PROJECT QUALITY TOOL: A TOOL FOR PROJECT SUCCESS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Pallavi Srichinta

In Partial Fulfillment
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

December 2012

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PROJECT QUALITY TOOL: A TOOL FOR PROJECT SUCCESS

By

Pallavi Srichinta

The Supervisory Committee certifies that this ***disquisition*** complies with

North Dakota State University's regulations and meets the accepted standards

for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kenneth Magel
Chair

Dr. William Perrizo

Dr. Simone Ludwig

Dr. Jacob Glower

Approved:

12/20/2013	Brain M. Slator
Date	Department Chair

ABSTRACT

This paper proposes a solution to the current changing requirements communication problem in an offshore on-site software development model. The proposed model is a web-based tool where the user in a project team can enter the new Requirements, map them to Design, create Test Cases from design, Execute them, and track failed ones by creating Defects. When the requirements change the existing tools available in the market, the changes are not communicated to the entire project team; leaving the Quality Assurance team verifying old (incomplete) requirements which ultimately costs more time, money and delays the project delivery.

In this paper, a prototype tool intended to automatically handle the above-mentioned communication problems whenever requirements are changed after the design is in place. The prototype manages the gap between on-site and offshore teams and adds value to the project development by saving time, money, and improving the quality of the final product.

ACKNOWLEDGEMENTS

To My Parents, My Husband, My Daughters and My Sister for all the support and encouragement in completing this Project.

To My Advisor for his continuous support, guidance and patience.

To My Committee for their time, support and suggestions.

To Stephanie for her help during the entire process.

DEDICATION

To My Father and Mother

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

DEDICATION v

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF ABBREVIATIONS xi

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. LITERATURE REVIEW 5

2.1. Bugzilla 6

2.2. HP Quality Center 6

2.3. JIRA 8

2.4. Subversion 8

2.5. TRAC 8

2.6. GEMINI 9

2.7. IBM Rational ClearQuest 9

2.8. Shortcomings of Existing Tools 10

CHAPTER 3. RESEARCH APPROACH 12

3.1. Overview of Project Quality Tool 12

3.2. Brief Introduction about Project Quality 14

3.3. Functional Overview of Project Quality Tool 15

3.4. Project Quality Tool Architecture 29

3.5. Flow Charts of the Project Functionality 30

CHAPTER 4. RESEARCH EVALUATION 34

CHAPTER 5. CONCLUSION 42

REFERENCES 43

LIST OF TABLES

Table Page

1. Experiment Project Quality Tool versus Other Tools Mentioned in Chapter 2 34

LIST OF FIGURES

Figure Page

1. *Silverlight Architecture* from Microsoft MSDN Library Entity Data Model 13

2. Log-in Screen 15

3. Project Selection Screen 16

4. Home Screen 16

5. Requirements Submit Button 17

6. Requirements Data Grid 17

7. Options and Update Button in Requirements Tab 18

8. Design Tab 19

9. JAD Session Link 20

10. Development Status Dropdown 20

11. Options in Design Tab 21

12. Design Data Grid 22

13. Test Suite Tab 22

14. Options in Test Suite Tab 23

15. Update Button in Test Suite Tab 24

16. Test Execution Tab 24

17. Test Execution Data Grid 25

18. Defects Tab 26

19. Update Button in Defects Tab 26

20. Reports Tab 27

21. Project Quality Tool Database Diagram 29

22. Flow Chart of User Logging In and landing in project Home Screen Page 30

23. Flow Chart of Inserting Requirements in a Selected Project 31

24. Flow Chart of Updating Requirements and Project Team Email Notification 32

25. Flow Chart of Inserting Design Document and Design Phase Details in a Project 33

26. Sample Gift Card Project Requirements in Requirements tab.. 36

27. Sample Gift Card Project Design Tab 37

28. Sample Gift Card Project Test Suite tab 37

29. Sample Gift Card Project Unit Test Results Link... 38

30. Sample Gift Card Project Changed Requirements.. 39

31. Sample Gift Card Project Requirement Change Email Notification 39

32. Sample Gift Card Project Design Change Email Notification.. 40

33. Sample Gift Card Project Report Showing Unmapped Requirements 41

LIST OF ABBREVIATIONS

QA	Quality Assurance
UAT	User Acceptance Testing
TCERs	Test Cases and Expected Results
SDLC	Software Development Life Cycle

CHAPTER 1. INTRODUCTION

In the current offshore onsite model of software development, delivering high quality, testable code in a timely manner is a challenge. Fundamental to developing such a code is a clear understanding and communication of the project's requirements. Unfortunately, complete and correct communication of the code's requirements is difficult using current development tools. This can result in a code which does not meet the requirements, incorrect test procedures, and possibly even a code which is the opposite of what the client has asked for. In order to save time and money, many companies outsource some of the phases of software development. During this process, the analysis phases such as planning and requirements gathering are often done onsite in order to be closer to the users (and likewise improve communications) while design, development, and testing are performed offshore. Towards the end of the development cycle, deployment and support are often brought back to the on-site facility prior to delivery. The design and code reviews happen but they often focus on data types and data flow rather than end-to-end functional requirement details since the product has not yet been developed. Once feedback from the reviews is incorporated into the design, development begins.

Even greater challenges to developing codes arise when the requirements change during the course of development. This creates problems with communications – especially since teams are required to act fast. Typically, changed requirements are communicated to on-site technical leaders, who are then responsible for assuring that these changed requirements are communicated to offshore development and quality assurance team members. Complicating matters, offshore development and quality assurance members can be a long distance away, in different time zones, with different work schedules. This can result in the changed requirements not being communicated to the correct people at the right time. If the changed requirements do not reach the quality assurance teams, they will be designing test cases from the original incorrect document. This, in turn, results in the Test Cases and Expected Results (TCERs) not being updated according to the changed requirements. This may result in two things: 1) Testing something wrong/outdated, or 2) re-testing/rotation is a challenge due to lack of documentation in the TCERs. Both are very stressful and may result in incomplete testing due to the crunch time. Quality assurance (QA) would finish the system testing, integration testing and inform the on-site facility that they are ready for User Acceptance Testing (UAT) at the last minute. Since the UAT is coordinated completely on-site, the user scenarios will be documented with continuous UAT prep meetings, and as the offshore QA team could not test it thoroughly, there is a fair chance that the defects would be identified in UAT. And this goes on between development and user acceptance testing until the UAT defects are fixed.

Another problem in the offshore on-site model is that the Unit Testing is often not performed by the developers. Code is passed to the quality assurance server, thinking that the QA team is responsible for the entire testing. This results in degradation of the testing effort as the testing team is now doing and digging for the unit test defects during their system testing phase, and sometimes the code will not even be in an executable condition because of issues in the test box. To make matters worse, the Quality Assurance team is often not allocated extra time in spite of the increased workload.

To overcome these problems, this paper proposes the Project Quality Tool. This is an easy to use tool for enterprise requirements specification, design documentation, test planning, test execution and defect tracking; features that, to the author’s knowledge, are not available in other tools that are currently on the market.

The Project Quality Tool simplifies the Software Development Life Cycle (SDLC), and reduces human intervention in communication of the changed requirements. This tool not only assures that the changed requirements are communicated and incorporated into the project design, but also into the subsequent phases like testing. The logical interconnection between the requirements and design is that the design and test suite is aiding the tool to be able to coordinate the pending tasks. To solve the unfinished Unit Testing problem and also some other issues, this tool includes features that makes the Unit Testing Results mandatory in order to begin the testing phase of any delivered code.

The Project Quality Tool is similar to a dashboard where business analysts put their requirements together, and developers upload their design documents. In addition, JAD sessions are recorded and stored in .wav files on a shared drive pointing the path in the design dashboard, and also uploading the whiteboard pictures to the dashboard, plus storing the unit test results for each module.

In the Test Suite tab, the QA testers would work on creating the test cases based on the design document and detailed requirements. If the requirements are updated, then the email is triggered to the project team to update the design document; and, if the design is updated, then an email notification will be sent to the project team to update the TCERs.

In the Test Execution tab, QA testers would pull the TCERs from Test Suite to run them in sets like System Testing Cycle 1, Integration Testing Round 1, or they can pull the TCERs into a set each day to monitor their execution on a daily basis. The former is recommended for small to large projects, while the latter is recommended for tiny enhancements.

There is a Defect tab which actually holds the defect or issue information. The testers, as they see any scenario failing, would open a defect in this tab and then wait for the defect to be fixed, then work on test execution after testing the fixes first, as defects take priority over the new functional testing.

The Reports tab is an information tab for the project team. The Project Status Report is generated with a button click, and can be used to monitor and track these changes in each team meeting; and, it will always be visible to all the offshore and on-site members in the project team.

CHAPTER 2. LITERATURE REVIEW

In this chapter, an overview is presented of the main problems encountered in offshore, on-site or distributed software development models which are relevant to this study. In addition, an overview of popular project quality tools currently available is presented.

What is Requirements Management? Requirements management is "a systematic approach to eliciting, organizing and documenting the requirements of the system and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system" (Leffingwell & Widrig, 2000).

McAllister stated that, "Software Applications often fail to satisfy User's expectations if developers misunderstand their requirements" (McAllister, 2006). One reason this happens is ineffective communication. "Miscommunication and misunderstanding between software developers and users are at the heart of the requirements dilemma" (DeBellis & Haapala, 1995, 2006).

According to George Stepanek (2012), software is normally commissioned for the needs of the users and managers, and not professional developers. These individuals understand the business process, but it is still very difficult to take into account all of the alternative flows and error conditions, and how the requirements are interrelated. Stepanek (2012) further stated that it is impossible to accurately blueprint software or draw up a complete set of requirements before the actual software development, as the users would gain insight into their needs as the software application starts to take shape (Stepanek, 2012).

To avoid developing something that does not fulfill the users' requirements, it is important to have traceability to the Software Development Life Cycle (SDLC) artifacts documented during the software's development cycle. Spanoudakis and Zisman (2005) define software traceability as the “ability to relate artifacts created during the development of a software system to describe the system from different perspectives” (pp. 256-273). The traceability assures that the required features are implemented and thoroughly tested. Based upon the author’s experience, traceability is performed after the fact, and is done manually by the Requirement Traceability Matrix (Sundaram, Hayes, Dekhtyar, & Holbrook, 2008).

Below are some of the tools that are available in the market which are being used by companies for tracking several phases in the software development process.

2.1. Bugzilla

Bugzilla is a web-based general purpose bug tracker and testing tool originally developed and used by the Mozilla project, and is licensed under the Mozilla Public License. Released as open source software by Netscape Communications in 1998, it has been adopted by a variety of organizations for use as a bug tracking system, and occasionally as a data source for project management software. It is used for both free and open source software, as well as proprietary projects and products (http://www.bugzilla.org/docs/4.2/en/pdf/Bugzilla-Guide.pdf).

2.2. HP Quality Center

HP Quality Center (QC), formerly known as Mercury Quality Center, is a web-based test management software tool from Hewlett-Packard, acquired from Mercury Interactive Corporation. HP Quality Center offers software quality assurance, including requirements management, test case management, test execution and defect tracking for IT and application environments (HPQC13). In HP Quality Center, the testing process includes five phases:

1. Specifying Releases
2. Specifying Requirements
3. Planning Tests

4. Running Tests
5. Tracking Defects

Release cycles are specified. In each release cycle, specific requirements that need to be tested are chosen for Test Planning. In Test Planning, test scripts are planned and documented. When the build becomes available, the scripts are run in the order that they are meant to be run. Defects identified in testing are tracked and re-tested. Once the requirements are created and test cases are approved, if the requirement is changed, there is no automatic tracking or notification sent to the project team to update the downstream deliverables. Hence, it leaves a large gap between what is required and what is tested. Whereas in Project Quality, all of the below-mentioned phases are integrated:

1. Specifying Requirements
2. Design Documentation
3. Test Case Design
4. Test Execution
5. Defects Tracking
6. Reports

If any requirement has been changed or added, the project team will receive notification and the downstream deliverables should be updated with the changed requirements. This helps us in requirement traceability, and in testing what the business actually wanted. Based on this, the subsequent phase of the software development life cycle puts the production support staff at ease. (http://en.wikipedia.org/wiki/HP_Quality_Center)

2.3. JIRA

JIRA is an issue tracking tool, and will not facilitate any other phases in software development. It was developed by Atlassian, commonly used for bug tracking, issue tracking, and project management. The product name JIRA is a truncation of "Gojira," the Japanese name for Godzilla. It has been developed since 2002. (http://wpc.29c4.edgecastcdn.net/8029C4/downloads/software/jira/downloads/documentation/JIRA%205.0%20Documentation%20(PDF)%2020120305.pdf).

2.4. Subversion

Subversion is a version control tool used to keep track of code versions. This tool is similar to a popular CVS tool, and is currently being used to maintain the new version and previous versions of source codes, and files relative to application and documentation. There is no provision in this tool to hold SDLC deliverables. (http://books.google.com/books?hl=en&lr=&id=b3TsOSvU5TUC&oi=fnd&pg=PR4&dq=Subversion+tool&ots=zTGB2W6gqA&sig=vPe9Bm_K-3LIksclw4MCM9JaUhM#v=onepage&q=Subversion%20tool&f=false).

2.5. TRAC

TRAC allows hyperlinking information between a bug database, revision control and wiki content. It also serves as a web interface to the following revision control systems: Subversion, Git, Mercurial, Bazaar, Perforce and Darcs. Other features include: project management (Roadmap, Milestones, etc.), ticket system (bug tracking, tasks, etc.), Wiki (syntax similar to MoinMoin), customized reporting and multiple project support (Trac13). (http://en.wikipedia.org/wiki/Trac)

2.6. GEMINI

Gemini is a proprietary web-based issue tracking and bug tracking system provided by Countersoft, Ltd. Gemini is based on Microsoft's .NET Framework. It was developed using C#, ASP.Net, and Microsoft SQL Server. Gemini supports integration with Visual Studio, Outlook, Subversion (SVN) and Windows Live Messenger. Gemini is proprietary software, but a freeware 10-user license is available, and optional source code licensing is available with the enterprise license (Gemini13). (http://en.wikipedia.org/wiki/Gemini_(issue_tracking_system)

2.7. IBM Rational ClearQuest

Rational ClearQuest is an enterprise level workflow automation tool from the Rational Software Division of IBM. Commonly, ClearQuest is configured as a bug tracking system, but it can be configured to act as a CRM tool, or to track a complex manufacturing process. It also has the ability to implement these functions together. ClearQuest is a client-server application although there is no ClearQuest "back-end." Rather, the ClearQuest clients utilize an existing database server. ClearQuest supports various back-end databases including Oracle, SQL Server and IBM DB2. ClearQuest has been criticized for price, which is four to five times that of competing products (IBMCQ13). (http://en.wikipedia.org/wiki/IBM_Rational_ClearQuest)

Some of the above-mentioned tools are open source and some are proprietary. But, in either case, these tools cannot provide centralized solutions for storing the details of end-to-end SDLC artifacts. Moreover, companies which use the above-mentioned tools should utilize a separate requirement traceability method to track if all of the requirements specified by the users made it to the code, and are perfectly tested.

