GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

OPEN SOURCE BUG TRACKING SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

SAYNAM JINDAL

(1613101655/16SCSE101433)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mr.Praveen Dominic, Asst. Prof.,

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"OPEN SOURCE BUG TRACKING SYSTEM"** is the bonafide work of "**SAYNAM JINDAL (1613101655)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL
PhD (CS), PhD(Management)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Mr.Tarun Kumar, Asst. Prof.,
Professor
School of Computing Science & Engineering

TABLE OF CONTENTS

List of Figures

Abstract

Bug Tracking for Improving Software Reliability (BTS) is an automated system that can be useful to employees and the managers in any functional organization. Bug Tracking System gives the facility to define the tasks in the organization and also allows the managers to track the bugs spent by the employee for that particular task. A report generation facility is supported in BTS that allows the managers to analyze which are those skills by employee are utilized and those which are not utilized. This tool can help managers for Bug estimation per project or application. This tool helps employees to document their Bugs and analyze

This project aims at creation of a Bug Tracking System. This project will be accessible to all developers and its facility allows developers to focus on creating the database schema and while letting the application server define table based on the fields in JSP and relationships between them. This system provides the following facilities.

The objectives of this system are:

- To keep track of employee skills and based on the skills assigning of the task is done to an employee.

- Employee does bugs capturing. It can be done on daily basis.

Various Reports are generated by this System for an employee and as well as to a manager

Introduction

General

Bug tracking is a system that is indispensable for any system that has to perform well. It is a tool that facilitates fixing of bugs faster and ensures the quality of software being developed or being used. These systems are widely used and they are treated as essential repositories that help in finding status of bugs and quickly resolving them thus the progress of the project can be ascertained. The potentiality and soundness of a good bug tracking system has no parallel in helping the systems to enhance the quality to meet the expectations of clients. Software engineers frequently use bug reports and try to fix them if there is enough information required. In a survey [1] conducted to software engineers of companies like Mozilla, Eclipse, and Apache found that the information items presented in bug reports have played very important role in fixing bugs. Insufficient or improper reports caused delay in fixing bugs and thus causing crossing deadlines. The information items that can be found in the bug reports include screenshots, test cases, expected behavior observed behavior, stack traces and steps to reproduce etc. Generally this is the minimum preferred information needed by engineers to fix bugs easily.

However, the prior research in this area [1] and [2] revealed that the bug reporters omitted these essential information fields in their bug reports making them poorly designed reports. Such reports are of little use to developers for the purpose of fixing bugs. When developers need very descriptive information and the bug reports lack such information, it leads to stalling of the project or delay in completion with other cause and effects. The main problem with current bug tracking systems is that they are simply storing some information fields into databases and not descriptive in nature. Developers expect descriptions beyond doubt and they are not found in the present bug reports. The proposed work in this paper addresses this problem by providing solid and usable steps or directions that can practically improve understandability of information and also ensure that essential data has been captured and stored in bug reports. We also show the results of a prototype bug tracking application and its results and efficiency in helping software developers to understand and fix bugs quickly. The four directions proposed by us are tool oriented improvements, information oriented improvements, process oriented improvements and user oriented improvements. They are elaborated in the later sections.

Vision

The purpose of Bug Tracking for improving software reliability is to provide better service to the administrator or useful for applications developed in an organization.

Scope

The Bug Tracking for Improving Software Reliability is a web based application that can be accessed throughout the organization. This system can be used for logging bugs against an application/module, assigning bugs to team

members and tracking the bugs to resolution. There are features like email notifications, user maintenance, user access control, report generators etc in this system.

Definition, Acronyms, Abbreviations

Bug - A software bug (or just "bug") is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

Overview

Bug tracking is the process of reporting and tracking the progress of bugs from discovery through to resolution, where a bug is defined as a deviation from requirements. Other terminology frequently used to describe this process include

- problem tracking
- change management
- fault management
- trouble tickets

Bug tracking systems are most commonly used in the coding and testing phases of the software development process. However, tracking systems can in fact be used for many other purposes such as general issue tracking, simple task lists, help desk situations or contact management, where the focus is on the tracking

aspect rather than what is being tracked. Even in software development, tracking systems are quite often not limited to simply tracking bugs, but extended to track feature requests or enhancements as well as enquiries.

Literature Review

Software development involves many challenges. One such challenge is the process of bug tracking. Bugs in the process of software development are unavoidable. However, they are to be tracked and fixed with some focus otherwise they are manifested into the final product and that leads to failure of software systems. An error, fault, failure, mistake, flaw can be called as “bug” with respect to a software system. SDLC has many phases. The development team may commit mistakes in any phase. However, when comes to coding it gets surfaced once execution is tested. Most of the bugs can be avoided by having correct analysis and design as per customer requirements. According to [6] bugs in software system cause the system to fail. It does mean that such software system can’t meet the quality and expectations of the client. When software system is able to meet the functional requirements given by the customer, which is said to be a quality product. When quality problem arises, customer gets dissatisfied. According to [7] a bug in software system is not an

accident bug that occurs due to specific reason. Tracking bugs has many phases and it has its own life cycle as shown in fig. 1.

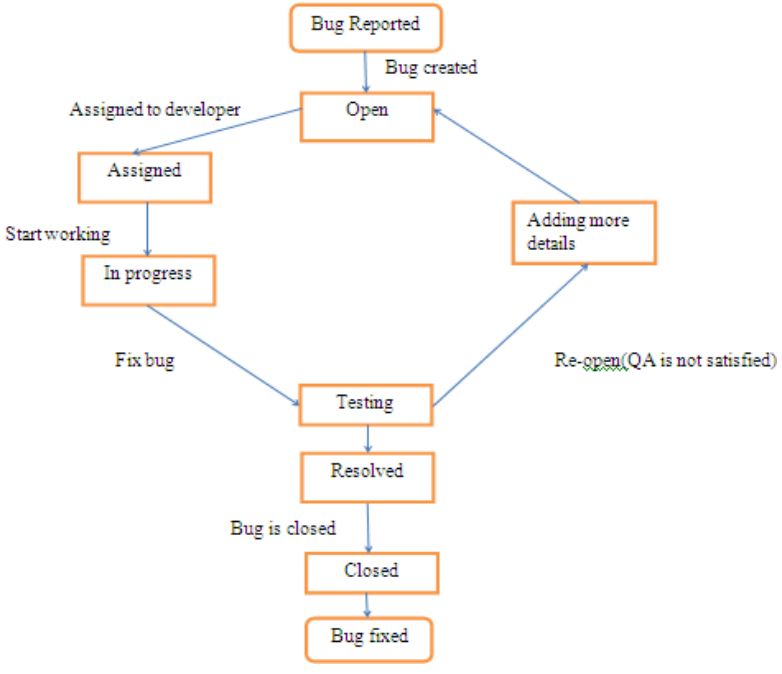


Fig.1- Bug Tracking Life Cycle [9]

As can be seen in fig. 1, an bug tracking is cyclic in nature. Right from its initial finding, until it is resolved or fixed permanently it has its lifecycle with certain phases. As bugs cause systems to act incorrectly bug tracking systems reduce the likelihood of having bugs in the system long time. Quickly the software development team can fix bugs and close them with suitable bug tracking system. Bug tracking improves quality of software being developed. Even after delivering software products bugs may be unearthed and still the tacking system is to be used to fix those bugs faster. When software system is being developed bug tracking is important and also challenging task [14]. Traditionally bug tracking is made simple. It does mean that many years down the line, software developers used spreadsheets to store bug details and fix them. However, using software like spreadsheet is not effective considering modern facilities such as emailing; RSS feeds, and change notification through SMS and so on. Nevertheless, it is very useful to use a spreadsheet

application like Excel for small scale projects with less complexity and less direction. As spreadsheets lack in security and not sophisticated when it comes to database operations, now a day's people are using relational databases to store bug reports and present them through a bug tracking system front end. Bug tracking system helps QA teams in software development cycles.

Software development companies certainly use a bug tracking system as it is essential. Bug tracking system is really required while developing every software product as many developers do not maintain adequate documentation with respect to customer requirements through the entire life cycle of the product [8]. There are many benefits of using a bug tracking system. As specified in [6], clear communication can be provided by a bug database. When compared to informal emails etc. well written reports with standards explain the bugs and the priorities very well. However, it is not easy to track bugs. It is a tedious task as specified in [6]. Many bug tracking systems are already available. They are either open source or commercial. Best bug tracking system among them can be used to fulfill customer requirements. When bug tracking systems are used many factors are to be considered. While using bug tracking system the factors to be observed are application setup, reporting process, and notification method. Before using a bug tracking system, it has to be set up. The well known bug- management application which is open source is "Bugzilla". In many open source products such as Linux, Eclipse, and Mozilla it is being used. There might be configuration problems with open source products. According to [10] free bug tracking systems like Bugzilla take more time to setup and do not appear user friendly. This is because its setup process is very complicated. Users feel it difficult while installing its server.

As specified in [11] Bugzilla is difficult to install and maintain though it has rich set of features. Its user interface is also not inspiring. Besides, for a

novice user it is not an easy task to install and use Bugzilla as observed in [12]. Even though steps are provided to install and use it, it is never easy to get it working for the first time. With respect to setup process it is not an easy task with respect to open source products like Bugzilla. It has to be made simple and easy for novice users as well. With respect to reporting process, bug reporting should be made very simple and efficient bug tracking is expected. The steps in the bug reporting have to be simplified to have smooth communication of bugs among the team of software development. When a web application has so many HTML pages and not tuned to have good navigation, it is difficult to work with such system. When it come to bug tracking on such web applications, problems reoccur as there are navigation problems prevailing in the system and people who are involved in bug tracking gets frustrated with the application. The complexity in such systems can be reduced by working out a smooth navigation and thus developers feel easy to work with bug tracking of such systems. There are some bugs tracking systems that make the reporting process a very tedious task. Instead of reducing burden on people, they really increase it as the reporting process is so complex.

Any bug tracking system can be of two types such as simple and complicated. However, ideally it must be simple and efficient. It should work fast and help people to track and fix bugs so as to develop quality products and satisfy clients. “Bug Genie” is another real world bug tracking system with very good features. However, its bug reporting process is not straightforward and so complicated. Moreover users who are involved in software development will have to add so many things to bug report that makes it much more complex. According to [13] a bug tracking system should help software engineers to effectively track bugs and thus develop a quality product that can satisfy their customers. Another important feature a bug tracking system should have is notifications. Even when software

engineers are not logged into system, they are to be notified about certain events and that must be part of overall workflow of the system. The notifications help team members to understand the changes in bug reports and the timelines to make themselves available. Notifications are part of automated bug tracking system. For instance, when a new bug is reported, all stakeholders pertaining to bug tracking are to be notified immediately. Many bug tracking systems like “BugTracker.NET” and “Flyspray” send mails as notifications. However, spam filters sometimes may block them thus making email notifications unreliable. RSS (Really Simple Syndication) feeds as provided by Web 2.0 technology is another way of making notifications. When users get notifications through RSS feeds, spam filters do not block them. According to [14], content subscribers of RSS get feeds automatically and thus the stakeholders of bug tracking process are aware of notifications. When RSS is used effectively and correctly users are quickly notified about changes and they need not have to be with a system always. This is the reason that a bug tracking system should have facility of RSS feeds that maximize the communication efficiency among the members who are involved in bug tracking. This paper contributes to the research of bug tracking systems yet in another way. It has proposed specific and effective directions to enhance the efficiency of bug tracking systems.

Almost all[3] bug tracking system organize the upcoming bugs i.e. setting the order of the bug and assign it to specific developer. It is a customary follow to outline filters for the anamoly within the system so that the focused bug list can be created, like a list of open crashing bugs or a list of anomalies appointed to a precised developer. Some bug tracker also offers report creation function, usually with the flexibilty to display pie charts and graphs. [4]This is completely necessary for creating quality measures concerning our system, which collectively with code coverage results must be used to direct futher testing efforts more efficiently.

There comes a range of bug tracking system nowadays in the market which are both open source and commercial.[11] Bug-tracker are important for any kind of organization which maintains large scale to medium scale systems. Bug tracking systems just document the process through which a anomaly report goes. The important data generated by this process is only obtained by the end of the process and usually isn't documented. The most important data that's to be achieved after the identification of a defect is the cause of the problem that occured. [5]The outdated manual system was messed up with a sequence of flaws, since the complete system had to be compelled to be controlled with hands, controlling and restoring the data was terribly annoying and prolonged, the record were never in an organized order.

When a system picks up a anomaly, it can conceive many problems. This can be notablly bothersome to client who may be inadequate to attain numerous areas of the system or to execute precise operations on the system due to a tint in the system. An obliging tool is essential to help the situation. It is more asserting to commence tracking bugs during the software building stage because builders are not very supportive in tracking down their flaws [7]. They acknowledge that it imperil both their creativity and competent ego.

Earlier clients described bugs by sending electronic mail to related technical department. This is a bit troublesome to keep track of the flaw reported as the emails are dispersed around email app. In this type of cases, the easiest organised way to keep track of a limited number of bugs is to use a spreadsheet program like excel [4].

As project devlops, the inaugural complication that occurs is that only one person can change the spreadsheet at a given time, clients are not able to modify

the spreadsheet all at once and it is not effective [9]. Another complication is that, if the server or computer functioning the spreadsheet crashes at the time of adjustment of bug, then the information can be conflicting. The reality is that it is approximately difficult to supervise bugs in simple spreadsheets, word documents or remember everything in one's head when it concerns a big scale project.

According to the analysis done by Black, R 1999, he settled that a bug tracking database aids clear connection concerning bugs.

Software engineers usually involves in repairing bugs within the system, the amount of time they consume on that will be diminished and the quality of the software will also increase, using the adequate bug tracking system.[5] This fact persuade us to make a conceptual framework which gives what is needed and improves the whole bug tracking system.

Software building companies assuredly use a bug tracker as it is effective. Bug tracker is actually required while building each software product as a lot of builders do not organize adequate documentation with respect to clients needs through the complete life cycle of the product[8].

System Analysis

Existing System

- The existing system consists of entering the details in the Microsoft Excel Sheets for the storing of the data. When a manager needs information of the employee he searches for the specified file in the file system. He opens the file and takes the information. Report Generation done manually by copying the content of the different files into another file. The Manually generated report was then printed.

Limitations in Existing System

- Information retrieval is a very big process.
- Lack of organization of the files may porn to information loss due to accidental deletion of files.

- No security because the files are visible to the users.
- Report generation will be a big task.

Proposed System

The Proposed system is a browser which is completely related to online system, which provides the centralized database. It stores bugs data and description of the particular bug data. It can also create Excel reports and PDF documents based on the information in its database.

The method of deciding the earliest point in the design and implementation of a code that a bug was generated is termed as root-cause analysis, performing root-cause analysis is very crucial for many reasons. [12]Firstly it keeps the one from repairing symptoms. Secondaly it authorize to enhance the capacity of evaluation methods for the phase of development cycle in which the bug was made known. Finally, it simplifies the search of similar bugs faster in the future. [11]The performance and function alloted to the software as a segment of system engineering are processed by creating a complete data description, a detailed behavioral and functional, an implication of design constraints and performance requirements, suitable verification criteria and other information relevant to requirements. Statistical summaries of the root-cause analysis can also be very helpful. It will tell you wherever you would like to improve your skills or use further additional protective programming ways to refrain the matter in the future.

The projected system is graphical user interface based which is very easy to use and all the inputs to be taken are easily understood even to a non expert. The system is designed at a block level. [13]The blocks are generated on the idea of research done on the trouble recognization, completely other blocks have to be compelled to be generated for various dissimilar functions priority is placed on

decreasing the data flow among the blocks. [7]This project is formed with using relational database management system (RDBMS) that uses MySQL for all the transaction statements.

Apart from that, notification is a very significant feature in a bug finder as the users can not log into a system all the time. Notifications are indeed a workflow feature as it helps in tracking and managing bugs. Notifications helps to keep team members well informed about any vital alterations to bugs, like modifications in precedency or addition of newly finded information. System can inform development heads once new bugs are introduced and are assigned to them.

Really simple syndication(RSS) feeds offers different ways to keep users up to date, client need RSS reader to accept coming feeds and the feeds won't be obstructed by spam filter [6]. Rss is a distribution channel to furnish content to subscribers and it will raise the awareness of users concerning on upgraded problems in the system. If RSS is employed appropriately users will simply and rapidly acquire the upgraded content at once while not having to spent all the time staying on the system. Fullfiling every anomaly report is a list of important facts like the edition of the software, the operating system edition on the device, any joined external hardware and the nature of the hardware and anything deemed pertinent. I'm suggesting a framework for the bug tracking system which has two extra features I.e. automatic bug assignment and duplicate bug detection, it automatically assigns the newly occured.

Advantages over Existing System

- The performance is increased due to well designed database.

- Security is increased
- Time saving in report generation
- Easy to update the details

Feasibility Study

Economic Feasibility

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses.

There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study.

This is because, .at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

Software Requirement Specification

Software Requirements

The minimum software requirement specifications for developing this project are as follows:

Operating System	:	Windows XP/ Windows 7/8
User Interface	:	HTML, CSS
Client-side Scripting	:	JavaScript
Programming Language	:	Java
Web Applications	:	JDBC Servlets, JSP

Database : Oracle/Access

Hardware Requirements

The minimum hardware requirement specification for developing this project is as follows:

Processor : Pentium IV

Hard Disk : 40GB

RAM : 256MB

System Design

Software Design

The main focus of the analysis phase of Software development is on “What needs to be done”. The objects discovered during the analysis can serve as the framework or Design. The class’s attributes, methods and association identified during analysis must be designed for implementation language. New classes must be introduced to store intermediate results during the program execution.

Emphasis shifts from the application domain of implementation and computer such as user interfaces or view layer and access layer. During analysis, we look at the physical entities or business objects in the system, that is, which players and how they cooperate to do the work of the application. These objects represent tangible elements of the business.

During the Design phase, we elevate the model into logical entities, some of which might relate more to the computer domain as people or employees. Here his goal is to design the classes that we need to implement the system the difference is that, at this level we focus on the view and access classes, such as how to maintain information or the best way o interact with a user or present information.

AUTHENTICATION

Functional Description

a. Login to the system through the first page of the application.
b. Change the password after login to the application.
c. See his/her details and change it.
d. Help from the system.

Maintenance

Functional Description

- User Maintenance: Creating, Granting & Revoking access and deleting users from application.

- Component Maintenance: Creating a component (application being developed / enhanced), Granting & Revoking access on components to Users and Marking a component as “Active” or “Closed”.
- Bug Tracking: Creating, Assigning Bugs to users, Modifying and Closing a Bug. A Bug screen should at least have following details
    - Bug Number and Title
    - Bug priority
    - Date created

- Find User: A search screen to find users and display results
- Find component: A search screen to find components and display results
- Find Bug: A search screen to find Bugs and display results
- Report: Generate reports on Bugs

Modules:

Authenticate User: The Bug Tracking System first activates the login form. Here the user enters the User name and password and our system starts the authentication process in which the username and password are matched with the existing username and password in the database. If the password matches then it is allowed to the main page else it warns the user for Invalid User name and password. After successful authentication the system activates menus. The activity log also prepared for failures and security.

Admin: This module has the entire access to all other modules, admin creates the project and assigning the projects to the created manager, adding members to the managers, assigning bugs based on the priority. Can update the manager,

members and access to the particular project data. Generating reports based on the managers report submission.

Manager: Manager has the full access to the particular project assigned by the admin and controls the team members access to the bugs assigned. Has the permission to generate the reports and update the information of team members and adding members to the project.

Developer: Can access the task or bug assigned by the manager, view assigned projects and resolving the assigned bug. Developer can view the bugs list assigned by the manager.

Tester: Tester can access to the projects or bugs assigned by the manager, can view the assigned projects and can add a new bug to the list and send the bug back to the manager. Tester can login to the system and access the assigned projects list.

Reports: Both Admin and Manager can access this module and generate the reports based on the requirements.

Bug Details

- **Bug Details** - In this module the user is provided with the facility for adding bugs or updating the existing bugs. As the number of bugs for a product can be very large this system is provided with efficient filtering. The user can filter the bugs based on the priority, database, operating system and status. After the user applies filter the list of bugs are displayed from the database.
- **Bug History** - Here the bug history is maintained. All the solutions given for the bug resolution by various users are stored. As the bug needs several techniques or methods for resolution it is important to store the history of the bug.
- **Bug Assignee** - This displays the list of users for whom the bug is assigned for resolution. As the bug need to be resolved for completing the product

several user are assigned to find a solution for the bug. The user can add this bug to a new user or he can modify the existing user details.

• **Bug Attachments** - This gives a list of attachments for a particular bug. The bug can be of any type it can be a database bug or a GUI bug. So while you add a bug you need to provide with the details of bug. So the file attachments can be a document, database file or an image file. All then attachments are stored in a location along with the size and type of the file. Here the user can add a new attachment or can change the details of existing files.

Bug Tracking

• **Track Hierarchy** -All the bugs saved in the database will have a particular hierarchy. There might be bugs which can be related to the earlier bugs saved in thedatabase so our system is provided with a hierarchy. And user can add child nodes in this hierarchy or he can modify the existing values ofthe nodes. This hierarchy is based on the parent child relation ship between the bugs.

• **Track Resolution** - This displays a list of all solutions provided by the users allotted to a bug. This stores the action type and the necessary resolution providedby the user.

• **Track Resources** - This displays list of resources allotted to the project. As the bugs need to be resolved resources are provided for the bugs. These Resourceswill be the resources allotted to the project. The resources are allotted based on the rating of the employee.

View

• **Product Bug Hierarchy** - This module is just for displaying the hierarchy for the easy Look of the bugs. Here the bugs are displayed in the form of parent child nodes. As it is difficult for the user to look at the vast number of bugs in the database. And one cannot easily access the relation between the bugs.

• **Product User Hierarchy** - This module if for displaying the users allotted to the bug. The users along with their name and designation are displayed in this

module. Even in the allotment of resources there can be hierarchy between the employees depending on their designation. So this module simplifies the hierarchy among the employees.

Functions

- ✓ Admin
- ✓ Manger
- ✓ Developer
- ✓ Tester
- ✓ Reports

Design process:

During the design phase the classes identified in object-oriented analysis Must be revisited with a shift focus to their implementation. New classes or attribute and Methods must be an added for implementation purposes and user interfaces.

The following are some of the vies of software design life cycle. They are

- UML Diagrams
- Data Flow Diagrams
- Data Base Design

UML Diagrams

Unified Modeling Language

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

This UML diagrams must include the following:

- Class diagram
- Interaction Diagram
- Use case Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram

Class Diagram

Bug Tracking System Class Diagram describes the structure of a Bug Tracking System classes, their attributes, operations (or methods), and the relationships among objects. The main classes of the Bug Tracking System are Bugs, Projects, Managers, Testers, Developers, Bug Types.

