DEFSOURCE

_____

A Project

Presented

to the Faculty of

California State University, Chico

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Amarjit Singh

Fall 2011

DEFSOURCE

A Project

by

Amarjit Singh

Fall 2011

APPROVED BY THE DEAN OF GRADUATE STUDIES
AND VICE PROVOST FOR RESEARCH:

_____
Eun K. Park, Ph.D.

APPROVED BY THE GRADUATE ADVISORY COMMITTEE:

_____
Abdel-Moaty M. Fayek
Graduate Coordinator

_____
Tyson Henry, Ph.D., Chair

_____
Seung-Bae Im, Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

List of Figures

Figure Page

Figure Page

ABSTRACT

DEFSOURCE

by

Amarjit Singh

Master of Science in Computer Science

California State University, Chico

Fall 2011

DefSource is a practical and effective tool that integrates a bug tracking system with test case management features. By using such a solution, a quality assurance engineer can directly tag the failing test case into its associated bug report. This saves the effort of having to duplicate the steps run to reproduce the defect. DefSource is capable of providing intelligent reports that can be used by management teams to make important decisions such as the readiness of the product to be shipped. Development and Quality assurance teams can also use these reports to gauge the progress of their respective teams. DefSource provides a search feature that can be used to filter test cases, bug reports and helps generate specific reports.

Being a web based solution, DefSource is accessible to anyone irrespective of their physical location. This approach is especially useful for distributed teams which need 24/7 access to such a tool. DefSource is therefore operating system independent

and has little to no footprint on the physical machine on which it is accessed. Since DefSource will be used by multiple teams for different reasons, the user interface is tailored based on the user role. For example, the primary view for the QA user is test case module whereas the development user has access to the bug tracking module.

CHAPTER I

INTRODUCTION

This paper introduces DefSource, a practical and effective tool that integrates a bug tracking system with test case management features. While bug tracking systems and test case management systems are readily available, consolidated systems are rare. There are distinct advantages when using DefSource as opposed to using independent systems, which are explained in detail in this paper.

Software development companies cannot afford to release products that do not meet their customer's expectations. The quality of the product directly relates to its success. Towards that end, companies are always in need of tools that can help track the quality of their product during the product's development stage. A bug tracking system is a tool that helps improve the product's quality.

Most software development life cycle models, whether agile or waterfall, advocate intense phases of testing and defect fixing. This is to ensure a quality product release. A test case management tool aids project management and quality assurance alike to monitor the testing done on the product and gauge the overall quality of the product.

Bug Tracking Systems

A bug tracking system is a software application that facilitates the documentation and tracking of software programming errors (also called bugs or defects). It may be regarded as an issue tracking system [1]. A typical bug tracking system is at the very least a repository of defects found in the product. Using this tool, users can create new defects, edit previously filed defects or simply reference them. It is one of the most important tools that product development teams utilize to monitor the quality holes in the product. Intelligent reports such as “number of high priority bug fixes” and “number of deferred bugs,” generated by bug tracking systems, are used by the project management teams to decide if the product is ready for release.

DefSource’s bug tracking module is a powerful tool that implements multiple features in addition to the basic ones mentioned above. It provides a search engine that uses filters such as name, author, or specific project. Additionally, it provides different interfaces for different users based on their privilege levels. This means that while the product architect could have access to all defects from all projects; a junior engineer might only have read/write access to defects from his current project.

Test Case Management

Prior to developing test case management tools, testing organizations could rely on paper based solutions to plan, design, manage and track their testing projects. However, these days with engineering teams distributed all over the world, such paper based management can no more be relied upon. A portable solution has become a necessity so that there is easy accessibility to it, irrespective of where these teams might

physically be. A good test case management tool is one that maintains a database of all test cases that need to be run for a project. Additionally, it maintains the results of performed tests and creates appropriate reports for management. Examples of popular test case management tools include Qatrac, Qstar and TCMT.

Integrated Solution

Consider a typical scenario during the testing phase: *Using the test case management tool, quality assurance engineers run assigned test cases. If a defect is found while executing a test case, it is logged into the bug tracking system. This bug is resolved by the assigned development engineer and retested by the quality assurance engineer to verify the fix corrects the problem.*

Since test case management and bug tracking go hand in hand, it is helpful that these tools are able to communicate with each other. Unfortunately, there are few products available that integrate both tools. DefSource is an attempt to consolidate test case management with bug tracking. Using DefSource, for the above mentioned scenario: *The quality assurance engineer can directly tag the exact test case to the defect filed. This is helpful for the development engineer since he now knows what exact test causes the defect in the product*.

Documentation Layout

- Chapter II presents related work on bug tracking systems and test case management systems and provides key recommendations for designing each.
- Chapter III provides an overview of the solution provided by DefSource.

- Chapter IV explains the rationale behind using Dot Net technologies and SQL Server for the development of DefSource. It then delves into important implementation details of the project.
- Chapter V provides a step by step guide to using DefSource. All webpages used in DefSource are explored in detail. All roles and modules are explained in depth as well.
- Chapter VI concludes the documentation with a summary of DefSource and lessons learnt while developing it.

CHAPTER II

BACKGROUND

Bug Tracking Systems

Brief Overview

A bug tracking system is a software application that is designed to help quality assurance and programmers keep track of reported software bugs in their work. It may be regarded as a type of issue tracking system. The primary benefit of using such a system is to provide a clear centralized overview of development requests and their state. A prioritized list of pending items (often called backlog) provides valuable input when defining the product roadmap, or maybe just “the next release” [2].

Design Questions

In this section, we look at important research done for designing a Bug Tracking System. Key recommendations are added at the end of each discussion.

When a bug is entered into the bug tracking system by Quality Assurance, customer service or development themselves; it is initially in an “unassigned” state. Typically, development teams conduct triage meetings regularly (sometimes every day or even a few times a day depending upon the urgency and stage of the development) to review and discuss unassigned defects.In these meetings, developers make use of information that has been explicitly recorded in the bug tracker, along with their own

personal knowledge of the project and the team workload balance, to make assignment decisions [3].

Previous studies of bug trackers have looked at how the information presented in them are used for understanding the triaging process within an organization [4]. Aranda and Venolia [4] conducted a study to understand the co-ordination process when fixing bugs in Microsoft. Through thorough analysis of bug history, they identified that the status of the group with respect to milestones and deadlines bore significant consequences to kind and speed of decisions made when new bugs were found. They concluded that the social, organization and technical knowledge played a big hand in decision making during triaging meetings.

Recommendation. When designing a bug tracking system, add a field that lists all developers who have domain expertise or have previously worked on the said module of the product .This feature will expedite the defect assignment or triaging process.

The quality of a defect report goes a long way in expediting its triaging process. When filing a defect, certain guidelines must be followed so that the triaging process can be expedited.

Goldberg [5] explains the very importance of including the detailed steps to reproduce the defect in the defect report. Not all defects are reproducible a 100% of times. In such cases, a detailed description of what was happening on the system at the time can be very useful. Other important aspects like the build used, severity and product module must be added in bug details.

Recommendation. A bug tracking system should ensure that all mandatory fields are filled.

These mandatory fields should include the build used, severity of the defect, reproducibility level and specific Module of the product where the defect was found. Also the “steps to reproduce” should be included as a template for the very first comment in the bug report.

Quite a few times, a defect entered in the bug tracking system is found to be a duplicate of a previously filed defect. This is not desirable as it is a waste of time for the triaging team who go through the entire report only to find that it has already been reported before. In a pressing schedule especially, as is the case with most projects these days, it is important to ensure that the defect is not a duplicate. If the defect is a duplicate of a previously filed defect but has some value addition, it is a better idea to append the new findings in that defect rather than filing a new defect.

Recommendation. A bug tracking system should include a powerful search engine that searches previous defects based on Abstract, Title, Project Version and/or Author.

In a corporate environment, a bug tracker is used by development, quality assurance, project managers, customer service, field engineers, managers etc. It is therefore imperative that the Bug Tracking tool is designed keeping all stake holders in mind. Ideally, the database of defects should be available to all. Bertram et al. [06] conducted a study wherein they deduced that a role based bug tracker is extremely helpful in small size development teams. By role based, they meant that the data in bug tracker should be available in a need-to-know basis. There should a level of abstraction

for information that might be available in the bug tracker yet not be helpful for some. Similarly, there should be emphasis on aspects of the bug tracker data that are especially needed based on the role of the employee.

Recommendation. Users for the bug tracking systems must be created on the basis of their role. Each role must have different privilege levels. For example, users from QA or Development team should have complete access but partners and 1st level support teams should only have limited access to the database of defects.

In their research about resolution time for bugs, Hooimeijer and Weimer [7] observed that defects with more comments and clearer language typically get resolved faster. This was because developers had more relevant and clear data to understand what the exact symptoms were and how to reproduce them repeatedly.

Zimmerman et al. [8] suggest some vital improvements that existing Bug tracking systems must include. There is a need to make the bug trackers more tool, information, process and user centric. Collection of information via stack traces from developers, the quality of defect reports from the reporter, the quality of defect triaging are some ways to achieve these improvements. In their case study, Just et al. [9] provide recommendations for the next generation of bug tracking systems .These recommendations include maintaining the reputations of bug reporters, creating different User Interfaces for different users, support for bug reports in foreign languages and their translation.

Recommendation. Maintain a database of defect author reputations. Attempt a multilingual bug tracking system. Create a role based user interface for the bug tracking system.

While fixing an assigned defect, a developer must maintain regular comments in the bug tracking system so that all concerned are well informed of the progress. However the developer is also expected to maintain the same comments in the source code as well version control for the benefit of reference at a later time. This can be both a tedious as well as a duplicated task. David Russell et al. [10] suggest that for integrated source control and defect management systems, a developers comment in the source should automatically be updated as comments in the defect tracking system as well. This will increase the efficiency of the developer as well ensure that the exact same comments are added at all required places.

Recommendation. For integrated systems, allow automatic comment updates in source code, defect tracking systems and version control systems for improving efficiency

Comparison Between Existing Bug Tracking Systems

There are innumerable defect tracking systems available in the market these days. However, finding the tool that best fits the requirements of the company requires careful evaluation. At the very least, one needs to answer the following questions before investing into a bug tracking tool:

- What are the different roles and responsibilities of the people who will use the system?
- What is the workflow for managing and resolving bugs? Identify the steps in the process and determine who is responsible for each step.
- What information is needed to track for each bug?

- What reports and metrics are needed?
- Are different levels of access to different users required?

The answers to the questions help identify the requirements for the bug tracker. These requirements can then be translated into a “feature list”:

- Adaptability
- Change history
- Version control integration
- Customizable fields
- Ease of use
- E-mail notifications
- Reports
- Security
- Web-based client
- Workflow [11, p. 1]

Tracking systems can be differentiated as web based and standalone applications. With the idea to have their teams work 24x7 on the product, it is a norm to find wide spread distributed teams (all over country and sometimes even worldwide). Web based Bug tracking systems are preferred in these cases as it is then available anywhere and anytime. Very few bug tracking systems are developed using Microsoft .NET technologies and most of these systems are not free. Others are free to use but are closed source.

Bugzilla, by Mozilla Organization, is inarguably the most popular free and open source bug tracking tool available .The primary reason for Bugzilla’s immense popularity is the ease of use for the end user. The User Interface is elegantly simple yet intuitive enough that even the most inexperienced user is comfortable using it. In spite of the simple User Interface, Bugzilla implements nearly all required features and is a powerful bug tracking tool. It organizes the defect tracking process by providing a

centralized system for entering and assigning defects and tracking their progress. It features advanced reporting, inter-bug dependency graphing, integrated product-based security schemas, available integration with CVS, extensive configurability and it is fully customizable [12].

Bugzilla is mainly developed to be used on Linux Platform or Windows platform with “Cygwin.” The backend to store the bugs, issues and feature enhancements is MySql. Also, the support is provided free via email or newsgroups by team members. Main features and benefits listed by Bugzilla are as follows:

- Optimized database structure for increased performance and scalability.
- Excellent security to protect confidentiality.
- Advanced query tool that can remember your searches.
- Integrated email capabilities.
- Editable user profiles and comprehensive email preferences.
- Comprehensive permissions system.
- Improve communication.
- Increase product quality.
- Improve customer satisfaction.
- Ensure accountability.
- Increase productivity. [12]

ITracker is an open source ticket management system that is available free via Sourceforge.com.

It is a Java based lightweight client that provides only the minimal features such as email notifications, detailed history per defect and project based buckets. Additionally, Itracker does provide a plug in for the Eclipse IDE. “Eclipse is an extensible development platform and application frameworks for building software” [13] Among its limitations, the disallowance of group level permissions is a major issue. It is

for this reason that ITracker is not recommended for enterprise level organizations; however, it works out fine for small and medium business organizations.

"Roundup" is a template-based bug tracking system developed in Python. Python (2.3+) is the basic requirement for this bug tracking system [14]. It supports "sqlite," "metakit," "mysql" and "postgresql" databases. A distinct advantage that Roundup holds is that the Web interface and database implementation are fully customizable. To make database management easier, command prompt tool are provided. This product provides most of the issue/bug tracking functionalities. Automated mail notification, support for more than one database, customizable web interfaces as well as database schema are few of the major features of this software. Similar to "ITracker," this system does not provide support for grouping users. Also, integration with VSS (Visual Source safe" or some other repository system is not provided in this software. Most of the customization can be achieved by modifying the scripts. Even though the system provides completely customizable web interface, the customization needs a lot of knowledge about coding, scripting and web interfaces. Another drawback is, product has no interface for generating reports or to display them. The product does not provide support for the user to customize issue or bugs view.

"BUGTrack" is a bug/issue tracking system developed by ForeSoft. This tracking system is a web based paid tracking system. The companies using this system do not need to install anything on their servers. The web site is managed and maintained by ForeSoft. The companies have to pay to the "ForeSoft" for maintaining the defect records. As the installation and maintenance is done by "ForeSoft," companies will not

have to worry about the platforms and databases supported by the system. This system provides most of the features required.

Major drawback of this system is companies have to pay ForeSoft for maintaining bugs. Another basic problem with this system is that companies using BUGTrack cannot modify or customize it whenever they need to. Instead, they have to rely on the “ForeSoft” support team to do the modifications. One of the features this system lacks is that it does not support integration with any source repository systems [15].

Table 1 summarizes all the pros and cons of the aforementioned defect tracking tools.

DefSource incorporates all the features mentioned above for its Bug tracking module.

Test Management Tool

Brief Overview

Test management tools are used to structure automated tests and manual test processes, and to easily manage multiple environments. Quality assurance teams use these types of tools as a single application for managing test cases, environments, automated tests, defects and project tasks [16].

Design Questions

In this section, we look at important research done for designing a bug tracking system. Key recommendations are added at the end of each discussion.

