Low Level Design Document

**for**

Bug Track

### Software Bug Reporting & Tracking System

# Table of Contents

[Table of Contents ii](#_TOC_250017)

[Revision History ii](#_TOC_250016)

1. [Introduction 1](#_TOC_250015)
   1. Purpose…………………………………………………………………………………………............3
   2. System Requirements……………………………………………………………………………3
2. Detailed System Design………………………………………………………………………4
   1. UI Diagram………………………………………………………………………………………6

2.1.1Admin dashboard ………………………………………………………………………….6

2.1.1.1Active Bugs………………………………………………………………………………7

2.1.1.2Bug History……………………………………………………………………………..10

2.1.1.3Manage User……………………………………………………………………………12

2.1.2Engineer Dashboard………………………………………………………………………16

2.1.2.1 Active bugs……………………………………………………………………………..17

2.1.2.2 Bug History……………………………………………………………………………..18

2.1.3 customer Dashboard …………………………………………………………………….21

2.1.3.1 Report bug ……………………………………………………………………………..22

2.1.3.2 Active bugs …………………………………………………………………………….24

2.1.3.3 Bug History…………………………………………………………………………….26

* 1. Class Diagram…………………………………………………………………………………..27
  2. Sequence Diagram……………………………………………………………………………...30

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Arunima Jayaraj | 30/06/2022 | Initial Draft | 1.0 |

# Introduction

## Purpose

Bug Track is a software that helps to maintain a database that is useful to store data of the reported bug’s within an Application and generating reports upon request. It helps in assembling, organizing and managing the information of the Engineers and reported bug’s as required by the user. Bug Track allows a small company or organization to maintain details of reported bug’s information in a more operative manner that will help them to manage and resolve bug’s effectively .This system will reduce all the manual work and the whole process can be managed just through single clicks and edits. It reduces the manual paperwork through it and gives proper information of bug’s that have been recorded. Bug Track System makes easy for the Admin to keep a track and monitor the reported bug’s and manage them.

## System Requirements

**Hardware Requirements:**

* Minimum Processor
* RAM
* Hard Disk, etc..

**Software Requirements:**

* OS : Windows 10 or above
* IDE: Visual Studio 2017 or above

# Detailed System Design

# 2.1 UI Diagram

# 

# Login page:

# The system allows the field to enter username and password to login.

# Admin provide default password to the users initially .lately user can change the password.

# The system will validate the username and password.

# If the login is successful then user (Admin/Engineer/Customer) can login and view any information

# 

# Login page:

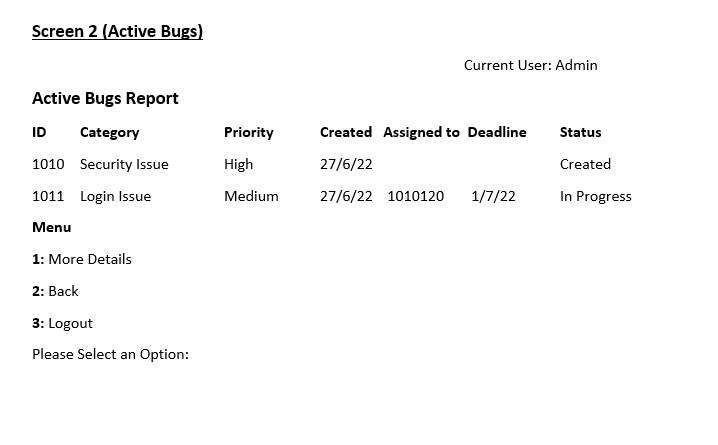
# The system will show an error message if it's an invalid User.



