BUG Tracking System project report that i have made

BBA (Jagannath International Management School)

Introduction of the Project Bug Tracking System:

Bug Tracking System is the system which enables to detect the bugs. It not merely detects the bugs but provides the complete information regarding bugs detected. Bug Tracking System ensures the user of it who needs to know about a provide information regarding the identified bug. Using this no bug will be unfixed in the developed application.

The developer develops the project as per customer requirements. In the testing phase the tester will identify the bugs. Whenever the tester encounters number of bugs he adds the bug id and information in the database.

The tester reports to both project manager and developer. The bug details in the database table are accessible to both project manager and developer.

When a customer puts request or orders for a product to be developed. The project manager is responsible for adding users to Bus Tracking System and assigning projects to the users.

The project manager assigns projects to the developers. The developer develops the projects as per customer requirements. The project manager itself assigns the developed applications to the Testers for testing. The tester tests the application and identify the bugs in the application. When the tester encounter no. of bugs, he generates a unique id number for each individual bug. The bug information along with its id are mailed to the project manager and developer. This is Bug Report. These are stored in the database. This is useful for further reference.

Bug information includes the bug id, bug name, bug priority, project name, bug location, bug type.This whole process continues until all the bugs are got fixed in the application.

The bug report is mailed to the project manager and the developer as soon as the bug is identified. This makes that no error will go unfixed because of poor communication. It makes ensure that anyone who needs to know about a bug can learn of it soon after it is reported.

Bug Tracking System plays an vital role in the testing phase. But it supports assigning projects for the developer, tester by the project manager. The Bug Tracking System maintains the different users separately i.e., it provides separate environments for project manager, developer and tester.

Abstract of the Project Bug Tracking System:

For many years, bug-tracking mechanism is employed only in some of the large software development houses. Most of the others never bothered with bug tracking at all, and instead simply relied on shared lists and email to monitor the status of defects. This procedure is error-prone and tends to cause those bugs judged least significant by developers to be dropped or ignored.

Bug Tracking System is an ideal solution to track the bugs of a product, solution or an application. Bug Tracking System allows individual or groups of developers to keep track of outstanding bugs in their product effectively. This can also be called as Defect Tracking System.

The Bug Tracking System can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance.

Some salient features are …

1. Product and Component based
2. Creating & Changing Bugs at ease
3. Query Bug List to any depth
4. Reporting & Charting in more comprehensive way
5. User Accounts to control the access and maintain security
6. Simple Status & Resolutions
7. Multi-level Priorities & Severities.
8. Targets & Milestones for guiding the programmers
9. Attachments & Additional Comments for more information
10. Robust database back-end

Modules & Description

1. **Authenticate User**

The Bug Tracking System first activates the login form. Here the user enters the User name and password and our system starts the authentication process in which the username and password are matched with the existing username and password in the database. If the password matches then it is allowed to the main page else it warns the user for Invalid User name and password.

After successful authentication the system activates menus. The activity log also prepared for failures and security.

2. **Products**

- **List Of Products**

  After successful authentication the user is provided with the list existing products. Here the user can view the details of products and can modify the existing products. This project even provides the facility of adding new projects.

- **Product Versions**

  All the products are maintained in several versions. As it is not possible to complete the whole project in a single version Features required for the product are categorized into several version with dead lines. And the versions are completed according to their dead line dates. Here the user can add new versions to a product or can modify the existing details of version.

- **Product Users**

In order to complete the project each product is allotted with Resources or users. First all the employees with their names and qualifications are stored in the database. Each user is allotted to the product based on their rating,

Qualification and designation. For each user Effective date is stored which specifies the total period a user is valid for that product.

3. **Bug Details**

- **Bug Details**

  In this module the user is provided with the facility for adding bugs or updating the existing bugs. As the number of bugs for a product can be very large this system is provided with efficient filtering. The user can filter the bugs based on the priority, database, operating system and status. After the user applies filter the list of bugs are displayed from the database.

- **Bug History**

  Here the bug history is maintained. All the solutions given for the bug resolution by various users are stored. As the bug needs several techniques or methods for resolution it is important to store the history of the bug.

- **Bug Assignee**

  This displays the list of users for whom the bug is assigned for resolution. As the bug need to be resolved for completing the product several user are assigned to find a solution for the bug. The user can add this bug to a new user or he can modify the existing user details.

- **Bug Attachments**

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

4. **Bug Tracking**

- **Track Hierarchy**

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

- **Track Resolution**

This displays a list of all solutions provided by the users allotted to a bug. This stores the action type and the necessary resolution provided by the user.

- **Track Resources**

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

5. **View**

- **Product Bug Hierarchy**

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

- **Product User Hierarchy**

  This module if for displaying the users allotted to the bug. The users along with their name and designation are displayed in this module. Even in the allotment of resources there can be hierarchy between the employees depending on their designation. So this module simplifies the hierarchy among the employees.

6. **Search**

Our system provides with the feature of advanced search technique. Generally Number of bugs for a project increased tremendously so if we want to know about a particular bug It takes much amount of time. With the search screen provided one can filter the bug's base on priority, product, severity, database and type of operating system. He can also list the bugs between particular time based on the start date and end date. After Searching it displays a list of bugs. From this list the user can modify the existing bugs or can add a new bug.

7. Admin

- **Users**

  All the users of this system are displayed in this module. One can add new user or can update the details of an existing user. Here the password provided by the user is encrypted before saving them to the database for proper security. This module saves the details like address, phone and email.

- **Configuration**

  All the Values that we are using in this system are configurable. Values like status, priority and others can be added dynamically on the screen. Suppose if we limit these fields by hot coding them and if the user wants to add a new value again he has to come to the developer of the product. So In order to avoid this it is provided with the feature of adding values from the screen. And the user can change the status to In Active whenever he wants.

- **Log View**

  In order for the efficient Tracking of the system logs are maintained. As the logs will be in vast it will be a problem for user for checking the database. The Log View module can be searched based on the user and Records between a start date and end date.

8. Logout

In this once the user clicks on Log out First the session variable is killed and then the system is redirected to the login page.

9. Prepare Logs

At all the stages, whenever user performs an operation by clicking a button, automatically the Bug Tracking System logs the activity.

SYSTEM ANALYSIS

INTRODUCTION TO SYSTEM ANALYSIS

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

Existing System

In the existing system the exams are done only manually but in proposed system we have to computerize the exams using this application.

- Lack of security of data.
- More man power.
- Time consuming.
- Consumes large volume of pare work.
- Needs manual calculations.
- No direct role for the higher officials

Proposed System

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

- Security of data.
- Ensure data accuracy's.
- Proper control of the higher officials.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study .The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

TECHNICAL FEASIBILITY

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within latest technology. Through the technology may become obsolete after some period of time, due to the fact that never version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.1 PRODUCT PERSPECTIVE

Web traffic can be analyzed by viewing the traffic statistics found in the web server log file, an automatically-generated list of all the pages served. A hit is generated when any file is served. The page itself is considered a file, but images are also files, thus a page with 5 images could generate 6 hits (the 5 images and the page itself). A page view is generated when a visitor requests any page within the web site – a visitor will always generate at least one page view (the main page) but could generate many more.. Web Traffic Analyzer is aimed towards the vendors who want to reach out to the maximum cross-section of customer and common people who can be potential customer. This project envisages bridging the gap between the seller, the retailer and the customer. Web Traffic Analyzer should be user-friendly, 'quick to learn' and reliable software for the above purpose. OSM is intended to be a stand-alone product and should not depend on the availability of other software. It should run on both UNIX and Windows based platform.

2.2 PRODUCT FUNCTIONS

Web traffic is the amount of data sent and received by visitors to a web site. It is a large portion of Internet traffic. This is determined by the number of visitors and the number of pages they visit. Sites monitor the incoming and outgoing traffic to see which parts or pages of their site are popular and if there are any apparent trends, such as one specific page being viewed mostly by people in a particular country. There are many ways to monitor this traffic and the gathered data is used to help structure sites, highlight security problems or indicate a potential lack of bandwidth — not all web traffic is welcome. Some companies offer advertising schemes that, in return for increased web traffic (visitors), pay for screen space on the site. Sites also often aim to increase their web traffic through inclusion on search engines and through Search engine optimization.

2.3 USER CHARACTERISTICS

We have 2 levels of users

- User module: This is a normal level of user who will be very few number of functionality for website
- Administration module: This user is an admin type who has full rights on the system.

2.4 GENERAL CONSTRAINTS

The amount of traffic seen by a web site is a measure of its popularity. By analysing the statistics of visitors it is possible to see shortcomings of the site and look to improve those areas. It is also possible to increase (or, in some cases decrease) the popularity of a site and the number of people that visit it.

2.5 ASSUMPTIONS AND DEPENDENCIES

All the data entered will be correct and up to date. This software package is developed using JSP as front end which is supported by Aapache Server system. MySQL as the back end which is supported by Window 7.

3.1.1 User Interface

- HTML has been used for developing the User Layout for the system
- JavaScript has been used for creating all the validations and client side scripting functionality
- CSS has been used for designing the web pages of the system

3.1.2 HARDWARE INTERFACE:

- Processor : Intel Pentium IV or more
- Ram : 512 MB or more
- Cache : 1 MB
- Hard Disk : 10 GB recommended

3.1.3 Software Interface:

- Client on Internet: Web Browser, Operating System (any)
- Web Server: Operating System (any), Apache 2
- Database: MySQL
- Scripting Language: JSP, JavaScript, JQuery

3.1.4 Communication Protocol

Following protocols are required to be permitted on the server side

- HTTP incoming request

3.2 Functional Requirements

- The system runs of apache server so it is needed that server must have apache server version 2.0 available
- We have used JSP for server side scripting so the current version of JSP must be available on the server
- MySQL database has been used for storing the data of the website
- HTML has been used for creating the layout of the web application
- CSS has been used for creating the designing of the webpages
- JavaScript scripting language has been implemented on the system for performing all of the Client Side Server Validation.

3.4. Classes and Objects of the Project

- Login Class: Used for performing all the operations of the login functionality.
- Page Class: Class for managing all the operations of the page.
- Traffic Class: Class for managing the traffic of the website
- IP Class: It has been used for storing all the IPs which hits the website
- Users Class: Class for managing all the user operations
- Permission Class: This class has been used for managing all the permissions level opeations.

3.5. Non-Functional Requirements

- Performance: System should be able handle multiple users at a time using any of the web browsers.
- Reliability: Database updating should follow transaction processing to avoid data inconsistency.

- Availability: The project will be deployed on a public shared server so it will be available all the time and will be accessible anywhere of the world using internet.
- Security: We have implemented a lot of security mechanism to avoid to hack the system by outer world.
- Maintainability: It is very easy to maintain the system. The system has been developed on JSP so anyone who has the knowledge of JSP, can easily maintain the system
- Portability: Yes this system is portable and we can switch the servers very easily.
- Browser Compatibility: The project being web based required compatibility with at least the popular web browsers. Microsoft Windows XP and above, Linux and Macintosh being the current popular operating system and Microsoft Internet Explorer, Mozilla Firefox, Opera, Safari and Google Chrome being the currently popular web browser.

Operating System / Browsers	Win 2000	WinXP	WinXPSP2	Win Vista	Win 7	Mac OS	Linux
	Modern Browsers						
IE 8.0	N/A	SUPP	SUPP	SUPP	SUPP	N/A	N/A
IE 7.0	N/A	N/A	N/A	N/A		N/A	
IE 6.0	N/A	N/A	N/A	N/A		N/A	
Firefox 3.5	N/A	SUPP	N/A	N/A		N/A	
Opera 9.23	N/A	SUPP	N/A	N/A		N/A	
Safari 9.27	N/A	SUPP	N/A	N/A		SUPP	
	"Legacy" Old Browsers						
IE5.5	N/A	N/A	N/A	N/A		N/A	
Netscape	N/A	N/A	N/A	N/A		N/A	

Security Testing of the Project

Testing is vital for the success of any software. no system design is ever perfect. Testing is also carried in two phases. first phase is during the software engineering that is during the module creation. second phase is after the completion of software. this is system testing which verifies that the whole set of programs hanged together.

White Box Testing:

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions or loops. all logical parts of the software checked once. errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decisions on their true and the false side are exercised , all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

Black Box Testing:

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. black box testing tests the input, the output and the external data. it checks whether the input data is correct and whether we are getting the desired output.

Alpha Testing:

Acceptance testing is also sometimes called alpha testing. Be spoke systems are developed for a single customer. The alpha testing proceeds until the system developer and the customer agree that the provided system is an acceptable implementation of the system requirements.

Beta Testing:

On the other hand, when a system isto be marked as a software product, another process called beta testing is often conducted. During beta testing, a system is delivered among a number of potential users who agree to use it. The customers then report problems to the developers. This provides the product for real use and detects errors which may not have been anticipated by the system developers.

Unit Testing:

Each module is considered independently. it focuses on each unit of software as implemented in the source code. it is white box testing.

Integration Testing:

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. modules are integrated by using the top down approach.

Validation Testing:

Validation testing was performed to ensure that all the functional and performance requirements are met.

System Testing:

It is executing programs to check logical changes made in it with intention of finding errors. a system is tested for online response, volume of transaction, recovery from failure etc. System testing is done to ensure that the system satisfies all the user requirements.

Implementation and Software Specification Testings

Detailed Design of Implementation

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures, to evaluate design and operating specifications and/or provide the basis for further modification.

Technical Design

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

Test Specifications and Planning

This activity prepares detailed test specifications for individual modules and programs, job streams, subsystems, and for the system as a whole.

Programming and Testing

This activity encompasses actual development, writing, and testing of program units or modules.

User Training

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

Acceptance Test

A final procedural review to demonstrate a system and secure user approval before a system becomes operational.

Installation Phase

In this phase the new Computerized system is installed, the conversion to new procedures is fully implemented, and the potential of the new system is explored.

System Installation

The process of starting the actual use of a system and training user personnel in its operation.

Review Phase

This phase evaluates the successes and failures during a systems development project, and to measure the results of a new Computerized Transystem in terms of benefits and savings projected at the start of the project.

Development Recap

A review of a project immediately after completion to find successes and potential problems in future work.

Post-Implementation Review

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

THE STEPS IN THE SOFTWARE TESTING

The steps involved during Unit testing are as follows:

a. Preparation of the test cases.
b. Preparation of the possible test data with all the validation checks.
c. Complete code review of the module.
d. Actual testing done manually.

e. Modifications done for the errors found during testing.
f. Prepared the test result scripts.

The unit testing done included the testing of the following items:

1. Functionality of the entire module/forms.
2. Validations for user input.
3. Checking of the Coding standards to be maintained during coding.
4. Testing the module with all the possible test data.
5. Testing of the functionality involving all type of calculations etc.
6. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, We integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

The steps involved during System testing are as follows:

- Integration of all the modules/forms in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.
- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.

The System Testing done included the testing of the following items:

1. Functionality of the entire system as a whole.
2. User Interface of the system.
3. Testing the dependent modules together with all the possible test data scripts.
4. Verification and Validation testing.
5. Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing. Clients at their end did this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery.

There are other six tests, which fall under special category. They are described below:

- Peak Load Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.
- Storage Testing: It determines the capacity of the system to store transaction data on a disk or in other files.
- Performance Time Testing: it determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.
- Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.
- Procedure Testing: It determines the clarity of documentation on operation and uses of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer.
- Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

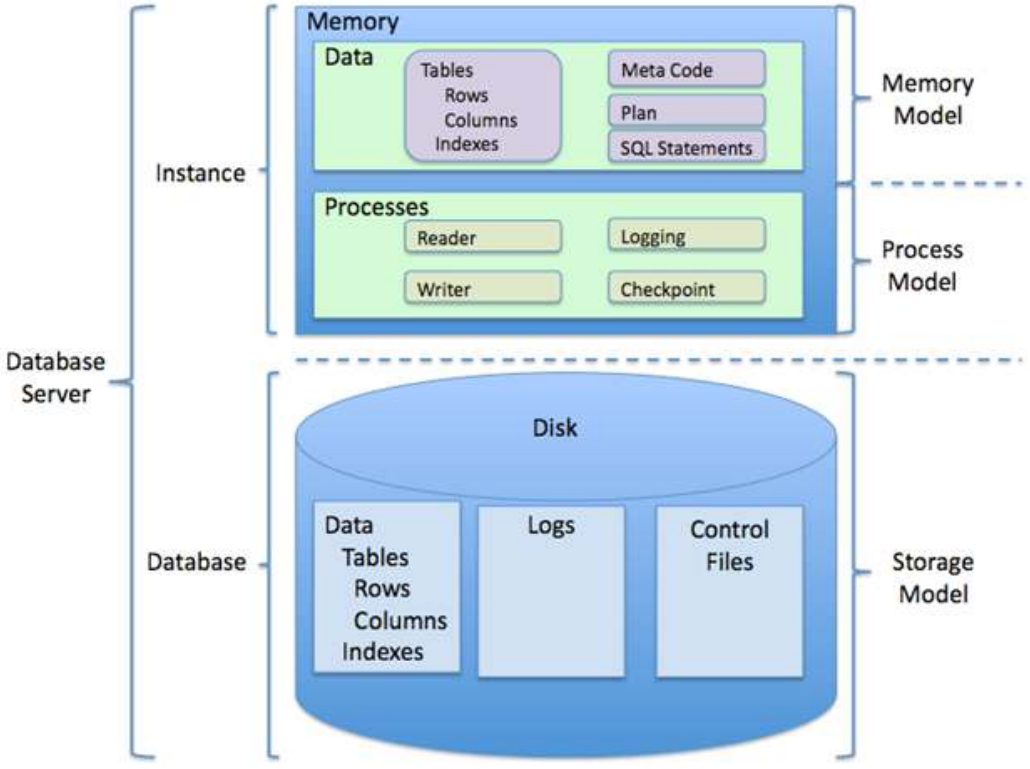
Project Category

Relational Database Management System (RDBMS) : This is an RDBMS based project which is currently using MySQL for all the transaction statements. MySQL is an open source RDBMS System.

