Defect Tracking System is used to track the work performed to resolve a defect that also gives the facility to define the tasks in the organization and also allows the managers to track the Defects & time spent by each employee for that particular task.

Defect Tracking System

Date-05/09/2012

Abhinav Singh, Gaurav Soni & Prateek Rajan

Abstract

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

This project aims at creation of a Defect 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 assignment of the task is done to an employee.

• Employee does Defects 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

Table of Contents:

Table of Figures

Chapter 1: Introduction

1.1 Purpose & Scope

The purpose of Defect Tracking for improving software reliability is to provide better service to the administrator or useful for applications developed in an organization. The "Defect Tracking for Improving Software Reliability" is a web based application that can be accessed throughout the organization. This system can be used for logging Defects or Bugs against an application/module, assigning them to team members and tracking them for resolution. There are features like email notifications, user maintenance, user access control, report generators etc. in this system.

1.2 Development Approach

For this project our strategy is to follow Spiral Model of SDLC. We found this model appropriate for this software application because it is enterprise level software and has significant size which would require reviews of developed prototype. This will help the customer to find the risks and abort the project if risks are deemed too great. This is relatively very efficient & effective way of dealing with software of such magnitude.

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 and 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 involve 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.

1.3 Overview

In this document we are providing detailed description of current system and proposed system. All functional and non-functional requirements are mentioned in chapter 3. Detailed UML diagrams like use cases, class diagrams, sequence diagrams have been created with all the details of the proposed system architecture. Explanation of each class, components, modules, sub-systems is provided with all the content required to understand the design framework, and development process.

Testing is done on different modules and results are shown in chapter 6. Coding standard, Glossary, definitions, Graphical user interface snapshots are provided in the Appendix.

Chapter 2: Current System

The current system is mostly concerned with the storing defects onto the file system with little to no traceability, making the tracking of the defects difficult at a later time. Most of the work of inserting the defects and tracking them back at some later point of time requires human intervention and is done manually. This makes the system limited and hence results in degraded performance.

- Information retrieval is a very big process & it becomes very difficult to handle huge databases manually with same efficiency & at the same time with the increase in the database the time to retrieve the concerned information also increases manifolds.
- Lack of organization of the files makes it prone to information loss due to accidental deletion of files.
- No security because the files are visible to the users. More over every user has the same level of access to the files.
- Report generation is a big task and precision is as much important as output is.
- Most of the work is done by humans with minimum to no intervention by machines. Humans are subjected to other factors like stress, emotions etc. that may reduce their work efficiency which is not the case with the machines, hence prolonged & maintained efficiency

Chapter 3: Proposed System

We are proposing a Defect Tracking System that will help the companies in tracking the raised defects in the software projects Defect tracking is the process of reporting and tracking the progress of Defects from discovery through to resolution, where a Defect is defined as a deviation from requirements. Other terminology frequently used to describe this process includes:

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

Defect 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 Defects, but extended to track feature requests or enhancements as well as enquiries.

Advantages of the proposed system are:

- ✓ Efficient centralized database schema
- ✓ Increased security with access only to authorized personnel.
- ✓ Quick report generation
- ✓ Easy to update the records and track the defects

3.1 Functional Requirements:

FR-1 Administrator shall be able to Login to the system.

FR-2 The system shall allow administrator to add new design department.

FR-3 The system shall allow administrator to Add/ Edit new defects.

FR-4 The system shall allow administrator to Add/ Edit priority to the defects.

FR-5 The system shall allow administrator to add new projects to the system.

FR-6 The system shall allow administrator to add new modules to the existing projects.

FR-7 The system shall allow administrator to generate reports corresponding to the status of each defect i.e. is it under process, completed or pending?

FR-8 The system shall allow administrator to add new employee or update existing employee's status in the system.

FR-9 The system shall allow administrator to change/update the status of the defects.

FR-10 The system shall allow administrator to assign the bugs to a particular employee new defects.

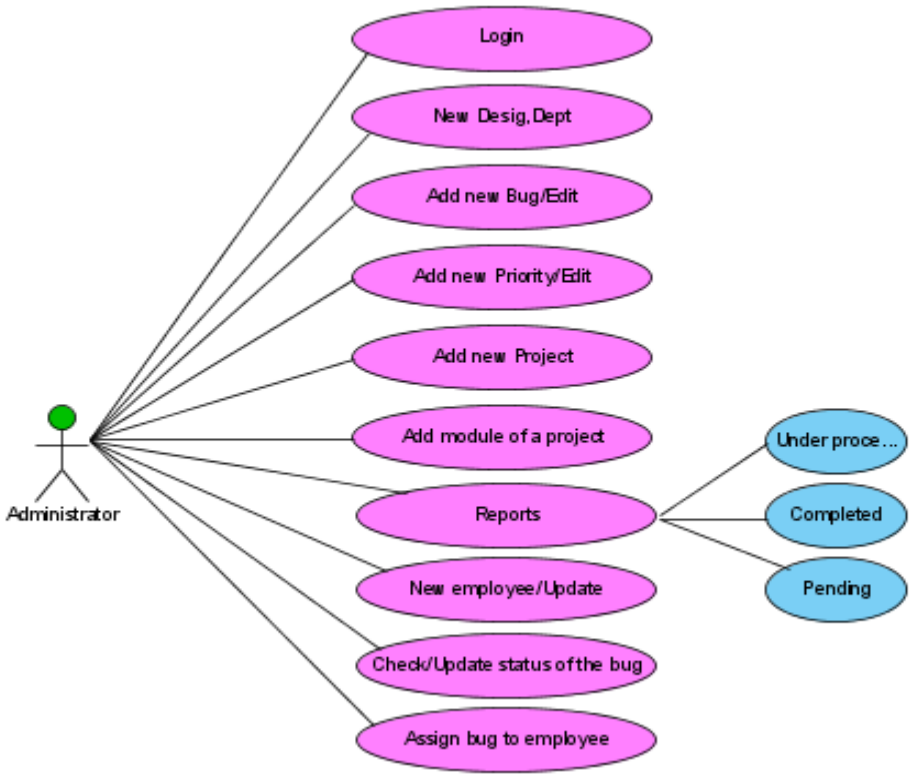


Figure 1: UseCase for Admin

FR-11 The system shall allow the tester to login to the system.

FR-12 The system shall allow the tester to post new bugs in the system.

FR-13 The system shall allow the tester to check the status of the existing bug in the system.

FR-14 The system shall allow the tester to view the information related to each bug in the system.

FR-15 The system shall allow the tester to view the priority assigned to each bug by the administrator.

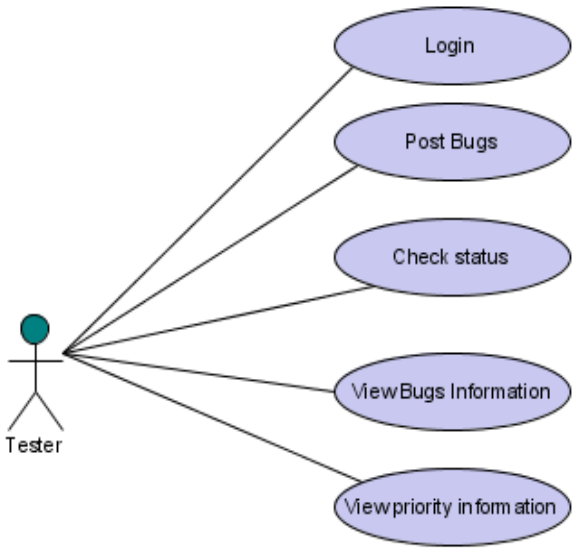


Figure 2: UseCase for Tester

FR-16 The system shall allow the developer to login to the system.

FR-17 The system shall allow the developer to view the bugs assigned to him.

FR-18 The system shall assist the developer to resolve the bugs.

FR-19 The system shall allow the developer to view the priorities assigned to each bug.

FR-20 The system shall allow the developer to view the bug reports.

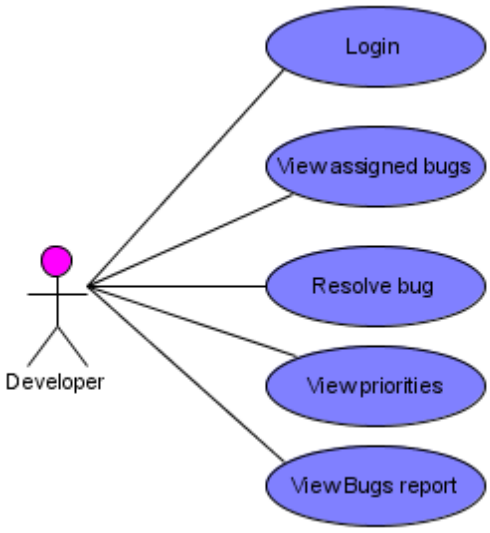


Figure 3: UseCase for Developer

3.1a Non-functional requirements:

NFR-1 The system shall be able to submit/search or any other activities done through the system in less than 5 seconds.

NFR-2 The system shall allow the users to navigate between pages in less than 2 to 3 seconds.

NFR-3 The administrator, manager, developer and tester shall be able to generate error free report within a maximum of 45 seconds (irrespective of size of data).

NFR-4 The system shall use Oracle database engine to run queries.