2.8. Shortcomings of Existing Tools

Based upon the author’s experience, having worked in the IT industry for eight years, the requirements that any end-to-end project tracking tool should have are as follows:

1. Facilitate requirement gathering for multiple projects by business analysts.
2. Provide ability to store design document details in the same portal by the Application Development Team.
3. Provide ability to map and trace the requirements to the design documents.
4. Provide ability to store JAD Session discussion details and white board sessions which are very important for production support.
5. Provide ability to store and make Unit Testing mandatory before moving onto the System Testing phase.
6. Provide ability to map the Design to Test Cases which in turn maps to the Requirements.
7. Provide ability to create Test Cases and Expected Results by QA Team.
8. Provide ability to track the Test Execution.
9. Provide ability to track the Defects identified until resolved.
10. Provide ability to broadcast the CHANGED REQUIREMENT to the Project Team when the project is in-flight.
11. Provide ability to broadcast the CHANGED DESIGN to the Project Team when there is any change in Design when the project is in-flight.
12. Provide ability to broadcast the CHANGED Test Case and Expected Results to the Project Team when the project is in-flight.
13. Provide ability to Review the artifacts from all the phases in the SDLC.

14. Provide ability to print the up-to-date Reports for management and project team review.
15. Cost effective.
16. Ease of use.
17. Project documentation centralization.
18. Compatibility to any browser.
19. Platform independent and machine independent.
20. Works faster because of the architecture.
21. Better warranty when handing off to the production support team.
22. Provide Live Traceability from Requirement to Design, Design to Test Cases, and Test Cases to Test Execution.

None of the previously described software packages meet all of these requirements. As a result, a new package, termed the *Project Quality Tool*, is developed and presented in the following chapters.

CHAPTER 3. RESEARCH APPROACH

This chapter presents the research approach used, and a proposed solution to overcome the limitations of the most commonly used tools available in the market. This solution is called the *Project Quality Tool* hereafter. The advantages and significance of each module will be presented for each module as well.

3.1. Overview of Project Quality Tool

The Project Quality Tool is a web-based application, developed to support multiple phases of a software development life cycle. It is developed in C# using Visual Studio 2010, and the front end is developed in Silverlight 4.0. The database server is SQL Server 2008. The following paragraph explains in detail why I have chosen this framework.

Microsoft Silverlight is a cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences, and rich interactive applications for the web. *Silverlight* uses the Extensible Application Markup Language (XAML) to ease User Interface development (e.g. controls, animations, graphics, layout, etc.) while using managed code or dynamic languages for application logic. *Silverlight* is very quick in response. The beauty of the tool is that it reads the user's input and updates the display without refreshing the whole page. I have created the Project Quality Tool using C# code with the help of the Visual Studio development tool.

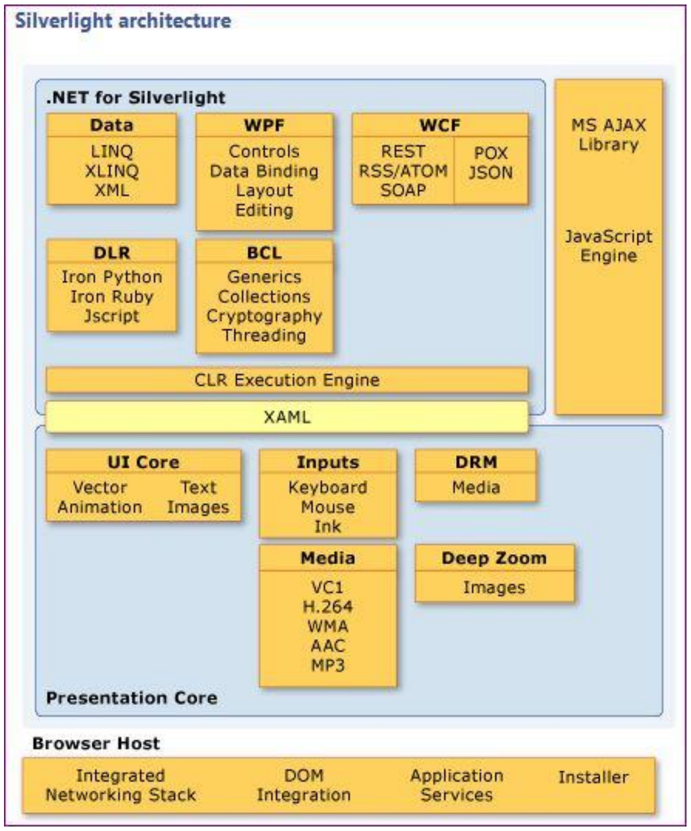


Figure 1. Silverlight Architecture from Microsoft MSDN Library Entity Data Model

The Entity Framework enables developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema.

The Entity Framework is an object-relational mapper that reduces the impedance mismatch between the object-oriented world of .NET Framework developers and the world of relational databases. It enables developers to primarily interact with an application’s conceptual model, using familiar object-oriented techniques. In the Entity Framework, you can work with data in the form of domain-specific objects and properties, such as User and User’s email, without having to concern yourself with the underlying database tables and columns where this data is stored. Developers can issue data access operations against the conceptual model, and the

Entity Framework translates the operations into relational database actions. There are two major layers in an Entity Framework application:

1. The modeling layer
2. The object layer

Microsoft SQL Server is a database management and analysis system for data warehousing solutions. The SQL Server contains a variety of features and tools that we can use to develop and manage the databases and solutions. SQL Server is much more robust and scalable than a desktop database management system such as Microsoft Access.

3.2. Brief Introduction about Project Quality

The following sections will demonstrate various functionalities of Project Quality Tool:

1. Login Interface – Existing users would log on.
2. Project Selection – User can select the desired project from the list of projects drop down box.
3. Home Screen – It consists of six tabs each for a different phase of a project life cycle.
    a. Requirement tab – Business analysts would view, insert, update, delete and review requirements.
    b. Design tab – Developers would store the design documents, JAD session recording link, unit test results, edit, delete them and get the design documents reviewed.
    c. Test Suite tab – QA team would write the test cases, edit, delete them and have the test cases and expected results reviewed.

d. Test Execution tab – QA team would pull the test cases and organize them into test sets, execute the TCERs as the code is delivered, and begin testing the code only after receiving the Unit Test Results from development team.

e. Defect tab – QA would open the defects tab if any test case fails; then the defect would finish the defect life cycle in this tab.

f. Reports tab – this tab would give updated information about the project status at the click of a button.

4. Logout Screen – This screen will be displayed after the user hits the log out button and will give an option to re-log-in.

3.3. Functional Overview of Project Quality Tool

This section explains in detail about the Project Quality Tool's functionality. The users would log-in to the application with their assigned Username and Password. The system would take the user to the Project Selection screen, only if the password matches the stored password in the database. If the password is incorrect, it throws an error on the screen to re-enter the correct one. Figure 2 below shows the log-in screen snapshot.

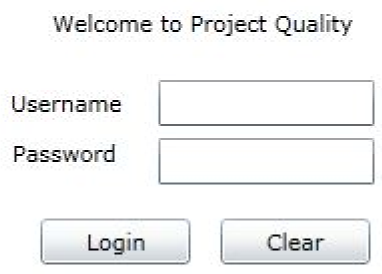


Figure 2. Log-in Screen

The Project Selection page would allow the user to choose the desired project that he/she wants to log-in to. The user can be part of multiple projects, thus, this option would allow the user work on each of them without mixing the requirements and other subsequent phases. Figure 3 below shows the project selection screen snapshot.

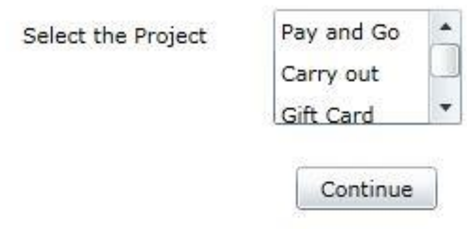


Figure 3. Project Selection Screen

The Home Screen is the main page of the application. It hosts the entire project life cycle details in one place. The home screen is organized into several tabs as discussed in section 3.2.3. Figure 4 below shows the home screen.

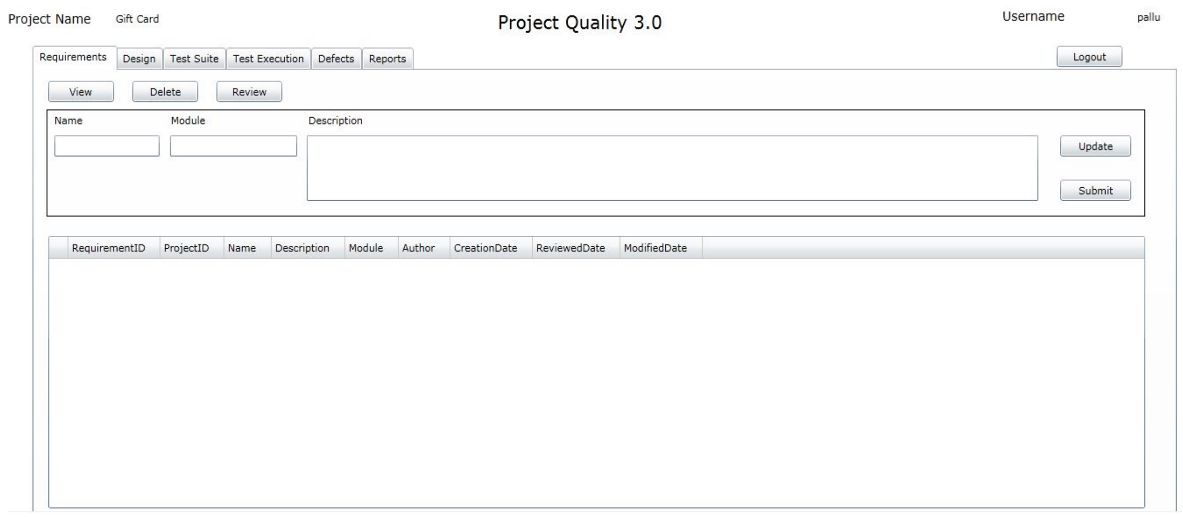


Figure 4. Home Screen

The Home Screen Requirements tab would hold details about the specific requirements. Here, the business analyst can add the requirements to the selected project by filling in the Requirement Name, Module and Description, and then click on Submit to insert a new requirement. Figure 5 below shows the Requirements Submit Button.

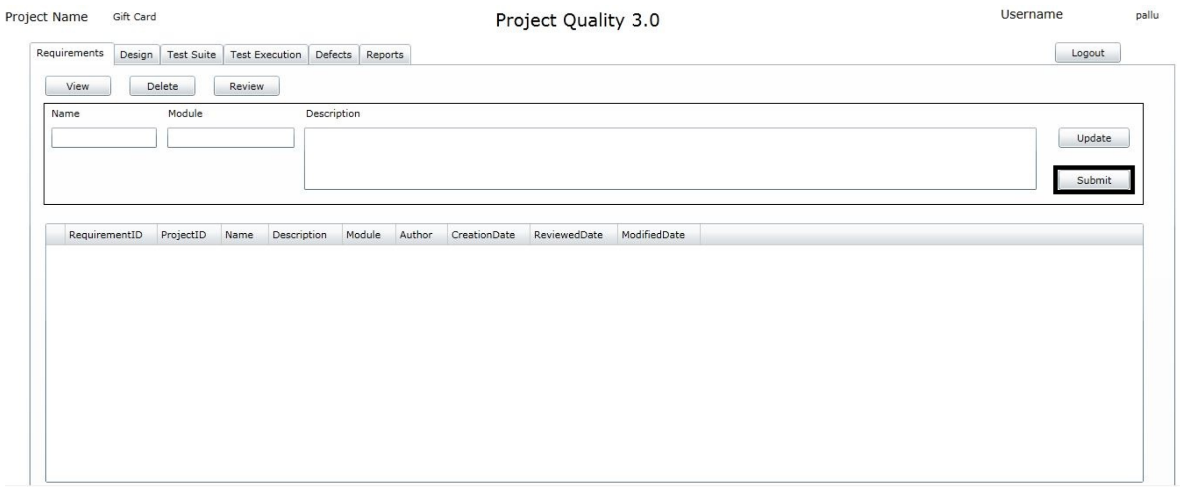


Figure 5. Requirements Submit Button

Once the requirements are added, the screen gets refreshed in the data grid. Figure 6 below shows the Requirements Data Grid.

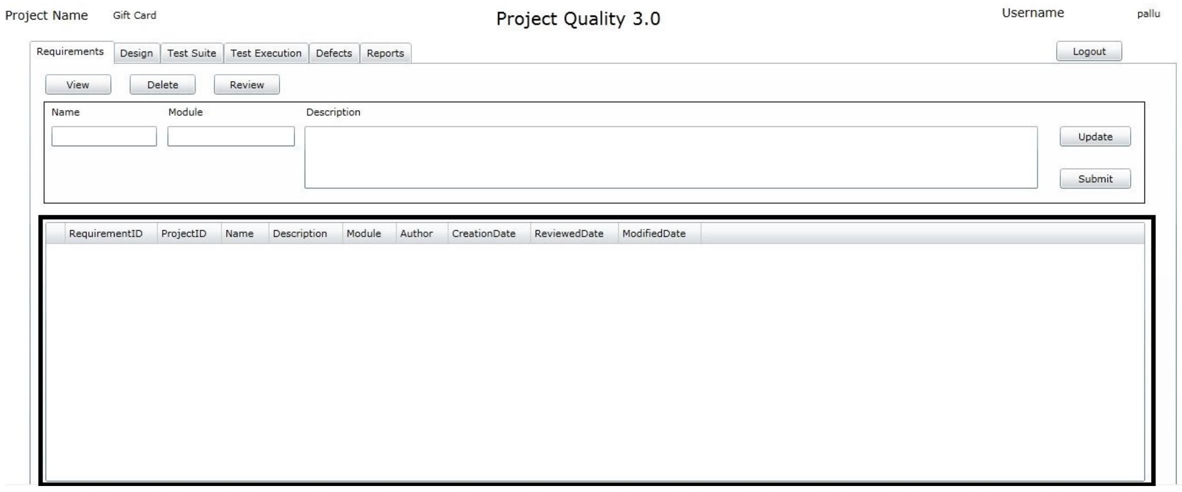


Figure 6. Requirements Data Grid

Anyone who logs into a particular project can view the requirements by clicking the View button on the Requirements tab. We can delete a requirement by selecting a particular requirement, and pressing the Delete button on the Requirement tab. In the same way, we can update a requirement by selecting the Required Requirement, which would populate the text in respective text boxes above the data grid. The business analyst can make the desired updates to the requirements, and then click on the Update button to incorporate the changes into the database. Figure 7 below shows the Options and Update button in the requirements tab.

