for the proposed solution

3.1.3.1 Functional Requirements

Identified functional requirements are mentioned below.

- Login

User should be able to login to the system and later they should be able to change login credentials. Each type of user should be granted with authority to access system functions according to their privilege levels.

- Create test suites and test cases

Un-authorized users should not be able search browse and view test cases

- Test dashboards, test cases and daily statuses should not be visible to un-auth usres.

Authenticated users should be able to view the dashboard of assigned projects

- View dashboards

Authenticated users should be able change password

- Manage users

Tester (user) should be able to create test suites and add test cases.

- Search, browse and export to csv test cases

Administrators should be able to create users and projects. Administrators should be able to add users to projects and modify allocations as well.

- Manage users

3.1.3.2 Nonfunctional Requirements

Nonfunctional requirements are discussed under following categories. These requirements must be achieved at a system-wide level rather than at a unit level.

- Performance – System should be able to handle 50 users simultaneously.
- Availability – System should be available to access over the internet and should operate properly on IE, Google Chrome, and Fire Fox browsers.
- Security – Unauthorized users should not be able to access system or the production servers.
- Maintainability – System design should facilitate the implementation and deployment of requirement changes without complete re-installation of the system.
- Legal – Terms and Condition, Policy page should be implemented.

3.2 Design

This sub chapter describes the design for the proposed solution. This chapter will provide aid for software development with high level architecture of the system and details of its components. Further constraints and assumptions will be discussed.

3.2.1 Architectural Design

In order to address functional and nonfunctional requirements system is developed following client server architecture. System will be installed on central server and made accessible through a web browser. This enables users to access the system using various kinds of devices. And the application uses responsive bootstrap theme such that they could view the site on different devices

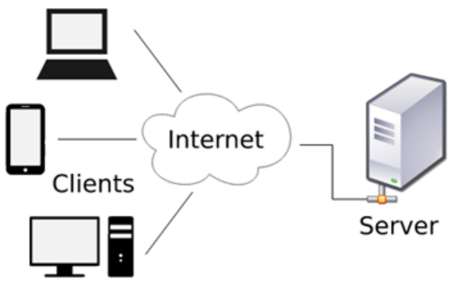


Figure 3.2 Client Server Architecture

Software architecture concerns about separating system of interest into abstract sub systems. This enables the development of the system to be organized and traceable. In case of the proposed solution three-tier architecture is selected since it is suitable for internet driven web applications. Figure 3.3 illustrates communication and order of the three-tier architecture.

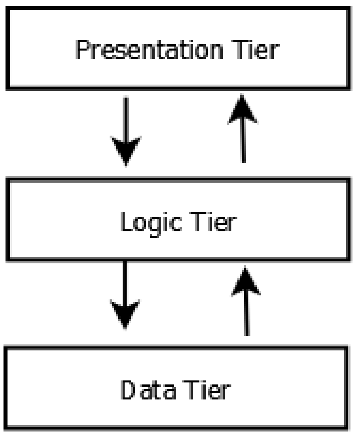


Figure 3.2 Three-Tier Architecture

3.2.2 Component Level Design

Following figure shows high level decomposition of the system.

