TEST ENGINEER: A MANAGEMENT & PLANNING SYSTEM

Deepa Nandi
B.E, Vishveshwariah Technological University, India, 2005

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2011
Student: Deepa Nandi

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator

Dr. Nikrouz Faroughi          Date

Department of Computer Science
Abstract

of

TEST ENGINEER: MANAGEMENT AND PLANNING SYSTEM

by

Deepa Nandi

With the increase in sophistication in Software Development, producing reliable and robust software has become one of the most important software concerns. Software testing is the key to measure reliability and robustness of the software and executing tests in a systematic way would help measure these qualities accurately.

Test Engineer provides an integrated environment in order to assist in the software testing effort. This project will focus on the enhancement to Test Engineer that was originally developed by Kevin Kho at CSUS as a part of his Master’s project. The goal is to enhance the software to be user friendly and robust thereby providing features that can help plan and maintain test execution systematically.

_____________________, Committee Chair
Dr. Ahmed Salem, Ph.D.

_____________________
Date
DEDICATION

I dedicate this Master’s project to my parents, Nagarathna Nandi and Lingaraj Nandi, and my husband, Charan Shivaprasad. Without their encouragement, understanding and love, none of what I’ve accomplished in my life would be possible.

I would also like to thank Professor Ahmed Salem, my project advisor for his constant support and guidance throughout the course of this project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedication</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td><strong>Chapter</strong></td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Project Background</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Goal of the Project</td>
<td>4</td>
</tr>
<tr>
<td>2. REQUIREMENT ANALYSIS</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Brief Description of System’s Main Features</td>
<td>6</td>
</tr>
<tr>
<td>2.2 User Characteristics</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Product Perspective</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Product Features</td>
<td>10</td>
</tr>
<tr>
<td>2.5 System Interactions</td>
<td>12</td>
</tr>
<tr>
<td>2.6 Functional Structure</td>
<td>18</td>
</tr>
<tr>
<td>3. SYSTEM DESIGN</td>
<td>20</td>
</tr>
<tr>
<td>3.1 Architectural Design</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Database Design</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Technical Design</td>
<td>31</td>
</tr>
<tr>
<td>4. USER INTERFACE DESIGN</td>
<td>38</td>
</tr>
<tr>
<td>4.1 Interface Page Flow</td>
<td>38</td>
</tr>
<tr>
<td>4.2 Configuration Requirements</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Interface Design Rules</td>
<td>39</td>
</tr>
<tr>
<td>4.4 Restrictions and Limitations</td>
<td>40</td>
</tr>
</tbody>
</table>
5. SYSTEM IMPLEMENTATION ................................................................. 41
   5.1 Technology Choices ................................................................. 41
   5.2 Methodology ............................................................................. 44
   5.3 Functions and Includes .......................................................... 48
   5.4 Application Modules and Components .................................... 50
   5.5 Table Declarations ............................................................... 58
5. APPLICATION SCREENSHOT ............................................................. 64
   6.1 Login Screen ............................................................................. 65
   6.2 Menu Bar .................................................................................. 66
   6.3 Task List Screen ....................................................................... 67
   6.4 Project Menu Screen .............................................................. 68
   6.5 Project Overview ................................................................. 69
   6.6 Feature Overview ................................................................. 70
   6.7 Test Case Overview .............................................................. 72
   6.8 Step Overview Page ............................................................... 74
   6.9 View History Screen .............................................................. 76
   6.10 Generate Charts ................................................................. 77
7. CONCLUSION AND FUTURE WORK ................................................... 79
  Appendix ......................................................................................... 80
  Glossary of Terms ........................................................................... 80
  Bibliography ...................................................................................... 82
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Table 1.1: Comparison of the available features in various tools.</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Table 3.1: Users Table Schema</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>Table 3.2: Employee Table Schema</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>Table 3.3: Project Table Schema</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>Table 3.4: Feature Table schema</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>Table 3.5: Testcase Table Schema</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>Table 3.6: Case_Notes Table Schema</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>Table 3.7: Result_State Table Schema</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>Table 3.8: Up_Files Table Schema</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>Table 3.9: Case_History Table Schema</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>Table 3.10: Feature_History Table schema</td>
<td>28</td>
</tr>
<tr>
<td>12</td>
<td>Table 3.11: Project_History Table schema</td>
<td>29</td>
</tr>
<tr>
<td>13</td>
<td>Table 4.1: The restrictions and limitations for users of the system.</td>
<td>40</td>
</tr>
<tr>
<td>14</td>
<td>Table 5.1: List of Functions and Includes</td>
<td>40</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Figure 2.1 Hierarchical structure from test project to test step</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Figure 2.2 UML Diagram illustrating system’s main features</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Figure 2.3 Existing core features of the Test Engineer tool</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Figure 2.4 Test case result update</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>Figure 2.5 Enhancement feature to view the test case by state</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Figure 2.6 Email notification feature</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Figure 2.7 Chart generation feature</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>Figure 3.1 System Architecture</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Figure 3.2 Entity Relationship diagram</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>Figure 3.3 Sequence Diagram- Permission feature to access project details</td>
<td>32</td>
</tr>
<tr>
<td>11</td>
<td>Figure 3.4 Sequence Diagram - Case history creation and display</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>Figure 3.5 Sequence Diagram - Email notification during project deletion</td>
<td>34</td>
</tr>
<tr>
<td>13</td>
<td>Figure 3.6 Sequence Diagram - Completion rate of a project</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>Figure 3.7 Sequence diagram - Update to a test case resulting in the trigger</td>
<td>37</td>
</tr>
<tr>
<td>15</td>
<td>Figure 4.1 Interface Page Flow</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>Figure 6.1 Login Screen</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>Figure 6.2 Differences between the menu option for admin and other user</td>
<td>66</td>
</tr>
<tr>
<td>18</td>
<td>Figure 6.3 Task List for Admin</td>
<td>67</td>
</tr>
<tr>
<td>19</td>
<td>Figure 6.4 Project Menu</td>
<td>68</td>
</tr>
<tr>
<td>20</td>
<td>Figure 6.5 Project Overview</td>
<td>69</td>
</tr>
</tbody>
</table>
21. Figure 6.6 Feature overview ................................................................. 70

22. Figure 6.7 Test case overview ............................................................. 72

23. Figure 6.8 Step Overview page ......................................................... 74

24. Figure 6.9 Page displays a delete button if a screenshot is already uploaded. ..... 75

25. Figure 6.10 View History log for test case Invalid Login ...................... 76

26. Figure 6.11 Generate Chart ................................................................. 77

27. Figure 6.12 Display Estimate versus Actual Hours of test feature .......... 78
Chapter 1

INTRODUCTION

Software testing is a vital process in the software development life cycle that is aimed to evaluate an attribute or capability of a system and to determine that it meets the requirements criteria. With the increasing sophistication in the technology, the complexity of the applications has equally increased. This in turn leads to complexity in testing, thus leading to a huge number of test cases. Managing resources and scheduling plans for test execution becomes crucial.

The conventional way of planning and management of testing was with the help of Microsoft Excel Spreadsheets. This was an unproductive way of managing resources since there was no integrity in the data. Testers provided status updates for the week and functional leads maintained the spreadsheets and updated based on the weekly statuses. There was no regulatory compliance which could account for test execution. Let alone the integrity and compliance issues, managing hundreds and thousands of test cases and their descriptions and assigning and distributing it to the team on an individual basis was a nightmare. This often led to increased cost and resources for management and planning.

Test Engineer – Planning and Management is an open source tool which provides solution to plan on resource allocation and manage testing with a systematic approach. Test case information is stored in a centralized database and the information can be accessed based on the permission levels of the user.
1.1 Project Background:

“The Test Engineer – Planning and Management” [1] tool was first implemented by Kevin Kho as a part of his Master’s project in California State University Sacramento in Fall 2009. The tool provides the basic features that can help a user manage multiple projects, features, and their test cases. The tool also provides the option to generate a report where in the report gives an overview of the project and related features and test cases.

When compared with the tools available in the market like Seapine [13], Testmanagement tool [14], Quality Center(HP) [5] and Rational Software Testing tool (IBM) [6], Test Engineer tool lacked some of the essential features listed below.