TABLE 1
PROS AND CONS OF BUG TRACKING TOOLS

Features	Bugzilla	ITracker	Roundup	BugTrack
Platform / Framework	Linux, Windows with Cygwin	Platform independent, J2EE	Runs on most of the platforms	Installation is not needed. Web site is managed by "SkyeyTech" company
Database	MySql	Database independent	sqlite, metakit, mysql and postgresql	Installation is not needed. Web site is managed by "SkyeyTech" company.
Customizable	Low: Mainly just for presentation	High: Allows user defined fields, reports	High, but difficult to customize	Low , needs support from SkyeyTech
Distributed team framework	Yes, Web interface	Yes, Web interface	Yes, Web interface	Yes, Web interface
Support for different Projects or products	Yes	Yes	Yes	Yes
User permissions for projects or products	Yes	Yes	Yes	Yes
User Grouping	No	No	No	Yes
Group Permissions for Projects or products	No	No	No	Yes
E-mail Notifications	Yes	Yes	Yes	Yes
Issue Searching	Yes, also provides facility to search the issues or bugs using Regular expressions.	Yes	Yes	Yes

Table 1 (Continued)

Features	Bugzilla	ITracker	Roundup	BugTrack
Attachment Support	Yes	Yes	Yes	Yes
Reporting	Yes	Yes	No	Yes
Cost	Free to install and use	Free to install and use	Free to install and use	Paid
Open Source	Yes	Yes	Yes	No

Hiren Desai performed a research for the development of test case management system (TCMS) at BellSouth Telecommunications. At the time, Different test groups within the company had different formats for writing test cases with nearly no means of recording and comparing the results of their testing. TCMS was born out of necessity and the need for robust software that could maintain test cases (in a standard format) in a centralized repository was growing. With quick reports, test case generation, project based distribution and automation script management, TCMS has become the benchmark for all test management tools [17].

Recommendation. When designing a test case management tool, features such as report generation and automation script management are mandatory

Wegener et al. [18] conducted a study on real time systems using genetic algorithms and found that evolutionary/genetic programming were some of the few techniques that aided test case generation among tools.

Recommendation. Use genetic programming for automation test case generation.

Ahmed Ibrahim Safana et al. [19] conducted a study to develop the SpiraTeam test management system. From their research, they concluded a good test management system is one where the entire testing process can be integrated into a single environment. Additionally, the tool should be flexible enough to have the ability to migrate and integrate data when required. In other words, it must have good data processing software.

Recommendation. A test case management tool should include the feature to integrate and migrate test cases when required.

Developing a test management system for large complex systems has its own intricacies.

Continuous data capture of test results with respect to traceability and failure rate detection becomes utmost important in such project environments. Sigrid Eldh et al. [20] stress that three necessary features that should be incorporated in such systems: a loosely coupled integration of various tools, use of test specifications as a release criteria, and automatic traceability of failures. Manual tests should also be contained in the same test management tool as this reduces the problem of testers having to go through two different environments to look for their test cases.

Recommendation. For large sized projects, manual tests should be incorporated in the test case management tool.

Comparison Between Existing Test Case Management Tools

In this section, we compare HP Quality Center and TestRail test management tools.

HP Quality Center, by Hewlett Packard, is by far the most popular test case management tool available these days. It has a market share of 61.7%.The primary reason for Quality Center's success is that it is a one-stop shop for all software management tool requirements. It provides risk analysis, defect management as well as quality management through a single console based application. The latest version of Quality Center includes the following features:

- Built-in sophisticated requirements management functionality.
- Risk-based testing based on business requirements.
- Cycle and release management for agile test methodologies.
- Enhanced functional and regressive testing capabilities.

In agile cycles, understand how to prioritize tests based on the risk to the business, and plan and manage testing cycles.

- **Risk-based decision-making:** Powerful risk-based quality management capabilities enable you to calculate how much effort to spend on testing each requirement based on the business risk level of the requirement and the resources available. For each requirement, the software automatically calculates business criticality and failure probability. What-if scenario functionality also helps you estimate the resources it will take to test each requirement.

- **Release and cycle management:** HP Quality Center helps you track activities across releases and across testing cycles for individual releases so that you can maintain quality and increase efficiency. Easy-to-read graphs provide a visual representation of critical issues such as planned versus actual coverage or release readiness. Analysis tools, moreover, allow you to evaluate the efficacy of progressive testing cycles so that you can

identify the point of diminishing returns at which the effort and resources required for another testing cycle outweigh the likelihood of uncovering defects [21].

TestRail, by Gurock software, is a popular test case management tool.

> Its intuitive web-based user interface makes it easy to create test cases, manage test suites and coordinate your entire testing process. Easily track and follow the status of individual tests, milestones and projects with dashboards and activity reports. Get real-time insights into your testing progress and boost productivity with personalized todo lists, filters and email notifications. [22]

TestRail provides the flexibility to group test cases per test cycle. This is helpful for planning testing phases. Intelligent reports can then be created that provide the required information such as failed test cases, execution rate per user etc. For each test case, the ability to add attachments is extremely important when one needs to attach specific tools to test a particular test case. This feature is also provided by TestRail.

Conclusion

From the research study done in Bug Tracking Systems and Test Management tools, it is evident that these tools have been an essential part of the software development process from the very beginning. While almost all software companies incorporate both of these tools separately, it is very rare to find a comprehensive tool that does both. It is for this reason that DefSource was the chosen subject for this Project. It implements all the recommendations provided in this chapter and attempts to alleviate the concerns of development, quality assurance, and project management.

CHAPTER III

THE SOLUTION

General Information

DefSource has been developed based upon the key recommendations outlined in the previous chapter. DefSource is a module-based solution that is customized per user role. There are three user roles that can be used to login into the DefSource website. These roles are administrator, quality assurance, and developer. Each of these roles is explored in detail in this chapter. By the end of this chapter, the end-user will attain knowledge about the various features of the application based on the each of the roles mentioned earlier and will attain the understanding to operate the application with ease.

User Authentication

A bug tracking system contains confidential information about the product. It is therefore a security requirement to have user authentication for anyone using such a system. A user can have one of three roles: administrator, quality assurance, or developer. The administrator is responsible for creating user accounts for both quality assurance and developers. Additionally, the administrator also manages projects, project modules and other administrative tasks. When a quality assurance engineer or developer needs to access DefSource for the very first time, they need to login with credentials provided by

the administrator. Upon logging in, the user can change account information like username, password, and email.

Role-based Modules

Quality Assurance

A user with quality assurance privileges will be able to view DefSource's test case management module and bug tracking module. Upon logging in, the QA user is displayed with a dashboard listing quality assurance specific details such as adding new test cases, generating test case reports, etc.

Test Case Creation. Using the test case management module, the user can create test cases for each module of the project with multiple tags as mentioned below:

- Title of the test case.
- Name of the assigned QA to execute the test case.
- Expected time required to complete the test case.
- Type of the test (functional test, unit test, or system test).
- Test Phase (pre alpha, alpha, and beta).
- Prerequisites to run the test.
- Description/instructions/steps to perform the test.
- Expected result(s) when the test case is run.
- Start date.
- Due date.

Test Case Execution and Update. During the testing phase, quality assurance engineers are assigned test cases to execute. Upon executing a test case, DefSource

allows the user to enter information such as result of the test case execution, time to execute the test case and extra comments or findings. This information is important to the management team as it helps generate important reports such “list of failed test cases” etc. When the assigned user for the test case has performed the test and updates its status, the author (user who created the test case) will be notified to verify it.

Bug Filing. If a quality assurance engineer discovers a bug in the product while executing a test case, he can file a bug report into DefSource’s bug tracker module. DefSource provides the added convenience of tagging the exact test case that caused the discovery of the defect.

Reporting. Following are the reports that can be generated for test case management module.

- Summary, which will have following template:
  - Successful/failed total.
  - For following tags Project, Module, Assigned to, Author, and Phase.
- List of test cases by Status failed or Successful.
- List of test cases created/completed in given date range.
  - List of test cases by started/due in day/week/month etc.
  - List of test cases assigned to specific QA.
  - List of test cases authored by particular QA.
- List of test cases which are completed on schedule time, before schedule time or after schedule time
  - Details of specific test case.

Developer Role

A user with developer privileges will be able to view DefSource's Bug Tracker Module. Upon logging into the system, the user is displayed with a dashboard listing bugs details like bugs assigned to his name and generating bug reports. When a quality assurance engineer files a bug report the assigned development user is notified by e-mail. He can accept or reject the bug by changing its status to 'Accepted' or 'Invalid'. If the developer decides to work on the bug ('Accepted' state), the Status of the bug is changed to 'Open'.

Bug Resolution. The developer user can resolve the bug either as 'fixed' or 'can't fix'. Other fields that need to be included in the bug resolution are time taken to fix the bug and additional comments. Once the bug is fixed and its status is changed, the author (QA Engineer who created bug entry) will be notified to verify the fix. Upon Successful verification of the fix, the user can mark it as 'verified' or 'closed'. However, if the QA is not satisfied with the fix, he can change its status to 'reopen'. In this case, the defect is reassigned to the developer.

Reporting. The following reports can be generated for Bug Tracker module.

- Summary, which will have following template:
  - Open Accepted Invalid Won't Fix Fixed.
  - Reopen Verified Closed.
  - For following tags Project, Module, Assigned to, Author, Type and Priority.
- List of bugs by status.
- List of bugs created/completed in given date range.

- List of bugs by started/due in day/week/month etc.
- List of bugs assigned to specific developer.
- List of bugs authored by particular QA.
- List of bugs which are completed on schedule time, before schedule time or after schedule time.
- List of bugs for specific project.
- List of bugs for specific module.
- List of bugs for specific test case.
- Details of specific bug.

Feature Considerations for DefSource 2.0

- Bug history. DefSource maintains the history of comments for both bug reports and test cases. These are especially useful when analyzing hard to reproduce bugs or the evolution of testing with successive releases.
- Project management role. DefSource provides an additional user in a Project Manager role. This user is responsible for making sure that test cases/bugs associated with the project are followed-up.

CHAPTER IV

IMPLEMENTATION

Object-Oriented Life Cycle Model

DefSource was developed using the object oriented life cycle model. The object-oriented life cycle model of software development comprises of five phases: requirement analysis, system specifications, design, implementation, testing, and maintenance (Figure1). The requirement analysis phase is used to define application domain. In the system specifications phase software requirements are addressed and

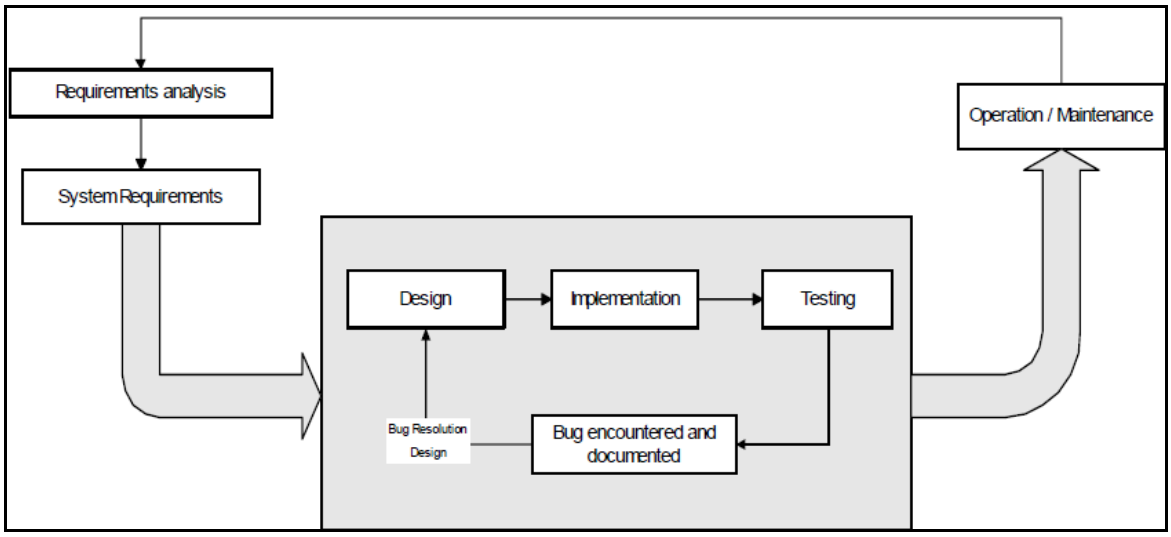

Fig. 1. Object-oriented life cycle model.

Source: Adapted from N. X. Huy, "The Phases of a Software Project," *Software Engineering*, n.d. Retrieved October 16, 2011 from the World Wide Web: http://www.netnam.vn/unescocourse/se/13.htm

objectives are defined. In this model, design and implementation phases are closely associated and are carried out simultaneously.

Requirement Analysis and System Specifications

The first step of any product development is requirement analysis and system specifications. Based on the recommendations provided in Chapter II as well as research on general software trends, the requirement analysis yielded the following results:

- Web-based solution. In today's world of distributed computing and development, a primary requirement of a solution such as DefSource is that it should be web based. It consolidates the information at a single centralized location and is accessible from anywhere using internet. This helps in reducing overhead of managing bugs or test cases at several different locations and also gives better view of the current status of the product. Another advantage of choosing a web-based design is that the user is not required to install a specific operating system to use the product. The system is virtually platform independent for the users. User can use the web browsers like Microsoft Internet Explorer (IE), Mozilla, Firefox, and Netscape to use the tool.

- User based modules. Since DefSource will be used by multiple teams for different reasons, it is necessary that the user interface is tailored based on the user role. For example, the primary view for the QA user should be the test case module whereas the development user should have access to the bug tracking module.

- Reports. Reporting feature should be implemented for both bug tracking and test case management modules. It is important for engineering and management that

intelligent reports are made available. For example, reports can help gauge the product's quality in terms of test case execution results.

- Search capabilities. DefSource will provide the search feature that will assist in finding specific test cases, bug reports based on projects, authors and other filter options. This requirement is particularly important since complex products will have numerous test cases and modules.

Different platforms and servers are available to develop web-based applications. DefSource was developed in .NET framework.

The .NET Framework is a development and execution environment that allows different programming languages and libraries to work together seamlessly to create Windows-based applications that are easier to build, manage, deploy, and integrate with other networked systems [23]. This was the primary reason why .NET was the chosen platform for DefSource.

.NET framework includes unique features like support for different programming languages, support for different platforms, standard networking protocols and specifications, support for libraries developed in different languages. Studies and benchmark of .NET framework conducted by some companies shows that .NET framework outperformed J2EE for web hosting, web services. DefSource implements its web client using the integrated ASP.NET which offers scalability and performance benefits over Active Server pages (ASP) technology. DefSource took advantage of the feature to create custom web controls that are usually referred as user controls. User controls are the customized controls developed by developer and can be reused on different web forms. This helps in reduced development time for DefSource as there was

no need to rewrite same code repeatedly. When developing web pages in ASP.NET, developer has two language choices VB.NET and C# for backend code development. As C# is derived from C++ and java, it is easier for traditional developers to develop the application in C#. Major features of C# that DefSource utilized are:

1. C# is modern object oriented language supported by .NET framework.
2. Many problems like memory management of C++ are taken care of in C#.
3. C# does not allow direct memory operations which are usually know as unsafe operations.
4. Easier for traditional developers who are more familiar with non-presentation to develop application in using C#.