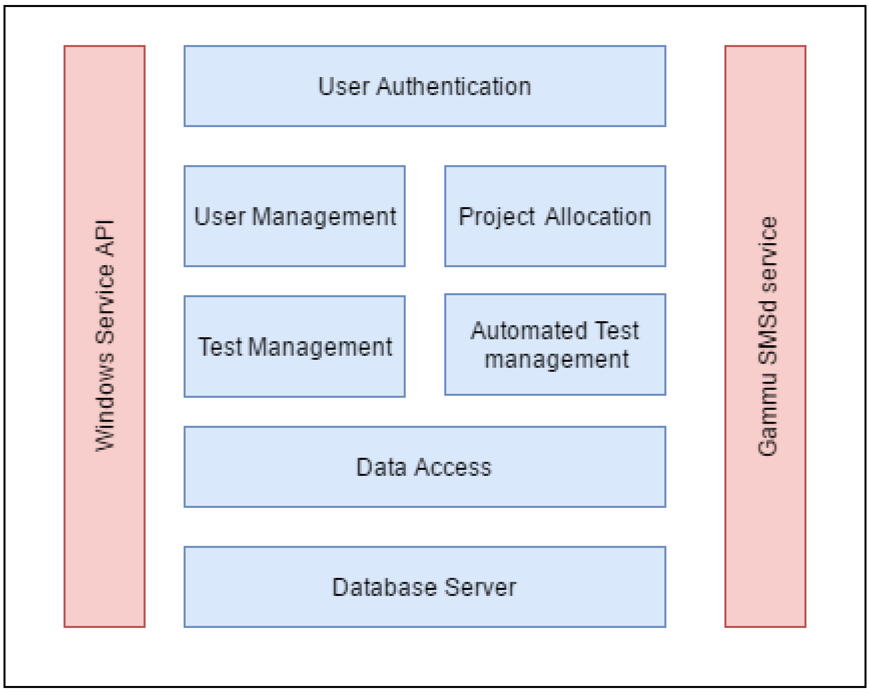


Figure 3.4 Component diagram

3.2.3 User Interface Design

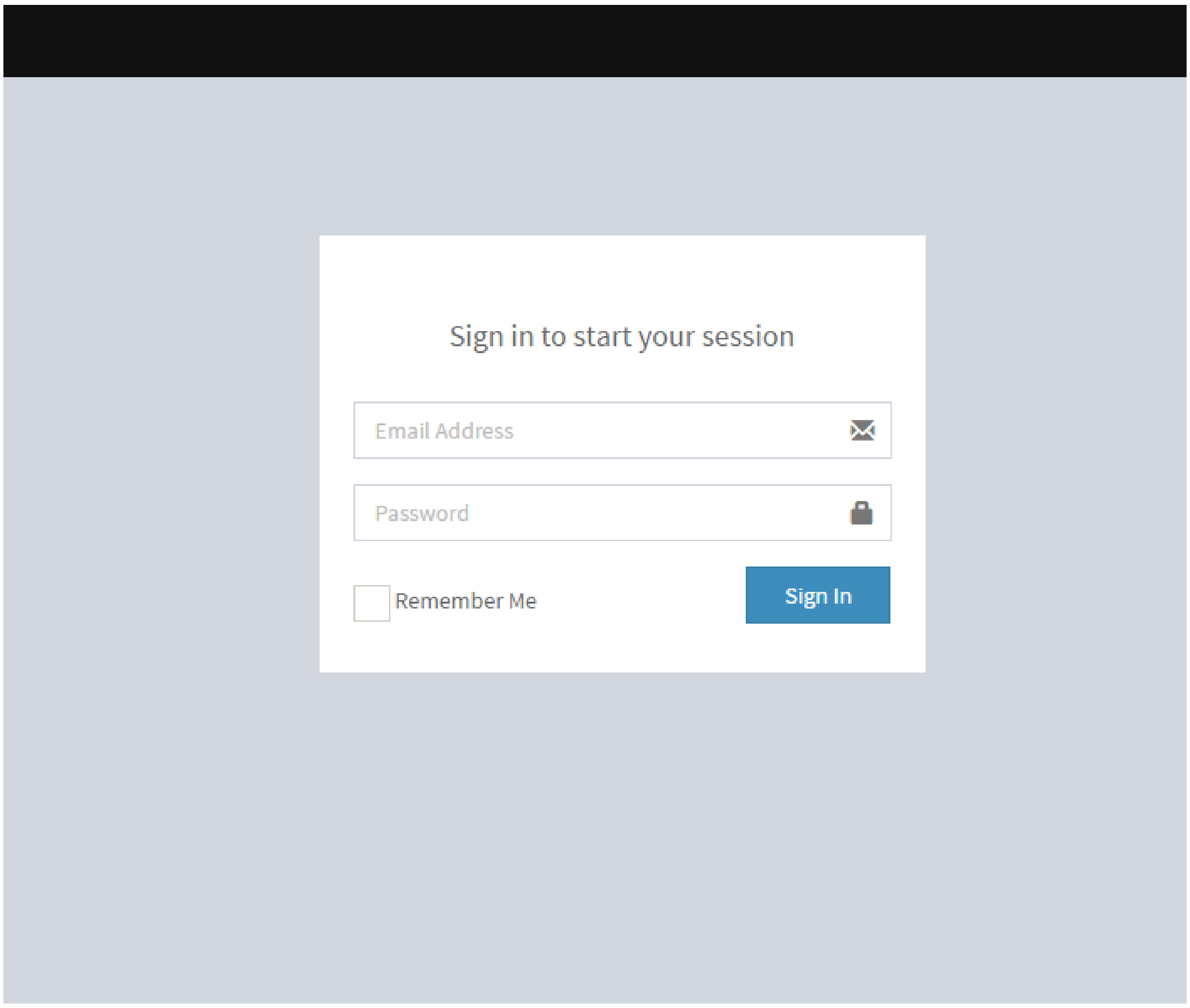
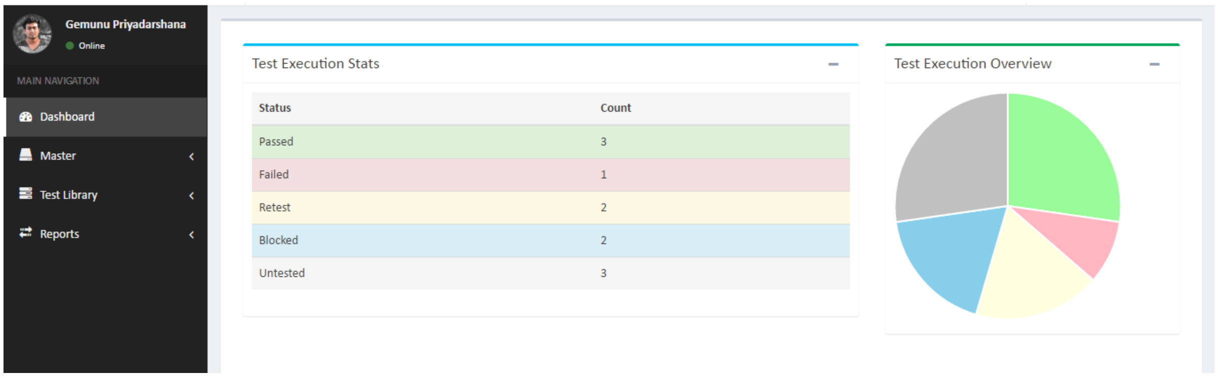


Figure 3.5 Login Window

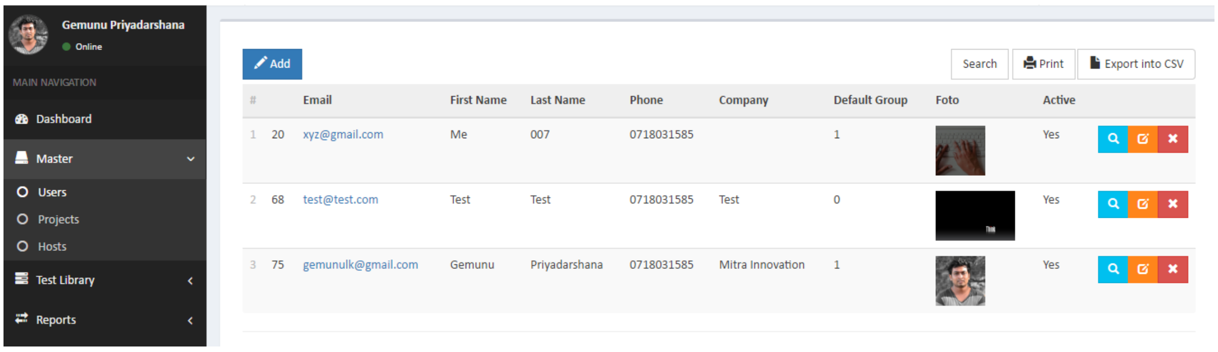
Dashboard



Status	Count
Passed	3
Failed	1
Retest	2
Blocked	2
Untested	3

Figure 3.6 Dashboard Window

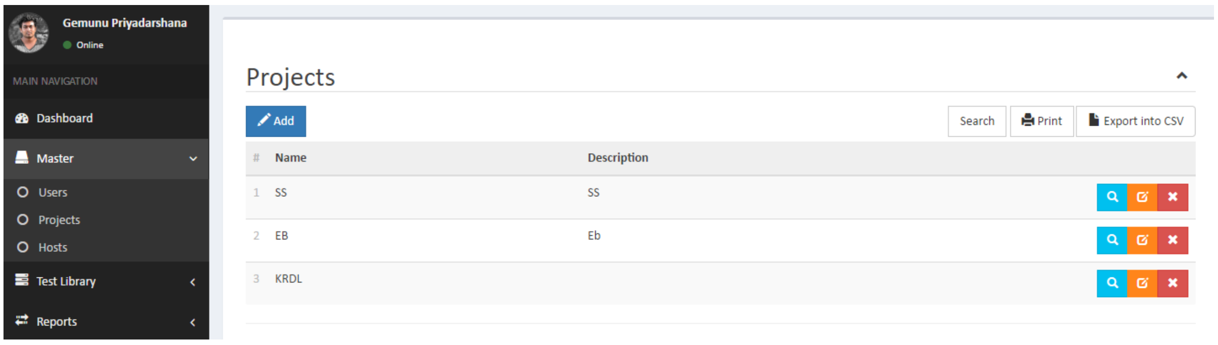
User management interface



#		Email	First Name	Last Name	Phone	Company	Default Group	Foto	Active
1	20	xyz@gmail.com	Me	007	0718031585		1		Yes
2	68	test@test.com	Test	Test	0718031585	Test	0		Yes
3	75	gemunulk@gmail.com	Gemunu	Priyadarshana	0718031585	Mitra Innovation	1		Yes