Table 1.1: Comparison of the available features in various tools.

<table>
<thead>
<tr>
<th>Features</th>
<th>Test Engineer</th>
<th>Seapine</th>
<th>Quality Center</th>
<th>Rational Software Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plan Creation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Case Specification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User Assignments</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Role-based permissions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Schedule Timeline</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Status/Results Reports</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Graphical Summaries</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Screenshot Attachments</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Email Notifications</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Usability Features</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enforce Regulatory Compliance</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Tools like Seapine also provide video tutorials [4] which help in identifying the
differences between the tools. Identifying these differences led to the idea of upgrading
Test Engineer to match with industry standard features for the tool.
1.2 Goal of the Project:

The goal of the project is to enhance the application with additional features that can improve the usability of the tool and make the tool robust. As a part of enhancement to the existing tool, the following features are added:

1. Test Step: Currently the tool provides the ability to add features and test cases to a project. However, the tool does not provide the ability to add steps to the test case. Currently, the test steps can be entered in the textbox as a series of steps. But this will not validate that each step of the test case is tested and followed the order in which the steps need to be executed. With the addition of test steps per test case, the user will be able ensure that all the steps in the test case are executed and are executed at the same order for consistent testing.

2. History Log: The most important feature of a management tool is to have the ability to log history of the events that occur. Historical information helps the user to better estimate future testing efforts, costs and the delivery date. It also helps in tracking any suspicious activities that has occurred and will help restore the data in the order of the events that were executed.

3. Charts to display various metrics that are needed for project management. These include

4. Completion rate per test feature, test case or test project.

5. Overview of the estimated hours versus the actual hours of the project per feature.

6. A trend of the project progress.
7. Multiple user login with different security permissions based on groups. This is an essential feature since the tool should not allow access to unauthorized users. The tool should provide the functionality to add user, add group and set permissions to the group. The users should be restricted to view, create, delete or update based on their permission level and this should be enforced per test project level.

8. Notes: It is extremely important that software is very well documented so that the users can kick start with the software with ease. This can be just written up as notes for the entire software or help pages can be added on each page for the users to better understand all features that are available on that particular page.

9. Email Notifications: The tool should provide the functionality of automatically notifying the user when a task/test case is assigned to him/her, when a user deletes a project or a feature, when the test case or feature has completed testing. The feature can also include adding additional recipients for notification when creating test cases or test features. This feature would help the user to plan based on the tasks assigned to him/her.

10. Screenshot attachments: Attaching screenshots upon validating each test step would be a great way to validate each and every step of the test case. This would insure that every step is tested and behaves the way it is expected to work.
Chapter 2

REQUIREMENT ANALYSIS

This chapter describes the perspective of the product, the various features of the product, its usage, hardware requirements, assumptions and dependencies. Further the chapter also gives an in depth analysis of the functional and non-functional requirements with use cases.

2.1 Brief Description of System’s Main Features:

Test Engineer is a web based tool used to plan the testing phase of the software development life cycle. It provides the users the ability to manage projects and its test cases. Various test projects can be created, updated or can also be deleted and a history log is maintained to account for these actions. Each project branches down to several functional features and each functional feature further branches down to multiple test cases. These test cases can be positive test cases or negative. The test case consists of a list of steps that needs to be executed in an orderly fashion for the successful completion of the test case. Each step of the test case is validated and screenshots are attached based on the actual results of the project.
Test Engineer provides various roles like administrator, functional lead and tester. The tool is structured for the users to navigate based on their usage and the role of the user. The users can view all the tasks assigned to them as well as search tasks assigned to other users. The tool also provides functionality to assign tasks to users. Emails are sent to users upon the assignment of task to each user, upon any updates or deletes to notify the appropriate user about the events.

The tool also accompanies a software documentation that guides the users on the installation and usage of the tool. No personal training is required for installation or for the usage of the software.

Figure 2.1 Hierarchical structure from test project to test step.
2.2 User Characteristics:

Test Engineer is designed to support three different roles:

1. Administrator
2. Functional Lead
3. Tester

Role of Administrator:

Administrator is the super user of the system and has access to all the features of the system. Admin can create, update or delete user accounts, test projects, test features, test cases and/or test steps. History is logged in order to account for the action of each user and admin is also accountable for his actions.

Role of Functional Lead:

Functional Lead is the user responsible for a test project. The role of a lead begins from the project level and his permissions are restricted to the project he is working on. A functional lead can update a project he is assigned to and can create, update or delete on any features below the project in the hierarchy. Notifications are sent upon the update or deletion of any features, test cases or test steps through email. Chart generation, report generation, scheduling and task assignment are some additional features provided to this user.
Role of Tester:

The tester’s visibility is restricted to the test case he is assigned to. He can update certain fields of the test case and create, update or delete any test steps. He can access the information of the project, its test features and its test cases in the view mode only. The responsibility of the tester mainly focuses on the test steps where in the tester validates each step of the test case, attaches a screenshot and updates the status to show the execution progress. Notifications are sent upon a task assignment or any update in the test case or test step information.

2.3 Product Perspective:

The product is built upon a web server utilizing PHP for server-side scripting and MySQL for database access. The web server will be able to handle multiple connections to ensure all users are able to access the system. The product will be considered a standalone application; while the product can be built upon, the base product will not be integrated with another system.
2.4 Product Features:

The product is designed with a variety of features that match the industrial standards. Test Engineer was developed with some basic features which included:

As a part of enhancement to Test Engineer, the following functional requirements were added to improve on the usability and to make it robust.

Enhancement Functional Requirements:

1. Display Task Assignments based on login user.
2. Create/Update/Delete test steps.
3. Attachment of screenshots per test step.
4. Display test cases by status.
5. Send email notification on task assignment.
6. Send email notification on deletion of a project or feature.
7. Send email notification on completion of the test case.
8. View History log on Update/ deletion of project.
9. View History log on Update/ deletion of feature.
10. View History log on Update/ deletion of test case.
11. View History log on creation/ Update/ deletion of user accounts.
12. Report generation by project.
14. Chart generation to display completion rate per project.
15. Chart generation to display completion rate per feature.
16. Trend chart for the project progress.

17. Chart generation of actual versus expected results.

18. Delete test cases.

19. Delete test features.

20. Delete test project.

21. Role based usage of the system.

Enhancement Non Functional requirements:

1. Application support for compliance through history management.

2. Application security through role based usage of the system.

3. Update on user interface for enhanced usability.

4. System support for multiple database and server configuration.
2.5 System Interactions:

This section will provide an overview of the user interactions with the Test Engineer System. This is illustrated with the help of UML use case diagrams and focusing mainly on the features that are added as an enhancement to the existing project.

![Figure 2.2 UML Diagram illustrating system’s main features](image-url)
The Figure 2.2 gives the overview of the features of the system and the user’s interaction with the system.

With a wide range of features, the scope of the system is quite high leading it into several components as displayed below in the Figure 2.3 [1]. These components represent the existing core features of the Test Engineer tool.

Figure 2.3 Existing core features of the Test Engineer tool
Figure 2.4 Test case result update.

The enhancement features are centered around the existing components of the system and certain components where replaced to enhance the usability of the system. The approach developed for storing the results of the test cases is completely modified. The result of the test case is mainly centered upon the successful completion of all the steps of the test case in the order of execution.
The modifications to the result component lead its way to many new features for the system and reduced the complexity of storing the results of the test cases. The results of the test cases are now stored as a set of defined status Open, In Progress, Passed, Failed and Inconclusive. This resulted in the emergence of a new functional feature to view test cases by its status.

![Figure 2.5 Enhancement feature to view the test case by state](image)

This new feature is embedded as a part of the test suite/test feature overview, which displays the information of the test feature and all the test cases associated to it. With the addition of this new feature, the results of all the test cases associated to the test feature are now displayed and are further linked to display test cases by their status only.
The email notification feature is added as a common component amongst several components of the system. Users are notified during events like project/feature/case creation, deletion or modification. Notifications are also sent upon the successful completion of the test cases.