DefSource is hosted on Internet Information services (5.0) since it is fully integrated with Visual Studio .NET. This makes it easier to debug DefSource which is developed in ASP.NET.

Design and Implementation Phase

The design for DefSource is based upon the n-tier architecture. In n-tier architecture, the design is divided into number of logical layers. Each layer has its own unique functionality. Every layer interacts with the layer directly below and the layer above it as shown in Figure 2. DefSource has implemented the n-tier architecture with the following benefits:

1. Separation of data presentation from data access.
2. Easy to expand or add new features.

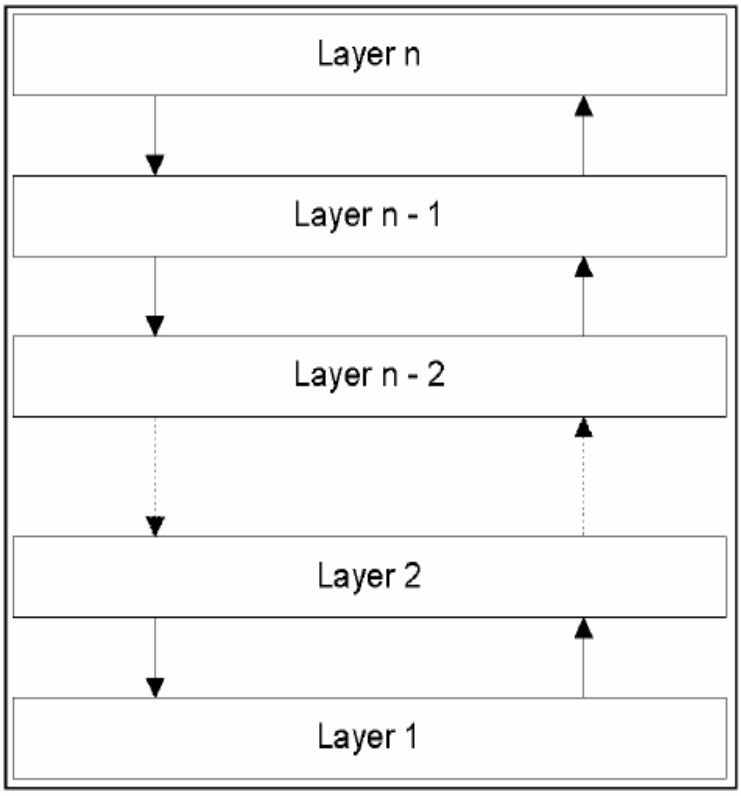

Fig. 2. The n-tier architecture.

3. Restricted access to database from presentation layer. This helps to reduce the vulnerability of exposing database access methods or database objects to the real world. This also helps to increase the level of security.

For the bug tracking system, the design is divided into three main logical layers. The architecture is shown in Figure 3.

1. Data Access Layer: This layer handles all the database operations like connecting to database, retrieval of data from database, updating data to database. It encapsulates all the libraries.

The database layer takes care of object persistence. An object in the object layer can write itself to one or more tables. The database layer was implemented using SQL Server.

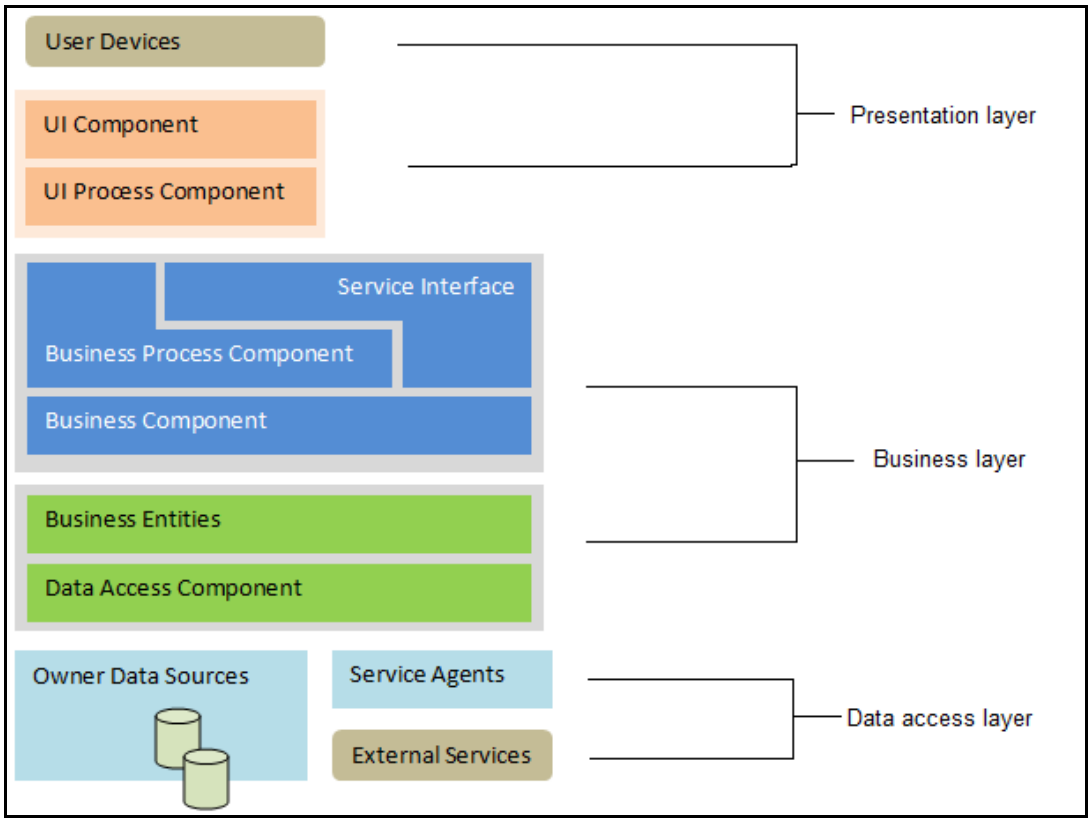

Fig. 3. Presentation, business, and data access architecture.

Persistence Component: This component manages the persistence of the business entities.

Data Access Component: This component abstracts the access to different application data providers such as the database.

Microsoft Data Access Application Block: This is the main component used for database interaction; it encapsulates ADO.NET providing a simple API for accessing database data.

2. Data Engine and Business Logic Layer: Data engine layer executes different queries to get data from database using Data Access Layer. The business layer is the core layer of the application. This holds all the business components implementing the

business rules/logics. It is shared by all the application subsystems and holds all the components that describe typical business entities used in the system. DefSource implements various classes at the Business layer.

An example of how DefSource creates its business rules is: *When a QA user wants to create a bug for a failed test case, the business logic layer must ensure that the following business rules are met*:

i. The user is authenticated as a QA user and

ii. The test case result for the associated test case is marked as 'failed' before entering the test case ID in the bug report.

Some of the important components are:

i. Business component: This component holds all the business rules and logic.

ii. Data validation component: This component holds all the rules that help validate business objects before they are used in business transactions in the system. For the example mentioned previously, DefSource's Business Logic layer validates the business objects by

- Validating the user session to confirm that the logged in user is a quality assurance user and
- Send a request to data access layer to query the database and check the result of that particular test case based on its ID.

iii. Reporting component: This component manages the access, retrieval and routing of reporting tasks in the system.

The middle tier components was developed using C#.

3. Presentation Layer: Presentation layer consists of logic to display appropriate data to user in most simple and usable way. This layer gets required data from data engine layer and business logic layer. The application is template driven and the administrators are able to control the look and feel of the application by changing the style sheets. The presentation Layer is a web client. The web client is the main access point for the application. It is built using a standard MVC (Model-View-Controller) architecture pattern. Technically ASP.Net is the presentation layer. This Layer is implemented in the Web/Application server (IIS).

- **Model:** This is the domain-specific representation of the information on which the application operates. In DefSource, the model is developed such that it contains the database dependant code. However, it does not contain any view-dependant code.
- **View:** This renders the model into a form suitable for interaction, typically a user interface element. For Defsource, the view is implemented using ASPX pages. These pages provide the final presentation of the tool for the end user.
- **Controller:** This responds to events, typically user actions, and invokes changes on the model or view as appropriate. The Controller forwards user actions from the View to the Model, and the Model forwards events to the View (and ultimately to the user) by the Controller as well. DefSource implements controller for providing a user specific UI. For example, the controller edits the view by providing different user interfaces for admin, developer and quality assurance users when they click the “login” button after entering the credentials.

CHAPTER V

USING DEFSOURCE

This chapter provides an in-depth step-by-step guide to use the DefSource application website.

Conventions

When using DefSource, there are certain functions that are used frequently. For example, filling out mandatory fields of a form like Bug Report, searching for certain test cases or other such records. It is therefore imperative that the set of conventions that regulate such functions be observed and understood well. Observing the details of these conventions would familiarize the user to operate DefSource effectively.

Mandatory Fields

The mandatory fields are marked with a "*" symbol in any of the screens. These fields need to be entered compulsorily. A failure in doing so results in erroneous message and/or a failure to save the data on that form.

Example. When creating a user profile, a failure to enter the mandatory fields causes multiple error messages to show up. Figure 4 displays an example of such errors.

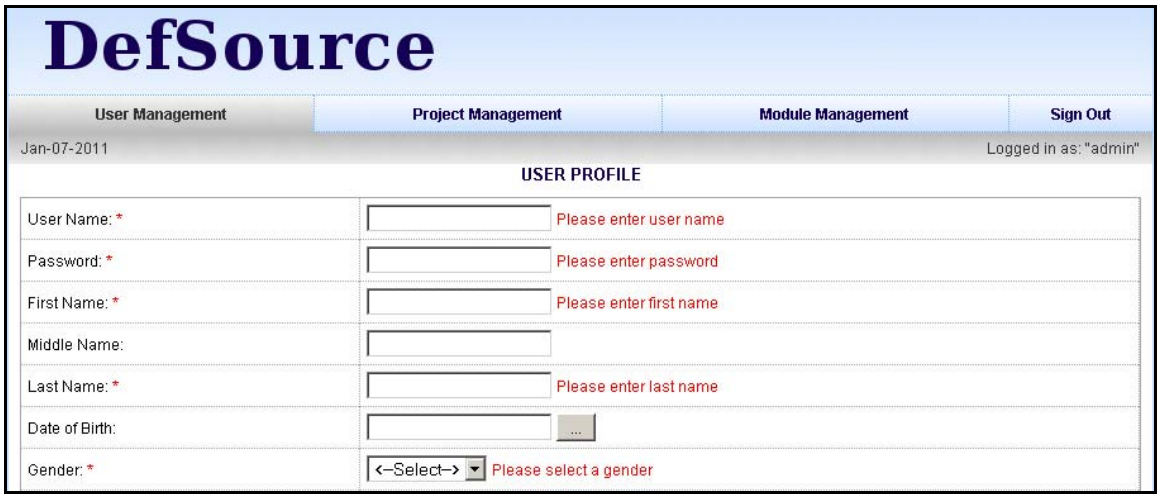

Fig. 4. Mandatory fields.

Searching Records

This feature is used to search and fetch a set of records as per the defined criteria. To search a record, one should define the search criterions and click the “Search” button as shown.

Example. Figure 5 shows the List Users page where “User Name,” “First Name,” “Last Name,” and “Active” are the search criterion. Records are fetched based on the search parameters provided.

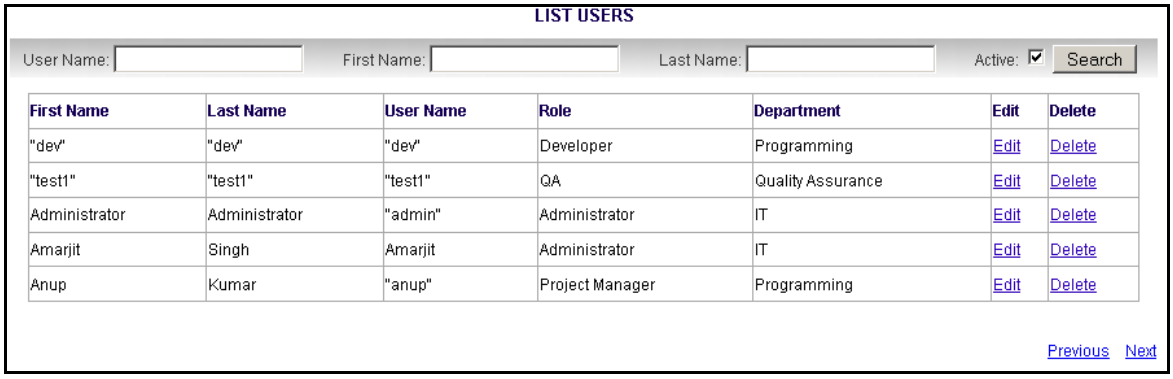

Fig. 5. Search filters in the ‘list users’ page.

If there are more records than what can be viewed in one page, “Previous” and “Next” links are provided to navigate between pages.

Using the Application

This section provides an in-depth explanation of each page available on the DefSource website.

User Authentication

As a part of authentication methodology, classified users are required to **“Sign In”** into the application using the appropriate URL link provided by the administrator. This is an important security requirement since DefSource contain confidential information about the product including ‘bugs’ that will not be fixed.

To log in:

- ✓ Launch the web browser
- ✓ Enter the URL (http://defsource.com)

This would open up the Login Screen (Figure 6).

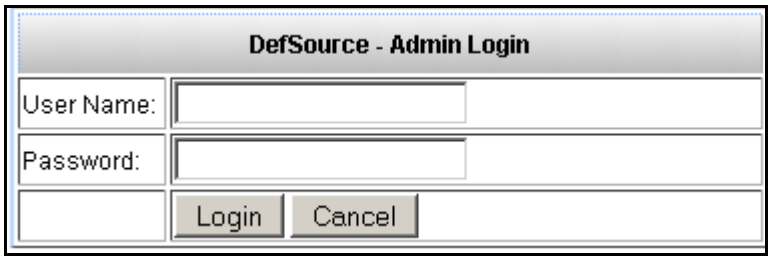

Fig. 6.User authentication.

The user needs to put in the credentials. Based on the role defined, the application displays the appropriate home page. If the logged in user has the role of an

"Administrator," he would get to screen (Figure 7). An administrator user is responsible for creating projects and modules within those projects as well as user management tasks such as creating new users and assigning them appropriate privileges.

