

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in

Information Technology

By

**VEMANA AKHIL NEERAJ
(Reg.No.37120083)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

JANUARY 2021



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **V.AKHIL NEERAJ** (Reg.no.37120083) who carried out the project entitled “**DEFECTS TRACKING FOR IMPROVING PRODUCT QUALITY AND PRODUCTIVITY** ” under our supervision from DEC 2020 to JAN 2020.

Internal Guide

E.BRUMANCIA M.E,(Ph.D.,)

Head of the Department

Dr.R SUBHASHINI M.E,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I ,V.AKHIL NEERAJ (Reg.No.37120083) hereby declare that the Project Report entitled “**DEFECTS TRACKING FOR IMPROVING PRODUCT QUALITY AND PRODUCTIVITY**” done by me under the guidance of **E.BRUMANCIA M.E(Ph.D).**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology .

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** or their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T SASIKALA M.E Ph .D**, Dean, School Of Computing and **Dr .R SUBHASHINI M.E Ph. D** ,Head of the Department, Dept .of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **E.BRUMANCIA M.E(Ph.D)**for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Information Technology who were helpful in many ways for the completion of the project.



To
The Head of the Department,
Department of Information Technology,
Sathyabama Institute Of Science And Technology,
Chennai, Tamil Nadu.

Dear Sir/Madam,

Sub : Internship Project Confirmation Letter.

This is to confirm and certify VEMANA AKHIL NEERAJ (37120083),
B.Tech(IT), student of your college has been admitted to do a project titled "DESIGN
AND IMPLEMENTATION OF PROJECT TRACKING SYSTEM TO FIND DEFECTS FOR
IMPROVING PRODUCT QUALITY" on JAVA Technology at "Middleware Information
Technology Services Pvt. Ltd."

The Project duration is from 05th December 2019 to 05th January 2020.

The Project will be done under the guidance of our technical team.

for Middleware Information Technology Services Pvt Ltd.,

Project Guide.

Middleware Information Technology Services Pvt Ltd

Level 1, Plot No 113, Phase III, Kamalapuri Colony, Jubilee Hills, Hyderabad - 500073
Contact : +91 - 40 4952 1122 Website : www.mitservices.net Email : info@mitservices.net



ABSTRACT

Defect Tracking for Improving Product Quality and Productivity for Improving Software Reliability is an automated system that can be useful to employees and the managers in any functional organization. Defect Tracking for Improving Product Quality and Productivity gives the facility to define the tasks in the organization and also allows the managers to track the Defects spent by the developer for that particular task. A report generation facility is supported in DTS that allows the managers to analyze which are those skills by employee are utilized and those which are not utilized.project or application.This tool helps employees to document their Defects and analyze.

Defect Tracking for Improving Product Quality and Productivity gives the facility to define the tasks in the organization and also allows the managers to assign the bugs or defects to the developer.The developer will solve the problem in the system and send it to the test phase.In test phase if any bugs are present then it will identify it and send it to organizer.Then the process will be repeated till the proper out has obtained.By this way this tool helps employee to document their defects and analyze.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE.NO
1	ABSTRACT LIST OF ABBREVIATIONS LIST OF FIGURES INTRODUCTION	
	1.1 INTRODUCTION TO THE PROJECT	1
	1.2 AIM AND SCOPE	1
	1.3 FEASIBILITY STUDY	1
	1.4 SYSTEMS	3
2	ALGORITHMS AND METHODS	
	2.1 SDLC METHODOLOGY	4
	2.2 ACTIVITY DIAGRAM	6
3	SYSTEM DESIGN AND TESTING PLAN	
	3.1 MODULES	7
	3.2 INPUT DESIGN	8
	3.3 DATA FLOW DIAGRAM	10
	3.4 UML DIAGRAM	11
	3.5 TEST PLAN	15
	3.6 SYSTEM REQUIREMENTS	19
	3.7 NON-FUNCTIONAL REQUIREMENTS	19
	RESULTS AND DISCUSSIONS	
	4.1 SUMMARY	22
	4.2 SCREEN SHOTS	23
5	CONCLUSION AND FUTURE ENHANCEMENTS	
	5.1 CONCLUSION	36
	5.2 FUTURE ENHANCEMENT	37
	5.3 REFERENCES	37
6	APPENDIX	
	6.1 SOURCE CODE	38

LIST OF FIGURES

FIGURE NO.	TITLE OF THE FIGURE	PAGE NO.
2.1	Activity Diagram	6
3.1.2	System Architecture	7
3.3	Data Flow Diagram	9
3.4.1	Use Case Diagram	10
3.4.1	Class Diagram	11
3.4.1	Sequence Diagram	12
3.4.1	Component Diagram	13

LIST OF ABBREVIATIONS

ACRONYM

DFD

UML

EPL

JDT

IDE

GPS

EXPANSION

Data Flow Diagram

Unified Modeling Language

Eclipse Public License

Java Development Tools

Integrated Development
Environment

Global Positioning System

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO THE PROJECT

Defect Tracking for Improving Product Quality and Productivity for Improving Software Reliability is an automated system that can be useful to employees and the managers in any functional organization. Defect Tracking for Improving Product Quality and Productivity gives the facility to define the tasks in the organization and also allows the managers to track the Defects spent by the developer for that particular task. A report generation facility is supported in DTS that allows the managers to analyze which are those skills by employee are utilized and those which are not utilized. project or application. This tool helps employees to document their Defects and analyze.

1.2. AIM AND SCOPE :

1.2.1. AIM:

The main aim of the project is “To track the defects for improving product quality and productivity”.

1.2.2. SCOPE:

Defect Tracking System is very useful for removing defects from project module. Only if features mentioned in document are extended. In future users may also be possibly notifying the name/Id of defect creator so that organization can prevent their systems.

1.3. Feasibility Study

1.3.1. 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?

1.3.2. 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.

1.3.3.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.

1.4 SYSTEMS:

1.4.1 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.

DISADVANTAGES OF EXISTING SYSTEM :

- Information retrieval is a very big process.
- Lack of organization of the files may lead 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.

1.4.2 PROPOSED SYSTEM

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

ADVANTAGES OF PROPOSED SYSTEM :

- The performance is increased due to well designed database.
- Security is increased
- Time saving in report generation
- Easy to update the details

CHAPTER 2

ALGORITHM AND METHODS

2.1.SDLC METHODOLOGY:

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
 1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
 2. Defining the requirements of the second prototype.
 3. Planning an designing the second prototype.
 4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:

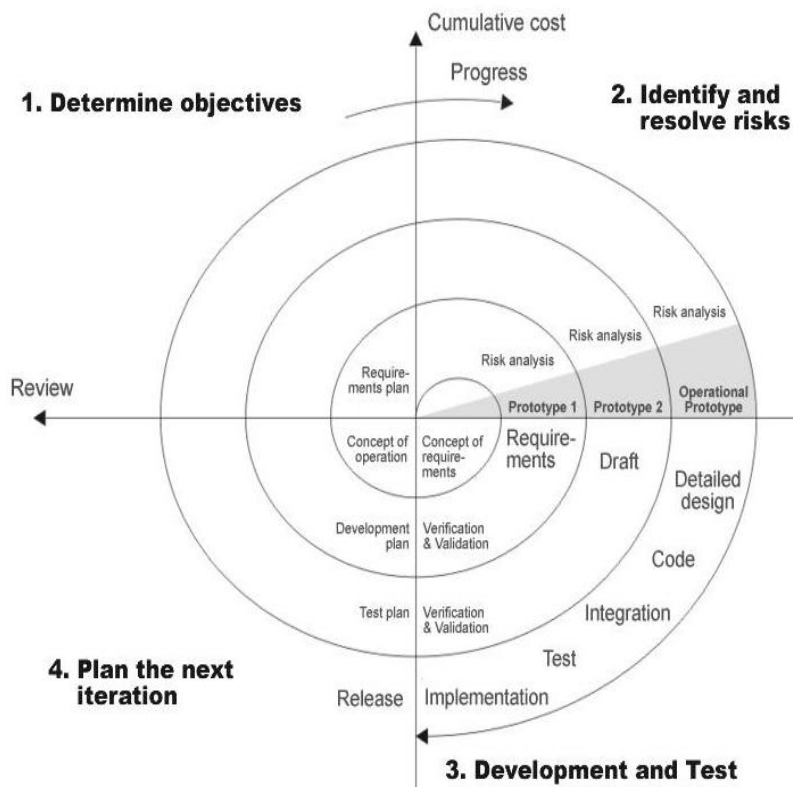


Fig 1.0-Spiral Model

ADVANTAGES

- Estimates(i.e. budget, schedule etc .) become more realistic as work progresses, because important issues discovered earlier.
- It is more able to cope with the changes that are software development generally entails.
- Software engineers can get their hands in and start working on the core of a project earlier.

2.2 ACTIVITY DIAGRAM:

ACTIVITY DIAGRAM

Admin:

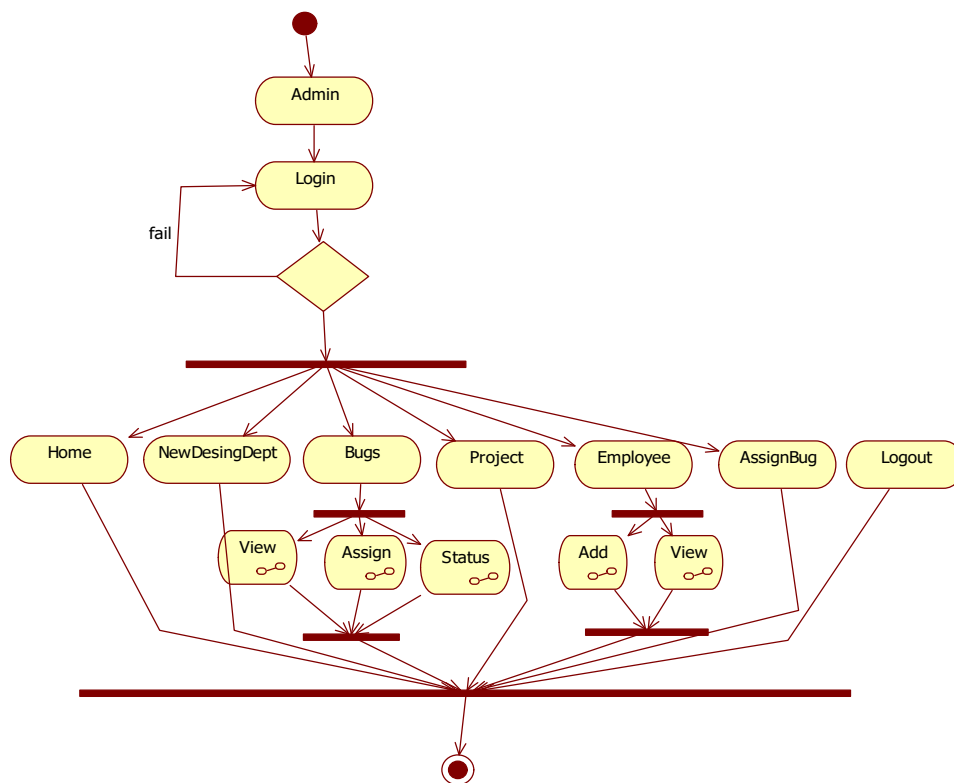


Fig 2.1. Activity diagram

CHAPTER 3

SYSTEM DESIGN AND TESTING PLAN

3.1.1 MODULES:

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 Defects 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 Defects 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 Defect assigned by the manager, view assigned projects and resolving the assigned Defect. Developer can view the Defects list assigned by the manager.

Tester: Tester can access to the projects , can view the assigned projects and can add a new Defect 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.

3.1.2 SYSTEM ARCHITECTURE

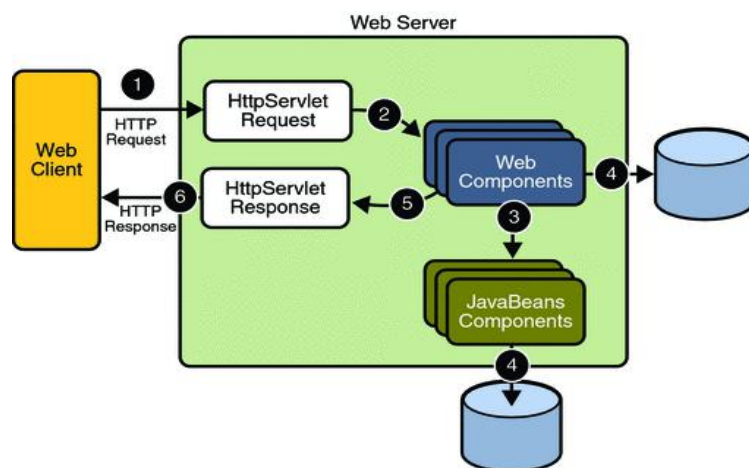


Fig .3.1.2 shows system architecture

3.2 INPUT DESIGN:

3.2.1. THE PRESENTATION LAYER

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

3.2.2.THE BUSINESS RULES LAYER

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

3.2.3.THE DATA ACCESS LAYER

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

3.2.4.THE DATABASE LAYER

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flatfiles,etc.

In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

3.3. DATA FLOW DIAGRAM (DFD)

A data flow diagram is a way of representing a flow of data of a process or a system. The DFD also provides the information about the input and output of each entity and the process itself.

DFD- LEVEL

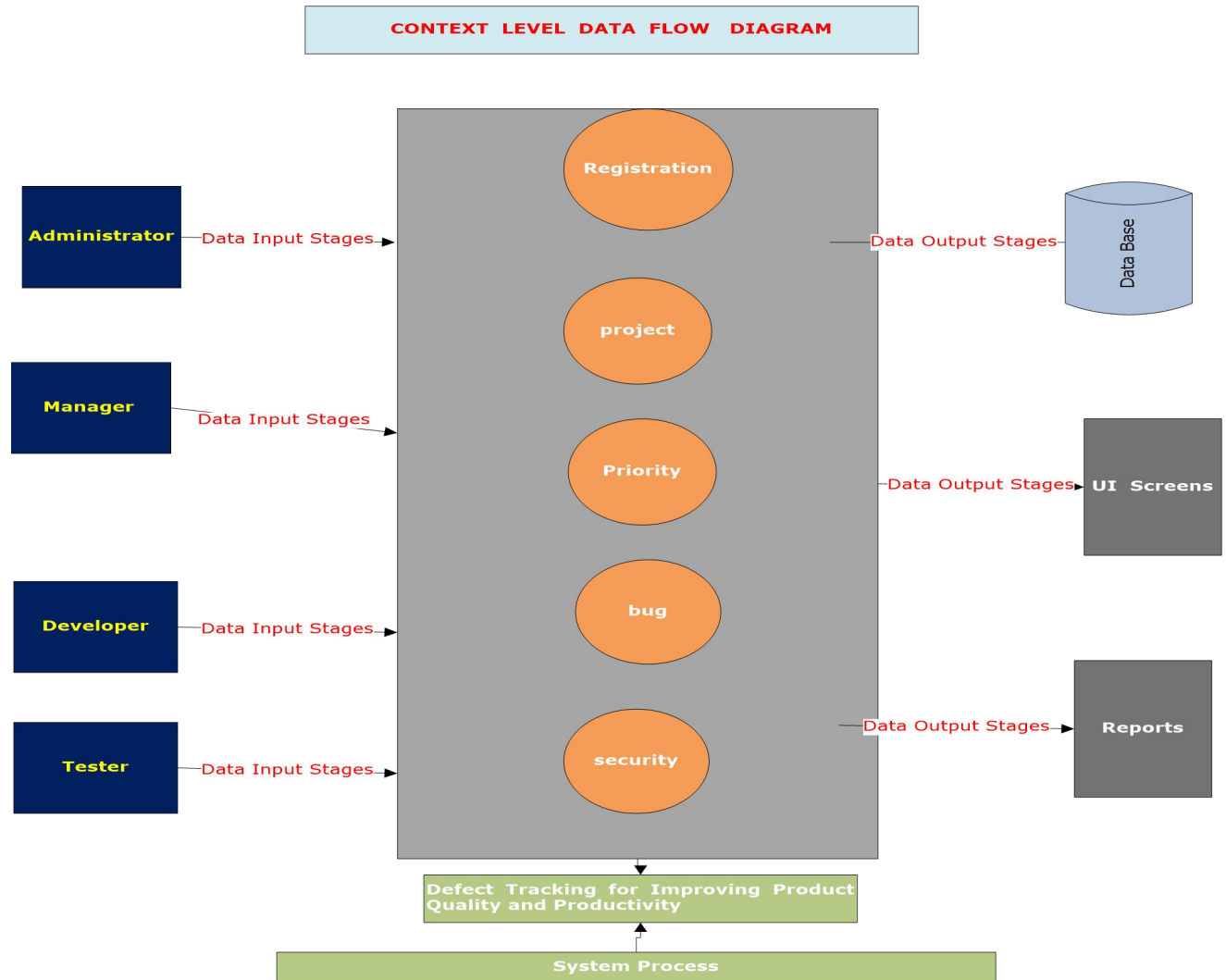


Fig 3.3. shows data flow diagram

3.4. UML DIAGRAMS

Unified Modelling language includes a set of graphic notations techniques to create a visual models of object oriented software intensive systems

3.4.1 TYPES OF UML DIAGRAMS

- Use Case Diagram
- Class diagram
- Sequential diagram
- Component diagram

Use case diagram :

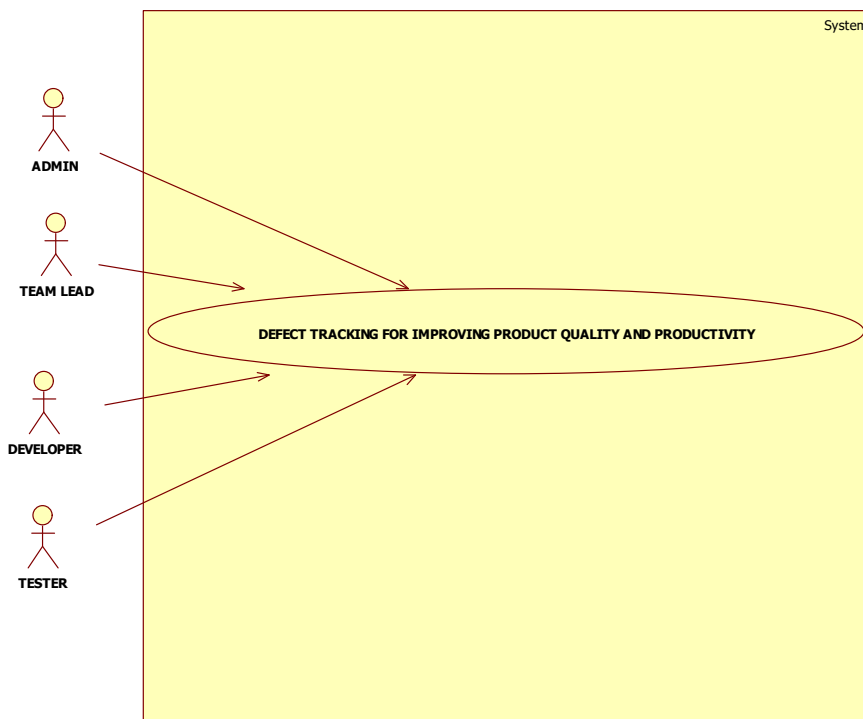


Fig. 3.4.1 Shows use case diagram

CLASS DIAGRAM :

In software Engineering , a class diagram in the unified modeling language is a type of structure diagram that describes the structure of a system by showing the systems ,classes ,their structures,operations and their relationships among objects.