Figure 2.6 Email notification feature
The system extends its functionality for generating charts for the comparison of expected versus actual hours taken per feature/test case, for the progress of the project, for the completion rate of the project and features.

Figure 2.7 Chart generation feature
2.6 Functional Structure:

The functional structure is the structured representation of all the functions within the modeled system. The purposes of the functional structure are to describe the functions and processes, assist with discovery of information needs and establish a basis for determination of the project scope. The Figure 2.6 illustrates the functional structure of the existing system.
Figure 2.8 Functional Structure of the system
Chapter 3

SYSTEM DESIGN

This chapter aims at creating a technical solution that satisfies the functional requirements of the system. This includes preparation of the environment needed to build the system, preparation of the data and constructing components of the system that translate the functional requirements.

3.1 Architectural Design:

Test Engineer is a web based tool that comprises of the following 4 major components - client workstation, webserver, email exchange server and a database.

![System Architecture Diagram]

Figure 3.1 System Architecture
Description of Components:

Client Workstation –

Users interact with the internet enabled workstation through javascript and flash enabled browsers. The interaction between the client workstation and the user is a two communication. Example: Internet explorer 7,8,9, Mozilla FireFox 7.0, 8.0.

Database –

Database stores all the data that needs to be persisted. It receives information from the webserver to create, update or delete any information. The interaction between the database and the web server is a two way communication.

Example MySql, DB2, Oracle, etc

Web Server –

Web server hosts the application and stores all the dynamic php and html pages. The web server interacts with the client workstation and gets the input from the user and updates any necessary data changes that need to be persisted.

Example Apache, Tomcat, Websphere

Email Exchange server –

“Email exchange server”[10] is used to send automatic emails to the users. The webserver interacts with the email exchange server and based on the information received from the webserver, the email exchange server sends automatic emails to clients. The communication is always one way with email server.

Example: Sendmail, Microsoft Exchange Server, Exim, etc.
3.2 Database Design:

The following sections will provide users with explicit descriptions of the database schema and its table relationships. For details on database declarations, please refer to the “System Implementation” chapter.

Table Descriptions and Relationships

This section will provide readers with brief descriptions of the tables in the Test Engineer’s database and their relationships with other tables.

USERS: Functional Description

The Users table is used to store user account information. A unique User ID will be used as the primary key of the table.

EMPLOYEE: Functional Description

The Employee table holds user’s information like name, e-mail address and phone number. Additional fields may be added as they are needed. This table will use the Users’ User ID as its primary/foreign key.

PROJECT: Functional Description

The Project table stores all the information about the project which includes project name, version, developer, source path, execution path, username and the created date. This table will use the project id as its primary key.
FEATURE: Functional Description
The Feature table will store high-level details about the test features (name, version, input, output, description). Records in this table will hold a many-to-one relationship with its corresponding records in the project table. There may be many feature records that correspond to a record in a project, but a feature record may only correspond to one project.

TESTCASE: Functional Description
The test case table will store high-level details about all the test cases that are created. (name, revision, completion rate, state id). Records in this table will hold a many-to-one relationship with its corresponding records in the Feature table. There may be many test case records that correspond to a particular Feature, but a test case record may only correspond to one feature.

TESTSTEP: Functional Description
The test step table will store high-level details about all the test steps that are created. Records in this table will hold a many-to-one relationship with its corresponding records in the test case table. There may be many test step records that correspond to a particular case, but a test step record may only correspond to one test case.

CASE_NOTES: Functional Description
This table will store the forum discussions about each test case thread. Records in this table will hold the information about all user posted comments. The “testcase” table’s case_id will also be used as a foreign and primary key.
RESULT_STATE: Functional Description

This table will be used to constants that refer to all valid/invalid testing states (passed, failed, pending).

UP_FILES: Functional Description

This table will store each of the binary files that have been uploaded for a specific test step. Along with the uploaded BLOB file, the table will store the file details (name, size, description), upload information, and the foreign key of the feature, case and the test step that it was uploaded from.

PROJECT_HISTORY: Functional Description

The project history table will store all the details about the projects that are created, deleted or updated. Records in this table will hold a many-to-one relationship with its corresponding records in the Project table. There may be many project history records that correspond to a particular project, but a project history record may only correspond to one project in the database.

FEATURE_HISTORY: Functional Description

The feature history table will store all the details about the features that are created, deleted or updated. The table stores information of the user, action and the time of action. Records in this table will hold a many-to-one relationship with its corresponding records in the Feature table. There may be many feature history records that correspond to a particular feature, but a feature history record may only correspond to one project in the database.
CASE HISTORY: Functional Description

The case history table will store all the details about the test cases that are created, deleted or updated. The table stores information of the user, action and the time of action. Records in this table will hold a many-to-one relationship with its corresponding records in the case table. There may be many case history records that correspond to a particular case, but a case history record may only correspond to one case in the database.
Database Schema

The following tables represent the database schemas for each table included in the Test Engineer’s MySQL database. An entity-relationship diagram (ERD) can be found in Figure 3.2. For more details on how to declare these tables through a command line interface, please refer to the Table Declarations section in Chapter 4.

Table 3.1: USERS Table Schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uid</td>
<td>int</td>
<td>PK: unique id for the user</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>User's login name</td>
</tr>
<tr>
<td>password</td>
<td>varchar</td>
<td>User's password</td>
</tr>
</tbody>
</table>

Table 3.2: EMPLOYEE Table Schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uid</td>
<td>int</td>
<td>PK,FK: unique id for the user</td>
</tr>
<tr>
<td>Fname</td>
<td>varchar</td>
<td>User's first name</td>
</tr>
<tr>
<td>Lname</td>
<td>varchar</td>
<td>User's last name</td>
</tr>
<tr>
<td>Phone</td>
<td>varchar</td>
<td>User's phone number</td>
</tr>
<tr>
<td>Email</td>
<td>varchar</td>
<td>User's e-mail address</td>
</tr>
</tbody>
</table>

Table 3.3: PROJECT Table Schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_id</td>
<td>int</td>
<td>PK: unique id for the project</td>
</tr>
<tr>
<td>project_name</td>
<td>varchar</td>
<td>Project's name</td>
</tr>
<tr>
<td>version</td>
<td>varchar</td>
<td>Project's version number</td>
</tr>
<tr>
<td>developer</td>
<td>text</td>
<td>Project's developer</td>
</tr>
<tr>
<td>src_path</td>
<td>text</td>
<td>Path to project source code</td>
</tr>
<tr>
<td>exe_path</td>
<td>text</td>
<td>Path to project executable</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>Username of the last user to update the record</td>
</tr>
<tr>
<td>create_date</td>
<td>datetime</td>
<td>Date/time of the last update to the record</td>
</tr>
</tbody>
</table>
Table 3.4: FEATURE Table schema

<table>
<thead>
<tr>
<th>feature</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feat_id</td>
<td>int</td>
<td>PK: unique id for the feature test suite</td>
</tr>
<tr>
<td>project_id</td>
<td>int</td>
<td>FK: unique id for the project that the test suite was created</td>
</tr>
<tr>
<td>feat_name</td>
<td>varchar</td>
<td>Test suite name</td>
</tr>
<tr>
<td>input</td>
<td>text</td>
<td>Type of input that is expected for this suite</td>
</tr>
<tr>
<td>output</td>
<td>text</td>
<td>Type of output that is expected for this suite</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Details about the test suite</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>Username of the last user to update the record</td>
</tr>
<tr>
<td>create_date</td>
<td>datetime</td>
<td>Date/time of the last update to the record</td>
</tr>
</tbody>
</table>