Classes of Bug Tracking System Class Diagram:

- **Bugs Class** : Manage all the operations of Bugs
- **Projects Class** : Manage all the operations of Projects
- **Managers Class** : Manage all the operations of Managers
- **Testers Class** : Manage all the operations of Testers
- **Developers Class** : Manage all the operations of Developers
- **Bug Types Class** : Manage all the operations of Bug Types

Classes and their attributes of Bug Tracking System Class Diagram:

- **Bugs Attributes** : bug_id, bug_developer_id, bug_tester_id, bug_title, bug_type, bug_description

- **Projects Attributes** : project_id, project_developer_id, project_tester_id, project_name, project_assign, project_last_date, project_type, project_description
- **Managers Attributes** : manager_id, manager_name, manager_mobile, manager_email, manager_username, manager_password, manager_address
- **Testers Attributes** : tester_id, tester_name, tester_mobile, tester_email, tester_username, tester_password, tester_address
- **Developers Attributes** : developer_id, developer_name, developer_mobile, developer_email, developer_username, developer_password, developer_address
- **Bug Types Attributes** : bug_type_id, bug_type_title, bug_type_description

Classes and their methods of Bug Tracking System Class Diagram:

- **Bugs Methods** : addBugs(), editBugs(), deleteBugs(), updateBugs(), saveBugs(), searchBugs()
- **Projects Methods** : addProjects(), editProjects(), deleteProjects(), updateProjects(), saveProjects(), searchProjects()
- **Managers Methods** : addManagers(), editManagers(), deleteManagers(), updateManagers(), saveManagers(), searchManagers()
- **Testers Methods** : addTesters(), editTesters(), deleteTesters(), updateTesters(), saveTesters(), searchTesters()
- **Developers Methods** : addDevelopers(), editDevelopers(), deleteDevelopers(), updateDevelopers(), saveDevelopers(), searchDevelopers()
- **Bug Types Methods** : addBug Types(), editBug Types(), deleteBug Types(), updateBug Types(), saveBug Types(), searchBug Types()

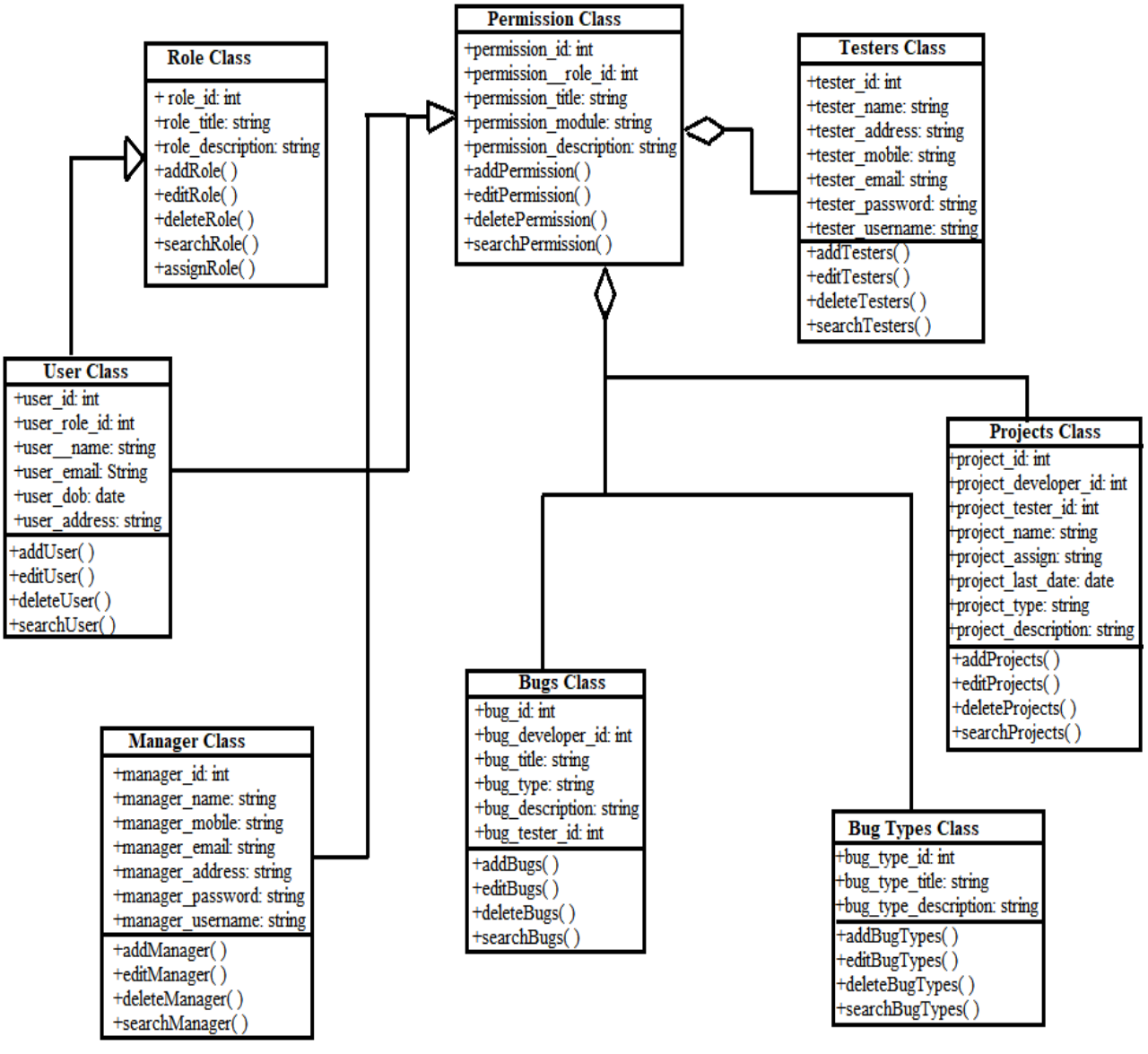


Fig-2 Class Diagram

Use-Case Diagram

This Use Case Diagram is a graphic depiction of the interactions among the elements of Bug Tracking System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of Bug Tracking System. The main actors of Bug Tracking System in this Use Case

Diagram are: Super Admin, System User, Developer, Tester, who perform the different type of use cases such as Manage Bugs, Manage Projects, Manage Managers, Manage Testers, Manage Developers, Manage Bug Types, Manage Users and Full Bug Tracking System Operations. Major elements of the UML use case diagram of Bug Tracking System are shown on the picture below.

The relationships between and among the actors and the use cases of Bug Tracking System:

- **Super Admin Entity** : Use cases of Super Admin are Manage Bugs, Manage Projects, Manage Managers, Manage Testers, Manage Developers, Manage Bug Types, Manage Users and Full Bug Tracking System Operations
- **System User Entity** : Use cases of System User are Manage Bugs, Manage Projects, Manage Managers, Manage Testers, Manage Developers, Manage Bug Types
- **Developer Entity** : Use cases of Developer are View Bugs, Change Status, Add Comments
- **Tester Entity** : Use cases of Tester are Create Bugs, Assign Bugs, Bug Reports

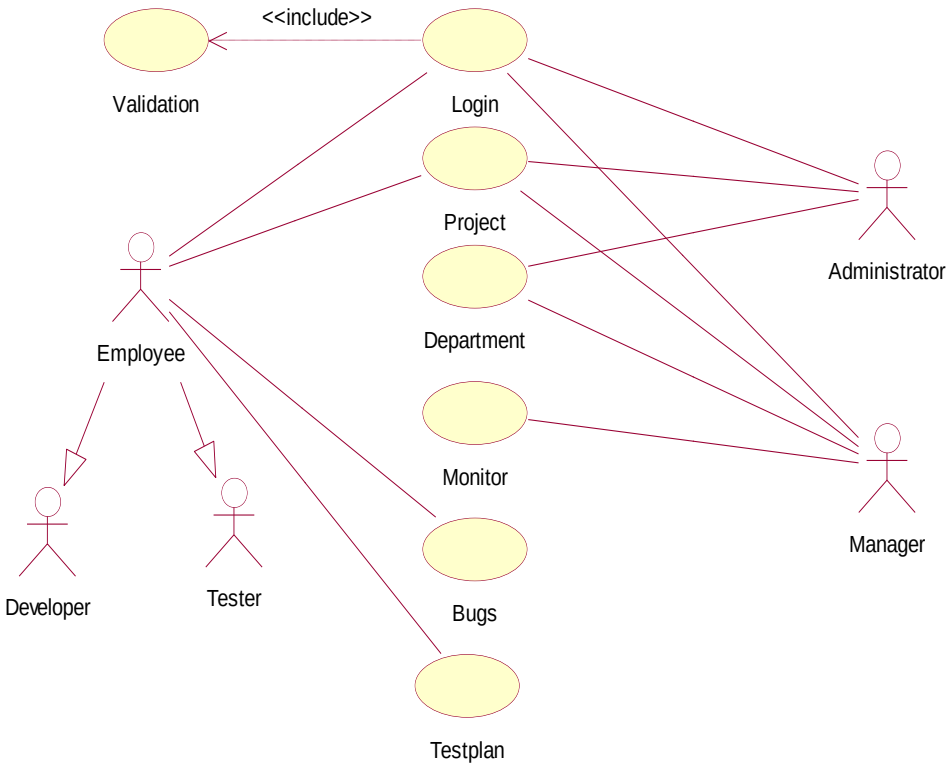


Fig-3.1 Overall Use case Diagram

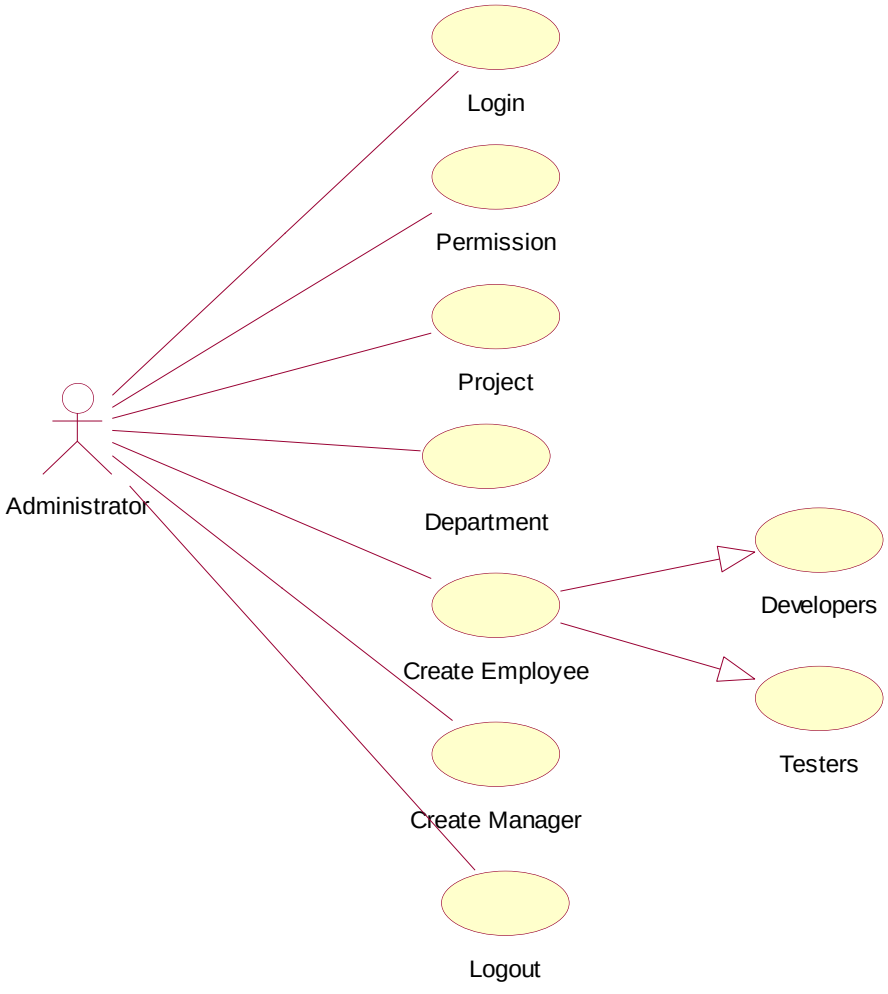


Fig-3.2 Administrator Use case

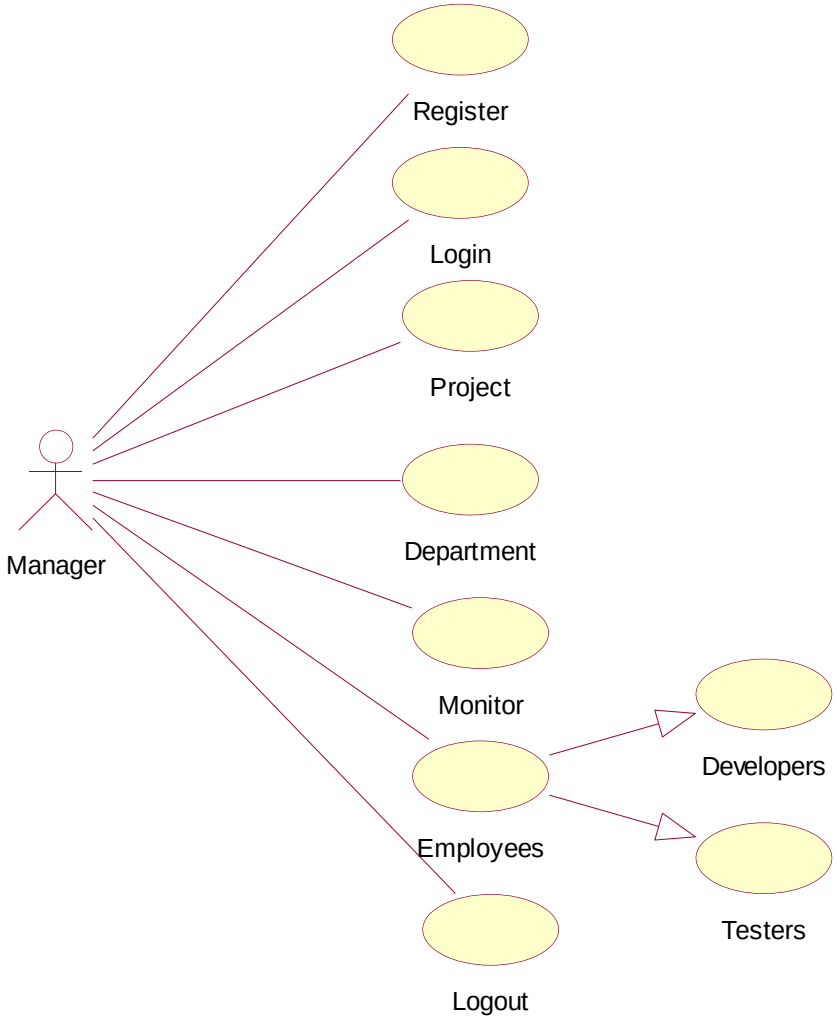


Fig-3.3 Manager Use case

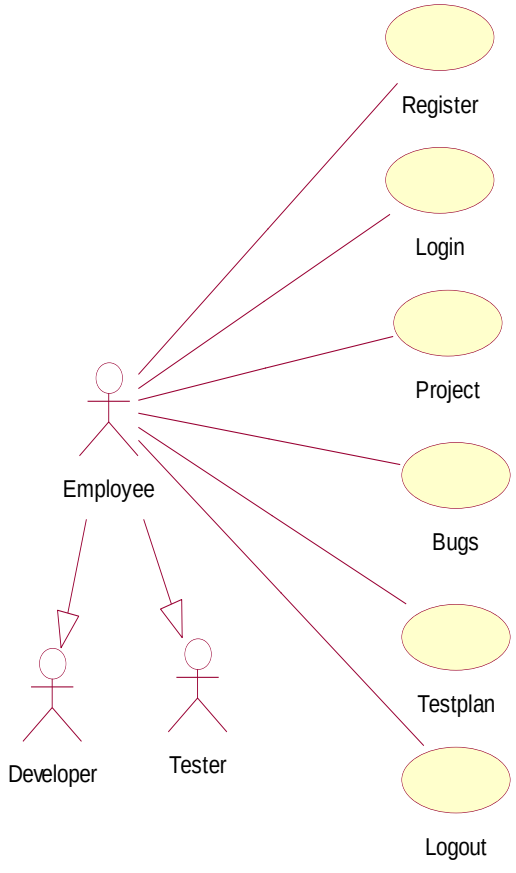


Fig-3.4 Employee use case

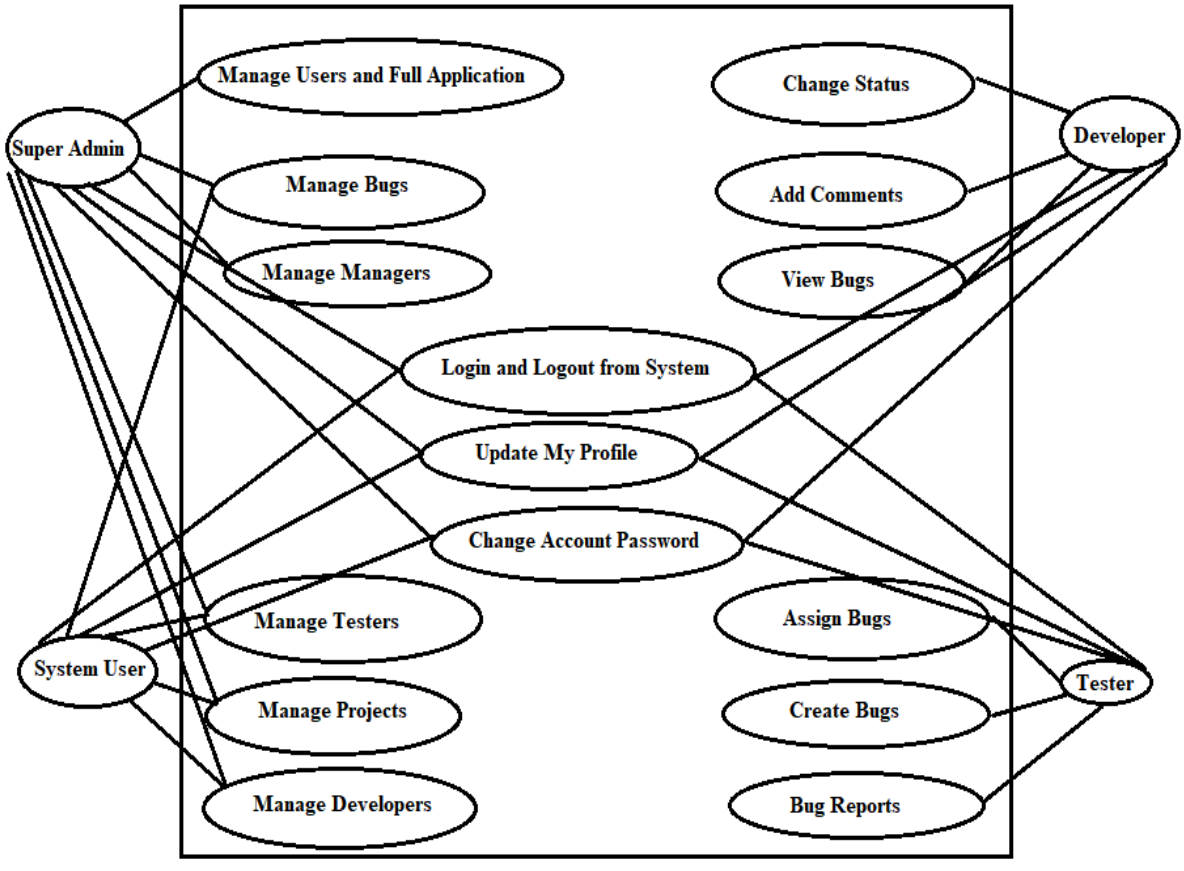


Fig-3.5 Use case Diagram

Sequence Diagram

This is the **UML sequence diagram of Bug Tracking System** which shows the interaction between the objects of Managers, Testers, Bug Types Projects, Developers, . The instance of class objects involved in this UML Sequence Diagram of Bug Tracking System are as follows:

- Managers Object
- Testers Object
- Bug Types Projects Object
- Developers Object
- Object

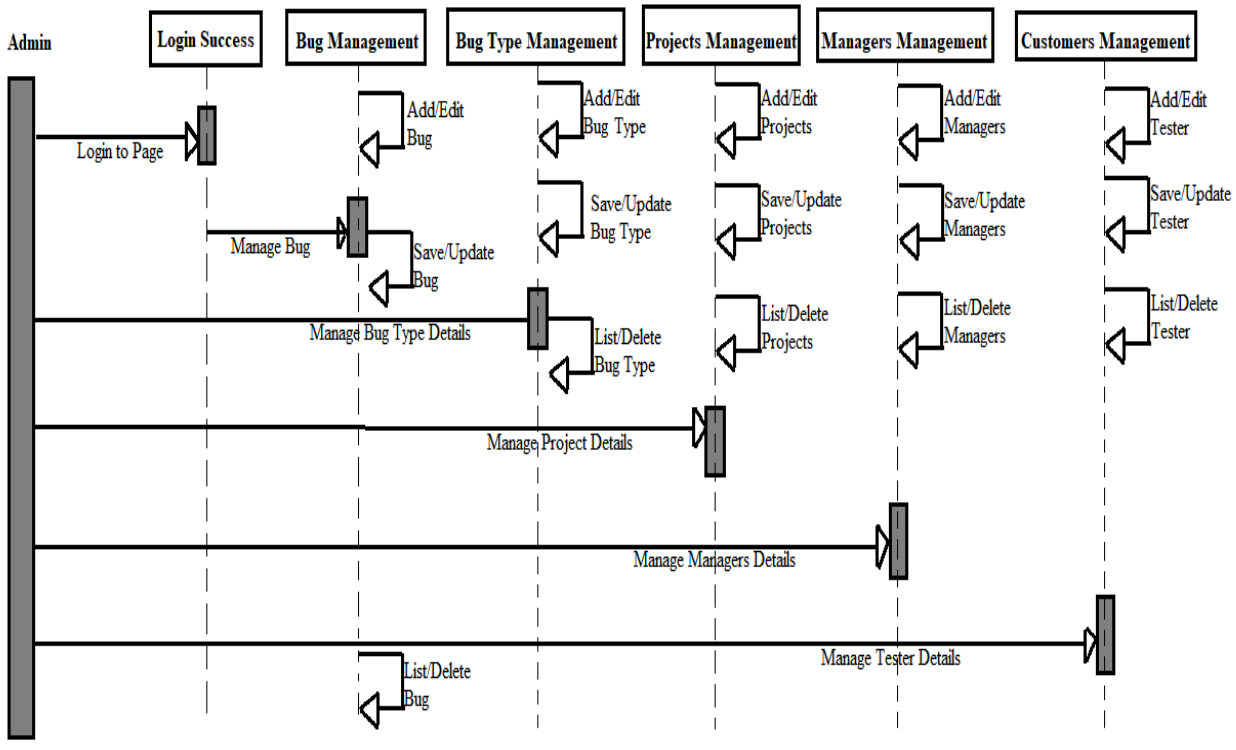


Fig-4.1 Sequence Diagram

Login Sequence Diagram of Bug Tracking System:

This is the **Login Sequence Diagram of Bug Tracking System**, where admin will be able to login in their account using their credentials. After login user can manage all the operations on Bug Types Projects, Managers, Testers, , Developers. All the pages such as Testers, , Developers are secure and user can access these page after login. The diagram below helps demonstrate how the login page works in a Bug Tracking System. The various objects in the , Bug Types Projects, Managers, Testers, and Developers page—interact over the course of the sequence, and user will not be able to access this page without verifying their identity.