In the diagram , classes are represented with bases which contain three parts.

- The top part contains the name of the class.
- The middle point contains the attributes of the class.
- The bottom part gives the methods or operations in the class can take or undertake .

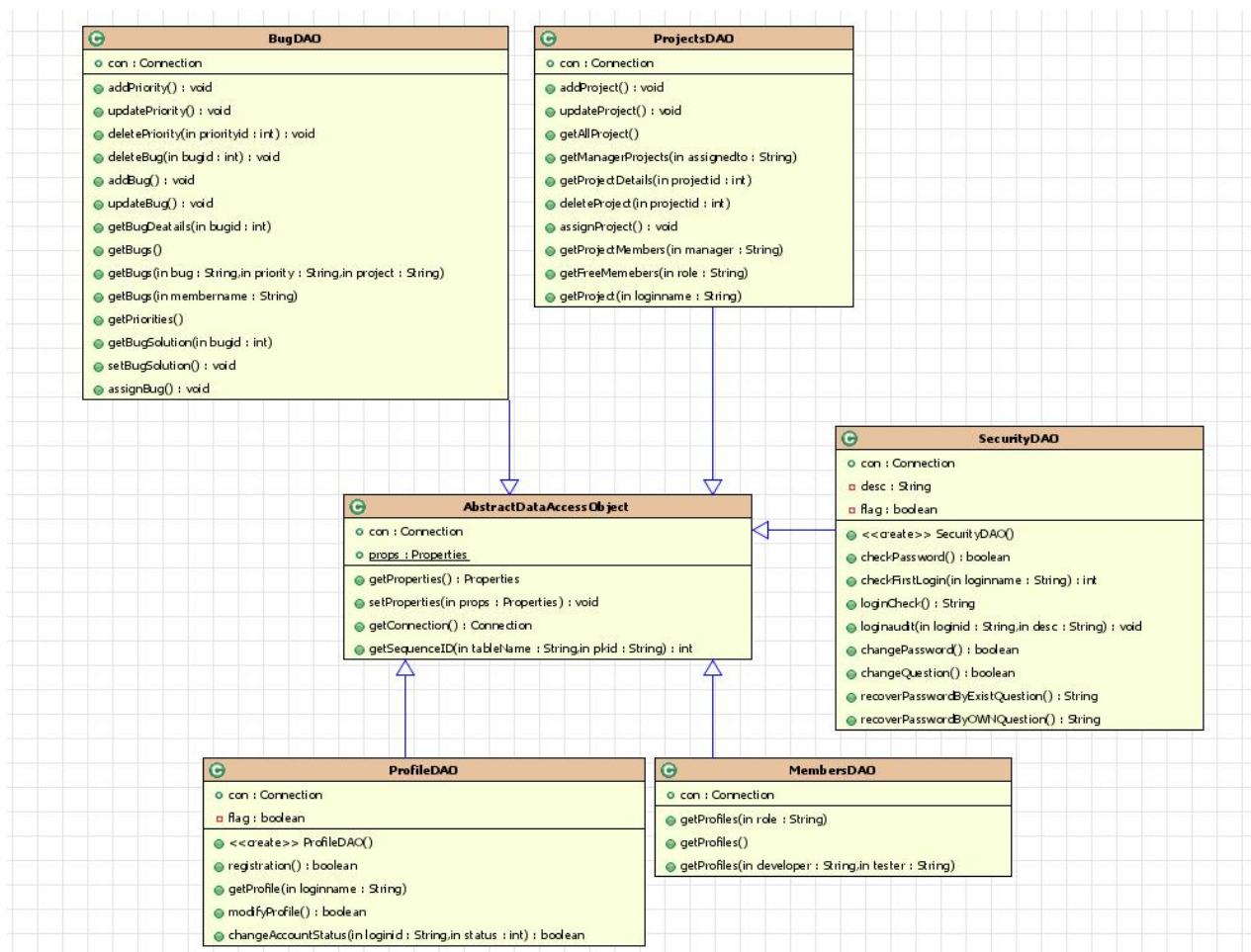


Fig.3.4.1 Shows a class diagram

SEQUENCE DIAGRAM :

- The Sequence diagram shows the ordering of the messages .
- It shows the order of the messages that the admin should perform.
- The interaction that takes place in a collaboration that either realizes a use case or an operation.
- High-level interactions between user of the system and the system,between the system and other systems,or between subsystems.

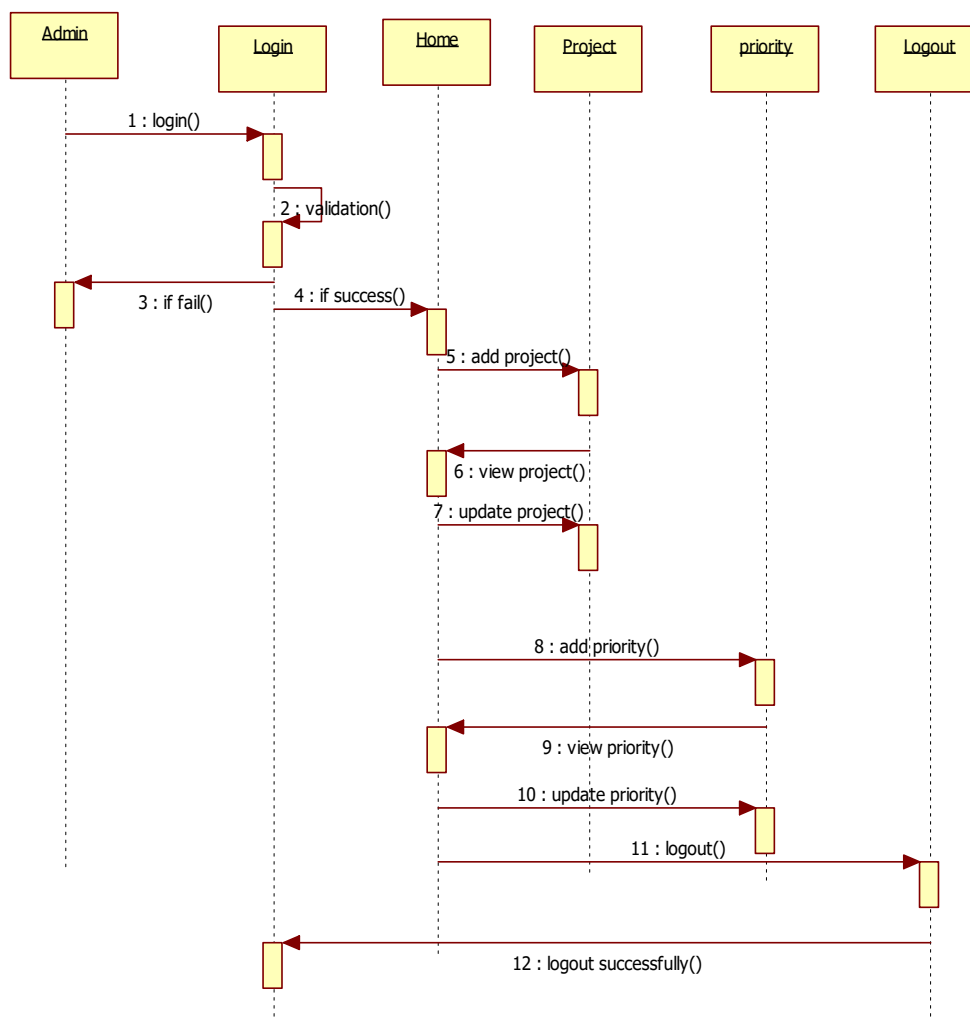


Fig. 3.4.1 Shows sequential diagram

COMPONENT DIAGRAM :

- A component diagram also known as a UML component diagram .
- It describes the organization and wiring of the physical components in a system.
- Component diagrams are often drawn to help model implementation details and double check that every aspect of the systems required.

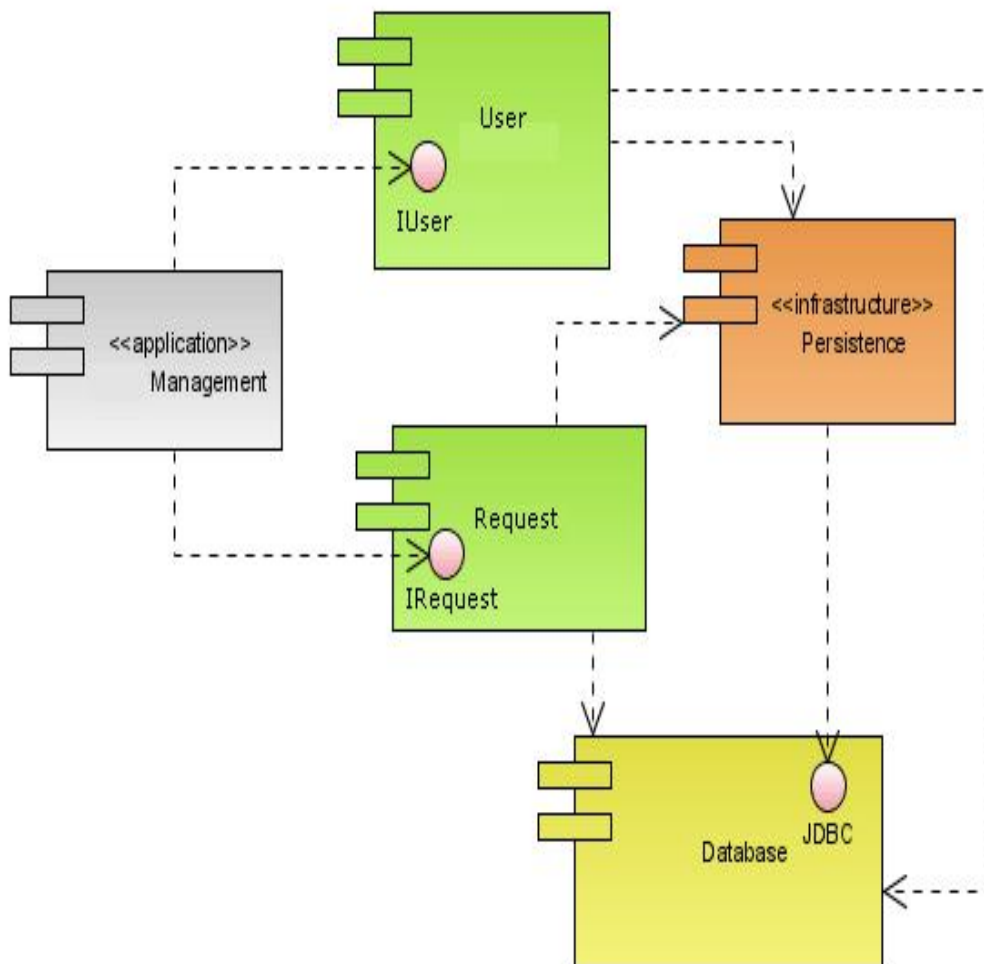


Fig .3.4.1 Shows a component diagram

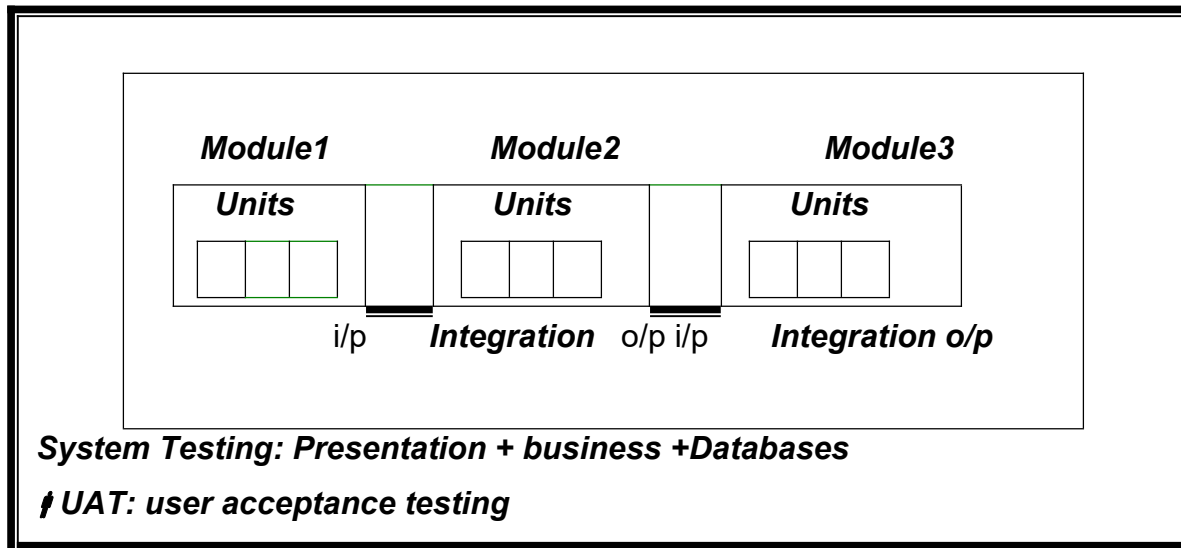
3.5. TEST PLAN:

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.
- Testing --- The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence
- Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script /program did not execute properly.

Testing Methodologies:

- **Black box Testing:** is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
Usually Test Engineers are involved in the black box testing.
- **White box Testing:** is the testing process in which tester can perform testing on an application with having internal structural knowledge.
Usually The Developers are involved in white box testing.
- **Gray Box Testing:** is the process in which the combination of black box and white box tonics' are used.

Levels of Testing:



STLC (SOFTWARE TESTING LIFE CYCLE)

Test Planning:

1. Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.
 2. This document involves the scope of testing,
 3. Objective of testing,
 4. Areas that need to be tested,
 5. Areas that should not be tested,
 6. Scheduling Resource Planning,
 7. Areas to be automated, various testing tools
- Used....

Test Development:

1. Test case Development (check list)
2. Test Procedure preparation. (Description of the Test cases).
3. Implementation of test cases. Observing the result.

Result Analysis:

1. Expected value: is nothing but expected behavior Of application.

2. Actual value: is nothing but actual behavior of application

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

Types Of Testing:

🌱 > **Smoke Testing:** is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

🌱 > **Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

🌱 > **Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

🌱 > **Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

🌱 **Static Testing:** is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

🌱 **Dynamic Testing:** is the testing which is performed on an application when it is being executed.ex: Functional testing.

🌱 **Alpha Testing:** it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

🌱 **Beta-Testing:** it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

🌱 **Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

🌱 **Compatibility testing:** it is the testing process in which usually the products are

tested on the environments with different combinations of databases (application servers, browsers...etc) In order to check how far the product is compatible with all these environments platform combination.

👤 **Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

👤 **Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

TCD (Test Case Document:

Test Case Document Contains

- **Test Scope (or) Test objective**
- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Academic details of student project:

Test scope:

- Test coverage is provided for the screen “ Academic status entry” form of a student module of university management system application
- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test

cases, Positive test cases, Negative test cases respectively

3.6. SYSTEM REQUIREMENTS

3.6.1. Software Requirements

Operating System	:	Windows XP/2003 or Linux/Solaris
User Interface	:	HTML, CSS
Client-side Scripting	:	JavaScript
Programming Language	:	Java
Web Applications	:	JDBC, JNDI, Servlets, JSP
IDE/Workbench	:	MyEclipse8.6
Database	:	Oracle10G
Server Deployment	:	Tomcat6.0

3.6.2. Hardware Requirements

Processor	:	Pentium IV
Hard Disk	:	40GB
RAM	:	256MB

3.7 NON-FUNCTIONAL REQUIREMENTS:

Performance Requirements

Some Performance requirements identified is listed below:

- The database shall be able to accommodate a minimum of 10,000 records of students.
- The software shall support use of multiple users at a time. There are no other specific performance requirements that will affect development.