# 2.1.1Admin Dashboard:

# The system will allow the admin to choose the options .press 1 to show Active Bugs , 2

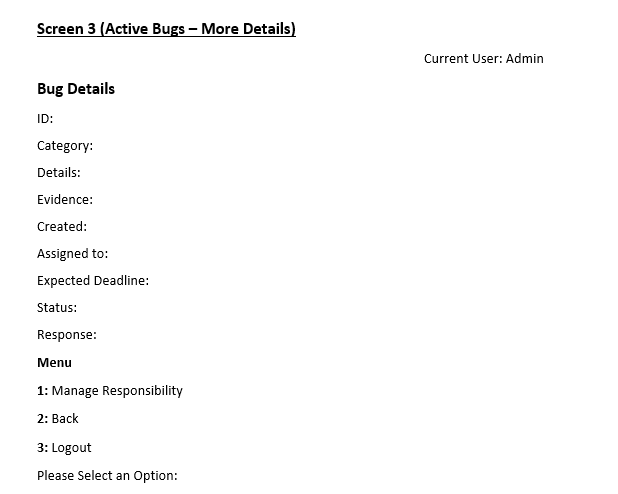
# For showing Bug History option 3 for manage user and option 4 to logout.



# 2.1.1Admin Dashboard:

# 2.1.1.1Active bugs

# The system will enable the Admin to view all the active bug’s details in the database. The system will display all the active bugs. The system will allow the admin to choose the options .press 1 to show More Details of bugs, press 2 back to admin page and 3 for logout

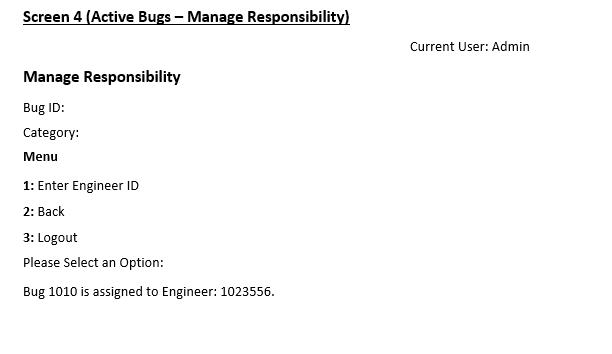


# 2.1.1Admin Dashboard:

# More Details

# The system will enable the Admin to view bug’s details like Id, category, details, evidence , created, assigned to, expected deadline, status and Response.

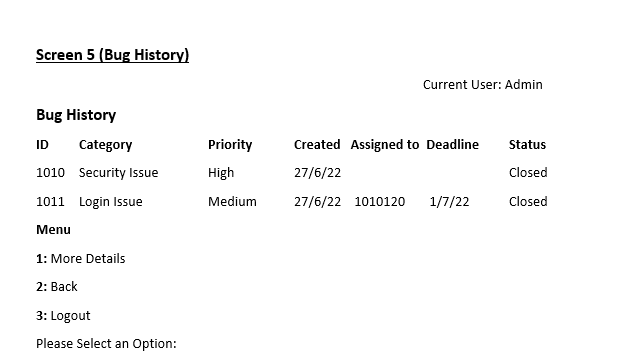
# The system will allow the admin to choose the options .press 1 for manage responsibility, 2 for back to main menu and 3 to logout



# 2.1.1Admin Dashboard:

# Manage Responsibility

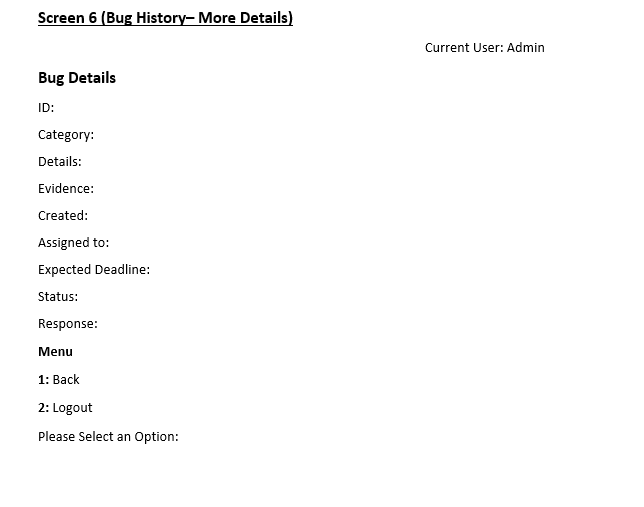
# Admin can see the assigned bud Id by entering the engineer Id