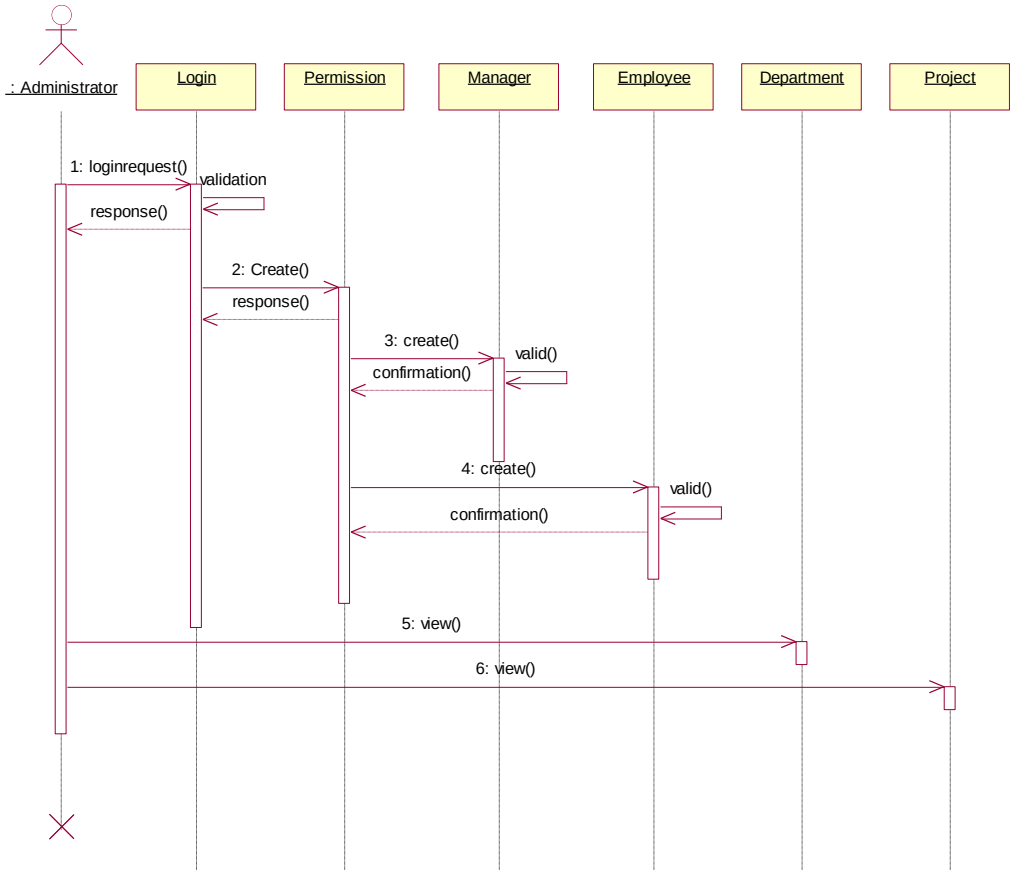


Fig-4.2 Administration Sequence Diagram

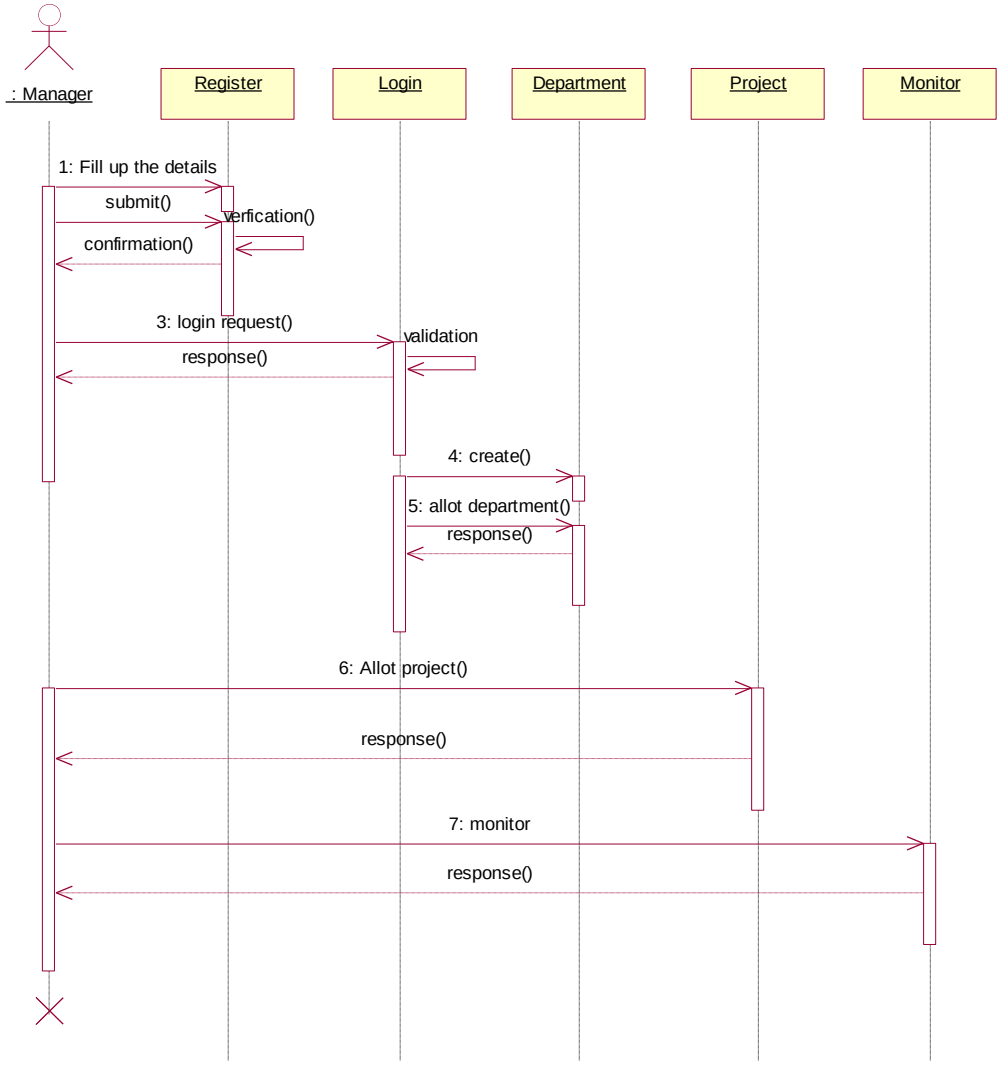


Fig-4.3 Manager Sequence

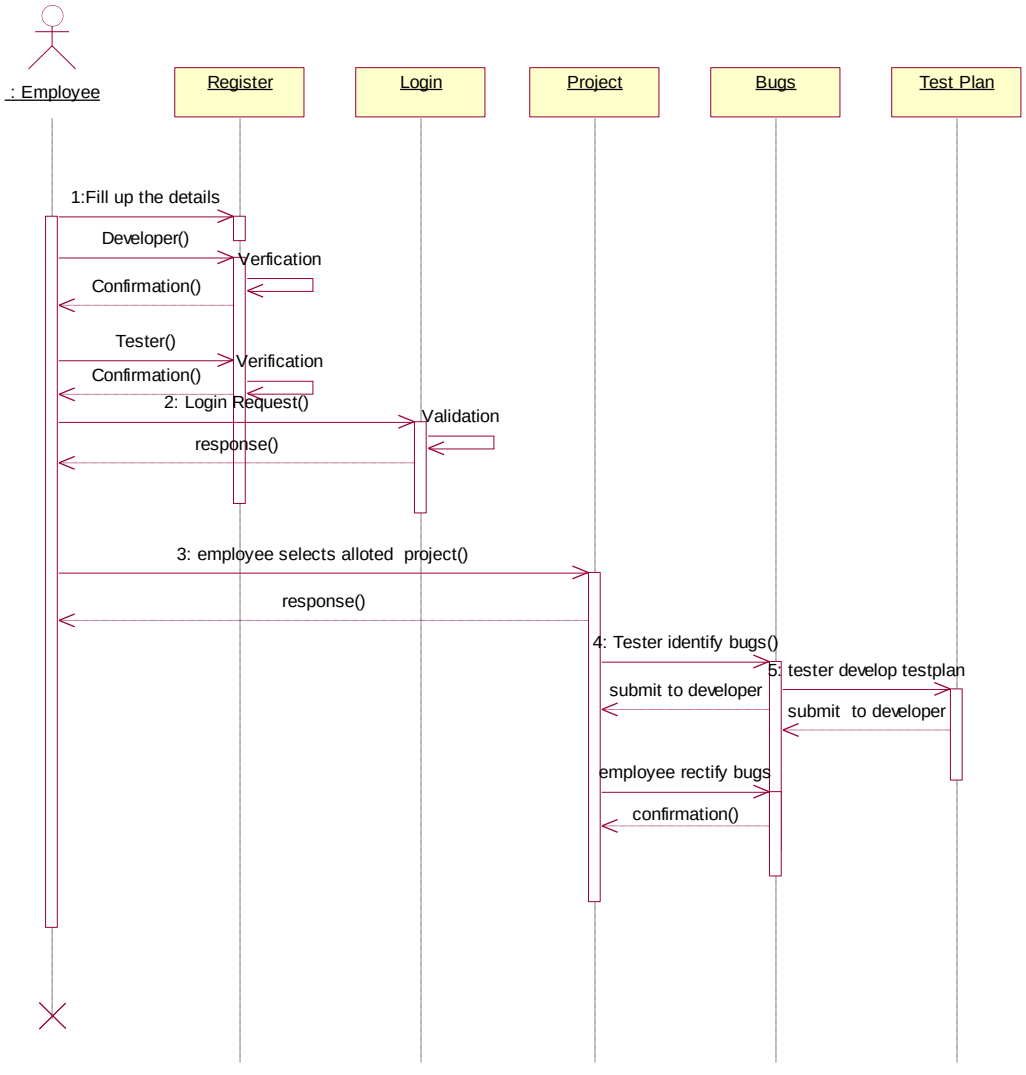


Fig-4.4 Employee Sequence

Collaborations Diagram

Collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all its parts.

Collaboration is also the specification of how an element, such as a classifier or an operation, is realized by a set of classifiers and associations playing specific roles used in a specific way

Contents

Collaboration diagrams commonly contain the following:

- Objects
- Links
- Messages

Like all other diagrams, sequence diagrams may contain notes and constrains.

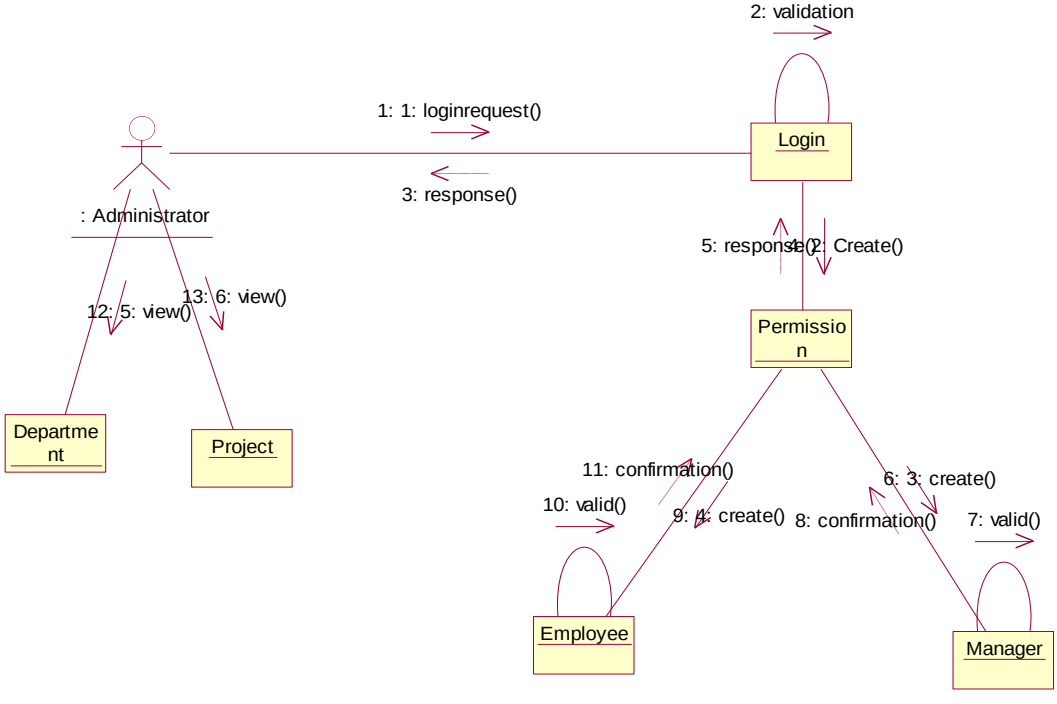


Fig-5.1 Administrative Collaboration

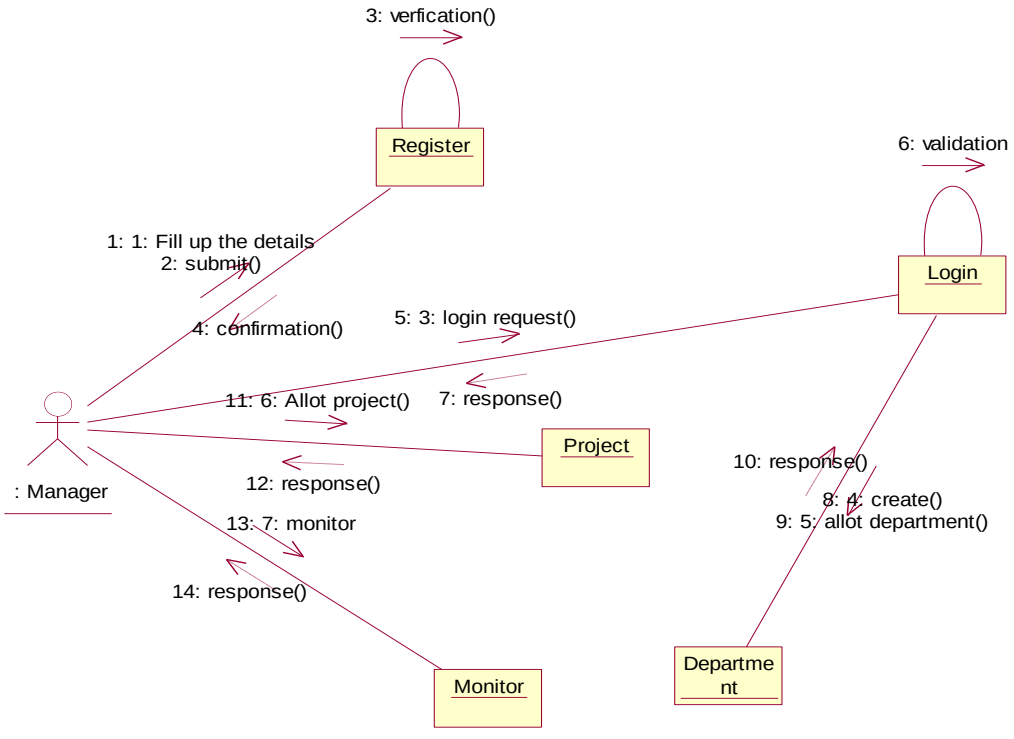


Fig-5.2 Manager Collaboration

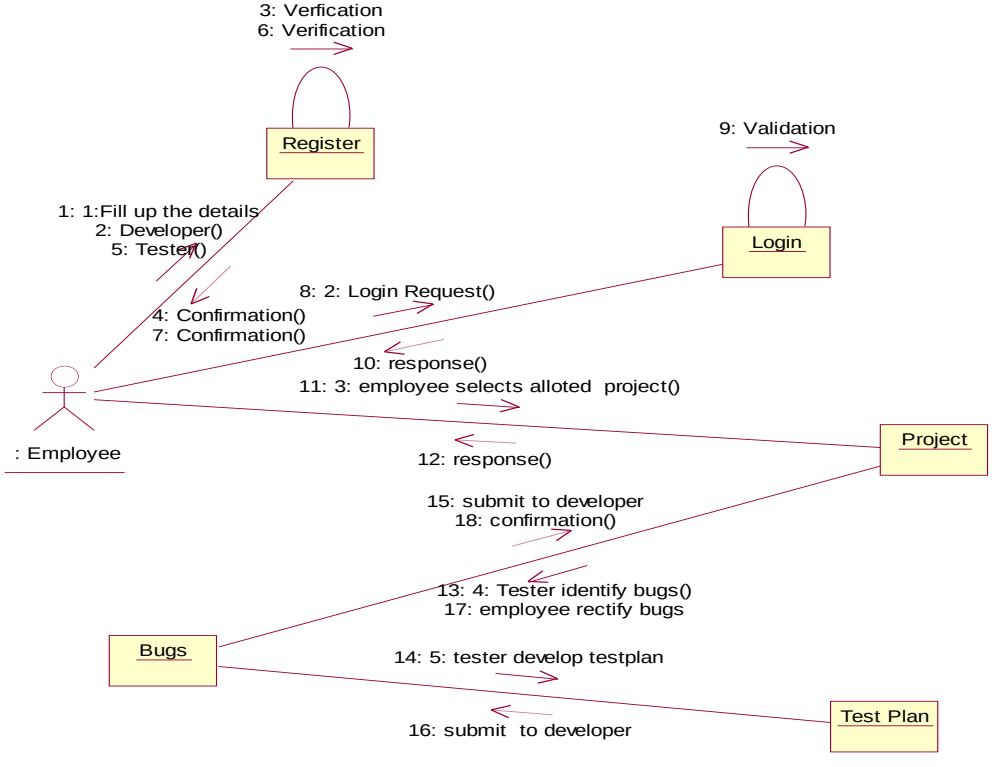


Fig-5.3 Employee collaboration

Component Diagram

This is a **Component diagram of Bug Tracking System** which shows components, provided and required interfaces, ports, and relationships between the , Bug Types Projects, Bugs, Managers and Testers. This type of diagrams is used in Component-Based Development (CBD) to describe systems with Service-Oriented Architecture (SOA). **Bug Tracking System UML component diagram**, describes the organization and wiring of the physical components in a system.

Components of UML Component Diagram of Bug Tracking System:

- Component
- Bug Types Projects Component
- Bugs Component
- Managers Component
- Testers Component

Featues of Bug Tracking System Component Diagram:

- You can show the models the components of Bug Tracking System.
- Model the database schema of Bug Tracking System
- Model the executables of an application of Bug Tracking System
- Model the system's source code of Bug Tracking System

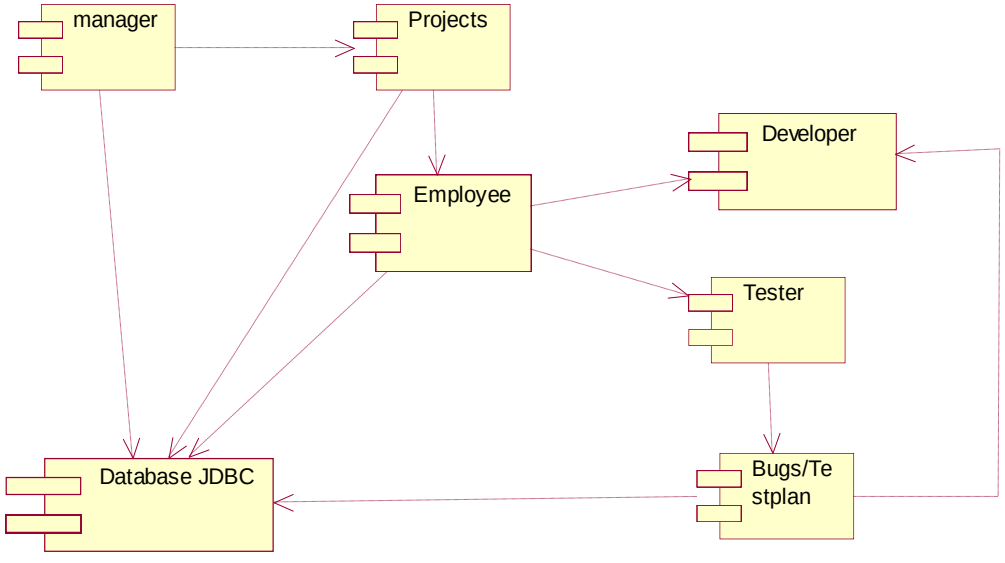


Fig-6 Component Diagram

Deployment Diagram

A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.

Graphically, a deployment diagram is collection of vertices and arcs.

Contents

Deployment diagram commonly contain the following things:

- Nodes
- Dependency and association relationships
- Like all other diagrams, deployment diagrams may contain notes and constraints.
- Deployment diagrams may also contain components, each of which must live on some node.
- Deployment diagrams may also contain packages or subsystems, both of which are used to group elements of your model into larger chunks.

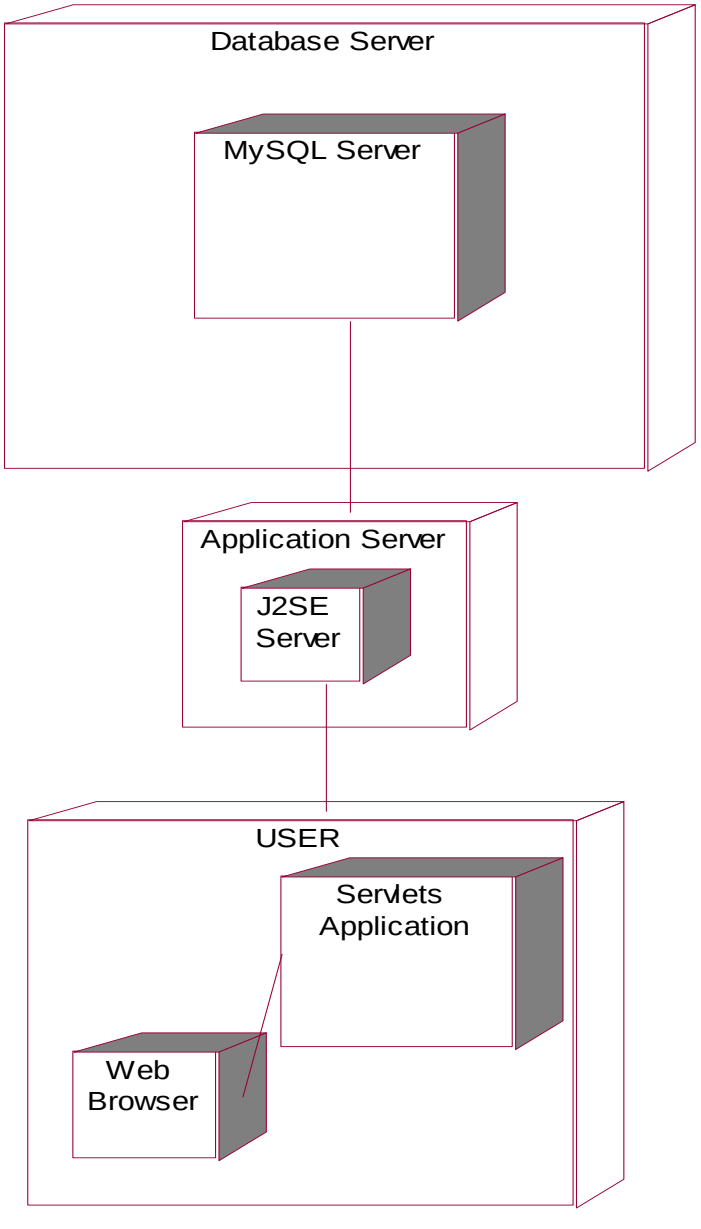


Fig-7 Deployment Diagram

Activity Diagram

This is the **Activity UML diagram of Bug Tracking System** which shows the flows between the activity of , Bug Types Projects, Managers, Testers, Bugs.