Security Requirements

Some of the factors that are identified to protect the software from accidental or malicious access, use, modification, destruction, or disclosure are described below.

Specific requirements in this area could include the need to:

- Utilize certain cryptographic techniques
- Keep specific log or history data sets
- Assign certain functions to different modules
- Restrict communications between some areas of the program
- Check data integrity for critical variables
- Later version of the software will incorporate encryption techniques in the user/license authentication process.
- The software will include an error tracking log that will help the user understand what error occurred when the application crashed along with suggestions on how to prevent the error from occurring again.
- Communication needs to be restricted when the application is validating the user or license.(i.e., using https)

Portability Requirements

Some of the attributes of software that relate to the ease of porting the software to other host machines and/or operating systems.

This may include:

Java is used to develop the product. So it is easiest to port the software in any environment.

Maintainability

The user will be able to reset all options and all stored user variables to default settings.

Reliability

Some of the attributes identified for the reliability is listed below:

- All data storage for user variables will be committed to the database at the time of entry.
- Data corruption is prevented by applying the possible backup procedures and techniques.

Usability requirements

Some of the usability requirements identified for this system are listed below:

- A logical interface is essential to an easy to use system, speeding up common tasks.
- Error prevention is integral to the system and is provided in a number of formats from sanity hacks to limiting free-text input.

Availability

- All cached data will be rebuilt during every startup. There is no recovery of user data if it is lost. Default values of system data will be assigned when necessary

Software System Attributes

- There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. The following items provide a partial list of examples.
- The input system will allow for inputting numbers, operands, special symbols and letters of the alphabet.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 SUMMARY

Defect Tracking for Improving Product Quality and Productivity for Improving Software **Reliability** is an automated system that can be useful to employees and the managers in any functional organization. Defect Tracking for Improving Product Quality and Productivity gives the facility to define the tasks in the organization and also allows the managers to track the Defects spent by the developer for that particular task. A report generation facility is supported in DTS that allows the managers to analyze which are those skills by employee are utilized and those which are not utilized. project or application. This tool helps employees to document their Defects and analyze.

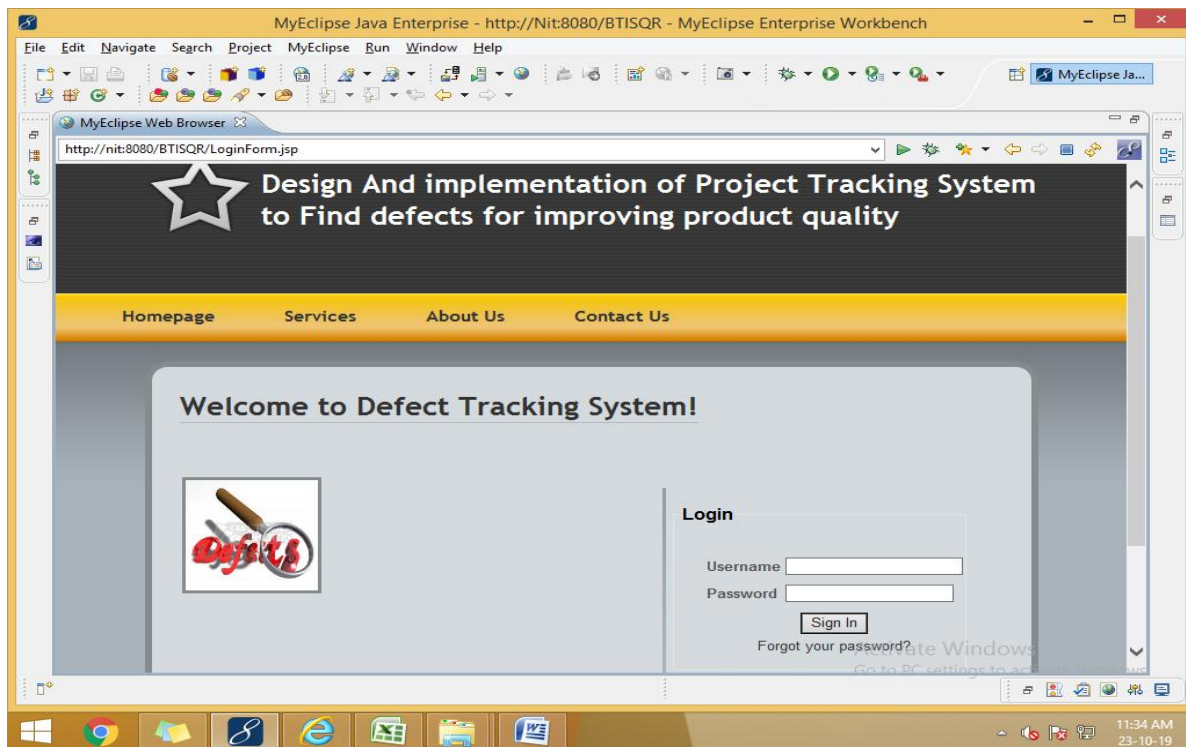
Defect Tracking System is very useful for removing defects from project module. Only if features mentioned in document are extended. In future users may also be possibly notifying the name/Id of defect creator so that organization can prevent their systems.

Defect Tracking for Improving Product Quality and Productivity gives the facility to define the tasks in the organization and also allows the managers to track the Defects spent by the developer for that particular task. A report generation facility is supported in DTS that allows the managers to analyze which are those skills by employee are utilized and those which are not utilized. project or application. This tool helps employees to document their Defects and analyze.

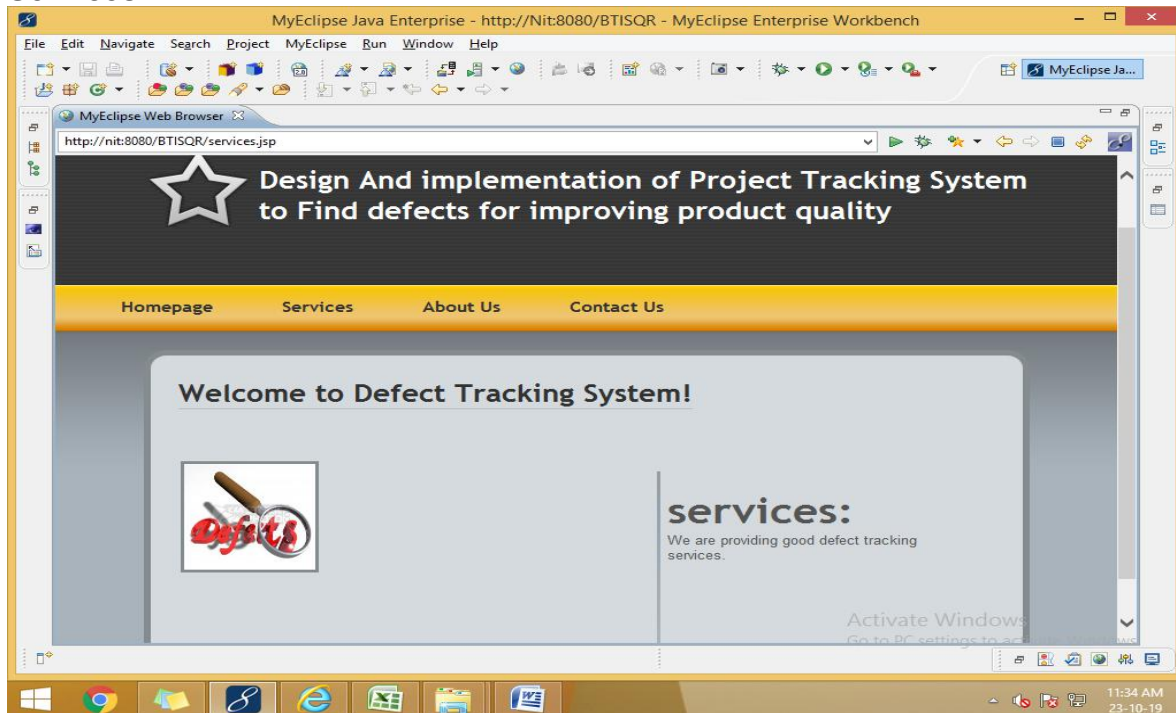
4.2 SCREEN SHOTS

Admin Screens:

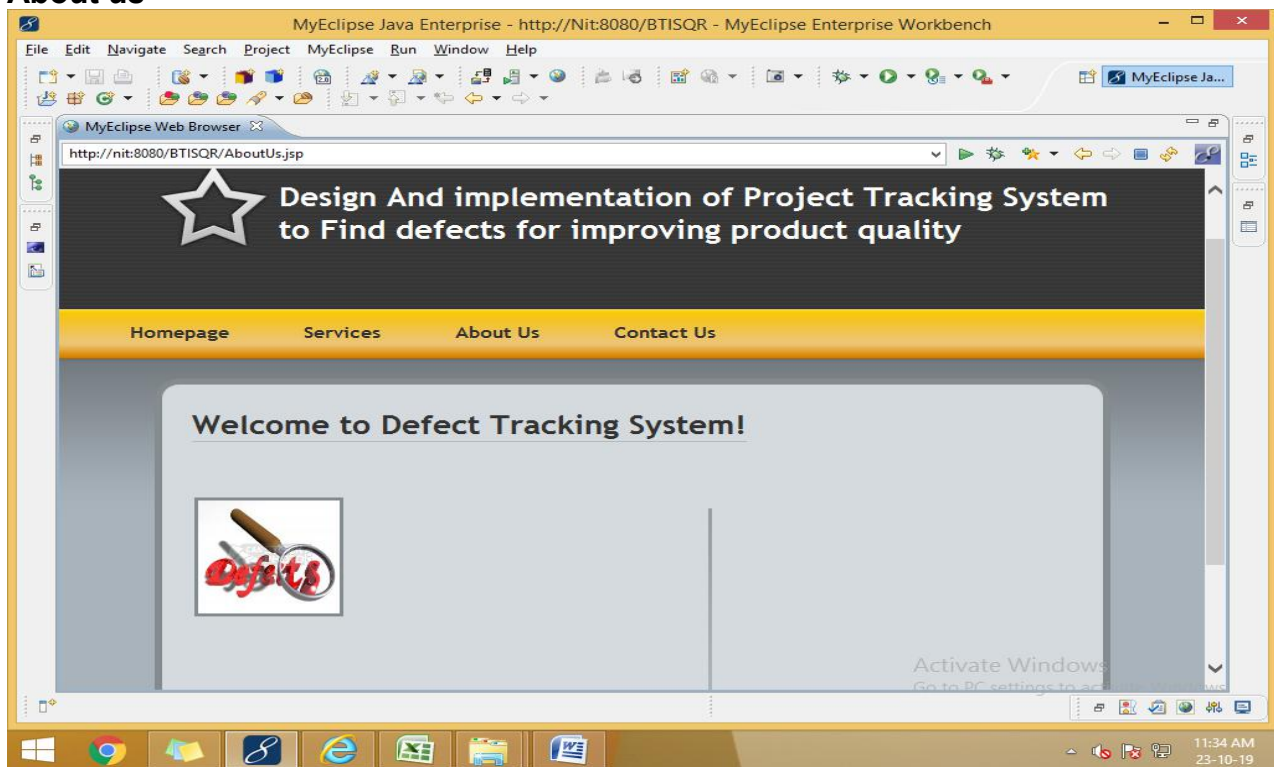
Home page



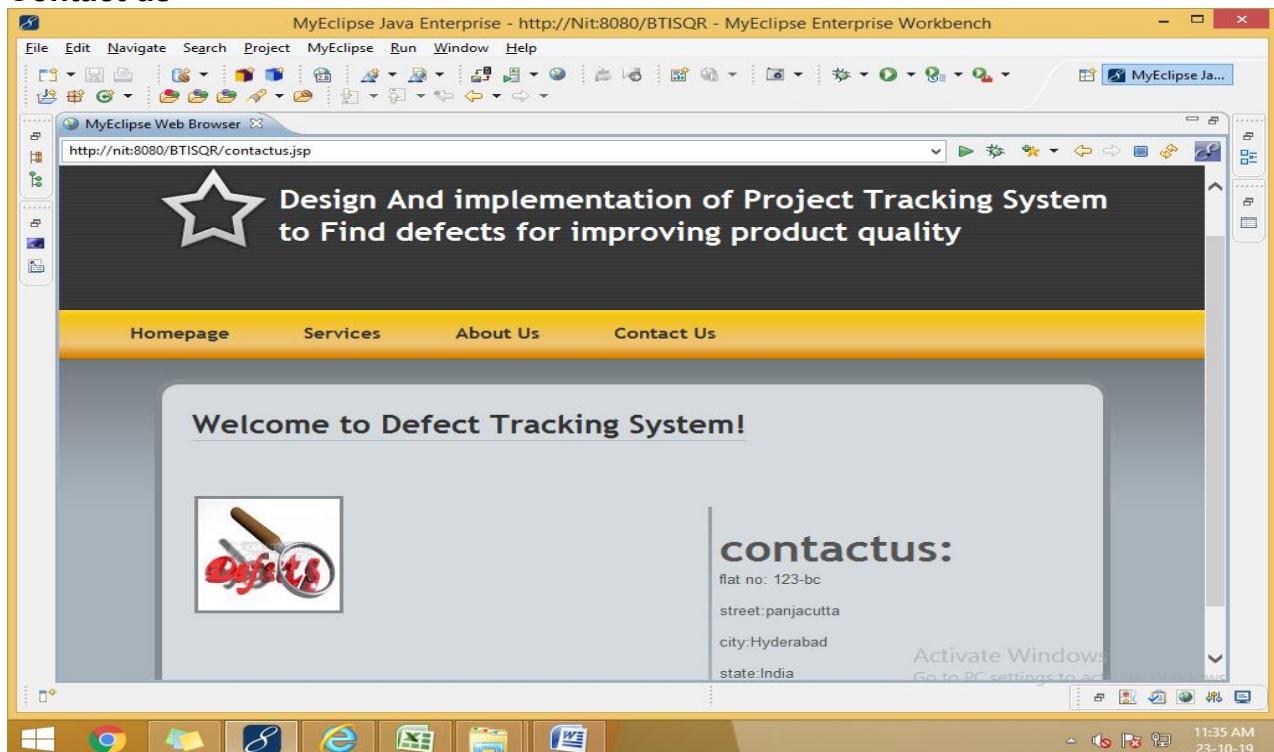
Services



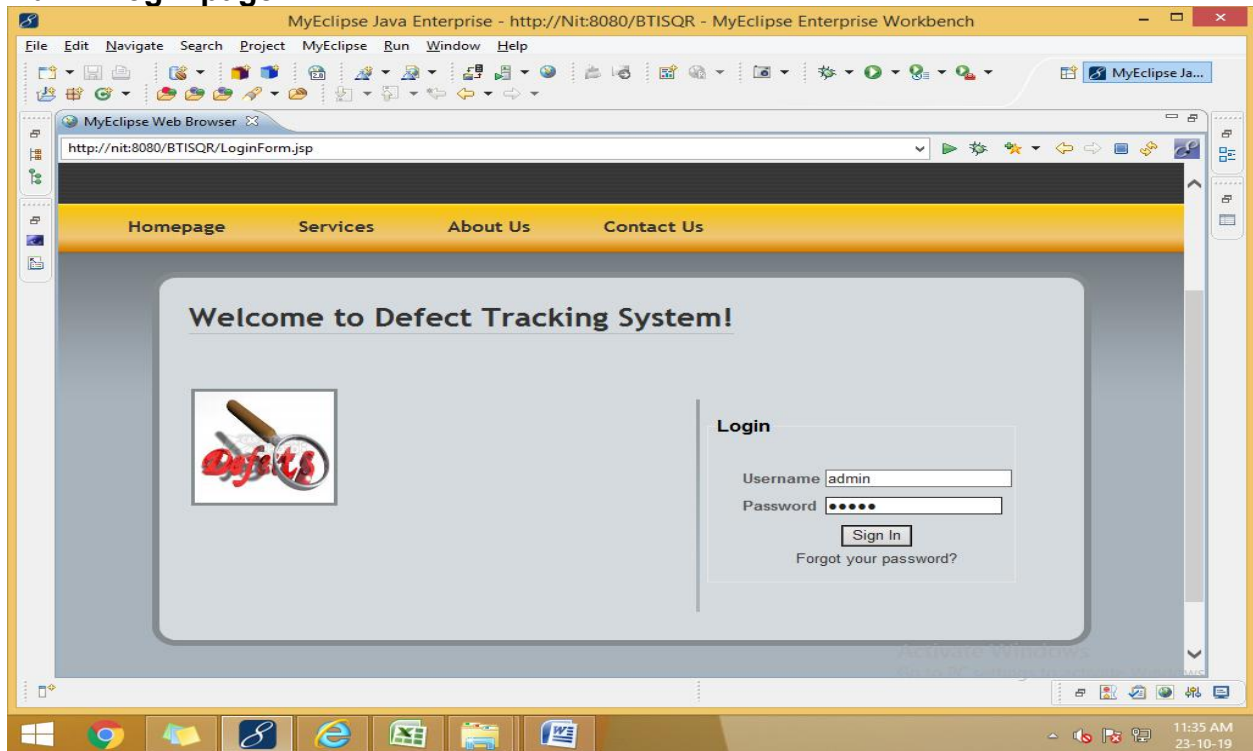
About us



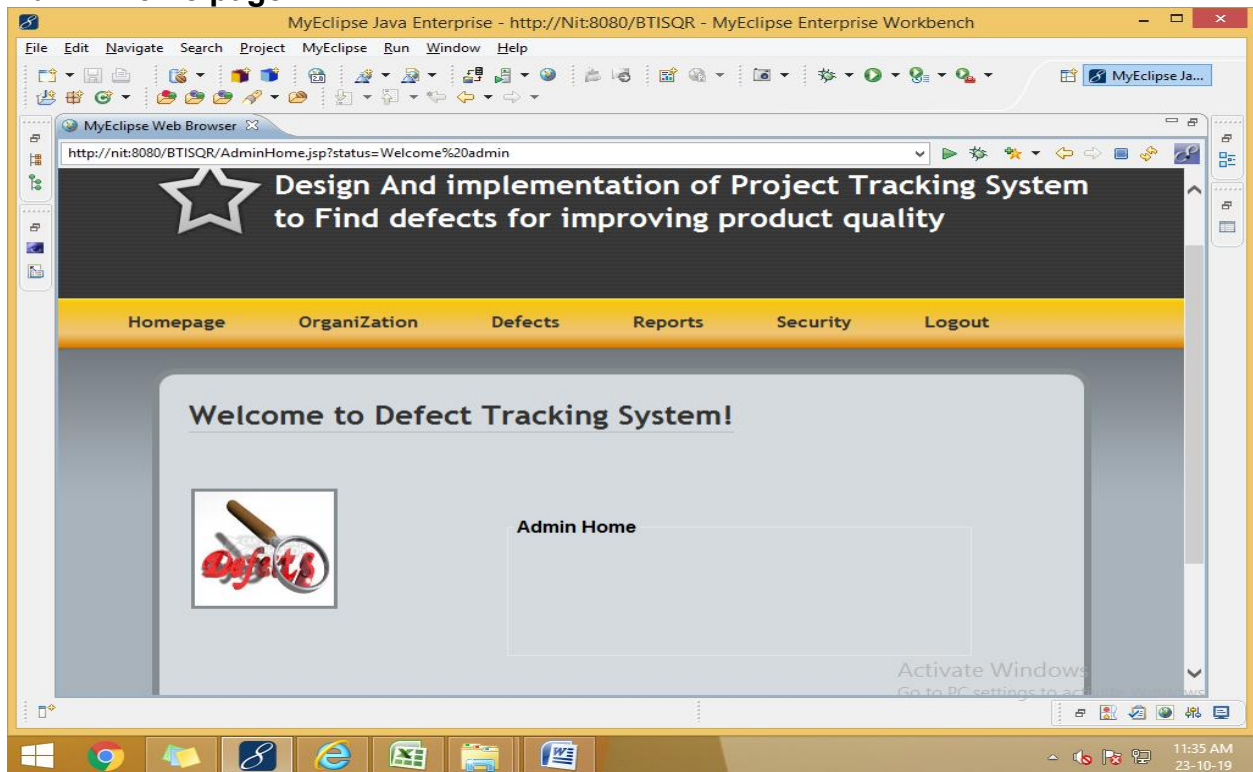
Contact us



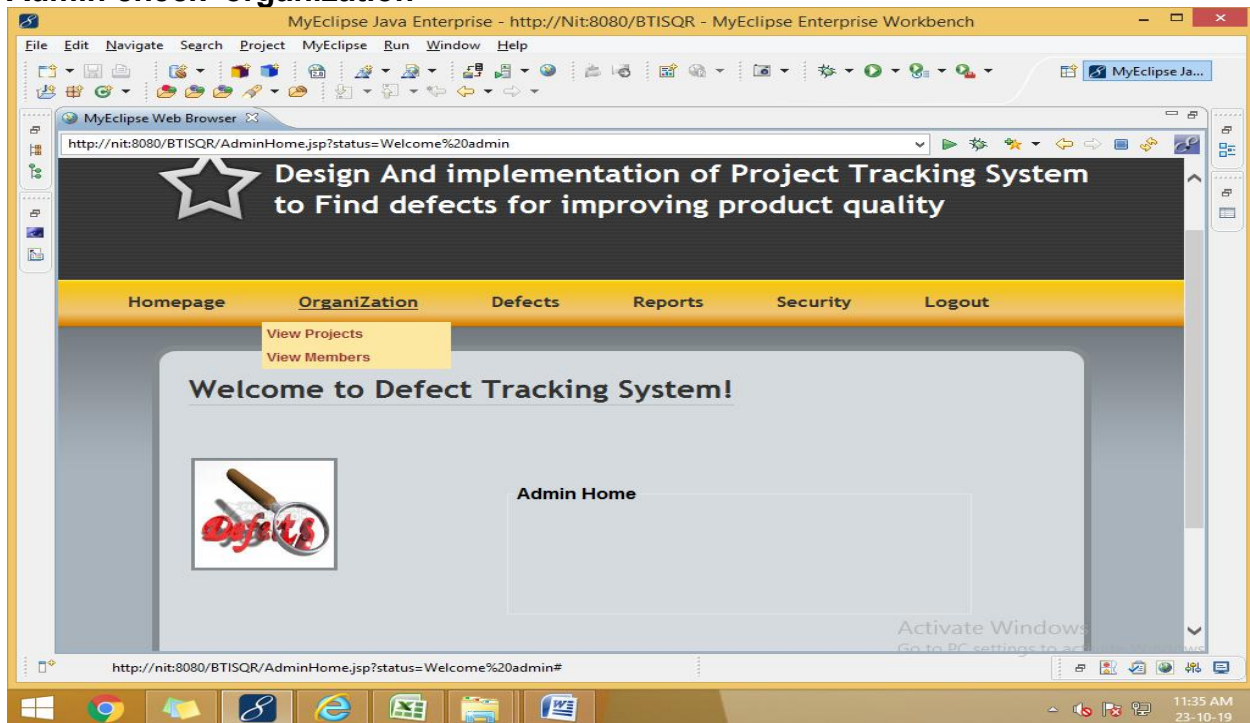
Admin login page



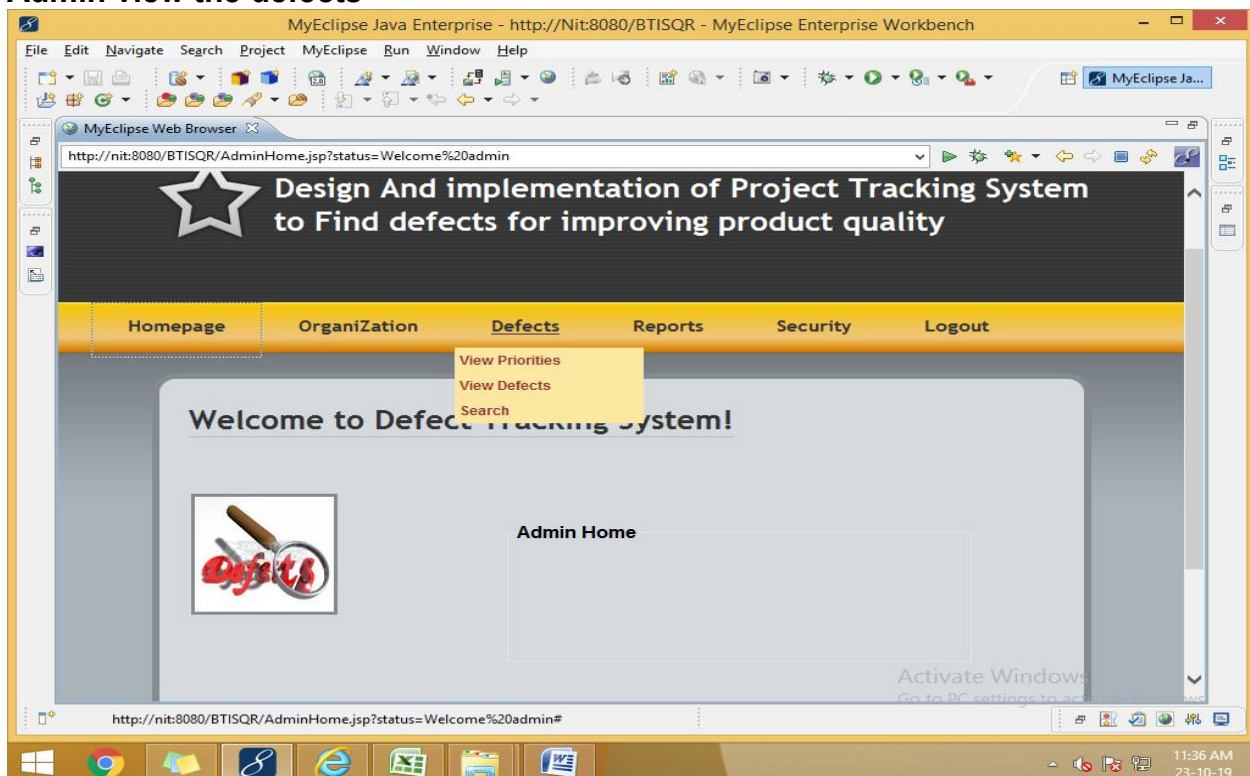
Admin home page



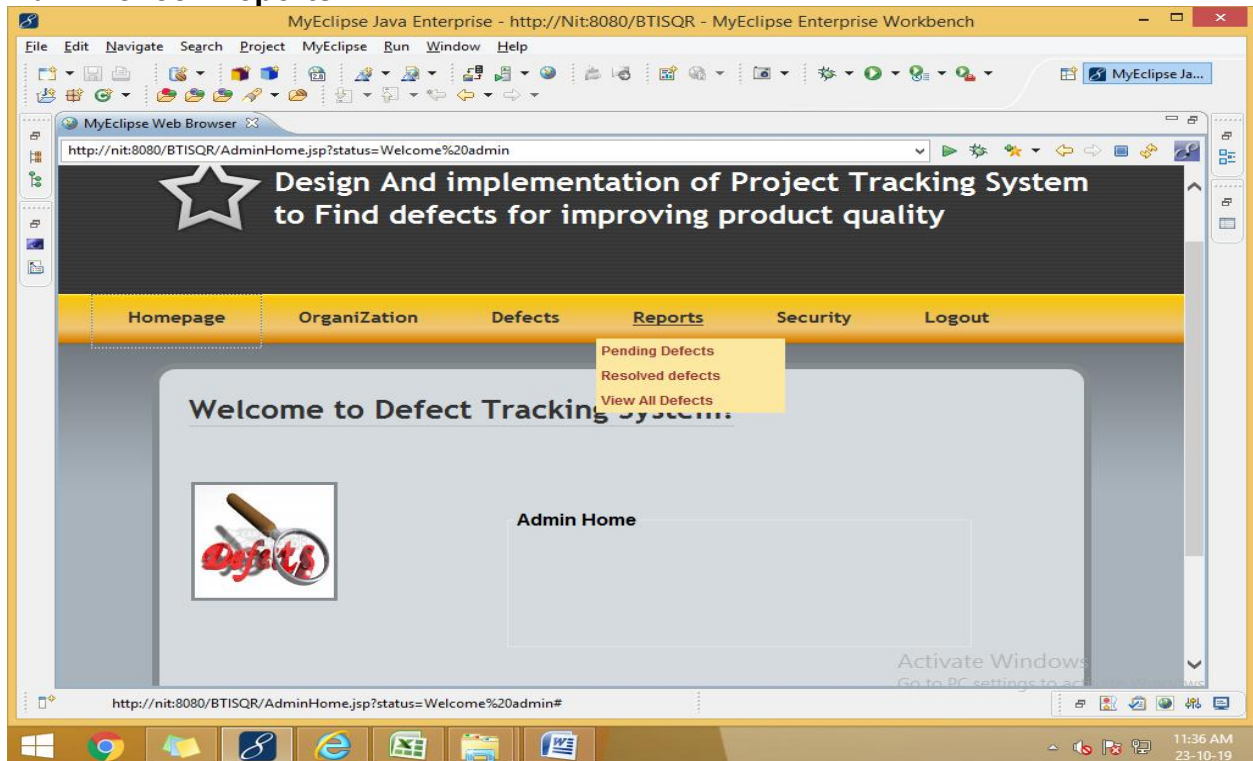
Admin check organization



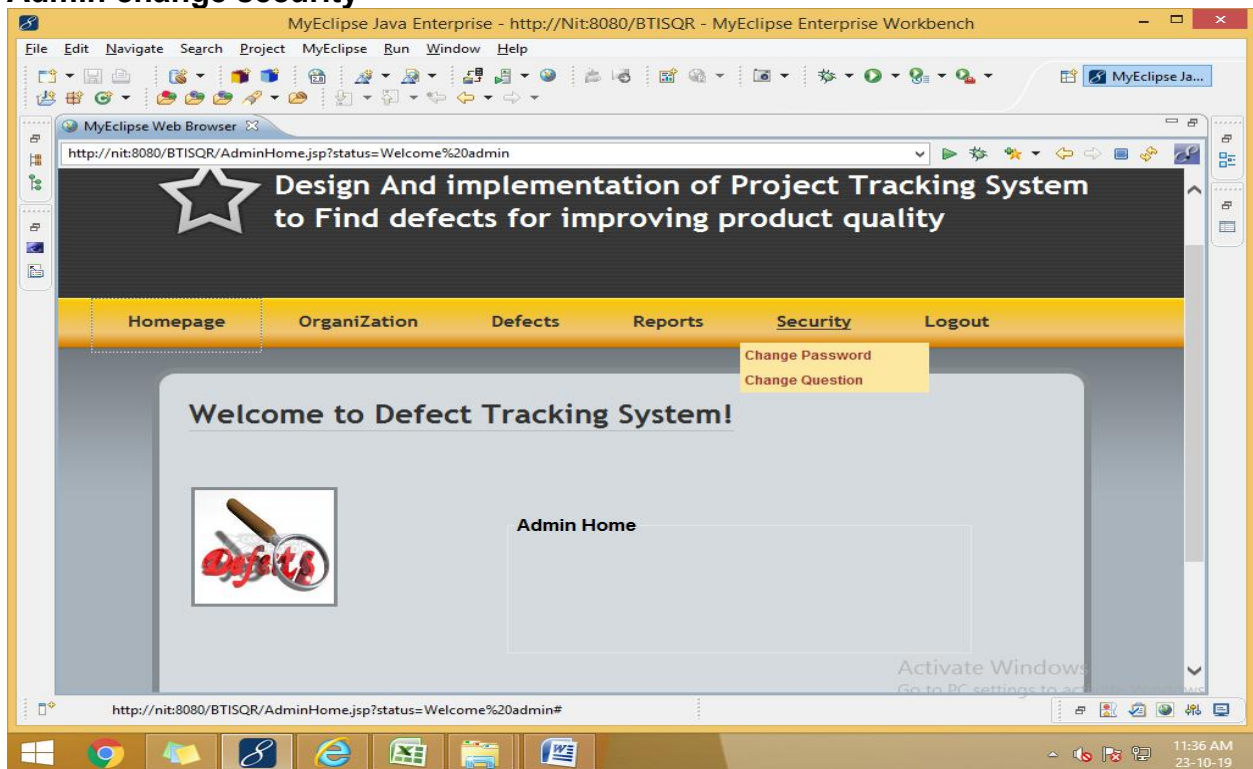
Admin view the defects



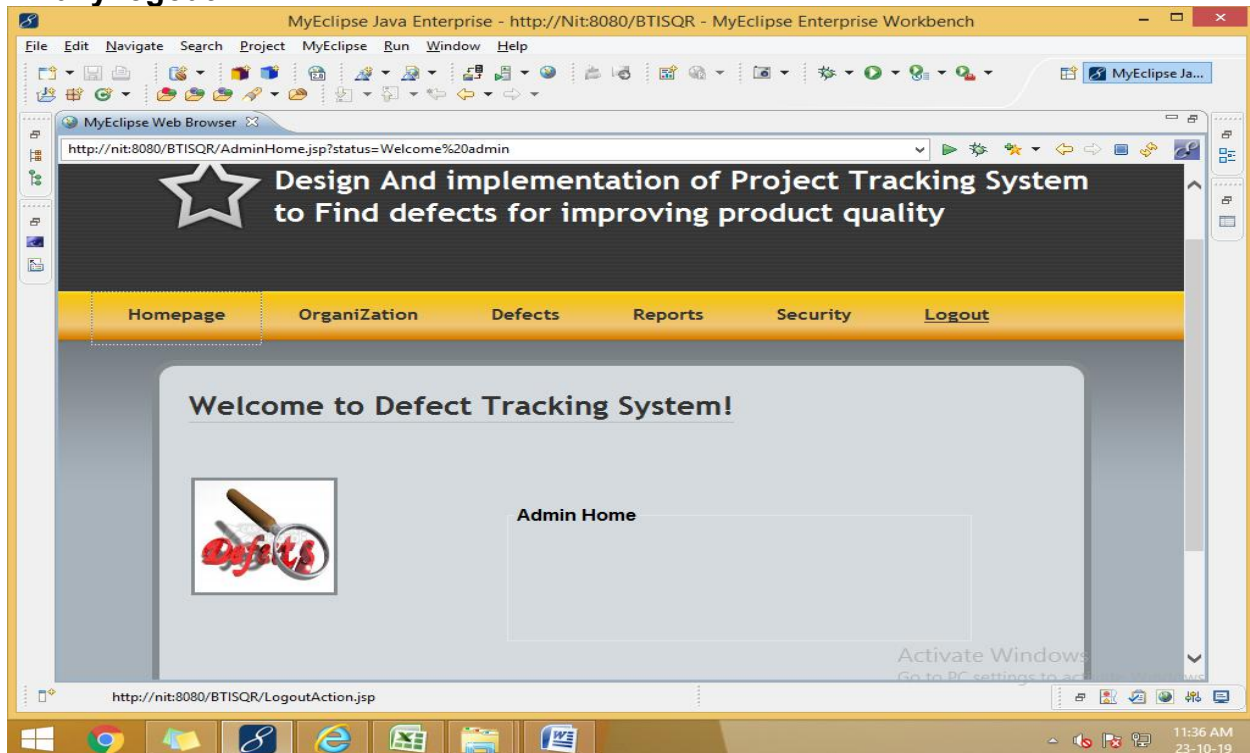
Admin check reports



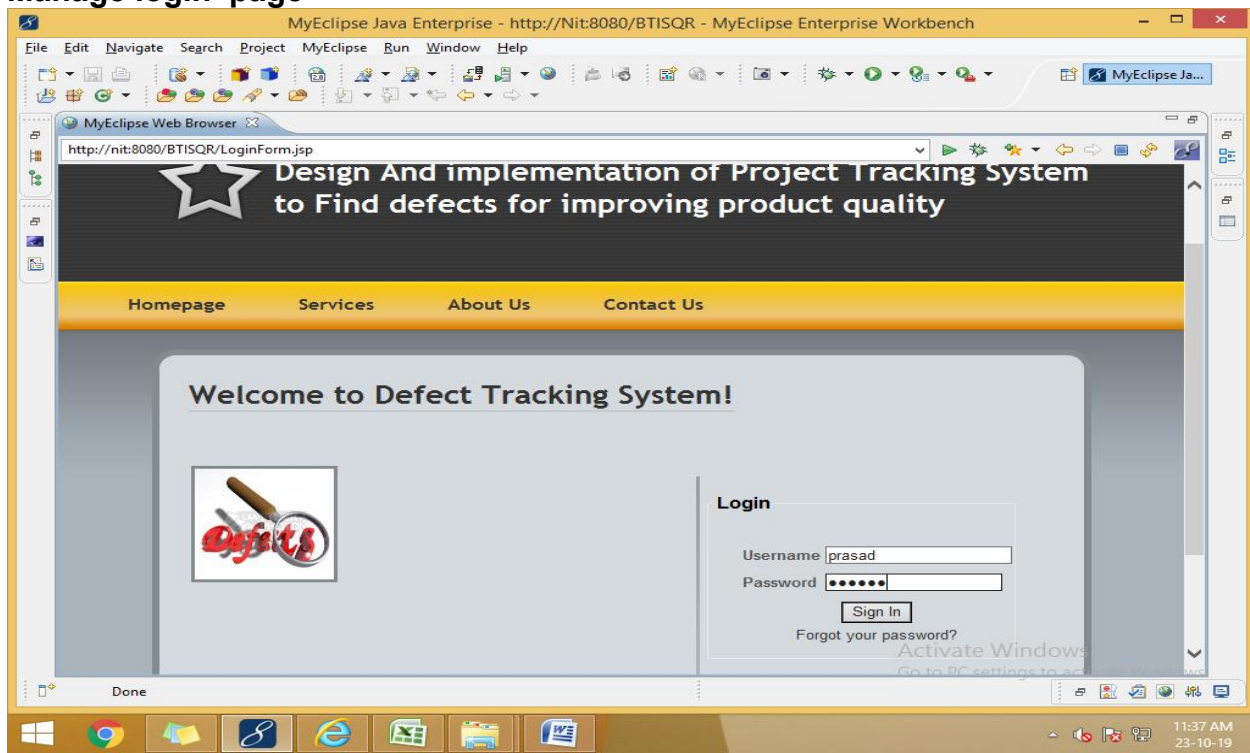
Admin change security



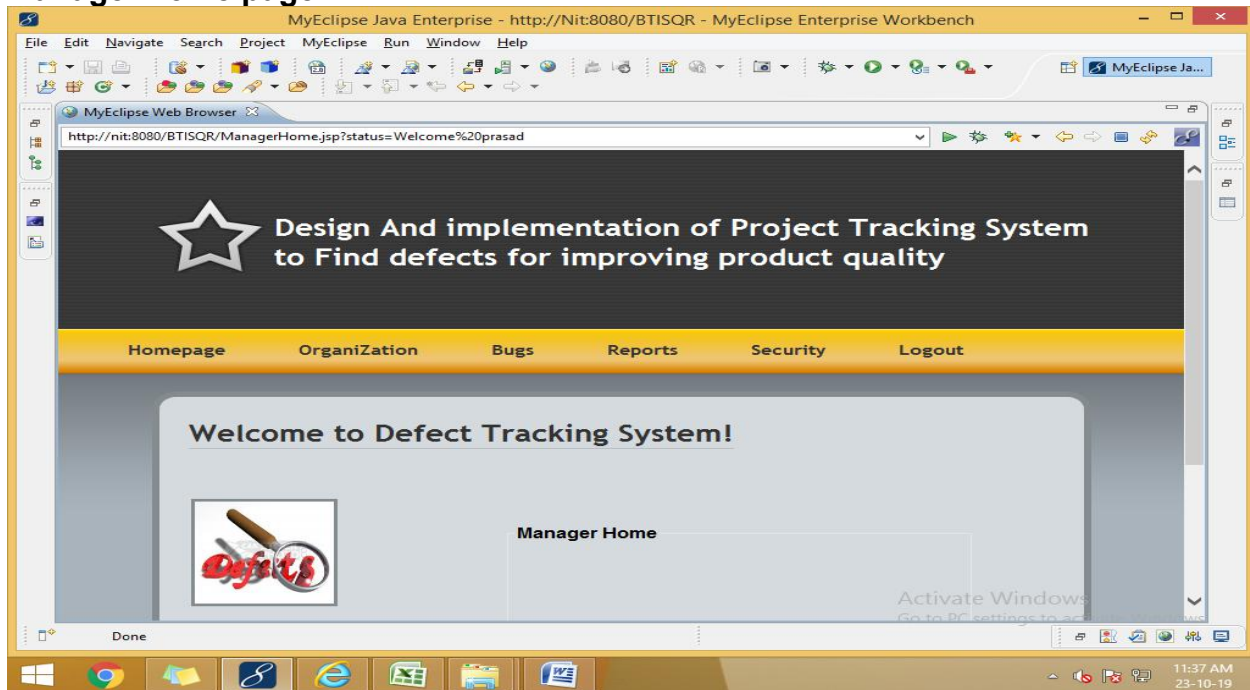
Finally logout



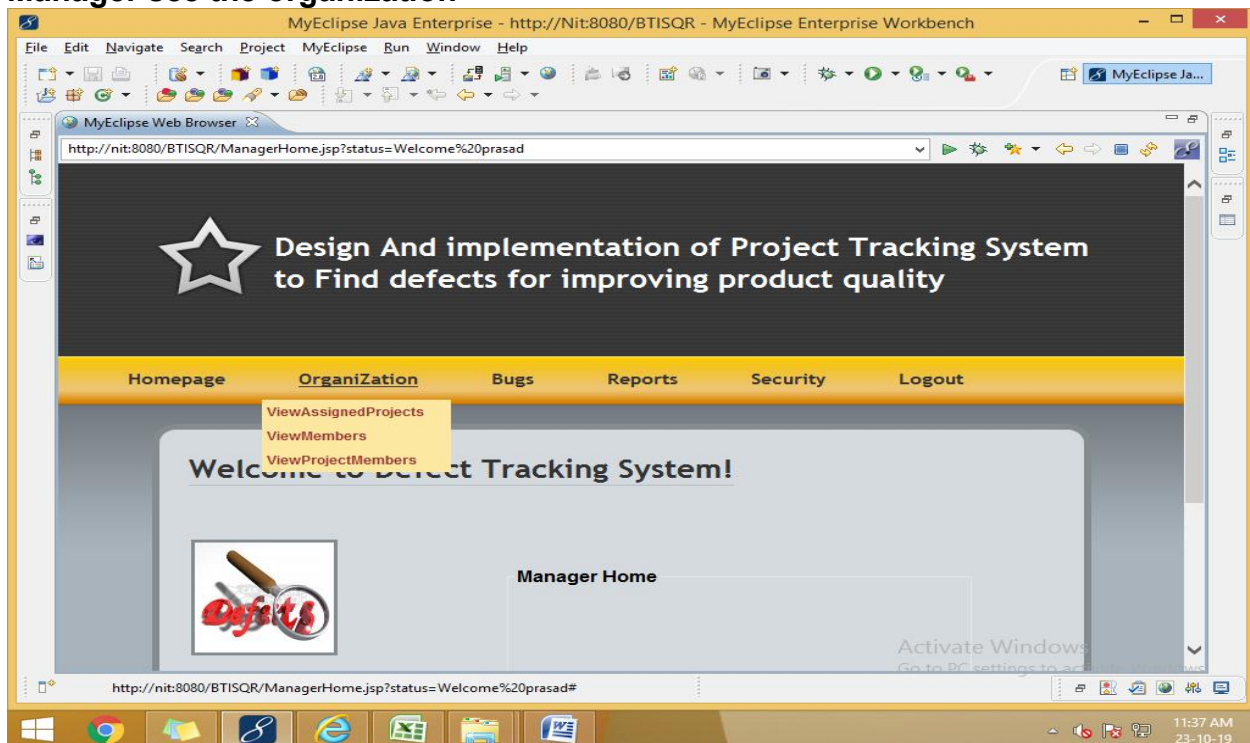
Manage login page



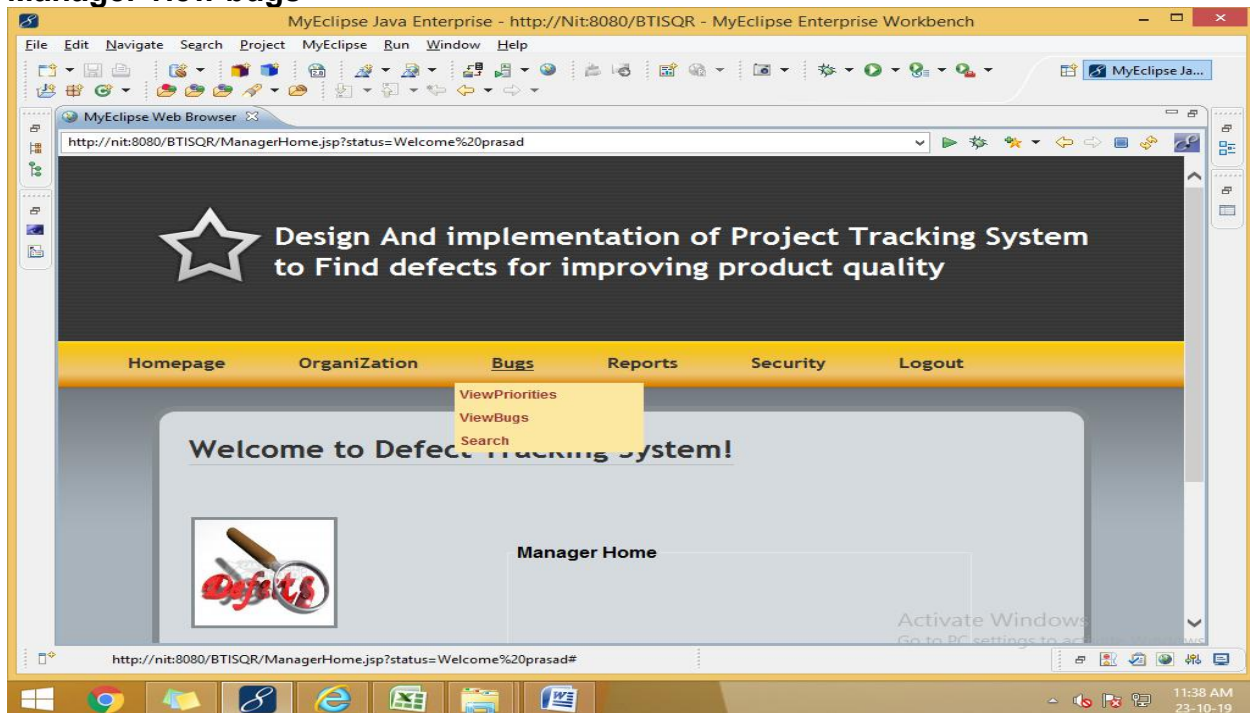
Manager home page



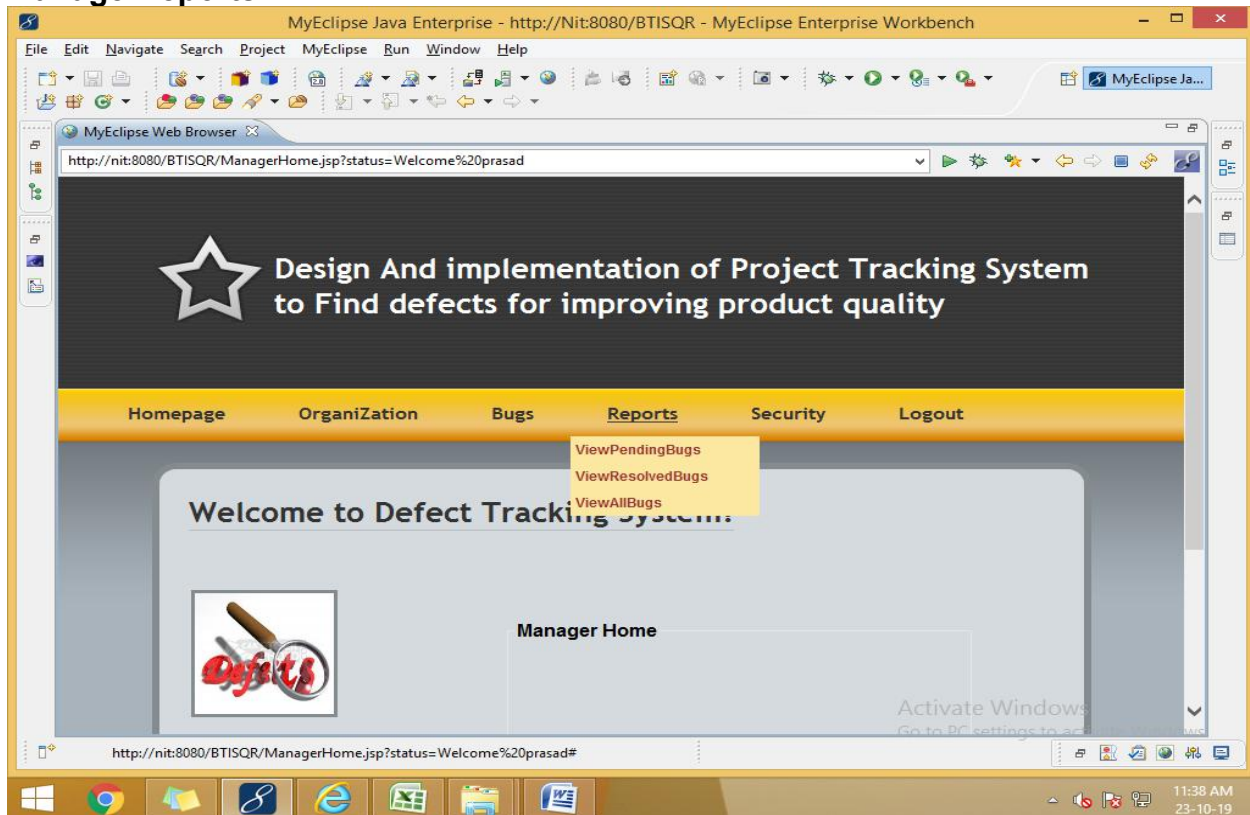
Manager see the organization



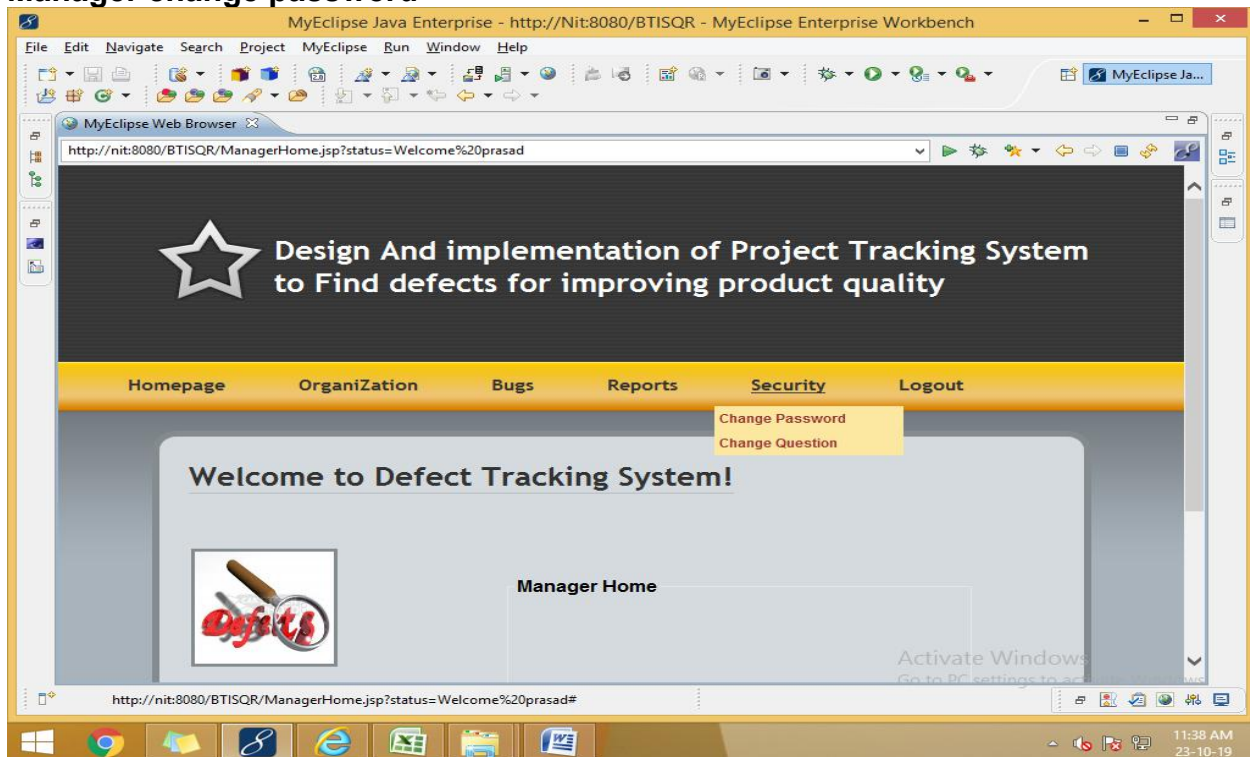
Manager view bugs



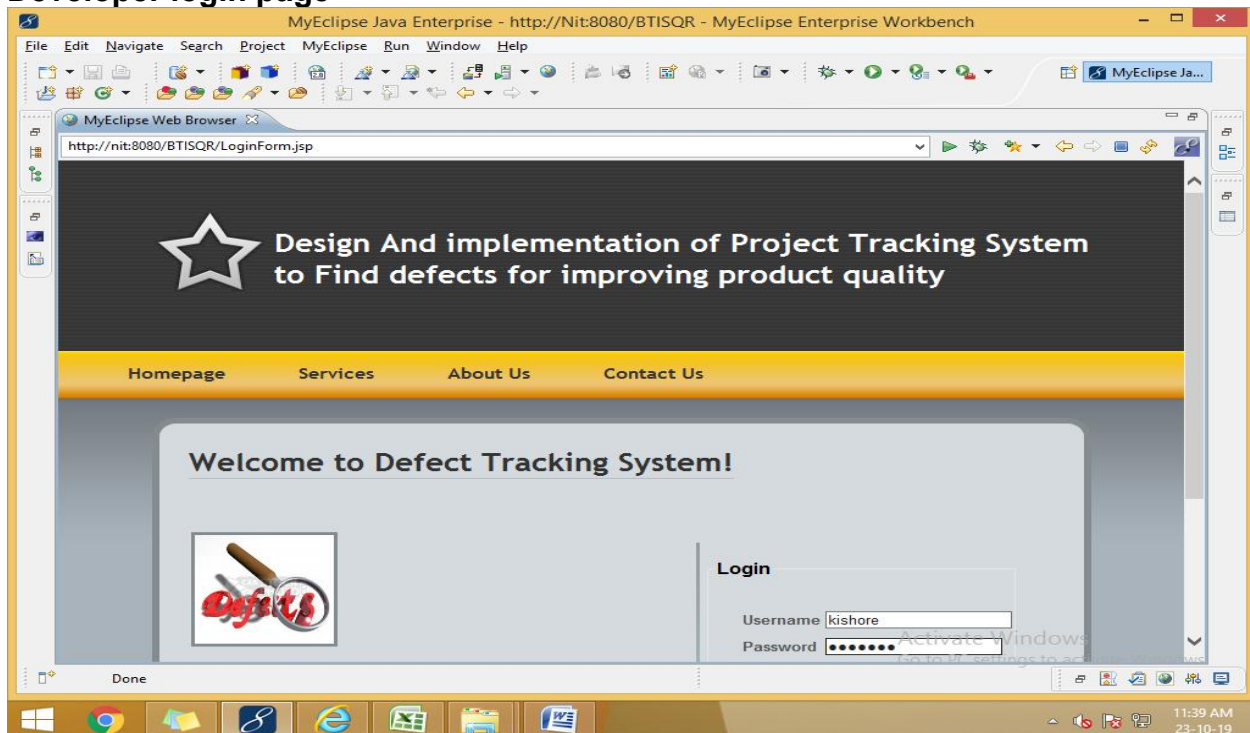
Manager reports



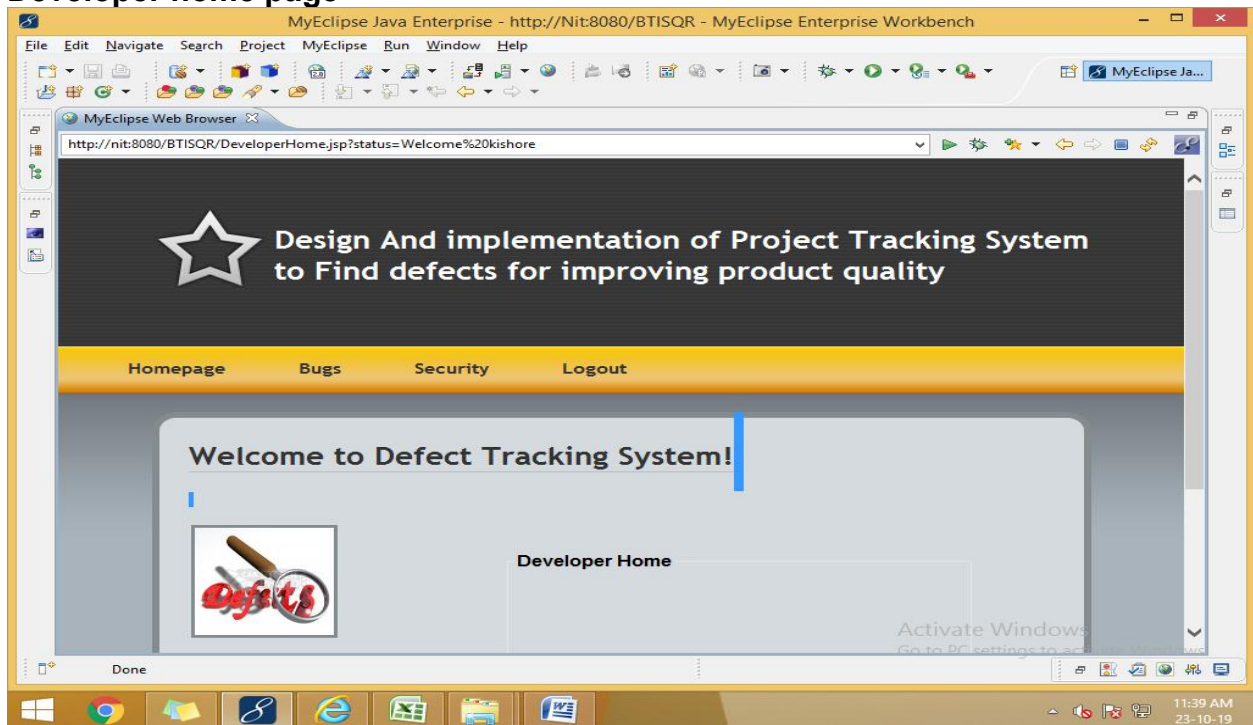
Manager change password



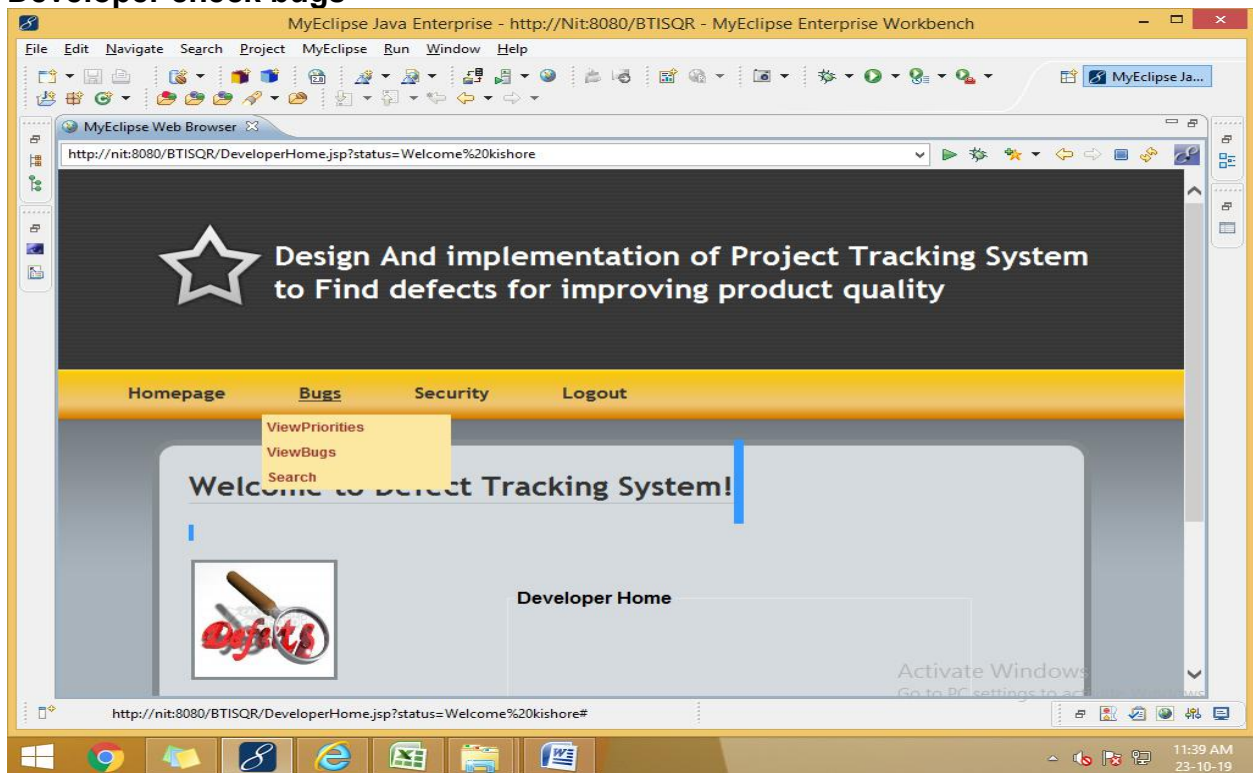
Developer login page



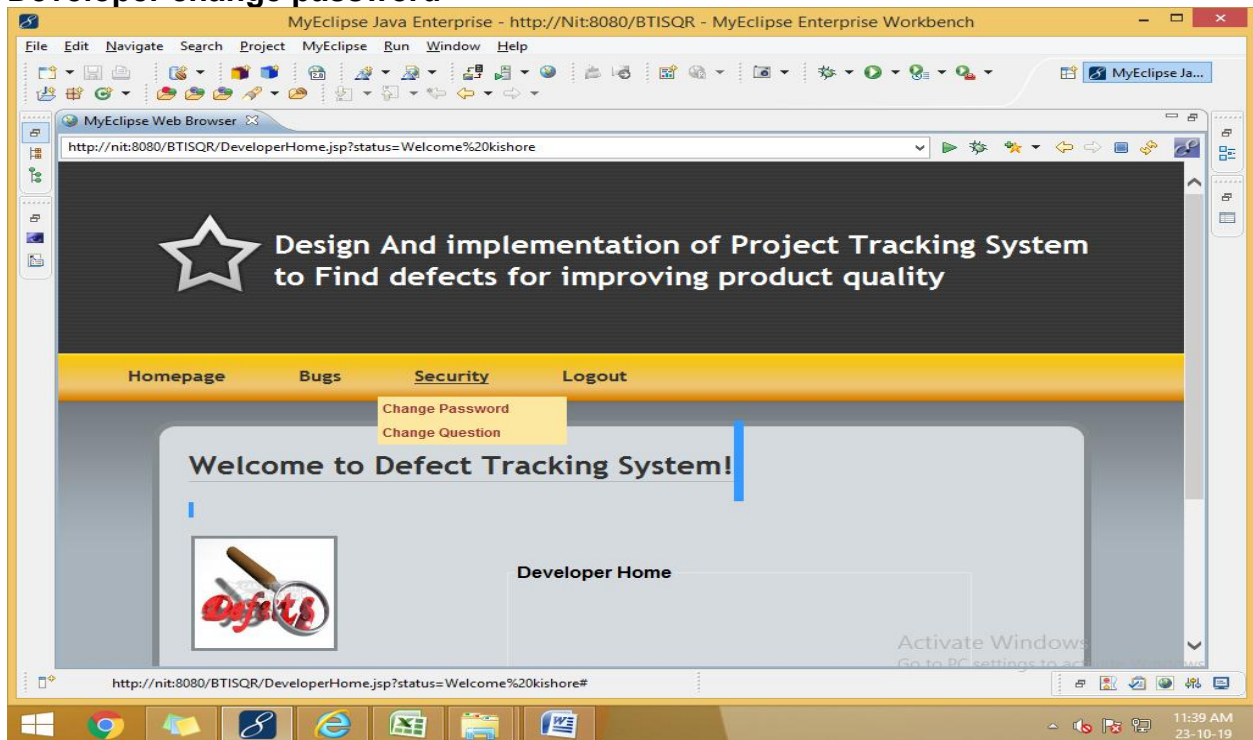
Developer home page



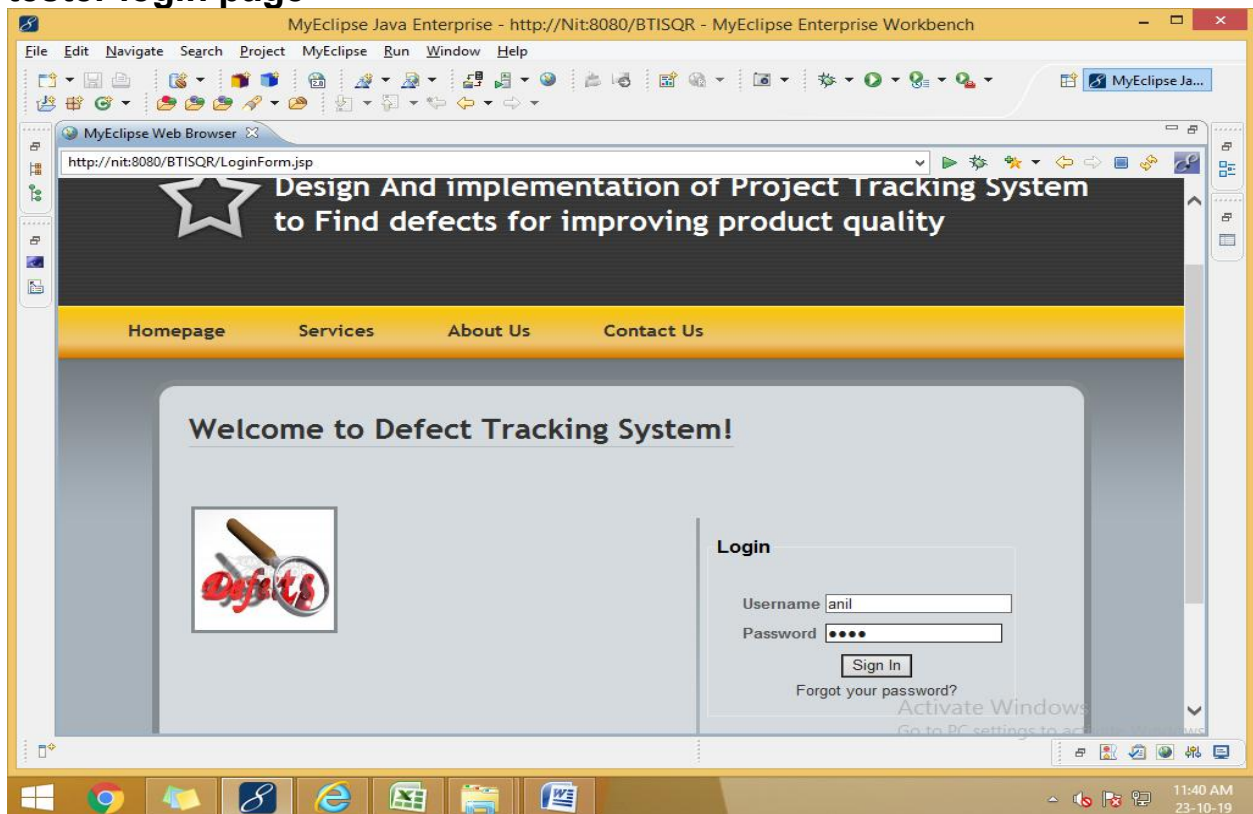
Developer check bugs



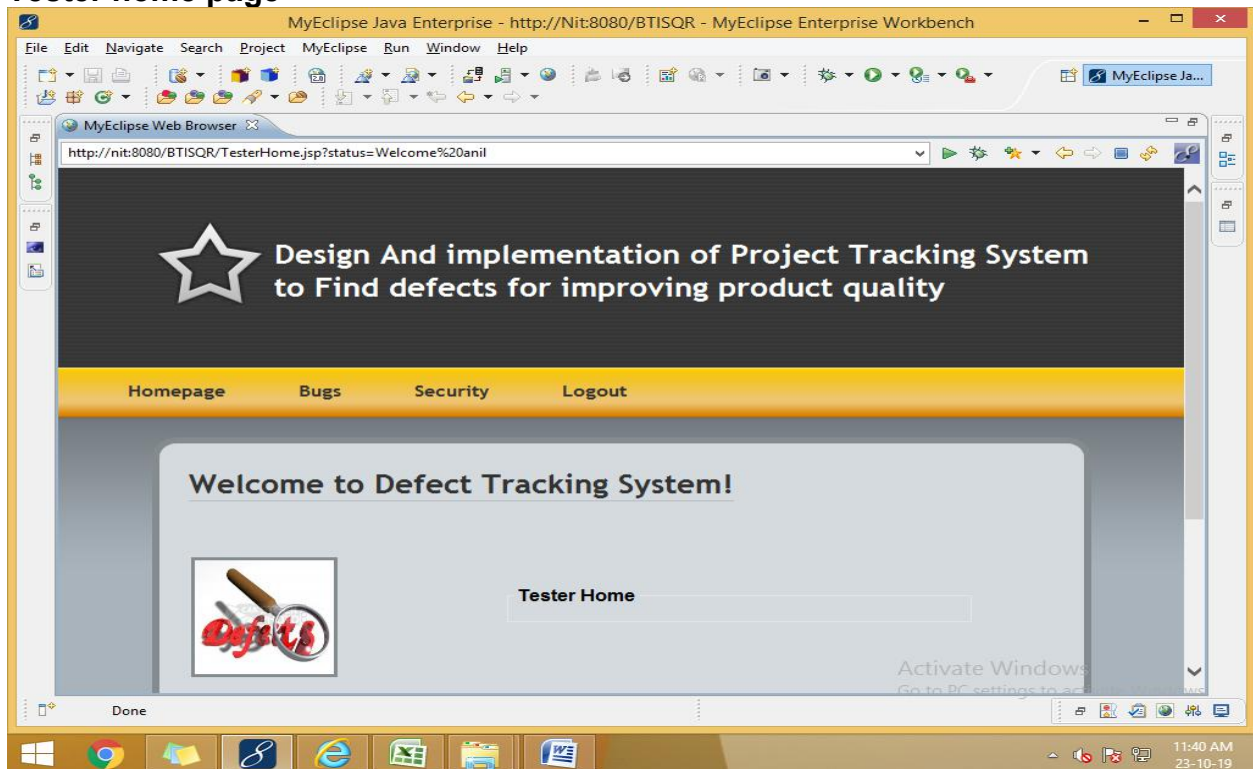
Developer change password



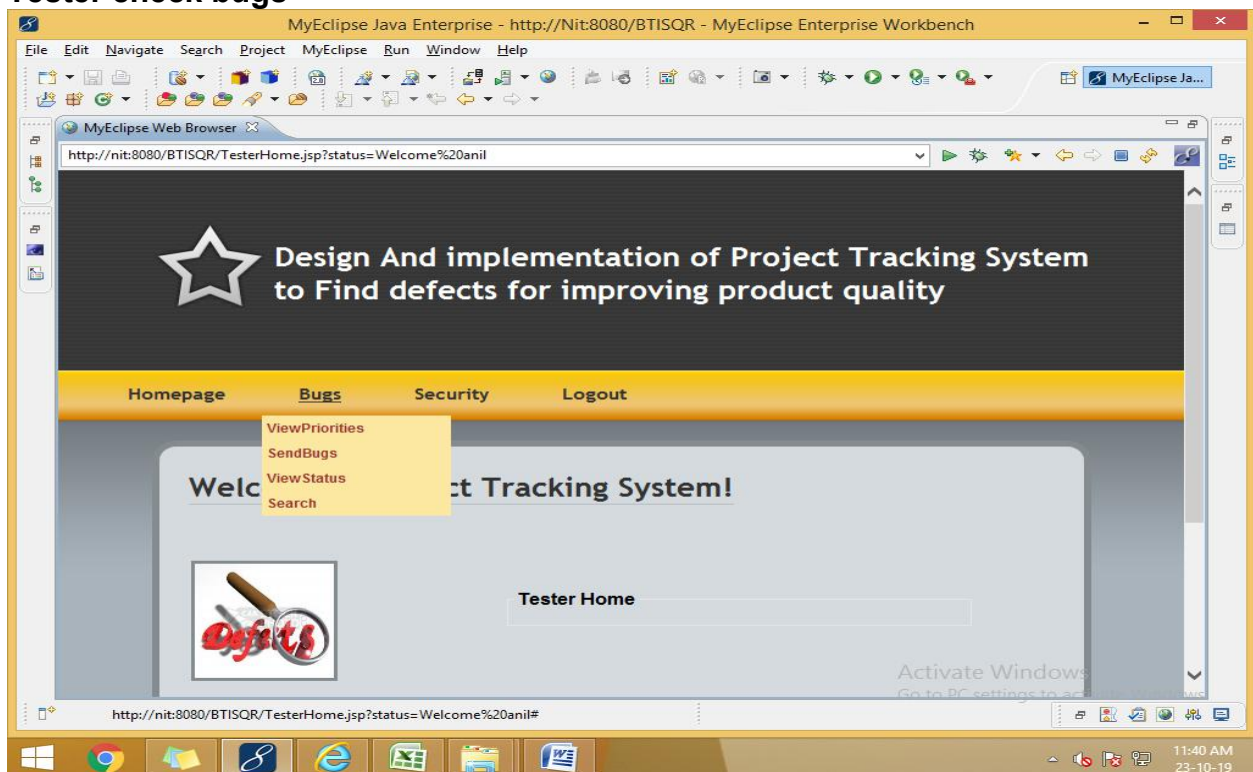
tester login page



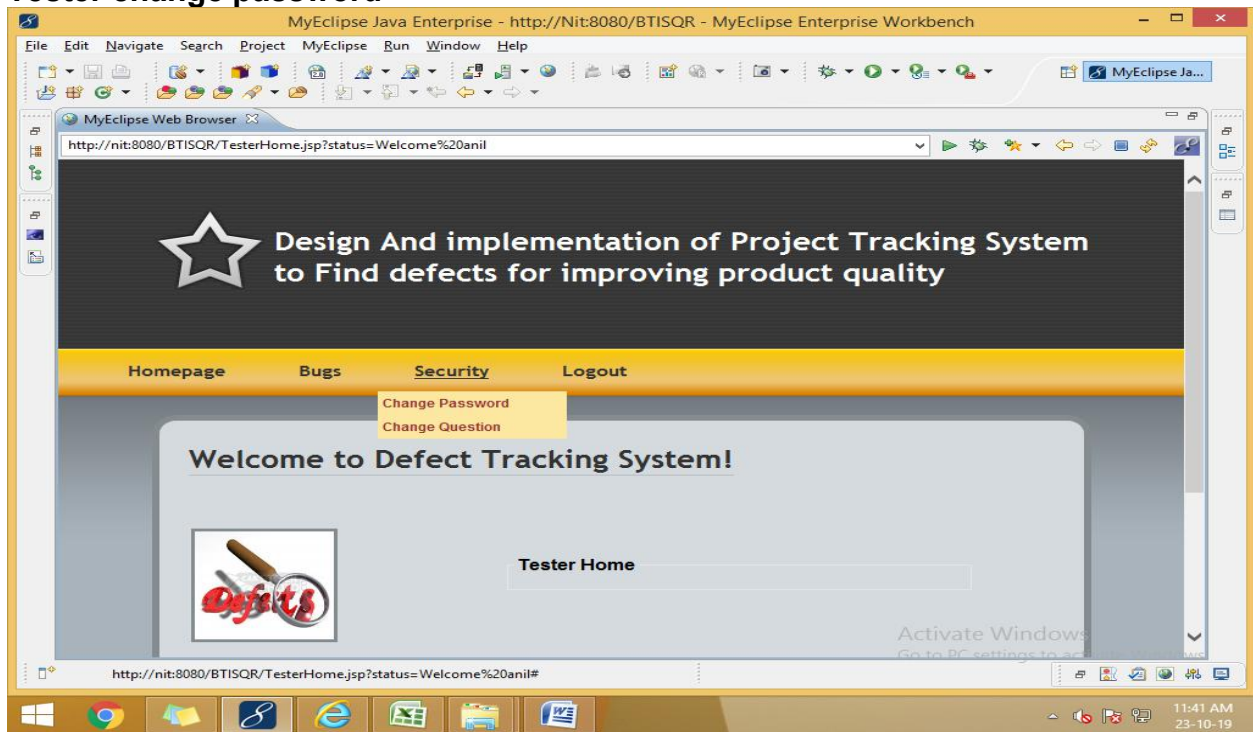
Tester home page



Tester check bugs



Tester change password



CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1. CONCLUSION

Defect Tracking System is very useful for removing defects from project module. Only if features mentioned in document are extended. In future users may also be possibly notifying the name/Id of defect creator so that organization can prevent their systems. The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- (i) It's a web-enabled scheme.
- (ii) This task offers user to enter the data through simple and interactive forms. This is very useful for the client to enter the desired information through so much simplicity.
- (iii) The user is mainly more worried about the validity of the data, whatever he is entering. There are checkson every stages of any new creation, data entry or update so that the user cannot enter the invalid data, which can build problems at later date.
- (iv) Sometimes the user finds bug in the later stages of using Project that he needs to update some of the Information that he entered earlier. There are options for him by which he can update the records. Moreover there is limitation for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.