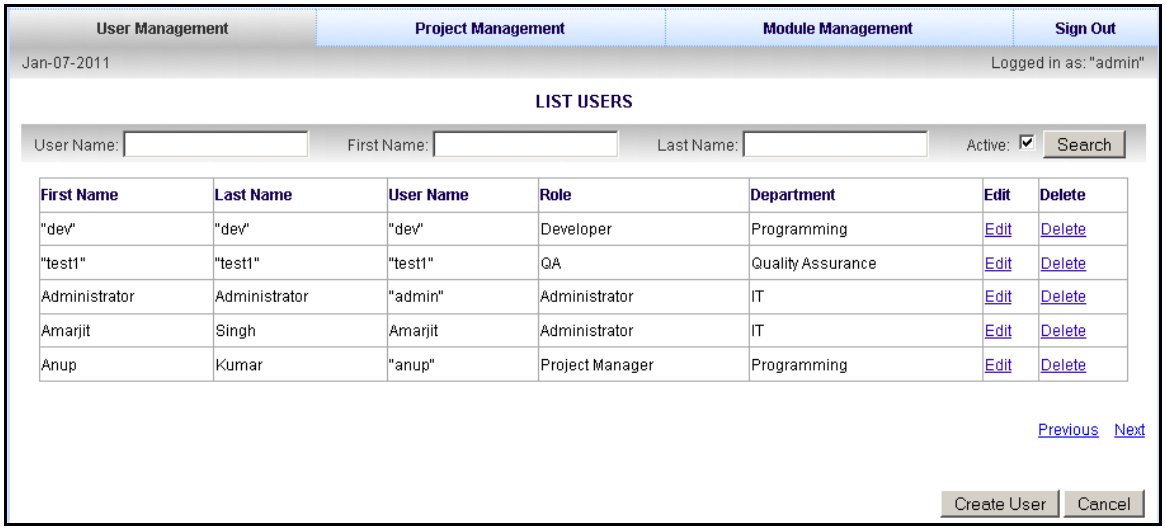

Fig. 7. Administrator user home page.

If the user logs in with the 'Developer' privileges, the appropriate home page opens with 'My Tasks' highlighted (Figure 8).

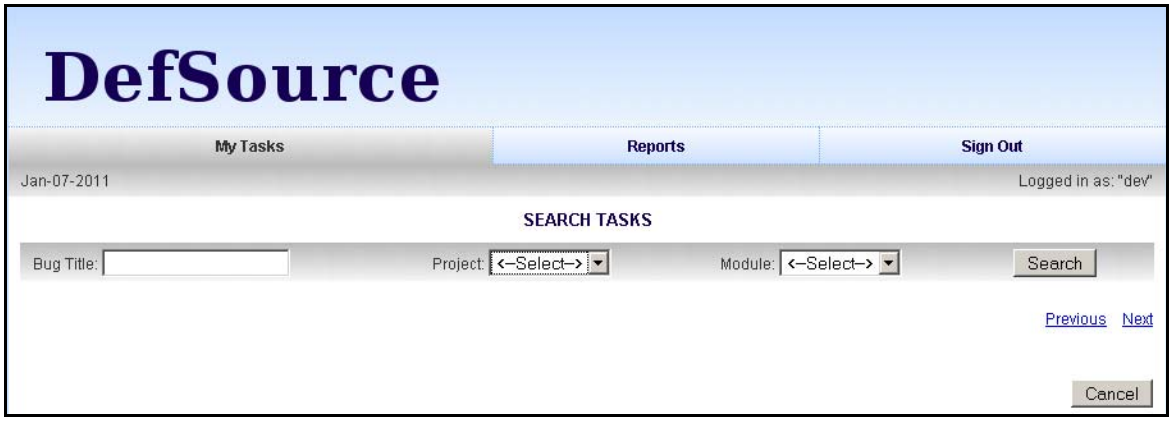

Fig. 8. Developer user's home page.

A developer user can check all assigned defects under the ‘My Tasks’ pane. The search option helps in filtering defects by their name, project or module. The reports pane is used for generating intelligent bug related reports.

If the logged in user has the role of a “Quality Analyst,” the user views “My Tasks” as the home page (Figure 9).

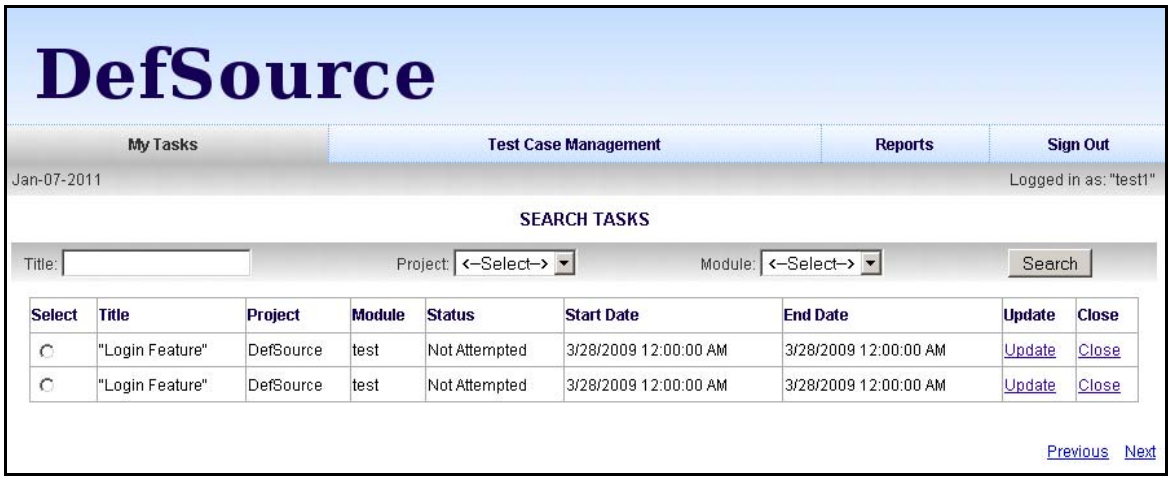

Fig. 9. Quality analyst user’s home page.

The ‘My Tasks’ pane lists all test cases that are assigned to this user. The search feature at the top of the page helps in filtering particular test cases based on title, project, or module.

If the logged in user has the role of a “Project Manager,” the user views “Project Management” as the home page (Figure 10).

Role-based Modules

I. Administrator Module. A user with an administrator role has the following authorizations:

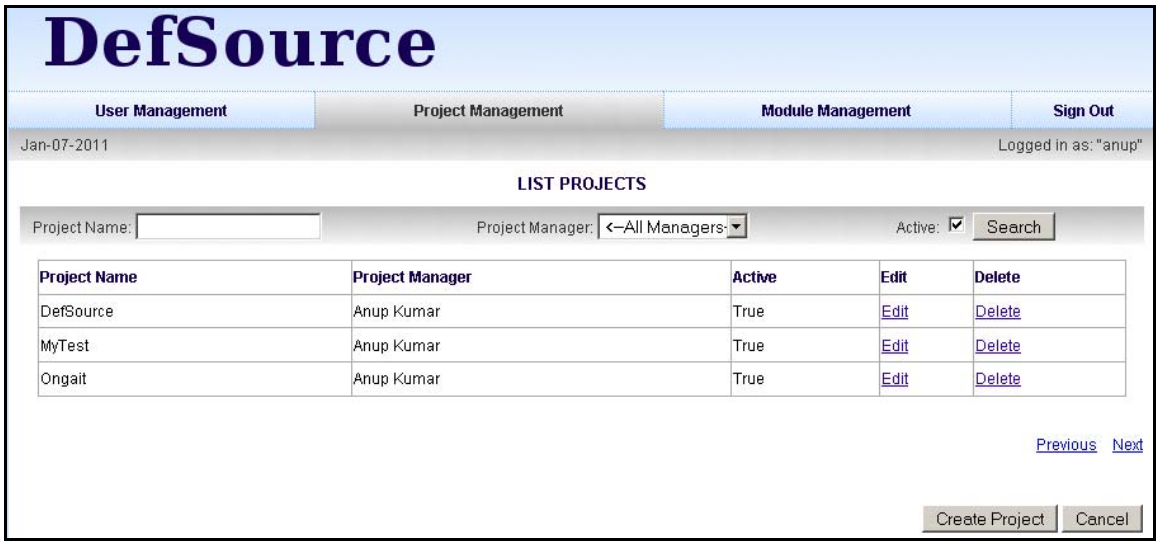

Fig. 10. Project manager user's home page.

- ✓ User Management: Create/Update/View/Delete User Information.
- ✓ Project Management: Create/Update/View/Delete Project Details.
- ✓ Module Management: Create/Update/View/Delete Module Information.

Each of these authorizations is available as a separate tab in the 'admin' homepage.

- **User Management.** From the home page, an administrator can create a new user by clicking the "Create User" button as shown (Figure 11). For example, an administrator would create a new user for new hires or employees added to a new project.

When the "Create User" button is clicked; the administrator gets to the 'User Creation' screen (Figure 12). The user is asked to fill in the following information.

- o User Name *
- o Password *
- o First Name *

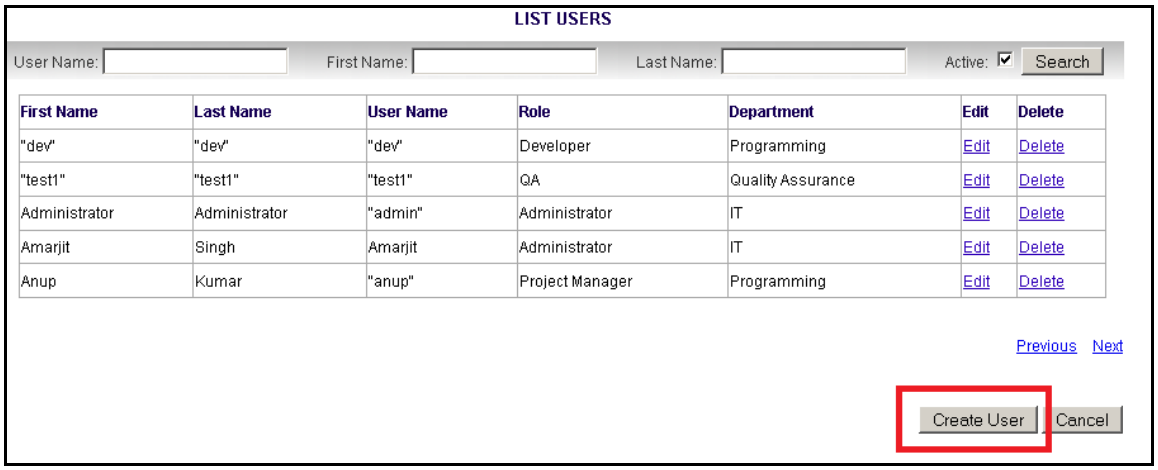

Fig. 11. Create user button shown under 'List Users' pane for the administrator user.

- Middle Name
- Last Name *
- Date of Birth
- Gender *
- Role
- Department *
- Office Phone
- Extension #
- Home Phone
- Mobile Phone
- Official Email
- Personal Email
- Active *

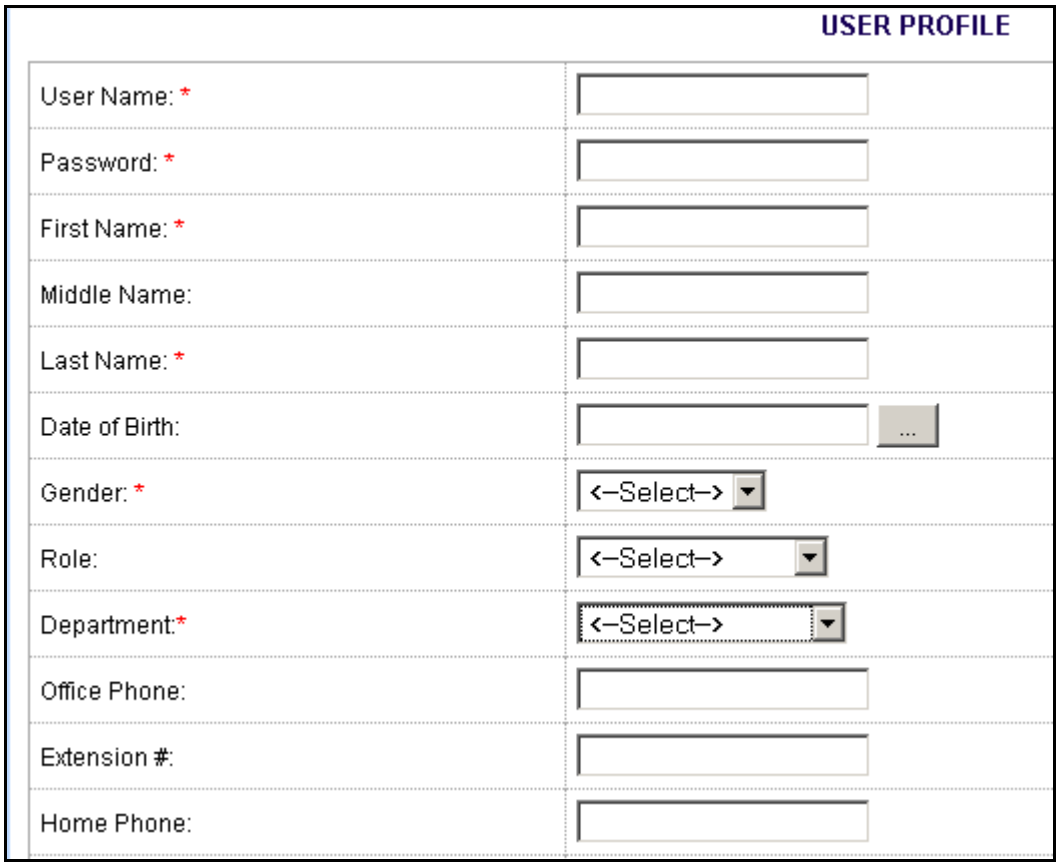

Fig. 12. The user profile form.

Upon entering all the information and clicking the “Save User” button (Figure 13), a new user is created with the given credentials and role. Clicking the “Cancel” button would cancel the process and take the user back to the user listing screen.

In order to update a user’s information for any reasons, the administrator can click on the “Edit” hyper link from the “User Listing” screen (Figure 14).

This will take the user to the “Edit User” screen (Figure 15) which is pre-loaded with the current information as shown below. Any field can now be updated as required.

Fig. 13. Save User button.

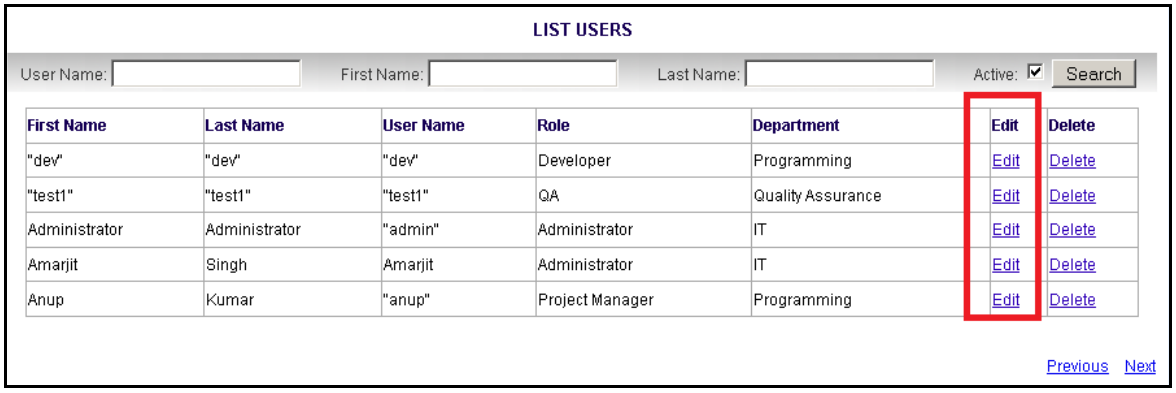

Fig. 14. Edit links highlighted.