3.2 Use Case Diagrams

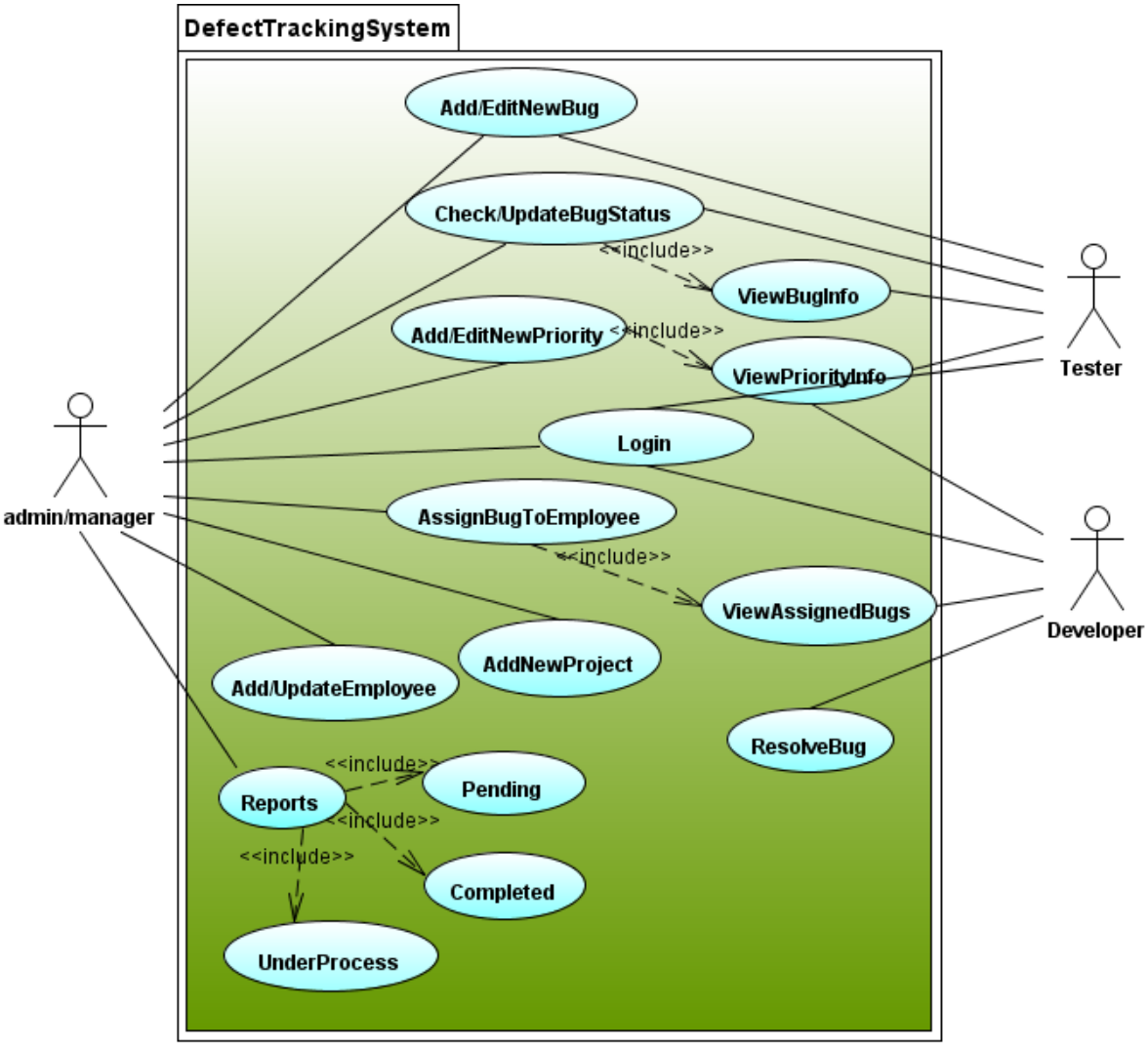


Figure 4: Use case Congregated

Admin: This module has the entire access to all other modules, admin creates the project and assigning the projects created to the manager, adding members to the project, assigning **defects** based on the priority. It can update the manager, members and access to the particular project data. Generating reports based on the managers' report submission.

Manager: This module has all administrative features to access once role is assigned by an administrator.

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 or Defects assigned by the manager, 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: Admin or Manager can access this module and generate the reports based on the requirements.

Chapter 4: Architecture

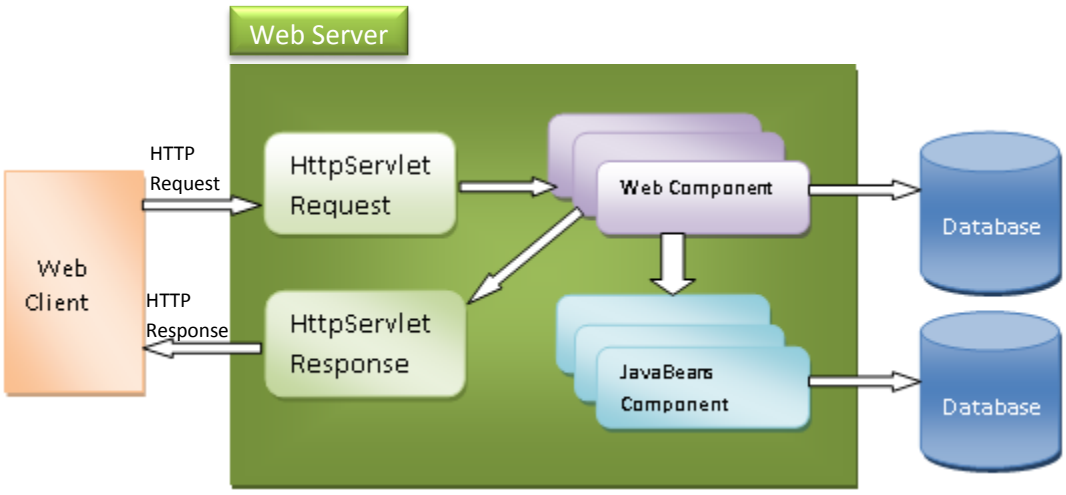


Figure 5: Architecture

There are a number of layers that work in collaboration to create an environment to get things done. Following is a brief introduction of the layers that are working in our project.

1. **THE PRESENTATION LAYER**

   Also known as the client layer, this layer is dedicated to present the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2. **THE BUSINESS RULES LAYER**

   Encapsulation of the Business rules or the business logic is done at this layer. Advantage of this layer is that any changes in Business Rules can be easily handled, also if the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the other. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process.

3. **THE DATA ACCESS LAYER**

This layer helps 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.

4. **THE DATABASE LAYER**

Database Components such as DB Files, Tables, Views, etc. is part of this layer. The database can be created using SQL Server, Oracle, Flat files, 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.

4.1 Overview:

Design Goals of the project are made to optimize the performance of the product. Developers should optimize the code processing and functionalities in the software project. For our defect tracking system (DTS) we have decided to meet certain design goals described below:

- Search or submission request or any other activities done through the system in must be accomplished in less than 5 seconds.
- Menu/page navigation must not take more than 2-3 seconds.
- Reports must be generated in less than 45 seconds
- Oracle database engine will be used to run queries

Architectural Diagram including major subsystems:

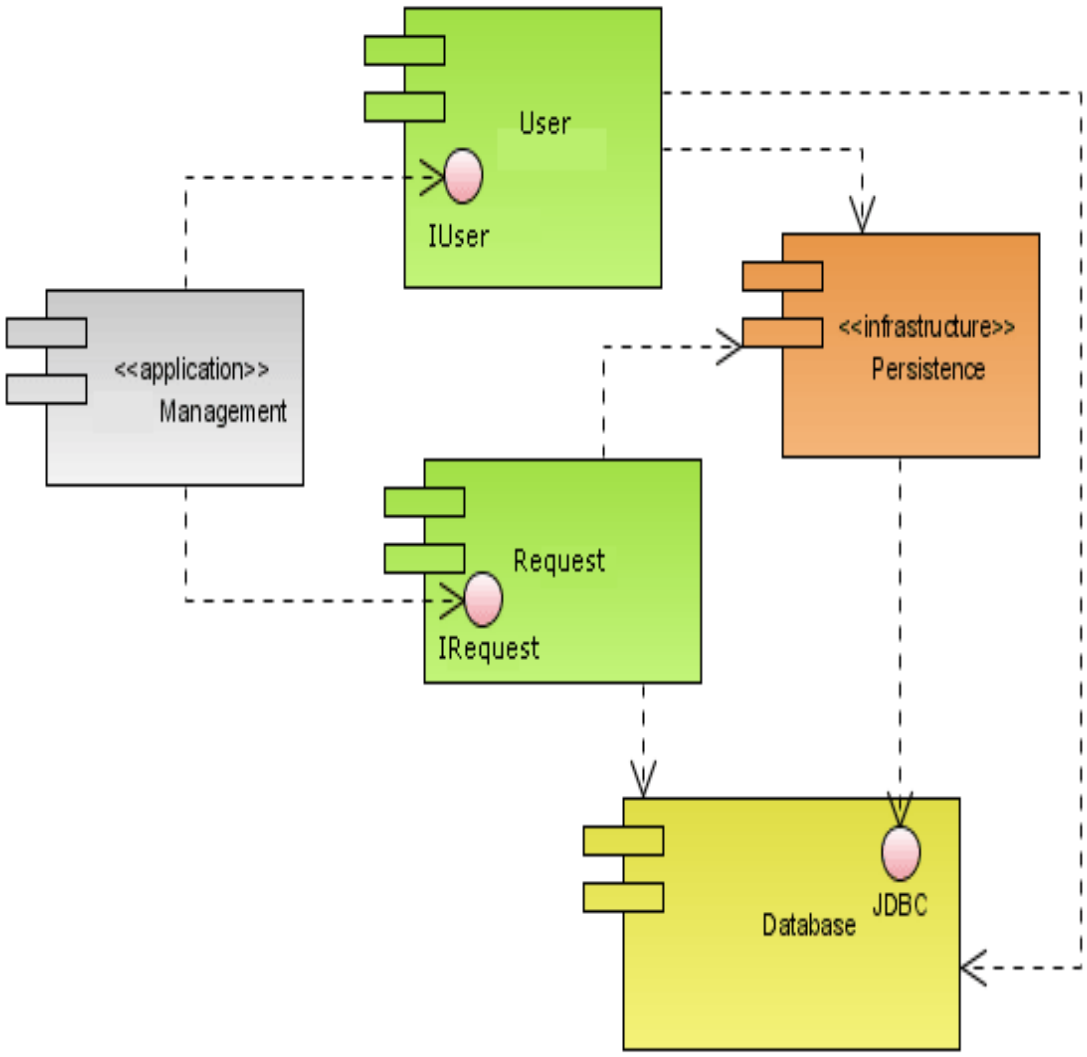


Figure 6: Subsystem Architecture

User Component:

In DTS, our team has defined 4 categories of users i.e. Administrator, Manager, Tester, Developer. Different users have different level of authorization to access data from the system. These users use application's user interface to interact with the system and database.

Management/Administration Component:

System access authorization or we can say Role assignment to use the system to the users is managed by Management component. Only administrator is allowed to access the management component/subsystem to assign managers, testers and developers their tasks and role to access the system features.

Database Component:

Users' data, Login details, Bug details, Priority list, Solution details are saved in different tables that are connected with the application. Users can retrieve data from the application using the database components that provide the connectivity to the database (Oracle), firing queries, viewing tables etc. on the basis of user authorization.

System Request Component:

Any requests made to the system ranging from page navigation, searching for bugs, looking for team members, finding assigned bugs to a team member (tester, developer or manager) to submit a form, all are managed or maintained by the System Request component. Basically, this component is responsible for HTTP requests made during client and server side interactions.