- Current Defect tracking systems do not effectively elicit all of the information needed by developers. Without this information developers cannot resolve defects in atimely fashion and so we consider that improvements to the way defect tracking systems collect information are essential. This is likely to speed up the process of resolving bugs. In the future, I will move from the current prototype of the interactive system to a full-scale system that can deal with a variety of information to gather.

5.2. FUTURE ENHANCEMENT

Limitations of the system:

- Only the permanent employees can access the system.
- System works in all platforms and its compatible environments.
- Advanced techniques are not used to check the authorization.

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

- 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

5.3. REFERENCES :

- N. Schuessler and K. Axhausen, "Processing raw data from global positioning systems without additional information," *Transp. Res. Rec., J. Transp. Res. Board*, no. 2105, pp. 28–36, 2009.
- J. Wolf, "Using GPS data loggers to replace travel diaries in the collection of travel data," Georgia Inst. Technol., Citeseer, Tech. Rep., 2000.
- J. Ogle, R. Guensler, W. Bachman, M. Koutsak, and J. Wolf, "Accuracy of global positioning system for determining driver performance parameters," *Transp. Res. Rec., J. Transp. Res. Board*, no. 1818, pp. 12–24, 2002.

APPENDIX

6.1 SOURCE CODE

```
package com.dts.btisqr.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;
import com.dts.btisqr.model.Bug;
import com.dts.core.dao.AbstractDataAccessObject;
import com.dts.core.util.CoreHash;
import com.dts.core.util.DateWrapper;
import com.dts.core.util.LoggerManager;
public class BugDAO extends AbstractDataAccessObject {
    Connection con;
    public void addPriority(Bug bug)
    {
        try
        {
            int priorityid = getSequenceID("PRIORITY", "priorityid");
            con = getConnection();
            String name = bug.getPriorityName();
            PreparedStatement pst = con.prepareStatement("insert
into priority values(?,?)");
            pst.setInt(1, priorityid);
```

```

        pst.setString(2, name);
        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}

public void updatePriority(Bug bug)
{
    try
    {
        con = getConnection();
        int priorityid = bug.getPriorityID();
        String name = bug.getPriorityName();
        PreparedStatement pst = con.prepareStatement("update

```

```

priority set prioritytype=? where priorityid=?");
        pst.setInt(2, priorityid);
        pst.setString(1, name);
        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}

public void deletePriority(int priorityid)
{
    try
    {
        con = getConnection();
    }
}

```

```