# 2.1.1Admin Dashboard:

# 2.1.1.2Bug History

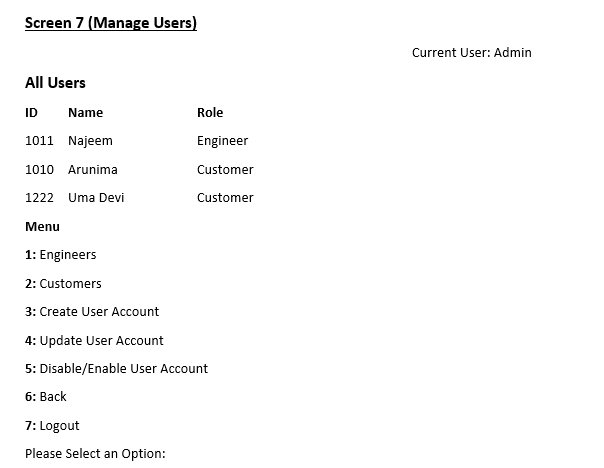
# The system will show Id, Category, priority, created date, assigned to which Engineer, deadline and current status.



# 2.1.1 Admin Dashboard:

# More Details

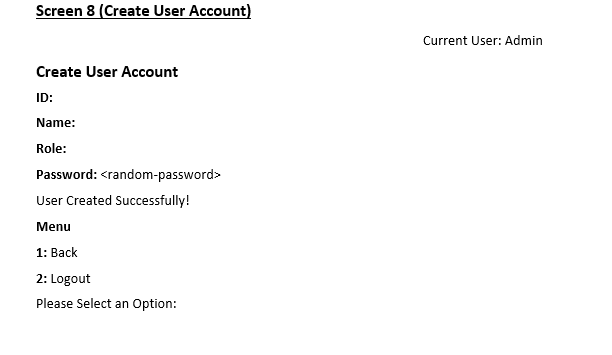
# The system will enable the Admin to view bug’s details like Id, category, details, evidence , created, assigned to, expected deadline, status and Response.



# 2.1.1Admin Dashboard:

# 2.1.1.3.Manage User

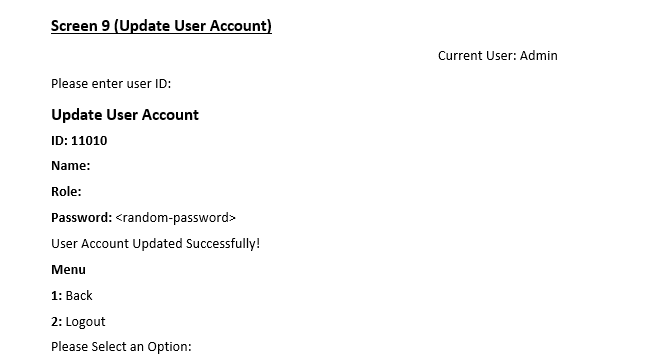
# Shows all users details such as Id, user’s Names, and Role. Option 1 to show all engineers details, option 2 to show all customer details, option 3 for create new user account, option 4 for update user account , option 5 to enable/disable the user, option 6 for go back to admin main menu and option 7 for logout.



# 2.1.1Admin Dashboard:

# Create User Account

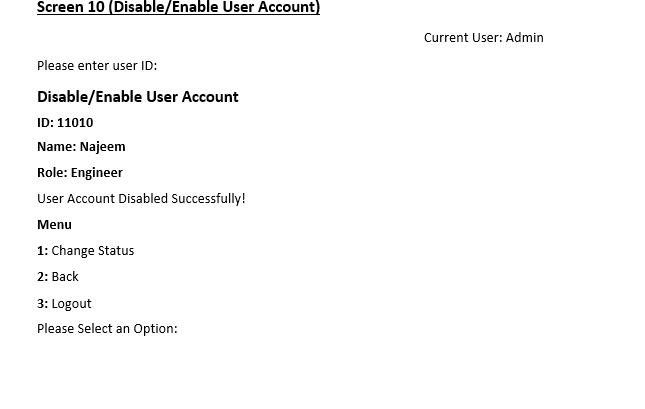
# The system will enable the Admin to create user details .Admin can create user of Id, Name, role and password .The system will display a message user created successfully.