Brief Introduction about RDBSM :

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

RDBMSs have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and much more since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have been challenged by object databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational database, and XML databases.



Implementation Methodology:

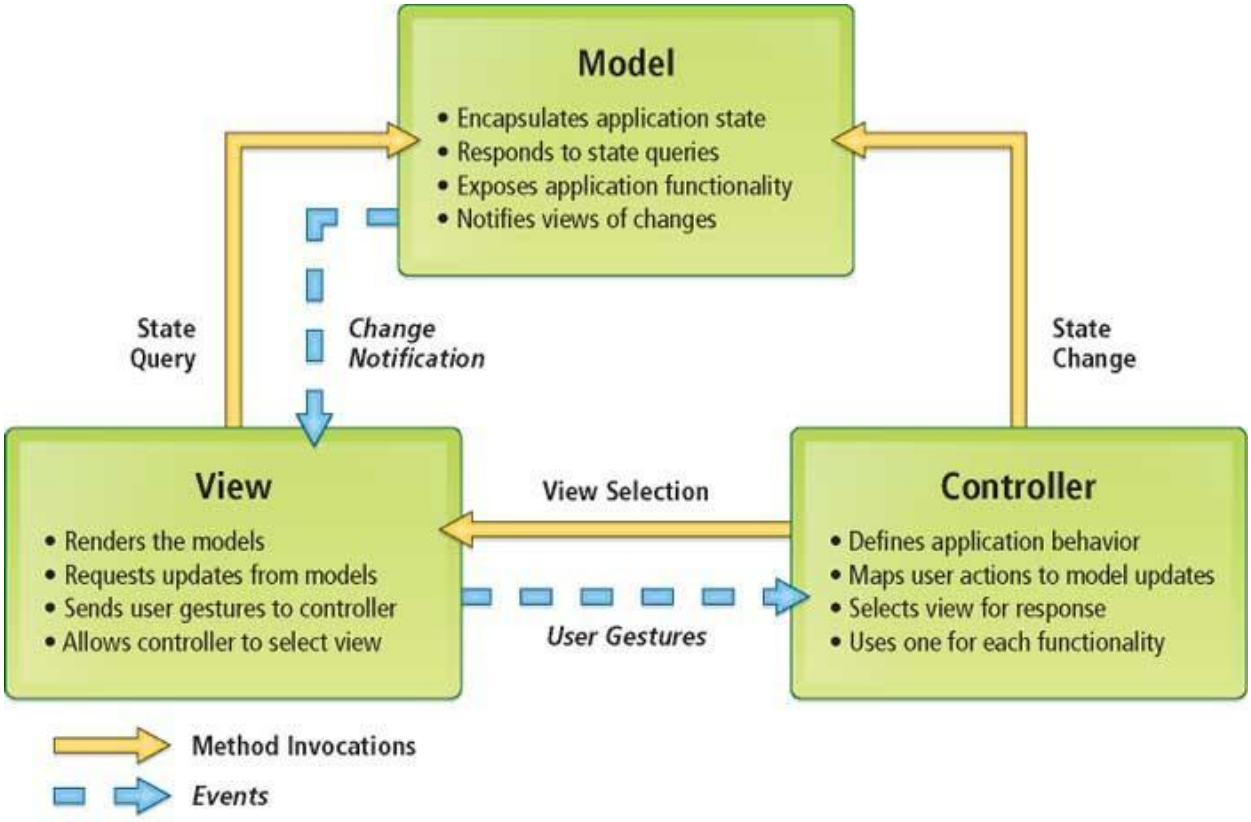
Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

- **Model** - The lowest level of the pattern which is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of the data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

MVC (Model View Controller Flow) Diagram

PROJECT SCHEDULING



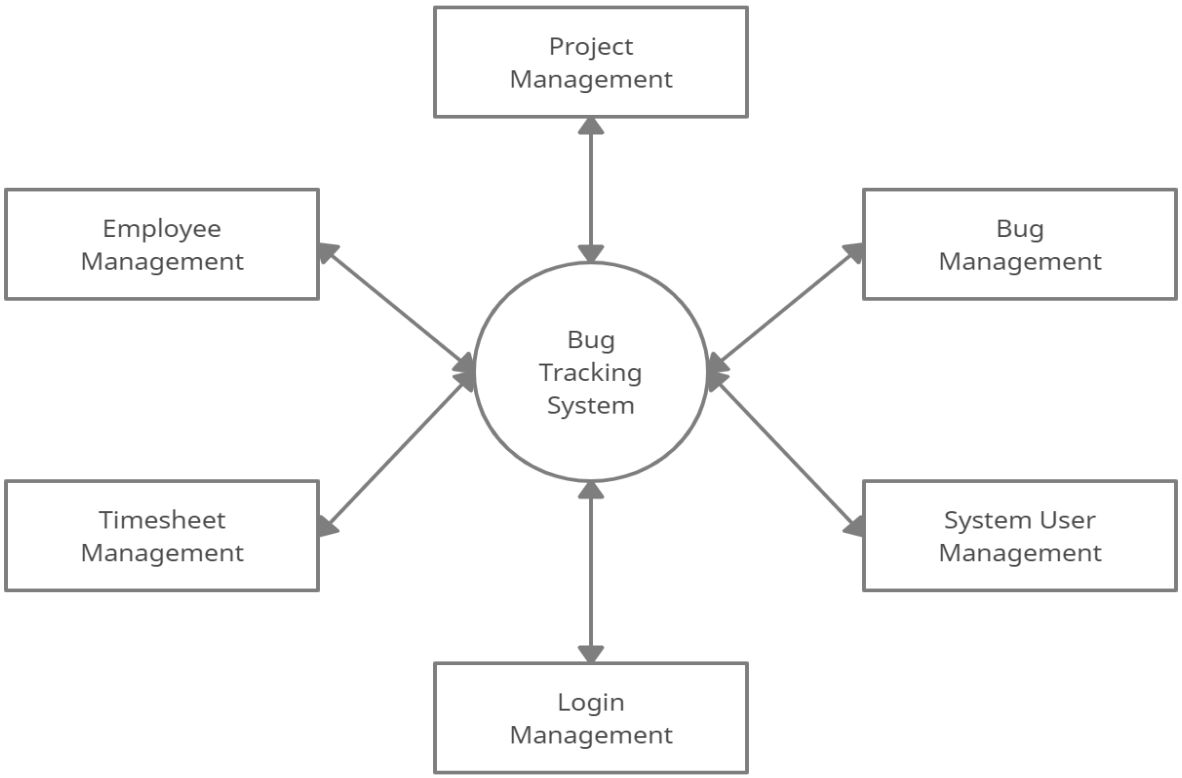
An elementary Gantt chart or Timeline chart for the development plan is given below. The plan explains the tasks versus the time (in weeks) they will take to complete.

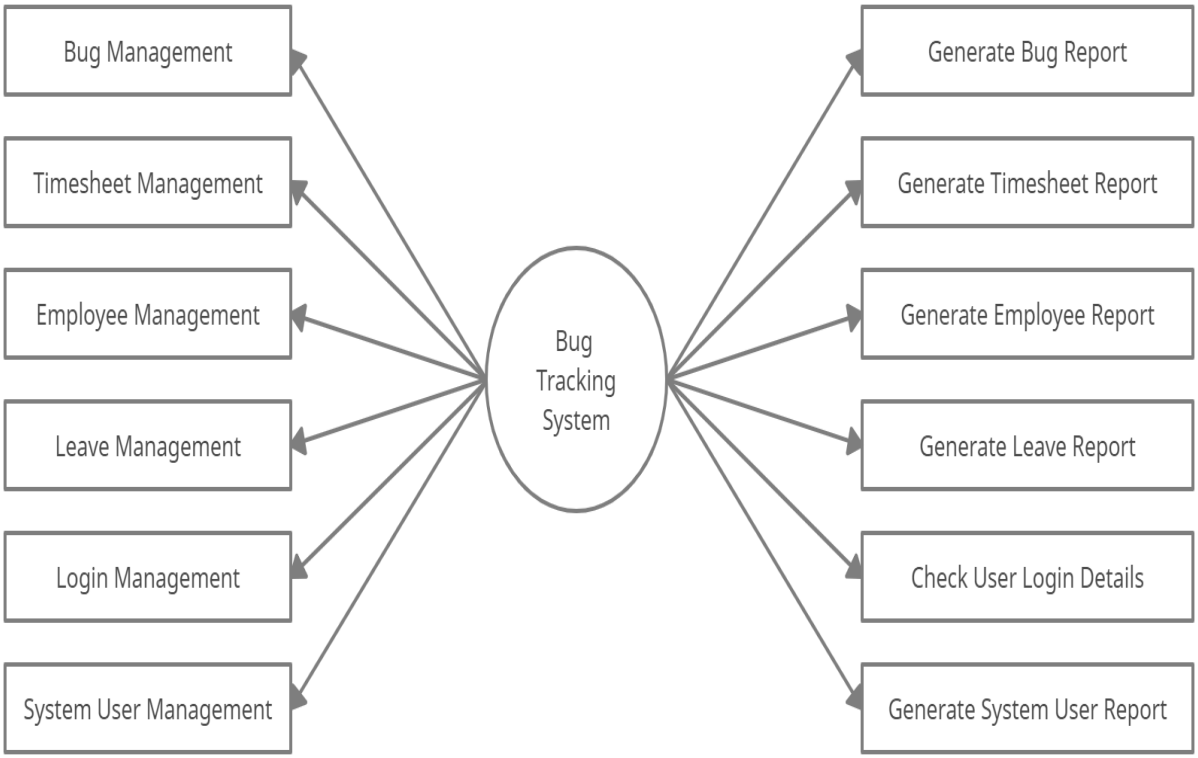
	January				February				March			
Requirement Gathering												
Analysis												
Design												
Coding												
Testing												
Implement												
	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4

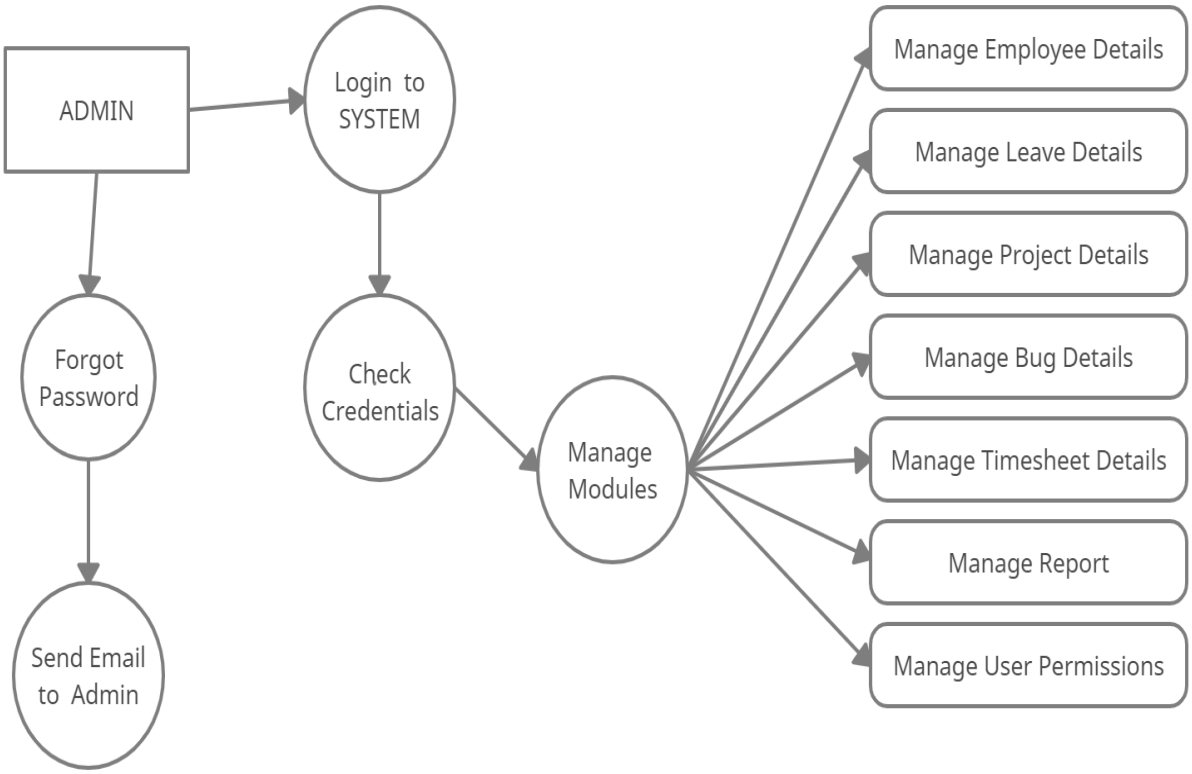
W*i*'s are weeks of the months, for i =1, 2, 3, 4

USE CASE DIAGRAMS

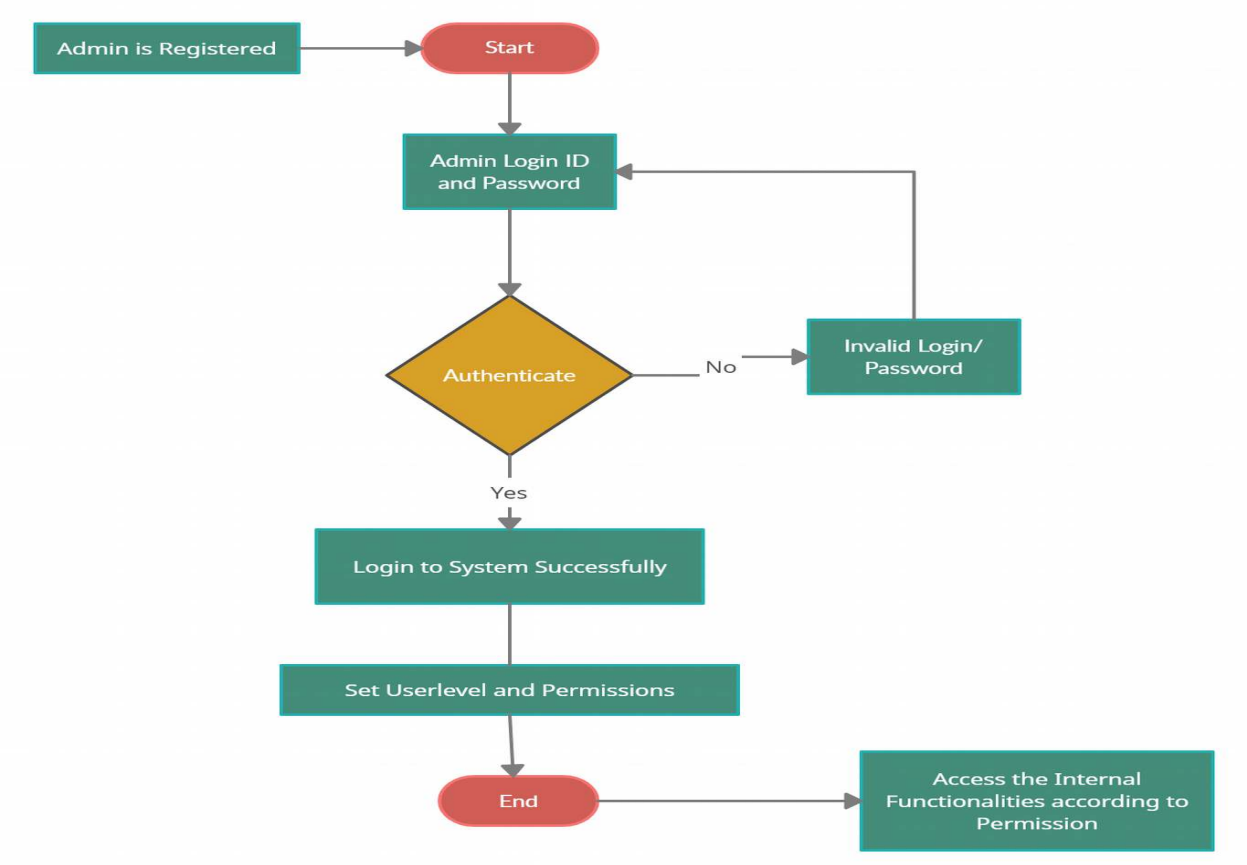
DFD DIAGRAMS







FLOWCHART



Identification of Need

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records. In present, work done in the electricity board is performed manually which is a great headache for the department .The reason behind it is that there is lot of information to be maintained and have to be kept in mind while running the business .For this reason we have provided features Present system is partially automated (computerized), actually existing system is quite laborious as one has to enter same information at three different places.