        PreparedStatement pst = con.prepareStatement("delete
from priority where priorityid=?");
        pst.setInt(1, priorityid);
        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}

public void deleteBug(int bugid)
{
    try
    {
        con = getConnection();
        PreparedStatement pst = con.prepareStatement("delete

```

```
from bugs where bugid=?");
```

```
        pst.setInt(1, bugid);

        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}
```

```
public void addBug(Bug bug)
{
```

```

try
{
    int bugid = getSequenceID("BUGS", "bugid");
    con = getConnection();
    String name = bug.getBugName();
    int projectid = bug.getProjectID();
    String raiseddate = bug.getRaisedDate();
    String raisedby = bug.getRaisedBy();
    int priorityid = bug.getPriorityID();
    String desc = bug.getDescription();
    String status = bug.getStatus();
    PreparedStatement pst = con.prepareStatement("insert into
bugs values(?,?,?,?,?,?,?,?)");
    pst.setInt(1, bugid);
    pst.setString(2, name);
    pst.setInt(3, projectid);
    pst.setString(4, DateWrapper.parseDate(new Date()));
    pst.setString(5, raisedby);
    pst.setInt(6, priorityid);
    pst.setString(7, desc);
    pst.setString(8, status);
    pst.executeUpdate();
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}

```

```

        finally
        {
            try
            {
                con.close();
            }
            catch(Exception e)
            {
                LogManager.writeLogWarning(e);
            }
        }
    }

    public void updateBug(Bug bug)
    {
        try
        {
            con = getConnection();
            int bugid = bug.getBugID();
            String name = bug.getBugName();
            int projectid = bug.getPriorityID();
            String raiseddate = bug.getRaisedDate();
            String raisedby = bug.getRaisedBy();
            int priorityid = bug.getPriorityID();
            String desc = bug.getDescription();
            String status = bug.getStatus();

            PreparedStatement pst = con.prepareStatement("update

```

```

bugs set bugname=?,projectid=?, " +
    "raiseddate=?,raisedby=?,priority=?,description=?, " +
    status=? where bugid=?");
    pst.setString(1, name);
    pst.setInt(2, projectid);
    pst.setString(3, raiseddate);
    pst.setString(4, raisedby);
    pst.setInt(5, priorityid);
    pst.setString(6, desc);
    pst.setString(7, status);
    pst.setInt(8, bugid);

    pst.executeUpdate();
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {

```

```

        LogManager.writeLogWarning(e);
    }
}

```

```

public Bug getBugDeatails(int bugid)
{
    Bug bug = null;
    try
    {
        con = getConnection();