This architectural style is very helpful to meet our design goals because all these components contribute to provide great modularity to the system. All our modules are divided in different subsystem categories as user related information in User Component, Administration related modules comes user Management component, bugs related information like, IDs, issues, solutions, priorities are assigned under Database Component and any interaction made to the system by the user (requests made to the system) is part of System Component.

All these components play important role in our DTS (Defect Tracking System) application that gives flexibility in code development and makes the system robust as proper modularity is provided using these subsystems/components written above.

4.2 Subsystem Decomposition

1. **User Component:** This component provides the application interface to make requests to the database/servers/web-components.

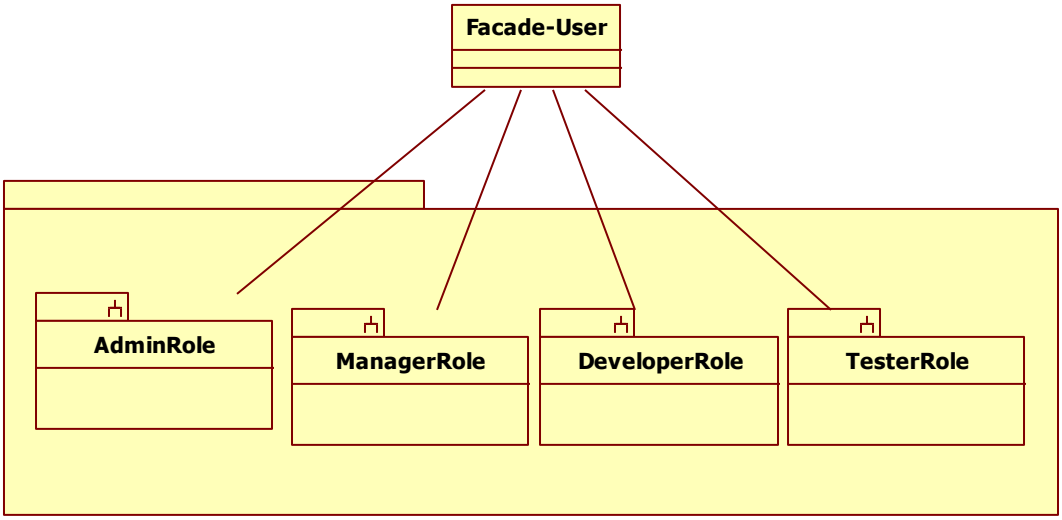


Figure 7: FacadeUser

Services provided by each subsystem:

Admin: All role properties are assigned to a user using this subsystem to have access to all the components, sub-components and modules available in the application.

Manager: This subsystem is serving to assign a manager role to access the data related to the projects once assigned to a manager user.
Developer: This subsystem is to provide access to the user who can view the bugs, priority level (but cannot edit them), work on the piece of code and mark the bug issue as completed or under-process etc.

Tester: This subsystem serves as role provider to tester who can raise bugs, edit bugs, report bugs which can be assigned to the developer by managers later.

2. Management/Administration Component:

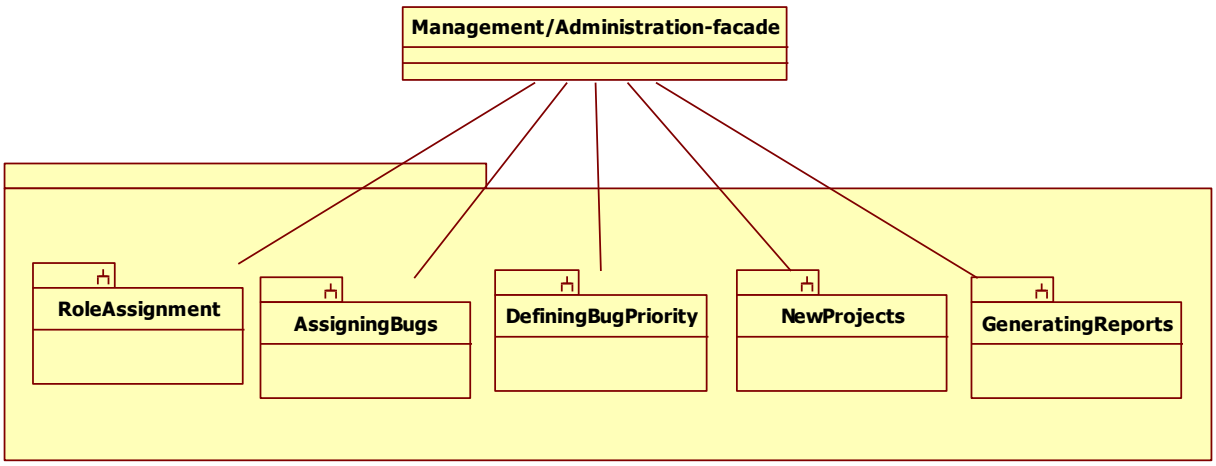


Figure 8: Facade-Management

Role Assignment: This subsystem is used to assign the roles to the users. Roles definitions are defined under User Component which will be used to give particular authorization to the user based on his role in the company.

AssigningBugs: Bugs can be raised by the tester working on a project. But which bug will be resolved by which developer is the task of management.

DefineBugPriority: Level of severity of bugs needs to be defined by management also. Developers act according to the priority level of the bug.

NewProject: Management look for new software projects in which the bugs/Defects needs to be detected and resolved. This subsystem let the management enter the details of new projects that can be assigned to different managers later.

Reports: Report generation is a very important aspect that is involved in out DTS project. Reports are used to generate data of the past work. A history of records can be pulled out using this subsystem. For Example: list of pending defects (with pending status), resolved defects (with completed status) and list of all the defects.

3. Database Component:

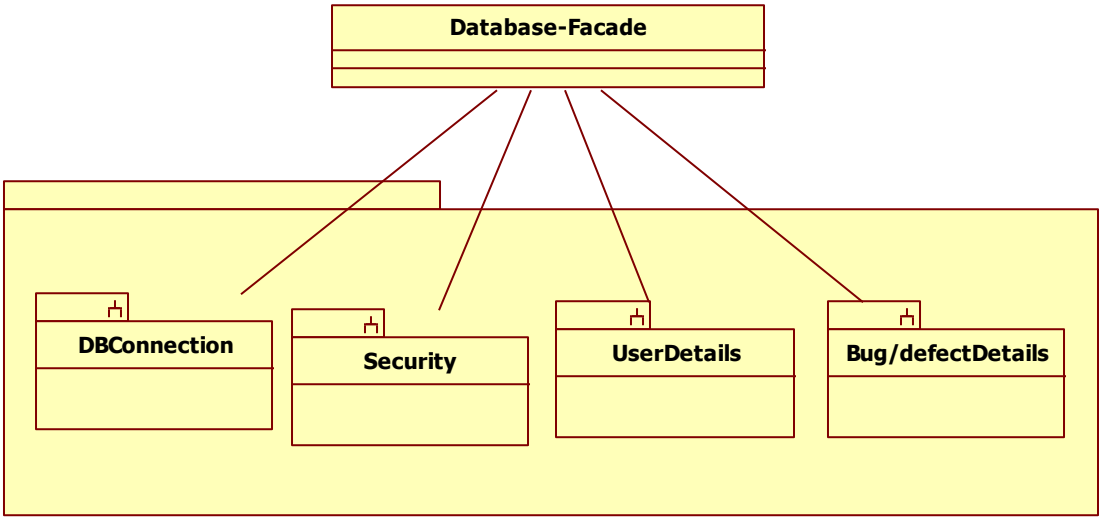


Figure 9: Facade-Database

Subsystems:

DBConnection: This is to provide the database connectivity among the application and database.

Security: Login details like username, passwords, fist and last name and other user related information required to implement security is placed here. Any changes made to username/Passwords will be stored here as soon as a user made these changes.

UserDetails: Users information, like role, email and contact information is served with this component

Bug/defects: This subsystem of database is to set and get the information of defects raised by a tester. Each bug/defect is linked to a particular project in the database**.**

4. System Request Component:

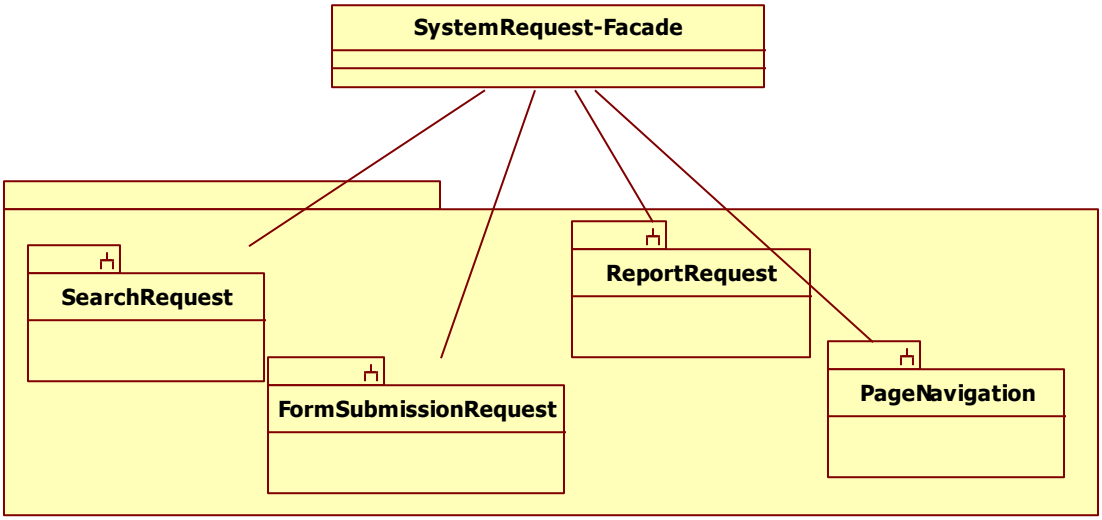


Figure 10: facade SystemRequest

SearchRequest: This is to provide service to the user (Admin, manager, developer or tester) to search into the database for required information. For example: A project can be searched using project name, which will run a query into the database to pull out the information from various tables to list Project name, start date, End Date and status.

FormSubmissionRequest: There are forms to create a new user, new project, new bug etc. to make the request to the system.

ReportRequest: Reports can be requested from the application using this subsystem.

PageNavigation: Pages can be easily navigated using hyperlinks provided on Menu navigation, table navigation etc. This service is provided by PageNavigation subsystem.

4.3 Persistent Data Management

In our Defect tracking system, we are using Oracle 10g database for data storage. We are running oracle on local machine. Database connectivity is done by following the regular database connectivity steps. This database is stored on the local machine which is secured by a username and password. This database can only be accessed using Oracle iSQLPlus Workspace using the assigned user and password, thereby enforcing the data security.