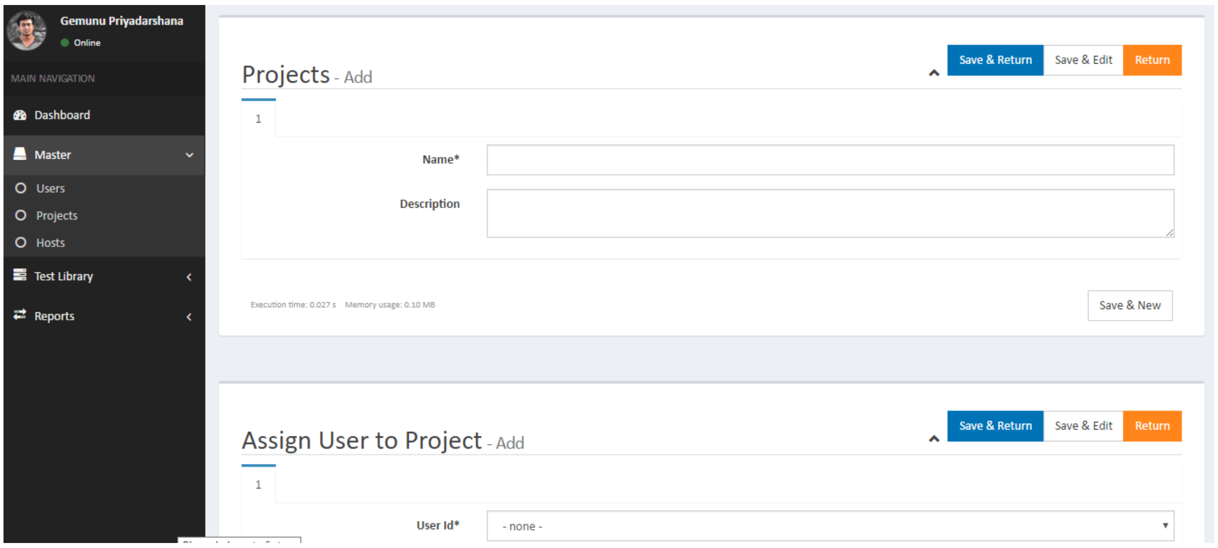
Figure 3.7 User management Window

Project management interface



#	Name	Description
1	SS	SS
2	EB	Eb
3	KRDL	

Figure 3.8 Project management Window



Projects - Add

Save & Return | Save & Edit | Return

1

Name*	
Description	

Execution time: 0.027 s Memory usage: 0.10 MB

Save & New

Assign User to Project - Add

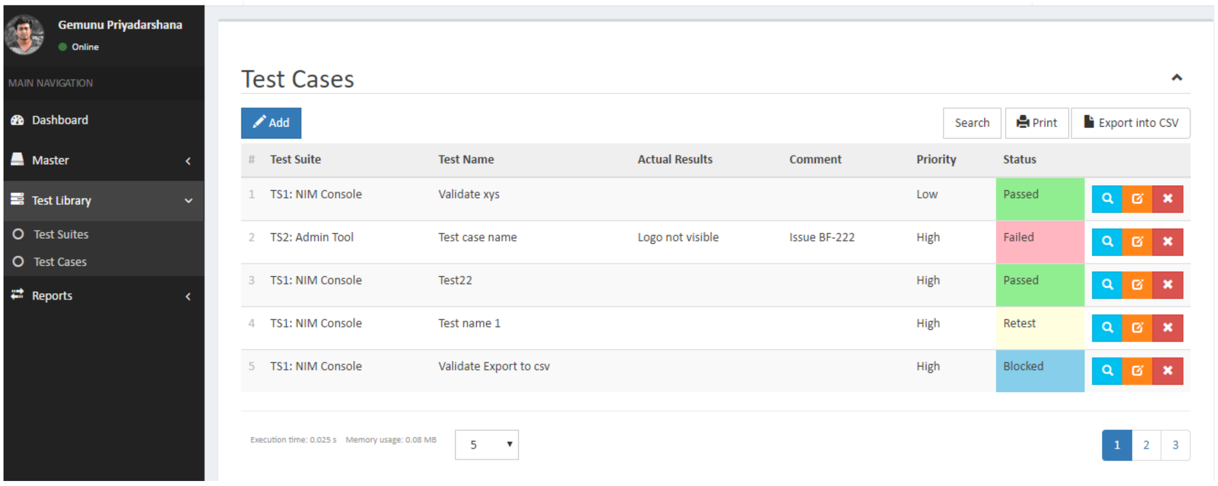
Save & Return | Save & Edit | Return

1

User Id*	- none -

Figure 3.9 User Project Window

Test management interface



Test Cases

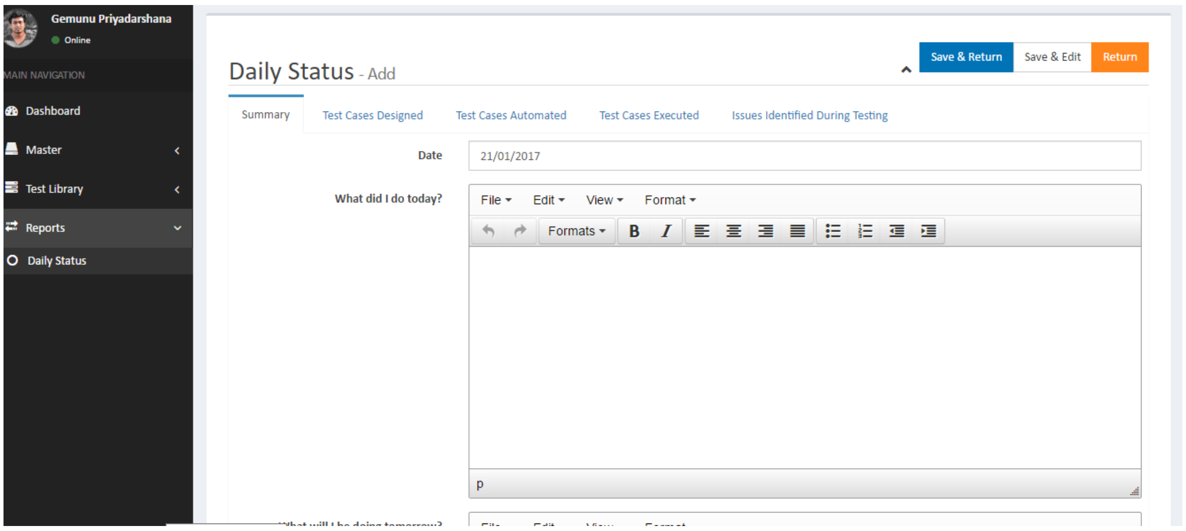
Add | Search | Print | Export into CSV

#	Test Suite	Test Name	Actual Results	Comment	Priority	Status
1	TS1: NIM Console	Validate xys			Low	Passed
2	TS2: Admin Tool	Test case name	Logo not visible	Issue BF-222	High	Failed
3	TS1: NIM Console	Test22			High	Passed
4	TS1: NIM Console	Test name 1			High	Retest
5	TS1: NIM Console	Validate Export to csv			High	Blocked

Execution time: 0.025 s Memory usage: 0.08 MB 5 | 1 2 3

Figure 3.10 Test management Window

Daily status interface



Daily Status - Add

Save & Return | Save & Edit | Return

Summary | Test Cases Designed | Test Cases Automated | Test Cases Executed | Issues Identified During Testing

Date	21/01/2017
What did I do today?	p

Figure 3.11 Daily Status Window

3.2.4 Database Design

Following figure shows the relational database design for the system.