Table 3.5: TESTCASE Table Schema

<table>
<thead>
<tr>
<th>feature</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>case_id</td>
<td>int</td>
<td>PK: unique id for the test case</td>
</tr>
<tr>
<td>feat_id</td>
<td>int</td>
<td>FK: unique id for the suite that the test case was created</td>
</tr>
<tr>
<td>case_name</td>
<td>varchar</td>
<td>Test case title</td>
</tr>
<tr>
<td>revision</td>
<td>tinytext</td>
<td>Test case revision number</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Brief description of the test case</td>
</tr>
<tr>
<td>process</td>
<td>text</td>
<td>Process required to run the test case</td>
</tr>
<tr>
<td>setup</td>
<td>text</td>
<td>Setup instructions for the test case</td>
</tr>
<tr>
<td>exp_results</td>
<td>text</td>
<td>Expected results of the test case</td>
</tr>
<tr>
<td>pass</td>
<td>text</td>
<td>Criteria for meeting passing requirements</td>
</tr>
<tr>
<td>fail</td>
<td>text</td>
<td>Criteria for classifying the test case results as a failure</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>Username of the last user to update the record</td>
</tr>
<tr>
<td>create_date</td>
<td>datetime</td>
<td>Date/time of the last update to the record</td>
</tr>
</tbody>
</table>

Table 3.6: CASE_NOTES Table Schema

<table>
<thead>
<tr>
<th>feature</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>case_id</td>
<td>int</td>
<td>FK: unique id for the test case these commented on</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>Username of the user who posted the comments</td>
</tr>
<tr>
<td>last_updated</td>
<td>datetime</td>
<td>Date/time that the comments were posted</td>
</tr>
<tr>
<td>user_notes</td>
<td>text</td>
<td>Text of the user comments</td>
</tr>
<tr>
<td>deleted</td>
<td>varchar</td>
<td>Flag used to track whether the comments are visible to others</td>
</tr>
</tbody>
</table>

Table 3.7: RESULT_STATE Table Schema

<table>
<thead>
<tr>
<th>feature</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state_id</td>
<td>int</td>
<td>PK: unique id corresponding to the state</td>
</tr>
<tr>
<td>status</td>
<td>tinytext</td>
<td>Description of the result state</td>
</tr>
<tr>
<td>active</td>
<td>varchar</td>
<td>Flag used to show whether the state is valid/invalid</td>
</tr>
</tbody>
</table>
Table 3.8: UP_FILES Table Schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_id</td>
<td>int</td>
<td>PK: unique corresponding to the uploaded file</td>
</tr>
<tr>
<td>feat_id</td>
<td>int</td>
<td>FK: unique id for the feature that the file corresponds to</td>
</tr>
<tr>
<td>case_id</td>
<td>int</td>
<td>FK: unique id for the test case that the file corresponds to</td>
</tr>
<tr>
<td>username</td>
<td>varchar</td>
<td>Username of the user who uploaded the file</td>
</tr>
<tr>
<td>up_date</td>
<td>datetime</td>
<td>Date/time that the file was uploaded</td>
</tr>
<tr>
<td>file_blob</td>
<td>longblob</td>
<td>Binary data for the uploaded file</td>
</tr>
<tr>
<td>file_name</td>
<td>text</td>
<td>Name of the file that was uploaded</td>
</tr>
<tr>
<td>file_size</td>
<td>int</td>
<td>Size of the file that was uploaded (in bytes)</td>
</tr>
<tr>
<td>file_notes</td>
<td>text</td>
<td>Notes about the uploaded file</td>
</tr>
<tr>
<td>step_id</td>
<td>int</td>
<td>FK: unique id for the step that the file corresponds to</td>
</tr>
</tbody>
</table>

Table 3.9: CASE_HISTORY Table Schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>case_hist_id</td>
<td>int</td>
<td>PK: unique id for the case history</td>
</tr>
<tr>
<td>case_id</td>
<td>int</td>
<td>FK: unique id for the case that the history was created</td>
</tr>
<tr>
<td>case_name</td>
<td>varchar</td>
<td>Test case title</td>
</tr>
<tr>
<td>Revision</td>
<td>tinytext</td>
<td>Test case revision number</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Brief description of the test case</td>
</tr>
<tr>
<td>state_id</td>
<td>int</td>
<td>FK: unique id storing the type of result state</td>
</tr>
<tr>
<td>completion_rate</td>
<td>int</td>
<td>Percentage completion of test case</td>
</tr>
<tr>
<td>updated_by</td>
<td>text</td>
<td>row update user of test case that the history associates to</td>
</tr>
<tr>
<td>Type</td>
<td>text</td>
<td>type of history – update, create or delete.</td>
</tr>
<tr>
<td>update_ts</td>
<td>datetime</td>
<td>date and time when the row was updated.</td>
</tr>
</tbody>
</table>

Table 3.10: FEATURE_HISTORY Table schema

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feat_hist_id</td>
<td>int</td>
<td>PK: unique id for the feature history</td>
</tr>
<tr>
<td>feat_id</td>
<td>int</td>
<td>FK: unique id for the feature that the history was created</td>
</tr>
<tr>
<td>feat_name</td>
<td>varchar</td>
<td>feature title</td>
</tr>
<tr>
<td>Version</td>
<td>tinytext</td>
<td>Test case version number</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Brief description of the test case</td>
</tr>
<tr>
<td>est_hours</td>
<td>int</td>
<td>the estimated number of hours</td>
</tr>
<tr>
<td>act_hours</td>
<td>int</td>
<td>The actual number of hours</td>
</tr>
<tr>
<td>updated_by</td>
<td>text</td>
<td>row update user of test case that the history associates to</td>
</tr>
<tr>
<td>Type</td>
<td>text</td>
<td>type of history – update, create or delete.</td>
</tr>
<tr>
<td>update_ts</td>
<td>datetime</td>
<td>date and time when the row was updated.</td>
</tr>
</tbody>
</table>
Table 3.11: PROJECT_HISTORY Table schema

| proj_hist_id  | int   | PK: unique id for the project history |
| proj_id       | int   | FK: unique id for the project that the history was created |
| proj_name     | varchar | proj title |
| Version       | tinytext | Test case version number |
| description   | text   | Brief description of the test case |
| est_hours     | int    | the estimated number of hours |
| act_hours     | int    | The actual number of hours |
| updated_by    | text   | row update user of test case that the history associates to |
| Type          | text   | type of history – update, create or delete. |
| update_ts     | datetime | date and time when the row was updated. |
In order to maintain consistency and avoid duplication of data, the database tables need to be normalized. This could be accomplished by establishing relationships between the tables by defining the primary and secondary keys to the table. Such relationships are illustrated in Figure 3.2.

Figure 3.2 Entity Relationship diagram
3.3 Technical Design:

When using the system, users will be lead through interactive user interface pages that helps them step through the testing process. But while users are presented with a fairly simple process, the same can’t be said about what happens behind the scenes. In this section, we illustrate the interactions that occur behind the scenes and go in depth to understand the technical design of the features that were added as a part of enhancement to the existing project.

The following are discussed in the subsequent pages:

1. User Permissions
2. View History
3. Email Notification
4. Chart Generation
5. Trigger Completion Rate Calculation.

User Permissions:

There are three types of users for the system – Administrator, Functional lead and Tester. Each user has permissions to specific set of permissions. A Functional Lead can have permission to create, update and delete any information related to the project he is assigned to. Similarly, the tester is restricted to create, update or delete at the test case level only.
The Sequence diagram in the Figure 3.3 illustrates the user permissions feature of the system. In this scenario, the tester logs into the application with a valid username and a password and the system validates his identity and his permission level and logs him into the system. Upon successful login, the tester searches for a project and the system checks if the tester is assigned to any of the test cases associated to this project. If yes, the system navigates to the project details page and if no, it navigates to access denied page.

View History:

The Test Engineer tool was upgraded to be accountable for any data changes in the system. The history feature provides information about all the activities in the system which includes creation, deletion or modification of project, feature, test case or accounts.
Figure 3.4 Sequence diagram - Case history creation and display.

The Figure 3.4 illustrates the View history feature of the application. In this scenario, the tester logs into the application and is defaulted to the task assignment page. The tester selects a test case to work on. The system loads the test case selected, loads the test feature information of the test case and also loads the test project information of the test case. The page is then navigated to the test case overview page where the tester can edit or update the test case. Upon change in the test case information, the system updates the data in the database and logs the history of change. The change is recorded along with the username and the date timestamp of change. The system returns nothing to the tester.

When the tester clicks on the view history button, he is navigated to the test case history page where all the history information related to the test case is displayed.
Email Notifications:

The system is enhanced to notify the users through email during certain events like task assignment, deletion of a project, deletion of a test feature or a test case. This feature helps the administrator and the functional leads to track any changes in the system.