Following points should be well considered:

- Documents and reports that must be provided by the new system: there can also be few reports, which can help management in decision-making and cost controlling, but since these reports do not get required attention, such kind of reports and information were also identified and given required attention.
- Details of the information needed for each document and report.
- The required frequency and distribution for each document.
- Probable sources of information for each document and report.
- With the implementation of computerized system, the task of keeping records in an organized manner will be solved. The greatest of all is the retrieval of information, which will be at the click of the mouse. So the proposed system helps in saving the time in different operations and making information flow easy giving valuable reports.

DATA DICTIONARY:

This is normally represented as the data about data. It is also termed as metadata some times which gives the data about the data stored in the database. It defines each data term encountered during the analysis and design of a new system. Data elements can describe files or the processes.

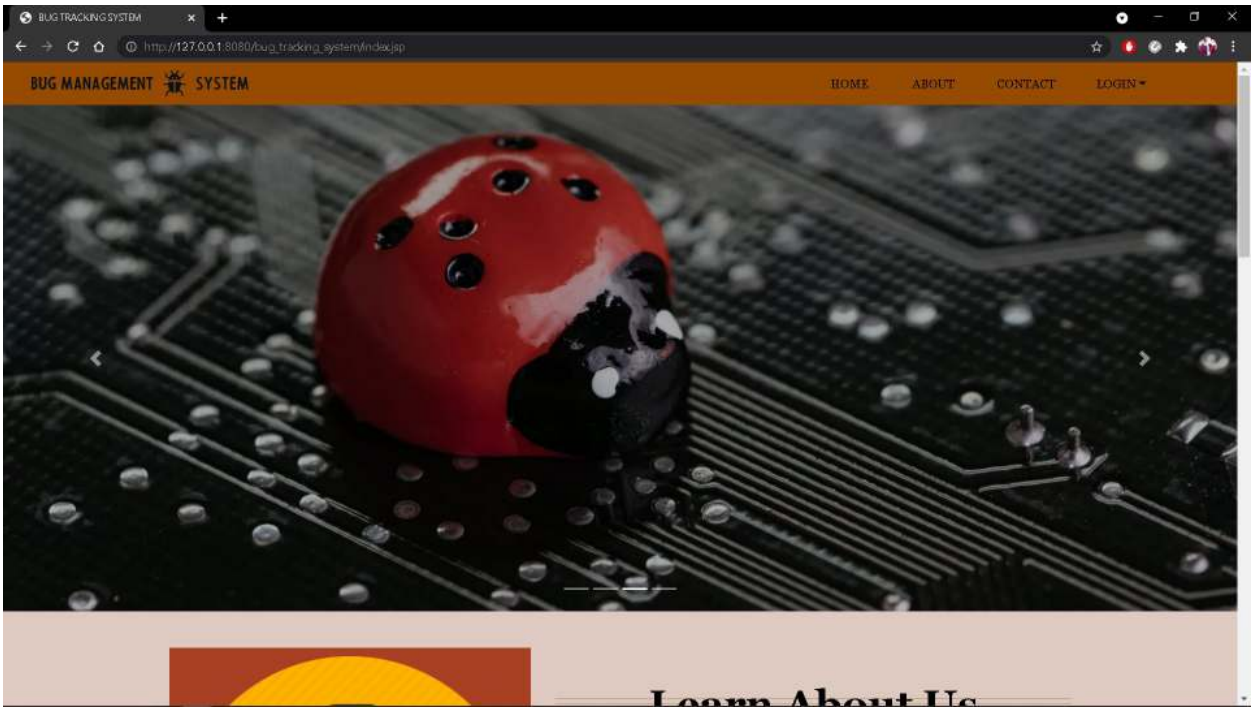
Following are some major symbols used in the data dictionary

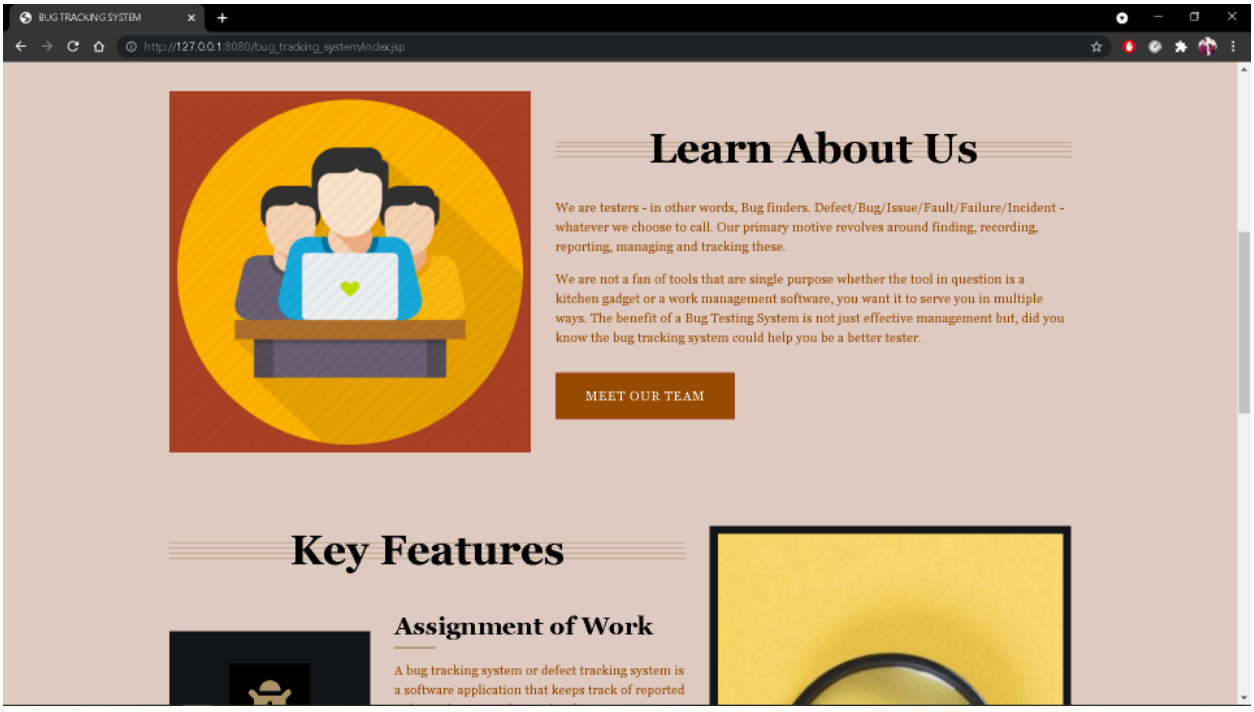
- = equivalent to
- + and
- [] either/ or
- () Optional entry

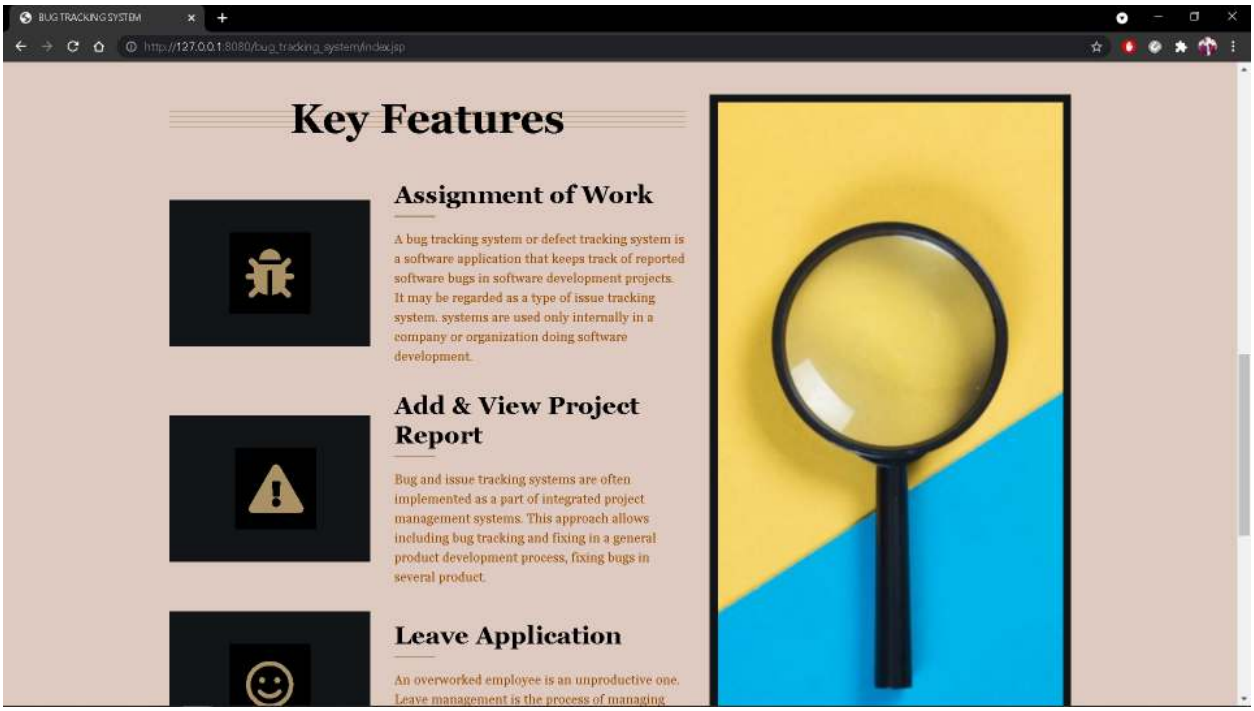
Following are some rules, which defines the construction of data dictionary entries:

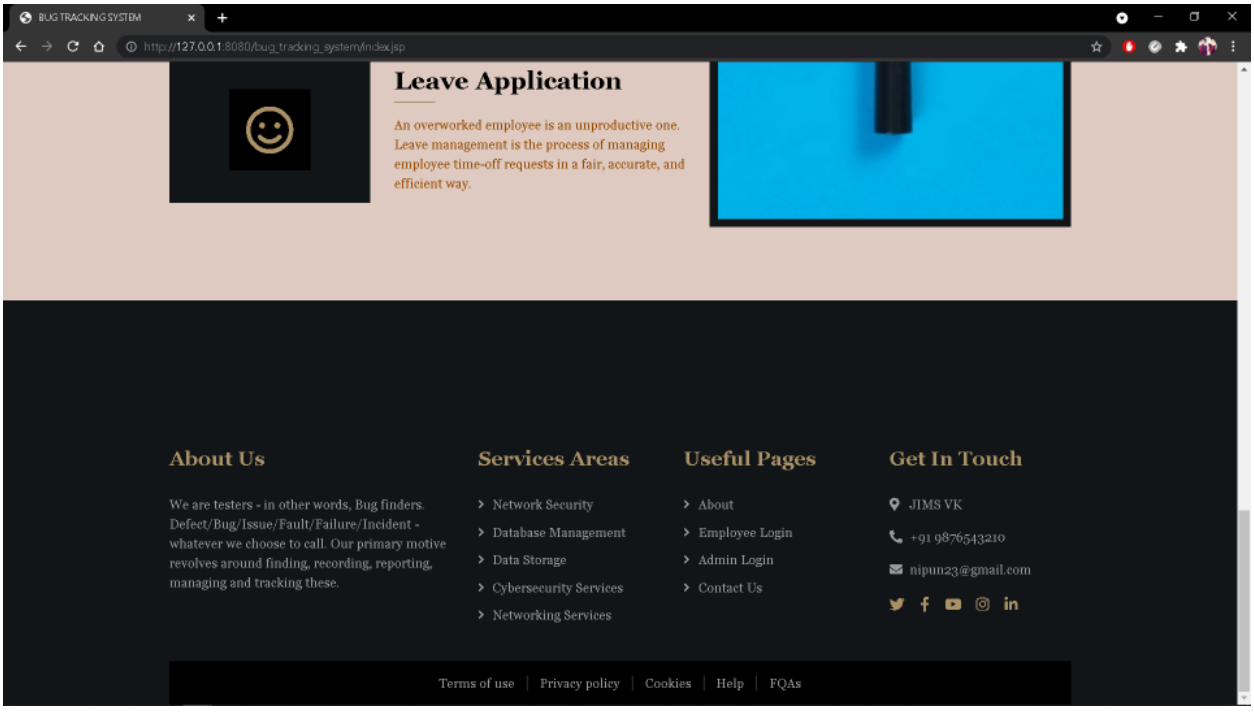
1. Words should be defined to understand for what they need and not the variable need by which they may be described in the program .
2. Each word must be unique. We cannot have two definition of the same client.
3. Aliases or synonyms are allowed when two or more enters shows the same meaning. For example a vendor number may also be called as customer number.
4. A self-defining word should not be decomposed. It means that the reduction of any information in to subpart should be done only if it is really required that is it is not easy to understand directly.

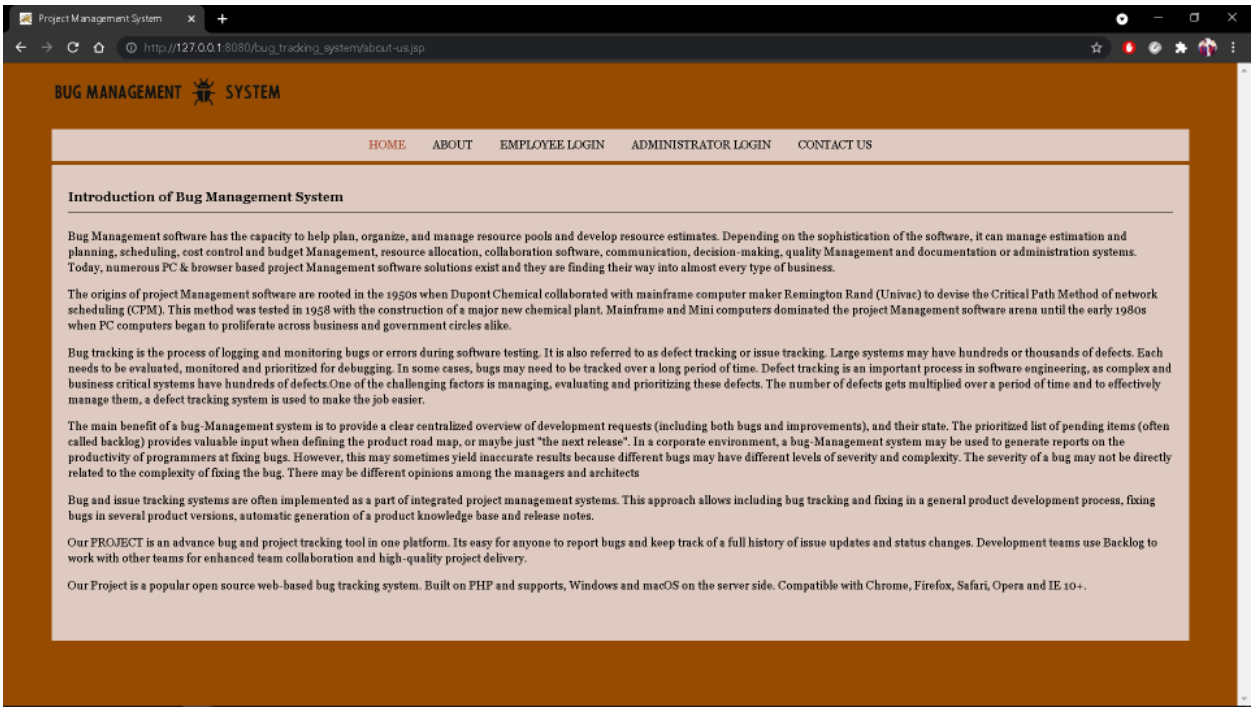
Data dictionary includes information such as the number of records in file, the frequency a process will run, security factor like pass word which user must enter to get excess to the information.