        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from bugs where
bugid="+bugid);
        if(rs.next())
        {
            bug = new Bug();
            bug.setBugID(rs.getInt(1));
            bug.setBugName(rs.getString(2));
            bug.setProjectID(rs.getInt(3));
            bug.setRaisedDate1(rs.getDate(4));
            bug.setRaisedBy(rs.getString(5));
            bug.setPriorityID(rs.getInt(6));
            bug.setDescription(rs.getString(7));
            bug.setStatus(rs.getString(8));

```

```

        }
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return bug;
}

public CoreHash getBugs()
{
    CoreHash aCoreHash = new CoreHash();
    try
    {
        Bug bug = null;
        con = getConnection();
        int bugid = 0;

```

```

        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from bugs");

        while(rs.next())
        {
            bug = new Bug();
            bugid = rs.getInt(1);
            bug.setBugID(bugid);
            bug.setBugName(rs.getString(2));
            bug.setProjectID(rs.getInt(3));
            bug.setRaisedDate1(rs.getDate(4));
            bug.setRaisedBy(rs.getString(5));
            bug.setPriorityID(rs.getInt(6));
            bug.setDescription(rs.getString(7));
            bug.setStatus(rs.getString(8));
            aCoreHash.put(new Integer(bugid), bug);
        }
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();

```

```

    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
return aCoreHash;
}

public CoreHash getBugs(String bug,String priority,String project)
{
    CoreHash aCoreHash = new CoreHash();
    try
    {
        String query = "select * from bugs where";
        int count =0;
        if(!bug.equals("--Select--"))
        {
            query += " bugid="+bug;
            count = 1;
        }
        if(count==1 && !project.equals("--Select--"))
        {
            query += " and projectid="+project;
        }
        else if(!project.equals("--Select--"))
        {
            query += " projectid="+project;

```

```

        count = 1;
    }

    if(count==1 && !priority.equals("--Select--"))
    {
        query += " and priority="+priority;
    }
    else if(!priority.equals("--Select--"))
    {
        query += " priority="+priority;
        count = 1;
    }
    Bug bugs = null;
    con = getConnection();
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery(query);
    int bugid=0;
    while(rs.next())
    {
        bugs = new Bug();
        bugid = rs.getInt(1);
        bugs.setBugID(bugid);
        bugs.setBugName(rs.getString(2));
        bugs.setProjectID(rs.getInt(3));
        bugs.setRaisedDate1(rs.getDate(4));
        bugs.setRaisedBy(rs.getString(5));
        bugs.setPriorityID(rs.getInt(6));
    }