# 2.1.1 Admin Dashboard:

# UpdateUser Account

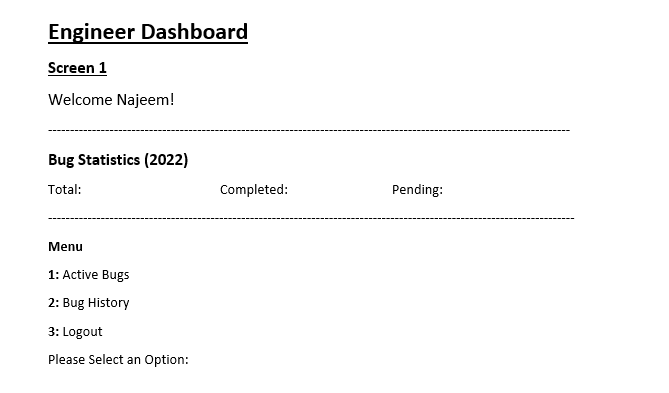
# The Admin should login to update user Details .The system will enable the Admin to view all the added user details in the database and update the details. The system will display a message user account updated successfully.



# 2.1.1Admin Dashboard:

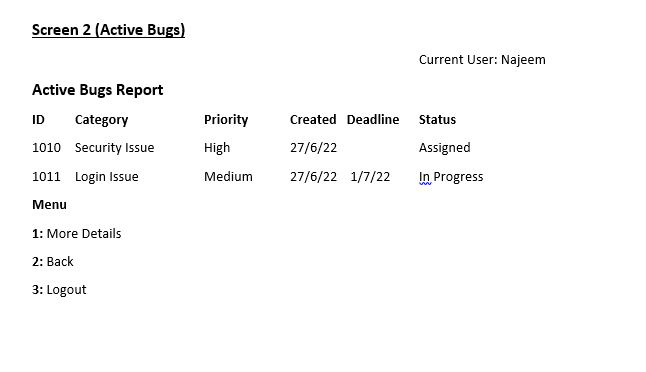
# Disable/Enable User Account

# The Admin can disable/enable user according to their Id .The system will display all the fields to disable/enable user .The system will ask to enter submit button .The system will show a message “User Account Disabled Successfully” on successful disable.



# 2.1.2 Engineer Dashboard:

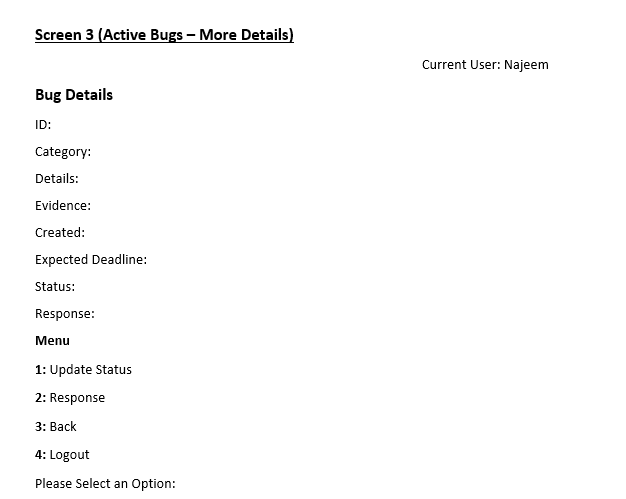
# If the login is successful then the Engineer can login and view active bugs and bug’s history



# 2.1.2 Engineer Dashboard:

# 2.1.2.1Active Bugs

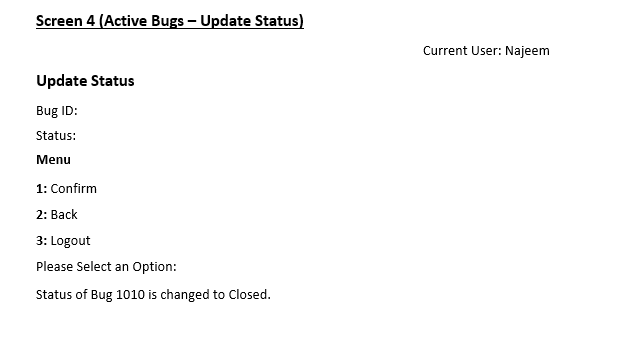
# The system will enable the Engineer to view all the active bug’s details in the database. The system will display all the bugs