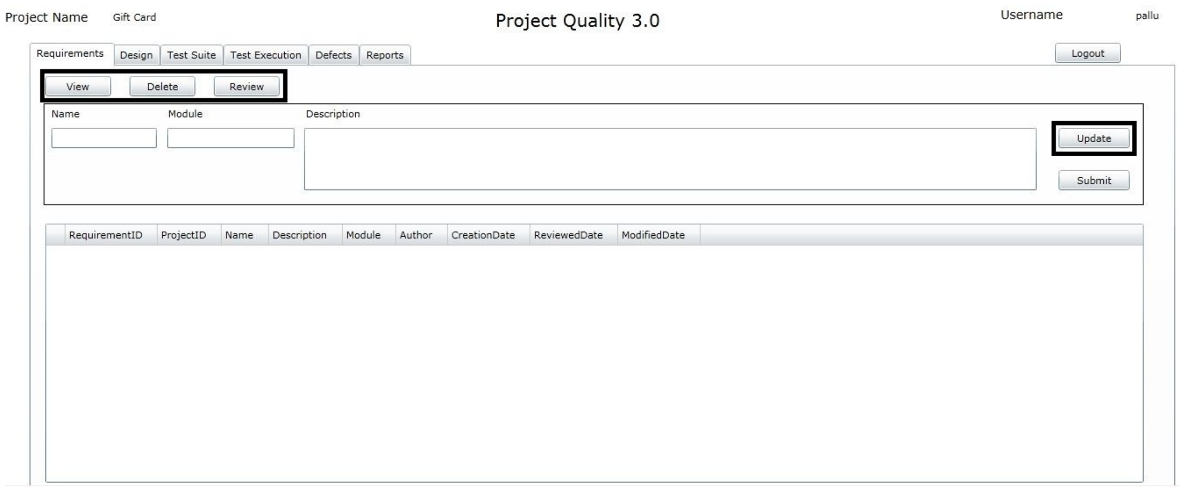
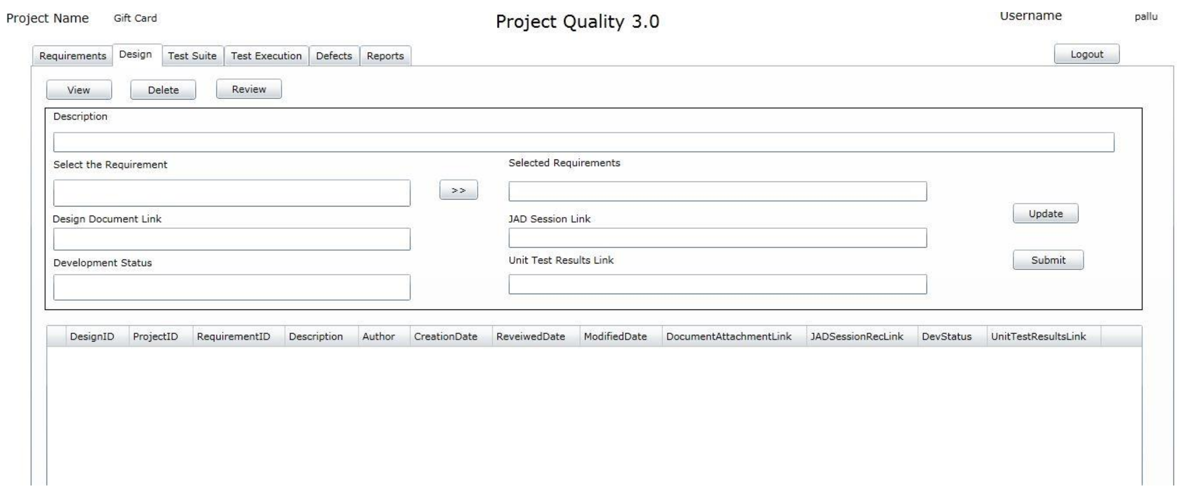


Figure 7. Options and Update Button in Requirements Tab

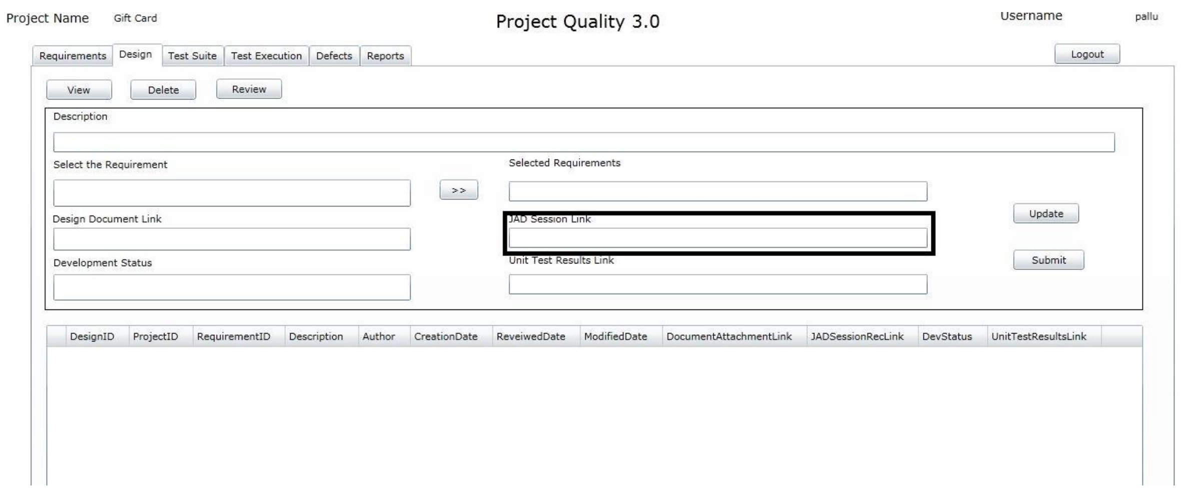
In the Home Screen Design tab, the developers can store the design document details. To insert the new design document details, the developer has to enter the description, select the requirements that this particular design document is mapped to, and enter the design document link in the section highlighted below. Figure 8 below shows the Design tab.



DesignID	ProjectID	RequirementID	Description	Author	CreationDate	ReveiwedDate	ModifiedDate	DocumentAttachmentLink	JADSessionRecLink	DevStatus	UnitTestResultsLink

Figure 8. Design Tab

An additional feature that I believe is very valuable to the project is the JAD session recording link. It is always a challenge to remember what all has been discussed during the JAD sessions. Sometimes, small things discussed during the JAD session can be very important, and have slipped during the course of development. Furthermore, people who were present at the time of the JAD session discussions may leave early. So, it is always important to store these JAD sessions, as they can add value to the knowledge transfer. It can also be helpful for the Quality Assurance team to review these JAD session recordings together to talk through the application flow. As each module will be tested by different QA personnel, if they know how they integrate with other modules even before doing the integration testing, then they would find the defects related to the functional flow very quickly in the Project Life Cycle. Figure 9 below shows the JAD session link.

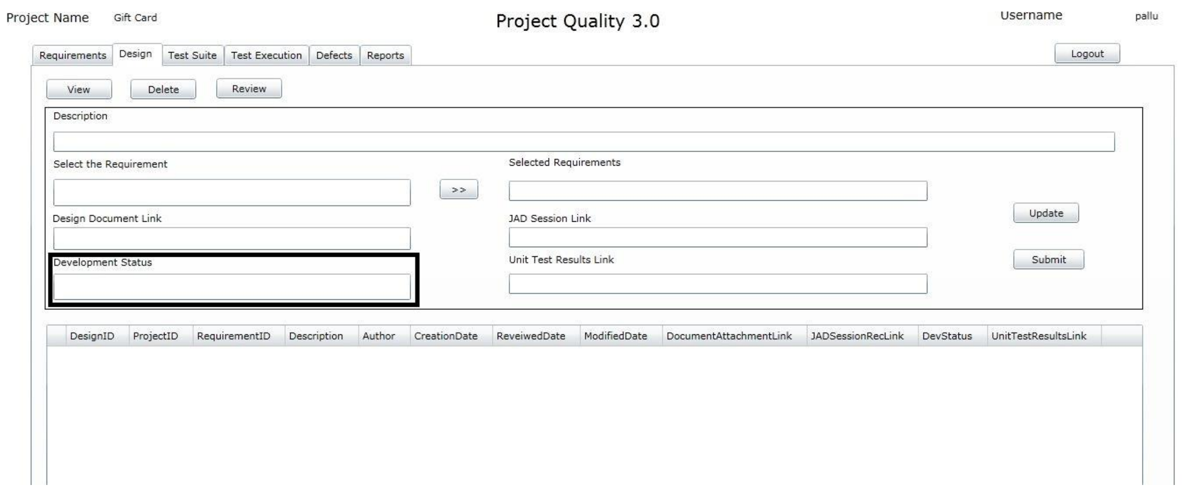

Project Name Gift Card

Project Quality 3.0

Username pallu

Figure 9. JAD Session Link

Development status is a drop-down status box that would be updated by developers to report the progress of the development phase to the project team. This would benefit the QA team in planning their testing efforts and resource allocations. Figure 10 below shows the Development Status dropdown.

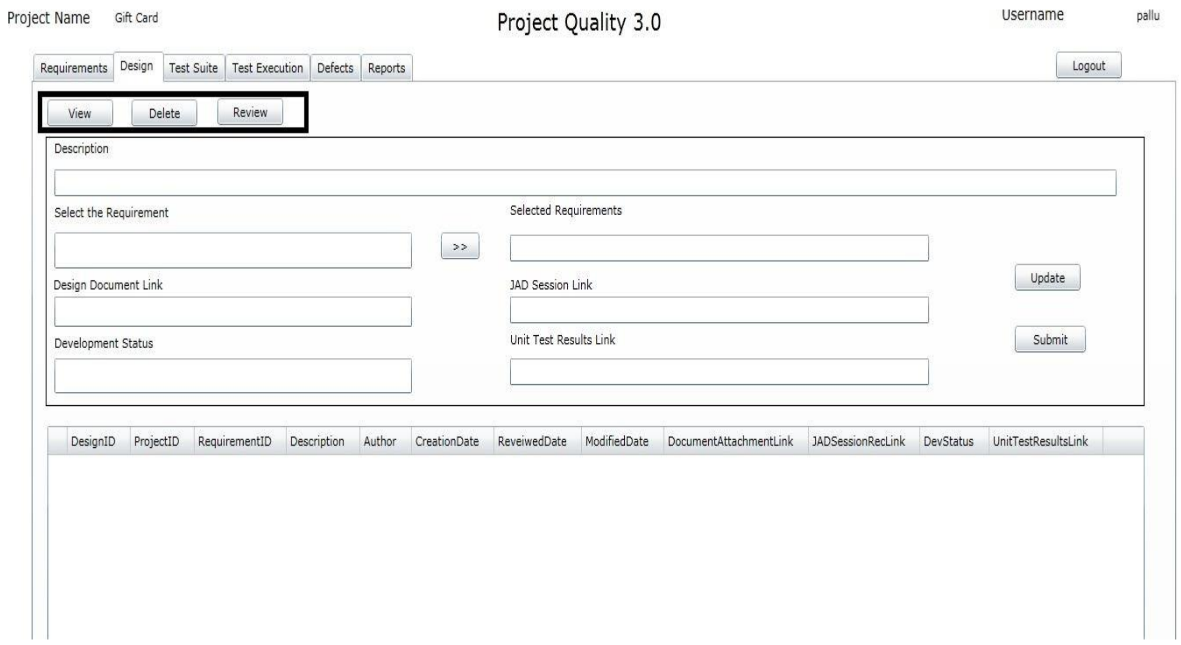

Project Name Gift Card

Project Quality 3.0

Username pallu

Figure 10. Development Status Dropdown

Another interesting and important place holder for developer’s tasks in the Design tab is the Unit Test Results link. Oftentimes, no one from the development team is performing the Unit Testing; they are just throwing the build over the wall, thinking that testing is the Tester’s responsibility. I disagree with this because testing is the tester’s job – just to make sure the requirements are met and business scenarios are working correctly. Without unit testing, the quality of the code will be very poor, and QA would not get that far in measuring the quality 100%. They would spend most of their time in finding Unit Test defects, and they may end up doing less system testing. Just like in the Requirements tab, here developers can view, insert, update and delete in order to perform all the necessary tasks based on their responsibility. Figure 11 below shows the View, Delete and Review buttons in the Design tab.



Project Name Gift Card

Username pallu

Figure 11. Options in Design Tab

Every time a new record is inserted, the design data grid gets refreshed. Figure 12 shows the Design Data Grid snapshot.