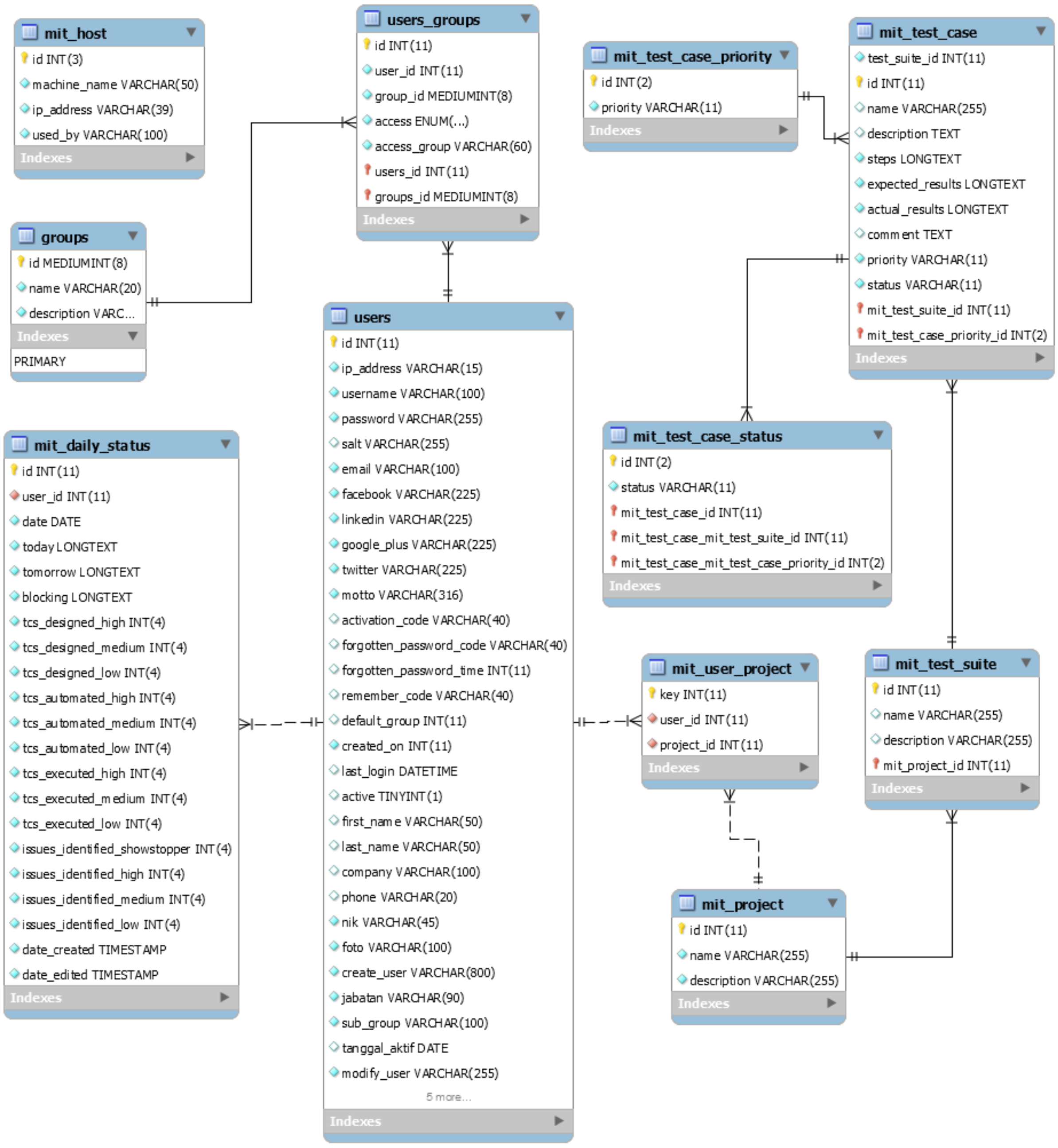


Figure 3.4 Database diagram

3.2.5 Constraints and Assumption

- Due to limitations in the tool use to automate testing, the HTML test results and logs generated by the automated test scripts won't be pushed back to the test management application.

- The windows service used to trigger automated test script(s) needs to be in running state.
- Automated test execution can only be done in windows environment as of now. To run in other platform, need to implement separate services for that.

Chapter 4: Implementation

This chapter describes the implementation of the proposed system. This chapter will cover development environment configuration, implementation decisions justification, module and namespace decomposition, and acknowledgement of open source and community, frameworks and software.

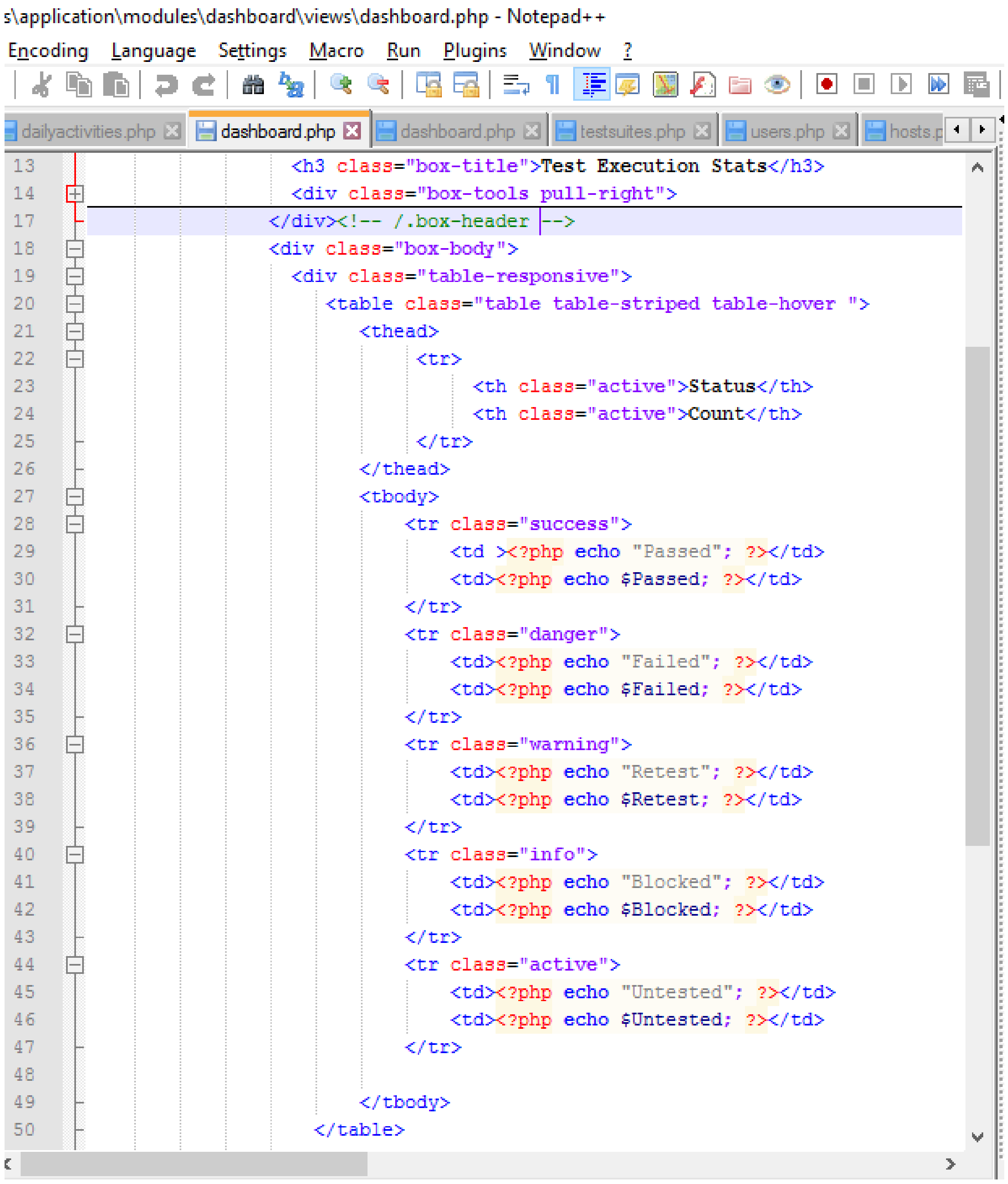
4.1 Development Environment

Development environment is described in perspective of software and hardware. Following software stack was selected for the development if the proposed solution.