The main activity involved in this **UML Activity Diagram of Bug Tracking System** are as follows:

- Activity
- Bug Types Projects Activity
- Managers Activity
- Testers Activity
- Bugs Activity

Features of the Activity UML diagram of Bug Tracking System

- Admin User can search , view description of a selected , add , update and delete .
- Its shows the activity flow of editing, adding and updating of Bug Types Projects
- User will be able to search and generate report of Managers, Testers, Bugs
- All objects such as (, Bug Types Projects, Bugs) are interlinked
- Its shows the full description and flow of , Testers, Bugs, Managers, Bug Types Projects

Fig-8.1 LOGIN PROCESS

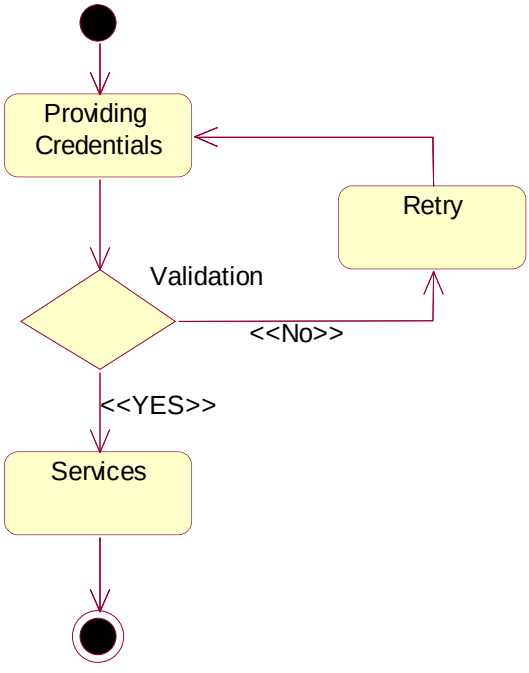


Fig-8.2 Registration Process

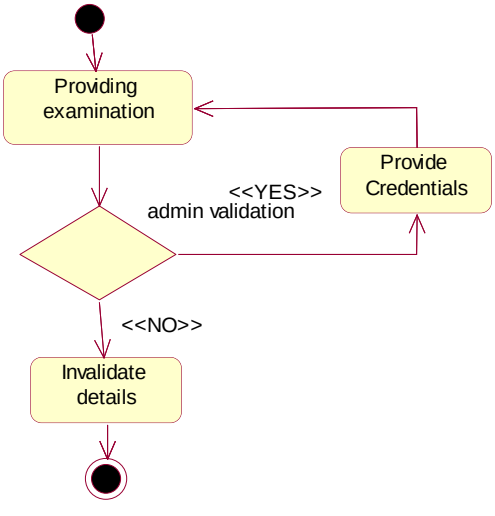


Fig-8.3 Manager Process

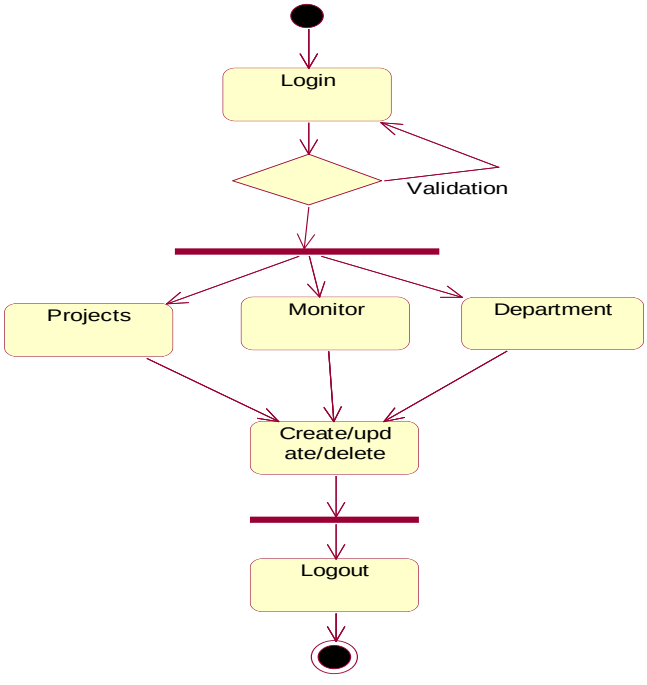


Fig-8.4 Administrator Process

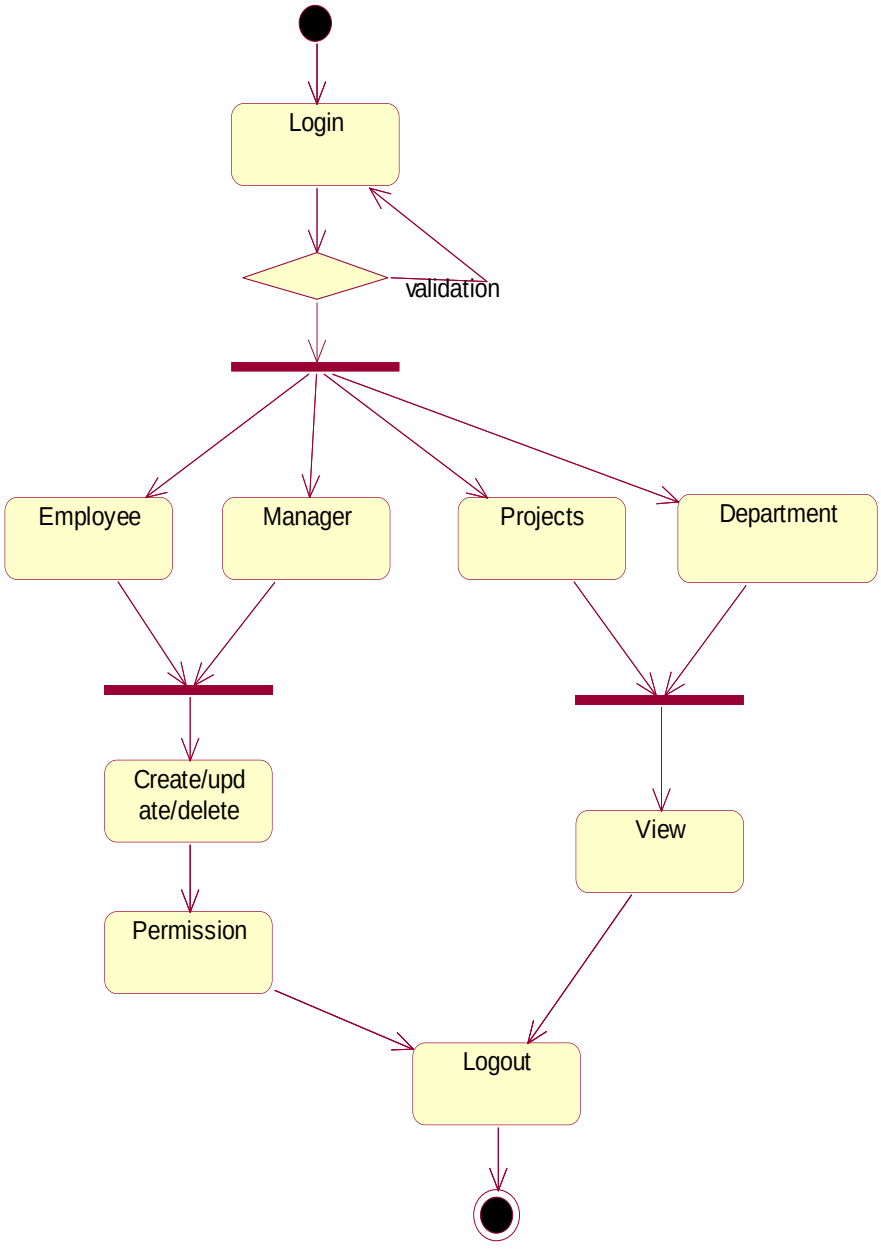
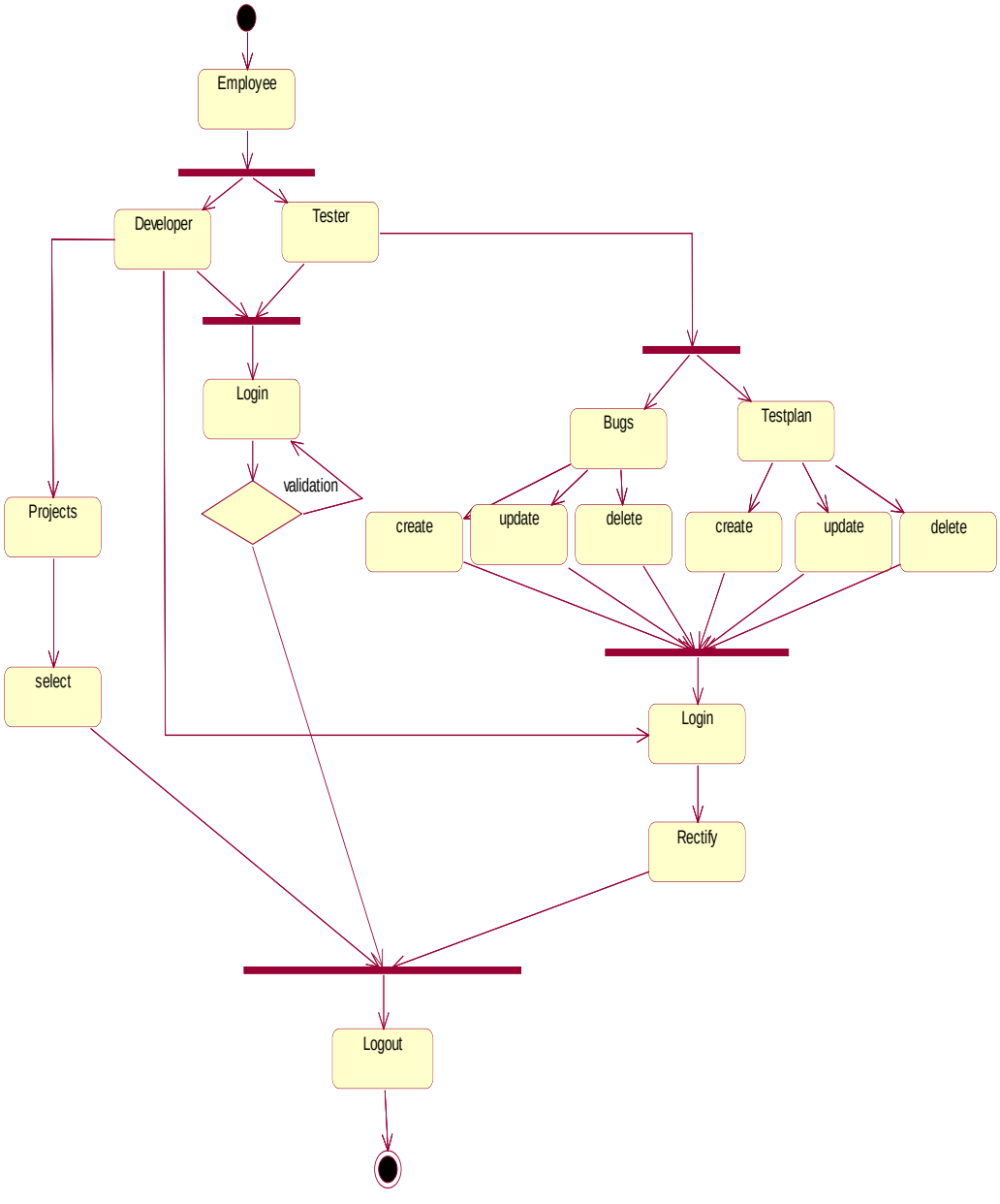

Login
validation
Employee
Manager
Projects
Department
Create/upd
ate/delete
View
Permission
Logout

Fig-8.5 Employee Process



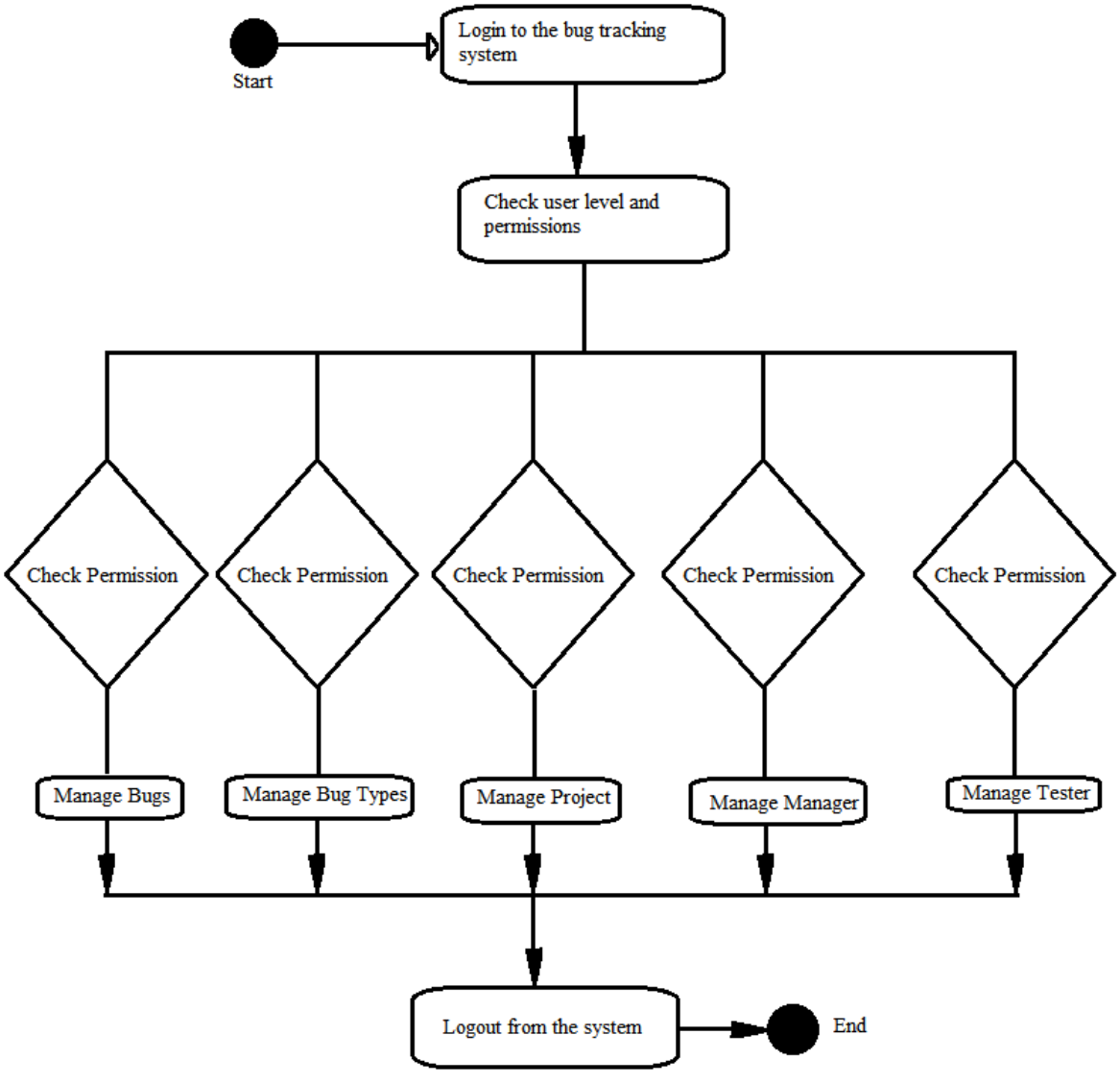


Fig-8.6 Activity Diagram

Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system. It can also be used for the visualization of data processing (structured design).

There are two types of DFDs. They are:

- Context Level DFD
- Top Level DFD

Context Level DFD

In the Context Level the whole system is shown as a single process.

- No data stores are shown.
- Inputs to the overall system are shown together with data sources (as External entities).
- Outputs from the overall system are shown together with their destinations (as External entities).