Figure 3.5 Sequence diagram - Email notification during project deletion.

The Figure 3.5 illustrates the email notification feature of the system. In this scenario, the user is deleting a project which is selected. The system gets all the test features, test cases and test steps associated to the project and delete them at various levels. The test steps associated to the test cases are deleted first followed by test cases. Upon successful deletion of test cases an email notification is sent to all the users who have tasks assigned.
with these test cases. Similarly test features are deleted and users are notified. Finally the project is deleted and here again the user is notified about the deletion of the project.

It is highly recommended not to delete a project with multiple features, test cases and test steps in place.

Charts Generation:

Charts provide a graphical representation of data. The Test Engineer tool provides the options to generate graphs for evaluating the completion rate of the project, completion rate of the feature, estimated time versus actual time of the project and estimated time versus actual time of the test feature.
The Figure 3.6 illustrates the chart generation feature of the system. The user selects the option to generate a chart based on the completion rate of the project. The system gets the project, feature and test case information and calculates the completion rate of all the test cases. Then the system follows an iterative process to calculate the completion rate for all the features and then finally calculates the completion rate of the test project.

For example: Consider a test project containing ten test features and each feature containing two test cases under them. Assuming that one test case passed in all the ten features, the completion rate is then calculated as 50% pass and 50% fail.
Trigger Completion Rate Calculation:

The completion rate can be defined as the percentage of testing that is complete. Test Engineer calculates the completion rate per test case and with the results obtained it calculates the completion rate for the test feature and then the project. It is necessary for the readers to understand when the completion rate calculation is triggered.

![Sequence diagram - Update to a test case resulting in the trigger.](image)

Figure 3.7 Sequence diagram - Update to a test case resulting in the trigger.

The Figure 3.7 describes a scenario when the completion rate calculation is triggered. In this scenario, the tester selects a test case to work on. The system loads the test steps for the test case. The tester verifies that the expected results match the actual result and updates the status based on the result obtained. The tester also uploads the screenshot and clicks on save. The system calculates the completion rate at this point and stores the information in the database.
Chapter 4

USER INTERFACE DESIGN

4.1 Interface Page Flow:

The major upgrade as a part of enhancement to the existing project was to improve on the design of the user interface. The user interface Fig 4.1[1] was restructured keeping the basic and most used features at the topmost level.

Figure 4.1 Interface Page Flow
4.2 Configuration Requirements:

The configuration requirements for the application include

- PHP compliant browser
- Javascript enabled browsers
- Flash enabled browsers

4.3 Interface Design Rules:

The interface must be designed using PHP to create dynamic HTML pages. Validation and data manipulation functions must be coded using either PHP or JavaScript. Additionally, in order to promote re-usable code, repeated sections should be separated into singular PHP “include” files.
4.4 Restrictions and Limitations:

The table 4.1 describes the access permissions for the users of the system – Administrator, Functional lead and tester. The top level components or the menu components are listed on the leftmost column and the subsequent columns describe the permission level for the users of the system.

Table 4.1: The restrictions and limitations for users of the system.

<table>
<thead>
<tr>
<th></th>
<th>Admin</th>
<th>Test Lead</th>
<th>Tester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasklist</td>
<td>view all tasklist</td>
<td>view only tasklist associated with his project</td>
<td>view only tasklist that are assigned to him</td>
</tr>
<tr>
<td>Projects</td>
<td>manage all projects</td>
<td>create new project and manage only his related projects</td>
<td>only view projects that are assigned to him</td>
</tr>
<tr>
<td>Features</td>
<td>manage all test suites</td>
<td>only manage test suites of associated projects</td>
<td>only view test suites of assigned tasklist</td>
</tr>
<tr>
<td>Test cases</td>
<td>manage all test cases</td>
<td>only manage test cases of associated projects</td>
<td>can manage test cases of assigned projects</td>
</tr>
<tr>
<td>Test steps</td>
<td>manage all test steps</td>
<td>only manage test steps of associated projects</td>
<td>can manage the test step of related test cases</td>
</tr>
<tr>
<td>Chart and Report Generation</td>
<td>Manage the charts and reports generation</td>
<td>Only generate charts and reports of associated projects</td>
<td>Only generate charts and of related test cases</td>
</tr>
</tbody>
</table>
Chapter 5

SYSTEM IMPLEMENTATION

This chapter describes the implementation details of the project. The implementation phase of a project is a critical step towards achieving the desired goals. It involves bringing the system design into action by following the requirements and also making it work. This chapter describes the technologies used for the project and also contains sample code to explain some of the main features in the application.

5.1 Technology Choices:
This project was initially developed with PHP scripting language and MySql database. Additionally javascript, html and cascading style sheets CSS were used to design the presentation layer of the application. In order to enhance the usability of the application, adobe flash technology is also used for the generation of charts.

PHP:

“PHP” [8],[9] is a server side scripting language originally designed for developing dynamic web pages. The beauty of php lies in the fact that the html code and php code can be embedded together for the presentation layer. In addition to being used to generate the dynamic interfaces, PHP was also used to create repeatable functions that were used for data validation, formatting, data and control output, and database connectivity.
MySQL:

MySQL is a relational database management system that is commonly used as the database component for web applications. It is freely available to the public and is distributed with open source licenses. It supports easy installation, configuration and editor or a client interface where the users can manually add the database and the tables without writing any sql code.

Cascading Style Sheets:

“CSS”[9] is a powerful styling language that allows developers to control the formatting of documents written in a markup language. With CSS, presentation attributes (font, background styles, control borders and sizes, etc.) are separated from the HTML markup and can be uniformly applied to web pages.

The ability to separate the code used to stylize the application outside of the content files allows the appearance of the system to be uniformly managed and controlled. Stored externally in a master style sheet (“style.css”), cosmetic adjustments to the entire system can be performed at a single point of reference and spares developers from having to modify multiple files when changes are implemented.

JavaScript

“JavaScript”[9] is an interpreted language that runs in the user's browser. JavaScript code works on any computer platform with a JavaScript capable browser. They enhance the dynamics and interactive features of Web pages by enabling calculations, checking forms, adding special effects, customizing graphics selections, data binding, and more.
Adobe Flash Player:

Adobe Flash player is a software for viewing multimedia, Rich Internet Applications and streaming video and audio, on a computer web browser or on supported mobile devices. Flash Player runs SWF files formats which delivers vector graphics, text, video, and sound over the Internet.
5.2 Methodology:

Test Engineer is a robust tool with a wide scope of features and in order to support the robustness of the application several methodologies are used. Below is a brief description of all the methodologies that were used.

Database Normalization:

Normalization is the process of organizing the data in the database to prevent inconsistent dependency and to eliminate redundancy. This includes creating tables, adding relationship between the tables and may be create intermediate tables if needed. Redundant data could often lead to inconsistent data, waste of disk space and can create problems during maintenance.

Test Engineer database is designed to use the First Normal Form and the second normal form to reduce the redundancy.

The First Normal Form

1. Eliminates repeating groups in individual tables
2. Identifies each set of data with a primary key.
3. Separates each set of related data into tables.

Example Usage of First Normal Form in the project: The project table stores records related to project information and each record of the database is identified uniquely.
Second Normal Form: When using Second normal form, the records do not depend on anything other than a table’s primary key. It separates sets of values that apply to multiple records into different tables. It also relates the tables with a foreign key.

Example Usage of Second Normal Form in the project: The test case and test feature tables contain foreign key that relate to test feature and test project table respectively.

Design Patterns:
Design patterns help structure code, although they do not say anything about how to implement it. They solve problems before we start to write code because they affect the design of programs by recognizing the possible abstractions in the problem. More than one pattern may apply, and within any pattern, many ways to implement it. We typically represent patterns with modules so their implementations are re-usable and abstract and make full use of encapsulation. The rest of the program does not need to know how the module works, and indeed, the more the program knows about the module’s workings, the more of a problem we have. Patterns promote loose coupling so that their implementation does not affect the rest of the program.