On clicking the “Save User” button, the admin user saves the updated information. Figure 16 shows the “Save User” button.

- **Project Management.** In the “Project Management” menu, the administrator will be able to navigate to the Project Listing screen. By default, this screen displays all active projects that are available in the system (Figure 17).

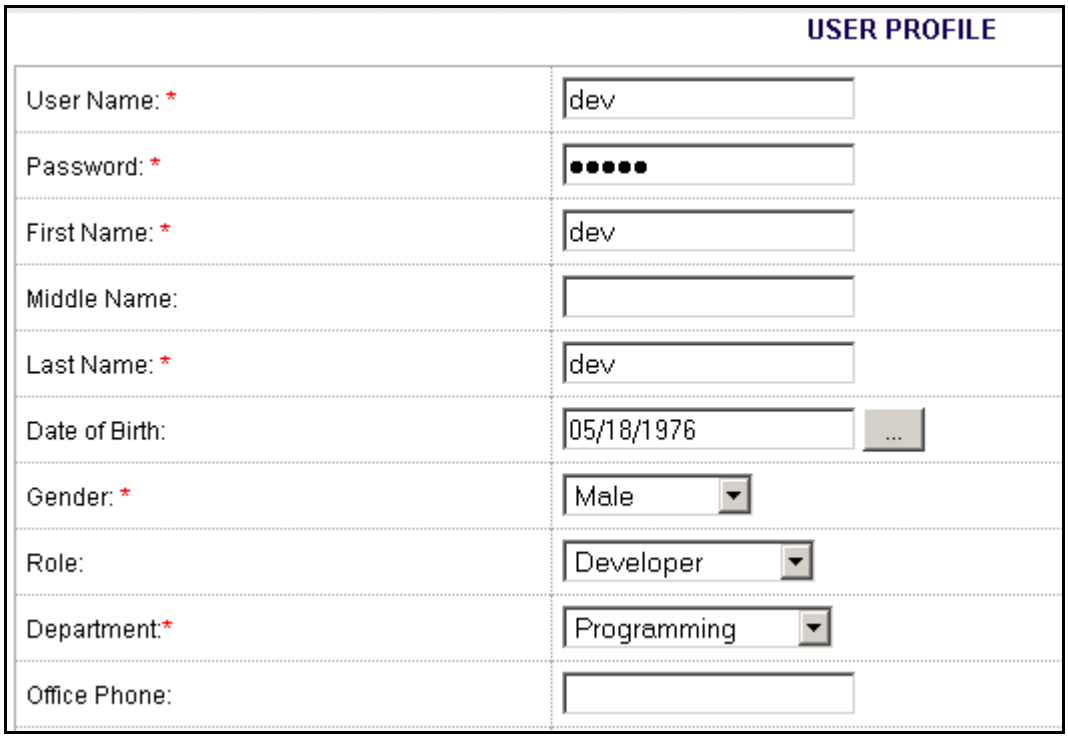

Fig. 15. The “Edit User” form.

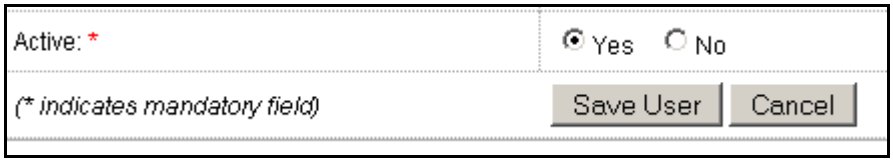

Fig. 16. Click “Save User” to save the updated profile.

To create a new project, click on the “Create Project” button as shown in the previous screen. This leads to the project creation form (Figure 18).

Upon entering all the required fields, the user should click on the “Save Project” button to save the newly created project. Clicking the “Cancel” button

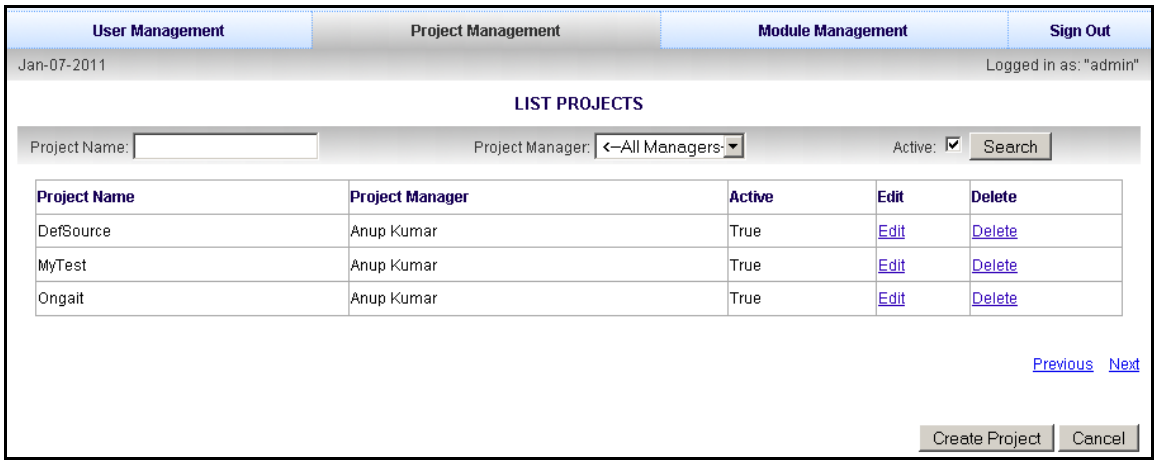

Fig. 17. Active projects listed under project management.

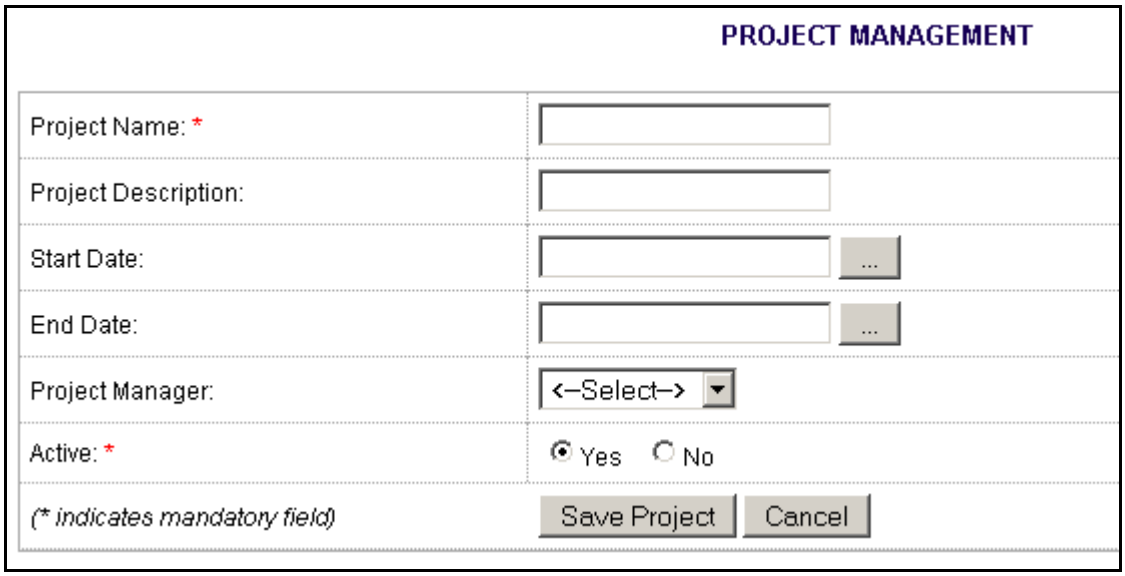

Fig. 18. New project creation form.

cancels the process and takes the user back to the Project Listing screen. This newly added project is now the last entrant in the project list.

To edit or delete an existing Project, click the appropriate ‘Edit’ or Delete’ link the Project Listing screen (Figure 19).

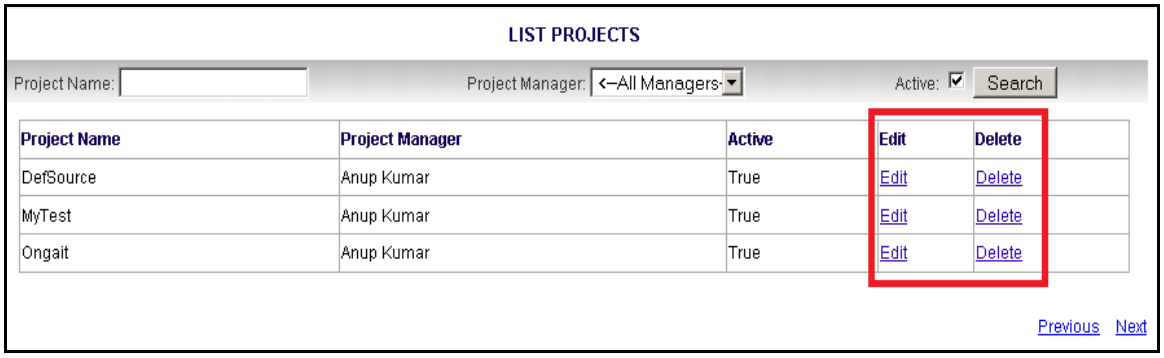

Fig. 19. Edit and delete hyperlinks displayed for all listed projects.

By clicking the “Edit” hyperlink, the user can update project information. The details stored earlier would be loaded in the Edit Project screen (Figure 20).

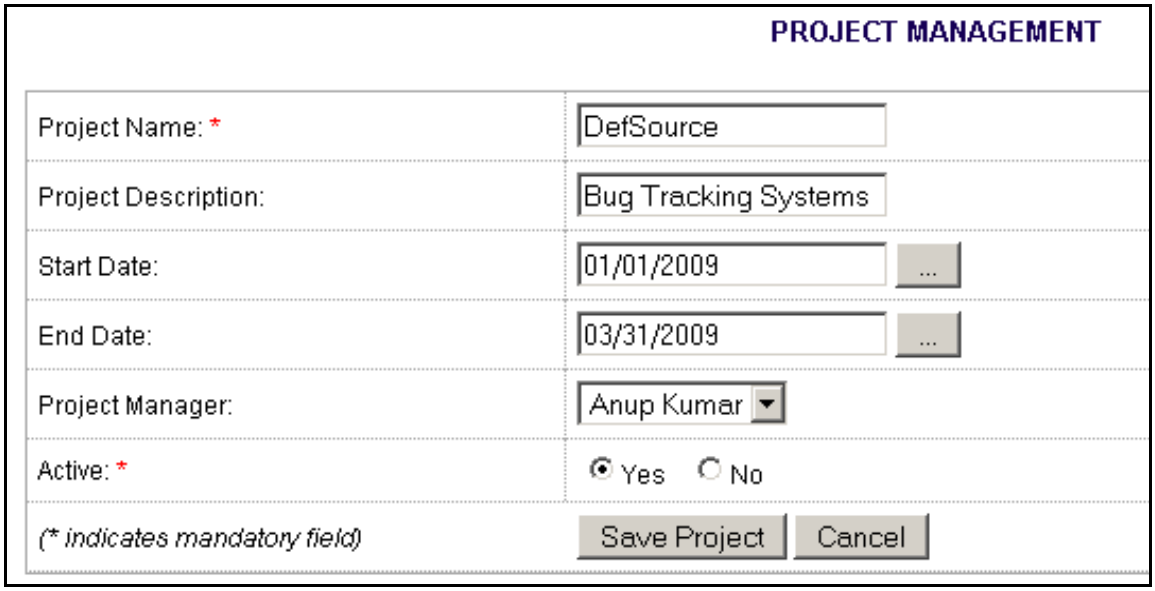

Fig. 20. Project profile form

Click the “Save Project” button to save the updated information. Clicking “Cancel” would take the user back to the listing screen.

- **Module Management.** On clicking the “Module Management” menu, the admin navigates to the Module Listing screen. By default, this screen displays all active modules that are available in the system (Figure 21).

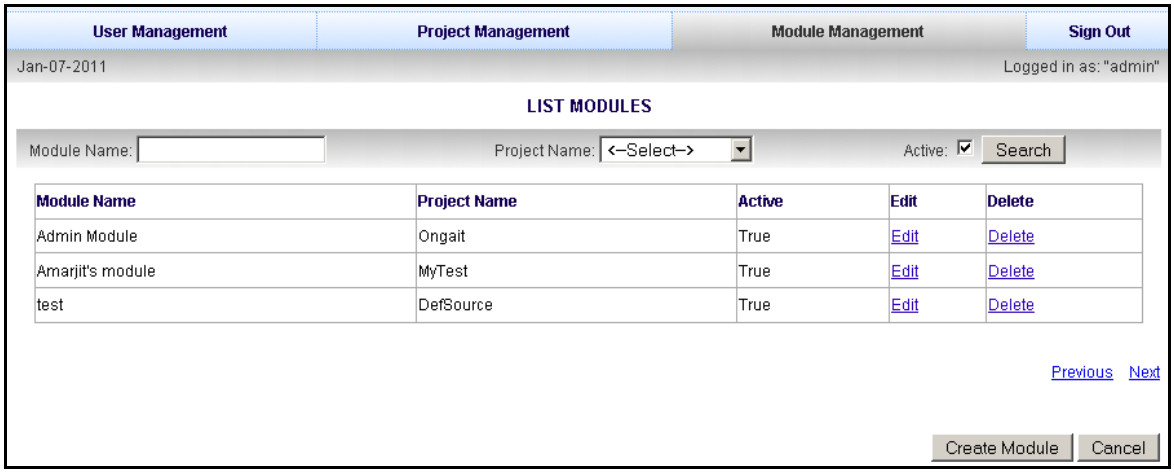

Fig. 21. Module management page.

To create a new module, click on the “Create Module” button as shown above. This leads the user to the Module Creation screen (Figure 22).

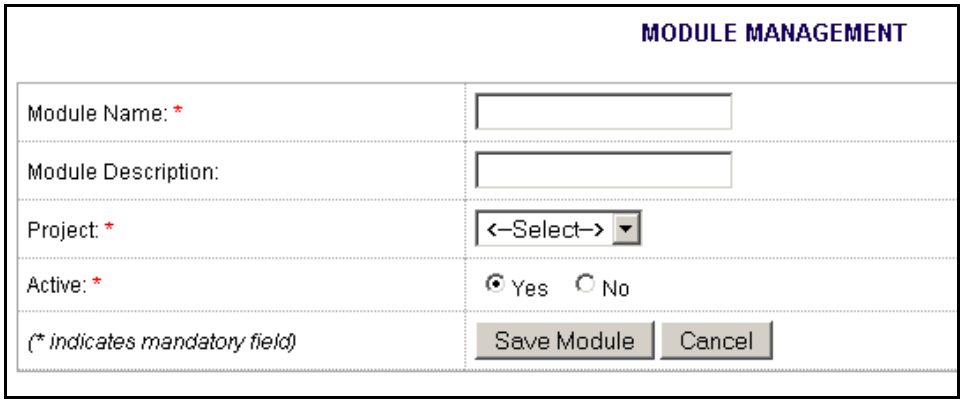

Fig. 22. Module creation page.