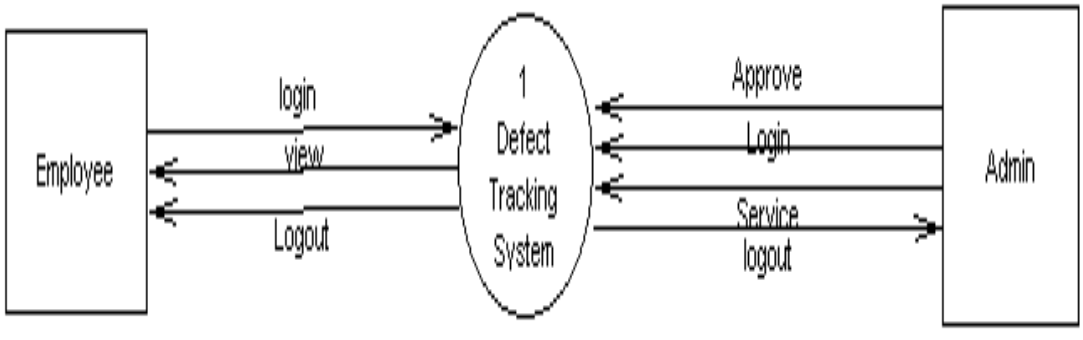


Fig-9 Context Level DFD

Fig-10.1

BTS - TOP LEVEL DIAGRAM

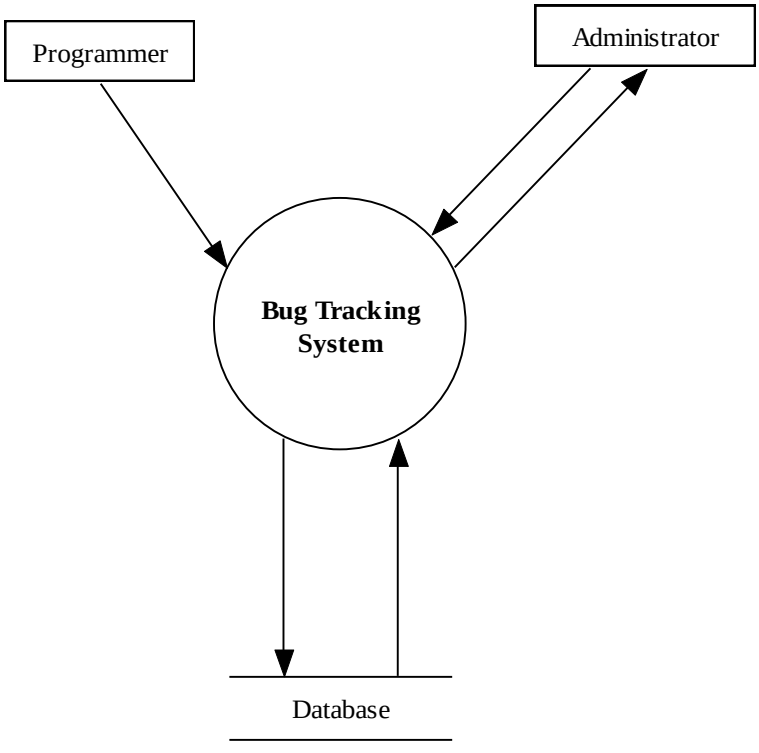


Fig-10.2

BTS - TOP LEVEL DIAGRAM

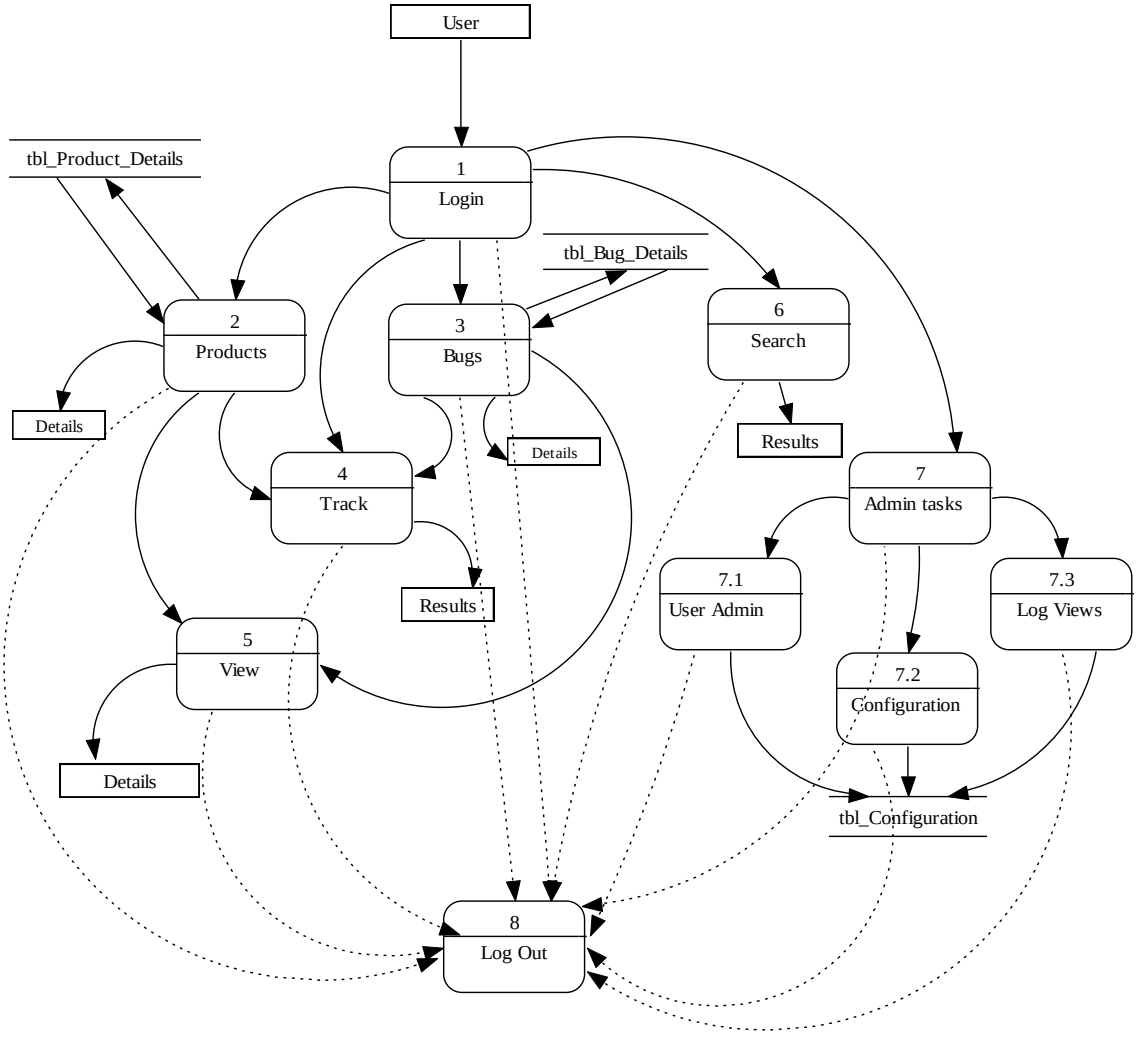


Fig-10.3

LOW LEVEL DIAGRAM - PRODUCTS

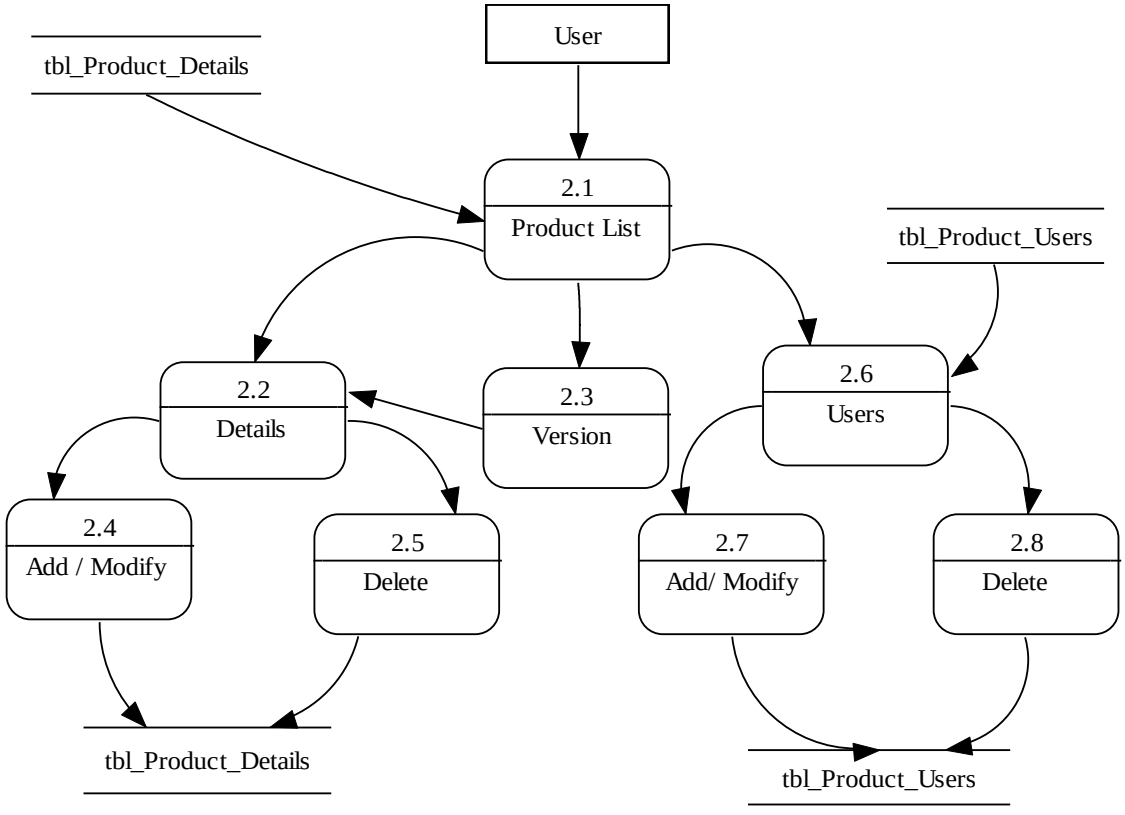


Fig-10.4

LOW LEVEL DIAGRAM - BUGS

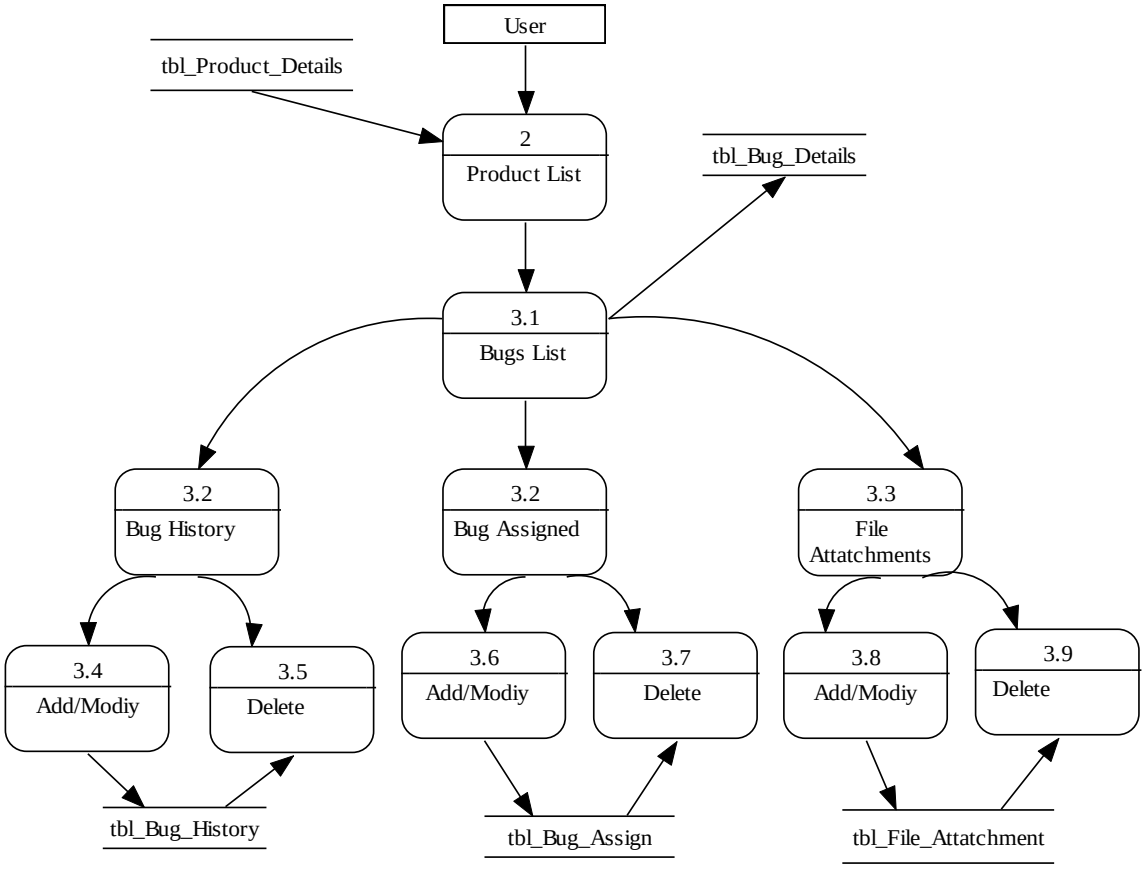


Fig-10.5

LOW LEVEL DIAGRAM - TRACKING

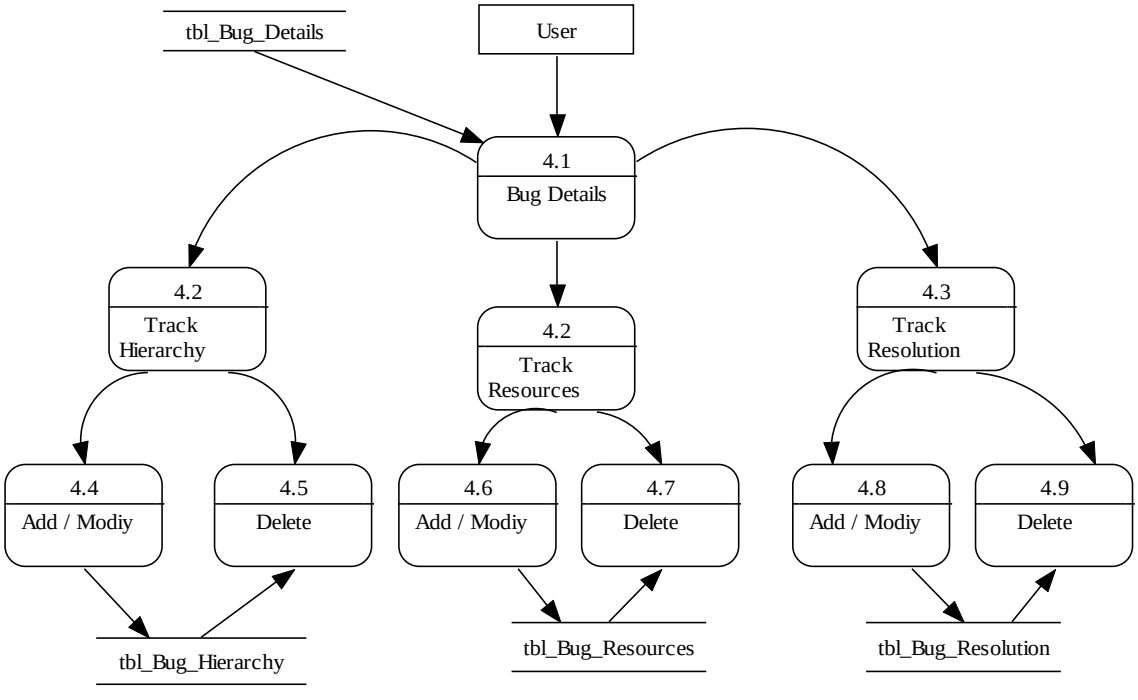


Fig-10.6

LOW LEVEL DIAGRAM - VIEW

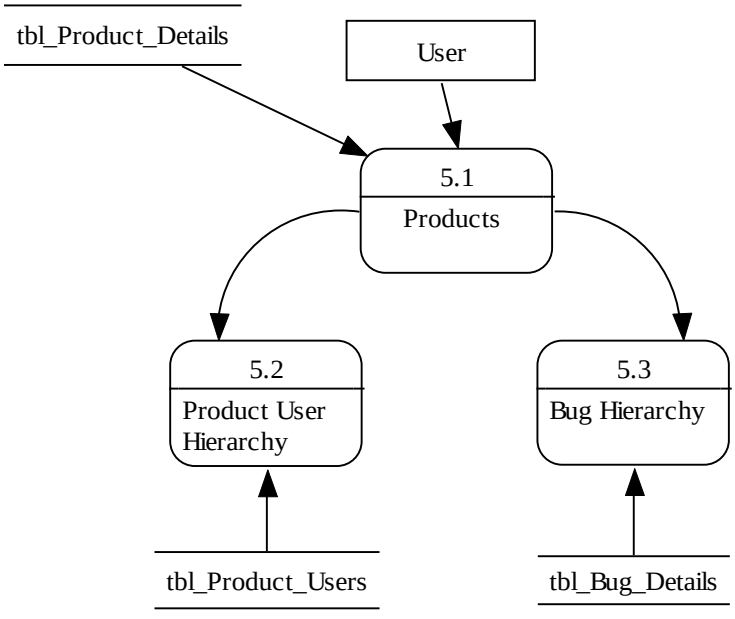


Fig-10.7

LOW LEVEL DIAGRAM - SEARCH

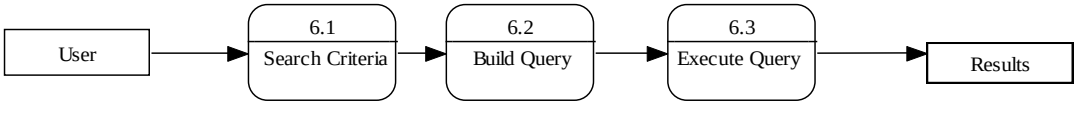
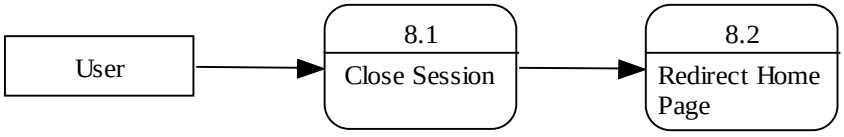


Fig-10.8

LOW LEVEL DIAGRAM - LOGOUT



Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views.

In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

E-R Diagram

This ER (Entity Relationship) Diagram represents the model of Bug Tracking System Entity. The entity-relationship diagram of Bug Tracking System shows all the visual instrument of database tables and the relations between Projects, Testers, Bugs, Bug Types etc. It used structure data and to define the

relationships between structured data groups of Bug Tracking System functionalities. The main entities of the Bug Tracking System are Bugs, Projects, Managers, Testers, Developers and Bug Types.

Bug Tracking System entities and their attributes :

- **Bugs Entity** : Attributes of Bugs are bug_id, bug_developer_id, bug_tester_id, bug_title, bug_type, bug_description
- **Projects Entity** : Attributes of Projects are project_id, project_developer_id, project_tester_id, project_name, project_assign, project_last_date, project_type, project_description
- **Managers Entity** : Attributes of Managers are manager_id, manager_name, manager_mobile, manager_email, manager_username, manager_password, manager_address
- **Testers Entity** : Attributes of Testers are tester_id, tester_name, tester_mobile, tester_email, tester_username, tester_password, tester_address
- **Developers Entity** : Attributes of Developers are developer_id, developer_name, developer_mobile, developer_email, developer_username, developer_password, developer_address
- **Bug Types Entity** : Attributes of Bug Types are bug_type_id, bug_type_title, bug_type_description

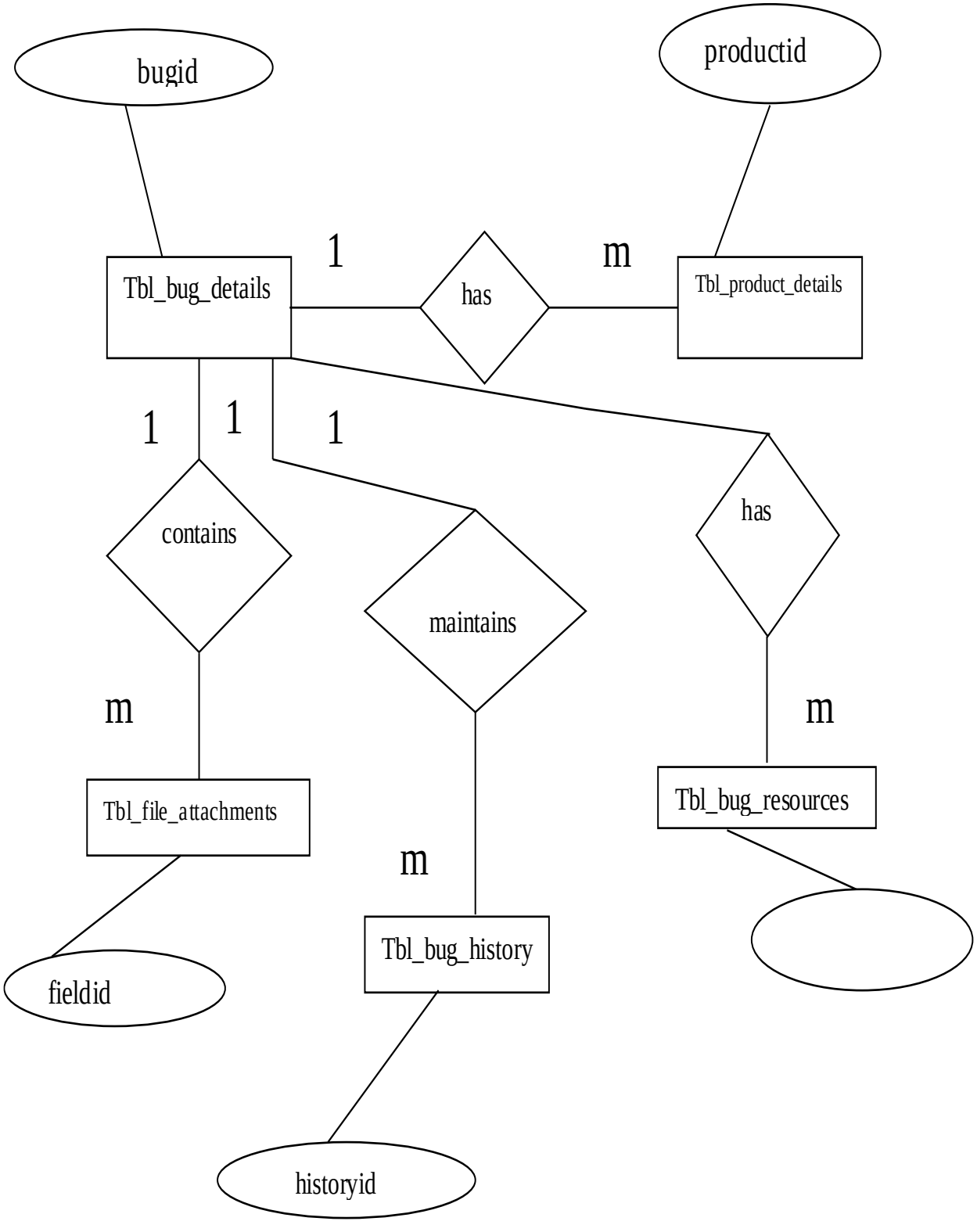
Description of Bug Tracking System Database :

- The details of Bugs is store into the Bugs tables respective with all tables
- Each entity (Bug Types, Managers, Developers, Projects, Bugs) contains primary key and unique keys.

- The entity Managers, Developers has binded with Bugs, Projects entities with foreign key
- There is one-to-one and one-to-many relationships available between Developers, Testers, Bug Types, Bugs
- All the entities Bugs, Developers, Managers, Bug Types are normalized and reduce duplicacy of records
- We have implemented indexing on each tables of Bug Tracking System tables for fast query execution.

Fig-11 E-R diagram

BugTracking



Data Dictionary

The data dictionary is used to create and store definitions of data, location, format for storage and other characteristics. The data dictionary can be used to retrieve the definition of data that has already been used in an application. The data dictionary also stores some of the description of data structures, such as entities, attributes and relationships. It can also have software to update itself and to produce reports on its contents and to answer some of the queries. Determining the Information Requirement The sole purpose of the MIS is to produce such information which will reduce uncertainty risk in a given situation.

The difficulty to determine a correct and complete set of information is on account of the factors given below:

1. The capability constraint of the human being as an information processor, a problem solver and a decision-maker.

2. The nature and the variety of information in precise terms.

3. Reluctance of decision-makers to spell out the information for the political and the behavioural reasons.

4. The ability of the decision-makers to specify the information. In spite of these difficulties, methods are evolved based on the uncertainty scale, starting from the low to the high level of uncertainty. If the uncertainty is low, seeking information requirement or needs is easy as against a very high level of uncertainty.

There are four methods of determining the information requirements. They are:

- Asking or interviewing
- Determining from the existing system
- Analysing the critical success factors
- Experimentation and modelling.

In this method a designer of the MIS puts questions or converses with the user of the information and determines the information requirements. Putting the questions is an art and it should be used properly to seek information. When the user has to select one answer from a finite set of answers a closed question should be asked. For example, "Which are the raw materials used for making a product?"But an open question is put, when the user has no precise knowledge but has an ability todetermine all answers to select one out of them? For example, "Which are the raw materials which can be used in a product?" In open questions, the answers may not be immediate but can be obtained by surveying the domain knowledge of the user. When multiple users or several decision-makers in similar functions or positions are involved, a brain storming session is performed to cover all possible answers to the questions. When several users are involved, group consensus can be sought to get the most feasible set of answers.

In a number of cases the existing system, which has been evolved after a number of years, and has been designed out of experience gives straightaway the requirement of information. In any situations, systems from other companies can give additional information requirements.

EFFECTIVENESS

Software engineers often involve in fixing bugs in the system under development. The time they spend on that can be reduced and the quality of software can be increased by using an effective bug tracking system. The initial information on a bug can help the engineers to resolve bugs faster thus saving development time and cost. Over a period of time development team can reduce lot of wastage of time with regard to tracking and fixing bugs with a sophisticated bug tracking system in hand. This fact motivates us to build a conceptual framework that provides required directions that can improve bug tracking systems. Out proposed framework is as given in fig.

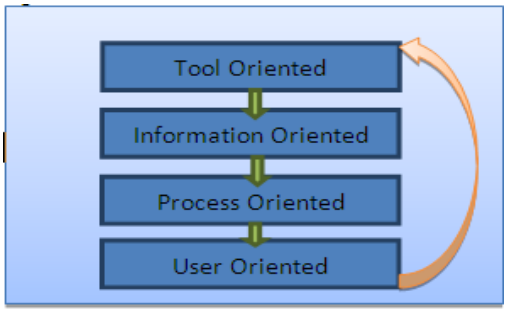


Fig-12

Proposed Directions to Enhance Bug Tracking Systems

As can be seen in fig. 12, the directions proposed are tool oriented, information oriented, process oriented and user oriented in nature. They are iterative directions that can be cycled back after each iteration.

Tool oriented

We recommend a bug tracking system to be tool oriented. Tool oriented does mean that the bug tracking systems can be configured to collect stack trace implicitly and add it to the report that contains bug details. It can improve the information collection capabilities by doing so. It can also use steps to make use of macro recorders, capture/replay tools that can be observed by software engineers later. From the observations, it is possible to capture test cases and fix the bugs easily [3]. Tool oriented

feature can enhance the capability of bug tracking system in terms of information collection that is quite relevant and can be readily used to fix bugs in the system. This will lead the effective tracking and fixing of bugs those results in quality of the software besides being productive. This direction can helps in transition into information oriented feature.

Information oriented

This is another direction from us that helps software developers to have improved focus on the collection of information that has to be kept in bug reports. Towards this end some sort of tools like Cuezilla [4] can be embedded into bug tracking systems. Such tools verify the information provided in bug reports and provide real world feedback that helps improve the quality of information. This will help software engineers to get motivated and have more focused work in solving or fixing bugs. Being information oriented with tool support can enhance the possibilities of checking whether the stack traces reported are complete and consistent in the given scenario. This direction when followed by real world bug tracking systems can lead to make it more process oriented.

Process oriented

This direction is meant for process oriented improvements. The process being followed in bug tracking systems can be improved further using this direction. Is does mean that all administrative activities pertaining to bug tracking and fixing come under process oriented feature. The process oriented bug tracking systems can also focus on the developer who is made responsible to fix bugs besides having a comprehensive bug report [1], [4]. Other advantages of process oriented feature are that developers can have better awareness on bug reports and thus they are aware of possible actions to be taken; it also helps developers to estimate time to be spent on specific bugs and schedule their time accordingly. Process oriented feature can influence user-oriented feature.

User oriented

Users do mean the developers and also bug reporters. This direction focuses on educating the reporters so as to enable them to collect correct information and how to collect it as well. This training helps both developers and reporters. The expected information in the bug report makes the developers to grasp it faster and act quickly to fix bugs in real time applications. This direction can influence the adaption of new tools in the process thus making it more robust and productive in nature.

The above mentioned directions provide a systematic approach in improving bug tracking systems. To support this we have built a prototype application that makes use of all directions and enhance the capabilities of bug tracking systems in the real world.