- Operating system: Microsoft Windows 8.1
- Web server: Apache web server
- Web browsers: IE, Firefox, Chrome
- PHP Version 5.5.15
- xCRUD Data management system
- MySQL

4.3 Code Artifacts

View

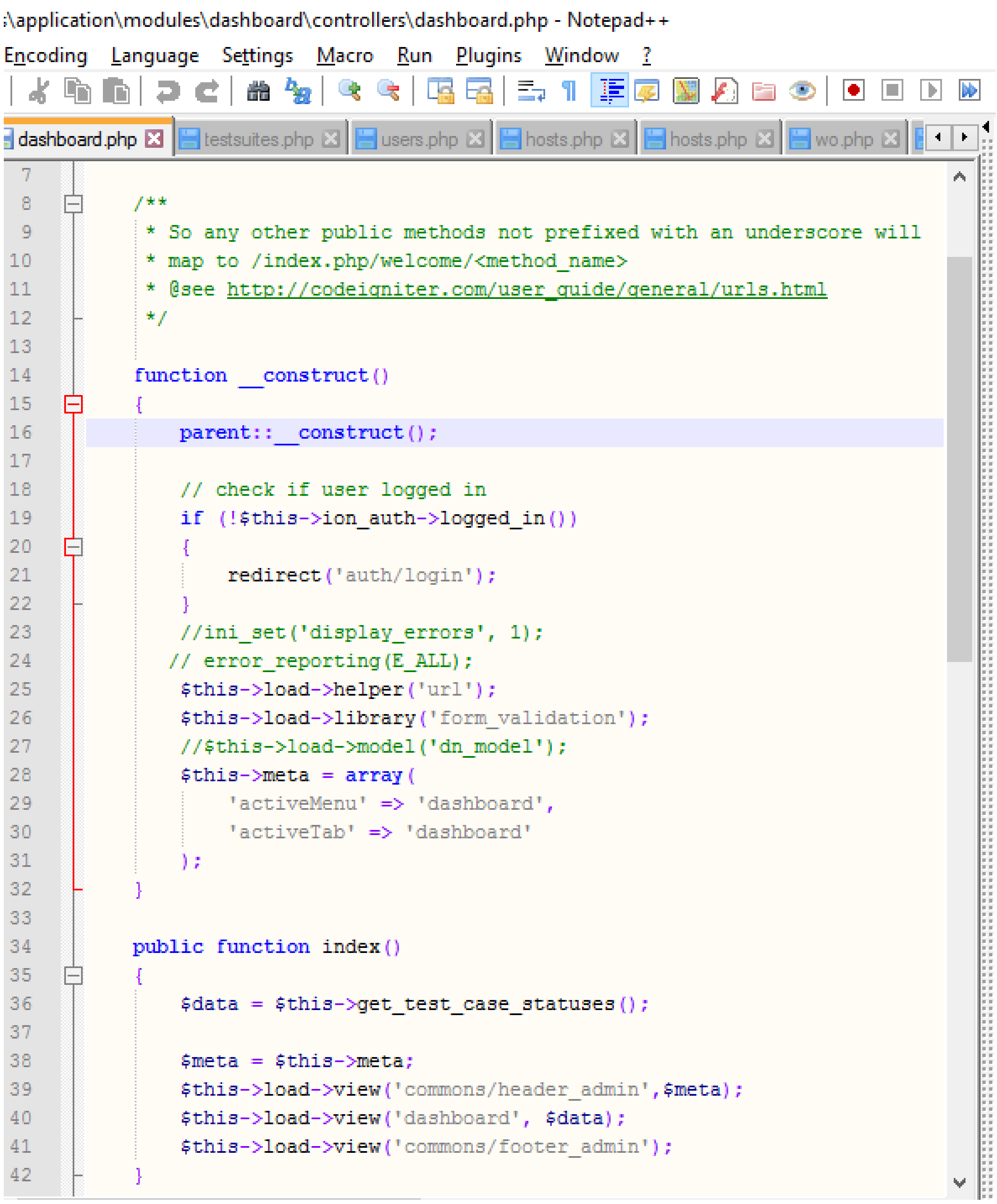

s\application\modules\dashboard\views\dashboard.php - Notepad++
Encoding Language Settings Macro Run Plugins Window ?
dailyactivities.php
dashboard.php
dashboard.php
testsuites.php
users.php
hosts.p

```
  <h3 class="box-title">Test Execution Stats</h3>
  <div class="box-tools pull-right">
</div><!-- /.box-header -->
<div class="box-body">
  <div class="table-responsive">
     <table class="table table-striped table-hover ">
        <thead>
             <tr>
                  <th class="active">Status</th>
                  <th class="active">Count</th>
             </tr>
        </thead>
        <tbody>
            <tr class="success">
                <td ><?php echo "Passed"; ?></td>
                <td><?php echo $Passed; ?></td>
            </tr>
            <tr class="danger">
                <td><?php echo "Failed"; ?></td>
                <td><?php echo $Failed; ?></td>
            </tr>
            <tr class="warning">
                <td><?php echo "Retest"; ?></td>
                <td><?php echo $Retest; ?></td>
            </tr>
            <tr class="info">
                <td><?php echo "Blocked"; ?></td>
                <td><?php echo $Blocked; ?></td>
            </tr>
            <tr class="active">
                <td><?php echo "Untested"; ?></td>
                <td><?php echo $Untested; ?></td>
            </tr>

        </tbody>
    </table>
```

Figure 4.3 View

Controller



:\application\modules\dashboard\controllers\dashboard.php - Notepad++

```
    /**
     * So any other public methods not prefixed with an underscore will
     * map to /index.php/welcome/<method_name>
     * @see http://codeigniter.com/user_guide/general/urls.html
     */

    function __construct()
    {
        parent::__construct();

        // check if user logged in
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login');
        }
        //ini_set('display_errors', 1);
       // error_reporting(E_ALL);
        $this->load->helper('url');
        $this->load->library('form_validation');
        //$this->load->model('dn_model');
        $this->meta = array(
            'activeMenu' => 'dashboard',
            'activeTab' => 'dashboard'
        );
    }

    public function index()
    {
        $data = $this->get_test_case_statuses();

        $meta = $this->meta;
        $this->load->view('commons/header_admin',$meta);
        $this->load->view('dashboard', $data);
        $this->load->view('commons/footer_admin');
    }
```

Figure 4.2 Contoller

4.4 Implementation of Non Functional Requirements

Implementation of non-functional requirements is completed using Framework features which were used in implementation. These features decreased the implementation effort drastically by allowing developer to focus on implementing functional requirements.