Login Details of the users are saved in LOGINDETAILS Table. This data is important and must be secured from any unauthorized access. This data must be saved or backed up on separate hard disk drive.

Chapter 5: Entity Relationship Diagram for DTS database

This entity diagram provides the information of our database schema, relations among tables including primary keys and other important details required in an ER diagram.

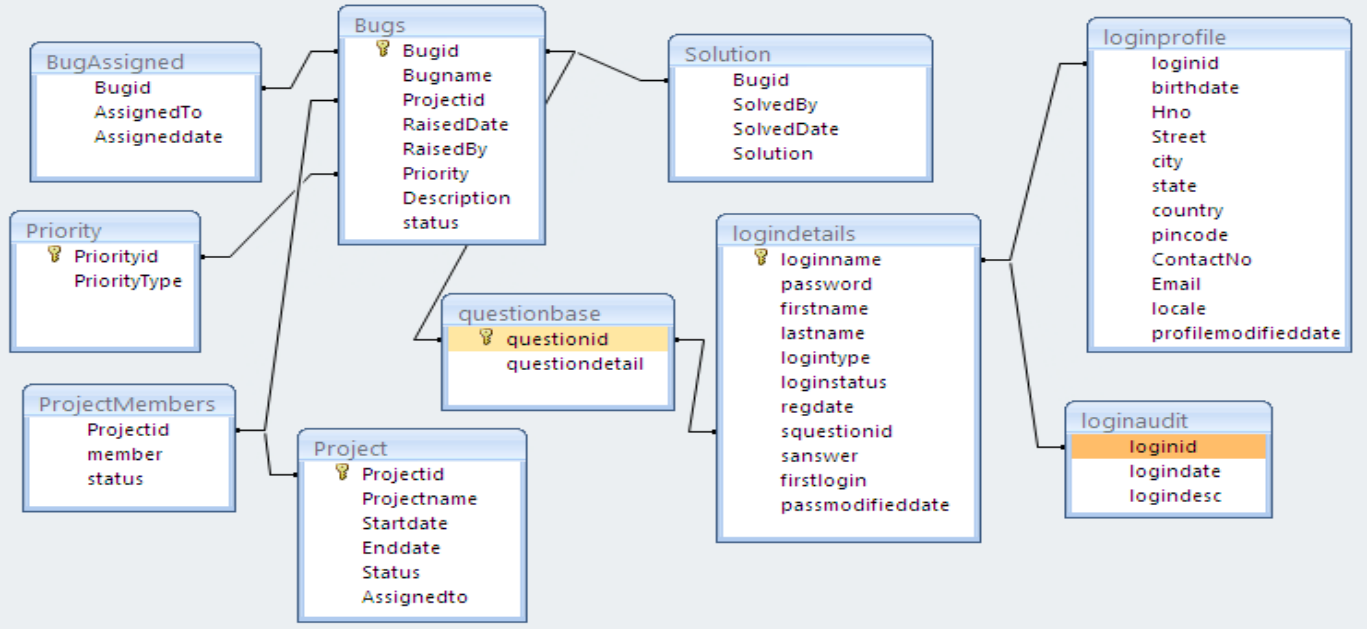


Figure 11: ER Diagram

Chapter 6: Object Design and Implementation

6.1 Static Model

This is the minimal class diagram for our DTS application without any detailed information about attributes or operations performed by classes

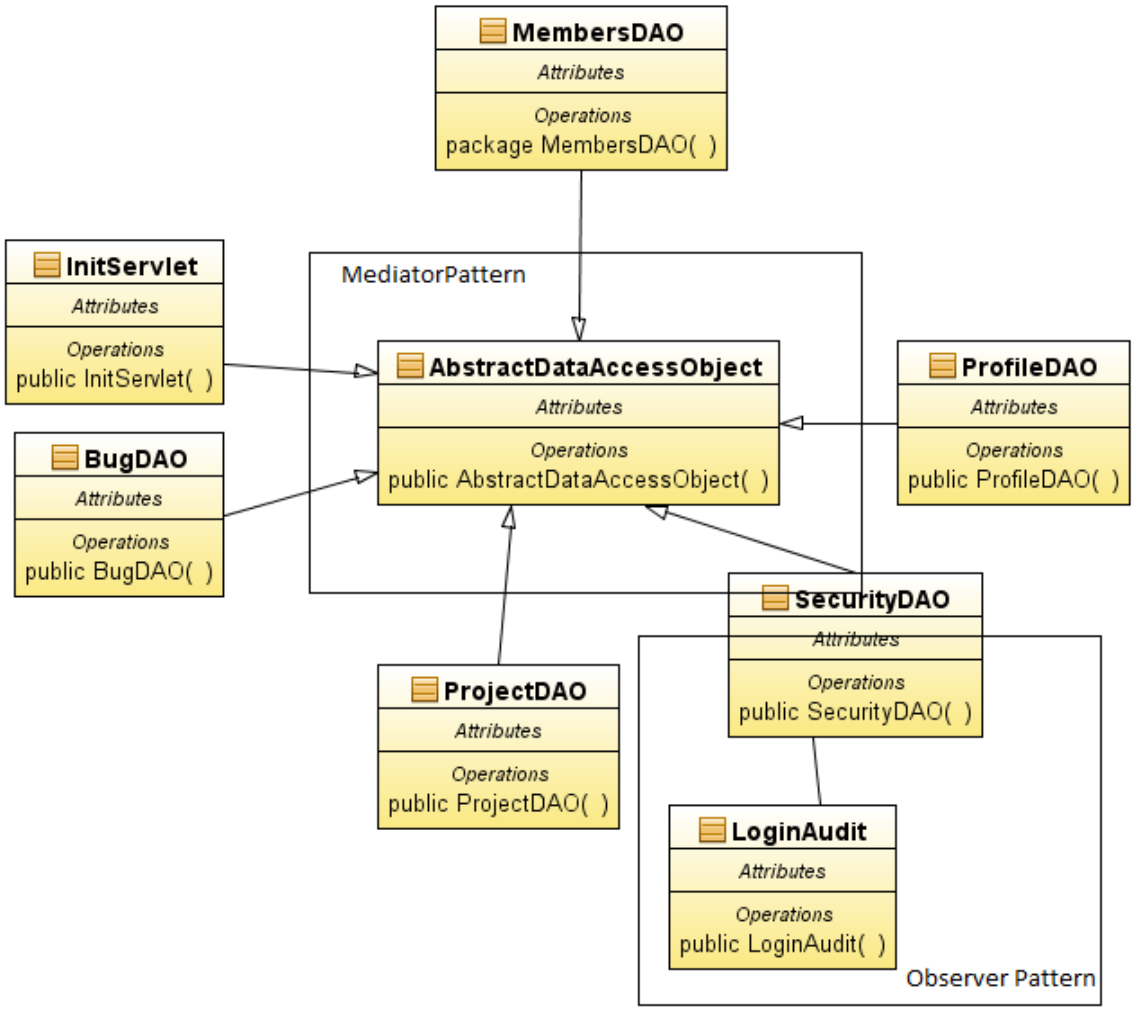


Figure 12: MinimalClassDiagram

MembersDataAccessObject:

Subsystem to which this class belongs: User

This class will take care of the different profiles of users. This class contains four roles of users-Admin, manager, tester and developer. Each user can view only those features of the application/system that are assigned to these profiles. This operation is implemented at the time of Login.

ProfileDataAccessObject:

Subsystem to which this class belongs: User

This class is used to register a user. Once a user is registered, it can create, modify or delete a profile of the user. A login ID will be assigned to each user with their registration status.

AbstractDataAccessobject:

Subsystem to which this class belongs: SystemRequest

This class is used to provide an interface in between database and system request. This class keeps the two different parts of the application isolated from each other. Any changes made to any part – database or the system itself will not affect each other.

SecurityDataAccessObject:

Subsystem to which this class belongs: Database

This class comprises of all the login details such as password, loginname, logincheck, login audit, changing password, change question, password recovery etc. This class is responsible to give authorization to the users by checking their username and password.

ProjectsDataAccessObject:

Subsystem to which this class belongs: Management

Projects DAS provide access to add a new project, update, assign manager to the project details and list of members involved in the project. This class can only be accessed by administrators with editing authority, other users are allowed for read-only view.

BugDataAccessObject:

Subsystem to which this class belongs: Database

The class is used to provide interface to add, edit or delete priorities, bugs; and set solutions, assign bugs to the users.

InitServlet:

Subsystem to which this class belongs: SystemRequest

This class comprises java init method that creates the instance of servlets just like constructor.