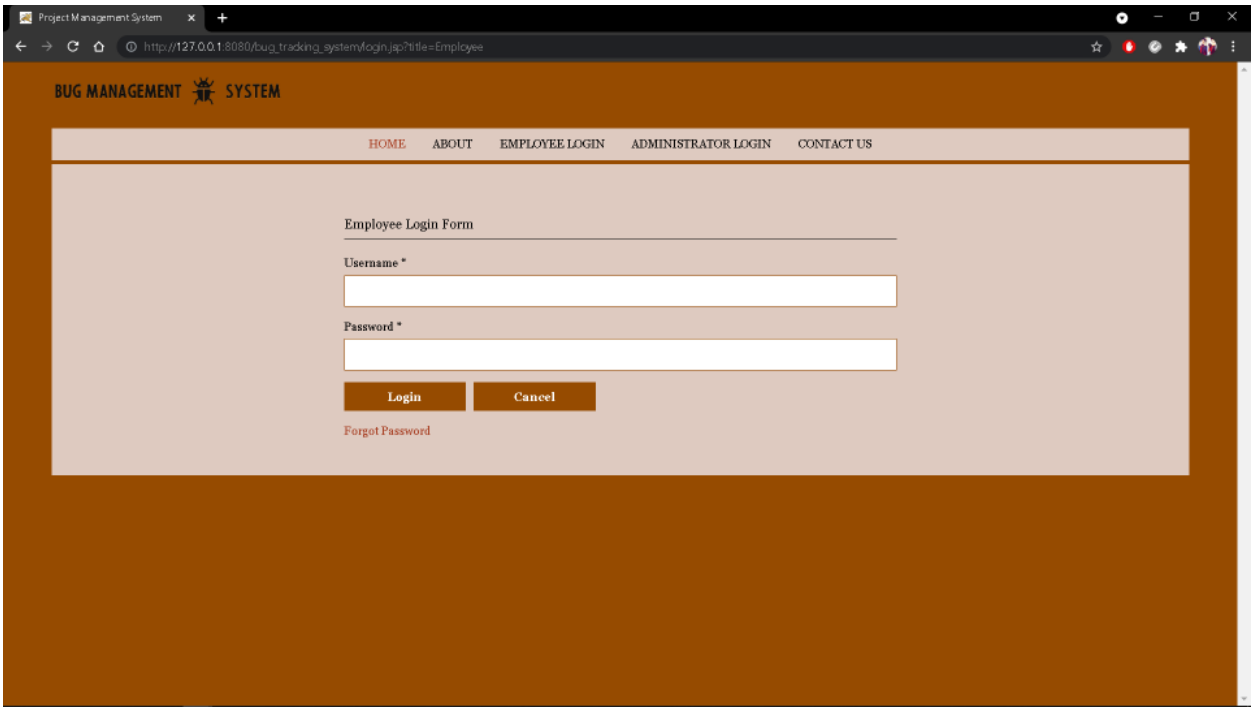


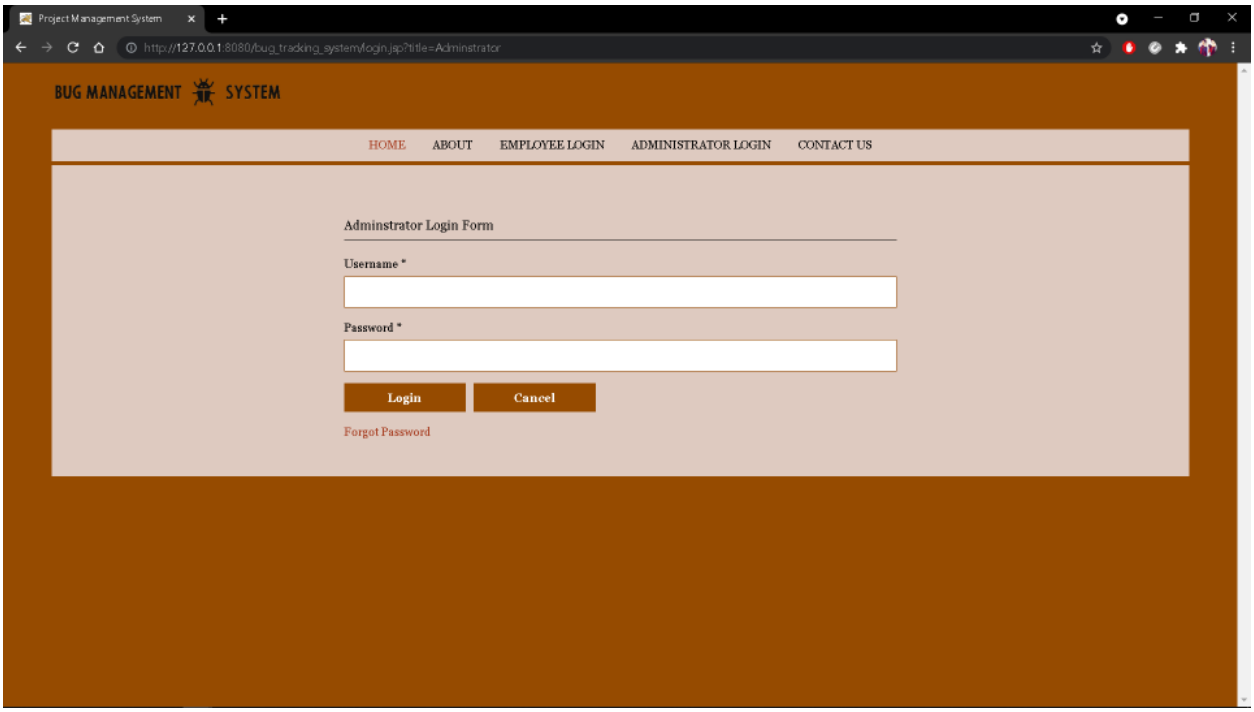


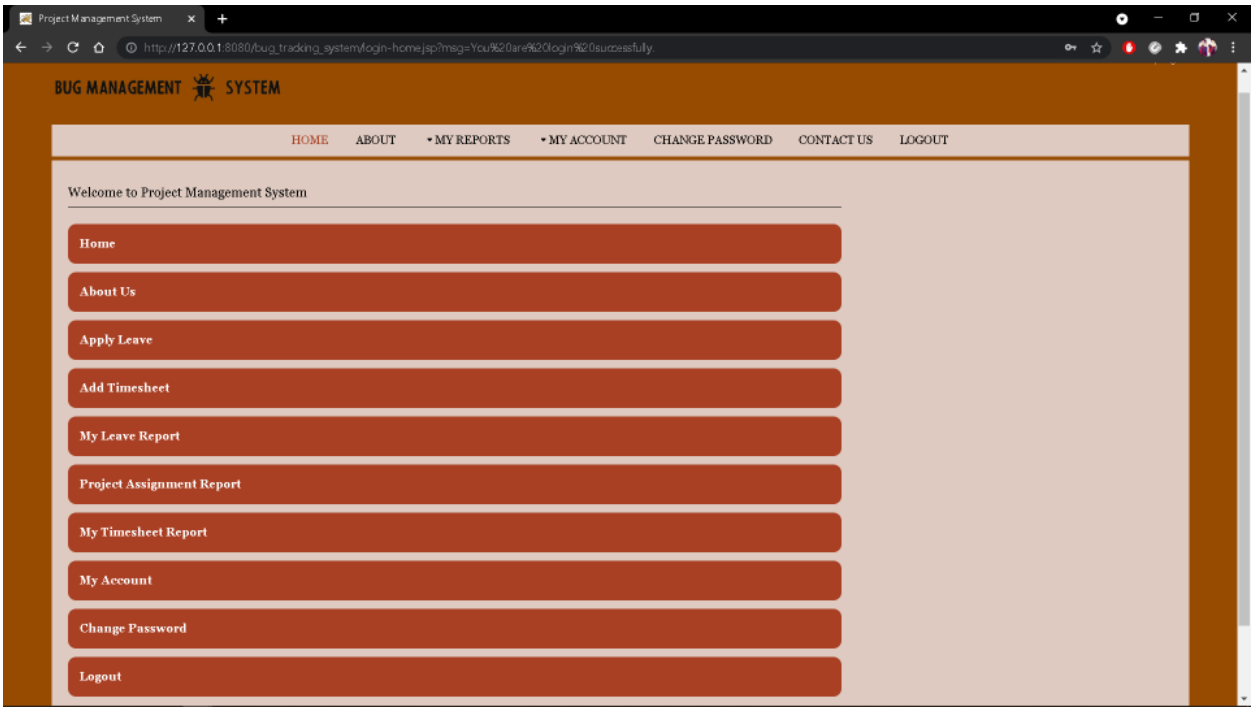


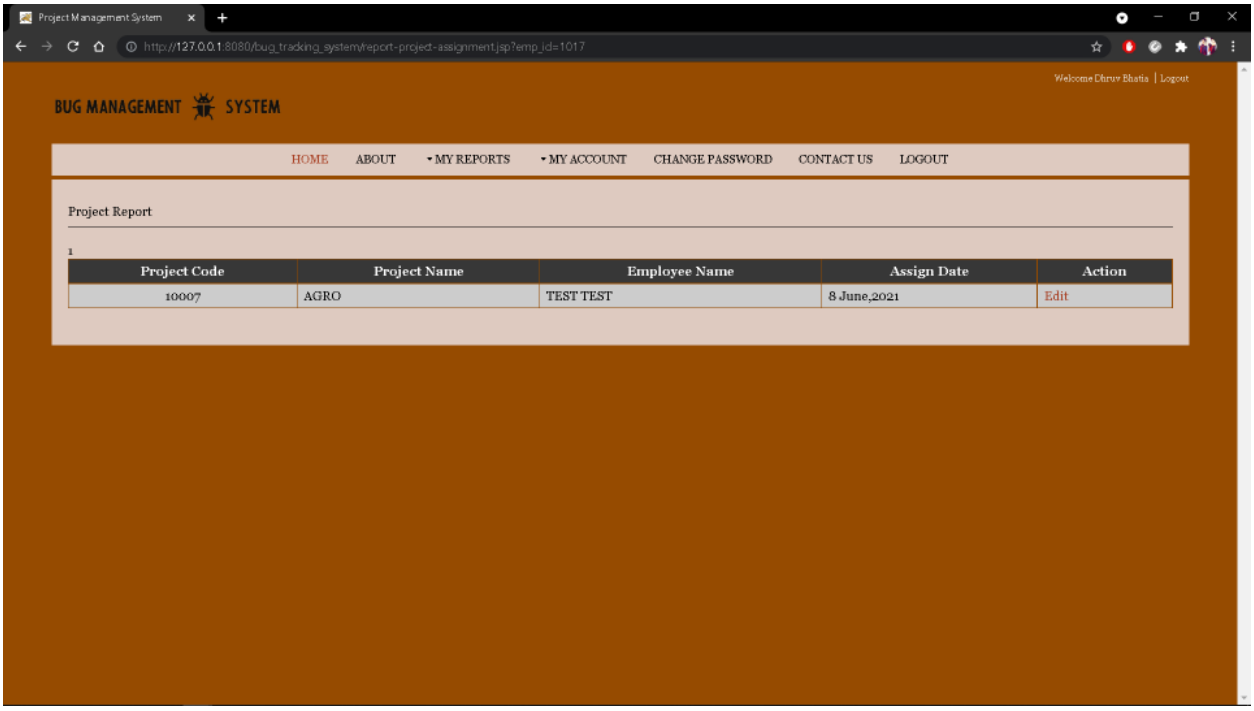


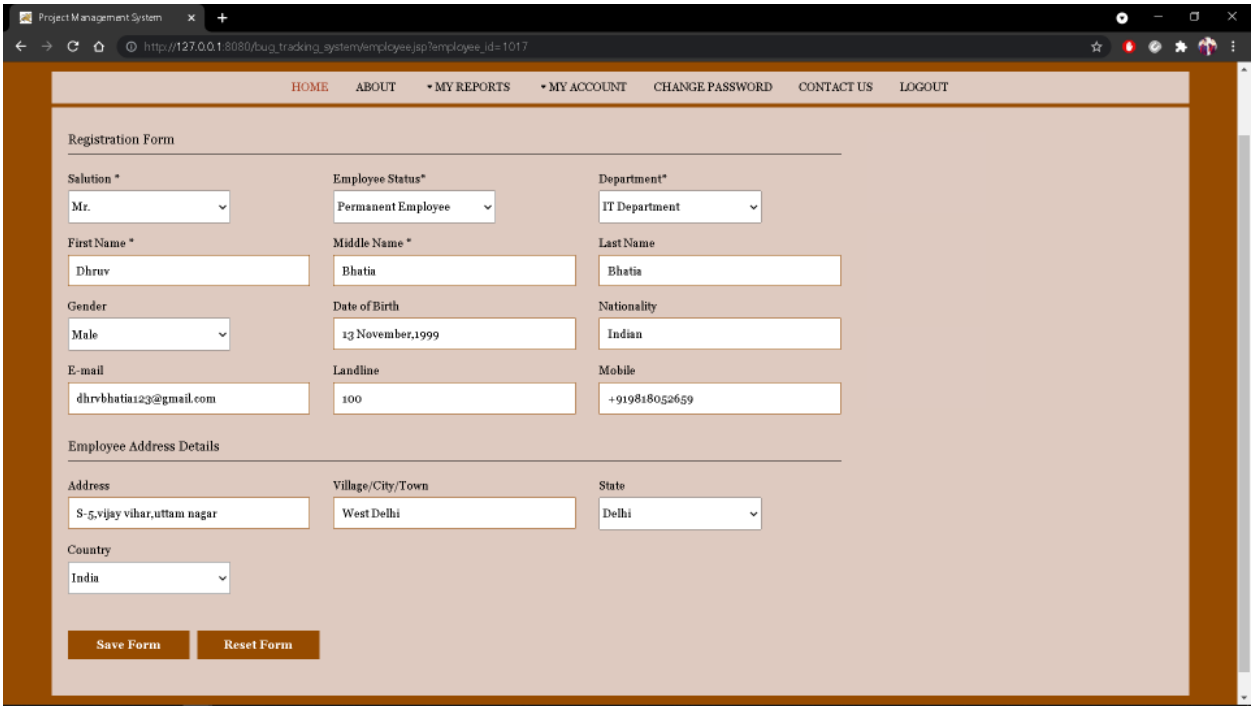


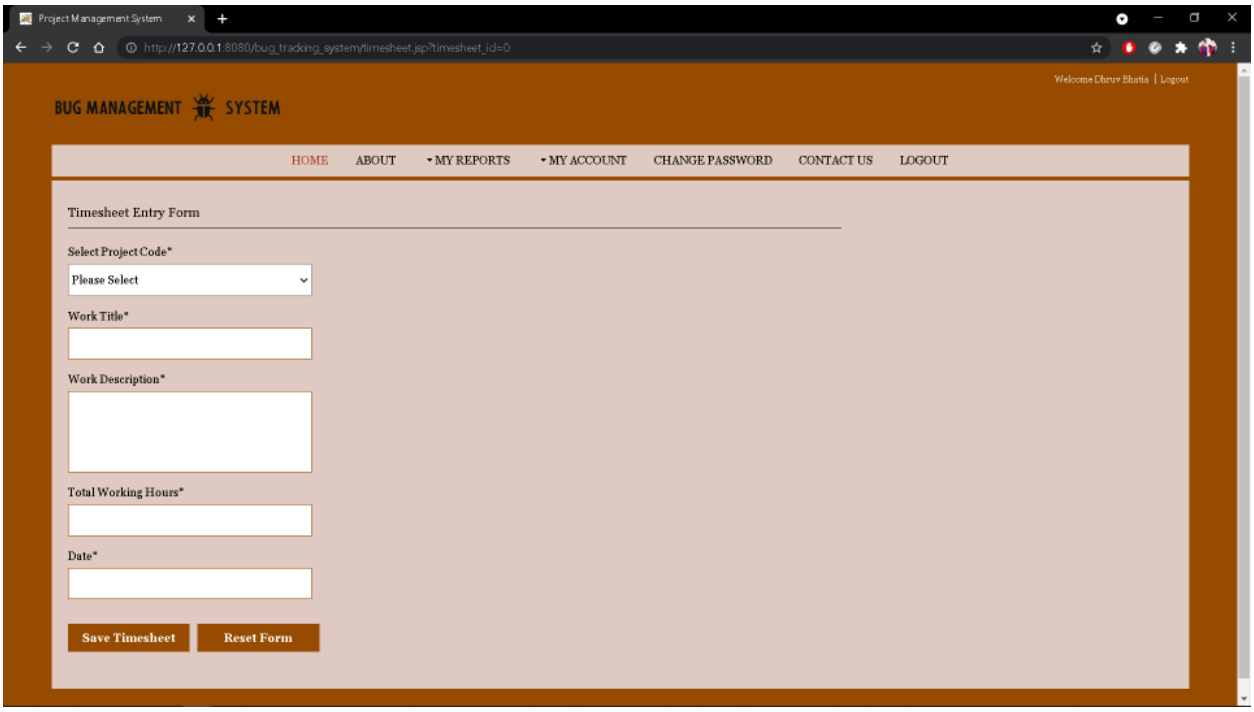


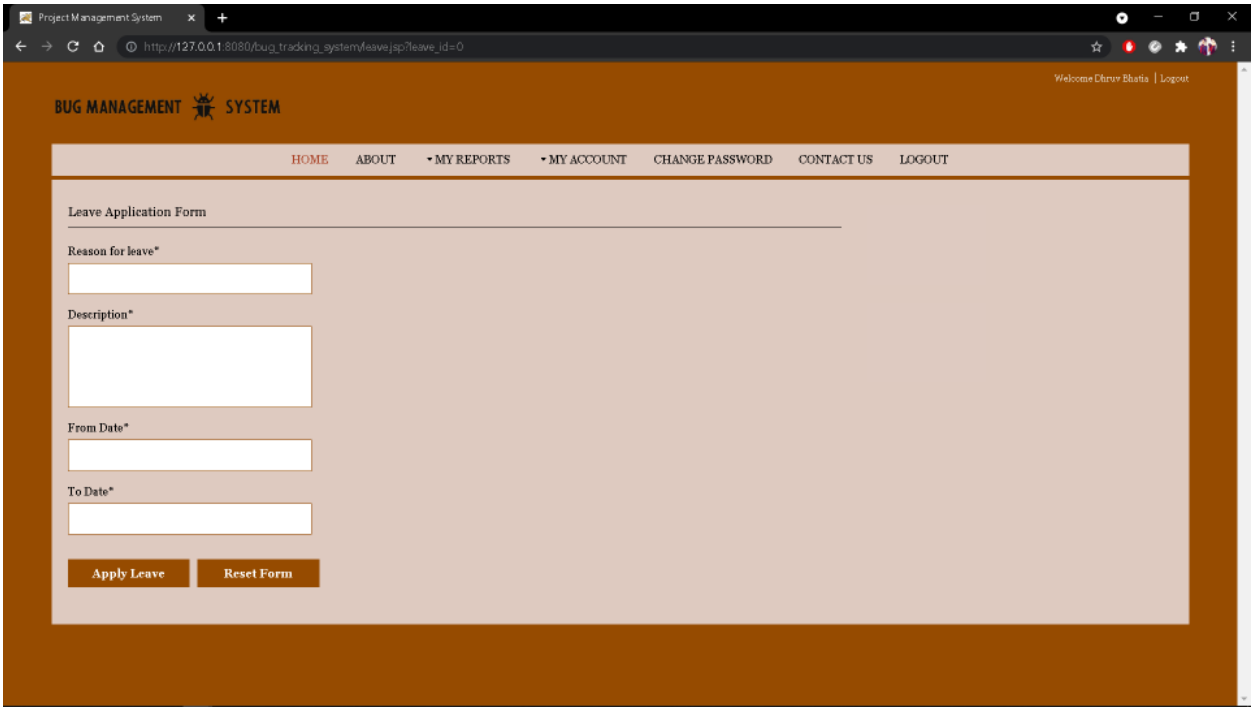


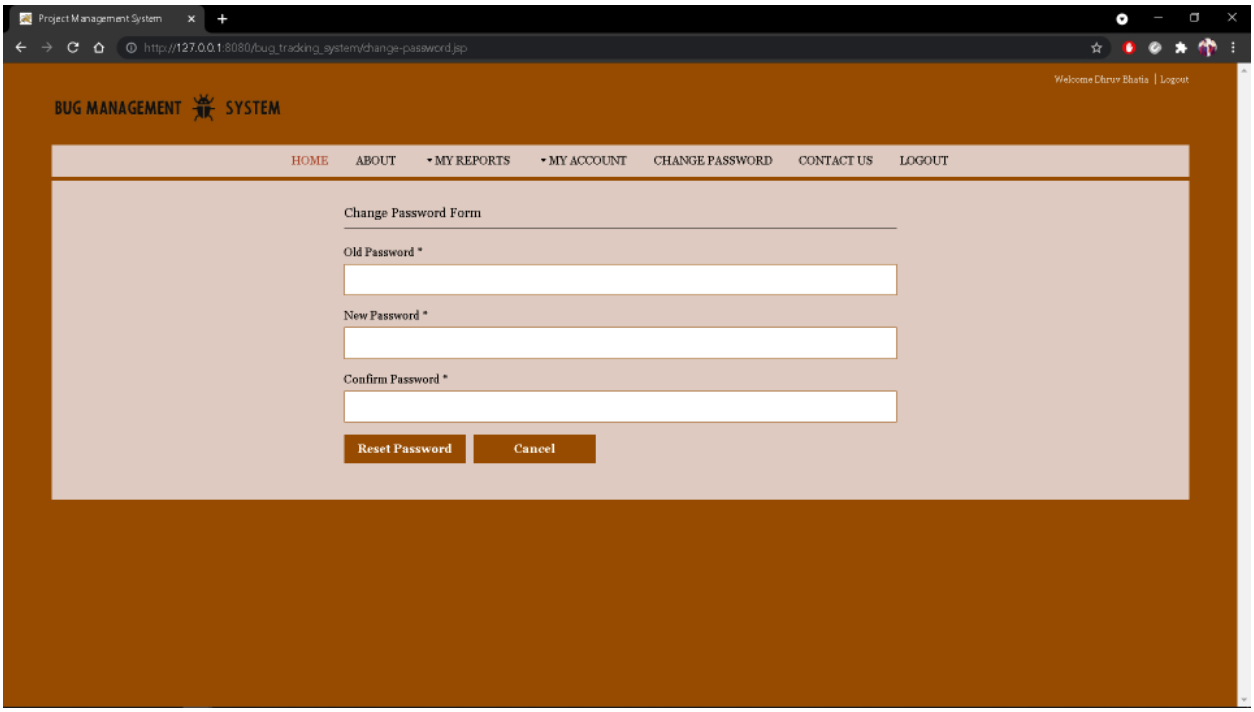


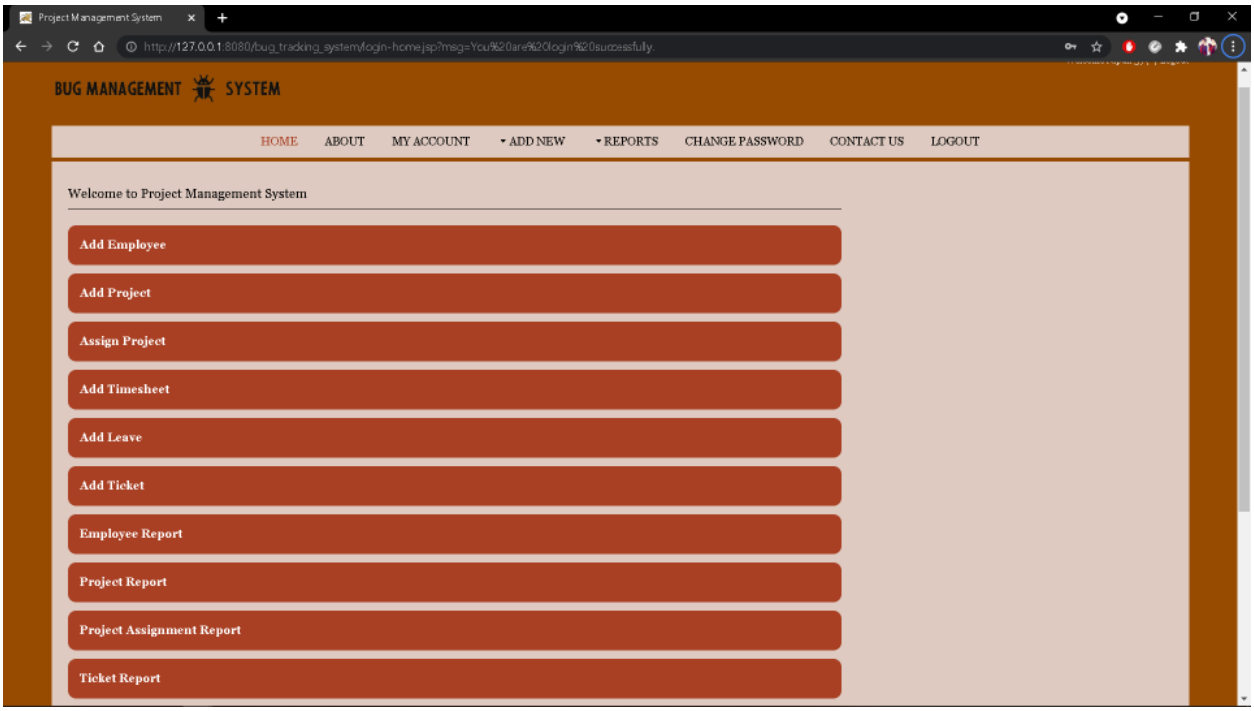


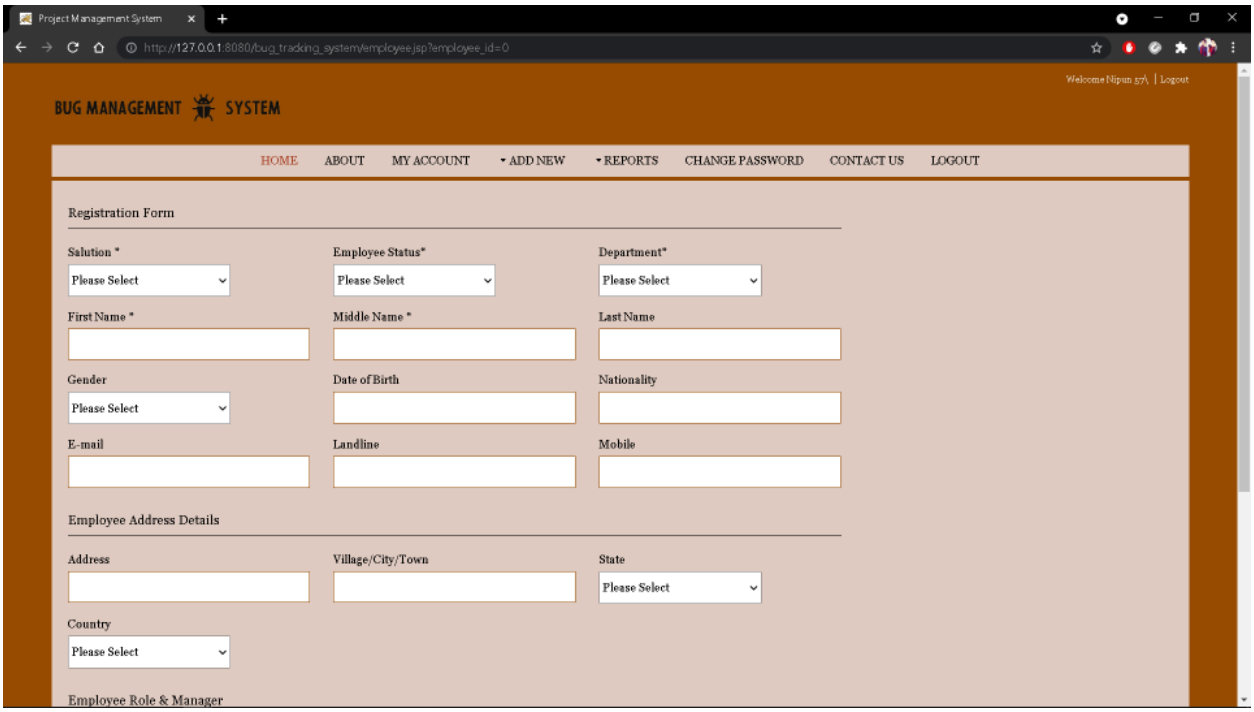


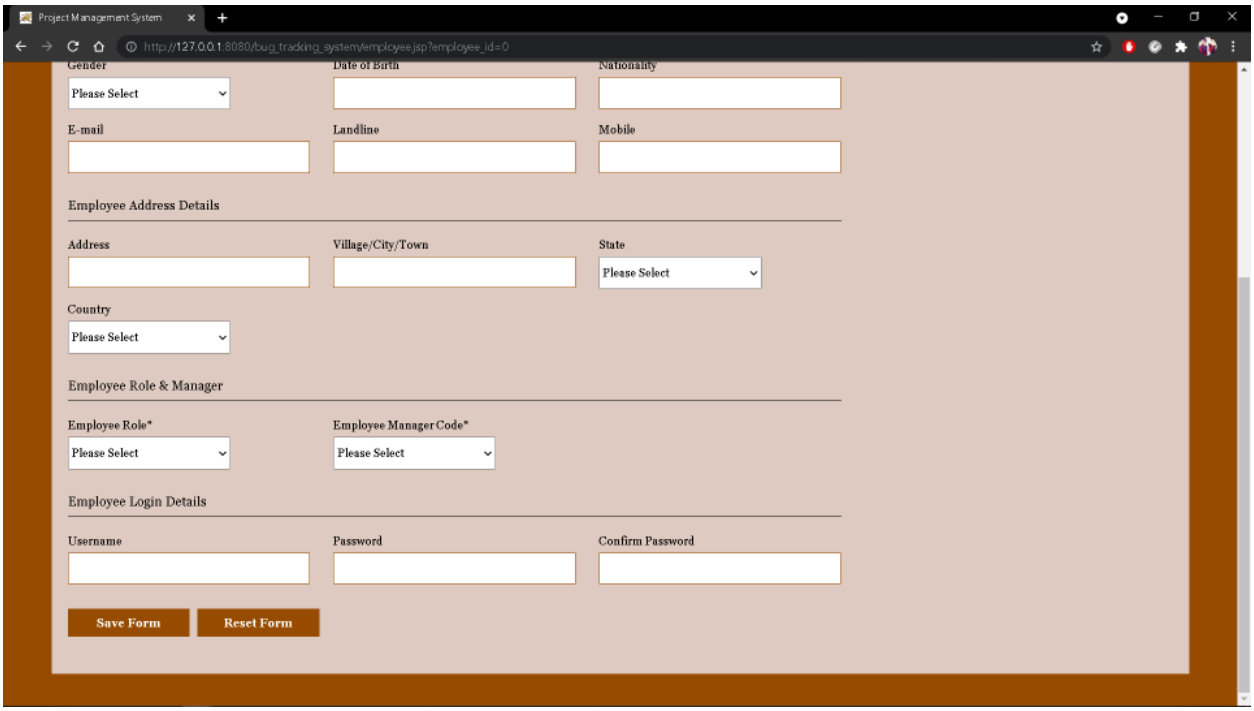


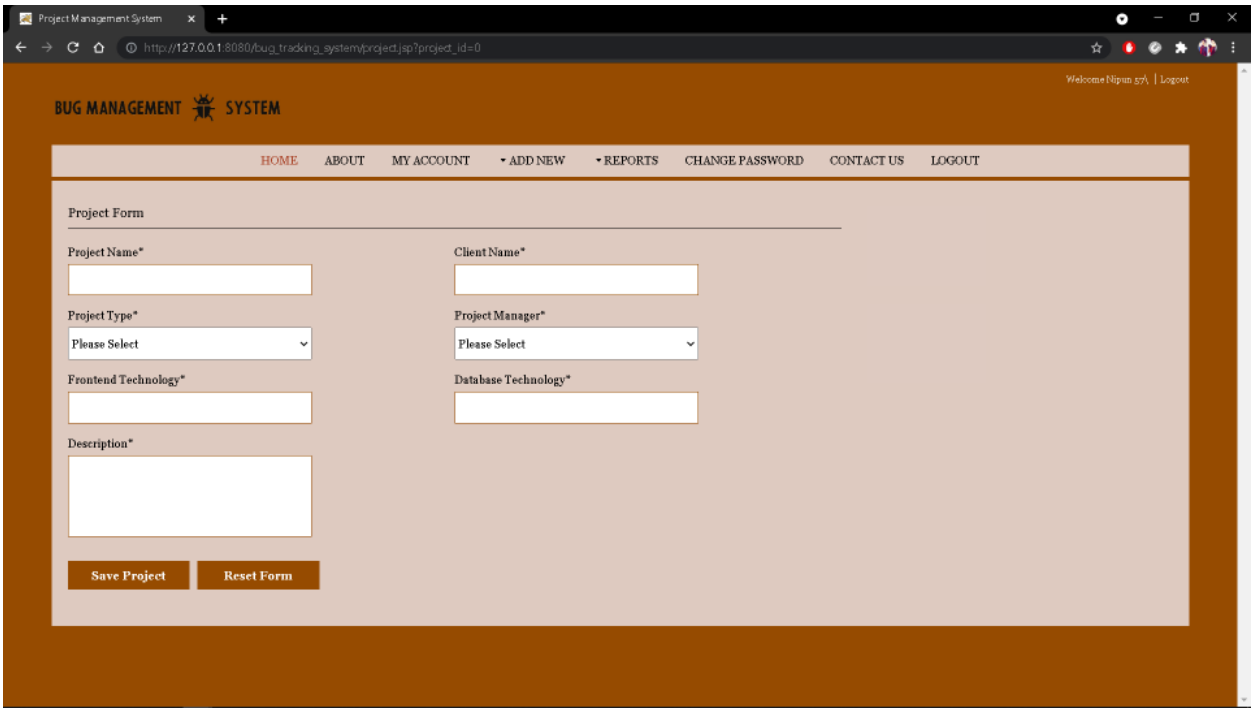


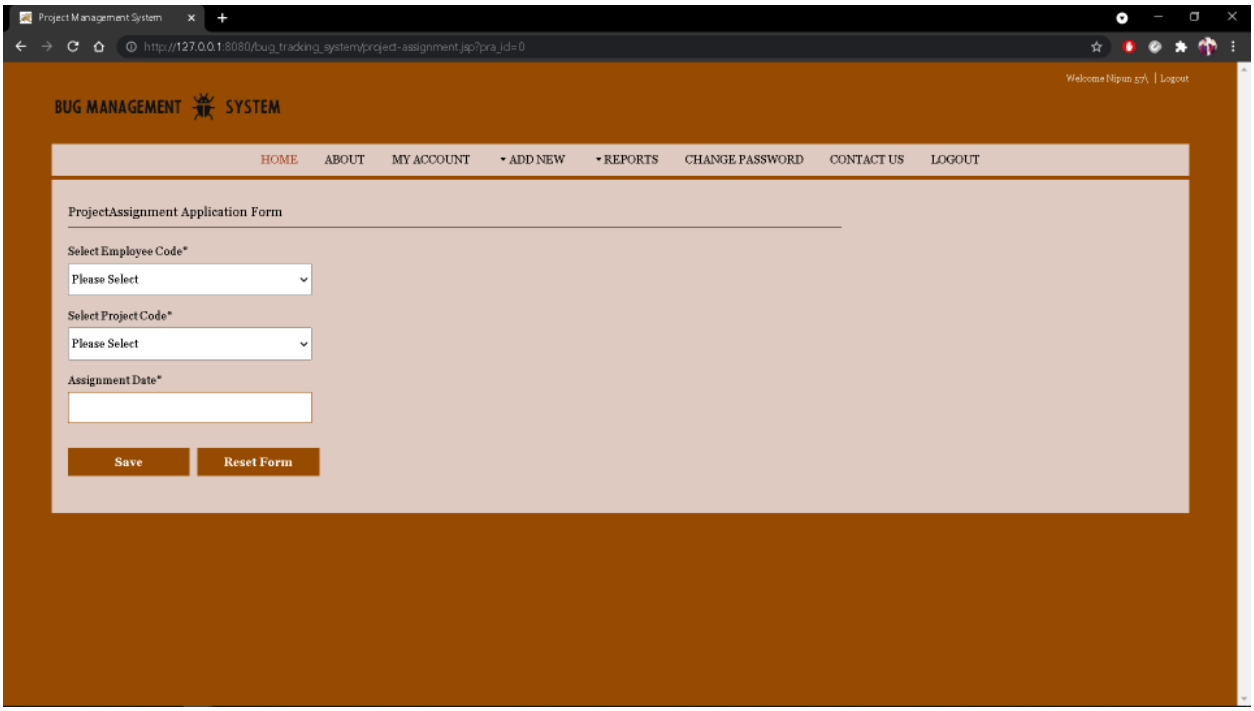


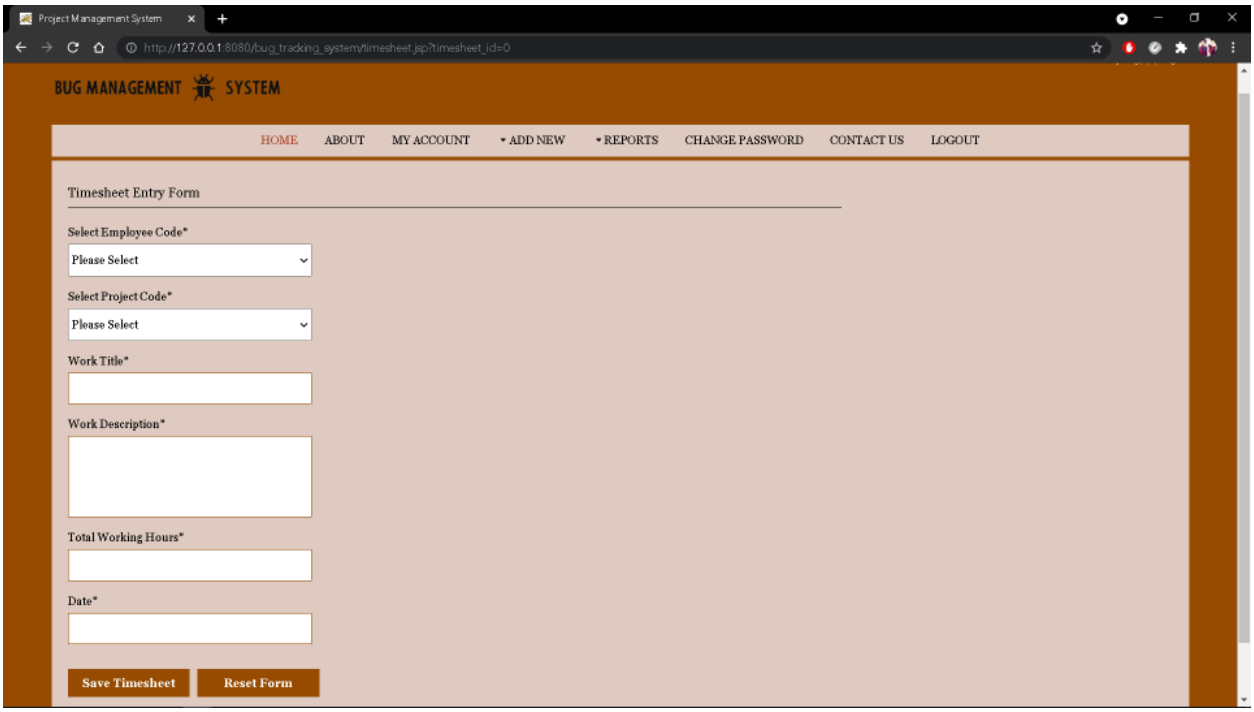


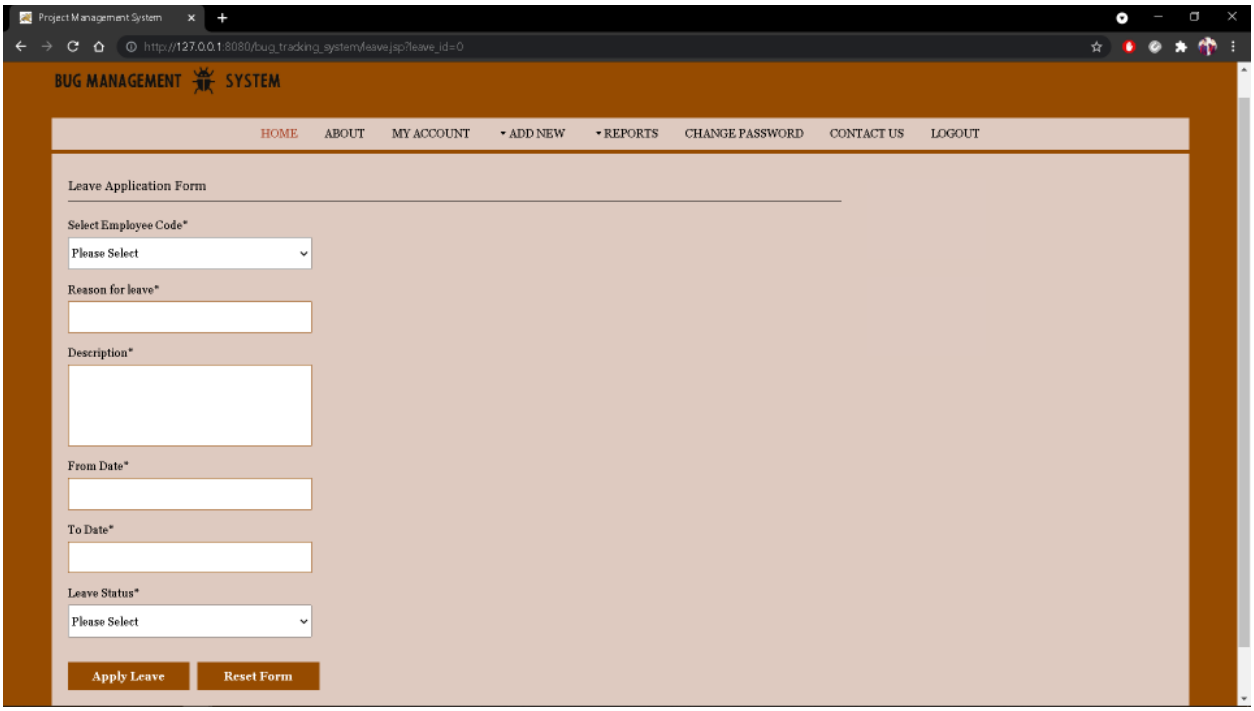