# 2.1.2Engineer Dashboard:

# More Details

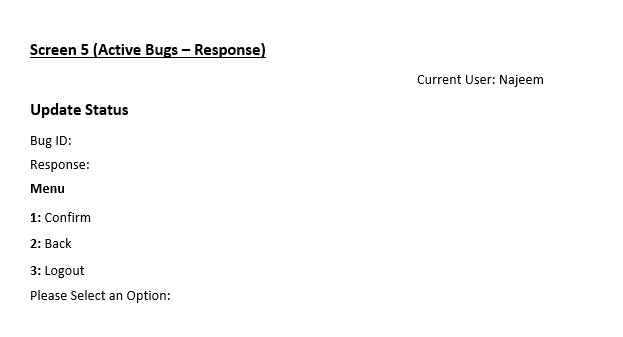
# The system will enable the Engineer to view all the active bug’s details in the database. The system will display all the bugs . The system will enable the Engineer to view bug’s details like Id, category, details, evidence, created, Expected Deadline , status and Response



# 2.1.2 Engineer Dashboard:

# Update Status

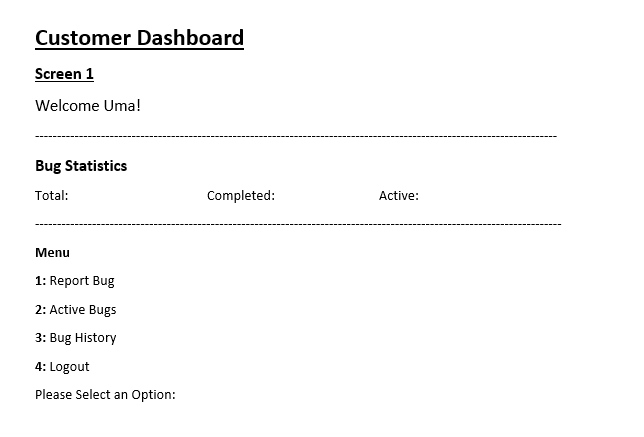
# The Engineer should login to update bug status. Engineer needs to update Details of bug. The system will display all the fields to update for bug’ Details and will ask to confirm bug status and system will show updated status of bug.