After entering all fields, click “Save Module” button to save the newly created module. Clicking the “Cancel” button cancels the process and takes the user back to the Module Listing screen.

To edit an existing module, click the appropriate “Edit” hyperlink. Editable information is now loaded in the Edit Module screen (Figure 23).

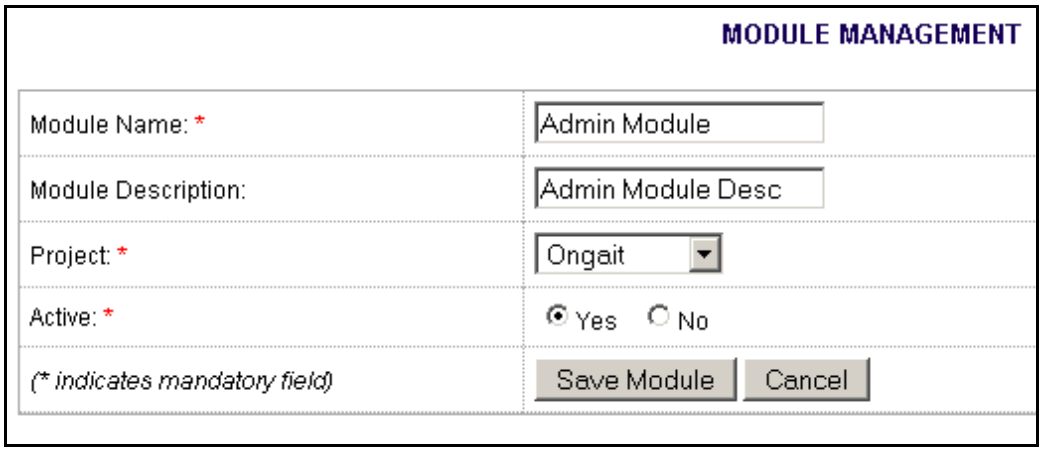

Fig. 23. The module edits form.

Click “Save Module” button to save the updated information. Clicking “Cancel” would take the user back to the listing screen without saving any new changes.

On clicking the “Sign Out” menu, the user would be logged out of the DefSource application (Figure 24).

Fig. 24. Click “Sign Out” to log out of DefSource.

II. QA Module. Using the sign on screen, the quality assurance users login into the DefSource application with QA privileges. The QA homepage contains tabs to “My Tasks,” “Test Case Management” and “Reports” web pages. Each tab is explained in detail below.

- **My Tasks.** By default, the “My Tasks” menu is highlighted and all the tasks assigned are listed (Figure 25). The user can search tasks using the following fields:

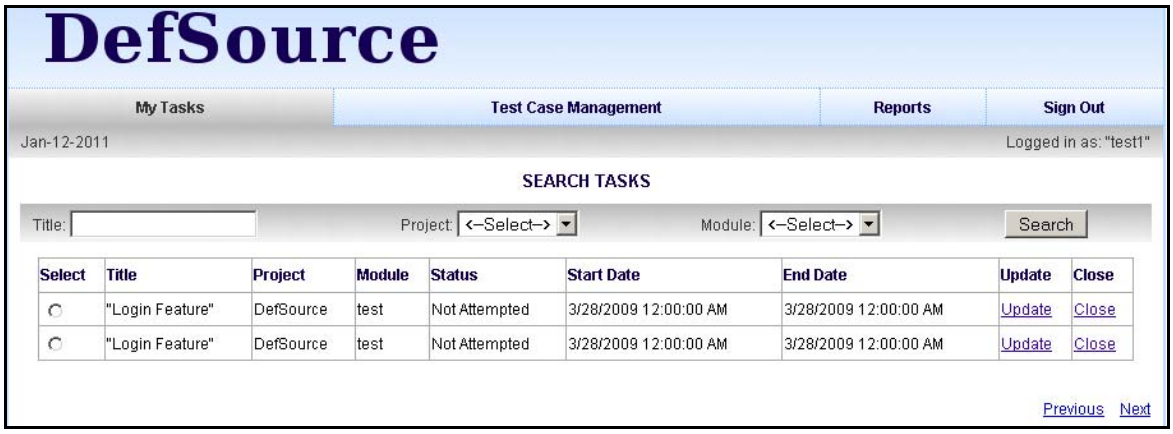

Fig. 25. My Tasks pane for the QA user.

» Title–This is a list of test cases by name.

» Project–This is a dropdown list with all the related projects assigned to the user.

» Module–All the modules for the selected project would be loaded to this dropdown.

Click the appropriate “Update” hyperlink in the listing to update the selected task. Clicking the “Close” hyperlink closes that particular task. If multiple tasks are

listed, the QA can navigate back and forth using the “Previous” and “Next” hyperlinks provided.

- **Test Case Management.** Upon clicking the “Update” hyperlink in the ‘My Tasks’ page, the user will be taken to the Test Case Management screen where he/she can update the selected test case (Figure 26). The details captured are as given below.

  - » Title–Title of the test case.
  - » Description–A brief summary of test case.
  - » Module–The module to which the test case belongs.
  - » QA Name–The name of the assigned quality assurance user.
  - » Expected Time–Expected time to complete the execution of the test case.
  - » Test Case Type–Functional, regression, system, etc.
  - » Test Phase - Phase of testing .For example, alpha, beta, etc.
  - » Prerequisites–Any pre requirements for running the tests.
  - » Instructions–This field requires the steps to run the test case.
  - » Expected Results–This field requires the expected results upon running the test case.
  - » Start Date–The start date for the test case execution.
  - » End Date–The date of completion for running the test case.
  - » Status–The execution result of the test case.
  - » Actual Time Taken–The time taken to run the test case.

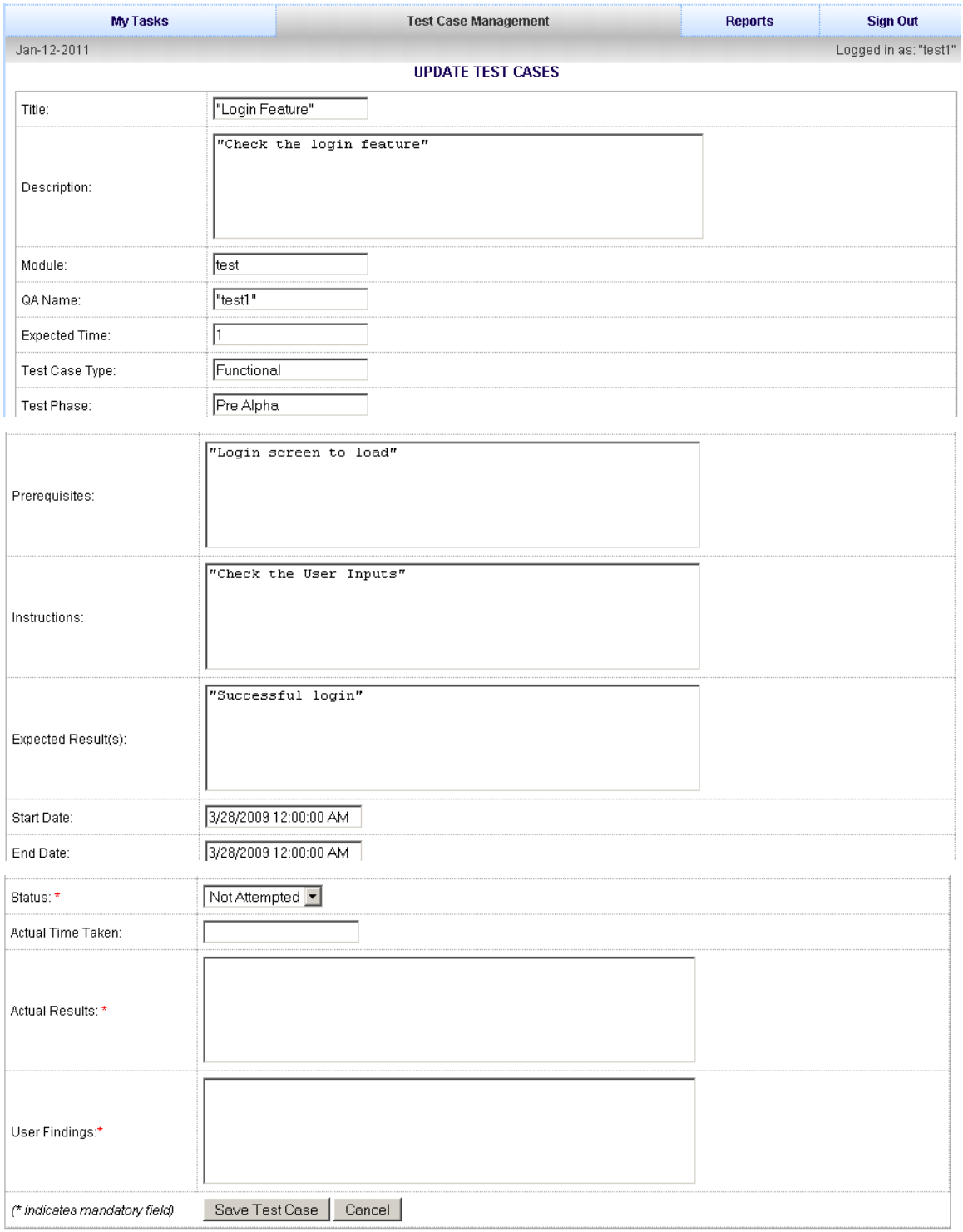

Fig. 26. The test cases update form.

» Actual Results–The actual result of execution of the test case.

» User Findings–Comments that the user needs to add.

Click the “Save Test Case” button to update the changes made to the selected test case. On clicking the “Cancel” button, the operation will be cancelled and the user returns to the “My Tasks” page.

- **Test Case Management–Listing.** Click the “Test Case management” menu to search for particular test cases. Below are the search parameters and the screen details.

» Title

» Project

» Module

A typical search result is shown in Figure 27.

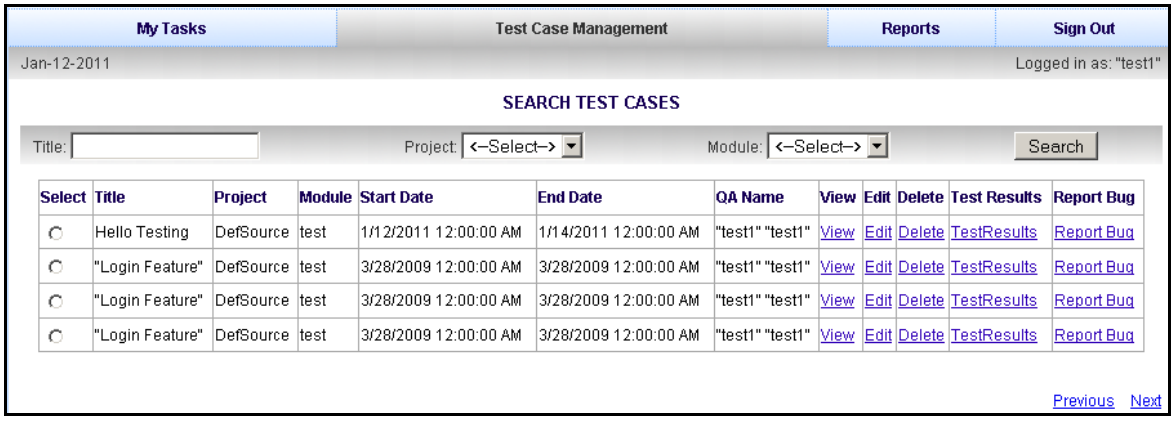

Fig. 27. Search results for test cases.

Click the “View” hyperlink, to open the selected test case in the display mode .In this mode, the user cannot edit the test case details. By Clicking the “Edit” hyperlink, the user can edit the selected test case while clicking the “Delete”

hyperlink deletes the selected test case. Upon clicking the “Test Results” hyperlink, the user is taken to the below screen. (Figure 28)

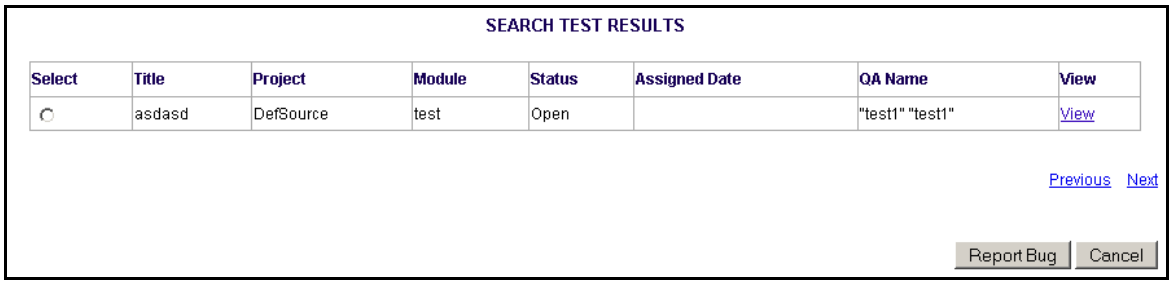

Fig. 28. The test results screen.

Click the “View” button to navigate to bug report details screen where he/she can view the details. On clicking the “Report Bug” button, a new bug can be recorded for the selected test case. The screen is as shown in Figure 29.

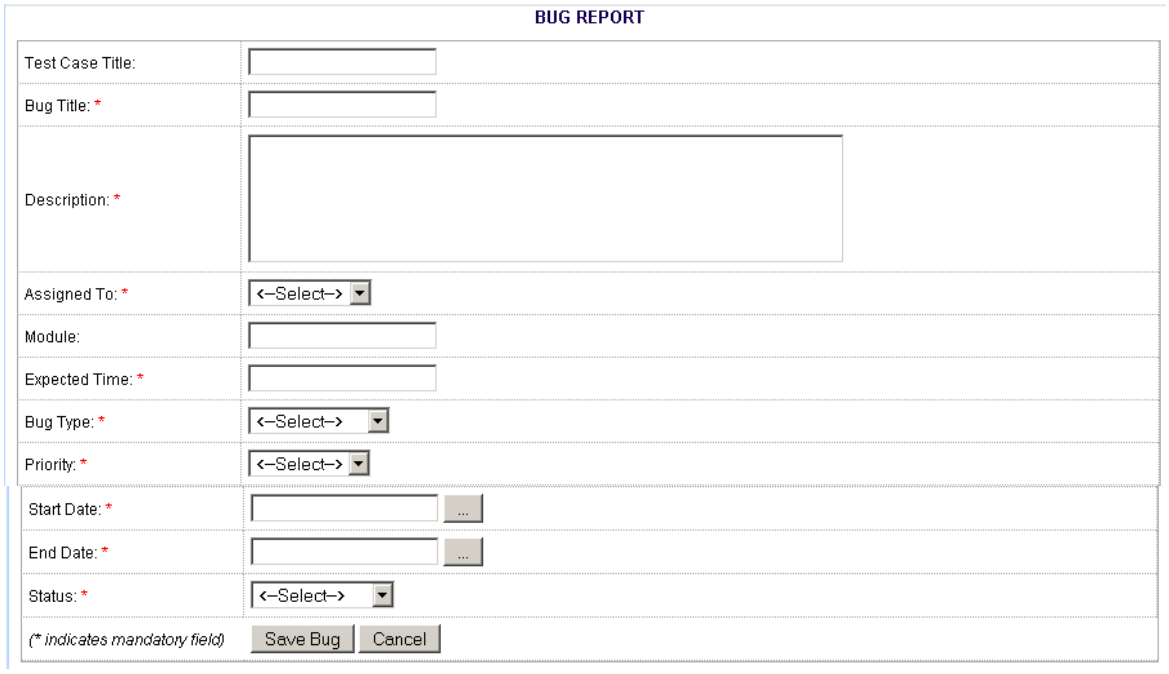

Fig. 29. Bug report page.