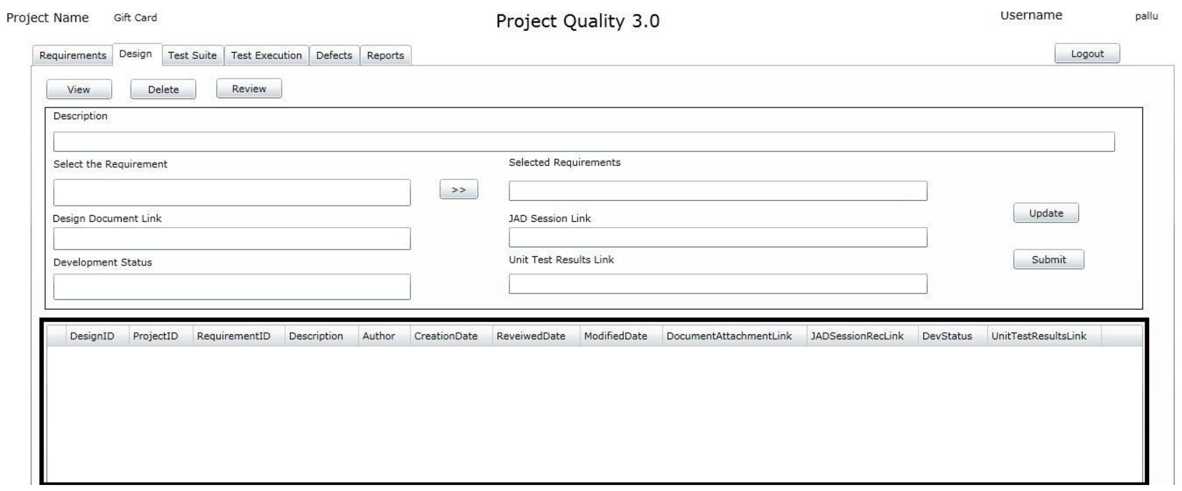


Figure 12. Design Data Grid

The Test Suite tab is for the QA team to create test cases and expected results. Here, they type the test case name, description, and expected result, then select the design document name to which this TCER is associated, and press the Submit button to insert the data into the database. After that, the changes made will get refreshed in the Test Suite data grid. Figure 13 below shows the Test Suite snapshot screen.

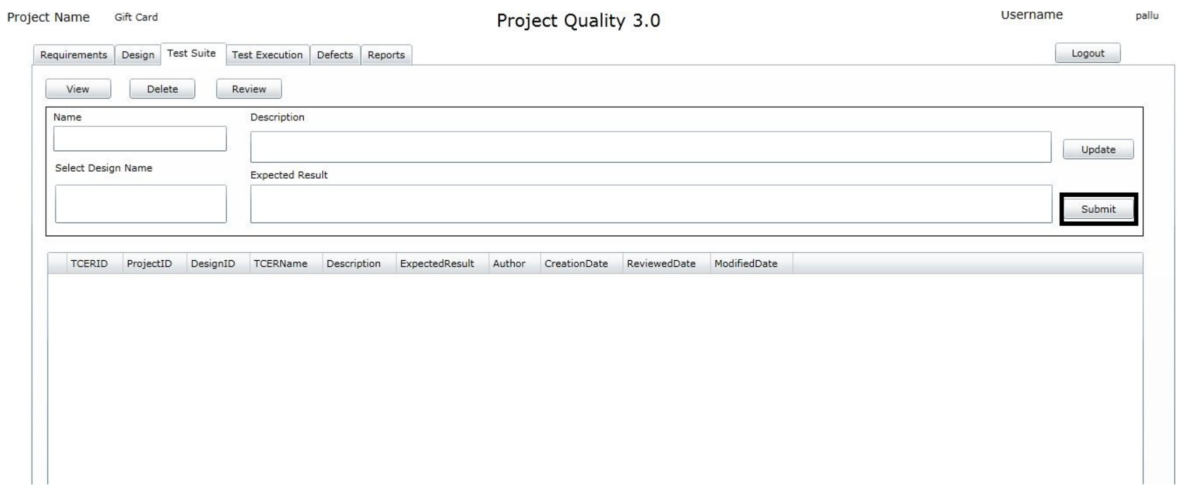


Figure 13. Test Suite Tab

The tester can view, insert, update and delete the test cases. Once the test case is inserted, the Test Suite Data Grid gets refreshed, and displays the newly inserted test case and expected results details. Figure 14 below shows a snapshot of the Test Suite tab options with View, Delete and Review buttons.

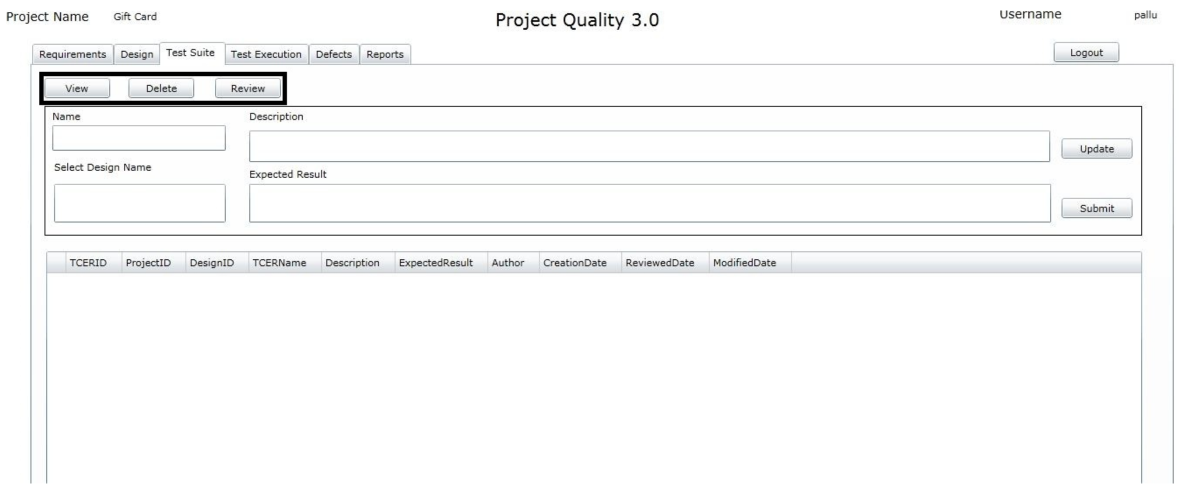


Figure 14. Options in Test Suite Tab

If the tester has to update any test case and expected results, he/she needs to select the test case that they would like to edit. Upon selection, the text boxes will be filled with the selected content, and the test case can be modified. The Update button can be clicked to submit the changes to the database. Figure 15 below shows a snapshot of the Test Suite tab with the Update button.

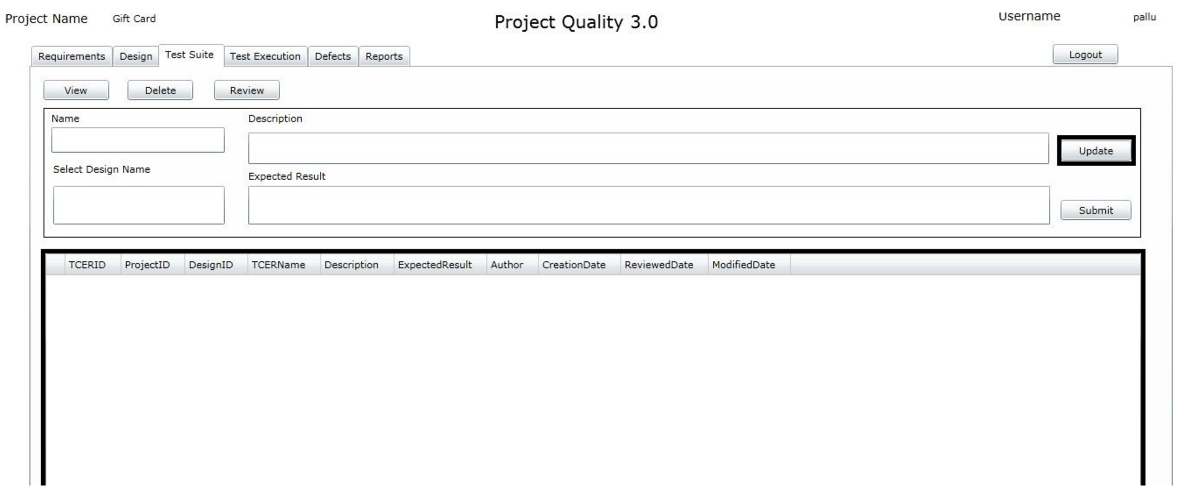


Figure 15. Update Button in Test Suite Tab

The Test Execution tab contains the details about the testing progress. Testers will create a test set name and description, selects the test case, and then hit Submit to add the TCER to the test set. Testers can organize the TCERs before they actually begin testing. Figure 16 below shows the screen snapshot of the Test Execution tab.

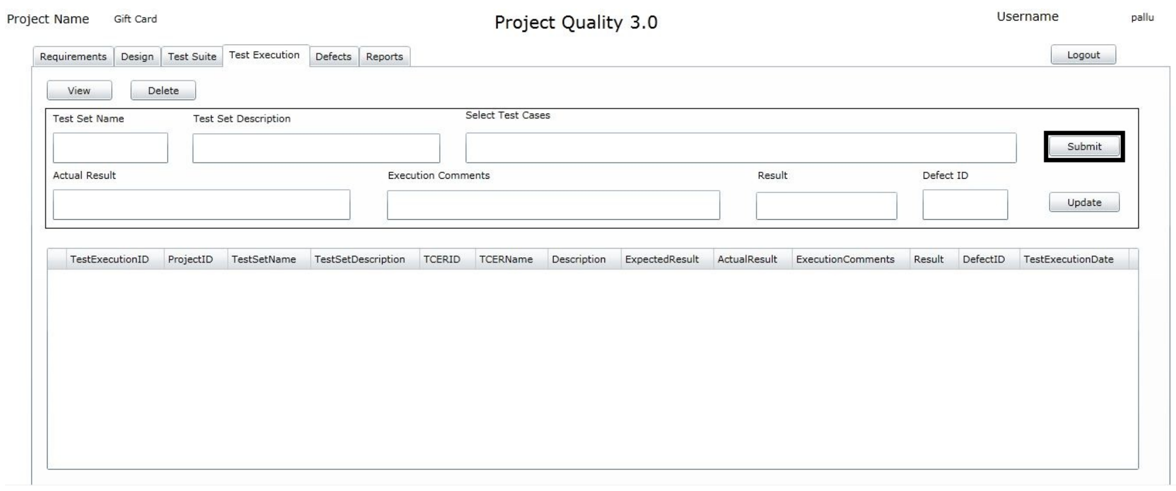
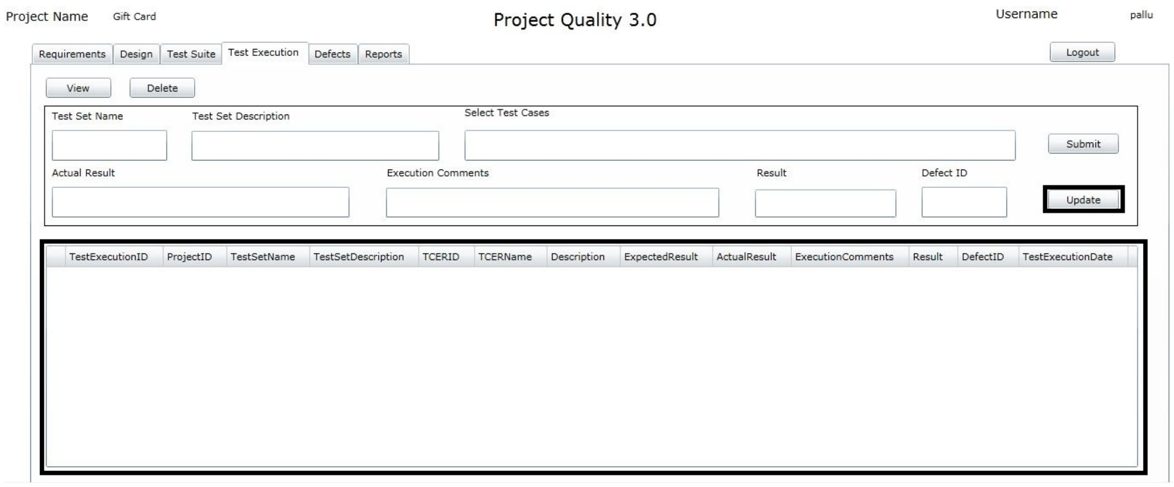


Figure 16. Test Execution Tab

Later, when they are ready for execution they can just select the desired test case, perform the necessary test execution steps, update the Actual Result, and write some execution comments as needed; they can also specify pass or fail results and if the TCER failed, a defect can be opened in the Defect tab and the Defect ID can be updated here in the Test Execution tab. Once the Update button is clicked, the data gets refreshed in the test execution data grid to show the latest database instance. Figure 17 below shows a screen snapshot of the Test Execution data grid.



TestExecutionID	ProjectID	TestSetName	TestSetDescription	TCERID	TCERName	Description	ExpectedResult	ActualResult	ExecutionComments	Result	DefectID	TestExecutionDate

Figure 17. Test Execution Data Grid

The Defects tab on the home screen is all about creating a defect when a test case failed in any testing phase. The defect would finish its life cycle in this tab as the changes occur. While creating a defect, the tester would specify the summary of the defect, description, select the Detected by from the drop down, select the assigned to from the drop down, specify the phase, severity, priority, status, select if it is reproducible and hit the submit button to create a defect. Figure 18 below shows a screen snapshot of the Defects tab.

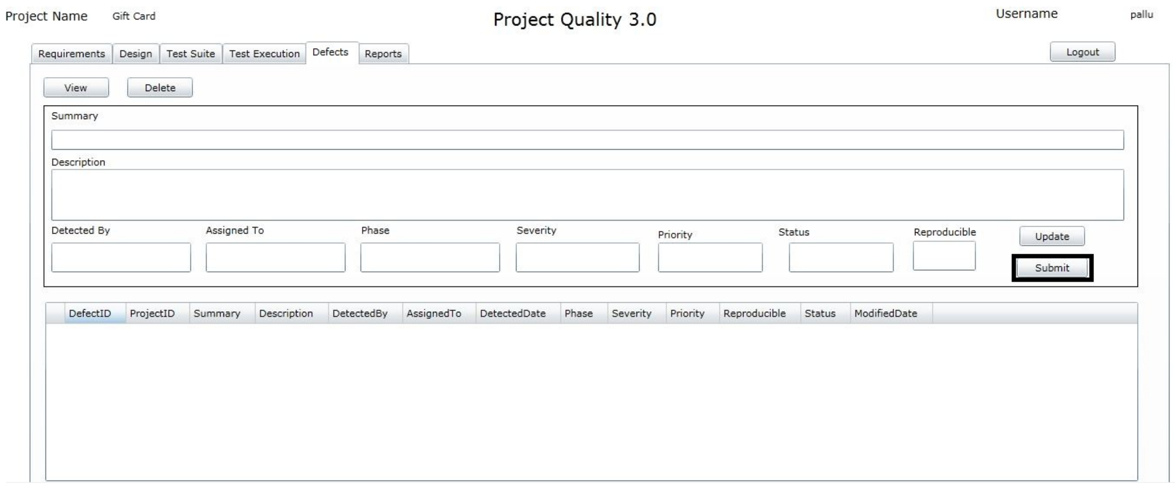


Figure 18. Defects Tab

When a defect is inserted, the data grid would be refreshed, and the new defect inserted will be displayed in the defect data grid. A defect can also be updated by the developers; they can change the defect status, and add some additional content to the description test box to add comments. As soon as the defect is updated, the data in the defects data grid will get refreshed with the new updated data. Figure 19 below shows the screen snapshot of the Update button in the Defects tab.