Singleton design pattern: This design pattern is used when only one object of a class needs to be instantiated at time. This pattern is implemented to configure the database connections where one global singleton object keeps a map of configuration key and value pairs. Apart from instantiating single object, this pattern also ensures that there is
only one place where the database configuration needs to be changed. This setup helps
the user to setup with any of the database available in the market.

Example code:

```php
<?php
$host = "localhost";
$user = "root";
$pass = "root";

$con = mysql_connect($host, $user, $pass);
if(!con)
    die('Could not connect: ' . mysql_error());
mysql_select_db("oldprojectdb", $con);
?>
```

The file dbopen.php contains the configuration details of the database. In order to
connect to a different database, the user needs to set the hostname, username and the
password for the database and the connection will be established.
Coding Conventions:

Coding conventions are a set of guidelines that are recommended for programming style, practices and methods for each part of the program. These are very essential to enhance the readability and maintenance of the code for a long term basis.

A set of rules are followed for the development of Test Engineer tool. This includes:

- Naming conventions followed for the database fields –
  - Fields of the database are written in lower case letters.
  - Fields having two or more words are separated by an underscore ( _ ) sign.
    
    Example : Fields like project_id

- Naming conventions followed for the development files –
  - The names of the development files are all written in lowercase letters.
  - Files having more than two words are separated by an underscore ( _ ) sign.
  - Files should always begin with the component name unless it is an independent file. Example: project_create.php, feature_insert.php
5.3 Functions and Includes:

The following listing in the Table 5.1 is an index of the PHP functions that were used throughout the development process to improve the organization and flexibility of the application.

Table 5.1: List of Functions and Includes

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agenda.js</td>
<td>Collection of methods and forms used for calendars.</td>
</tr>
<tr>
<td>case_select.php</td>
<td>Outputs a functional list of test cases for the active test suite.</td>
</tr>
<tr>
<td>charts_library</td>
<td>Collection of swf files for chart generation.</td>
</tr>
<tr>
<td>dbclose.php</td>
<td>Closes the connection to the MySQL database.</td>
</tr>
<tr>
<td>dbopen.php</td>
<td>Opens a connection to the webserver and host database.</td>
</tr>
<tr>
<td>email_notification.php</td>
<td>Collection of functions for email notification.</td>
</tr>
<tr>
<td>feat_select.php</td>
<td>Outputs a functional list of test suites for the active project.</td>
</tr>
<tr>
<td>footer.php</td>
<td>Outputs the formatted TEMP footer.</td>
</tr>
<tr>
<td>func_assigned.php</td>
<td>Collection of functions for assignments output at varied levels.</td>
</tr>
<tr>
<td>func_date.php</td>
<td>Collection of functions used for date formatting and output.</td>
</tr>
<tr>
<td>func_feat_box.php</td>
<td>Outputs a selection box for features in the active project.</td>
</tr>
<tr>
<td>func_file_list.php</td>
<td>Generates a file index based on input parameters.</td>
</tr>
<tr>
<td>func_login_req.php</td>
<td>Checks login status of current user and redirects as needed.</td>
</tr>
<tr>
<td>func_menu.php</td>
<td>Creates uniform submittal controls based on path/text input.</td>
</tr>
<tr>
<td>func_status_lists.php</td>
<td>Generates a &quot;Test Status Summary&quot; based on user input.</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>func_user_box.php</td>
<td>Creates an active user control.</td>
</tr>
<tr>
<td>header.php</td>
<td>Outputs the formatted TEMP header.</td>
</tr>
<tr>
<td>menu.php</td>
<td>Call to output the left-panel menu.</td>
</tr>
<tr>
<td>project_select.php</td>
<td>Outputs a functional list of available projects.</td>
</tr>
<tr>
<td>result_select.php</td>
<td>Outputs a functional list of test results for the active test case.</td>
</tr>
<tr>
<td>schedule_output.php</td>
<td>Creates a testing schedules based on passed input parameters.</td>
</tr>
<tr>
<td>style.css</td>
<td>Primary style sheet used for cosmetic formatting of the system.</td>
</tr>
</tbody>
</table>
5.4 Application Modules and Components:

The application can be divided into 8 major components which include:

1. Test Project,
2. Test Features,
3. Test Case,
4. Test Step,
5. Test Schedules,
6. Report and Chart generation,
7. Account Administration and
8. User Permissions.

Test Projects:

The test project module is the primary component for the Test Engineer management tool. This module focuses on the project level details. These application pages are as follows[1]:

- `project_create.php`
  - Input form for new test projects.
- `project_insert.php`
  - Database record insertion for test projects.
  - Receives input from `project_create.php`.
- `project_menu.php`
  - Bare-bones listing of created test projects.
- `project_select.php`
  - More complete listing of created test projects.
- `project_select2.php`
  - Accepts input from _select and _menu and loads the appropriate file.
- `project_update.php`
  - Update form used for accepting input for projects.
- `project_update2.php`
o Database record updates for test projects.
o Receives input from project_update.php.

- project_delete.php
  o Database record deletes for test projects.
o Receives input from the context.

- project_view.php
  o Read-only view of the current/selected project file.
o Makes call to func_status_lists for an updated testing summary.

- project_history.php
  o Accepts input from the project menu and load the history of the project.

- project_email_notification.php
  o Customizes the test project messages for email notifications.

Test Features:
The test feature module focuses on the feature-level details and uses the project that is in context for all its actions. These application pages are as follows:

- feature_create.php
  o Input form for new feature test suites.

- feature_insert.php
  o Database record insertion for test suites.
o Receives input from feature_create.php.

- feature_menu.php
  o Bare-bones listing of created test suites.

- feature_select.php
  o More complete listing of created test suites.

- feature_select2.php
  o Accepts input from _select and _menu and loads the appropriate file.
• feature_update.php
  o Update form used for accepting input for feature test suites.
• feature_update2.php
  o Database record updates for test features.
  o Receives input from feature_update.php.
• feature_delete.php
  o Database record deletes for test feature.
  o Receives input from the context.
• feature_view.php
  o Read-only view of the current/selected test suite file.
  o Makes call to func_status_lists for an updated testing summary.
  o Includes file upload/download form for test suite specific files.
• feature_history.php
  o Accepts input from the feature menu and loads the history of the feature.
• feature_email_notification.php
  o Customizes the test feature messages for email notifications.

Test Cases:

The test case module focuses on the case-level details and uses the case that is in context for all its actions. These application pages are as follows:

• case_create.php
  o Input form for new test cases.
• case_insert.php
  o Database record insertion for test cases.
  o Receives input from case_create.php.
• case_menu.php
  o Bare-bones listing of created test cases.
• case_select.php
- More complete listing of created test cases.
  - case_select2.php
    - Accepts input from _select and _menu and loads the appropriate file.
  - case_by_state.php
    - More complete listing of test cases by state.
  - case_update.php
    - Update form used for accepting input for feature test cases.
  - case_update2.php
    - Database record updates for test cases.
    - Receives input from case_update.php.
  - case_delete.php
    - Database record deletes for test case.
    - Receives input from the context.
  - case_view.php
    - Read-only view of the current/selected test case file.
    - Makes call to func_status_lists for an updated testing summary.
  - case_history.php
    - Accepts input from the case menu and loads the history of the case.
  - case_email_notification.php
    - Customizes the test case messages for email notifications.
  - case_completion_rate.php
    - Calculates the completion rate for the test case.
Test Steps:

The test step module focuses on the step-level details and uses the case and step that is in context for all its actions. These application pages are as follows:

- **step_create.php**
  - Input form for new test step.
- **step_insert.php**
  - Database record insertion for test step.
  - Receives input from step_create.php.
- **step_select.php**
  - More complete listing of created test step.
- **step_select2.php**
  - Accepts input from _select and _menu and loads the appropriate file.
- **step_update.php**
  - Update form used for accepting input for test steps.
- **step_update2.php**
  - Database record updates for test steps.
  - Receives input from step_update.php.
- **step_delete.php**
  - Database record deletes for test step.
  - Receives input from the context.
- **step_view.php**
  - Read-only view of the current/selected test step file.
  - Makes call to func_status_lists for an updated testing summary.
- **step_history.php**
  - Accepts input from the step menu and loads the history of the step.
- **step_email_notification.php**
  - Customizes the test step messages for email notifications.
Screenshot Upload:

The “screenshot” module contains all of the code necessary for uploading screenshots, downloading screenshots, outputting file type headers, and interface creation.