- **Reports.** Below is the list of reports available for the Quality Analysts. All reports can be generated based on specific filter options.
    - » Summary: This report lists the number of closed, open, passed and total test cases.
    - » List of test cases by Status Failed or Successful.
    - » List of test cases Created/Completed in given date range.
    - » List of test cases by started /due in day/week/month.
    - » List of test cases assigned to specific QA.
    - » List of test cases authored by particular QA.
    - » List of test cases which are completed on schedule time, before schedule time or after schedule time.
- **Summary Report.** For the summary report, the user can provide the following search parameters to fetch the appropriate report. This report is useful to get information of the overall test case status.
    - » Project
    - » Module
    - » Assigned To
    - » Author
    - » Phase

The report will list the number of closed test cases, un-attempted test cases and the total test cases as shown in Figure 30.

- **Test Cases by Status.** The user can search the test cases using the Project and Status search parameters. The screenshot given below summarizes the Test Cases by Status Report (Figure 31).

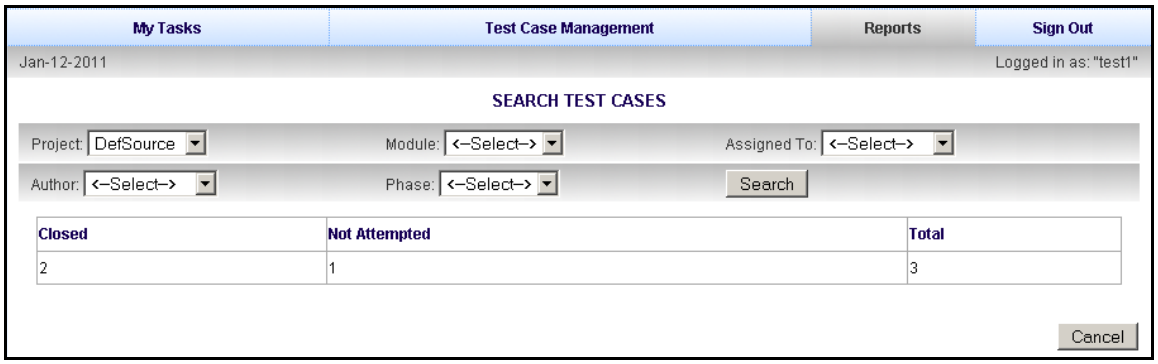

Fig. 30. Summary report

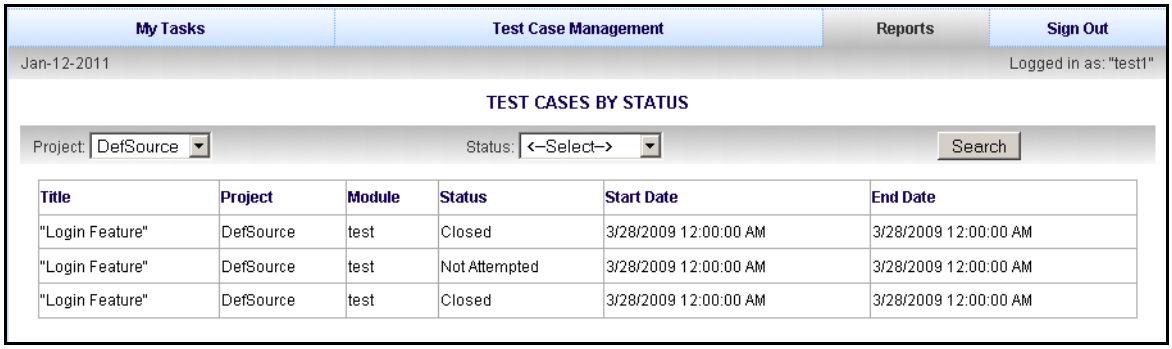

Fig. 31. Test cases by status.

- **Test Cases by Date.** The search parameters for this report include Project, From Date and To Date. The report is as shown in Figure 32.
- **Test Cases by Due Date.** The user will be able to search this report with the following search criterion (Figure 33).
  - » Project Name
  - » Start Date
  - » Due in a day

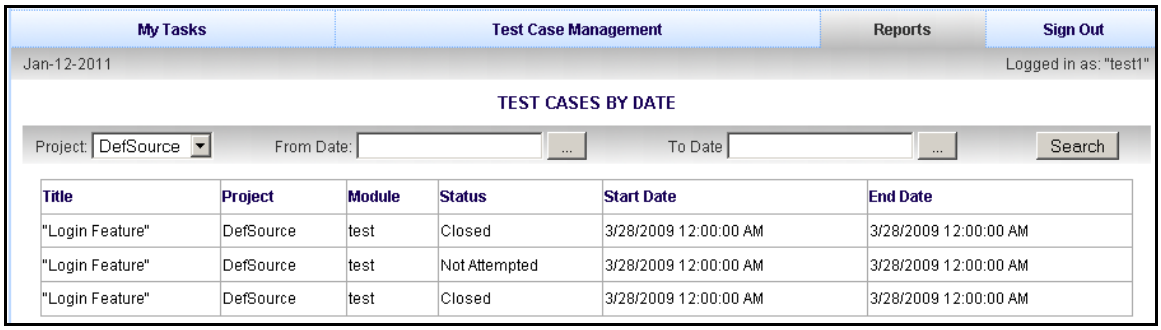

Fig. 32. Test cases by date.

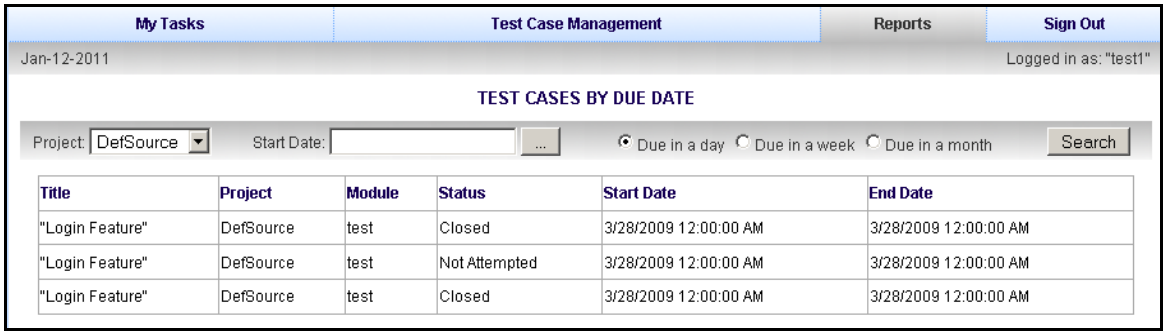

Fig. 33. Test cases by due date.

» Due in a week

» Due in a month

- **Test Cases Assigned to QA.** The following are the search parameters for this report (Figure 34).

» Project Name

» QA Name

- **Test Case Authored by QA.** The below mentioned report will be executed using the Project and Authored by search parameters (Figure 35).

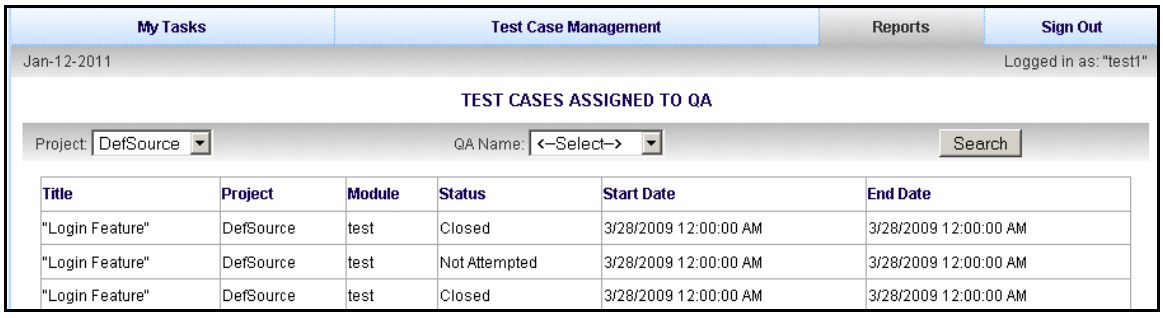

Fig. 34. Test cases assigned to QA.

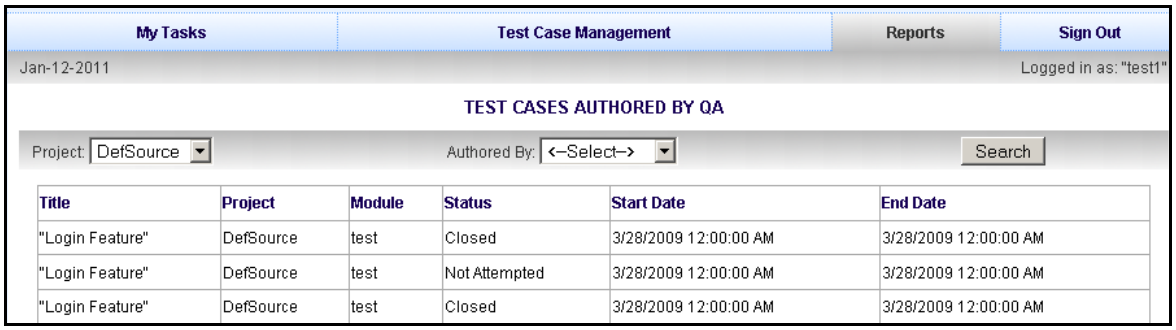

Fig. 35 Test cases authored by QA.

- **Test Cases by Schedule.** "Project," "Completed on Schedule," "Completed before Schedule," and "Completed after Schedule" are the search parameters for the test cases by schedule report (Figure 36). On clicking the "Sign Out" button, the user will be logged out of the application.

III. Developer Module. Once the "developer" logs into the system, he/she would be presented with the following menus: "My Tasks," "Reports," and "Sign Out." By default, the "My Tasks" menu is highlighted and the user would be able to search for the tasks assigned (Figure 37). Click "Edit" to edit the selected bug reported (Figure 38).

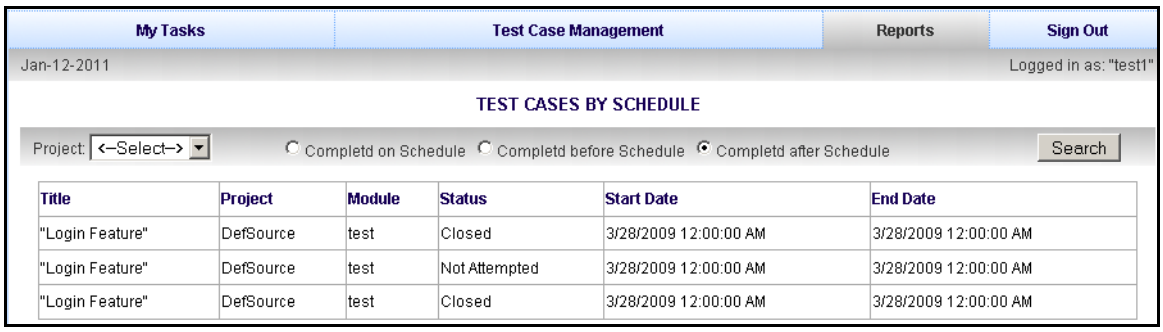

Fig. 36. Test cases by schedule.

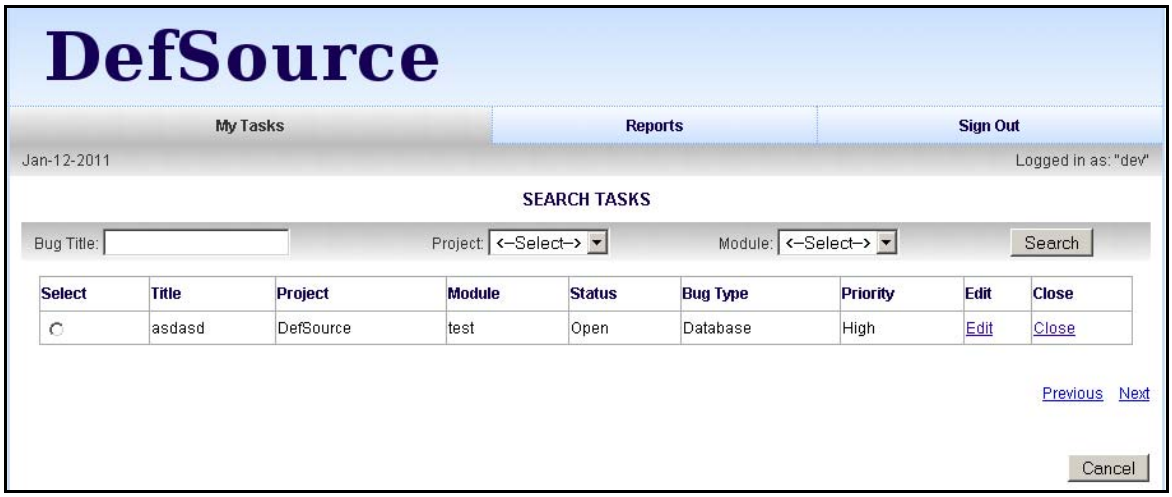

Fig. 37. 'My tasks' pane for the developer user.

The developer can save the bug after providing the appropriate resolution. To do this, click the "Save Bug" button. By clicking the "Close" hyperlink instead, the developer can close a bug reported by the QA.