6.2 Dynamic Model

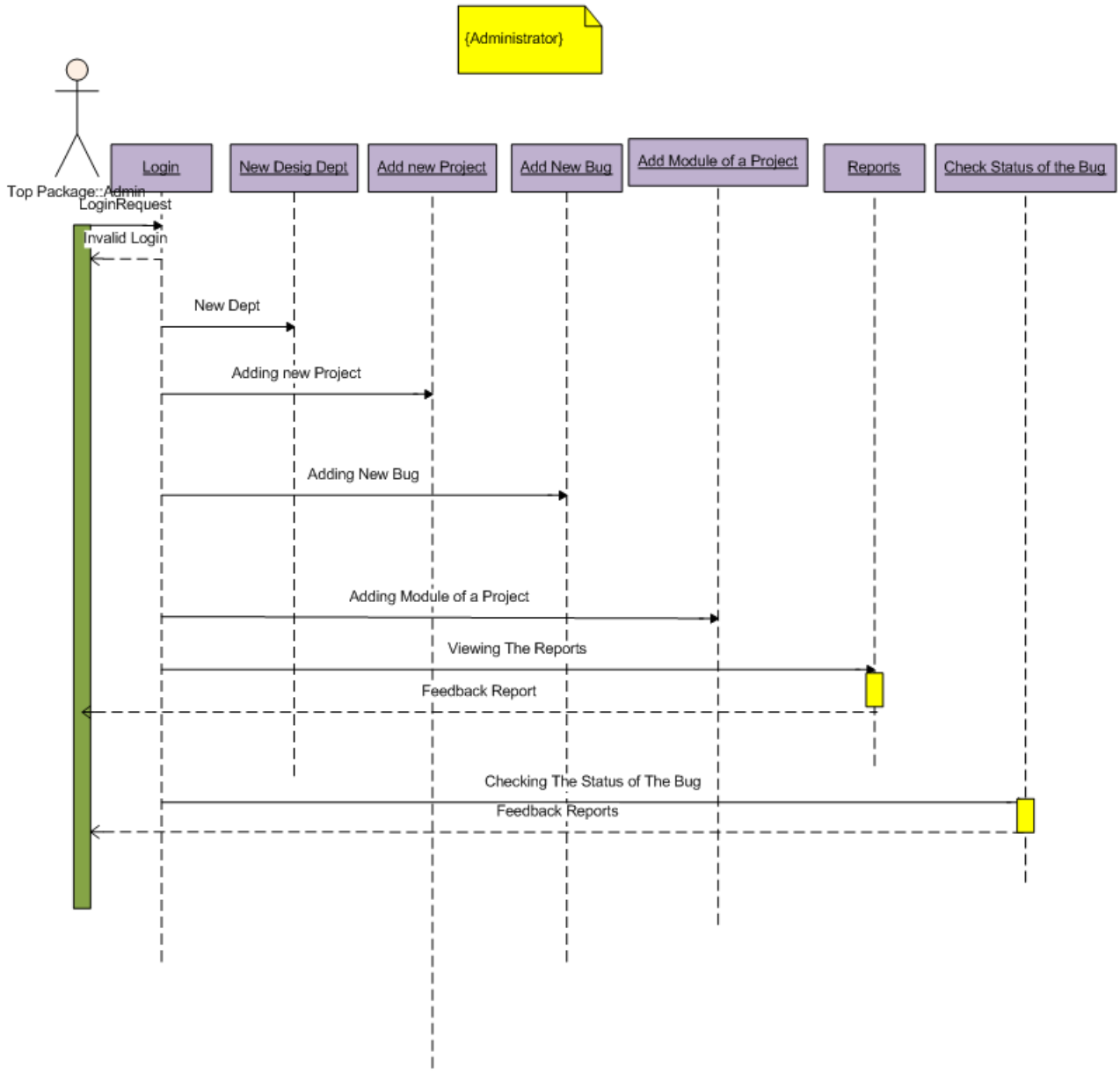


Figure 13: Sequence Diagram-Administrator

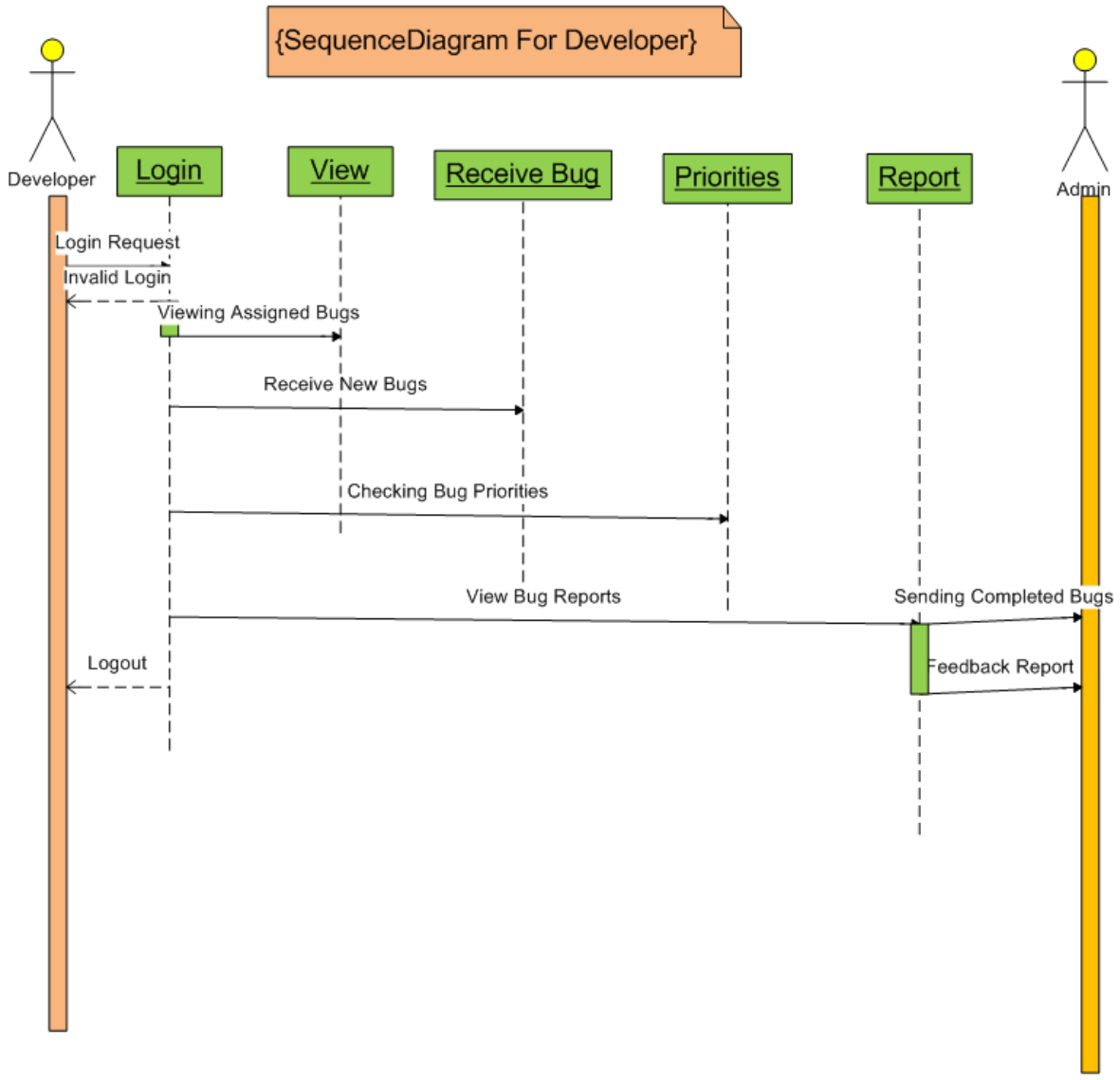


Figure 14:Sequence Diagram-Developer

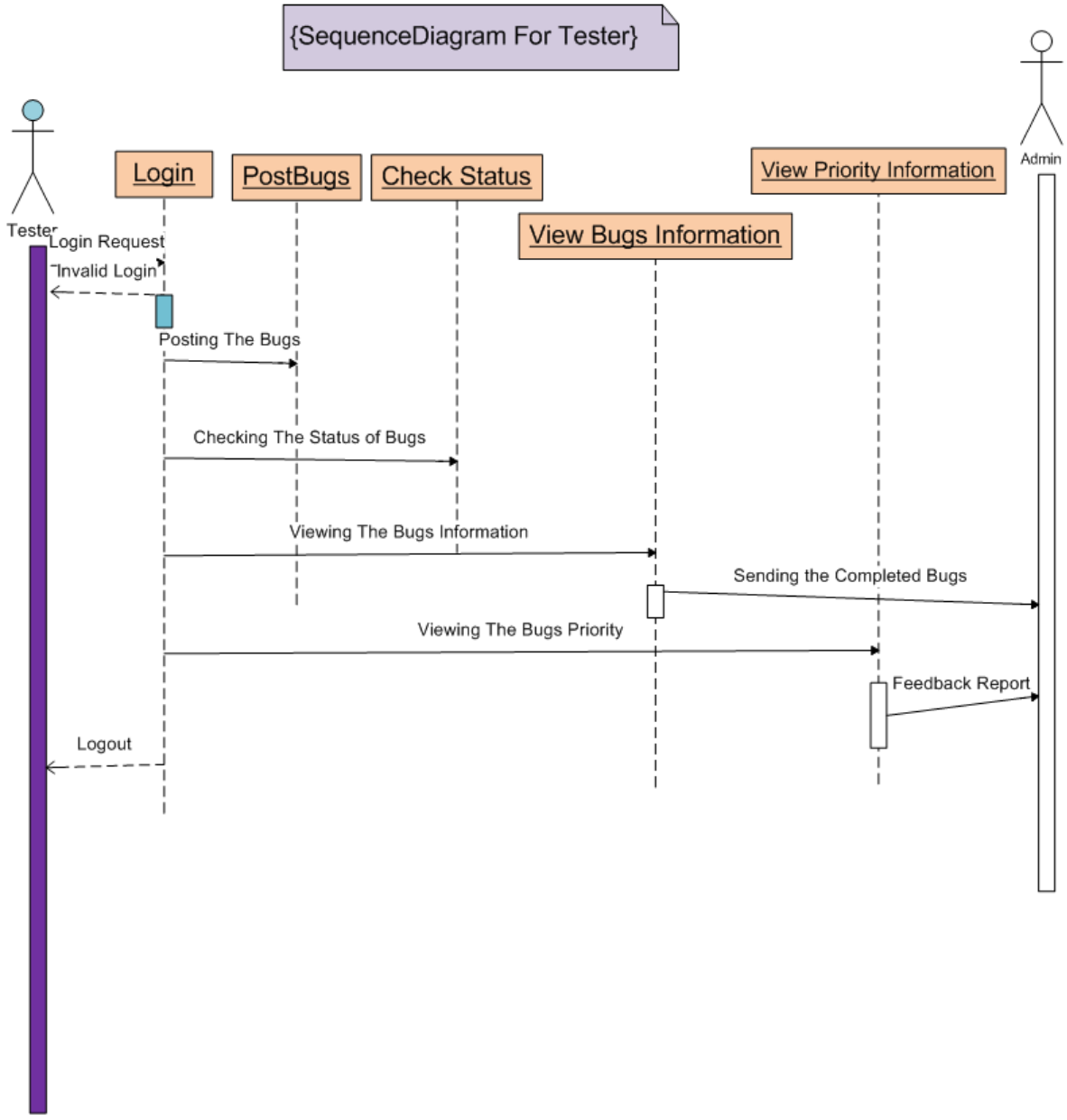


Figure 15: Sequence Diagram-tester

6.3 Algorithms (pseudo codes) suggested for the implementation of the major DAO classes

a) class BugDAO:

1. This class inherits the properties of **AbstractDataAccessObject** class.

2. provides the functionality for adding the priority using **getSequenceID("PRIORITY", "priorityid")** & **bug.getPriorityName()** mrthods.

3.provides the functionality for updating the bug priority by using **bug.getPriorityID()** method.

4.also provides functionality for deleting priority with use of **deleteBug(int bugid)** method.

5 provides functionality for bug solution by **getBugSolution(int bugid)** & bug deletion by **deleteBug(int bugid)** .

b) class MembersDAO:

1. provides the functionality for achiving the user profiles using **getProfiles(String role)** method.

c) class ProjectsDAO:

1. This class inherits the properties of **AbstractDataAccessObject** class.

2. provides the functionality for adding the project using **addProject(Project aProject)** method

3. provides the functionality for updating the project using **pdateProject(Project aProject)** method.