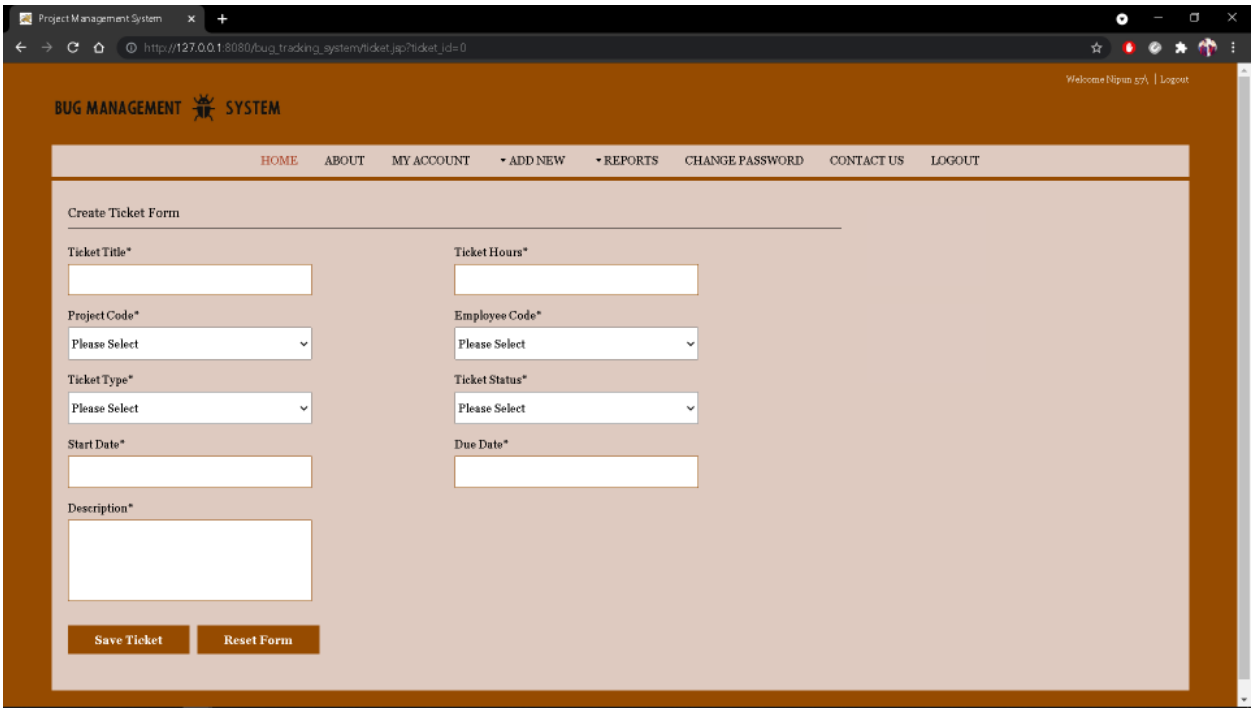


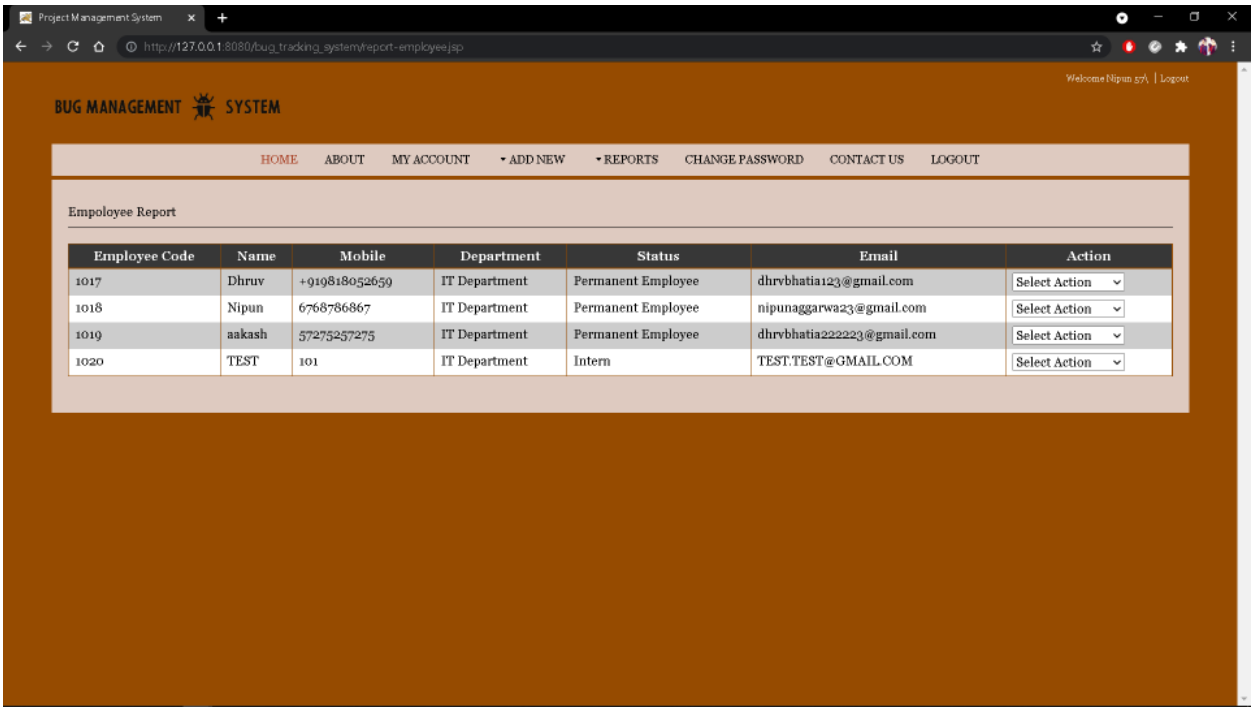


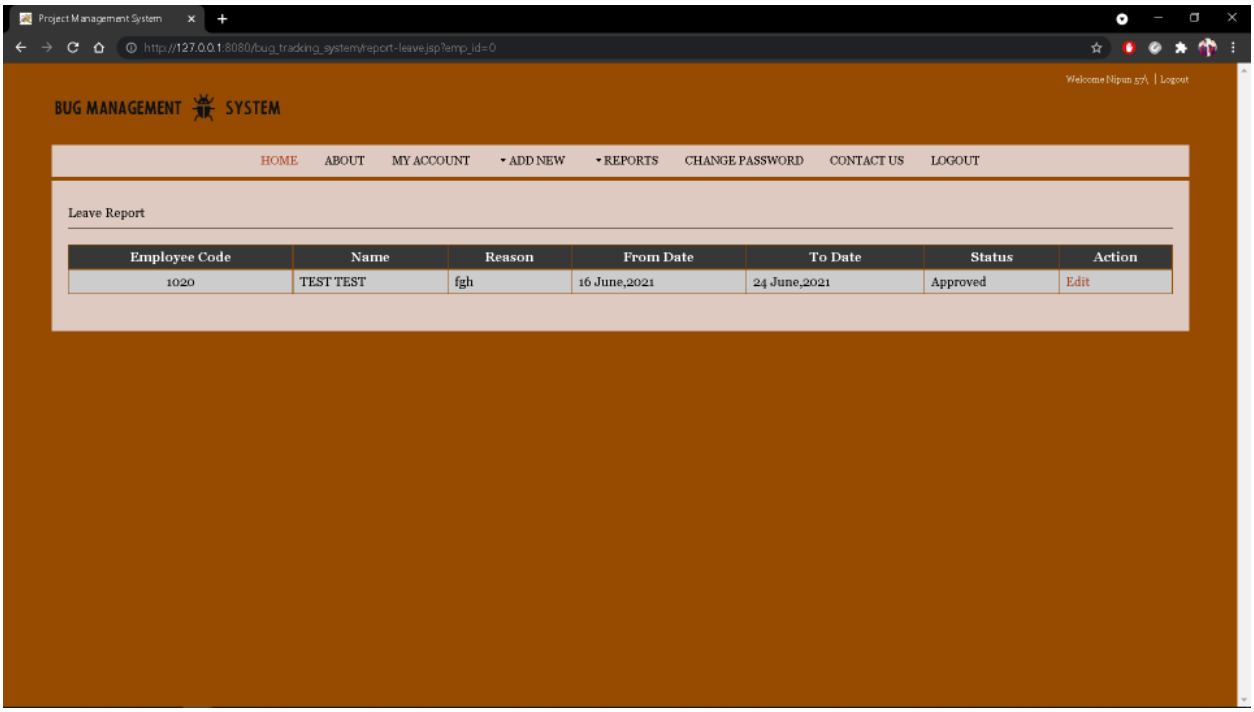


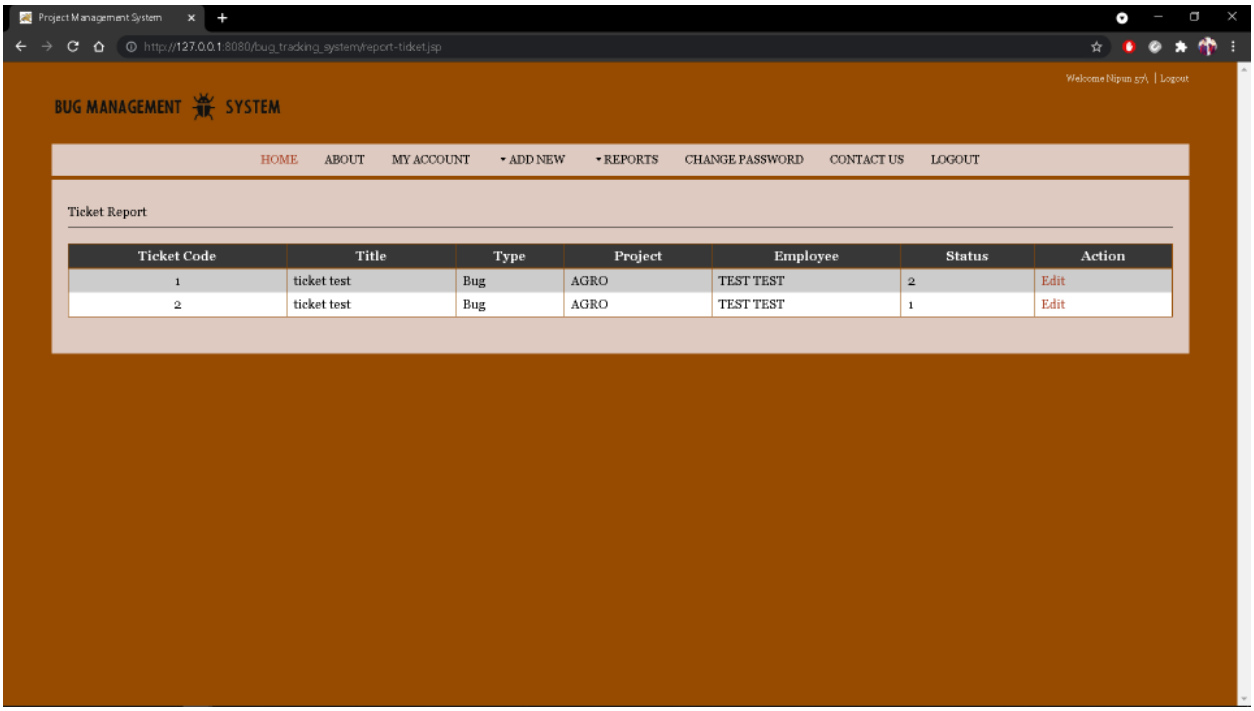


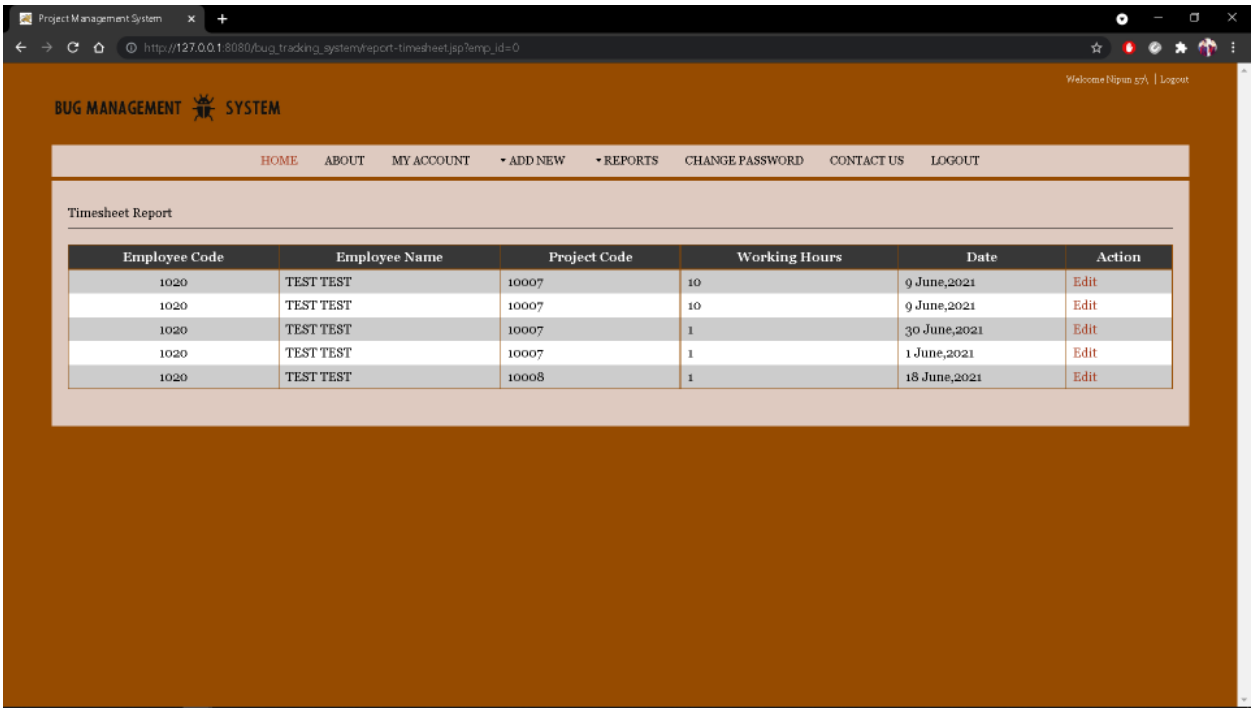


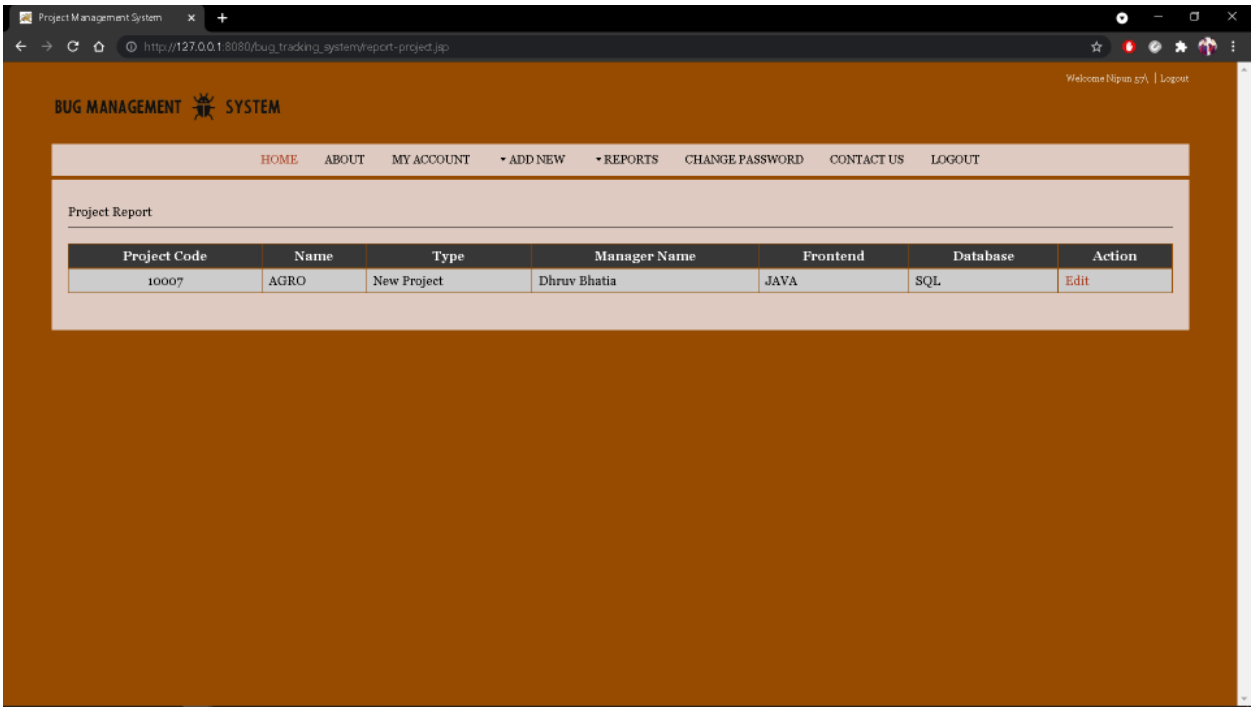


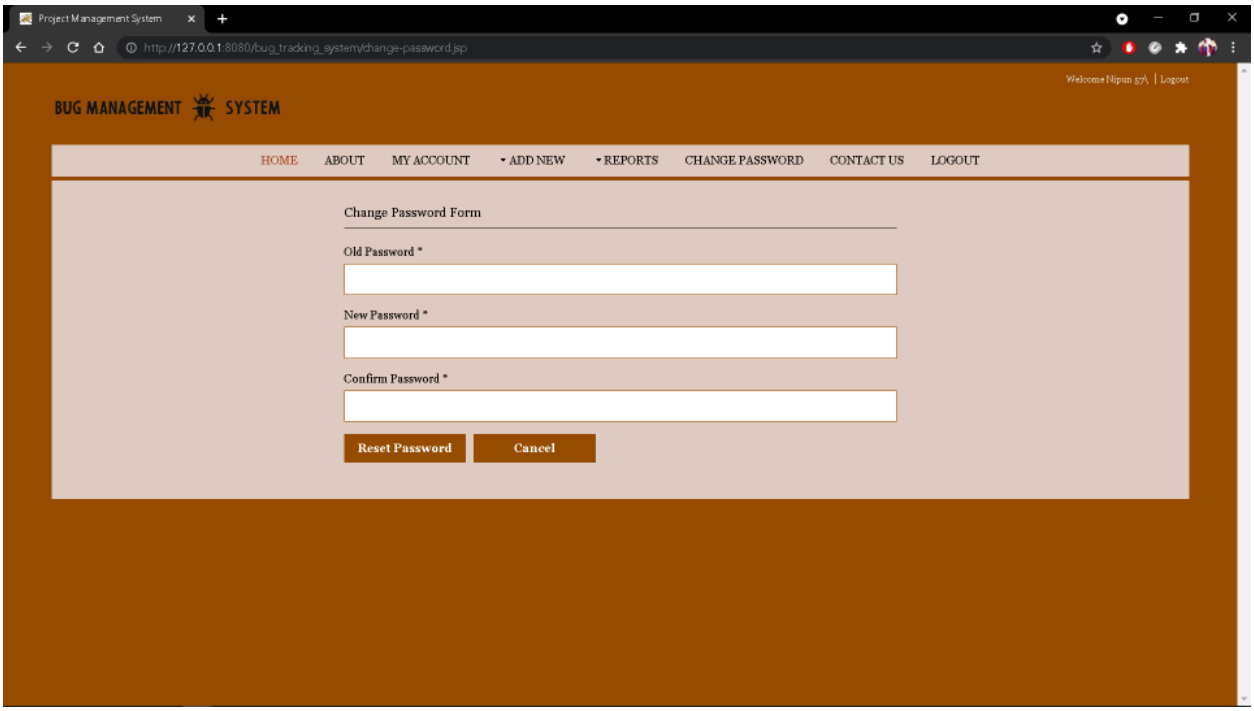


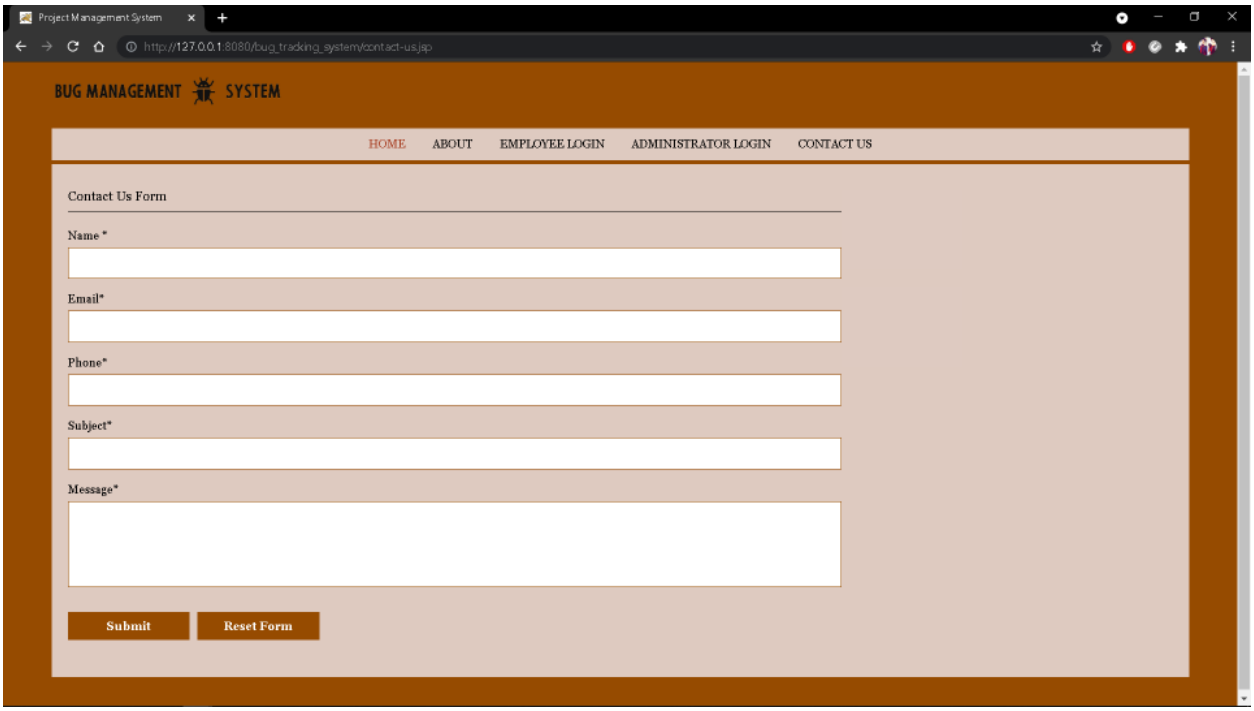












Coding of DB Connection

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

public class Connect
{
    public static Statement statement = null;
    public static Connection connection;
        public static ResultSet rs;
        public static PreparedStatement pstmt;
    /////Function for connect to the MySQL Server Database//////////
        public static void connect_mysql()
    {
        try
        {
```

```
                Class.forName("com.mysql.jdbc.Driver").newInstance();
            connection =
DriverManager.getConnection("jdbc:mysql://localhost/bug_tracking_system?" +
"user=root&password=root");
                statement=connection.createStatement();
            }
            catch(Exception e)
      {
                System.out.println(" Error : "+ e.toString());
      }
   }
      ////////Function for geting the Option//////////////////
      public static String getOptionList(String tableName,String idColumn,String
valueColumn,String Columns,int selID,String conn)
      {
            String SQL = "SELECT "+Columns+" FROM "+tableName+" where
"+conn;
            String Option="<option value=''>Please Select</option>";
            try
            {
                rs = statement.executeQuery(SQL);
                while(rs.next())
                {
                    int selectedID = rs.getInt(idColumn);
                    if(selectedID==selID)
                        Option+="<option value=\""+selectedID+"\"
Selected>"+rs.getString(valueColumn)+"</option>";
                    else
                        Option+="<option
value=\""+selectedID+"\">"+rs.getString(valueColumn)+"</option>";
```