# 2.1.2Engineer Dashboard:

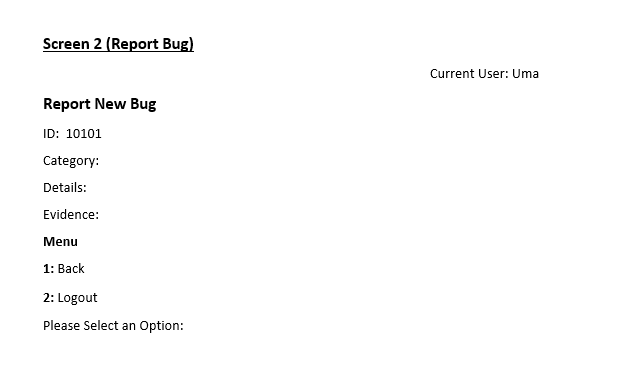
# Response

# The Engineer can respond to the user about the bug and system will ask to confirm response .



# 2.1.3 Customer Dashboard:

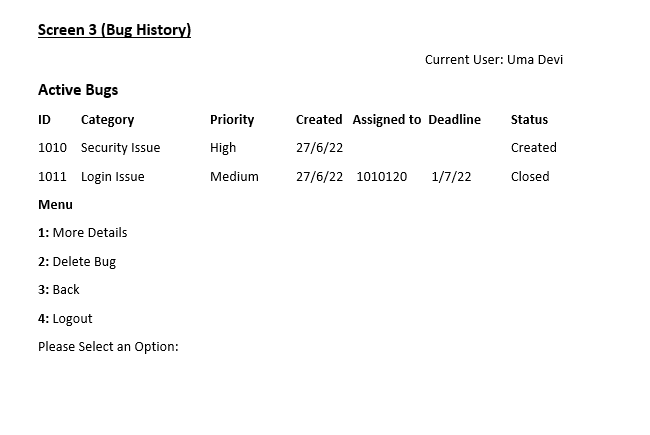
# The system will display the fields to fill up for adding username and password. The system will validate the username and password. The system will notify the user if the username or password is not valid. The system will show an error message if it's an invalid User. If the login is successful then the user can login and view bug’s history.



# 2.1.3 Customer Dashboard:

# 2.1.3.1Report Bug

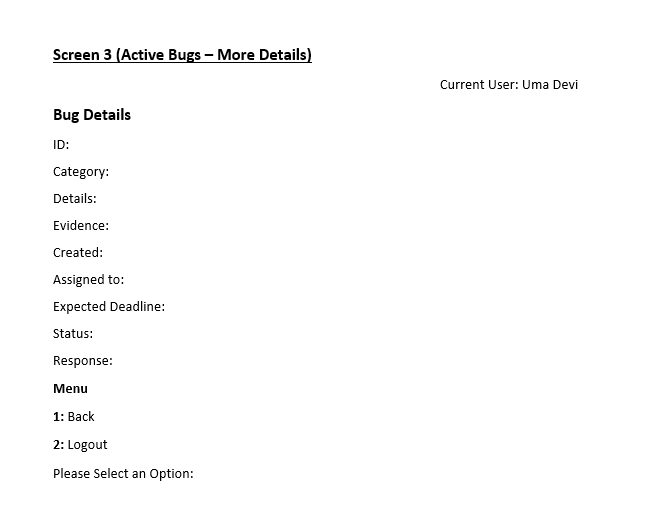
# The system will display the fields to fill up for reporting bug. The system will create an Id for the bug. The system will ask for category, details and evidence of the bug. The system will show a message on successful reporting of a bug.



# 2.1.3Customer Dashboard:

# Bug History

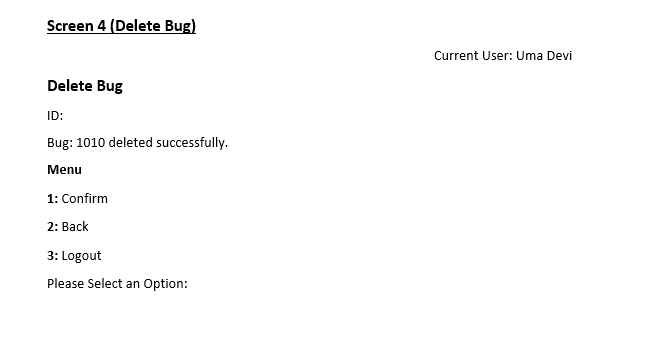
# The system will display bug’s history according to their Id. The system will show Id, Category, priority, created date, assigned to which Engineer, deadline and current status.



# Customer Dashboard:

# More Details

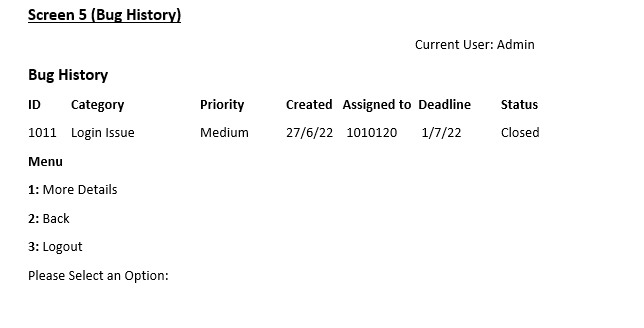
# The system will enable the customer to view bug’s details like Id, category, details, evidence, reported date ,expected deadline, status and response.



# Customer Dashboard:

# 3.Delete Bug

# Customer can delete the bug by using bug Id.



# Customer Dashboard:

# 4.Bug History

# The system will display bug’s history according to their Id. The system will show Id, Category, priority, created date, assigned to which Engineer, deadline and current status.