4.4.1 Performance

Performance of the systems was improved by file compression, html caching, reducing number of calls to the server and controlling the order of the script rendering.

4.4.2 Availability

Availability is considered in two perspectives, availability across different browsers/devices and availability over time. Availability across different browsers/devices is implemented by Bootstrap CSS framework. Bootstrap supports all the major browsers IE, Chrome, FireFox, Safari and devices with different screen sizes.

4.4.3 Security

Code Igniter framework comes with XSS filtering security. This filter will prevent any malicious JavaScript code or any other code that attempts to hijack cookie and do malicious activities. This reduces security risks.

4.3.4 Maintainability

System is designed and implemented in a modular manner using MVC pattern. Coupling between these modules are low. Communications between these modules are contracted with interfaces. Therefore change to a certain module can be done without effecting rest of the implementation. Use of comments, coding standards and unit tests may assist future changes to the system.

4.3.5 Portability

Apache web server, MySQL and PHP is used to develop the application. The application is currently deployed on XAMP server and this can easily be migrated to Linux environment and hosted in LAMP server.

Chapter 5: Evaluation and Testing

This chapter describes the evaluation and testing of the proposed system. This chapter will cover the scope of the testing, test plans, test cases, test execution and test results. Later the performance testing and acceptance testing is discussed.

5.1 Test Plan

This sub chapter describes the project work related details from a QA perspective. Testing approach for each test type, entry and exit criteria for each test type and defect life cycle is discussed.

5.1.1 Test Execution

Figure 5.1 shows the test execution approach for the system.

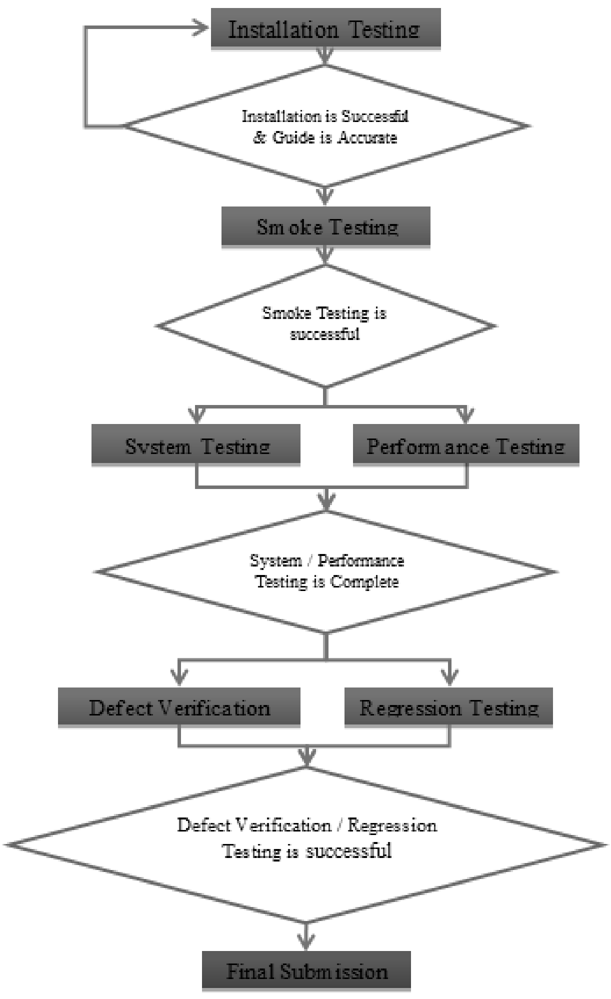

Installation Testing
Installation is Successful & Guide is Accurate
Smoke Testing
Smoke Testing is successful
System Testing
Performance Testing
System / Performance Testing is Complete
Defect Verification
Regression Testing
Defect Verification / Regression Testing is successful
Final Submission

Figure 5.1 Test execution approach

5.1.1.2 Smoke Testing

Smoke test is the process of examining the main components of the system, to ensure that they function as expected. Typically, Smoke testing is conducted immediately after an installation test. Successful completion of smoke testing indicates that the application is stable enough for testing to commence. The smoke testing will validate the following:

The accuracy of the main functionality of the application

The next level of testing can be carried out without any major interruptions.

Entry Criteria

- Installation test is completed.
- Relevant data is available and databases are setup.
- Application is started successfully.

Exit Criteria

- Smoke test is successfully completed.

5.1.1.3 System Testing

System testing will verify the system's functionality against the identified requirements specified in the chapter 3.

Entry Criteria

- Code is unit tested. All high and critical unit testing defects are fixed. (If there are any known defects in the release, these should be documented).
- Application is successfully installed.
- Smoke testing is successful.

Exit Criteria

- All the System test cases are executed.
- Any defects identified are recorded.

5.1.1.4 Regression Testing

The main focus of this testing is to ensure that the defect fixes have not impacted the functionalities which was testing in the System Testing cycle.

Entry Criteria

- Application is successfully installed.
- Smoke testing is successful.

Exit Criteria

- All known issues are documented (with workarounds, if necessary).

5.1.2 Defect Management

Figure 5.2 illustrates the defect life cycle.

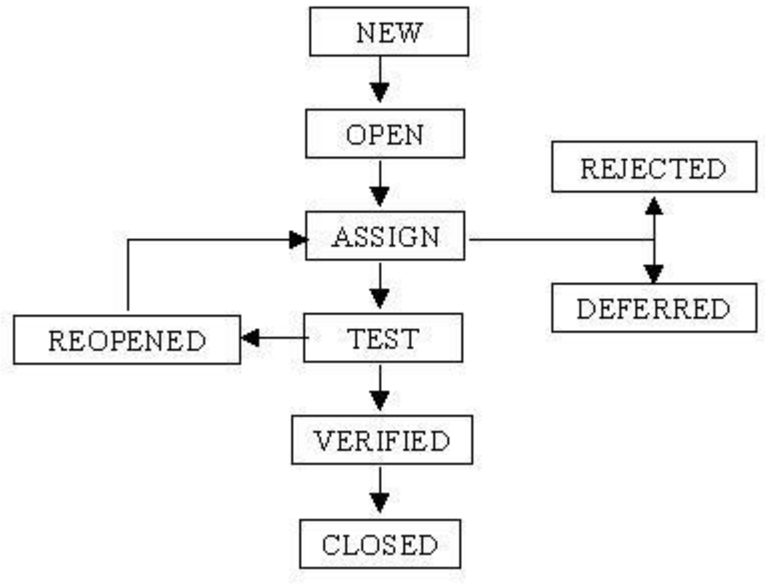


Figure 5.2 Defect life cycle

Chapter 6: Conclusion and Future Work

This chapter reviews the complete project work performed in an abstract level. Challenges faced throughout the project, and lessons learnt are also discussed.

6.1 Results and Review

Web application was developed and deployed to the apache web server, such that users can access the system via internet. Web application achieved all its objectives in the functional level, by covering different user roles and responsibilities. Operational level success needs to be measured after the application is opened for users and keeping live for a considerable amount of period.

- Increase productivity by 10% by removing manual excel maintains.
- Centralized QA activities to measure productivity by displaying both manual and automated test results in a single web application.
- Reduced time to switching application by allowing to work in different projects at the same time.
- Improved User Experience by making both mobile and desktop friendly test management solution.