- **Reports.** There are several reports available for the "Developers" for reference. These include:

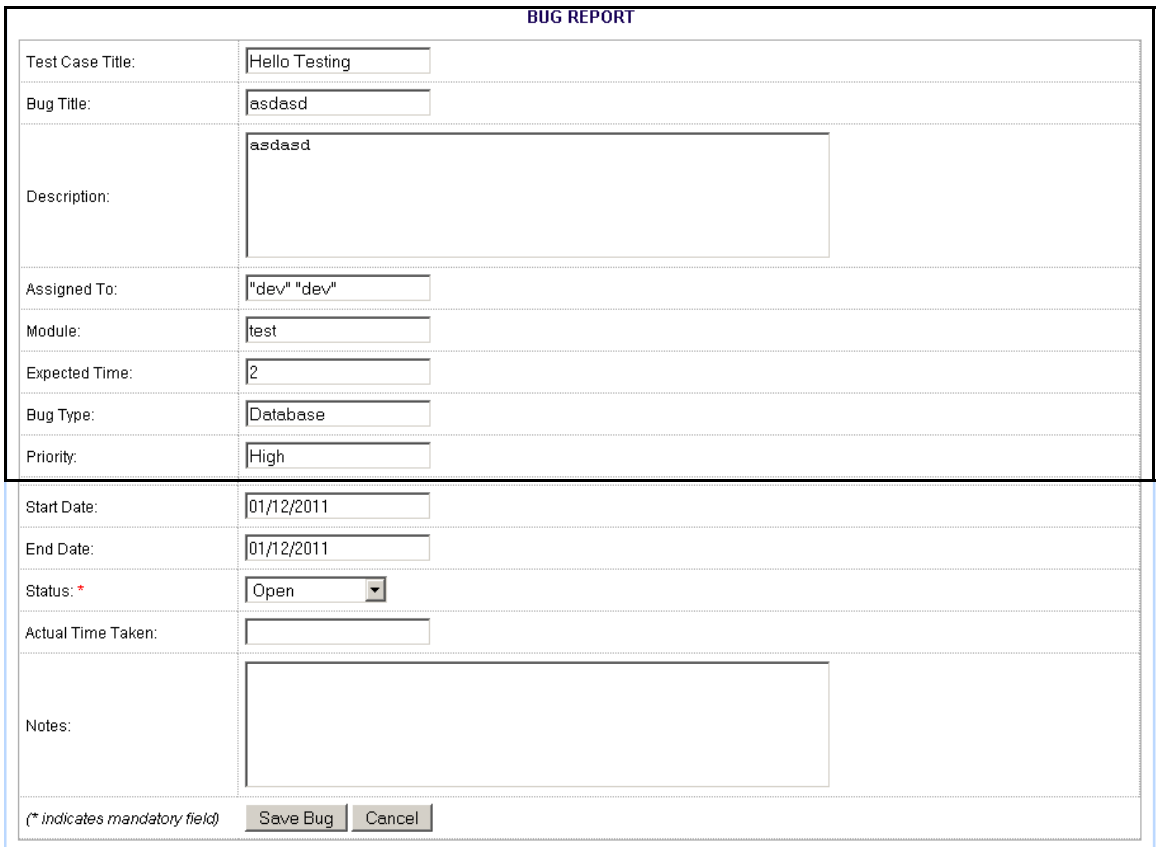

Fig. 38. Bug report edit page.

» Summary.

» List of bugs by Status Failed or Successful.

» List of bugs Created/Completed in given date range.

» List of bugs by started /due in day/week/month.

- **Summary.** This report includes Project, Module, Assigned To, Author, Bug Type and Priority as the search parameters. The report is as shown in Figure 39.

- **Test Results by Status.** The search parameters for this report include Project and Status. The report is shown in Figure 40.

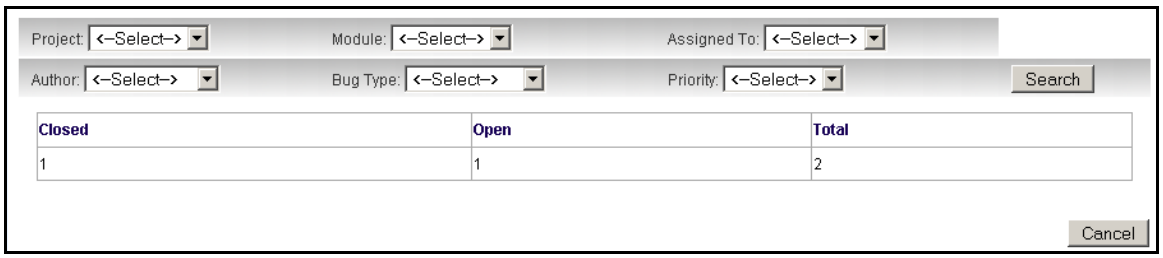

Fig. 39. Summary report for developer user.

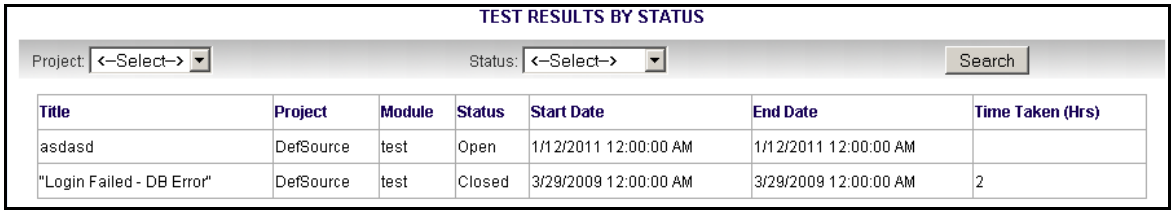

Fig. 40. Test results by status for developer user.

- **Test Results by Date.** The search parameters for this report include Project, From Date and To Date. The report is shown in Figure 41.

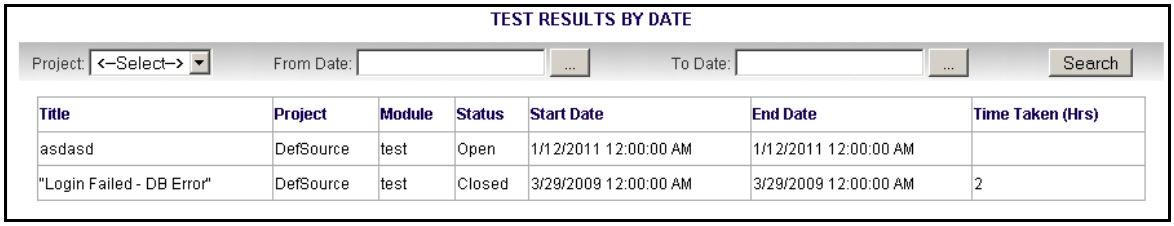

Fig. 41. Test results by date for the developer user.

- **Test Results by Due Date.** The search parameters for this report include Project, Start Date, due in a day, due in a week and due in a month options. The

report is shown in Figure 42. On clicking the “Sign Out” button; the user will be logged out of the application.

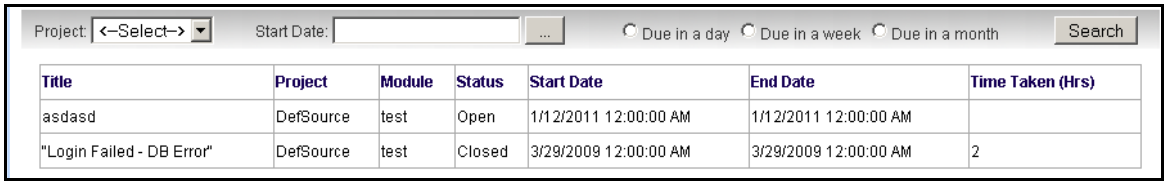

Fig. 42. Test results by due date for the developer user.

CHAPTER VI

CONCLUSION

DefSource provides a solution that resolves important problems faced by software companies with regards to bug tracking and test case management. It alleviates the issue of having to cross-reference a test case in a separate tool when troubleshooting a defect. Being a web-based solution, DefSource is accessible to anyone irrespective of their physical location. This approach is especially useful for distributed teams which need 24/7 access to such a tool.

DefSource builds on existing systems by implementing the recommendations provided in Chapter II. It provides a search engine to help find defects or test cases based on project or modules names.

Program management, development and quality assurance teams can equally benefit from the intelligent reports that can be generated from DefSource. This feature is implemented for both bug tracking and test case management. ‘Test case summary’ and ‘Bug resolution summary’ reports are examples of some of the important reports that development and quality assurance teams can frequently use to check the progress of their respective teams.

Development Experience

Using ASP.NET to develop the web interface facilitated quick development of DefSource as it supports user controls which allow code reusability. It took only 11 days to code the entire front end (ASPX pages). Standard components and controls provided by ASP.NET are easily customizable; which helped in modifying the controls according to the application's needs. C# was the language chosen for coding the backend for ASP.NET. Standard APIs and interfaces defined to connect to different databases helped in creating interfaces for the application. Support for object oriented programming by C# helped in developing scalable and reusable data access libraries.

Creating the database schema took the longest part of the development effort (15 days). This was because of the complexities between test cases and bug reports. Once those complexities were resolved, the other relationships were easier to create.

Lessons Learned

- **Need for data abstraction.** Initially, all queries to the database were implemented in the same classes that also rendered the final information to the end user. Using this approach caused messy code that was hard to understand and build upon. Therefore, there was a need to ensure that there is an abstraction of raw data maintained for the user level (UI). All database queries should be handled at a lower layer of the code, while an intermediate business layer should handle all data and rule validation. The user interface, which is the highest layer, should only be presenting the data in the way the user expects it. By using this approach, the code becomes modularized, thus looks cleaner and easier to understand.

- **Preplan for complex database relationship between test case and bug tracking tables.** Throughout the process of defining the database design for DefSource, the relationship between test case and bug tracking had to be kept in mind. For example, to file a defect, one needs to refer a test case table to get the test case ID and also a users table to associate the defect. A complex database is expected when designing a consolidated system such as DefSource.

Suggestions for DefSource 2.0

- Add graphs and charts for reports.
- Allow export of reports in XML or CSV format.
- Add a command line interface for performing database tasks such as backup, restore, purge etc.
- Add reminder notifications for pending tasks.
- Integrate source control abilities**.**

REFERENCES

REFERENCES

[1] Wikipedia, “Bug tracking system,” 2011. Retrieved September 3, 2011 from the World Wide Web: http://en.wikipedia.org/wiki/Bug_tracking_system.

[2] Wikipedia, “Bugtracker,” 2011. Retrieved September 3, 2011 from the World Wide Web: http://en.wikipedia.org/wiki/Bugtracker.

[3] P.J. Guo, T. Zimmerman, N. Nagappan, and B. Murphy,“Not My Bug! and Other Reasons for Software Bug Report Reassignments,” *Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW 2011)*, pp. 395-404. New York: ACM Press. 2011.

[4] J. Aranda and G. Venolia, “The Secret Life of Bugs: Going Past the Errors and Omissions in Software Repositories,” *Proc. IEEE 31st International Conference on Software Engineering*, pp. 298-308. Washington, DC: IEEE Computer Society. 2009.

[5] E. Goldberg, “Bug writing guidelines. Retrieved September 3, 2011 from the World Wide Web: https://bugs.eclipse.org/bugs/bugwritinghelp.html.

[6] D. Bertram, A. Voida, S. Greenberg, and R. Walker, “Communication, Collaboration, and Bugs**:** The Social Nature of Issue Tracking in Small, Collocated Teams,” *Proc. ACM Conference on Computer Supported Cooperative Work*, pp. 291-300. New York: ACM Press. 2010. Retrieved September 3, 2011 from the World Wide Web: http://grouplab.cpsc.ucalgary.ca/grouplab/uploads/Publications/Publications/2010-IssueTracking.CSCW.pdf

[7] P. Hooimeijer and W. Weimer, “Modeling Bug Report Quality,” *Proc. of the 22nd International Conference on Automated Software Engineering*, pp. 34–43. New York: ACM Press. 2007.

[8] T. Zimmerman, R. Premraj, J. Sillito, and S. Breu, “Improving Bug Tracking Systems,” *Companion to the 31th International Conference on Software Engineering (ICSE Companion 2009)*, pp.247-250. Retrieved September 3, 2011 from the World Wide Web: http://thomas-zimmermann.com/publications/files/zimmermann-icse-2009.pdf

[9] S Just, R Premraj, and T. Zimmerman, “Towards the Next Generation of Bug Trackers,” *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 82-85. Retrieved September 3, 2011 from the World Wide Web: http://www.cs.vu.nl/~rpremraj/papers/08-vlhcc.pdf

[10] D. Russell and N. Patel, “Increasing Software Engineering Efficiency Through Defect Tracking Integration,” *Proc. International Conference on Software Engineering Advances ‘06*, p. 5. Washington, DC: IEEE Computer Society. 2006.

[11] S. Blair, “A Guide to Evaluating a Bug Tracking System,” *MetaQuest*, 2004. Retrieved September 3, 2011 from the World Wide Web: http://www.metaquest.com/Downloads/Docs/EvaluatingABugTrackingSystem.pdf.

[12] Bugzilla.org, “Homepage,” 2011. Retrieved September 3, 2011 from the World wide web: http://www.bugzilla.org/.

[13] eclipse.org, “About The Eclipse Project,” n.d. Retrieved September 3, 2011 from the World Wide Web: http://www.eclipse.org/.

[14] SourceForge.net., “Roundup Features,” 2011. Retrieved September 3, 2011 from the World Wide Web:http://roundup.sourceforge.net/docs/features.html.

[15] Bugtrack.com, “Web-based bug tracking and project management,” *Homepage*, 2011. Retrieved September 3, 2011 from the World Wide Web: http://www.bugtrack.net/

[16] Wikipedia, “Test Management Tools,” 2011. Retrieved September 3, 2011 from the World Wide Web: http://en.wikipedia.org/wiki/Test_management_tools.

[17] H. Desai, “Test Case Management System,” IEEE Communications: The Global Bridge, vol. 3, pp. 1581-1585 vol. 3, 28 Nov- 2 Dec 1994 1994.

[18] J. Wegener, H. Sthamer, B.F. Jones, and, D.E. Eyres, “Testing Realtime Systems Using Genetic Algorithms,” *Journal of Software Quality*, vol. 6, no. 2, pp. 127-135, June 1997.

[19] A. I. Safana and S. Ibrahim, “Implementing Software Test Management using SpiraTeam Tool, ” *Proc. Fifth International Conference on Software Engineering Advances*, pp. 447-452. Washington, DC: IEEE Computer Society. 2010.

[20] S. Eldh, J. Brandt, M. Street, H. Hansson, and S. Punnekkat, “Towards Fully Automated Test Management for Large Complex Systems,” *Proc. 2010 Third International Conference on Software Testing, Verification and Validation*, pp. 412-420. Washington, DC: IEEE Computer Society. 2010

[21] Hewlett Packard, “HP Quality Center,” n.d. Retrieved September 3, 2011 from the World Wide Web: http://www.hp.com/hpinfo/newsroom/press_kits/2010/ HPSoftwareUniverseBarcelona2010/HP_Quality_Center_data_sheet.pdf

[22] Gurock.com, “Test Case Management Software for QA and Development Teams,” 2011. Retrieved September 3, 2011 from the World Wide Web: http://www.gurock.com/testrail/

[23] MSDN, “Overview of the .NET Framework,” *Microsoft .NET Framework Developer Center*, 2011. Retrieved September 3, 2011 from the World Wide Web: http://msdn.microsoft.com/en-us/library/a4t23ktk.aspx

[24] N. X. Huy, “The Phases of a Software Project,” *Software Engineering*, n.d. Retrieved October 16, 2011 from the World Wide Web: http://www.netnam.vn/unescocourse/se/13.htm