- screenshot_upload.php
  - This module checks the current project level, feature level, test step level and outputs the proper “downloadable file index” and “upload interface”.
  - Valid levels are “test step”

- file_view.php
  - Loads a selected file in a new window.
  - File type headers need to be manually outputted.
    - Current default header is for images
Account Administration and Permissions

Each file in this module deals with the code necessary for managing user accounts and privileges. All files prefaced by “perm_” are used to handle test suite level permissions; while all files prefaced by “user_” deal with user accounts.

- perm_list.php
  o Interface used for selecting and loading specific user permissions.
- perm_user.php
  o Interfaced for viewing and updating user permissions.
- user_add.php
  o Record insertion for new account and employee information.
- user_create.php
  o Account creation interface.
- user_list.php
  o Index file for user accounts and employee details.

Chart and Report Generation

The “Chart and Report Generation” module brings together all of the components necessary for creating complete project reports by running the appropriate queries and making external function calls.

- report_create.php
  o Checks the file type parameter from reports.php then requests report.
- report_htm.php
  o Takes the form input from reports.php and generates the report.
  o SQL queries and calls modified based on user-input.
- reports.php
  o User input page for report-type selections.
• charts.php
  o User input page for chart-type selections.

• chart_create.php
  o prepares input for the chart.

• chart_insert.php
  o inserts the chart type to the php page.
  o inputs from the chart_create.php
5.5 Table Declarations:

The following queries can be used to recreate the database tables and create associations between them.

CASE_HISTORY

CREATE TABLE IF NOT EXISTS `case_history` (  
`case_hist_id` bigint(20) NOT NULL AUTO_INCREMENT,  
`case_id` bigint(20) NOT NULL,  
`case_name` text NOT NULL,  
`revision` text NOT NULL,  
`description` text NOT NULL,  
`state_id` int(11) NOT NULL,  
`completion_rate` int(11) NOT NULL,  
`updated_by` text NOT NULL,  
`type` text NOT NULL,  
`update_ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
PRIMARY KEY (`case_hist_id`)  
) AUTO_INCREMENT=1 ;

CASE_NOTES

CREATE TABLE IF NOT EXISTS `case_notes` (  
`case_id` int(11) DEFAULT NULL,  
`username` varchar(20) DEFAULT NULL,  
`last_updated` datetime DEFAULT NULL,  
`user_notes` text,  
`deleted` varchar(1) DEFAULT NULL  
) ;
EMPLOYEE

CREATE TABLE IF NOT EXISTS `employee` (  
`uid` int(11) DEFAULT NULL,  
`fname` varchar(20) DEFAULT NULL,  
`lname` varchar(20) DEFAULT NULL,  
`phone` varchar(25) DEFAULT NULL,  
`email` varchar(30) DEFAULT NULL,  
UNIQUE KEY `uid` (`uid`)  
);  

FEATURE

CREATE TABLE IF NOT EXISTS `feature` (  
`feat_id` int(11) NOT NULL AUTO_INCREMENT,  
`project_id` int(11) DEFAULT NULL,  
`feat_name` varchar(32) DEFAULT NULL,  
`version` varchar(32) DEFAULT NULL,  
`est_hours` int(11) NOT NULL,  
`act_hours` int(11) NOT NULL,  
`description` text,  
`username` varchar(20) DEFAULT NULL,  
`create_date` datetime DEFAULT NULL,  
UNIQUE KEY `feat_id` (`feat_id`)  
) AUTO_INCREMENT=1 ;
FEATURE_HISTORY

CREATE TABLE IF NOT EXISTS `feature_history` (
  `feat_hist_id` bigint(20) NOT NULL AUTO_INCREMENT,
  `feat_id` bigint(20) NOT NULL,
  `feat_name` text NOT NULL,
  `version` text NOT NULL,
  `est_hours` int(11) NOT NULL,
  `act_hours` int(11) NOT NULL,
  `description` text NOT NULL,
  `updated_by` text NOT NULL,
  `type` text NOT NULL,
  `update_ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`feat_hist_id`)
) AUTO_INCREMENT=1;

PROJECT

CREATE TABLE IF NOT EXISTS `project` (
  `project_id` int(11) NOT NULL AUTO_INCREMENT,
  `project_name` varchar(32) DEFAULT NULL,
  `version` varchar(20) DEFAULT NULL,
  `developer` text,
  `src_path` text,
  `exe_path` text,
  `username` varchar(20) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  UNIQUE KEY `project_id` (`project_id`)
) AUTO_INCREMENT=1;
PROJECT_HISTORY

CREATE TABLE IF NOT EXISTS `project_history` (  
`proj_his_id` int(11) NOT NULL AUTO_INCREMENT,  
`project_id` bigint(20) NOT NULL,  
`project_name` text NOT NULL,  
`version` text NOT NULL,  
`src_path` text NOT NULL,  
`exe_path` text NOT NULL,  
`developer` text NOT NULL,  
`type` text NOT NULL,  
`updated_by` text NOT NULL,  
`update_ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
PRIMARY KEY (`proj_his_id`)  
) AUTO_INCREMENT=1 ;

RESULT_STATE

CREATE TABLE IF NOT EXISTS `result_state` (  
`state_id` int(11) NOT NULL AUTO_INCREMENT,  
`status` tinytext,  
`active` varchar(1) DEFAULT NULL,  
UNIQUE KEY `state_id` (`state_id`)  
) AUTO_INCREMENT=1 ;
TESTCASE

CREATE TABLE IF NOT EXISTS `testcase` (  
  `feat_id` int(11) DEFAULT NULL,  
  `case_id` int(11) NOT NULL AUTO_INCREMENT,  
  `case_name` varchar(32) DEFAULT NULL,  
  `revision` tinytext,  
  `description` text,  
  `state_id` int(11) NOT NULL DEFAULT '1',  
  `completion_rate` decimal(10,0) DEFAULT NULL,  
  `username` varchar(20) DEFAULT NULL,  
  `create_date` datetime DEFAULT NULL,  
  UNIQUE KEY `case_id` (`case_id`)  
) AUTO_INCREMENT=1 ;

TESTSTEP

CREATE TABLE IF NOT EXISTS `teststep` (  
  `step_id` bigint(20) NOT NULL,  
  `description` varchar(200) NOT NULL,  
  `case_id` bigint(20) NOT NULL,  
  `exp_result` varchar(200) DEFAULT NULL,  
  `status` varchar(4) NOT NULL DEFAULT 'Fail',  
  `attachment` mediumblob,  
  `create_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP  
);
CREATED TABLE IF NOT EXISTS `up_files` (  'file_id' int(11) NOT NULL AUTO_INCREMENT,  'project_id' int(11) DEFAULT NULL,  'feat_id' int(11) DEFAULT NULL,  'case_id' int(11) DEFAULT NULL,  'step_id' int(11) DEFAULT NULL,  'username' varchar(20) DEFAULT NULL,  'up_date' datetime DEFAULT NULL,  'file_blob' longblob,  'file_name' text,  'file_size' int(11) DEFAULT NULL,  'file_notes' text,  UNIQUE KEY `file_id` (`file_id`) ) AUTO_INCREMENT=1 ;

CREATED TABLE IF NOT EXISTS `users` (  'uid' int(11) NOT NULL AUTO_INCREMENT,  'username' varchar(20) DEFAULT NULL,  'password' varchar(32) DEFAULT NULL,  'access' text NOT NULL,  'create_date' datetime NOT NULL,  'create_user' varchar(20) NOT NULL,  'update_date' datetime NOT NULL,  'update_user' varchar(20) NOT NULL,  UNIQUE KEY `uid` (`uid`) ) AUTO_INCREMENT=1 ;
Chapter 6

APPLICATION SCREENSHOT

This chapter focuses primarily on the application screenshots and describes the usage and navigation between the application pages. The user interface of the application is enhanced with consistent fonts, limited consistent colors, and consistent font sizes all across the application. The navigation is also altered to provide an easy flow of the application for the users.
6.1 Login Screen:

![Image of login screen]

Figure 6.1 Login Screen

The login screen is upgraded with consistent colors, font and an updated Temp picture to enhance the look. This screen is used by the users to log into the application with their Username and password. The assumption is that the admin creates the user accounts and provides them to the users through email.
6.2 Menu Bar:

The menu bar is designed to intelligently display components based on the logged in user. It contains tabs for task list, project, test cases, history, assignments, reports, permissions, accounts and logout.