4. provides the functionality for getting all the projects using **CoreHash getAllProject()** method.

5. provides the functionality for getting all the aasigned projects using **CoreHash getManagerProjects(String assignedto)** method.

6. provides the functionality for seeing all the members using **getProjectMembers(String manager)** method..

d) class ProjectsDAOTest:

1. This class is mainly used for testing the available functionalities..

2. provides the functionality for testing the add project methods using **void testAddProject()** method.

3. provides the functionality for testing the updating method for the project using **void testUpdateProject()** method.

4. provides the functionality for testing all the projects using **void testGetAllProject()** method.

5. provides the functionality for testing all the managed projects using **void testGetManagerProjects()** method.

6. provides the functionality for testinging all the assigned projects using **void testAssignProject()**method.

6.4 Refer to Appendix D

Chapter 7: Testing process

7.1 Unit and System Tests:

Test Case Id: TC1

Purpose: To check the priority of the bug.

Precondition: int priorityid=bug.getPriorityID();

Inputs: Value of priorityid

Expected Results: If priorityid equals 1 then it is severe priority, if priorityid is 2 then it is warning.

Test Case Id: TC2

Purpose: Check whether the bug is assigned with severe priority or warning

Precondition: int priorityid= aBug.getAssignedTo();

Inputs: Value of priorityid.

Expected Results: If priorityid = 1 assign the bug with severe priority else if it is 2 assign with warning.

Test Case Id: TC3

Purpose: Inserting the value in projectid and projectname.

Precondition: int projectid = aProject.getProjectID() ;

String projectname= aProject.getProjectName();

Inputs: setvalue of projectid and projectname.

Expected Results: if setvalue =1=> enter projectid

If setvalue= 2=> enter projectname.

Test Case Id: TC4

Purpose: Inserting the start and ending date of the project

Precondition: String projectenddate= aProject.getEndDate();

String projectstartdate= aProject.getStartDate();

Inputs: Set value of startdate & enddate

Expected Results: if setvalue = 3 => start date

If setvalue =4=> end date.

Test Case Id: TC5

Purpose: Find the status of the project & find to whom it is assigned

Precondition: String projectstatus= aProject.getStatus();

String projectassignedto= aProject.getAssignedTo();

Inputs: Set value of projectstaus & projectassignedto.

Expected Results: if setvalue=5=> getprojectstatus

If setvalue=6=> get projectassignedto

Test Case Id: TC6

Purpose: To find the designation of employee.

Precondition: value= String getRole()

Inputs: String getMemberName();

Expected Results: if this.role= ‘devp’=> member is developer

If this.role=’test’=> member is tester

If this.role=’manage’=> member is manager

Test Case Id: TC7

Purpose: To get the raiser date of a defect.

Precondition: value= String getRaisedDate();

Inputs: String getBugName()

Expected Results: if this.setDate()=’Startingdate’=> then we have the date on which the bug was raised.

Test Case Id: TC8

Purpose: To check the successful or unsuccessful login into the system

Precondition: String loginid = regbean.getLoginID();

String oldpassword= regbean.getPassword();

Inputs: int LoginID, String Password

Expected Results: if password = = oldpassword && loginid =’True’=> Successful login

Test Case Id: TC9

Purpose: To check the authenticity of the existing user on login.

Precondition: String loginid=regbean.getLoginID();

String password= regbean.getPassword(); String role= " ";

Inputs: String loginid, String password, String role.

Expected Results: if loginid='True' && oldpassword= =password && role= 'True'=> authenticate user login.

Test Case Id: TC10

Purpose: Changing of the secret question for security.

Precondition: String loginid= regbean.getLoginID();

String password= regbean.getPassword();

int secretquestid= regbean.getSecretQuestionID();

String ownsecretquest= regbean.getOwnSecretQuestion();

String secretans= regbean.getSecretAnswer();

Inputs: String loginid, password;

int secretquestid ;

String ownsecretquest, secretans;

Expected Results: If checkPassword(regbean='True' && secretquestid=0) && secretquestid= rs.getInt(1); => secret question is changed.

Test Case Id: TC11

Purpose: Recovery of password using existing questions.

Precondition: String loginid=regbean.getLoginID();

Inputs: String loginid,

int secretquestid, int secretans

Expected Results: if int secretquestid= regbean.getSecretQuestionID() &&

int secretans= regbean.getSecretAns() => password is recovered.

7.2 Evaluation of Tests

Testing is one of the main stages in the development process in which we check the project for the different data sets and to see which are the cases in which the project failed. This helps in deciding which are necessary steps that can be taken to make sure that the system is immune to such kind of data inputs in the future. In our project we have used the above mentioned test cases. These test cases from TC1 to TC11 test all the important functionalities of the system and pinpoint the cases which make the system fail.

Our system failed for test cases TC1 & TC2 where if the priority of the bug was set to some other number other then 1 or 2 then system crashed. To overcome such problem in future we restricted the input from the user to only 1 & 2 using drop down list. The table below mentions the summary of all the test cases; which of them passed and which of them failed along with the recuperative action that we took to prevent the failing cases.

Test Case ID	Pass	Fail	Comment
TC1		✓	Restricted the user's input to 1 or 2 by using drop down list
TC2		✓	Restricted the user's input to 1 or 2 by using drop down list
TC3	✓		
TC4	✓		
TC5	✓		
TC6	✓		
TC7	✓		
TC8	✓		
TC9	✓		
TC10	✓		
TC11	✓		

Chapter 8: Glossary, acronyms and related definitions

1. **Administrator**: He is the person who has the whole authority over the system.
2. **Developer**: He is the person who is responsible to solve the raised bugs.
3. **Tester:** He is the person who is responsible to check the system & raise the bugs as encountered.
4. **Manager:** manager is the person who role is to supervise the developers and testers under him/her.
5. **Defect:** Defect can be defined as any anomaly that can disrupt the normal functionality of the system.
6. **Bug**: Bug & defect are used interchangeably in our project.
7. **Tracking:** Tracking means to trace down the life cycle of the bug through its resolution cycle.
8. **Software Reliability:** Reliability of a software means how immune is the software against the crashes and how well is the data preserved in case of one.
9. **SDLC:** Software development life cycle is the stages through which the software passes before it is completely deployed at the users end.
10. **UML:** Unified modeling language
11. **Façade:** It is the design patter that ensures that the data libraries can be used more effectively and efficiently.
12. **Subsystem:** It is the part of system that performs some specific function.
13. **Component:** Component is again a part of the system which provides some specific functionality.
14. **Class:** Class is a group of data and imbedded functions
15. **Module:** Module is part of the subsystem which interact together to make system work.
16. **System Request:** Request made to the server via system.
17. **Database:** The collection of all the data in a system
18. **Roles:** Role defines the access rights in a system. Each role has a specific set of rights assigned to it.
19. **Http:** Hyper Text Transfer Protocol
20. **Security**: Security authorizes user in logging into the system.

Chapter 9: Appendix

Appendix A: Use-Case Description

Use Case	Login
Actor	Admin/Manager, Tester, Developer
Precondition	System displays login page to the user
Postcondition	User login successful, if correct user and password is entered
Main path	User opens the browser System's home page is displayed System demands for user login User enters username and password If correct information is entered, user login to the system successfully
Alternative	Login failure if entered information is incorrect

Use Case	Add/EditNewBug
Actor	Admin/Manager, Tester
Precondition	User click on add new defects under 'defects' tab
Postcondition	User successfully add a new bug/defect or edit existing defect
Main path	Actor Login to the system Go to defects Click on view Defects Enter new defect or edit existing defect
Alternative	No access to add/edit defects

Use Case	Check/Update Bug status
Actor?,.	Admin/Manager, Tester
Precondition	User click on view defects under ‘defects’ tab
Postcondition	User successfully view or update bug/defect
Main path	Actor Login to the system Go to defects Click on view Defects View defects or make changes existing defect Changes displayed on view defects page
Alternative	No access to check/update the defects to the user

Use Case	Add/Edit New Priority
Actor	Admin/Manager
Precondition	User login as admin
Postcondition	User successfully view defects with added priority
Main path	Actor Login to the system Go to defects Click on view Defects Click on Add priority button to add priority Priority is displayed on view defects page
Alternative	No access to add/edit priority to the defects to the user

Use Case	Assign Bug to Employees
Actor	Admin/manager
Precondition	User login as admin
Postcondition	User successfully assigned bugs to employees (tester/developer)
Main path	Actor Login to the system Go to Organization Click on view Members Click on assign bugs Assigned bug is displayed on view defects page
Alternative	Failure to access to assign bugs to a user

Use Case	Add New Project
Actor	Admin
Precondition	User login as Admin
Postcondition	Actor Successfully added new project
Main path	Actor Login to the system Go to Organization Click on view Projects Click on Add New Added project is displayed on View Projects page
Alternative	Proper access to add a project is not given to the user

Use Case	Add/Update Employee
Actor	Admin/Manager
Precondition	User login as admin
Postcondition	Actor added new employee successfully
Main path	Actor Login to the system Go to Organization tab Click on view members Click on Add New Added employee is displayed on View member page
Alternative	Proper access to add an employee is not given to the user

Use Case	Resolve Bug
Actor	Developer
Precondition	Actor Login as developer
Postcondition	Developer successfully mark the bug as resolved
Main path	Actor Login to the system Go to Defects tab Click on view Defect Change status to resolve bug Bug status is displayed as ‘resolved’ in defects list
Alternative	Proper access to marl bug status is not given to the user

Use Case	Reports
Actor	Admin
Precondition	Actor Login as Admin
Postcondition	Admin view reports successfully
Main path	Actor Login to the system Go to Reports tab Click on pending defects, resolved defects, or view all defects Reports are displayed as per of click
Alternative	Proper access to 'view reports' is not given to the user