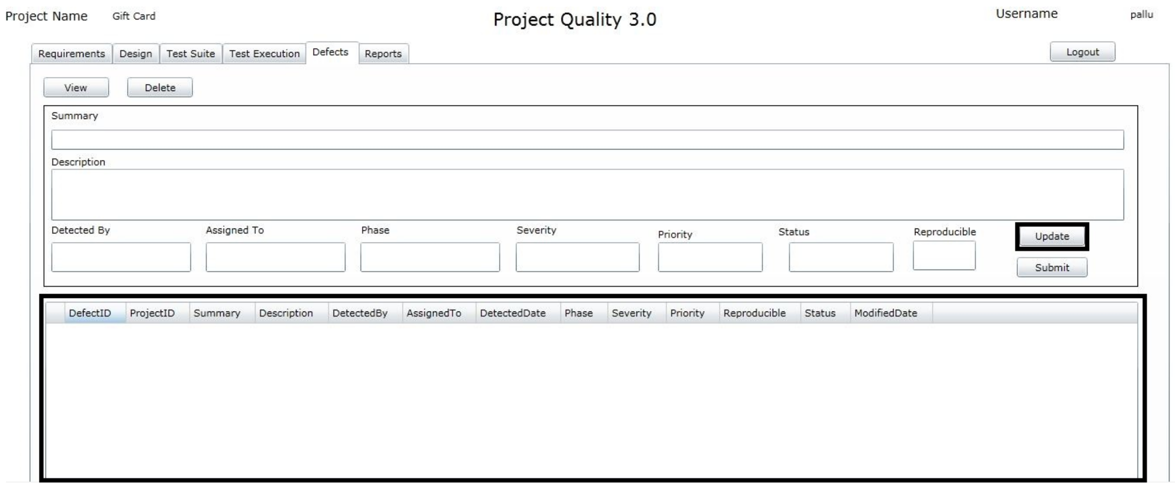


Figure 19. Update Button in Defects Tab

The Reports tab is used to generate a report at the click of a button for any particular project. Figure 20 below shows a screen snapshot of the Reports tab.

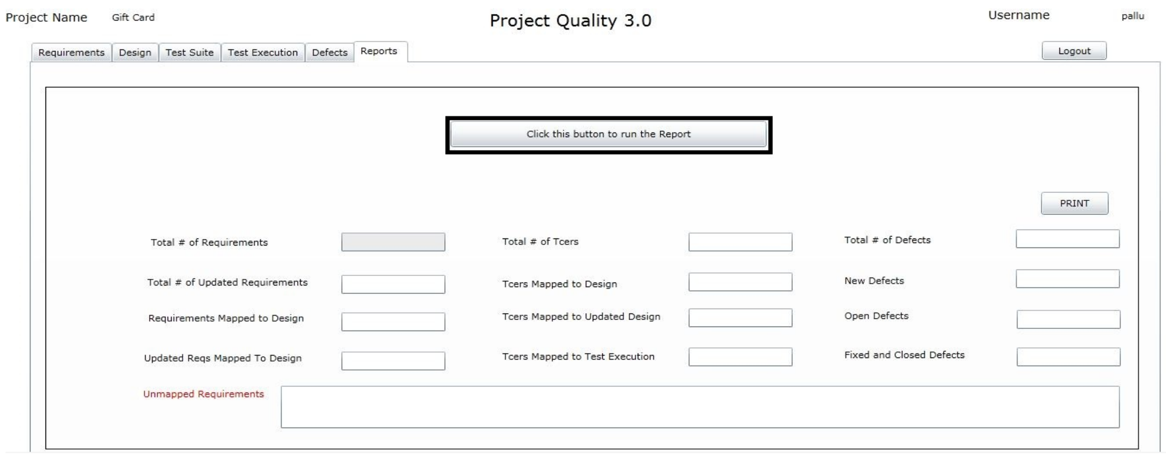


Figure 20. Reports Tab

A Report helps measure the Project progress and is very useful for the project team when a quick snapshot is needed. An explanation of the Reports generation tool categories is listed below.

1. Total Number of Requirements – This field gives the total number of requirements in a selected project.
2. Total Number of Updated Requirements – This field shows the total number of updated requirements in the selected project.
3. Requirements Mapped to Design – Technically, all the requirements in the (1) should be in this field.
4. Updated Requirements Mapped to Design – All the requirements in the (2) should be in this field.

5. Unmapped Requirements – If there are any requirements that are not mapped to any design they will be specified comma separated in this field. The Project Manager or the Leads can delegate them as Action Items and get them addressed by the project team.
6. Total number of TCERs – This field holds the total number of Test Cases and Expected Results created in the selected project.
7. TCERs Mapped to Design – The Test cases would be written off of the Design document and keeping Requirements in mind. This field shows the number of TCERs that are mapped to the Design documents.
8. TCERs Mapped to Updated Design – This field shows the number of TCERs that are mapped to the Updated Design documents.
9. TCERs Mapped to Test Execution – The TCERs that were created in the Test Suite tab should be pulled into the Test Execution tab for execution. The total number of TCERs pulled to test execution should be greater than or equal to TCERs.
10. Total Number of Defects – This field tells the number of defects created in this particular project.
11. New Defects – The defects that are created will have different stages before they finish their lifecycle. This field shows the "Newly Created" defects.
12. Open Defects – This field shows the defects that have been in the open status.
13. Fixed and Closed Defects – This field shows the defects that have been in fixed or closed status.

3.4. Project Quality Tool Architecture

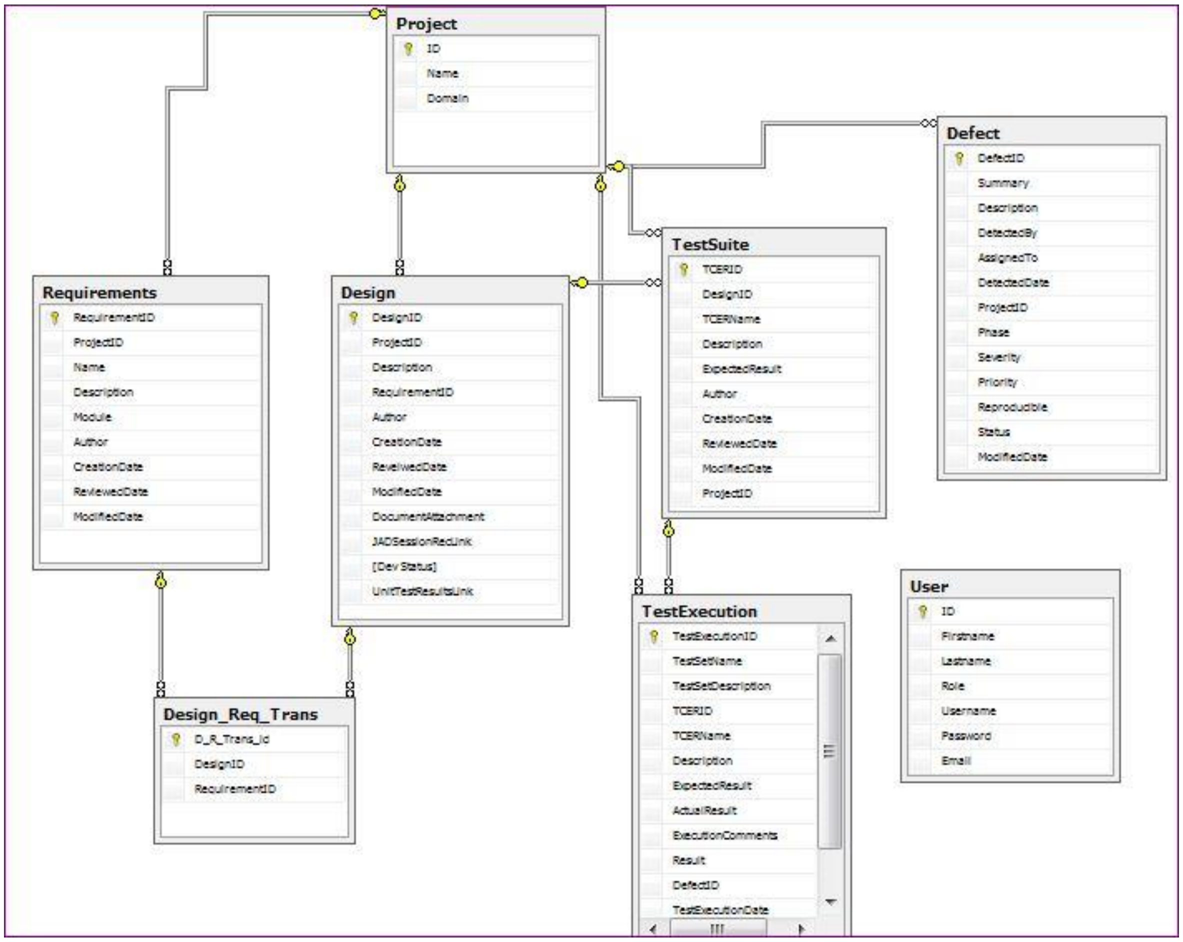


Figure 21. Project Quality Tool Database Diagram

3.5. Flow Charts of the Project Functionality

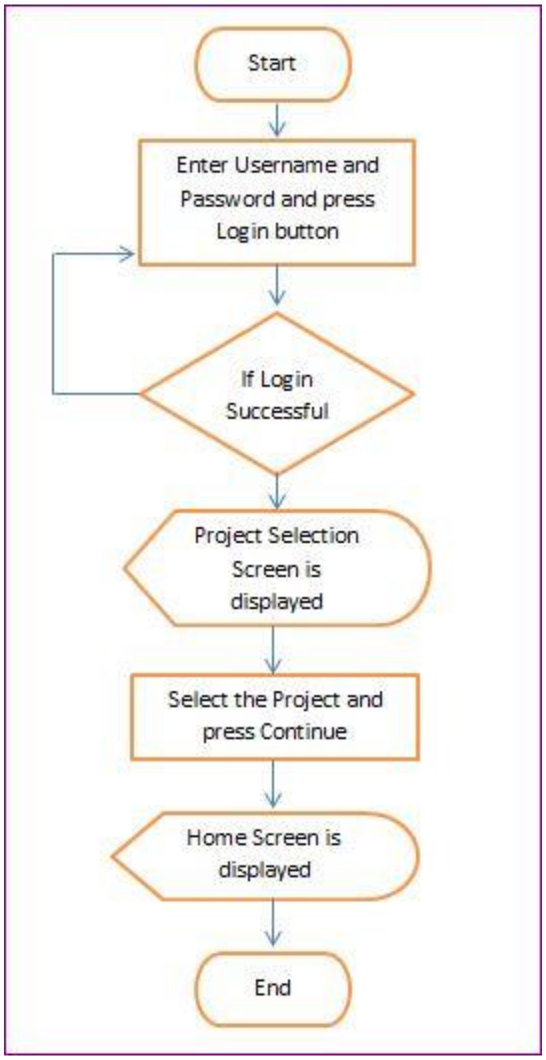

Start
Enter Username and Password and press Login button
If Login Successful
Project Selection Screen is displayed
Select the Project and press Continue
Home Screen is displayed
End

Figure 22. Flow Chart of User Logging In and landing in project Home Screen Page

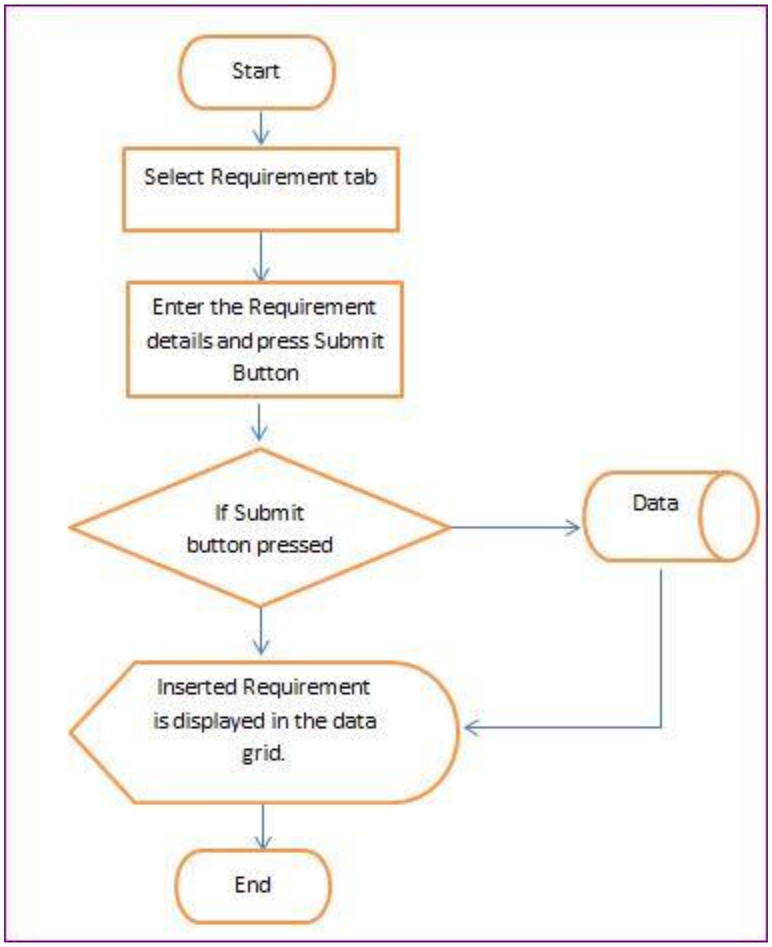

Start
Select Requirement tab
Enter the Requirement details and press Submit Button
If Submit button pressed
Data
Inserted Requirement is displayed in the data grid.
End