QUESTIONS IN BUG REPORT

When any software engineer presents a bug report, most probably, he is asked many questions. Some of them are what is the name of the product? What is the bug? in which component is the bug?; in which module is the bug?; in which method the bug is?; in which environment the bug arises?; in which platform the application is built?; in which OS the application runs?. The information given by developer who report bug might be incomplete initially. For this reason follow up questions required. They include do you have .NET framework installed? Can you provide a screenshot of the bug? Research revealed that the software developers may answer 2 questions out of three [5]. When a bug report is submitted by a developer, the follow up questions are to be asked immediately besides keeping the submitted bug report in hand. We recommend software development teams to have bug tracking systems that contain “build expert systems”. These systems ask all required questions to software engineer so as to make the work automated. The question to be given and answered by

developer is not static. The questions do not come sequentially. Moreover answer to a question determines the next possible question. Narrowing down the location of bug and to have accurate bug descriptions are features of expert. The following data is essential in order to build an expert system.

- Bug location information which is crucial to tacking bugs. Location gives you the line number, method, class and so on. This helps developers to move to that place with ease. Many software development environments (IDEs) allow the bugs to be located just by a click of button or a click.
- From the bugs list, machine learning models can be built that choose questions and also predict the location of the bug based on the responses that corresponds to the bugs.

  This paper provides a proof of study that makes use of data that is present in the bug reports. Thus we get collection of information that is essential in implementing a tool that can support automatic evaluation of the information.

Initial Experiment

We have done an experiment with Eclipse where a set of questions are supposed to predict the accurate location of a bug that can be fixed. We used around 20 most fixed files in the SDK of Eclipse and selected all bug reports of them. From those bug reports, more than 2500 cases are considered and a decision tree is built on the prediction of location of the bug fix of the name of the file. The following questions are used for which answers exist in the bug database of Eclipse.

- What is your name?
- What is the operating system?
- Which version of Eclipse is used?

- What is the bug?
- How severe the bug is?
- Which component is affected by the bug?
- What is the priority to be given to the bug?
- Which platform is affected?

The result of this experiment is a decision tree where the root node shows the defect distribution among all files. The decision tree helps in locating the file which contains the reported bug. We have built an application to demonstrate good features of a bug tracking system that follows the four directions discussed in this paper.

Main UI of Prototype Application

The application facilitates various stakeholders of the project to record bug details and track them leading to quick and efficient bug fixes. The tool is able to cater various groups of the software development personnel.

CODING AND IMPLEMENTATION

General

A programming tool or software tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task. The Chapter describes about the software tool that is used in our project.

JavaScript

JSP not only enjoys cross-platform and cross-Web-server support, but effectively melds the power of server-side Java technology with features of static HTML pages.

JSP pages typically comprise of: Static HTML / XML components.

-Special JSP tags.

-Optionally, snippets of code written in the java programming language called "script lets."

- **JSP Advantages**

Separation of static from dynamic content: In JSP, the logic to generate the dynamic content is kept separate from the static presentation templates by encapsulating it within external Java beans components. When a page designer makes any changes to the presentation template, the JSP page is automatically recompiled and reloaded into the web server by the JSP engine.

Write Once Run Anywhere: JSP technology brings the "Write Once, Run anywhere" paradigm to interactive Web pages.

Dynamic content can be served in a variety of formats: There is nothing that mandates the static template data within a JSP page to be of a certain format.

XML

PROJECT DESCRIPTION:

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
    <name>bugTrackingSystem</name>
    <comment></comment>
    <projects>
    </projects>
    <buildSpec>
        <buildCommand>

    <name>com.genuitec.eclipse.j2eedt.core.WebClasspathBuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>org.eclipse.jdt.core.javabuilder</name>
            <arguments>
```

```
            </arguments>
        </buildCommand>
        <buildCommand>

    <name>com.genuitec.eclipse.j2eedt.core.J2EEProjectValidator</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>

    <name>com.genuitec.eclipse.j2eedt.core.DeploymentDescriptorValidator</
name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>org.eclipse.wst.validation.validationbuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>

    <name>com.genuitec.eclipse.ast.deploy.core.DeploymentBuilder</name>
```

```
            <arguments>
            </arguments>
        </buildCommand>
    </buildSpec>
    <natures>

    <nature>com.genuitec.eclipse.ast.deploy.core.deploymentnature</nature>
        <nature>com.genuitec.eclipse.j2eedt.core.webnature</nature>
        <nature>org.eclipse.jdt.core.javanature</nature>
        <nature>org.eclipse.wst.jsdt.core.jsNature</nature>
    </natures>
</projectDescription>
```

JAVA TECHNOLOGY

LOGIN:

```
import java.io.IOException;
```

```
import javax.servlet.*;
import javax.servlet.http.*;

import connect.ConnectionProvider;

import java.sql.*;
public class Login extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        doPost(request, response);

    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        String loginid=request.getParameter("loginid");
        String password=request.getParameter("password");
        String type=request.getParameter("type");
```

```
ServletContext ctx=getServletContext();


boolean flag=false;
try{



   Connection con=ConnectionProvider.getConnection();


    System.out.println("Connection is got  "+con);
        Statement stmt = con.createStatement();
    String query = "select * from login where loginid ='" + loginid + "'
and type='"+type+"'";
    ResultSet rs = stmt.executeQuery(query);
    while(rs.next())
    {
        String pass= rs.getString("password");
        if( pass.equals(password)) {
            flag=true;
            break;
        }
    }
    if (flag){
```

```
                HttpSession session=request.getSession();
                session.setAttribute("login", "yes");
                session.setAttribute("type", rs.getString("type"));
                session.setAttribute("user", rs.getString("name"));
                session.setAttribute("loginid", rs.getString("loginid"));
                RequestDispatcher rd=null;
                if("Admin".equals(type))
                    rd=request.getRequestDispatcher("admin.jsp");
                else if("Expert".equals(type))
                    rd=request.getRequestDispatcher("expert.jsp");
                else if("User".equals(type))
                    rd=request.getRequestDispatcher("user.jsp");

                rd.forward(request, response);
            }
            else{
                RequestDispatcher
rd=request.getRequestDispatcher("login.jsp?msg=loginerror");
                rd.forward(request, response);
            }
        }catch(Exception e)
```

```
        {
            e.printStackTrace();
        }

    }

}
```

SIGNUP:

```
import java.io.IOException;
import java.sql.*;

import javax.servlet.*;
import javax.servlet.http.*;

import connect.ConnectionProvider;

public class SignUp extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
```

```
            throws ServletException, IOException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        String name=(String)request.getParameter("name");
        String email=(String)request.getParameter("email");
        String loginid=(String)request.getParameter("loginid");
        String password=(String)request.getParameter("password");
        String repassword=(String)request.getParameter("repassword");


        String type=(String)request.getParameter("type");



    if(""".equals(name)||""".equals(email)||""".equals(loginid)||""".equals(password)|
|""".equals(repassword)||!(password.equals(repassword))){
            if("User".equals(type)){
```

```
            RequestDispatcher
rd=request.getRequestDispatcher("signup.jsp?msg=blank");
            rd.forward(request, response);
            }else{
                RequestDispatcher
rd=request.getRequestDispatcher("addexpert.jsp?msg=blank");
                rd.forward(request, response);
            }


        }else{
            System.out.println("inside else.....");
            try{
                Connection con=ConnectionProvider.getConnection();
                String insertquery = "insert into login values(?,?,?,?,?)" ;
               PreparedStatement stmt = con.prepareStatement(insertquery);
               stmt.setString(1,loginid);
               stmt.setString(2,password);
               stmt.setString(3,name);
               stmt.setString(4,email);
               stmt.setString(5,type);
                 stmt.executeUpdate();
               stmt.close();
```

```
                con.close();
                if("User".equals(type)){
                    HttpSession session=request.getSession();
                    session.setAttribute("login", "yes");
                    session.setAttribute("type", type);
                    session.setAttribute("user", name);
                    session.setAttribute("loginid", loginid);
                    RequestDispatcher
rd=request.getRequestDispatcher("user.jsp");
                    rd.forward(request, response);
                }else{
                    RequestDispatcher
rd=request.getRequestDispatcher("admin.jsp");
                    rd.forward(request, response);
                }

            }
            catch(Exception e){

                if("User".equals(type)){
                    RequestDispatcher
rd=request.getRequestDispatcher("signup.jsp?msg=error");
```

```
                    rd.forward(request, response);
                }else{
                    RequestDispatcher
rd=request.getRequestDispatcher("addexpert.jsp?msg=error");
                    rd.forward(request, response);
                }
            }
        }

    }

}
```

REPORT BUG:

```
import java.io.IOException;

import java.sql.*;
import java.util.*;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import connect.ConnectionProvider;

public class ReportBug extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        String prodid=(String)request.getParameter("prodid");
        String prodname="";
```

```
String env=(String)request.getParameter("env");
String type=(String)request.getParameter("type");
String description=(String)request.getParameter("description");
String authorid=(String)request.getParameter("authorid");
String author=(String)request.getParameter("author");
if("".equals(description)){
    RequestDispatcher
rd=request.getRequestDispatcher("reportbug.jsp?msg=error");
    rd.forward(request, response);
}
else{

try{
    Connection con=ConnectionProvider.getConnection();
   Statement stmt=con.createStatement();
    ResultSet rs=stmt.executeQuery("select max(bugid) from bug");
    int bugid;
    if(rs.next()){
        bugid=rs.getInt(1);
    }else{
        bugid=0;
    }
```

```
            bugid++;
            rs.close();

            rs=stmt.executeQuery("select prodname, version from products
where prodid='"+prodid+"'");
            if(rs.next()){
                prodname=rs.getString(1)+" - "+rs.getString(2);
            }
            rs.close();
            stmt.close();

            String insertquery = "insert into bug values(?,?,?,?,?,?,?,?,?,?,?,?,?)" ;
            PreparedStatement prestmt = con.prepareStatement(insertquery);

            prestmt.setInt(1,bugid);
          prestmt.setString(2,authorid);
          prestmt.setString(3,author);
          prestmt.setString(4,prodid);
          prestmt.setString(5,prodname);
          prestmt.setString(6,env);
          prestmt.setString(7,type);
          prestmt.setString(8,description);
```

```
            Calendar cal = new GregorianCalendar();
            int year = cal.get(Calendar.YEAR);
            int mm = cal.get(Calendar.MONTH);
            int dd = cal.get(Calendar.DAY_OF_MONTH);
            java.sql.Date date=new java.sql.Date(year, mm, dd);
            System.out.println(date);
            prestmt.setDate(9,date);
            prestmt.setString(10,"-");
            prestmt.setString(11,"New");
            prestmt.setString(12,"-");
            prestmt.executeUpdate();
            prestmt.close();
              con.close();

              RequestDispatcher
rd=request.getRequestDispatcher("viewbugs.jsp");
              rd.forward(request, response);

         }catch(Exception e){            System.out.println(e);
              RequestDispatcher
rd=request.getRequestDispatcher("reportbug.jsp?msg=error");
```

```
                rd.forward(request, response);
            }
            }
        }
}
```

CONTEXT LISTENER:

```
package listener;
import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import connect.ConnectionProvider;
import create.CreateTable;

public class MyContextListener implements ServletContextListener
{
    public void contextDestroyed(ServletContextEvent sce)
    {
```

```
        System.out.println("Context is destryoed.....................");
    }

    public void contextInitialized(ServletContextEvent sce)
    {
        ServletContext ctx=sce.getServletContext();

        String path=ctx.getRealPath("tables");

        String permit=ctx.getInitParameter("permit");

        ConnectionProvider.load(path);


            if(permit.equals("yes"))
            {
                CreateTable.create(path);

                System.out.println("********************Table Created
Successfully**********");
            }
            else
```

```
            {
                System.out.println("*************No permistion to create
table***************");
            }
    }
}
```

CREATE TABLE:

```
package create;

import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;

import connect.ConnectionProvider;

public class CreateTable
{
    public static void create(String path)
    {
```

```
try
{
path =path+"\\"+"tables_sql.txt";


System.out.println("******Path of
Commands***************"+path);
FileInputStream fin=new FileInputStream(path);
Scanner sc=new Scanner(fin);
sc.useDelimiter("/");
String name=" ";

Connection con=ConnectionProvider.getConnection();

Statement st=con.createStatement();

int i=1;

while(sc.hasNext())
{
    name=sc.next();
    st.executeUpdate(name);
```

```
            System.out.println(" Token "+ i++ +"  "+name);
        }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

    }
}
```

CONNECTION PROVIDER:

```
package connect;
import java.io.FileInputStream;
import java.sql.*;
import java.util.Properties;

public class ConnectionProvider
{

    private static Connection con;
```

```
private static Statement st;
private static String user;
private static String password;
private static String driver;
private static String url;

    public static void load(String path)
    {

        try
        {
        path=path+"\\"+"db.properties";
        FileInputStream fin=new FileInputStream(path);
        Properties p=new Properties();
        p.load(fin);
        user=p.getProperty("user");
        password=p.getProperty("password");
        driver=p.getProperty("driver");
        url=p.getProperty("url");

        }
        catch(Exception e)
```

```
    {
        e.printStackTrace();
    }
}


public static synchronized Connection getConnection()
{
    try
    {

    if(con == null)
    {

Class.forName(driver);
Connection con=DriverManager.getConnection(url,user,password);

System.out.println("Connecitn ....."+con);
        return con;
    }

    }
```

```
            catch(Exception e)
            {
                e.printStackTrace();
            }
            return con;
        }
}
```

TOKEN:

```
import java.io.*;
import java.util.*;

class Token
{
    public static void main(String args[])throws Exception
    {
```

```
        Scanner sc=new Scanner(new
FileInputStream("D:\\project\\working\\IMS\\WEB-
INF\\tables\\tables_sql.txt"));


        sc.useDelimiter("/");
        String name=" ";
        int i=1;


        while(sc.hasNext())
        {
            name=sc.next();


            System.out.println(" Token "+ i++ +"  "+name);
        }


    }
}
```

SQL:

CREATE TABLE:

```
CREATE TABLE LOGIN(
 LOGINID VARCHAR2(10),
 PASSWORD VARCHAR2(10),
 NAME VARCHAR2(20),
 EMAIL VARCHAR2(30),
 TYPE VARCHAR2(10),
 PRIMARY KEY(LOGINID))
/
insert into login values('saynam', 'saynam', 'saynam Jindal',
'saynamjindal18@gmail.com', 'Admin')
/
CREATE TABLE PRODUCTS(
 PRODID VARCHAR2(10),
 PRODNAME VARCHAR2(15),
 VERSION VARCHAR2(10),
 PRIMARY KEY(PRODID))
/
insert into products values('101', 'JavaEdit', '1.01')
/
insert into products values('102', 'SoundRX', '2.0.2')
/
insert into products values('103', 'WordSolution', '3.0.2')
```

```
/
CREATE TABLE EXPERTS(
 EXPERTID VARCHAR2(10),
 PRODID VARCHAR2(10),
 PRIMARY KEY(EXPERTID))
 /
 CREATE TABLE BUG(
 BUGID VARCHAR2(10),
 AUTHORID VARCHAR2(10),
 AUTHOR VARCHAR2(20),
 PRODID VARCHAR2(10),
 PRODNAME VARCHAR2(25),
 ENV VARCHAR2(15),
 TYPE VARCHAR2(15),
 DESCRIPTION VARCHAR2(100),
 REP_DATE DATE,
 PRIORITY VARCHAR2(10),
 STATUS VARCHAR2(15),
 ASSIGN_TO VARCHAR2(10),
 PRIMARY KEY(BUGID))
 /
 CREATE TABLE SOLUTIONS(
```

```
BUGID VARCHAR2(10),
EXPERTID VARCHAR2(10),
EXPERT VARCHAR2(20),
SOLN VARCHAR2(200),
SOLN_DATE DATE,
PRIMARY KEY(BUGID, EXPERTID))
```

WEBSERVER:

```
<web-app>
 <display-name>Bug Tracking System</display-name>

    <listener>
        <listener-class>listener.MyContextListener</listener-class>
    </listener>

     <context-param>
        <param-name>permit</param-name>
```

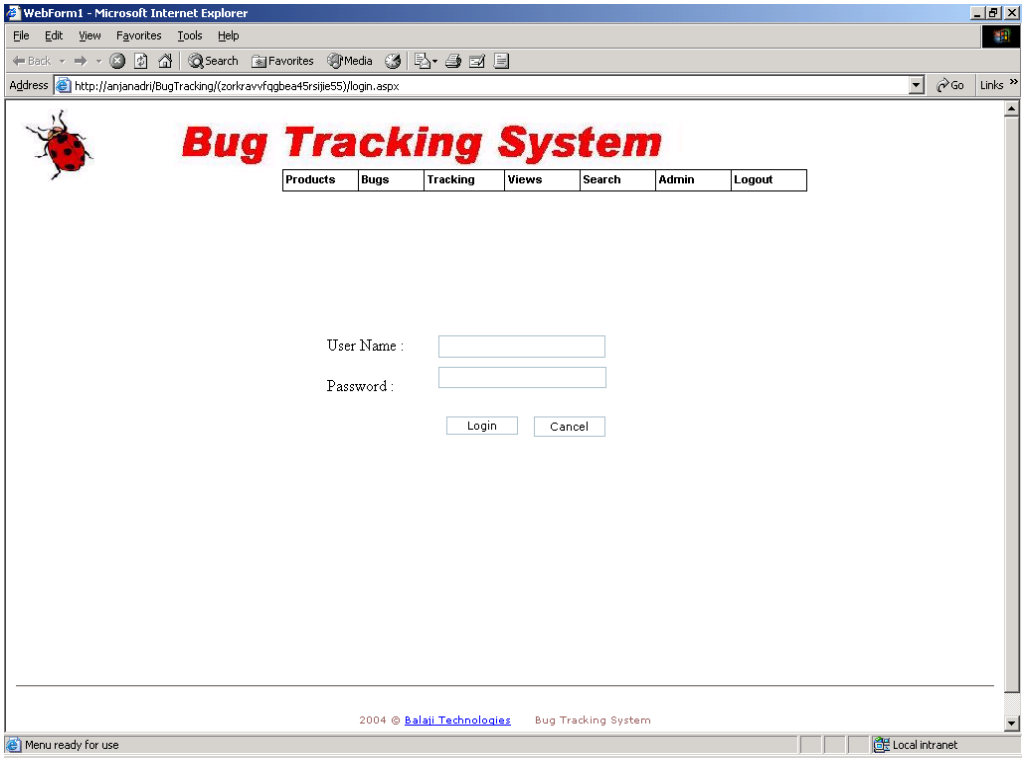
```
        <param-value>no</param-value>
    </context-param>

    <context-param>
        <param-name>user</param-name>
        <param-value>system</param-value>
    </context-param>
        <context-param>
        <param-name>password</param-name>
        <param-value>system</param-value>
    </context-param>
<servlet>
  <servlet-name>Login</servlet-name>
  <servlet-class>Login</servlet-class>
</servlet>

<servlet>
  <servlet-name>SignUp</servlet-name>
  <servlet-class>SignUp</servlet-class>
</servlet>

<servlet>
```

```
  <servlet-name>ReportBug</servlet-name>
  <servlet-class>ReportBug</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>Login</servlet-name>
  <url-pattern>/Login</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>SignUp</servlet-name>
  <url-pattern>/SignUp</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ReportBug</servlet-name>
  <url-pattern>/ReportBug</url-pattern>
</servlet-mapping>