```

```

        bugs.setDescription(rs.getString(7));
        bugs.setStatus(rs.getString(8));

        aCoreHash.put(new Integer(bugid), bugs);
    }
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
return aCoreHash;
}

public CoreHash getBugs(String membername)
{
    CoreHash aCoreHash = new CoreHash();

```

```

try
{
    Bug bug = null;
    con = getConnection();
    int bugid = 0;
    Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select
b.bugid,b.bugname,b.projectid,b.Raiseddate,b.raisedby,b.priority,b.description,b.status from bugs b,bugassigned ba where b.bugid=ba.bugid and
ba.assignedto='"+membername+"'");
    while(rs.next())
    {
        bug = new Bug();
        bugid = rs.getInt(1);
        bug.setBugID(bugid);
        bug.setBugName(rs.getString(2));
        bug.setProjectID(rs.getInt(3));
        bug.setRaisedDate1(rs.getDate(4));
        bug.setRaisedBy(rs.getString(5));
        bug.setPriorityID(rs.getInt(6));
        bug.setDescription(rs.getString(7));
        bug.setStatus(rs.getString(8));
        aCoreHash.put(new Integer(bugid), bug);
    }
}
catch(Exception e)
{

```

```

        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return aCoreHash;
}

public CoreHash getPriorities()
{
    CoreHash aCoreHash = new CoreHash();
    try
    {
        Bug bug = null;
        con = getConnection();
        int priorityid = 0;
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from Priority");

        while(rs.next())

```

```

        {
            bug = new Bug();
            priorityid = rs.getInt(1);
            bug.setPriorityID(priorityid);
            bug.setPriorityName(rs.getString(2));

            aCoreHash.put(new Integer(priorityid), bug);
        }
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return aCoreHash;
}

public Bug getBugSolution(int bugid)

```

```

{
    Bug bug = null;
    try
    {
        con = getConnection();

        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select
b.bugname,b.projectid,b.raiseddate,b.raisedby,b.priority," +
"b.description,ba.assignedto,ba.assigneddate,s.solvedby,s.solveddate," +
"s.solution from bugs
b,bugassigned ba, solution s where " +
"b.bugid=ba.bugid
and b.bugid=s.bugid and b.bugid="+bugid);
        if(rs.next())
        {
            bug = new Bug();
            bug.setBugName(rs.getString(1));
            bug.setProjectID(rs.getInt(2));
            bug.setRaisedDate1(rs.getDate(3));
            bug.setRaisedBy(rs.getString(4));
            bug.setPriorityID(rs.getInt(5));
            bug.setDescription(rs.getString(6));
            bug.setAssignedTo(rs.getString(7));
            bug.setAssignedDate1(rs.getDate(8));
            bug.setSolvedBy(rs.getString(9));

```

```

        bug.setSolvedDate1(rs.getDate(10));
        bug.setSolution(rs.getString(11));
    }
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
return bug;
}

public void setBugSolution(Bug bug)
{
    int i = 0;
    try
    {
        con = getConnection();

```

```

        con.setAutoCommit(false);
        PreparedStatement pst = con.prepareStatement("insert
into solution values(?,?,?,?)");
        PreparedStatement pst1 = con.prepareStatement("update
bugs set status=? where bugid=?");
        pst.setInt(1, bug.getBugID());
        pst.setString(2, bug.getSolvedBy());
        pst.setString(3, bug.getSolvedDate());
        pst.setString(4, bug.getSolution())
        pst1.setString(1, "Completed");
pst1.setInt(2, bug.getBugID())
        if(pst.executeUpdate()>0)
        {
            if(pst1.executeUpdate()>0)
                i=1;
        }
    }
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        if(i==0)
            con.rollback();
    }
}

```

```

        else
            con.commit();
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}

public void assignBug(Bug aBug)
{
    int i=0;
    try
    {
        con = getConnection();
        con.setAutoCommit(false);
        PreparedStatement pst = con.prepareStatement("insert into
bugassigned values(?,?,?)");
        PreparedStatement pst1 = con.prepareStatement("update bugs set
status=? where bugid=?");
        pst.setInt(1, aBug.getBugID());
        pst.setString(2, aBug.getAssignedTo());
        pst.setString(3, DateWrapper.parseDate(new Date()));
        pst1.setString(1, "Assigned");
        pst1.setInt(2, aBug.getBugID());
        if(pst.executeUpdate()>0)

```

```

{
    if(pst1.executeUpdate(>0)
    {
        i=1;
    }
}
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
    i=0;
}
finally
{
    try
    {
        if(i==0)
            con.rollback();
        else
            con.commit();
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}

```

```

    }
}

```

MEMBERSDAO:

```

package com.dts.btisqr.dao;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import com.dts.core.dao.AbstractDataAccessObject;
import com.dts.core.util.CoreHash;
import com.dts.core.util.LoggerManager;
import com.dts.dae.model.Profile;
public class MembersDAO extends AbstractDataAccessObject
{
    Connection con;
    public CoreHash getProfiles(String role)
    {
        CoreHash aCoreHash = new CoreHash();
        try
        {
            String loginname = "";
            Profile profile = null;
            con = getConnection();
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("select
            Id.loginname,Id.firstname,Id.lastname,lp.city,lp.ContactNo,lp.birthdate,lp.hn
            o,lp.street,lp.state,lp.country,lp.pincod,lp.email,Id.loginstatus      from
            logindetails  Id,loginprofile  lp  where  Id.loginname=lp.loginid  and

```

```

ld.logintype="" + role + "");
        while(rs.next())
        {
            loginname = rs.getString(1);
            profile = new Profile();
            profile.setLoginID(loginname);
            profile.setFirstName(rs.getString(2));
            profile.setLastName(rs.getString(3));
            profile.setCity(rs.getString(4));
            profile.setPhoneNo(rs.getString(5));
            profile.setBirthDate1(rs.getDate(6));
            profile.setHno(rs.getString(7));
            profile.setStreet(rs.getString(8));
            profile.setState(rs.getString(9));
            profile.setCountry(rs.getString(10));
            profile.setPincode(rs.getString(11));
            profile.setEmail(rs.getString(12));
            profile.setStatus(rs.getInt(13));
            aCoreHash.put(loginname,profile);
        }
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    return aCoreHash;
}

```

```

public CoreHash getProfiles()
{
    CoreHash aCoreHash = new CoreHash();
    try
    {
        String loginname = "";
        Profile profile = null;
        con = getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select
ld.loginname,ld.firstname,ld.lastname,lp.city,lp.ContactNo," +
    "lp.birthdate,lp.hno,lp.street,lp.state,lp.country,lp.pincod,lp.email," +
    "ld.loginstatus,ld.logintype from logindetails ld,loginprofile lp
where ld.loginname=lp.loginid");
        while(rs.next())
        {
            loginname = rs.getString(1);
            profile = new Profile();
            profile.setLoginID(loginname);
            profile.setFirstName(rs.getString(2));
            profile.setLastName(rs.getString(3));
            profile.setCity(rs.getString(4));
            profile.setPhoneNo(rs.getString(5));
            profile.setBirthDate1(rs.getDate(6));
            profile.setHno(rs.getString(7));
            profile.setStreet(rs.getString(8));
            profile.setState(rs.getString(9));

```

```

        profile.setCountry(rs.getString(10));
        profile.setPincode(rs.getString(11));
        profile.setEmail(rs.getString(12));
        profile.setStatus(rs.getInt(13));
        profile.setRole(rs.getString(14));
        aCoreHash.put(loginname,profile);
    }
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
return aCoreHash;
}

public CoreHash getProfiles(String developer,String tester)
{
    CoreHash aCoreHash = new CoreHash();
    try
    {
        String loginname = "";
        Profile profile = null;
        con = getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select
ld.loginname,ld.firstname,ld.lastname,lp.city,lp.ContactNo," +

        "lp.birthdate,lp.hno,lp.street,lp.state,lp.country,lp.pincode,lp.email," +

```

```

        "ld.loginstatus,ld.logintype from logindetails ld,loginprofile lp
where ld.loginname=lp.loginid" +
        " and (ld.logintype='"+developer+"' or
ld.logintype='"+tester+"");
        while(rs.next())
        {
                loginname = rs.getString(1);
        profile = new Profile();
        profile.setLoginID(loginname);
        profile.setFirstName(rs.getString(2));
        profile.setLastName(rs.getString(3));
        profile.setCity(rs.getString(4));
        profile.setPhoneNo(rs.getString(5));
        profile.setBirthDate1(rs.getDate(6));
        profile.setHno(rs.getString(7));
        profile.setStreet(rs.getString(8));
        profile.setState(rs.getString(9));
        profile.setCountry(rs.getString(10));
        profile.setPincode(rs.getString(11));
        profile.setEmail(rs.getString(12));
        profile.setStatus(rs.getInt(13));
        profile.setRole(rs.getString(14));
        aCoreHash.put(loginname,profile);
        }
    }
    catch(Exception e)
    {

```

```

        LogManager.writeLogWarning(e);
    }
    return aCoreHash;
}
}

```

PROJECTSDAO

```

package com.dts.btisqr.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import com.dts.btisqr.model.Project;
import com.dts.core.dao.AbstractDataAccessObject;
import com.dts.core.util.CoreHash;
import com.dts.core.util.LogManager;
public class ProjectsDAO extends AbstractDataAccessObject
{
    Connection con;
    //add new Project
    public void addProject(Project aProject)
    {
        boolean flag = true;
        try
        {
            int projectid = getSequenceID("Project" , "Projectid");
            con = getConnection();
            PreparedStatement pst = con.prepareStatement("insert into project

```

```

values(?,?,?,?,?,?)");
    pst.setInt(1, projectid);
    pst.setString(2, aProject.getProjectName());
    pst.setString(3, aProject.getStartDate());
    pst.setString(4, aProject.getEndDate());
    pst.setString(5, aProject.getStatus());
    pst.setString(6, aProject.getAssignedTo());
    pst.executeUpdate();
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
}

//Update Project
public void updateProject(Project aProject)

```

```

{
    boolean flag = true;
    try
    {
        con = getConnection();
        PreparedStatement pst = con.prepareStatement("update project set
projectname=?,startdate=?,enddate=?,status=?,Assignedto=?      where
projectid=?");

        pst.setString(1, aProject.getProjectName());
        pst.setString(2, aProject.getStartDate());
        pst.setString(3, aProject.getEndDate());
        pst.setString(4, aProject.getStatus());
        pst.setString(5, aProject.getAssignedTo());
        pst.setInt(6, aProject.getProjectID());
        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
    }
}

```

```

        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}

//get All Projects
public CoreHash getAllProject()
{
    boolean flag = true;
    CoreHash aCoreHash = new CoreHash();
    try
    {
        con = getConnection();
        int projectid = 0;
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from project");
        Project aProject = null;
        while(rs.next())
        {
            aProject = new Project();
            projectid = rs.getInt(1);
            aProject.setProjectID(projectid);
            aProject.setProjectName(rs.getString(2));
            aProject.setStartDate1(rs.getDate(3));
            aProject.setEndDate1(rs.getDate(4));
            aProject.setStatus(rs.getString(5));
        }
    }
}

```

```

        aProject.setAssignedTo(rs.getString(6));
        aCoreHash.put(projectid, aProject);
    }
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
return aCoreHash;
}

//get Assigned Projects
public CoreHash getManagerProjects(String assignedto)
{
    boolean flag = true;
    CoreHash aCoreHash = new CoreHash();
    try

```

```

{
    con = getConnection();
int projectid = 0;
Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("select * from project where
assignedto='"+assignedto+"'");
    Project aProject = null;
while(rs.next())
{
    aProject = new Project();
    projectid = rs.getInt(1);
    aProject.setProjectID(projectid);
    aProject.setProjectName(rs.getString(2));
    aProject.setStartDate1(rs.getDate(3));
    aProject.setEndDate1(rs.getDate(4));
    aProject.setStatus(rs.getString(5));
    aProject.setAssignedTo(rs.getString(6));
    aCoreHash.put(projectid, aProject);
}
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try

```

```

        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return aCoreHash;
}

//get All Projects
public Project getProjectDetails(int projectid)
{
    Project aProject = new Project();
    try
    {
        con = getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from project
projectid="+projectid);
        if(rs.next())
        {
            aProject = new Project();
            projectid = rs.getInt(1);
            aProject.setProjectID(projectid);
            aProject.setProjectName(rs.getString(2));
            aProject.setStartDate1(rs.getDate(3));

```

```

        aProject.setEndDate1(rs.getDate(4));
        aProject.setStatus(rs.getString(5));
        aProject.setAssignedTo(rs.getString(6));
    }
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}
return aProject;
}

//get All Projects
public Project deleteProject(int projectid)
{
    Project aProject = new Project();
    try

```

```

        {
            con = getConnection();
            Statement st = con.createStatement();
            st.executeUpdate("delete from project where
projectid="+projectid);
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
        finally
        {
            try
            {
                con.close();
            }
            catch(Exception e)
            {
                LogManager.writeLogWarning(e);
            }
        }
        return aProject;
    }
    //Assign Project to Member
    public void assignProject(Project aProject)
    {
        try

```

```

    {
        con = getConnection();
        PreparedStatement pst = con.prepareStatement("insert into
projectmembers values(?,?,?)");
        pst.setInt(1, aProject.getProjectID());
        pst.setString(2, aProject.getAssignedTo());
        pst.setInt(3, 1);

        pst.executeUpdate();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
}

```

//View Project Memebers

```

public CoreHash getProjectMembers(String manager)
{
    boolean flag = true;
    System.out.println(manager+"manager");
    CoreHash aCoreHash = new CoreHash();
    try
    {
        con = getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select
p.projectid,p.projectname,pm.member,ld.logintype " +
                                     "from logindetails ld,
projectmembers pm, project p " +
                                     "where
p.projectid=pm.projectid and pm.member=ld.loginname " +
                                     " and
p.Assignedto='"+manager+"'")
        Project aProject = null;
        int i = 0;
        while(rs.next())
        {
            aProject = new Project();
            aProject.setProjectID(rs.getInt(1));
            aProject.setProjectName(rs.getString(2));
            aProject.setMemberName(rs.getString(3));
            aProject.setRole(rs.getString(4));
            aCoreHash.put(new Integer(i++), aProject);

```

```

    }
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return aCoreHash;
}

//get Free Members
public CoreHash getFreeMemembers(String role)
{
    boolean flag = true;
    CoreHash aCoreHash = new CoreHash();
    try
    {
        con = getConnection();

```

```

Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select loginname from logindetails Id
where logintype='"+role+"'");
Project aProject = null;
int i = 0;
while(rs.next())
{
    aProject = new Project();
    aProject.setMemberName(rs.getString(1));
    aCoreHash.put(new Integer(i++), aProject);
}
}
catch(Exception e)
{
    LogManager.writeLogWarning(e);
}
finally
{
    try
    {
        con.close();
    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
}

```

```

        return aCoreHash;
    }
    //get Project by Loginname
    public CoreHash getProject(String loginname)
    {
        //Project aProject = new Project();
        CoreHash aCoreHash = new CoreHash();
        int i = 0;
        try
        {
            con = getConnection();
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("select pm.projectid,
p.projectname from projectmembers pm, project p " +
                                           " where
p.projectid=pm.projectid and pm.member='"+loginname+"' and
pm.status=1");
            while(rs.next())
            {
                Project aProject = new Project();
                int abc=rs.getInt(1);
                aProject.setProjectID(rs.getInt(1));
                System.out.println("abbbb"+rs.getInt(1));
                aProject.setProjectName(rs.getString(2));
                System.out.println("ccccc"+rs.getString(2));
                aCoreHash.put(new Integer(abc), aProject);
            }
        }
    }

```

```

    }
    catch(Exception e)
    {
        LogManager.writeLogWarning(e);
    }
    finally
    {
        try
        {
            con.close();
        }
        catch(Exception e)
        {
            LogManager.writeLogWarning(e);
        }
    }
    return aCoreHash;
}
}

```

PROJECT MODEL:

```

package com.dts.btisqr.model;
import java.sql.Date;
public class Project
{
    private int projectID;
    private String projectName;
    private String startDate;

```

```

private String endDate;
private Date startDate1;
private Date endDate1;
private String status;
private String assignedTo;
private String memberName;
private String role;
public String getMemberName() {
    return memberName;
}
public void setMemberName(String memberName) {
    this.memberName = memberName;
}
public String getRole() {
    return role;
}
public void setRole(String role) {
    this.role = role;
}
public int getProjectID() {
    return projectID;
}
public void setProjectID(int projectID) {
    this.projectID = projectID;
}
public String getProjectName() {
    return projectName;
}

```

```

}
public void setProjectName(String projectName) {
    this.projectName = projectName;
}
public String getStartDate() {
    return startDate;
}
public void setStartDate(String startDate) {
    this.startDate = startDate;
}
public String getEndDate() {
    return endDate;
}
public void setEndDate(String endDate) {
    this.endDate = endDate;
}
public Date getStartDate1() {
    return startDate1;
}
public void setStartDate1(Date startDate1) {
    this.startDate1 = startDate1;
}
public Date getEndDate1() {
    return endDate1;
}
public void setEndDate1(Date endDate1) {
    this.endDate1 = endDate1;
}

```

```
}  
public String getStatus() {  
    return status;  
}  
public void setStatus(String status) {  
    this.status = status;  
}  
public String getAssignedTo() {  
    return assignedTo;  
}  
public void setAssignedTo(String assignedTo) {  
    this.assignedTo = assignedTo;  
}  
}
```