Figure 23. Flow Chart of Inserting Requirements in a Selected Project

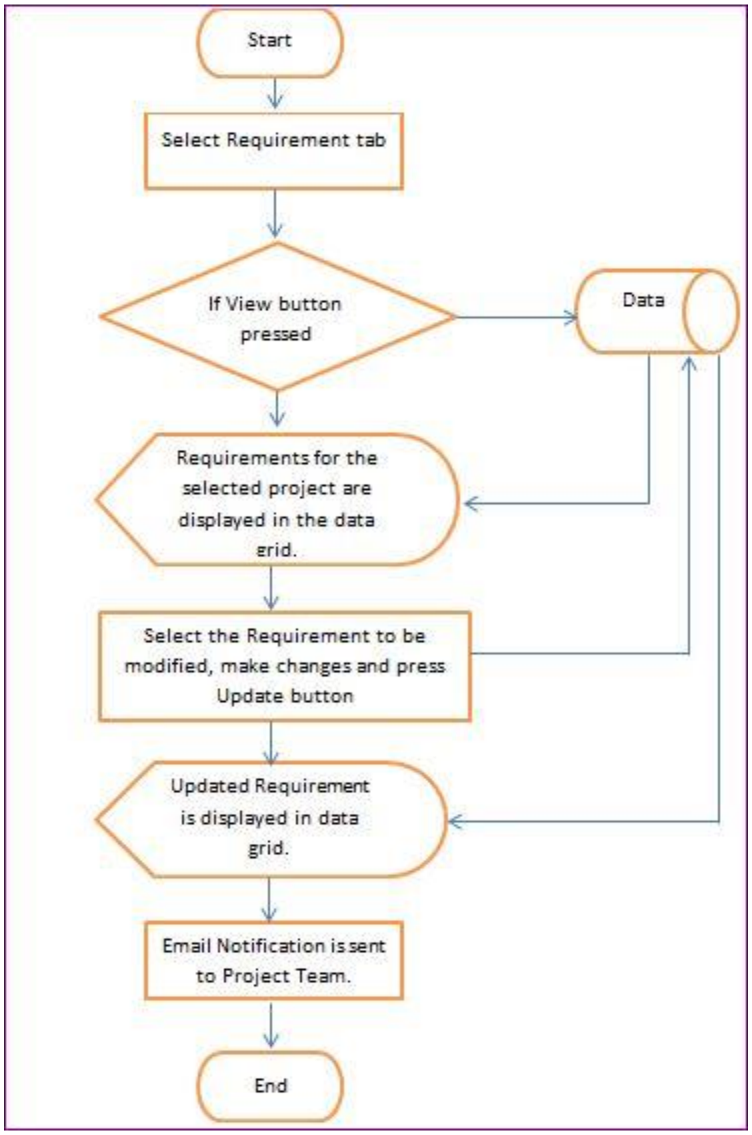

Start
Select Requirement tab
If View button pressed
Data
Requirements for the selected project are displayed in the data grid.
Select the Requirement to be modified, make changes and press Update button
Updated Requirement is displayed in data grid.
Email Notification is sent to Project Team.
End

Figure 24. Flow Chart of Updating Requirements and Project Team Email Notification

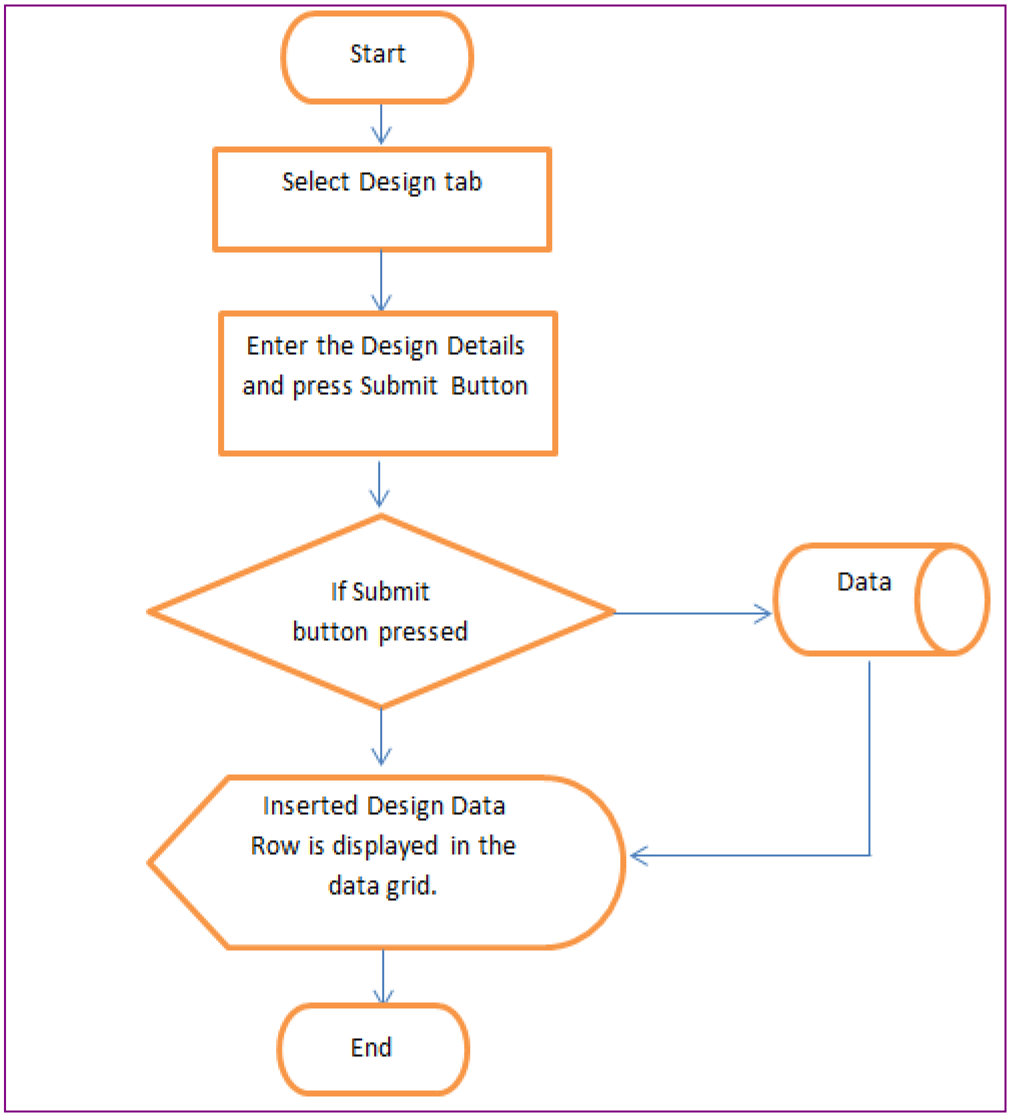

Start
Select Design tab
Enter the Design Details and press Submit Button
If Submit button pressed
Data
Inserted Design Data Row is displayed in the data grid.
End

Figure 25. Flow Chart of Inserting Design Document and Design Phase Details in a Project

CHAPTER 4. RESEARCH EVALUATION

In this chapter, the value of the Project Quality Tool is evaluated. To do this, a sample experiment is presented along with a sample project called the “Gift Card.” This will demonstrate how the Project Quality Tool will help in documenting and tracking the requirements, acting as a design documents repository, retaining JAD sessions forever, collecting Unit Test results, which gives us an ability to trace the requirements to the Test Cases and Expected Results and track the TCERs’ execution and defect management, giving a snapshot of the project status using reports. Table 1 below shows a summary of the experiment.

Table 1

Experiment Project Quality Tool versus Other Tools Mentioned in Chapter 2

Project	**# of Req**	**# of Changed Req (m)**	**# of Req correctly communicated (n)**	**# of Req miscommunicated**	**Traceability from Req to Design and Design to Test Cases**	**Criticality to Project**	**% of Project not Delivered**	**% of Project Delivered (p)**
With Tools Mentioned in Chapter 2 Literature Review								
1	30	3	0	3	N	Y	10%	90%
2	10	3	1	2	N	Y	20%	80%
3	20	5	0	5	N	Y	25%	75%
With Project Quality Tool								
1	30	3	3	0	Y	Y	0%	100%
2	10	3	3	0	Y	Y	0%	100%
3	20	5	5	0	Y	Y	0%	100%

If (m-n) = 0 then p will be 100%,

if (m-n) ≠ 0 then p will not be 100%

In the above experiment, if the difference between # of changed requirements and # of correctly communicated requirements is zero then the % of project delivered will be 100%. If the difference between # of changed requirements and # of correctly communicated requirements is not zero, then the % of project delivered will not be 100%.

Below is the sample project that I have set-up to show how Project Quality Tool helps in every phase of the Software Development Life Cycle, as well as how the in-built traceability would help communicate any change in requirements after the project is in-flight.

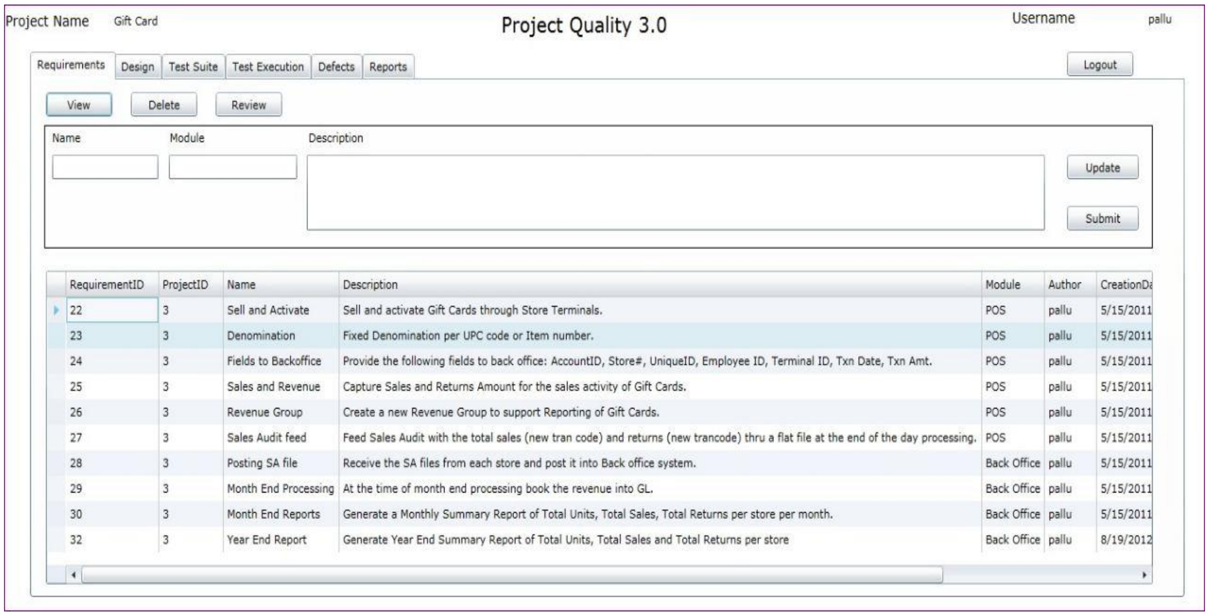
There is a retail grocery chain called 'XYZ MART' and it would like to introduce the Gift Card Program into their business before the holiday season. The main purpose of this project is to sell gift cards and allow Gift Card Redemption. For now, we will be concentrating on gift card sales, and not the redemptions. Below are the Requirements for the Gift Card Project:

1. Sell and Activate Gift Cards through Store Terminals.
2. Fixed denomination per UPC code/ Item Number.
3. Provide the following fields to back office: Account ID, Store #, Unique ID, Employee ID, Terminal ID, Transaction Date, Transaction Amount and Item Number.
4. Capture both Sales and Returns amount activity of the Gift Cards.
5. Create new revenue group to support reporting of Gift Cards.
6. Feed Sales Audit with the Total Sales (new transaction code) and returns (new transaction code) through a flat file at the end-of-the-day processing.
7. Receive the Sales Audit files from each store and post it into back office system.
8. At the time of month-end processing, book the Revenue into the General Ledger.

9. Generate a Monthly Summary Report of Total Units Sold, Total Sales Amount, Total Units Returned, Total Return Amount, per Store.

10. Generate Year-End Reports of Total Units Sold, Total Sales Amount, Total Units Returned, Total Returned Amount, per Store.

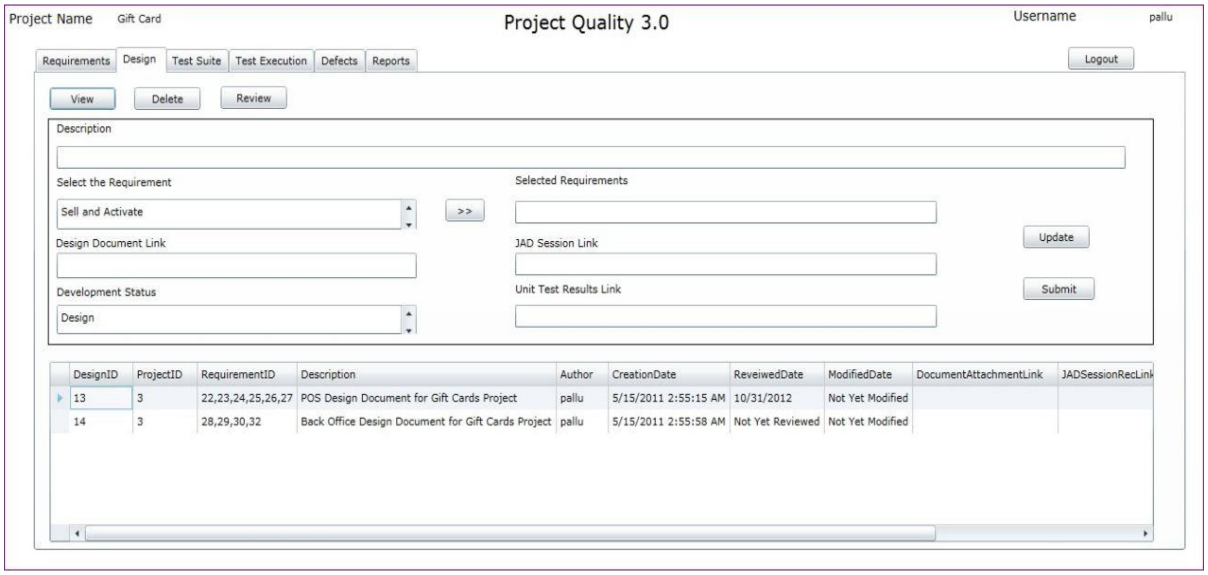
In the Requirements tab, the business analyst would create the requirements and have them reviewed by the business and IT partners. Figure 26 shows the Requirements tab snapshot from the tool.