<welcome-file-list>
   <welcome-file>default.jsp</welcome-file>
 </welcome-file-list>
```

</web-app>

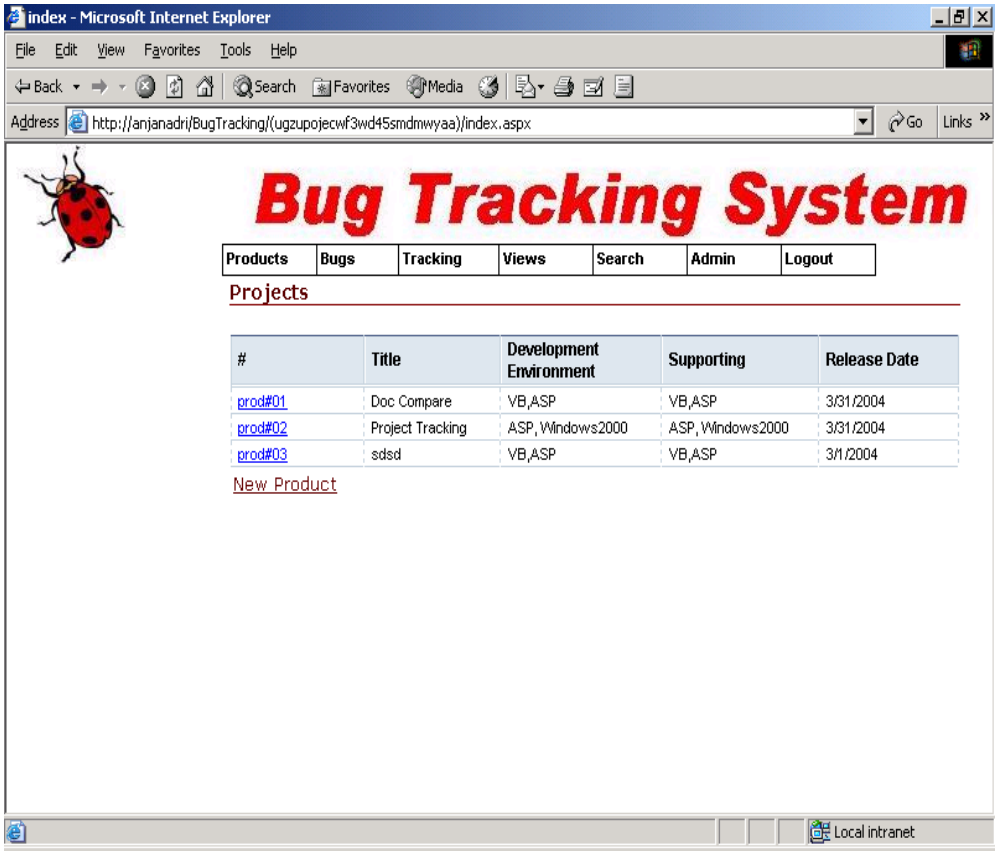
SNAPSHOT

1. LOGIN FORM:

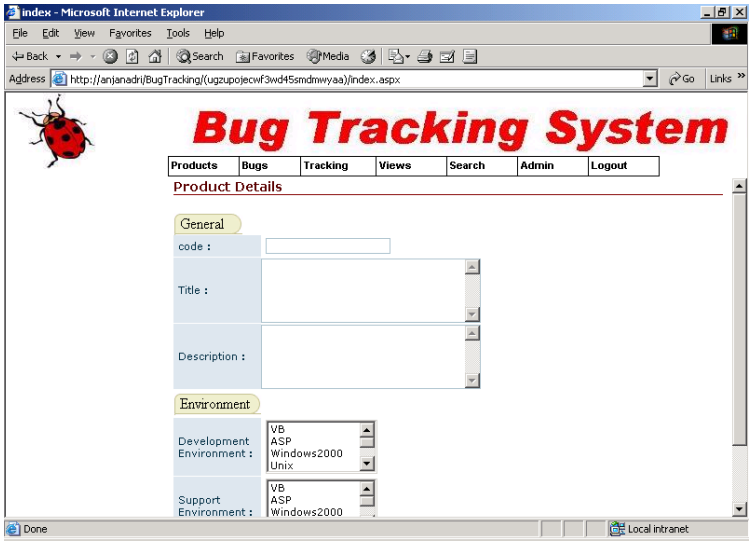

WebForm1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://anjanadri/BugTracking/(zorkravvfqgbea45rsijie55)/login.aspx
Go Links
Bug Tracking System
Products
Bugs
Tracking
Views
Search
Admin
Logout
User Name :
Password :
Login
Cancel
2004 © Balaji Technologies Bug Tracking System
Menu ready for use
Local intranet

MENUS

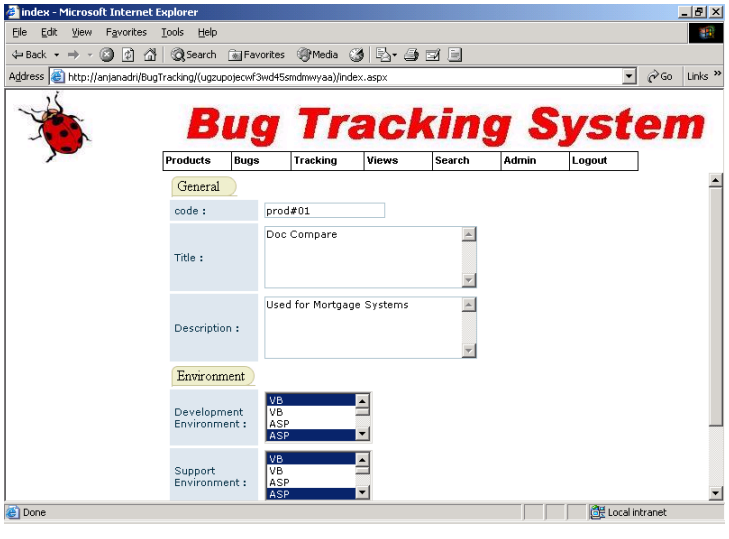
1. PRODUCTS
1. PRODUCT DETAILS



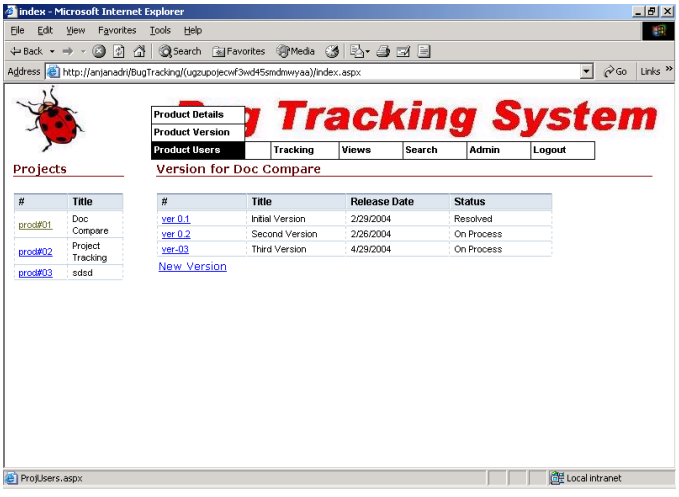
ADD PRODUCT DETAILS



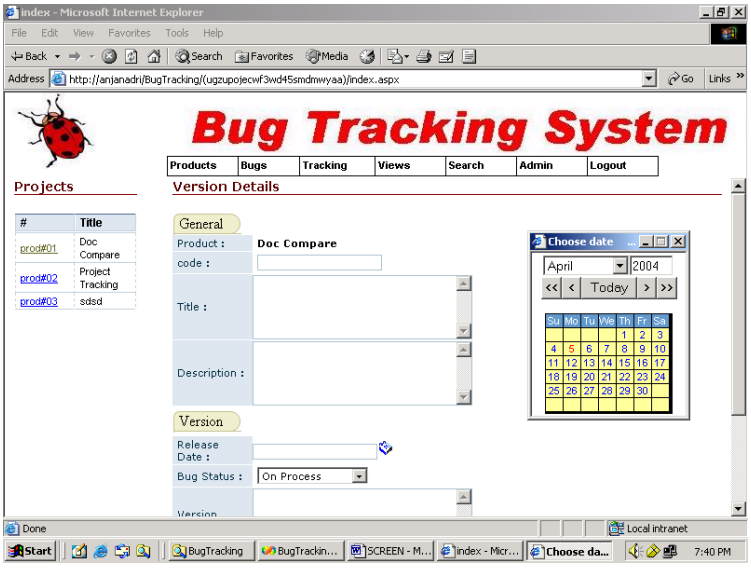
EDIT PRODUCT DETAILS



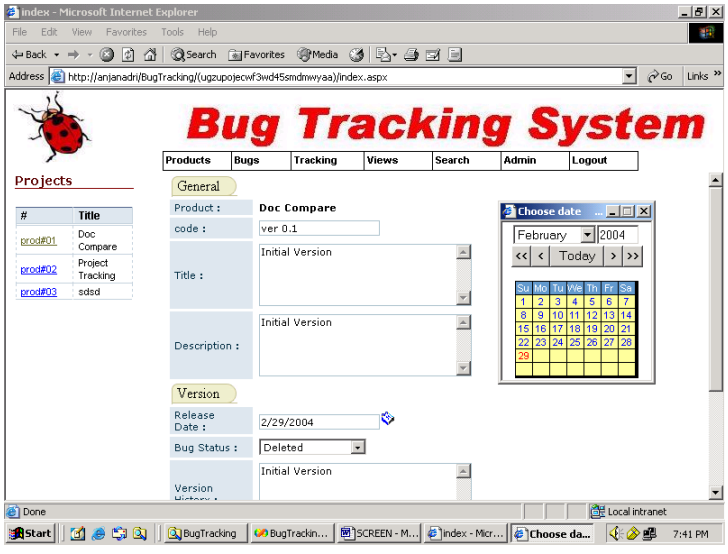
2. PRODUCT VERSION



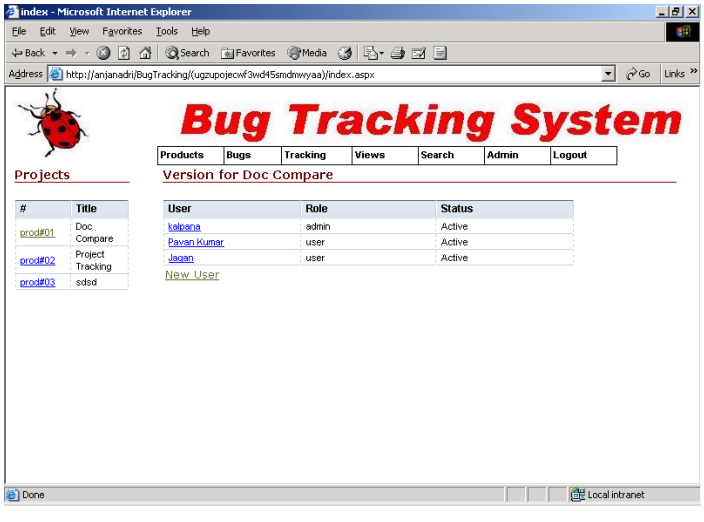
ADD PROJECT VERSION DETAILS



MODIFY PROJECT VERSION DETAILS

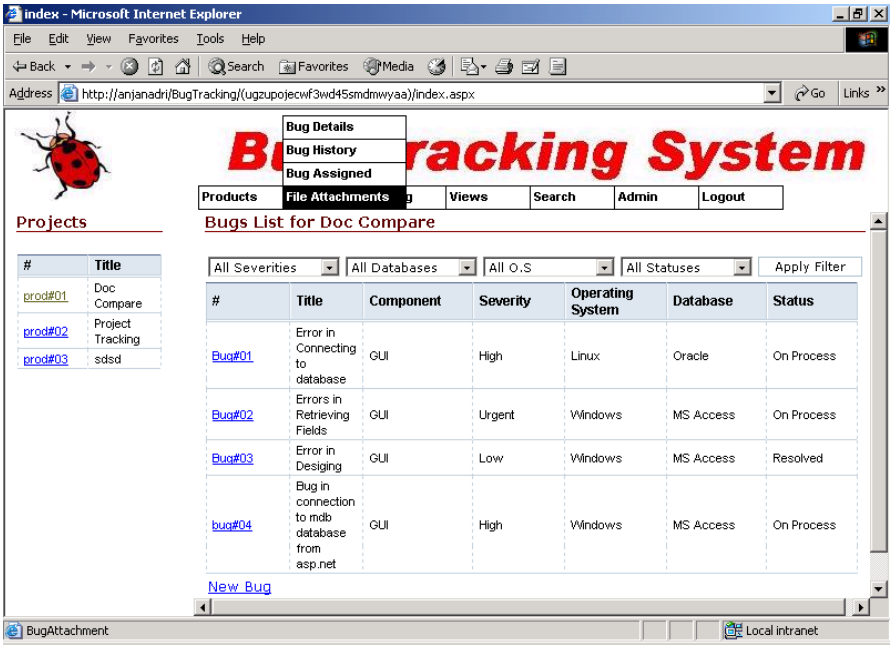


3. PRODUCT USERS

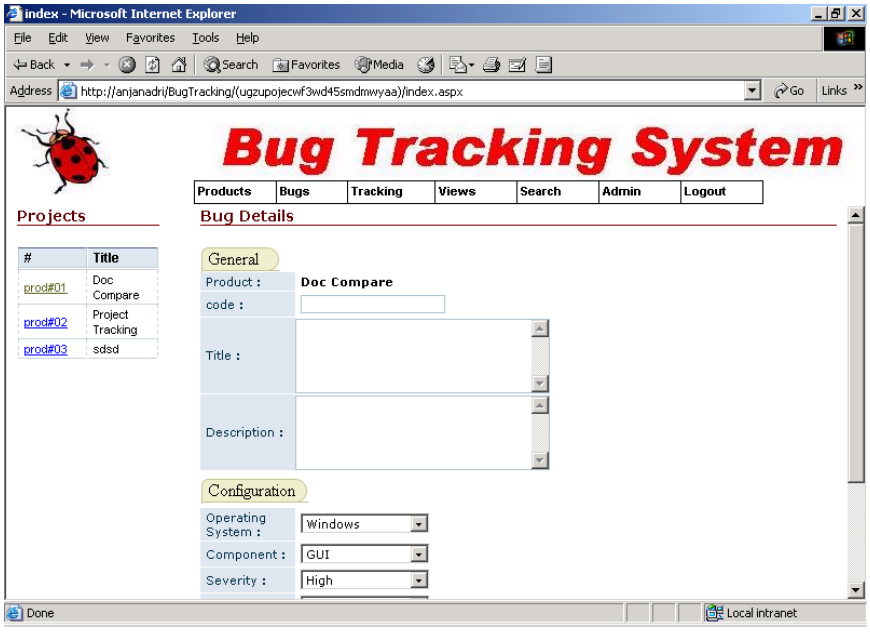


2.BUGS

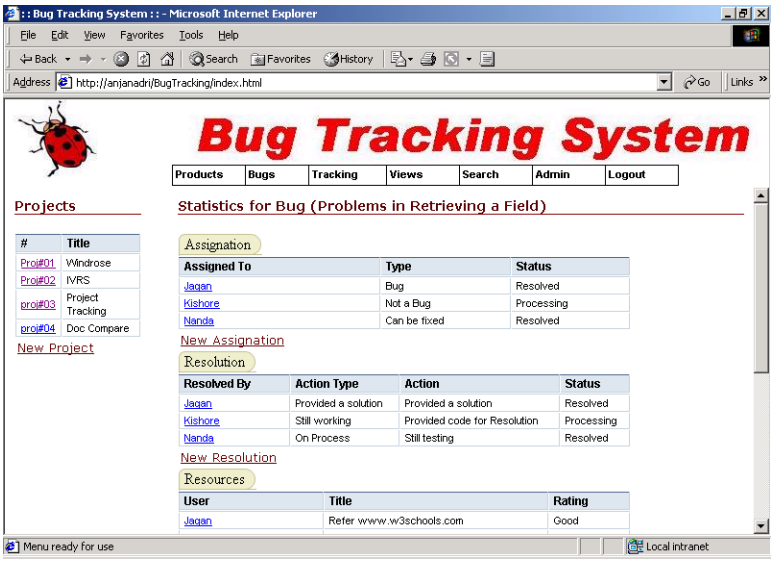
1.BUG DETAILS



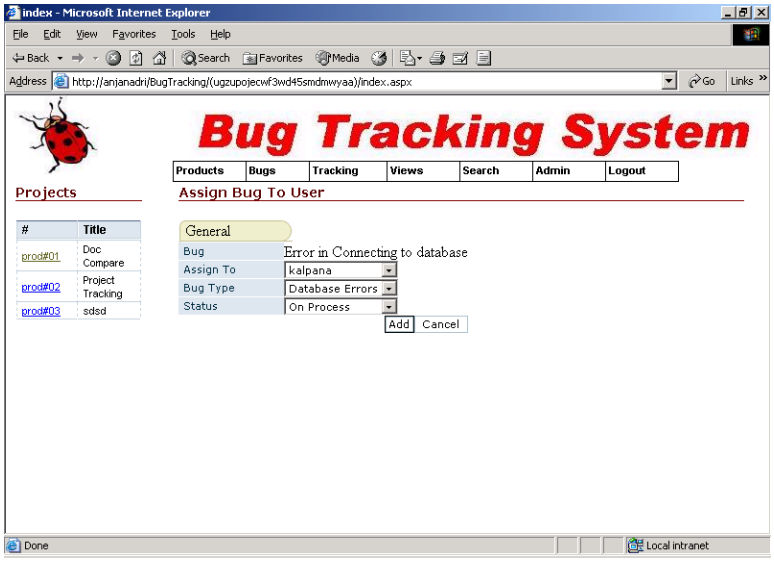
1. ADD BUG DETAILS



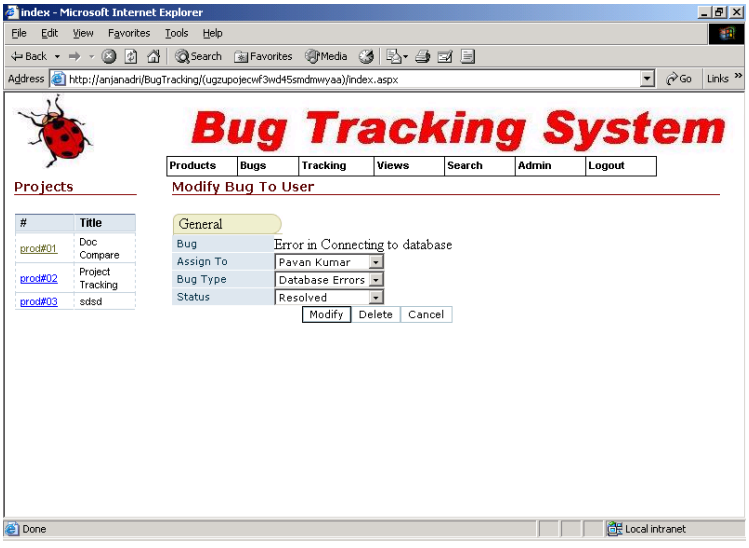
3. BUG ASSIGNED



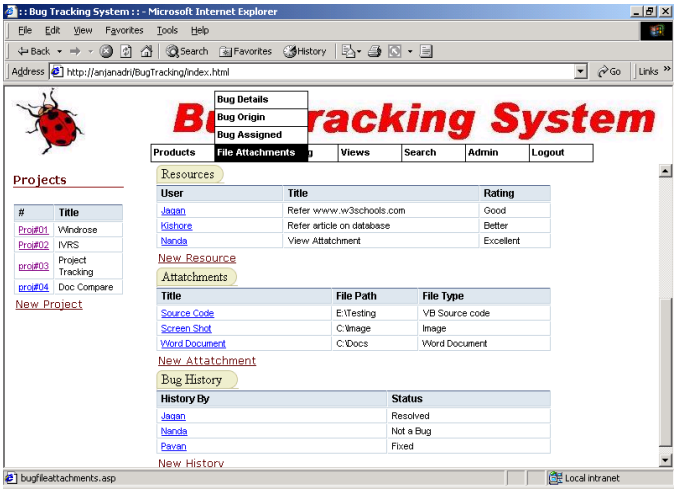
ASSIGNING BUG TO USER



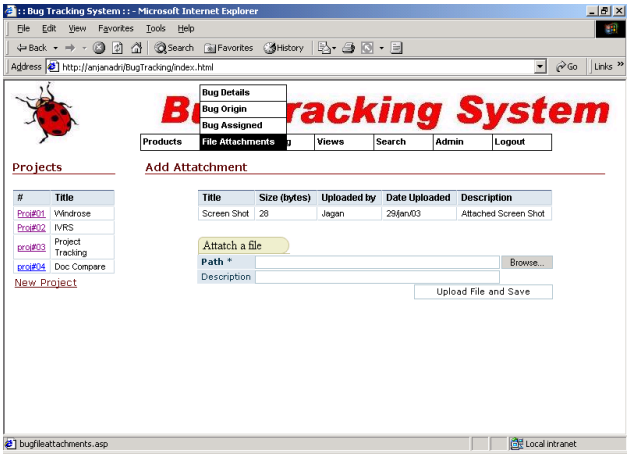
MODIFY BUG ASSIGNED TO USER



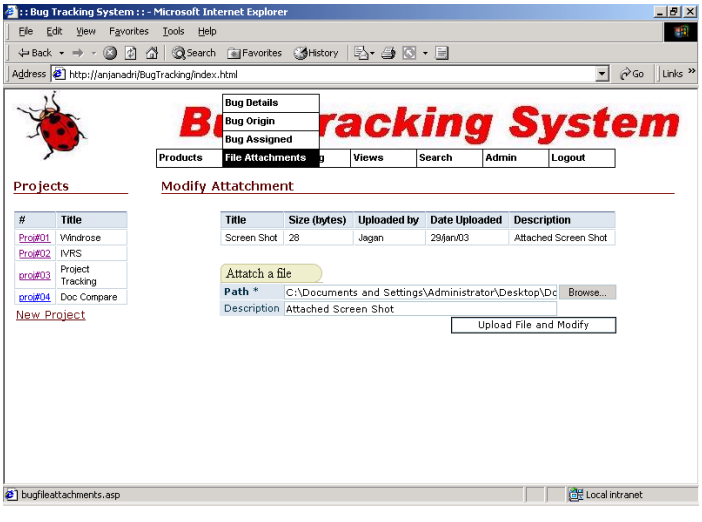
4. FILE ATTATCHMENTS



ADD NEW FILE ATTATCHMENT

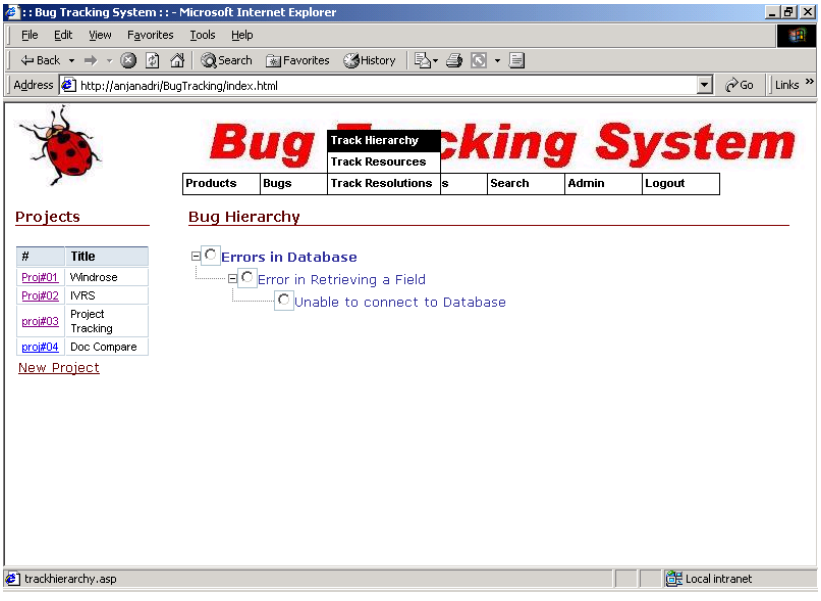


MODIFY FILE ATTATCHED

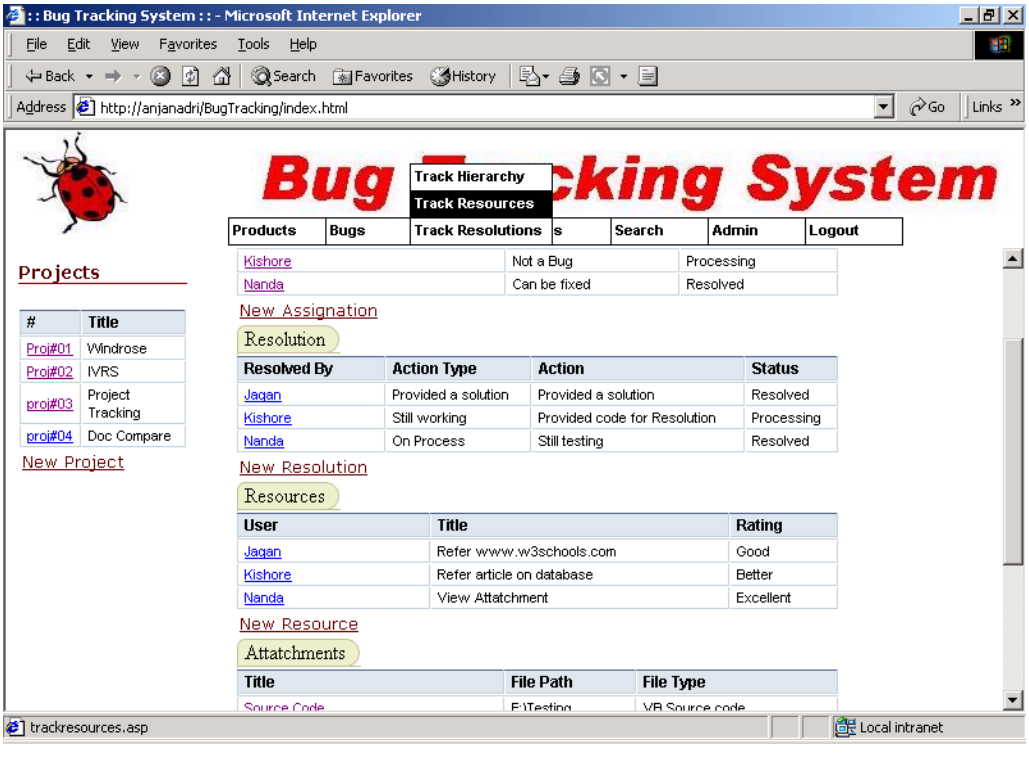


3. TRACKING

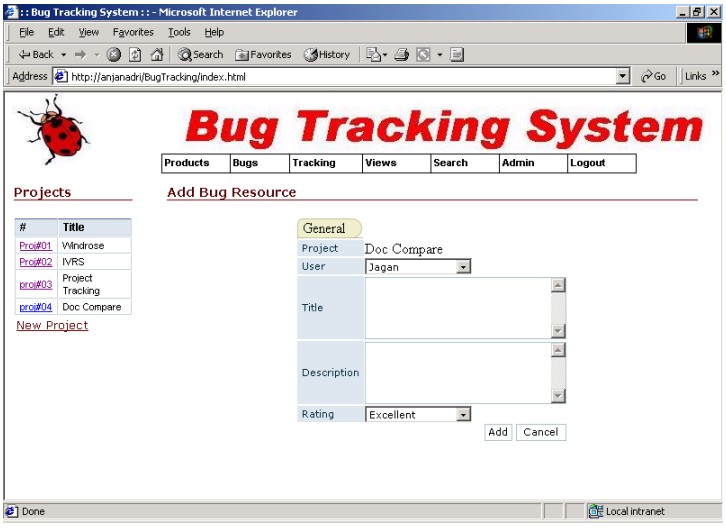
TRACK HIERARCHY



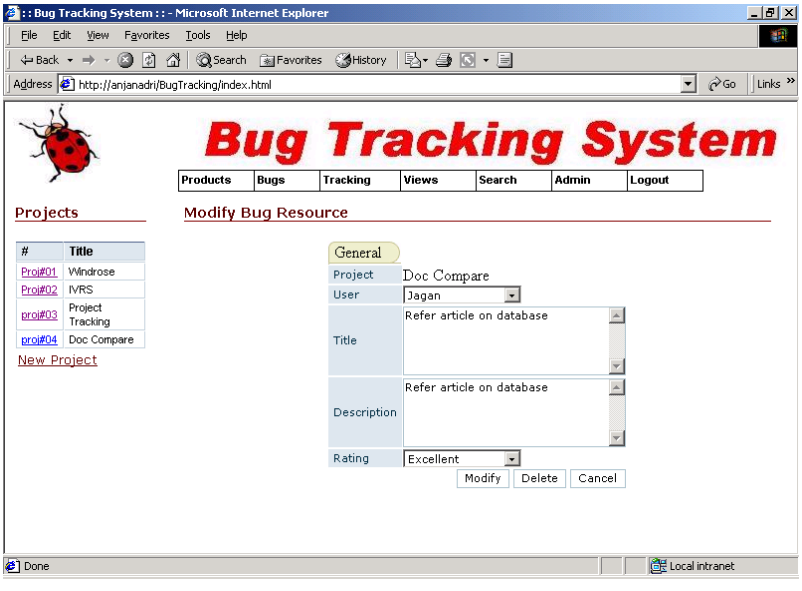
2. TRACK RESOURCES



ADD NEW RESOURCE TO BUG

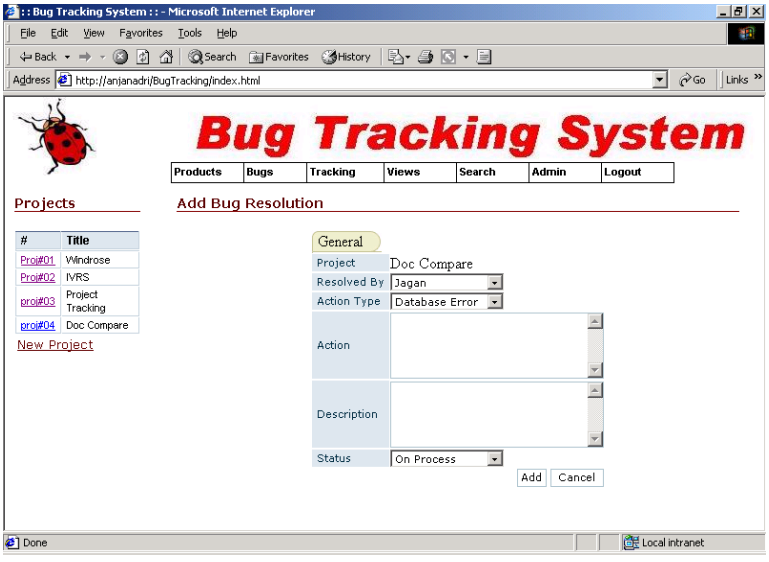


MODIFY EXISTING RESOURCE



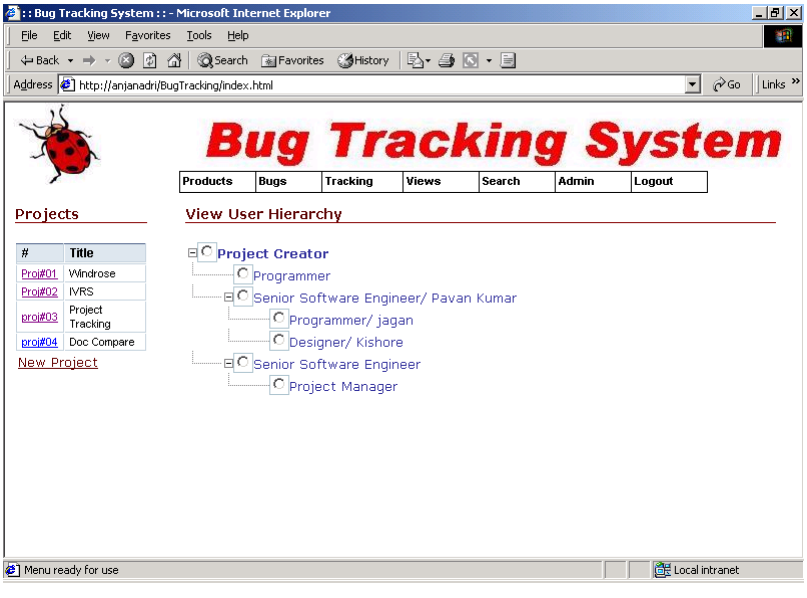
TRACK RESOLUTION

ADD NEW RESOLUTION

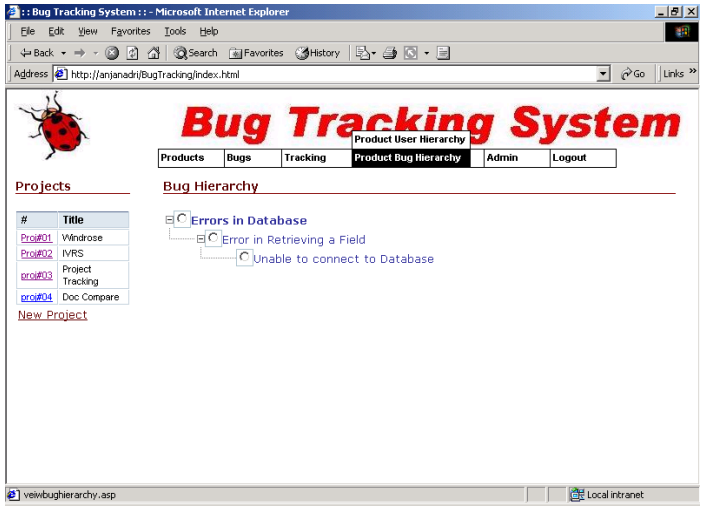


4. VIEWS

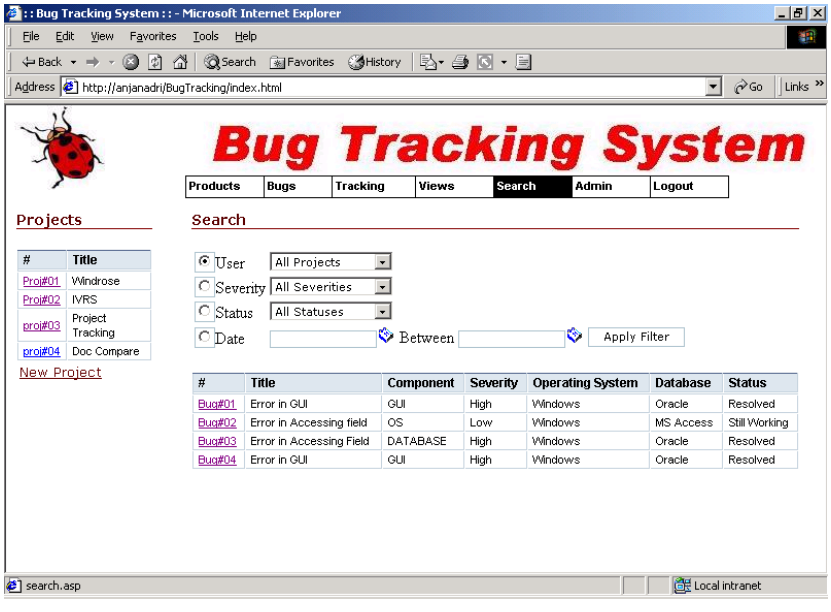
PRODUCT USER HIERARCHY



PRODUCT BUG HIERARCHY

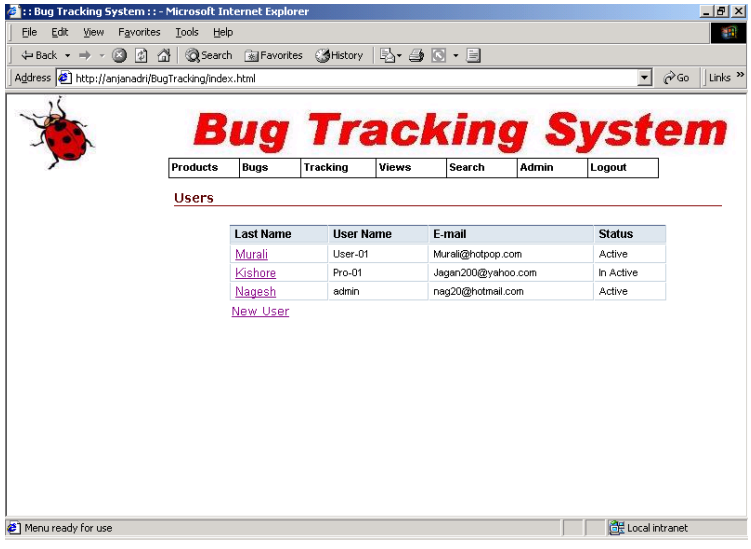


5. SEARCH

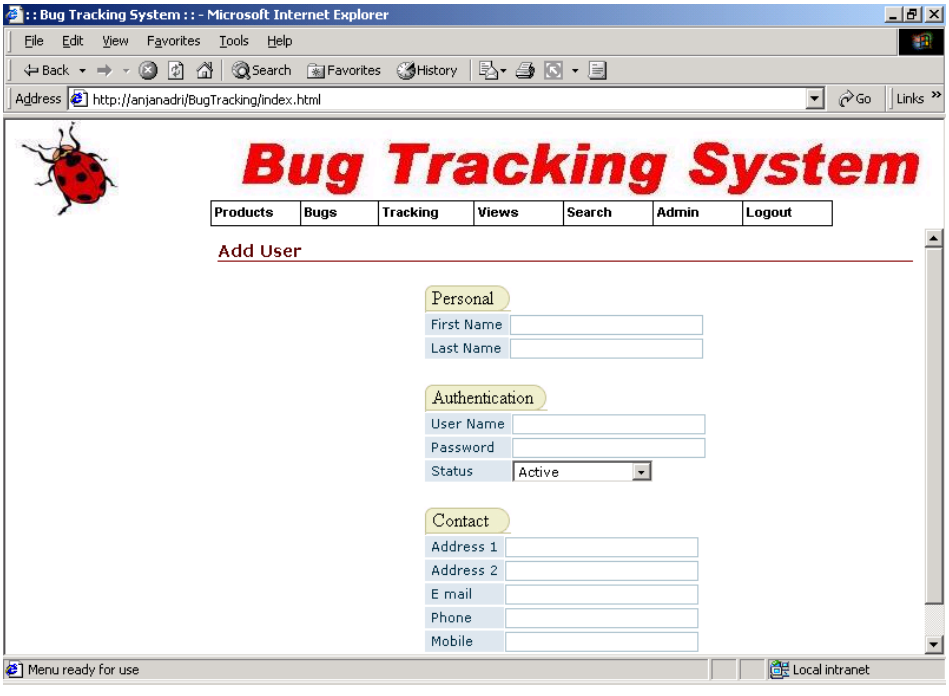


6. ADMIN

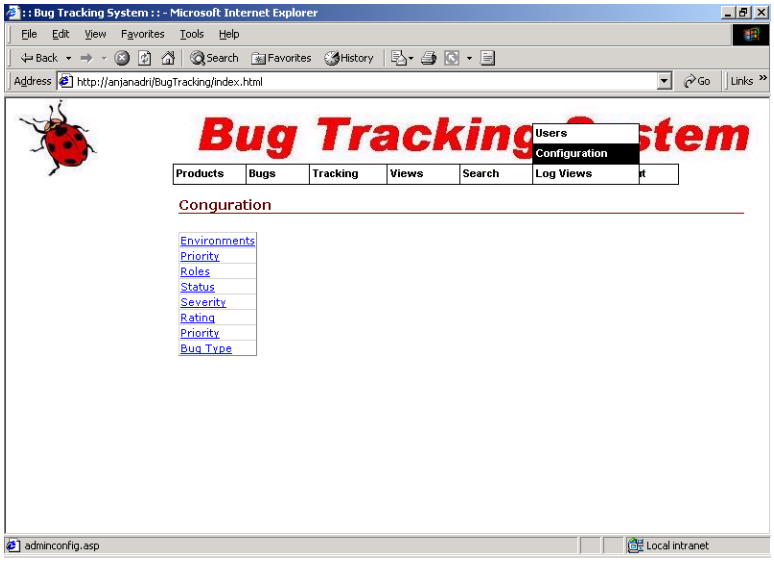
USER



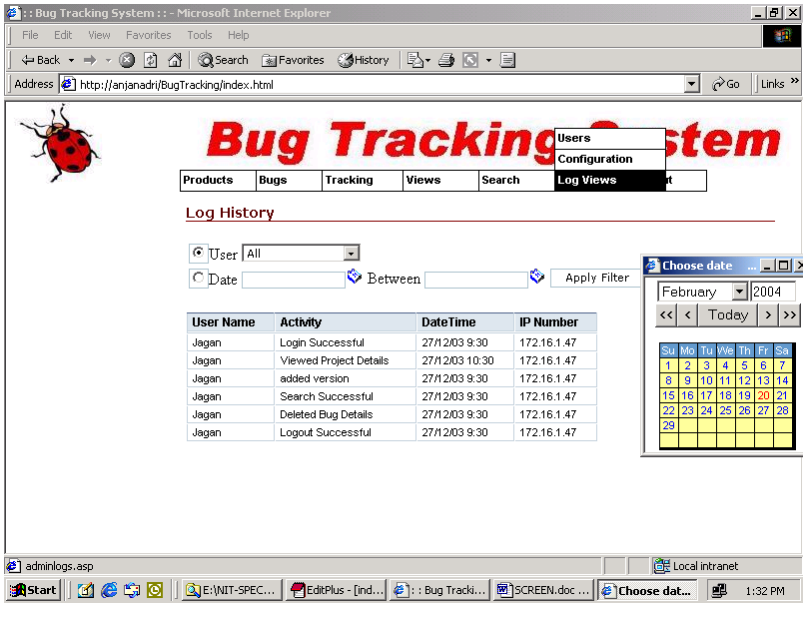
ADD NEW USER



CONFIGURATION



LOG



TESTING

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to

someone who matters within the project community.

Testing:

- **Testing Methodologies**
    - Black box Testing:
    - White box Testing.
    - Gray Box Testing.

- **Levels of Testing**
    - Unit Testing.
    - Module Testing.
    - Integration Testing.
    - System Testing.
    - User Acceptance Testing.
- **Types Of Testing**
    - Smoke Testing.
    - Sanitary Testing.
    - Regression Testing.
    - Re-Testing.
    - Static Testing.
    - Dynamic Testing.
    - Alpha-Testing.
    - Beta-Testing.
    - Monkey Testing.
    - Ext….
- **TCD (Test Case Documentation)**
- **STLC**
    - Test Planning.
    - Test Development.
    - Test Execution.
    - Result Analysis.

- Bug-Tracing.
- Reporting.

Introduction

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product

larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes.

It is commonly believed that the earlier a defect is found the cheaper it is to fix it.

	Time Detected				
Time Introduced	Requirements	Architecture	Construction	System Test	Post-Release
Requirements	1	3	5-10	10	10-100
Architecture	-	1	10	15	25-100
Construction	-	-	1	10	10-25

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process unit tests are written first,

by the programmers (often with pair programming in the extreme programming methodology). Of course these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed.

Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

FUTURE SCOPE

In past years, Bug finding systems are broadly used in the IT industry, specifically in software building. Hardware and software companies use these systems throughout the progress cycle to trace design and bug problems. The system keeps a collection of data that gathers all facts concerning reported and defected bugs. A good machine-driven bug tracking explication should contour the method of managing, raising and fixing problems. The system ought to be evolced along the time , it shouldn’t be designed just for internal groups however additionally for consumer or other end clients to report problems associated with the system. Web based applications will usually be tough to guide, particularly if they act more like a chain of HTML pages than an

application, some applicatipns based on web are perpetually reloading pages when answering to client input requires a circular trip between the user's browser and also the online server. The complexness of the system would possibly scale back the performance of work and users would possibly feel bothersome once they make use of the system.

If the system is difficult to use, demands heaps of data or is time overwhelming than the utilization of the system is restricted or the information might not be correct. Once a client proposes a report, he is inquired by various questions like: name of the product? plugin information, bulid id etc, obtaining replies from clients takes time and decelerates the process on active bugs. The system ought to consolidate and oragnize data in system and centralize data depository for bugs and changes in system. To alter the process of bug tracking system the system ought to be targeted on straightforward ideas and usefulnessto produce a simple to utilize the solution for medium and small businesses.

As a confirmation of concept, I also show a paradigm of bug tracking system that collects important data from the client and recognizes the files that need to be repaired to tackle the bug. The focus of my work is on improving the system with the idea of increasing the completness of the anomaly reports. Further research on the system can conclude further improvements in the tracker that can make it more effective and time saving.

Limitations and Future Enhancements :

Limitations of the system:

- Only the permanent employees can access the system.
- System works with windows'98 and its compatible environments.
- Advanced techniques are not used to check the authorization.

- Once the employee is registered to a course cannot drop, without completing.

Future Enhancements:

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Because it is based on object-oriented design, any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.
- Attendance module can be added
- sub admin module can be added

VALUATION

The proposed directions for improving bug tracking systems are tested using the prototype application. The application is capable of tracking bugs with sufficient information that leads to fixing bugs quickly and efficiently. This results in saving lot of time and expenditure. The development team can avoid over budget and crossing deadlines. The development time and cost can be reduced. The tool oriented nature of the

system reduces the burden of information collection as it can automatically locate stack traces and add the comprehensive data to bug report. The information oriented nature of the application facilitates integration of tools for obtaining real time feedback. The process oriented nature of the application allows easy administration activities pertaining to bug tracking. The user oriented nature of application educates developers and bug reporters on which kind of information to be provided for easy bug tracking.

PROJECT SUMMARY

This project Bug Tracking for Improving Software Quality and Reliability is to keep track of employee skills and based on the skills assigning of the task is done to an employee. Employee does bugs capturing. It can be done on daily basis. Various Reports are generated by this System for an employee and as well as to a manager.

This project will be accessible to all developers and its facility allows developers to focus on creating the database schema and while letting the application server define table based on the fields in JSP and relationships between them.

This application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using Java as front end and Oracle as back end in Windows environment. The goals that are achieved by the software are:

- ✓ Instant access.
- ✓ Improved productivity.
- ✓ Optimum utilization of resources.
- ✓ Efficient management of records.
- ✓ Simplification of the operations.
- ✓ Less processing time and getting required information.
- ✓ User friendly.
- ✓ Portable and flexible for further enhancement.

CONCLUSION

Software Development Lifecycle (SDLC) can never be completed without encountering bugs in the coding phase. The bugs thus surfaced are to be fixed faster and go ahead with development. Bug tracking needs necessary information to resolve bugs faster. The present systems for bug tracking are effective. However, they can be further improved by following certain standards. It does mean that bug tracking systems can be improved. This paper has done this. We proposed a framework with four directions that can help improve bug tracking systems. They are tool oriented, information oriented, process oriented and user oriented in nature. By following these directions, ideal bug tacking systems can be built. The directions are iterative in nature and one or more loops of directions while following make the bug tracking systems perfect. The more interactions followed while building a bug tracking system, the more effectively it works. To demonstrate the effectiveness of the proposed system, a prototype bug tracking system is built that asks relevant questions as per the enhanced framework with directions. The answers to the questions make the process of fixing bugs faster. Thus it fosters the bug fixing process with improvement in time taken to fix bugs. The proposed framework can further

improved in future by making it more robust and interactive and adapt to all kinds of software systems.

REFERENCES

[1] N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In FSE'08:Proceedings of the 16th International Symposium on Foundationsof Software Engineering, pages 308–318, November 2008.

[2] S. Breu, J. Sillito, R. Premraj, and T. Zimmermann.Frequently asked questions in bug reports.Technical report, University of Calgary, March 2009.

[3] S. Artzi, S. Kim, and M. D. Ernst.Recrash: Making software failures reproducible by preserving object states. In ECOOP'08: Proceedings of the 22nd European Object-Oriented Programming Conference, pages 542–565, 2008.

[4] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Duplicate bug reports considered harmful ... really? In ICSM'08: Proceedings of the 24th IEEE International Conference on Software Maintenance, pages 337–345, 2008.

[5] S. Breu, J. Sillito, R. Premraj, and T. Zimmermann.Frequently asked questions in bug reports.Technical report, University of Calgary, March 2009.

[6] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In ICSE'06: Proceedings of the 28th International Conference on Software engineering, pages 361-370, 2006.

[7]J. Aranda and G. Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. In ICSE'09: Proceedings of the 31st International Conference on Software Engineering (to appear), 2009.

[8]S. Artzi, S. Kim, and M. D. Ernst. Recrash: Making software failures reproducible by preserving object states. In ECOOP'08: Proceedings of the 22nd European Object-Oriented Programming Conference, pages 542-565, 2008.

[9]N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In FSE'08: Proceedings of the 16th International Symposium on Foundations of Software Engineering, pages 308-318, November 2008.

[10]S. Breu, J. Sillito, R. Premraj, and T. Zimmermann. Frequently asked questions in bug reports. Technical report, University of Calgary, March 2009.

[11]P. Fritzson, T. Gyimothy, M. Kamkar, and N. Shahmehri. Generalized algorithmic debugging and testing. In PLDI'91: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 317-326, 1991.

[12]P. Hooimeijer and W. Weimer. Modeling bug report quality. In ASE'07: Proceedings of the 22nd International Conference on Automated Software Engineering, pages 34-43, 2007.

[13]S. Just, R. Premraj, and T. Zimmermann. Towards the next generation of bug tracking systems. In VL/HCC'08: Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, pages 82-85, September 2008.

[14]A. J. Ko and B. A. Myers. Debugging reinvented: asking and answering why and why not questions about program behavior. In ICSE'08: Proceedings of the International Conference on Software Engineering, pages 301-310, 2008.

[15]B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. In PLDI'05: Proceedings of the 2005 ACM SIGPLAN ConferenceonProgramming Language Design and Implementation, pages 15-26, 2005.

E