Task List is the default page which displays all the tasks assigned to the user. Project tab navigates to display all the projects associated to the logged in user. Features and test cases tab navigates to display all the features and test cases of the selected project. History tab provides a link to view the history of actions on the project. The functional lead or admin can use the Assignment tabs to assign tasks to the users. The reports tab links to generation of reports and charts for the project.

Figure 6.2 Differences between the menu option for admin and other user.

The Figure 6.2 displays the differences in the menu options for admin and other users. The permissions and accounts tabs are used to set user permissions and to create accounts for the users. Administrators are the only users who can access the permissions and
accounts tabs in the menu and thereby hiding the functionality from the other users of the system. The Logout tab logs the user out of the session.

6.3 Task List Screen:

The application navigates to the task list screen upon successful login to the application. This page gives an overview of all the tasks that are associated to the logged in user.

![Task List for Admin](image)

**Figure 6.3 Task List for Admin**

The task list page displays all assigned and in progress test cases for the tester. The functional lead can view all the test features or test projects that are assigned or in progress state.
6.4 Project Menu Screen:

This page displays a list of projects associated to the logged in user. The page includes a table which gives brief information of the total number of features associated to the project, the version and the owner of the project.

![Figure 6.4 Project Menu](image)

The projects listed in the table hyperlink to the project overview page. The page also provides the feature to add a new project or search an existing project if not listed in the menu.
6.5 Project Overview:

The project overview page displays all the information of the project that is in context. The information can be modified upon the click of the Edit Project Details button which then enables editing of the fields. The page also provides a link to the project history page with the help of the View History button.

![Figure 6.5 Project Overview](image)

All the features that are associated to the project are also displayed in the tabular fashion. The table provides the total number of test features associated to the project, the owner and the version number. The features listed in the table also hyperlinks to the feature overview page.
6.6 Feature Overview:

The Feature overview page displays all the information of the test feature that is in context. The information can be modified upon the click of the Edit Feature Details button which then enables editing of the fields. The page also provides a link to the feature history page with the help of the View History button.

Figure 6.6 Feature overview

All the test cases that are associated to the test feature are also displayed in the tabular fashion. The table provides the total number of test cases associated to the project, the owner and the version number. The test cases listed in the table also hyperlinks to the test case overview page.
The testing status table displays the total number of test cases associated to the feature that are in assigned, pending, in progress, passed, failed and inconclusive states. The numbers hyperlinks to display the list of test cases based on their status only.
6.7 Test Case Overview:

The test case overview page displays all the information of the test case that is in context. The information can be modified upon the click of the Edit test case Details button which then enables editing of the fields. The page also provides a link to the test case history page with the help of the View History button.

![Test Case Overview](image)

Figure 6.7 Test case overview

All the test steps that are associated to the test feature are also displayed in the tabular fashion. The table lists the steps in the order of execution, the description of the steps, the expected result, actual result if the test case passed or failed and also displays the link to the screenshot of the executed step. New test steps can also be added to the test case
using the Create test Step button which navigates the Create Test step page. The steps also hyperlinks to the step overview page where the step can be edited and the screenshot can be attached.
6.8 Step Overview Page:

The step overview displays all the information of the test step that is in context. The information can be modified upon the click of the Edit test step button which then saves the fields that were edited. The page also provides a link to the test case overview page upon the click on cancel button.

Figure 6.8 Step Overview page

This page will be used by the tester to update the status of the test step and to attach screenshot for the test step. Upon the click on the browse button, the options are provided to attach the file in the respective folders. The file is then selected and uploaded using the Upload button.
Figure 6.9 Page displays a delete button if a screenshot is already uploaded. If a screenshot attachment exists, then the page gives the option to delete the screenshot and attach a new one upon deletion.
6.9 View History Screen:

This page displays the history log information for the test case in a tabular manner. All the user actions such as creation, deletion and modifications are logged and displayed on this page. The description also specifies the name of the user who made the modifications and the date and time of the modification is also listed. The table displays the most recent action on the top of the table and the least recent at the bottom of the table.

Figure 6.10 View History log for test case Invalid Login.
6.10 Generate Charts

This page displays the options necessary for the generations of charts. The user can select the test project from the dropdown and then select all the features or just a feature of his choice. The chart type gives the options for generating different types of charts for the option selected. Upon the click on the View Chart button the chart is generated for display.

Figure 6.11   Generate Chart

The page provides the option to generate the charts for

1. Completion rate of the project.
2. Completion rate of the feature.
3. Estimated versus Actual hours of the project.
4. Estimated versus Actual Hours of the feature.
5. A trend of the project progress.
Figure 6.12 Display Estimate versus Actual Hours of test feature.

The Figure 6.12 is the screenshot of a chart generated by selecting estimated versus the actual hours of all the test cases in the test feature. The numbers on the horizontal axis represents the test case identifiers and the number on the vertical axis represents the number of hours.
Chapter 7

CONCLUSION AND FUTURE WORK

This project is aimed to provide an open source tool for the Department of Software engineering to manage and plan test cases. The intent is to provide an opportunity for the students to get accustomed to tools available in the software industry and also help the department to manage their testing for their internal projects. This tool will help speed up the testing process and also prevent any duplication of work, thus saving time and resource necessary for planning and managing test cases.

Future work would include:

1. Packaging the software: This can include writing script to install and configure the webserver, database and the client.
2. Batch process or a stored procedure: To purge history records or to migrate the history records of a year.
3. Automated testing: The tool now supports only manual testing and can be enhanced to support automated testing.
4. Screenshot attachment: Updating the existing functionality of the screenshot attachment directly inside the tool rather than from the outside.
5. Integration with other tools: To provide flexibility to integrate with other tools especially the defect tracking tools like Rational Clearquest, JIRA, etc.
APPENDIX

Glossary of Terms

CSS: Cascading Style Sheets: a markup language used to standardize and style the formatting of web pages

Html: Hypertext markup language, the language used to create the basic makeup of the web pages

Hyperlink: A technical term for a link which on a click navigates to a new page.

Instantiation: A technical term used in object oriented methodology used referring to the action of creating an object instance.

JavaScript: Scripting language used for client-site web development

MySQL: The database access and programming language variant of SQL that is used by the host server

Permissions: Administered access to perform different actions within a system

PHP: Programming Language that is used for creating web pages that takes advantage of server-side programming

Project: An application or system made up of several components whose testing will be separated into test suites

Query: An information request made to the database. Answers to the question (formatted in SQL) are returned in a result (or record) set

Record/Result Set: A set of rows from a database, as well as meta-information about the query such as the column names, and the types and sizes of each column

Sequence Diagram: A kind of interaction diagram which depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
SQL: Structured Query Language, the language used to create, query, and delete data from a DBMS

Test Case: A set of conditions that a tester will check to determine whether a system will meet specified requirements

Test Feature: A collection of test cases that are grouped to test a specific behavior or function.

Test Steps: A list of steps that needs to be executed in an orderly fashion for the execution of a test case.

UML: Unified Modeling Language, a standardized specification language for object modeling used to create an abstract model of a system.
BIBLIOGRAPHY


Available:

[4] The Video tutorials, trial download link for the Seapine test track software [Online]
Available:

[5] Quality Center (HP) – A testing tool offered by HP [Online]
Available:


[8] The differences between PHP 4 and PHP 5[Online]

Available: http://www.w3schools.com/php/
Available: http://www.w3schools.com/css/default.asp
Available: http://www.w3schools.com/js/default.asp

[10] Steps to set up Email exchange server [Online]


Available: http://www.seapine.com/