RequirementID	ProjectID	Name	Description	Module	Author	CreationDa
22	3	Sell and Activate	Sell and activate Gift Cards through Store Terminals.	POS	pallu	5/15/2011
23	3	Denomination	Fixed Denomination per UPC code or Item number.	POS	pallu	5/15/2011
24	3	Fields to Backoffice	Provide the following fields to back office: AccountID, Store#, UniqueID, Employee ID, Terminal ID, Txn Date, Txn Amt.	POS	pallu	5/15/2011
25	3	Sales and Revenue	Capture Sales and Returns Amount for the sales activity of Gift Cards.	POS	pallu	5/15/2011
26	3	Revenue Group	Create a new Revenue Group to support Reporting of Gift Cards.	POS	pallu	5/15/2011
27	3	Sales Audit feed	Feed Sales Audit with the total sales (new tran code) and returns (new trancode) thru a flat file at the end of the day processing.	POS	pallu	5/15/2011
28	3	Posting SA file	Receive the SA files from each store and post it into Back office system.	Back Office	pallu	5/15/2011
29	3	Month End Processing	At the time of month end processing book the revenue into GL.	Back Office	pallu	5/15/2011
30	3	Month End Reports	Generate a Monthly Summary Report of Total Units, Total Sales, Total Returns per store per month.	Back Office	pallu	5/15/2011
32	3	Year End Report	Generate Year End Summary Report of Total Units, Total Sales and Total Returns per store	Back Office	pallu	8/19/2012

Figure 26. Sample Gift Card Project Requirements in Requirements Tab

After the requirement review is completed, the developers would work on designing the application. They would divide the requirements modules based on technology or functional area after brain-storming in the JAD sessions. The developers/development leads would share the design document for review by rest of the project team. For this step, he/she would be inserting the name, description, select the requirements that this design is mapped to and specify path of design document, specify the development status and JAD session recording into the details and hit submit to insert the record. They would get the design document reviewed, and proceed with the development effort.



Project Name: Gift Card
Username: pallu

Description:

Select the Requirement: Sell and Activate

Selected Requirements:

Design Document Link:

JAD Session Link:

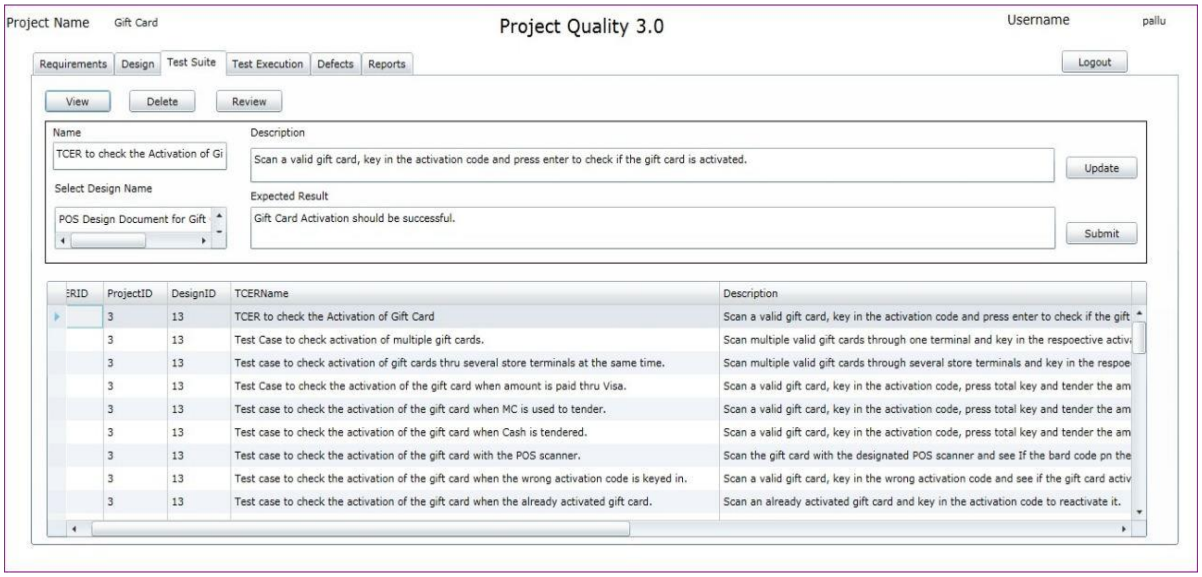
Development Status: Design

Unit Test Results Link:

DesignID	ProjectID	RequirementID	Description	Author	CreationDate	ReveiwedDate	ModifiedDate	DocumentAttachmentLink	JADSessionRecLink
13	3	22,23,24,25,26,27	POS Design Document for Gift Cards Project	pallu	5/15/2011 2:55:15 AM	10/31/2012	Not Yet Modified		
14	3	28,29,30,32	Back Office Design Document for Gift Cards Project	pallu	5/15/2011 2:55:58 AM	Not Yet Reviewed	Not Yet Modified		

Figure 27. Sample Gift Card Project Design Tab

Once the design has been reviewed, Quality Assurance would start writing the TCERs, and kick-start the test planning in the Test Suite tab.



Project Name: Gift Card
Username: pallu

Name: TCER to check the Activation of Gi

Description: Scan a valid gift card, key in the activation code and press enter to check if the gift card is activated.

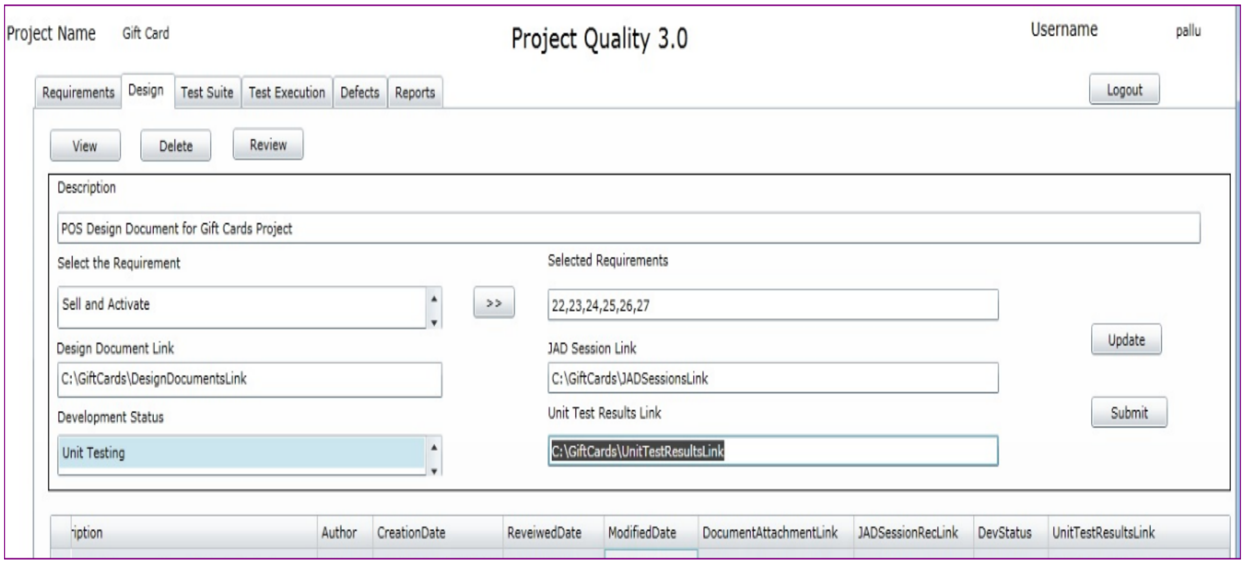
Select Design Name: POS Design Document for Gift

Expected Result: Gift Card Activation should be successful.

ERID	ProjectID	DesignID	TCERName	Description
	3	13	TCER to check the Activation of Gift Card	Scan a valid gift card, key in the activation code and press enter to check if the gift
	3	13	Test Case to check activation of multiple gift cards.	Scan multiple valid gift cards through one terminal and key in the respoective activ
	3	13	Test case to check activation of gift cards thru several store terminals at the same time.	Scan multiple valid gift cards through several store terminals and key in the respoe
	3	13	Test Case to check the activation of the gift card when amount is paid thru Visa.	Scan a valid gift card, key in the activation code, press total key and tender the am
	3	13	Test case to check the activation of the gift card when MC is used to tender.	Scan a valid gift card, key in the activation code, press total key and tender the am
	3	13	Test case to check the activation of the gift card when Cash is tendered.	Scan a valid gift card, key in the activation code, press total key and tender the am
	3	13	Test case to check the activation of the gift card with the POS scanner.	Scan the gift card with the designated POS scanner and see If the bard code pn the
	3	13	Test case to check the activation of the gift card when the wrong activation code is keyed in.	Scan a valid gift card, key in the wrong activation code and see if the gift card activ
	3	13	Test case to check the activation of the gift card when the already activated gift card.	Scan an already activated gift card and key in the activation code to reactivate it.

Figure 28. Sample Gift Card Project Test Suite tab

As soon as the build becomes available, the developers should complete the unit testing and upload their Unit Test Results before turning over the build to QA.



Description: POS Design Document for Gift Cards Project

Select the Requirement: Sell and Activate

Selected Requirements: 22,23,24,25,26,27

Design Document Link: C:\GiftCards\DesignDocumentsLink

JAD Session Link: C:\GiftCards\JADSessionsLink

Development Status: Unit Testing

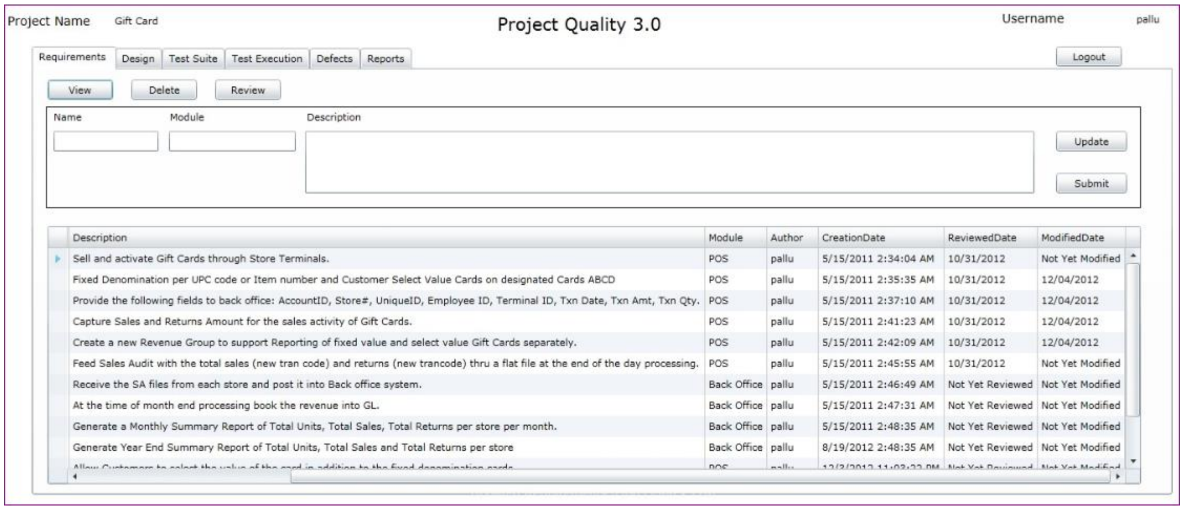
Unit Test Results Link: C:\GiftCards\UnitTestResultsLink

iption	Author	CreationDate	ReveiwedDate	ModifiedDate	DocumentAttachmentLink	JADSessionRecLink	DevStatus	UnitTestResultsLink

Figure 29. Sample Gift Card Project Unit Test Results Link

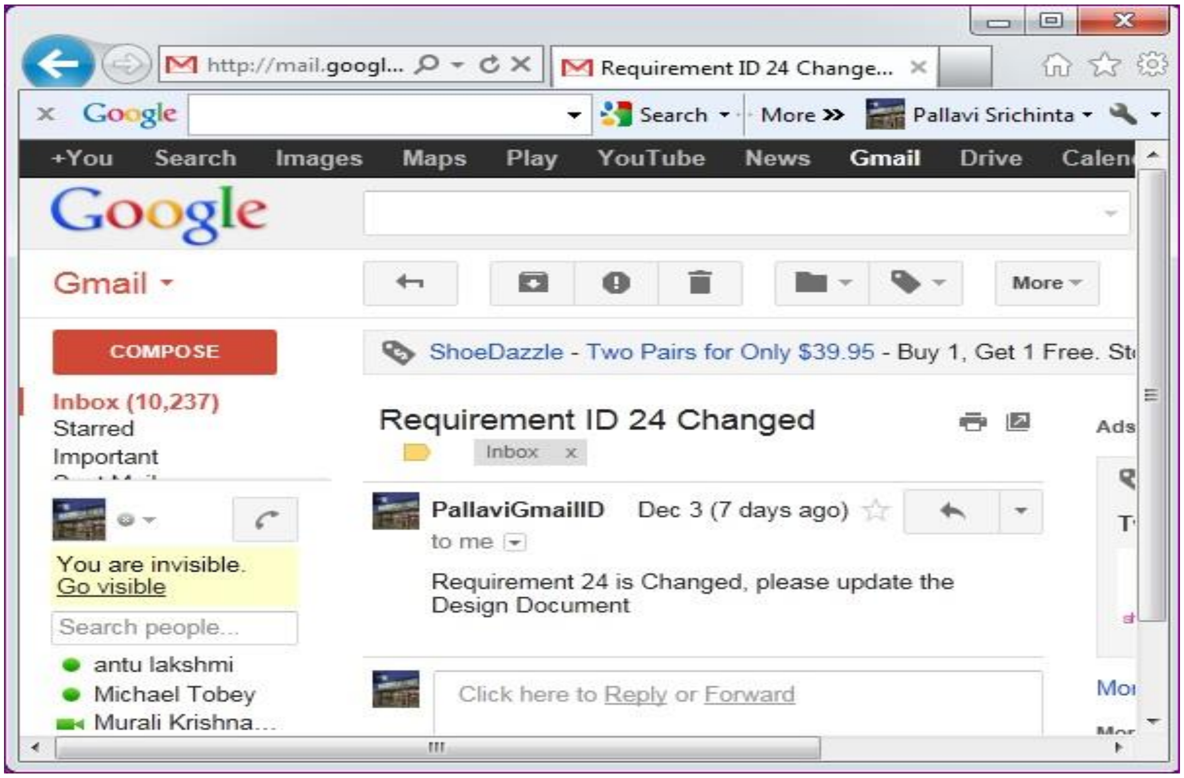
Quality Assurance would begin their System Testing after sanity checking the build. Note that this illustration is for an ideal environment with no change orders. Similarly, in our Gift Card Application, there was an additional requirement that is added i.e. to sell and activate non denominated gift cards. Now, the project is 80% complete; adding a new requirement at this stage would cause a lot of commotion, and there would be compromise in quality in order to accommodate this new requirement, and a wide communication gap could occur unnecessarily due to the last minute changes.