# 2.2 Class Diagram

# 

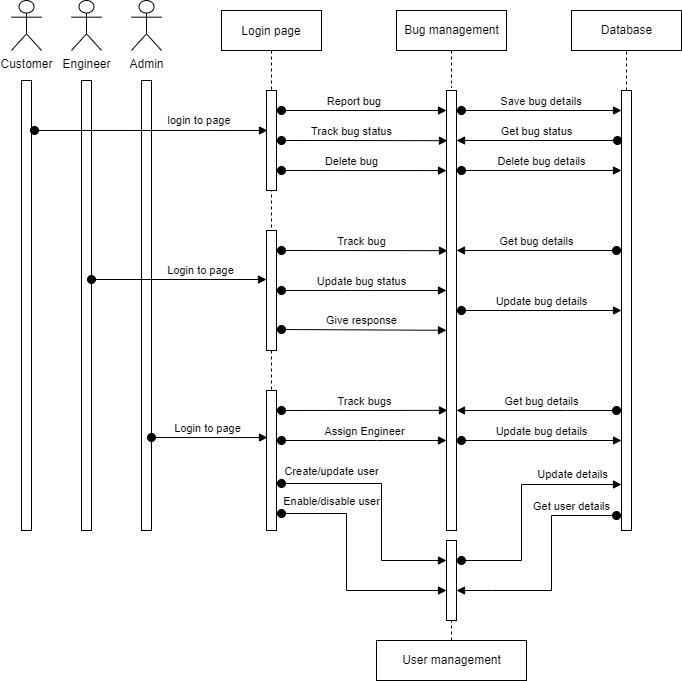
**User Class**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| User() | Default Constructor to initialize member variables |
| User(long int userId,char\* userName,char\* role,char\* password) | User defined constructor to assign user Id ,user name , role and password and return User object |
| bool Login() | To check login is successful or not and true or false |
| void LoadUserDashboard() | To load user dashboard |
| Int CountOfUsers() | To get the total number of users |
| String IsValidUser(long int userId) | To check the user is valid or not |
| User\* GetUserDetails() | To get user details |
| bool UpdatePassword(long int userId) | To change user password |
| bool ChangeUserStatus(long int id) | To change bug status |
| bool IsActiveUser() | To check user is active or not and return true or false |
| bool IsLogged() | To check user is logged in or not and return true or false |
| void Logout() | To logout |

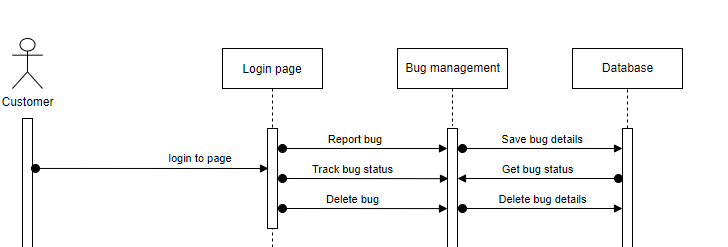
**Bug Class**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Bug() | Default Constructor to initialize member variables |
| Bug(long int bugId, char\* category, char\* description, char\* evidence, long int customerId, char\* priority) | User defined constructor to assign bug Id, category , description , evidence ,customer id , and priority |
| Int CountOfBugs() | To count the total Bug |
| Int CountOfActiveBugsOfUsers(long int userId) | To count the active bugs of user |
| Bug\* GetBugDetails() | To get bug details |
| Bool GetMoreDetails(long int BugId) | To get more details about the bug based on BugId |
| string GetPriority() | To get the priority of bug |
| long int ReportBug() | To return bug report |
| long int DeleteBug() | To delete bugs |
| bool SetEngineer(long int bugId,long int engineerId) | To assign engineer |
| Bool UpadateStatus(long int id,long int bugId) | To change user bug status |
| Long int LastBugId() | To get last bug Id |

# 2.3 Sequence Diagram



# 2.3.1 Customer



# 2.3.2 Engineer

# 

# 2.3.3 Admin

# 