Appendix B: Graphical User Interface Mock up

Figure 16: Snapshot1

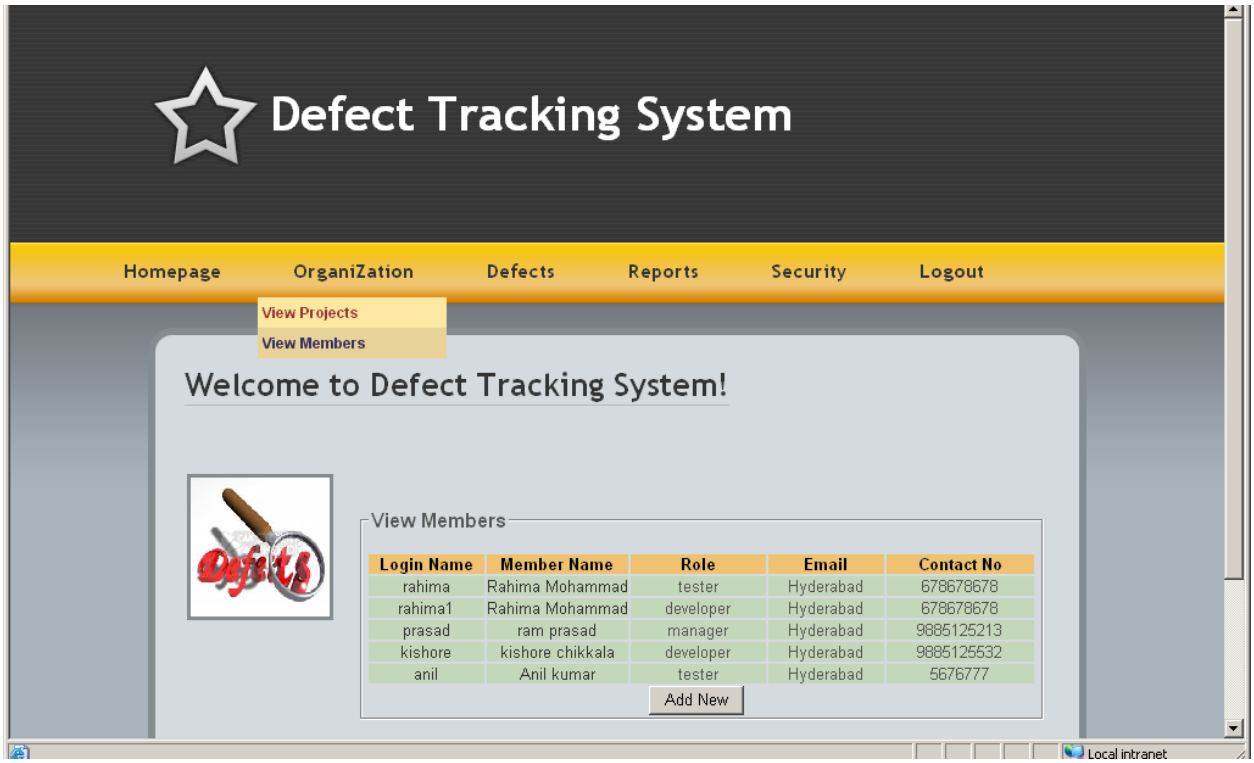


Figure 17:Snapshot2

Figure 18: Snapshot3

Figure 19: Snapshot4

Figure 20: Snapshot5

Figure 21: Snapshot6

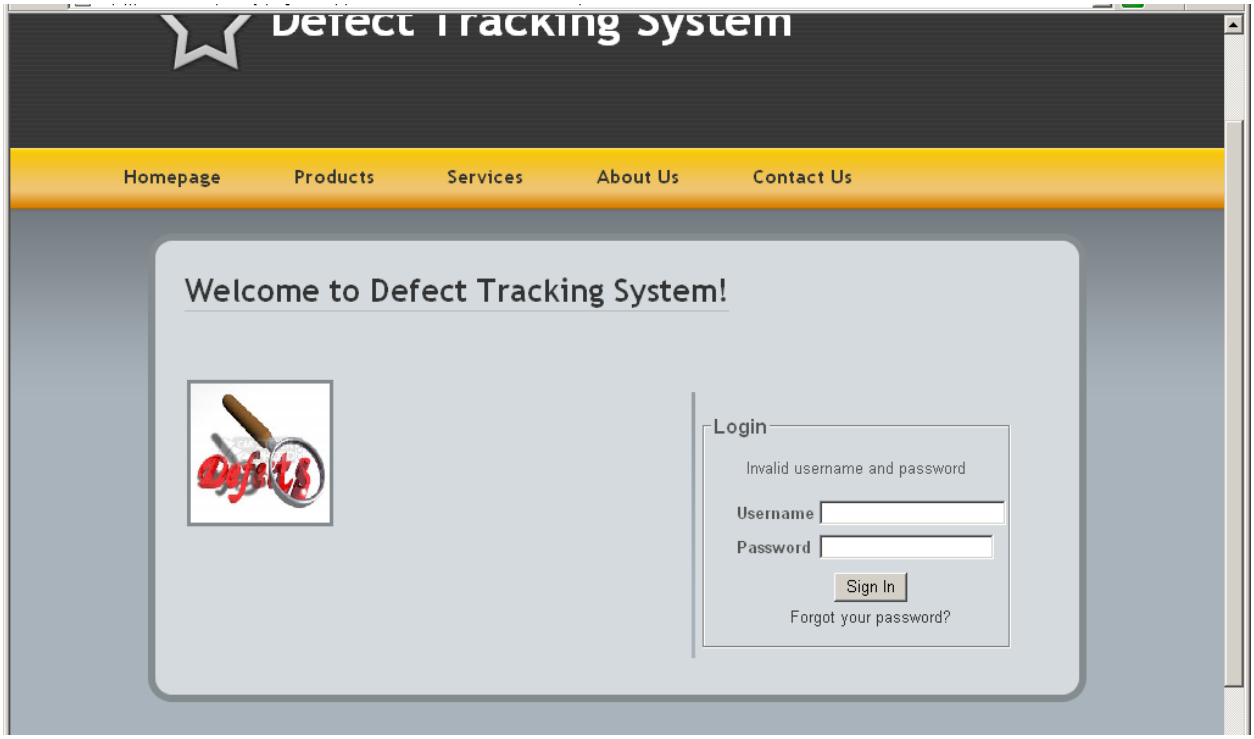
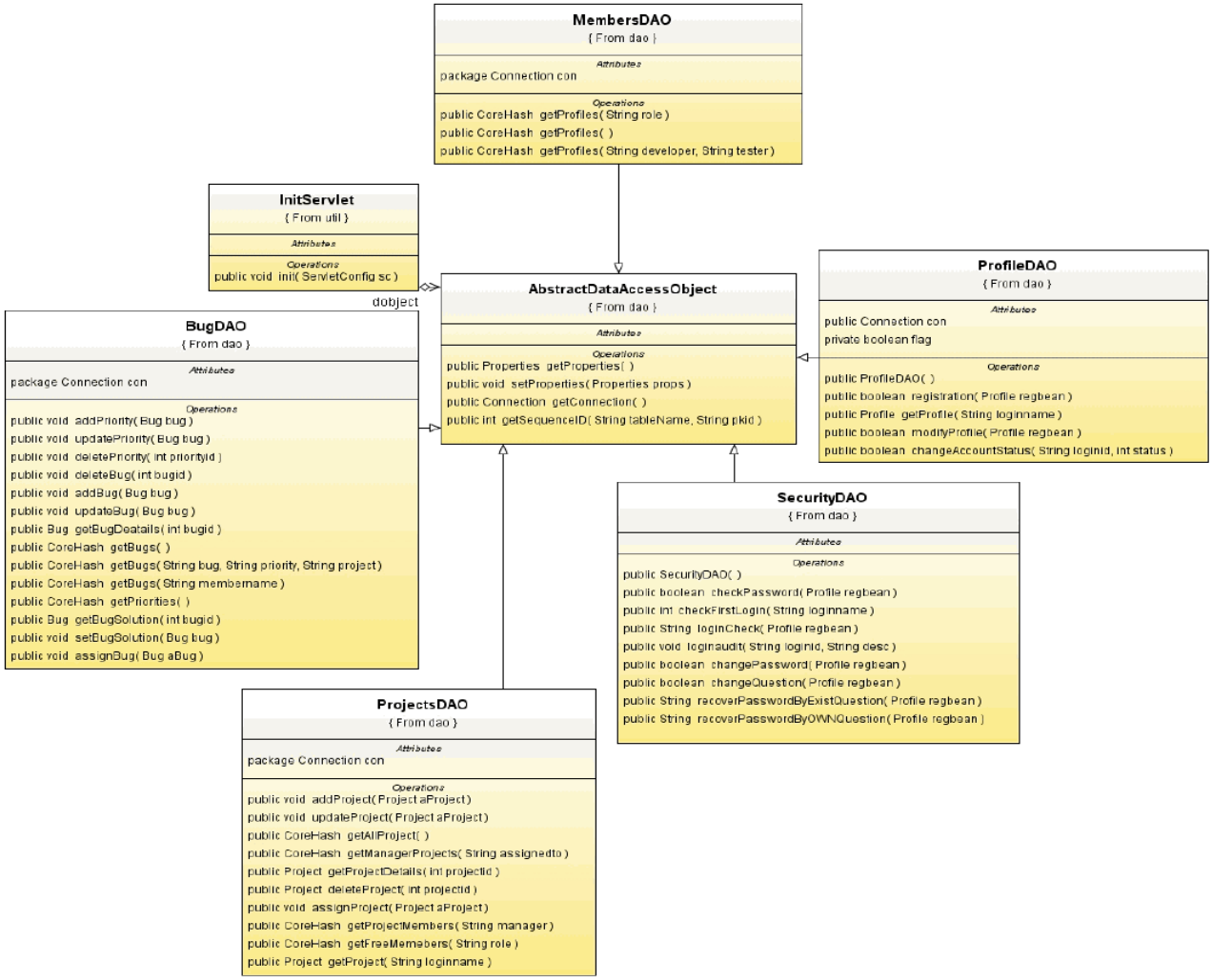


Figure 22: Snapshot7

Appendix C: Detailed Class Diagram



1. Package DAO

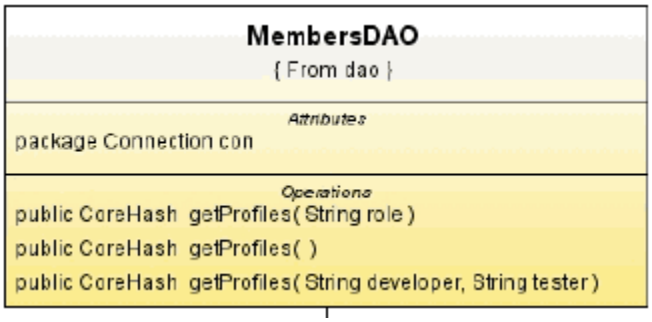


Figure 23: Class-MemberDAO

1.1 MembersDAO: - This class saves the profiles of different users into the database. It provides the getprofile() operation which is used to obtain the profile of the user. It also tells us whether the user is developer, manager, admin or a tester.