6.2 Challenges Faced During the Project

Automated test execution using the GUI was challenging. As the project came to the development of the test execution with a windows service, it proved to be time and effort consuming than estimated. Finally, the feature was implanted enabling automated test execution using GUI.

6.3 Lessons Learnt During the Project

The entire software development lifecycle helped to understand some of the software engineering theory with clarity. Specially, how software design practices are helpful when implementing software.

The intended application was intended to be used to test other software products. So the accuracy should be high and zero defects are expected.

Complex use cases took more effort and time due to lack of IDE support and frameworks.

6.4 Future Work

The automated test execution through the web application is currently supported only for windows platform only. The windows service enables this feature and for other platforms like Linux and Mac OS the supporting service needs to be re implement.

The html test results and logs generated by the automated test framework will not be push back to the test management application. It would be ideal to see the failed tests with the failed screen shots in a central place – in the test management application.

References

[1] Admin. (2017, January) Gurock. [Online].
http://www.gurock.com/testrail/

[2] Admin. (2017, January) http://php.net/. [Online].
http://php.net/manual/en/intro-whatis.php

[3] wikipedia.org. (2017, April) wikipedia.org. [Online].
https://en.wikipedia.org/wiki/Object-oriented_programming

[4] Microsoft MSDN. (2015, January) microsoft.com. [Online].
https://msdn.microsoft.com/en-us/library/x9afc042.aspx

Appendix A: Detailed Use Cases

Use Case ID	UC-1
Title	Create new users
Description	Admin user create new user(s)
Primary Actor	Admin User
Preconditions	User should be login to the System as Admin user
Post conditions	A new user should be added to the system with the default password.
Main Scenario	1. Admin user navigates to the Users module 2. Admin user enters user details and decides the user role 3. Admin user Save user details by clicking on Save & Return 4. A new user should be created in the system 5. Newly created user should be able to login to the system, providing correct user name and default password (which can be changed later using change password section)
Alternative Scenario	1. Admin user does not enter mandatory fields at step 2 and try to save 2. Application should display Error message.
Complexity	Medium
Priority	1

Use Case ID	UC-2
Title	Manage users
Description	Admin user manage users
Primary Actor	Admin User
Preconditions	Admin user has successfully created a user.
Post conditions	Changes made to the user profile should get persisted.
Main Scenario	1. Admin user navigates to the Users module 2. Clicks on edit user button for an already existing user 3. Un-tick the Active option and made user inactive and Save changes. 4. Inactive user no longer can login and with correct credentials Account is inactive message should be prompted.
Alternative Scenario	1. Admin user make changes to an existing user 2. Does not save the changes 3. Changes should not get saved
Complexity	High
Priority	1

Use Case ID	UC-3
Title	Create projects
Description	Admin creates projects
Primary Actor	Admin user
Preconditions	Admin user has logged in to the system and navigated to the Projects sub-section
Post conditions	New projects should be created
Main Scenario	1. Admin user Adds a new Project 2. Fill all mandatory fields and Saves 3. Newly created project should be visible under the projects section
Alternative Scenario	1. Admin users ads a new Project 2. User does not fill mandatory input parameters and saves 3. Application displays message when saving the Project.
Complexity	High
Priority	1

Use Case ID	UC-4
Title	Manage projects
Description	Admin user manage projects
Primary Actor	Admin user
Preconditions	Admin user has already created projects in the system
Post conditions	Projects can be managed by the Admin user
Main Scenario	1. Admin user edits a project 2. User saves changes 3. Changes made to projects should get saved.
Alternative Scenario	1. User does not fill mandatory input parameters and save 2. Application displays message when saving the project
Complexity	Medium
Priority	1

Use Case ID	UC-5
Title	Assign users
Description	Admin user assign users to projects
Primary Actor	Admin User
Preconditions	Admin user has created users and projects in the system
Post conditions	Users should get assigned to one or more projects.
Main Scenario	1. Admin user login to the system and navigates to the Users section 2. Admin user fills Assign users to project by selecting all mandatory fields 3. Saves assign to user
Alternative Scenario	1. User does not fill mandatory input parameters and save 2. Application displays message when saving the project
Complexity	Medium
Priority	1

Use Case ID	UC-6
Title	Change password
Description	User changes the existing password after login to the system.
Primary Actor	Registered User
Preconditions	User should be login to the system
Post conditions	User password is being changed
Main Scenario	1. User navigates to the password reset page 2. User enters the existing password 3. User enters the new password 4. User enters the same password to confirm and save changes 5. System changes the
Alternative Scenario	1. User enters two different passwords for new and confirm password fields 2. Saves changes 3. System displays message to enter same password
Complexity	Medium
Priority	1

Use Case ID	UC-7
Title	Login
Description	Registered user login to the system providing correct credentials.
Primary Actor	Registered user
Preconditions	Admin user has created a user in the system
Post conditions	User is logged in to the system
Main Scenario	1. User navigates to the login page 2. User enters user name and password 3. User clicks on Sign in button 4. User is authenticated and user will be redirected to the dashboard
Alternative Scenario	1. User enters incorrect credentials 2. Application displays message claiming incorrect credentials and user stays in the login window.
Complexity	Medium
Priority	1

Use Case ID	UC-8
Title	View dashboard
Description	Registered user logging to the system and view the dashboard.
Primary Actor	Registered user
Preconditions	User has logged in to the system
Post conditions	User views the dashboard
Main Scenario	1. User navigates to the Dashboard section. 2. User views the dashboard
Alternative Scenario	1. User is not logged in to the system 2. User cannot access the dashboard
Complexity	Medium
Priority	2

Use Case ID	UC-9
Title	View tests
Description	Registered user views test cases
Primary Actor	Registered user
Preconditions	User has allocated to a project with tests
Post conditions	User views test cases
Main Scenario	1. User navigates to the test library 2. User clicks on test cases 3. User can views test cases for the selected project
Alternative Scenario	1. User is not assign to a project 2. User is not allow to view test cases
Complexity	Medium
Priority	1

Use Case ID	UC-10
Title	Add tests
Description	Registered user adds test cases
Primary Actor	Tester
Preconditions	User has allocated to a project with tests
Post conditions	User adds test cases
Main Scenario	1. User navigates to the test library 2. User navigates to the test cases section 3. User adds test cases
Alternative Scenario	1. User is not assign to a project 2. User is not allow to add test cases
Complexity	High
Priority	1

Use Case ID	UC-11
Title	Search tests
Description	User searches specific test
Primary Actor	Tester
Preconditions	Test cases available in test library
Post conditions	User search tests
Main Scenario	1. User navigates to the test cases section 2. User searches for test case 3. User gets search result(s) depending on the search phrase
Alternative Scenario	1. User is not assigned to a project 2. User receives no search results
Complexity	High
Priority	1

Use Case ID	UC-12
Title	Modify tests
Description	User edits test cases
Primary Actor	Tester
Preconditions	Test cases available in test library for the tester
Post conditions	Modifications made to the tests get save.
Main Scenario	1. Tester selects a test and edits 2. Tester made changes to the test case 3. Tester saves changes 4. Changes made to the test are saved.
Alternative Scenario	1. Tester selects a test and edits 2. Tester made changes to the test case 3. Tester returns without saves changes 4. Changes made to the test are not saved.
Complexity	High
Priority	1