With the help of our Project Quality Tool, we can easily add this new requirement, or modify an existing requirement at any time during the project life cycle, and still be able keep track of this requirement change. The Tool will send out an email notification to the rest of the project team about the new requirements changes so that everybody in the project team would be aware of what this new requirement is, how this new requirement impacts their area, and how it needs to be addressed.



Description	Module	Author	CreationDate	ReviewedDate	ModifiedDate
Sell and activate Gift Cards through Store Terminals.	POS	pallu	5/15/2011 2:34:04 AM	10/31/2012	Not Yet Modified
Fixed Denomination per UPC code or Item number and Customer Select Value Cards on designated Cards ABCD	POS	pallu	5/15/2011 2:35:35 AM	10/31/2012	12/04/2012
Provide the following fields to back office: AccountID, Store#, UniqueID, Employee ID, Terminal ID, Txn Date, Txn Amt, Txn Qty.	POS	pallu	5/15/2011 2:37:10 AM	10/31/2012	12/04/2012
Capture Sales and Returns Amount for the sales activity of Gift Cards.	POS	pallu	5/15/2011 2:41:23 AM	10/31/2012	12/04/2012
Create a new Revenue Group to support Reporting of fixed value and select value Gift Cards separately.	POS	pallu	5/15/2011 2:42:09 AM	10/31/2012	12/04/2012
Feed Sales Audit with the total sales (new tran code) and returns (new trancode) thru a flat file at the end of the day processing.	POS	pallu	5/15/2011 2:45:55 AM	10/31/2012	Not Yet Modified
Receive the SA files from each store and post it into Back office system.	Back Office	pallu	5/15/2011 2:46:49 AM	Not Yet Reviewed	Not Yet Modified
At the time of month end processing book the revenue into GL.	Back Office	pallu	5/15/2011 2:47:31 AM	Not Yet Reviewed	Not Yet Modified
Generate a Monthly Summary Report of Total Units, Total Sales, Total Returns per store per month.	Back Office	pallu	5/15/2011 2:48:35 AM	Not Yet Reviewed	Not Yet Modified
Generate Year End Summary Report of Total Units, Total Sales and Total Returns per store	Back Office	pallu	8/19/2012 2:48:35 AM	Not Yet Reviewed	Not Yet Modified
Allow Customers to select the value of the card in addition to the fixed denomination cards.	POS	pallu	12/3/2012 11:03:22 PM	Not Yet Reviewed	Not Yet Modified

Figure 30. Sample Gift Card Project Changed Requirements



Requirement ID 24 Changed

PallaviGmailID Dec 3 (7 days ago)
to me

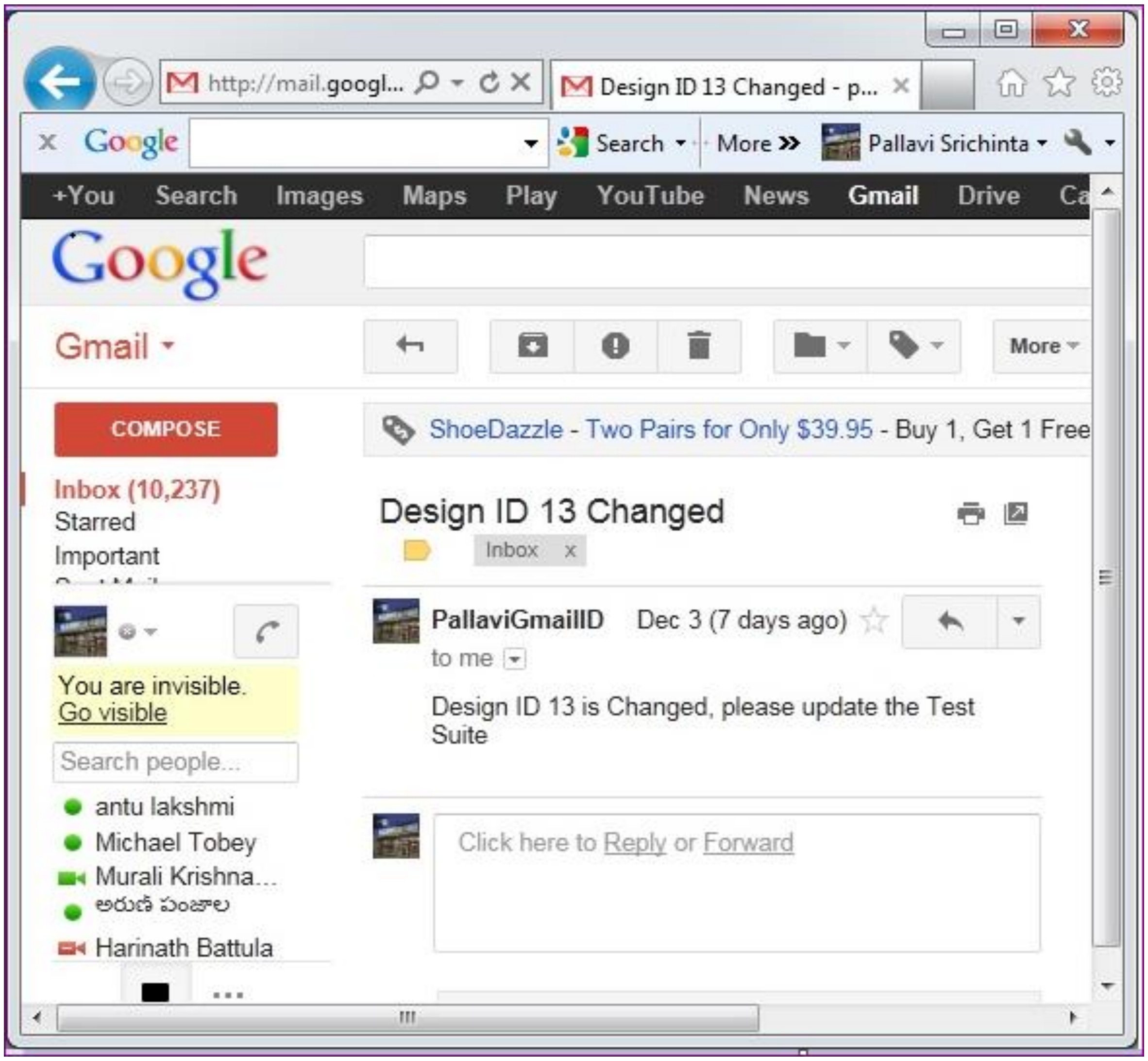
Requirement 24 is Changed, please update the Design Document

Click here to Reply or Forward

Figure 31. Sample Gift Card Project Requirement Change Email Notification

As the Project Quality Tool is capable of linking the requirements to design for traceability, any unmapped new requirements, or modified existing requirements would show up in the reports stating the problem area to the rest of the project team. This feature will help us to check the completeness of the project at any point in time.

Similarly, if a design document is updated after the TCERs are reviewed, then a notification will be sent to the project team stating that there is a change in the design document, and the subsequent phases may possibly be affected by these changes.



ShoeDazzle - Two Pairs for Only $39.95 - Buy 1, Get 1 Free

Design ID 13 Changed

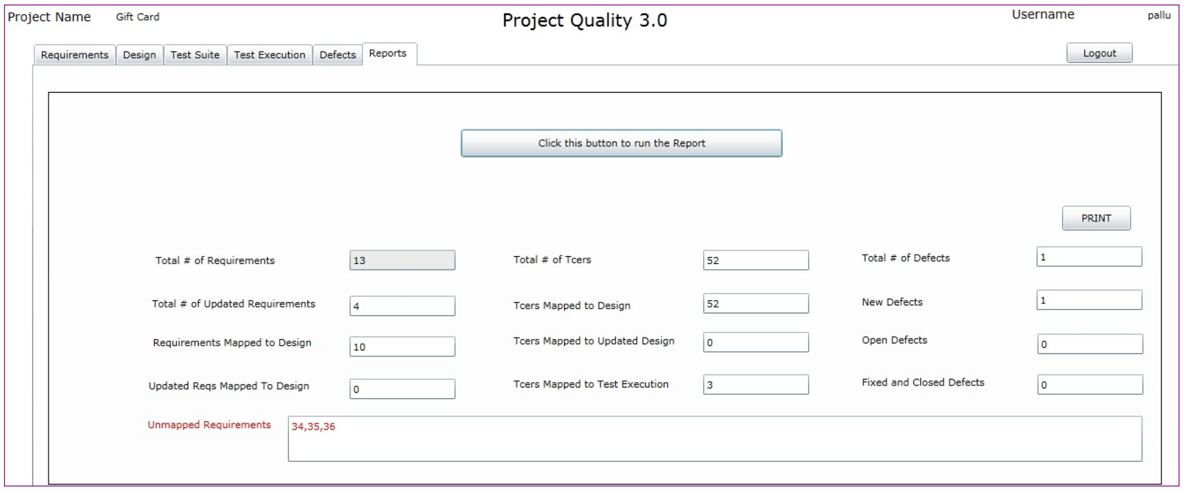
PallaviGmailID Dec 3 (7 days ago)

to me

Design ID 13 is Changed, please update the Test Suite

Click here to Reply or Forward

Figure 32. Sample Gift Card Project Design Change Email Notification


Project Name Gift Card

Project Quality 3.0

Username pallu

Total # of Requirements	13	Total # of Tcers	52	Total # of Defects	1
Total # of Updated Requirements	4	Tcers Mapped to Design	52	New Defects	1
Requirements Mapped to Design	10	Tcers Mapped to Updated Design	0	Open Defects	0
Updated Reqs Mapped To Design	0	Tcers Mapped to Test Execution	3	Fixed and Closed Defects	0

Unmapped Requirements 34,35,36

Figure 33. Sample Gift Card Project Report Showing Unmapped Requirements

If there are any unmapped requirements, the project team can identify them and make necessary updates to TCERs for the completeness of the project.

This Tool will be very helpful for small to medium projects, as the timeline will be very challenging for these projects. All of the large scale projects are well-planned and process-oriented as the timeline will be huge. This is also a good tool for the offshore and on-site model as the staff will be working in different shifts; with this centralized tool it would be helpful for transitioning. Not only this, but imagine the production support dilemma when the person who designed this is not available. With the JAD session recordings and other details, when the project is turned over to the production support team, they can do a better job in fixing the production issues quickly and efficiently.

CHAPTER 5. CONCLUSION

In this paper, the Project Quality Tool was presented. This Tool addresses issues in day-to-day software development in the offshore and on-site model-based companies and distributed development. As such, my Project Quality Tool fulfills the requirements specified in Chapter 2. It allows us to track requirements, designs and test case changes, and for the first time preserving the JAD sessions and unit test result details along with SDLC deliverables all in one portal.

While this tool can be a valuable asset, additional features would enhance it further and may be included in future versions. First, it would be convenient to have the ability to record the JAD sessions and important project meetings live, and have them compressed and stored directly in the respective Audio/Video folder for any given project. Second, future versions should include a feature for checking the SDLC deliverables automatically. Third, a feature to record and plan's project timeline into the tool would make the TCERs reusable between multiple projects with export functionality. These features will automate the entire process flow which will help the Project Managers track the progress made at any given point, and also help the Quality Assurance team re-use the regression test cases at Enterprise Level which would aid in reducing the time and redundancy during test case designing.

As it is, however, the proposed Project Quality Tool will be a useful tool when there are constant requirement changes or any other challenges due to team globalization as this tool would satisfy Users, Management and Software Development teams.

REFERENCES

DeBellis & Haapala. (2006, August). McAllister Dissertation. Retrieved November 27, 2013 from http://www.drjohnlatham.com/resources/2006_McAllister_Dissertation.pdf

Khanduja, J. (2009, March 2). *10 golden rules for requirements based testing*. Retrieved from http://itknowledgeexchange.techtarget.com/quality-assurance/10-golden-rules-for-requirements-based-testing/

Leffingwell, D. & Widrig, D. (2000). *Managing software requirements: A unified approach.* Boston, MA: Addison-Wesley.

Leffingwell, D. & Widrig, D. (2003). *Managing software requirements: A use case approach*. Boston, MA: Addison-Wesley.

McAllister, C. A. (2006, August). Requirements determination of information systems: User and developer perceptions of factors contributing to misunderstandings. Retrieved from http://www.drjohnlatham.com/resources/2006_McAllister_Dissertation.pdf

Nemani, R. & Ichu, E. (2011, July-August). The role of quality assurance in software development projects: Project failures and business performance. *International Journal of Computer Technology and Applications, 2*(4), 716-725.

Noble, S. (2004, October 28). *Why offshore outsourcing projects fail*. (White Paper). Santa Fe, NM: Global Sourcing Insights LLC. Retrieved from http://costkiller.net/tribune/Tribu-PDF/Why_outsourcing_fails.pdf

Spanoudakis, G. & Zisman, A. (2005, August). Software traceability: A roadmap. In Chang, S.K. (Ed.) *Advances in software engineering and knowledge engineering*, Vol 3: Recent advances. World Scientific Publishing, ISBN: 981-256-273-7.

Stepanek, G. (2012). *Software project secrets: Why software projects fail.* New York: Apress.

Sundaram, S. K., Hayes, J. H., Dekhtyar, A., & Holbrook, E. (2008, November 15). *Assessing traceability of software engineering artifacts.* London: Springer-Verlag. Retrieved from http://selab.netlab.uky.edu/homepage/publications/rej_2007_sundaram_hayes_dekhtyar_holbrook.pdf