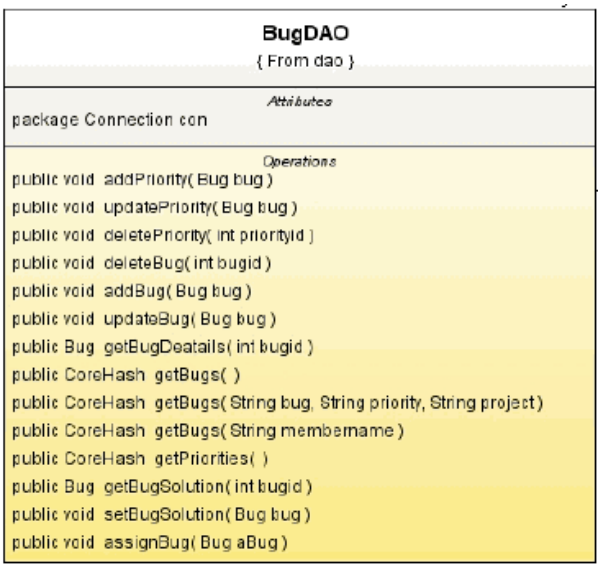


Figure 24: Class-BugDAO

1.2 BugDAO: - This class is responsible to store the information related to the bugs into the database. Using this class we can perform various operation on the bugs like using addPriority() we can add the priority to the bug, or we can update the existing priority of the bug using updatePriority(), deletion of priority can be accomplished through operation deletePriority(). Other operations like addBug(), getBugs(), setBugSolution(), assignBug() perform the same function as the named.

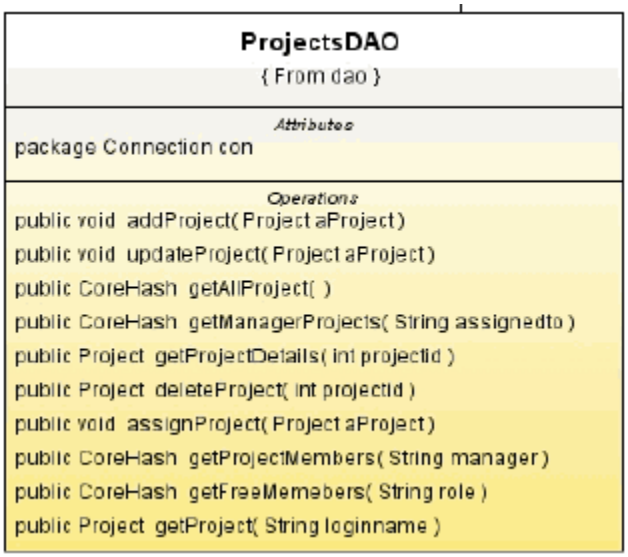


Figure 25: Class-ProjectDAO

1.3 ProjectDAO: - This class is responsible to do the operation on the projects in the system. Some of the operations that this class provides are, addProject() for project addition, updateProject() for making updating the project, getProjectDetails() to get the details related to the project, getManagerProjects() to get the managers and their respective projects as assigned to them, assignProject() is used to assign project to a member entity.

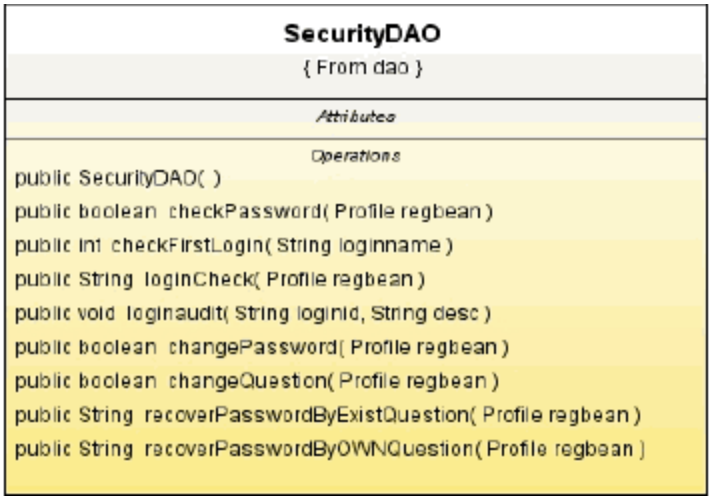


Figure 26: Class-SecurityDAO

1.4 SecurityDAO: - This class deals with security aspects of the system. Some of the operations that this class provides are, checkPassword() which provides the functionality to checking the password related to a particular username, loginCheck() which is used to check the login details of a user, changePassword() which is used to change the password of a particular user, changeQuestion() used for changing the password recovery question. Other important operation provided are recoverPasswordByExistQuestion(),are recoverPasswordByOWNQuestion().

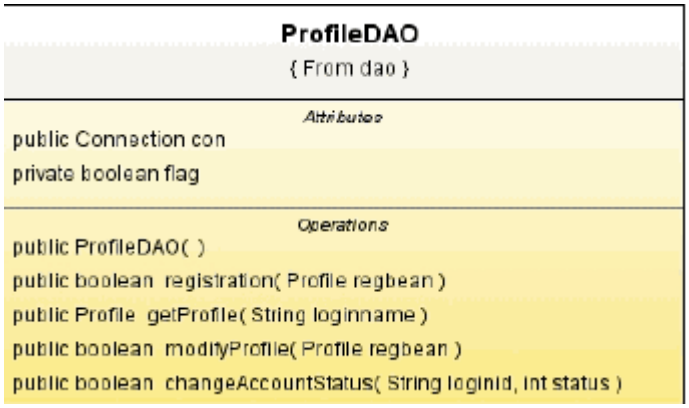


Figure 27: Class-ProfileDAO

1.5 ProfileDAO: - This class provides necessary operation to perform on the profiles of the user. Some of the major operations that are provided are, registration() which is used to register into the system, modifyProfile() which is used to modify the existing profile in the system, changeAccountStatus() which is used to change the status of a profile in the system

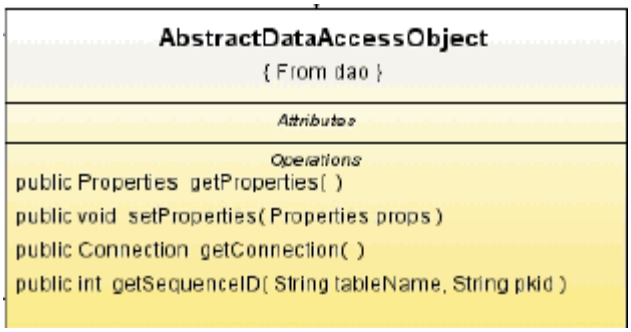


Figure 28: Class-AbstractDAO

1.6 AbstractDataAccessObject: -.This class is responsible to provide an interface between the browser and the database. Some of the operations that this class provides are, getConnection() to get a connection for the database, getSequenceID() to get the sequence of the table in the database.

2. Package Util:

Figure 29: Class-InitServelet

2.1 InitServlet: - This class provides the init operation that is used to initialize the servlet to help it handle the requests that it encounters from the browser.

Appendix D: Coding Standard

Introduction: Here we will describe our strategies & approach toward the existing project design. This section of the document provides the key constraints & nomenclature we followed for the coding & implementation of our system. This portion clears the coding conventions & guidelines that we followed for our software code. So with the help of this document one can understand the key steps & foundation strategies behind the structuring & implementation of the programming logics & the code structure.

Coding Standards: while writing the codes for different logics & their implementation we have followed the following nomenclature:

- Class name: all class names started with a capital latter. It can be a combination of two or more words, every new word starting with a capital latter.
- Method name: all methods name start with a small latter & can be a combination of two or more words with starting a new word by capital alphabet (except first word).
- Every class contains the descriptive note in its header section, explaining the functionality & target.
- Lots of single & multi-level comments sections were used for the detailed understanding of the code before implementation of a new method.

Indentation, Spacing & Alignment Standards: We have followed the following three indentation & Spacing approaches:-

1. Precede and follow single and compound delimiters by a space
   Example: if (a < -b)

2. Precede and follow binary operators by a space.
   Example: a + b

3. Precede unary operators by a space.
   Example: if (!isValid)

4. Use extra spaces, tabs, and blank lines to align closely related statements and declarations in adjacent lines.
   Example:

```
int     numberOfItems  = 10;       // variable names are all  aligned
int     numberOfErrors = 5;                  // variable values are also       aligned
double  errorPrecision = 0.005;  // comments are aligned as well
```

We have also followed some general conventions. Usually the programmers do for the good programming practices. We have used following conventions for our programmable coding units:

1. Do not use variables of anonymous or implicit types.
   Example:
   Bad Example:
   Abhinav = new {ID = R10, firstName = "Abhinav"};

   Good Example:
   Abhinav = new PersonnelMember(R10, "Abhinav");

2. Use named numbers instead of numeric literals. (e.g., VALUEOFPI for 3.14159)

3. Conceal any information irrelevant to the user of a package through the use of private and limited types. (i.e. class variables should be declared as private)

4. Handle exceptions at the lowest possible level at which the program can properly respond to the error. (i.e. exceptions should be caught and handled as soon as possible)

5. Subprograms (functions and procedures) should perform a single logical function. (i.e. use two different functions to calculate the mean and standard deviation instead of having one function calculate both)

6. Place all include, import, or similar statements together immediately after the header.
   Example:

```
/* ...
* end of header        */

        import java.util.*;
        import java.swing.*;
        ...
```

UI Guidelines: It is the only part of the system that is visible to the user so a proper UI design is an important requisite of any particular system. In development of our defect tracking system we have followed the following UI guidelines:

1. Use a consistent design and commands when developing the user interface.

2. Sequences of actions performed by users should be grouped and designed to provide closure to a user upon completing those actions.

3. Design the user interface so that the user feels in control of the system

4. Design the system to protect users from committing critical errors and include mechanisms for handling any errors that may occur.

5. The interface should provide informative feedback for users in response to their actions.

Coupling & Cohesion: Coupling and cohesion are the two main principles that need to be followed to make sure that the system performs in an appropriate manner. This makes code more readable and at the end of the day makes it easy to trace down and correct any error or bug in the system (Code). The rule of thumb here is to ensure that the system has **High Cohesion & Loose Coupling.** Cohesion ensures that there is high level of interaction between the component of a single module hence makes it self-reliant. On the other hand low coupling makes sure that there is minimum level of interaction between various modules of the system thus making sure that the individual modules are not too much reliant on each other & hence if one module fails it doesn't affect the other modules working.