Use Case ID	UC-13
Title	Delete tests
Description	User deletes test cases
Primary Actor	Tester
Preconditions	Test cases available in test library for the tester
Post conditions	Deleted test cases no longer available in the system
Main Scenario	1. Tester selects a test and deletes 2. Tester accepts the confirmation message 3. Deleted test case no longer available in the system
Alternative Scenario	1. Tester selects a test and deletes 2. Tester cancel the confirmation message 3. Test case is not deleted
Complexity	High
Priority	1

Use Case ID	UC-14
Title	Execute automated tests
Description	User execute an automated test using GUI
Primary Actor	Tester
Preconditions	Windows service runs in the host machine
Post conditions	Automated test get executed
Main Scenario	1. User select an automated tests 2. Runs the test 3. Automated test runs on the host machine
Alternative Scenario	1. Windows service is not running on the host machine 2. Automated test is not executed
Complexity	High
Priority	1

Use Case ID	UC-15
Title	Add daily status
Description	User adds daily status
Primary Actor	Tester
Preconditions	User is logged in to the system
Post conditions	Daily status get saved in the system
Main Scenario	1. User navigates to the daily status section 2. User fill daily statuses 3. User saves daily statuses 4. Daily status get saved in to the system
Alternative Scenario	1. User fill daily status 2. Returns without saving 3. Daily status is not saved
Complexity	High
Priority	1

Use Case ID	UC-16
Title	Modify daily status
Description	User modifies daily status
Primary Actor	Tester
Preconditions	Daily status is available in the system
Post conditions	Daily status get modified
Main Scenario	1. User navigates to the daily status section 2. User edits a daily status 3. User saves changes 4. Modified daily status get saved in to the system
Alternative Scenario	1. User modifies daily status 2. Returns without saving the changes 3. Daily status is not modified
Complexity	High
Priority	1

Appendix B: Detailed Test Results

User Login Related Test Cases

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Existing user login to the system	1. Navigate to the login page 2. Enter correct user name to the username field 3. Enter correct password to the password field 4. Click sign-in button	User should be login to the system and should display the dashboard.	Pass
TC002	Non-existing user login to the system	1. Navigate to the login page 2. Enter invalid credentials 3. Click sign-in button	User should not be able to login to the system	Pass
TC003	Disabled user login to the system	1. Login to the system as Admin 2. Navigate to the Users section 3. Select a user and click edit button 4. Un-tick the Active check box 5. Save changes 6. Now logout from the system 7. Try to login as the disabled user	Disabled user should not be able to log in to the system. Message should be prompted stating that user is inactive.	Pass
TC004	Deleted user login to the system	1. Login to the system as Admin 2. Navigate to the Users section 3. Select a user and click on delete button 4. Accept the confirmation dialog 5. Logout from the application 6. Now on the login page, enter credentials for the deleted user and click Sign-in button	Deleted user should not be able to login to the system	Pass

User Management Test Cases

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Adding a new user	1. Login to the system as the admin user 2. Navigate to the Users section 3. Click Add button 4. Fill user details 5. Save details	User details should be displayed in the User grid section. Newly added user should be able to login to the system providing the default password	Pass
TC002	Deactivating an existing user	1. Login to the system as admin user 2. Navigate to the User section 3. Select an existing user and un-tick the Active checkbox 4. Save changes	Disabled user cannot login to the system and message should be prompted stating user is disabled.	Pass
TC003	Deleting an existing user	1. Login to the system as admin user 2. Navigate to the User section 3. Select an existing user and click on Delete button 4. Accept the confirmation message	User should get deleted from the system and should be remove from the users grid	Pass
TC004	Search and Existing user	1. Login to the system as admin user 2. Navigate to the User section 3. Click on Search button 4. Now in the search text box, search a phrase and click "Go" button.	Filtered results should be displayed in the grid area.	Pass

Project management Test Cases

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Add Projects	1. On Projects page 2. Click Add button 3. Fill project name and description 4. Save project	Project should get saved and listed under projects grid.	Pass
TC002	Edit Projects	1. On Projects page 2. Select a project and click edit button 3. Make changes to the project details 4. Save project	Changes made to the project section should get saved and displayed on the project grid area.	Pass
TC003	Delete Projects	1. On projects page 2. Select a project and click delete button 3. Accept the confirmation alert	Project should no longer displayed under the project grid area	Pass
TC004	Assign users to Project	1. Click Add button under Assign user to project section 2. Select user ID and Project ID 3. Click Save button	User should get assign to the project and should get displayed in the grid area	Pass

Hosts Test Cases

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Add new hosts	1. On Hosts section 2. Click on Add new hosts button 3. Fill mandatory fields 4. Save hosts	Newly added host should get saved under the host grid	Pass
TC002	Edit hosts	1. On hosts section 2. Select a host and click on Edit button 3. Make changes and Save	Changes should get saved	Pass
TC003	Delete hosts	1. On hosts section 2. Select a host and click on delete button 3. Accept the confirmation alert	Host record should get removed from the hosts section	Pass

Test Cases for Test Suites

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Add test suites	1. On Test suites page 2. Click on Add button 3. Fill test suites section 4. Click Save button	New test suite should get added to the Test suite section	Pass
TC002	Edit test suite	1. On test suite page 2. Click on Edit button 3. Make changes to the test suite 4. Click Save button	Changes made to the test suite should get saved and reflected on the Test suite section	Pass
TC003	Delete test suite	1. On test suite page 2. Click on Delete button 3. Accept the confirmation alert	Test suite should get deleted	Pass

Test Cases for test case section

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Add test cases	1. On test case page 2. Click on Add button 3. Fill test cases section 4. Save test case	Newly added test case should be listed under test case grid	Pass
TC002	Edit test case	1. On test case page 2. Click on Edit button 3. Do some changes to the test case 4. Save changes	Changes made to the test case should get reflected on the grid area	Pass
TC003	Delete test case	1. On test case page 2. Click on delete button 3. Accept the confirmation alert	Test case should get deleted from the grid area	Pass
TC004	Add test case to a test suite	1. On test case page 2. Click on edit test case 3. Select test suite from the test suite dropdown 4. Save changes	Changes should get saved and test case should be now under the selected test suite	Pass
TC004	Execute test case	1. On test case page 2. Select a test case and click on edit button 3. Navigate to Execute section 4. Change the status 5. Click on Save button	Test case status should get changed and grid area should display corresponding color against the test case status	Pass

Test Cases for Daily status

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Add daily status	1. On daily status page 2. Click on Add button 3. Fill daily status 4. Save daily status	Daily status should get saved	Pass
TC002	Edit daily status	1. On daily status page 2. Click on Edit button for a daily status 3. Make some changes and Save	Changes should get saved	Pass
TC003	Delete daily status	1. On daily status page 2. Click on delete button for a daily status 3. Accept the confirmation alert	Selected test case should get deleted from the test case section	Pass

Test Cases for Dashboard

Test Case Id	Test Description	Steps	Expected Result	Status
TC001	Test Execution Stats	1. On Dashboard page 2. Validate Test execution stats display correct values	Test case count should be correct and status count also should tally with the Test case section count	Pass
TC002	Test Execution Overview	1. On Dashboard page 2. Validate test execution overview colors and count	Pie chart should display the correct values as per the test execution stats section	Pass