```
                }
            }
            catch(Exception e)
            {
                System.out.println("Error : "+e);
            }
            return Option;
        }
}
```

Coding of Employee Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Employee extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
   public Employee()
   {
            Connect.connect_mysql();
   }
   //////////Save User Details /////
   public String saveEmployee(HashMap employeeData)
   {


        String SQL = "INSERT INTO employee (employee_sal, employee_first_name,
employee_middle_name, employee_last_name, employee_gender,
employee_address, employee_village, employee_state, employee_country,
employee_landline, employee_mobile, employee_email, employee_status,
employee_deparment, employee_dob, employee_nationalty,
employee_manager_id, employee_role) VALUES
( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,  ?, ?, ?, ?, ?, ?, ?);";
        int record=0,last_inserted_id=0;
        String error = "";


        try
```

```
        {
                pstmt = connection.prepareStatement(SQL,
Statement.RETURN_GENERATED_KEYS);
                pstmt.setString(1,(String) employeeData.get("employee_sal"));
                pstmt.setString(2,(String)
employeeData.get("employee_first_name"));
                pstmt.setString(3,(String)
employeeData.get("employee_middle_name"));
                pstmt.setString(4,(String)
employeeData.get("employee_last_name"));
                pstmt.setString(5,(String) employeeData.get("employee_gender"));
                pstmt.setString(6,(String) employeeData.get("employee_address"));
                pstmt.setString(7,(String) employeeData.get("employee_village"));
                pstmt.setString(8,(String) employeeData.get("employee_state"));
                pstmt.setString(9,(String) employeeData.get("employee_country"));
                pstmt.setString(10,(String) employeeData.get("employee_landline"));
                pstmt.setString(11,(String) employeeData.get("employee_mobile"));
                pstmt.setString(12,(String) employeeData.get("employee_email"));
                pstmt.setString(13,(String) employeeData.get("employee_status"));
                pstmt.setString(14,(String)
employeeData.get("employee_deparment"));
                pstmt.setString(15,(String) employeeData.get("employee_dob"));
                pstmt.setString(16,(String)
employeeData.get("employee_nationalty"));
                pstmt.setString(17,(String)
employeeData.get("employee_manager_id"));
                pstmt.setString(18,(String) employeeData.get("employee_role"));


                record = pstmt.executeUpdate();
                /// Get the Last Insert ID ///
                rs = pstmt.getGeneratedKeys();
```

```
            if(rs.next())
            {
                last_inserted_id = rs.getInt(1);
            }

            pstmt.close();
        }
        catch(Exception e)
        {
            StringWriter writer = new StringWriter();
            PrintWriter printWriter = new PrintWriter( writer );
            e.printStackTrace( printWriter );
            printWriter.flush();
            String stackTrace = writer.toString();
            error+="Error : "+stackTrace;
            System.out.println(" Error : "+ e.toString());
        }

        /// Save Credentials to Login ////
    SQL = "INSERT INTO login (login_emp_id, login_user, login_password,
login_email, login_level) VALUES (?, ?, ?, ?, ?);";
    try
        {
            pstmt = connection.prepareStatement(SQL);
            pstmt.setInt(1,last_inserted_id);
            pstmt.setString(2,(String) employeeData.get("employee_user"));
            pstmt.setString(3,(String) employeeData.get("employee_password"));
            pstmt.setString(4,(String) employeeData.get("employee_email"));
```

```
            pstmt.setString(5,(String) employeeData.get("employee_role"));
            record = pstmt.executeUpdate();
            pstmt.close();
            connection.close();
        }
        catch(Exception e)
        {
            StringWriter writer = new StringWriter();
            PrintWriter printWriter = new PrintWriter( writer );
            e.printStackTrace( printWriter );
            printWriter.flush();
            String stackTrace = writer.toString();
            error+="Error : "+stackTrace;
            System.out.println(" Error : "+ e.toString());
        }

        return error;
    }

    /////////////////Function for getting Users Details/////////
    public HashMap getEmployeeDetails(int employee_id)
        {
       HashMap resultsArray = new HashMap();
       int count=0;
            try
            {
          String SQL =  "SELECT * FROM employee WHERE employee_id =
"+employee_id ;
```

```
          statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                    {
        resultsArray.put("employee_id",rs.getString("employee_id"));

resultsArray.put("employee_manager_id",Integer.parseInt(rs.getString("employee_
manager_id")));

resultsArray.put("employee_role",Integer.parseInt(rs.getString("employee_role")));

resultsArray.put("employee_sal",Integer.parseInt(rs.getString("employee_sal")));

resultsArray.put("employee_first_name",rs.getString("employee_first_name"));

resultsArray.put("employee_middle_name",rs.getString("employee_middle_name")
);

resultsArray.put("employee_last_name",rs.getString("employee_last_name"));
        resultsArray.put("employee_gender",rs.getString("employee_gender"));
        resultsArray.put("employee_address",rs.getString("employee_address"));
        resultsArray.put("employee_village",rs.getString("employee_village"));

resultsArray.put("employee_state",Integer.parseInt(rs.getString("employee_state")
));
```

```
resultsArray.put("employee_country",Integer.parseInt(rs.getString("employee_cou
ntry")));

        resultsArray.put("employee_landline",rs.getString("employee_landline"));

        resultsArray.put("employee_mobile",rs.getString("employee_mobile"));

        resultsArray.put("employee_email",rs.getString("employee_email"));


resultsArray.put("employee_status",Integer.parseInt(rs.getString("employee_status
")));


resultsArray.put("employee_deparment",Integer.parseInt(rs.getString("employee_d
eparment")));

        resultsArray.put("employee_dob",rs.getString("employee_dob"));


resultsArray.put("employee_nationalty",rs.getString("employee_nationalty"));
                        count++;
        }
                if(count==0)
                {
                        resultsArray.put("employee_id","");
                        resultsArray.put("employee_manager_id",0);
                        resultsArray.put("employee_role",0);
                        resultsArray.put("employee_user","");
                        resultsArray.put("employee_sal",0);
                        resultsArray.put("employee_first_name","");
                        resultsArray.put("employee_middle_name","");
                        resultsArray.put("employee_last_name","");
```

```
                        resultsArray.put("employee_gender","");
                        resultsArray.put("employee_address","");
                        resultsArray.put("employee_village","");
                        resultsArray.put("employee_state",0);
                        resultsArray.put("employee_country",0);
                        resultsArray.put("employee_landline","");
                        resultsArray.put("employee_mobile","");
                        resultsArray.put("employee_email","");
                        resultsArray.put("employee_status",0);
                        resultsArray.put("employee_deparment",0);
                        resultsArray.put("employee_dob","");
                        resultsArray.put("employee_nationalty","");

                    }
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
         }
       return resultsArray;
   }


        public String updateEmployee(HashMap employeeData)
        {
                String SQL = "UPDATE employee SET employee_sal = ?,
employee_first_name = ?, employee_middle_name = ?, employee_last_name = ?,
employee_gender = ?, employee_address = ?, employee_village = ?,
employee_state = ?, employee_country = ?, employee_landline = ?,
employee_mobile = ?, employee_email = ?, employee_status = ?,
employee_deparment = ?, employee_dob = ?, employee_nationalty = ?,
```

```
employee_manager_id = ?, employee_role = ? WHERE employee_id = ?";
            String error = "";

            int record=0;
            try
            {
                  pstmt = connection.prepareStatement(SQL);
                  pstmt.setString(1,(String) employeeData.get("employee_sal"));
                  pstmt.setString(2,(String)
employeeData.get("employee_first_name"));
                  pstmt.setString(3,(String)
employeeData.get("employee_middle_name"));
                  pstmt.setString(4,(String)
employeeData.get("employee_last_name"));
                  pstmt.setString(5,(String)
employeeData.get("employee_gender"));
                  pstmt.setString(6,(String)
employeeData.get("employee_address"));
                  pstmt.setString(7,(String)
employeeData.get("employee_village"));
                  pstmt.setString(8,(String)
employeeData.get("employee_state"));
                  pstmt.setString(9,(String)
employeeData.get("employee_country"));
                  pstmt.setString(10,(String)
employeeData.get("employee_landline"));
                  pstmt.setString(11,(String)
employeeData.get("employee_mobile"));
                  pstmt.setString(12,(String)
employeeData.get("employee_email"));
                  pstmt.setString(13,(String)
employeeData.get("employee_status"));
                  pstmt.setString(14,(String)
```

```
employeeData.get("employee_deparment"));

                    pstmt.setString(15,(String)
employeeData.get("employee_dob"));

                    pstmt.setString(16,(String)
employeeData.get("employee_nationalty"));

                    pstmt.setString(17,(String)
employeeData.get("employee_manager_id"));

                    pstmt.setString(18,(String)
employeeData.get("employee_role"));

                    pstmt.setString(19,(String) employeeData.get("employee_id"));

                    record = pstmt.executeUpdate();

                    pstmt.close();

                    connection.close();

            }

            catch(Exception e)

            {

                    StringWriter writer = new StringWriter();

                    PrintWriter printWriter = new PrintWriter( writer );

                    e.printStackTrace( printWriter );

                    printWriter.flush();

                    String stackTrace = writer.toString();

                    error+="Error : "+stackTrace;

                    System.out.println(" Error : "+ e.toString());

            }

            return error;

      }

      public boolean deleteEmployee(int employee_id)

      {

            return true;
```

```
        }


        /////Function for Getting the List////////////

        public String getStateOption(Integer SelID)

    {

                int selectedID = SelID.intValue();

        return
Connect.getOptionList("state","state_id","state_name","state_id,state_name",select
edID,"1");

    }

        /////Function for Getting the List////////////

        public String getEmployeeOption(Integer SelID)

    {

                int selectedID = SelID.intValue();

        return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");

    }

    /////Function for Getting the List////////////

        public String getRoleOption(Integer SelID)

    {

                int selectedID = SelID.intValue();

        return
Connect.getOptionList("roles","role_id","role_title","role_id,role_title",selectedID,"1"
);

    }

        /////Function for Getting the List////////////

        public String getCountryOption(Integer SelID)

    {

                int selectedID = SelID.intValue();
```

```
        return
Connect.getOptionList("country","country_id","country_name","country_id,country_
name",selectedID,"1");
    }


        /////Function for Getting the List////////////

        public String getSalutionOption(Integer SelID)
    {
                int selectedID = SelID.intValue();
        return
Connect.getOptionList("salutions","sl_id","sl_name","sl_id,sl_name",selectedID,"1")
;
    }


        /////Function for Getting the List////////////

        public String getStatusOption(Integer SelID)
    {
                int selectedID = SelID.intValue();
        return
Connect.getOptionList("status","status_id","status_name","status_id,status_name",
selectedID,"1");
    }


    /////Function for Getting the List////////////

        public String getDepartmentOption(Integer SelID)
    {
                int selectedID = SelID.intValue();
        return
Connect.getOptionList("department","dept_id","dept_name","dept_id,dept_name",s
electedID,"1");
    }
```

```
        ////////////////Function for getting all the Airport Details//////////////////
    public ArrayList getAllEmployee(String managerID)
        {
                int count=0;
                String error = "";
                String SQL = "SELECT * FROM employee";

        ArrayList resultArray = new ArrayList();
        try
                {
           if(!managerID.equals("0"))
                        {
                                SQL = "SELECT * FROM employee WHERE
employee_manager_id = "+managerID;
                        }
           statement = connection.createStatement();
           rs = statement.executeQuery(SQL);
           while(rs.next())
                        {
                                HashMap results = new HashMap();
                                results.put("employee_id",rs.getString("employee_id"));

        results.put("employee_role",rs.getString("employee_role"));

        results.put("employee_sal",rs.getString("employee_sal"));

        results.put("employee_first_name",rs.getString("employee_first_name"));
```

```
results.put("employee_middle_name",rs.getString("employee_middle_name"));

        results.put("employee_last_name",rs.getString("employee_last_name"));

        results.put("employee_gender",rs.getString("employee_gender"));

        results.put("employee_address",rs.getString("employee_address"));

        results.put("employee_village",rs.getString("employee_village"));

        results.put("employee_state",rs.getString("employee_state"));

        results.put("employee_country",rs.getString("employee_country"));

        results.put("employee_landline",rs.getString("employee_landline"));

        results.put("employee_mobile",rs.getString("employee_mobile"));

        results.put("employee_email",rs.getString("employee_email"));


results.put("employee_status",Integer.parseInt(rs.getString("employee_status")));


results.put("employee_deparment",Integer.parseInt(rs.getString("employee_depar
ment")));

        results.put("employee_dob",rs.getString("employee_dob"));

        results.put("employee_nationalty",rs.getString("employee_nationalty"));


results.put("employee_manager_id",Integer.parseInt(rs.getString("employee_mana
ger_id")));
                                count++;
```

```
            resultArray.add(results);
          }
        }
              catch(Exception e)
              {
          StringWriter writer = new StringWriter();
                      PrintWriter printWriter = new PrintWriter( writer );
                      e.printStackTrace( printWriter );
                      printWriter.flush();
                      String stackTrace = writer.toString();
                      error+="Error : "+stackTrace;
                      System.out.println(" Error : "+ e.toString());
      }
      return resultArray;
    }

       public String getDepartment(int dept_id)
       {
      HashMap results = new HashMap();
      String SQL = "";
      String value = "";
      int count=0;
              try
              {
          SQL =  "SELECT dept_name FROM department WHERE dept_id =
"+dept_id ;
          statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
```

```
        while(rs.next())
                {
                        value = rs.getString("dept_name");
        }
       }
             catch(Exception e)
             {
        System.out.println("Error is: "+ e);
       }
      return value;
   }

   public String getStatus(int status_id)
       {
      HashMap results = new HashMap();
      String SQL = "";
      String value = "";
      int count=0;
             try
             {
        SQL =  "SELECT status_name FROM status WHERE status_id = "+status_id
;
        statement = connection.createStatement();
        rs = statement.executeQuery(SQL);
        while(rs.next())
                {
                        value = rs.getString("status_name");
        }
```

```
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
         }
       return value;
   }
}
```

Coding of Leave Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Leave extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public Leave()
   {
                Connect.connect_mysql();
   }
        //////////Save User Details /////
        public String saveLeave(HashMap leaveData)
        {
                String SQL = "INSERT INTO `leave` (leave_employee_id,
leave_reason, leave_description, leave_from_date, leave_to_date) VALUES (?, ?, ?,
?, ?);";
                int record=0;
                String error = "";


                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String)
leaveData.get("leave_employee_id"));
```

```
                pstmt.setString(2,(String) leaveData.get("leave_reason"));
                pstmt.setString(3,(String) leaveData.get("leave_description"));
                pstmt.setString(4,(String) leaveData.get("leave_from_date"));
                pstmt.setString(5,(String) leaveData.get("leave_to_date"));

                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
            }
            catch(Exception e)
            {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
    }
    /////////////////Function for getting Users Details/////////
public HashMap getLeaveDetails(int leave_id)
    {
   HashMap results = new HashMap();
   int count=0;
            try
            {
```

```
        String SQL =  "SELECT * FROM `leave` WHERE leave_id = "+leave_id ;
        statement = connection.createStatement();
        rs = statement.executeQuery(SQL);
        while(rs.next())
                {
                        results.put("leave_id",rs.getString("leave_id"));

results.put("leave_employee_id",Integer.parseInt(rs.getString("leave_employee_id"
)));
                        results.put("leave_reason",rs.getString("leave_reason"));

      results.put("leave_description",rs.getString("leave_description"));

      results.put("leave_from_date",rs.getString("leave_from_date"));

      results.put("leave_to_date",rs.getString("leave_to_date"));
                        results.put("leave_status",rs.getString("leave_status"));

                        count++;
        }
                if(count==0)
                {
                        results.put("leave_id","");
                        results.put("leave_employee_id",0);
                        results.put("leave_reason","");
                        results.put("leave_description","");
                        results.put("leave_from_date","");
                        results.put("leave_to_date","");
                        results.put("leave_status",0);
                }
```

```
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
         }
       return results;
    }
    /// Update the Leave ////
        public String updateLeave(HashMap leaveData)
        {
                String SQL = "UPDATE `leave` SET leave_status = ? WHERE leave_id
= ?";
                String error = "";

                int record=0;

                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String) leaveData.get("leave_status"));
                        pstmt.setString(2,(String) leaveData.get("leave_id"));
                        record = pstmt.executeUpdate();
                        pstmt.close();
                        connection.close();
                }
                catch(Exception e)
                {
                        StringWriter writer = new StringWriter();
```

```
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
        }

        ////////////////Function for getting all the Airport Details//////////////////
    public ArrayList getAllLeave(String emp_id, String managerID)
        {
            String SQL = "SELECT * FROM `leave`,`employee` WHERE
leave_employee_id = employee_id";
            int count=0;
        ArrayList resultArray = new ArrayList();
        try
            {
                if(!managerID.equals("0"))
                {
                    SQL = "SELECT * FROM `leave`,`employee` WHERE
leave_employee_id = employee_id AND employee_manager_id = "+managerID;
                }
                if(!emp_id.equals("0"))
                {
                    SQL = "SELECT * FROM `leave`,`employee` WHERE
leave_employee_id = employee_id AND employee_id = "+emp_id;
                }
```

```
                    statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                    {
                              HashMap results = new HashMap();
                              results.put("leave_id",rs.getString("leave_id"));

results.put("leave_employee_id",Integer.parseInt(rs.getString("leave_employee_id"
)));
                              results.put("leave_reason",rs.getString("leave_reason"));
        results.put("leave_description",rs.getString("leave_description"));
        results.put("leave_from_date",rs.getString("leave_from_date"));
        results.put("leave_to_date",rs.getString("leave_to_date"));
                              results.put("leave_status",rs.getString("leave_status"));

        results.put("employee_name",rs.getString("employee_first_name")+"
"+rs.getString("employee_last_name"));
                              count++;
             resultArray.add(results);
          }
       }
              catch(Exception e)
              {
          System.out.println("Error is: "+ e);
      }
      return resultArray;
   }
```

```
        /////Function for Getting the List////////////

        public String getEmployeeOption(Integer SelID)

    {

                int selectedID = SelID.intValue();

        return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");

    }

}
```

Coding of Login Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Login extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public Login()
   {
                Connect.connect_mysql();
   }
        /////////////////Function for Update the airport//////////////////////
        public boolean changePassword(String old_password,String new_password,
int login_id)
        {
                String SQL;
                int count = 0;
                try
                {
                   SQL = "SELECT * FROM login WHERE login_password =
'"+old_password+"' AND login_id = "+login_id ;
                        statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())   count++;
```

```
                    if(count==1)
                    {
                        SQL = "UPDATE login SET login_password=? WHERE
login_id=?";
                        int record=0;
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,new_password);
                        pstmt.setInt(2,login_id);
                        record = pstmt.executeUpdate();
                        pstmt.close();
                        connection.close();
                    }
                }
                catch(Exception e)
                {
                    System.out.println(" Error : "+ e.toString());
                }
                if(count==0)
                    return false;
                return true;
        }
        /////////////////Function for geting the Single Airport Details/////////
    public boolean checkLogin(String login_user,String login_password)
        {
       int count=0;
                try
                {
          String SQL = "SELECT * FROM login WHERE login_user = '"+login_user+"'
AND login_password = '"+login_password+"'" ;
```

```
        statement = connection.createStatement();
        rs = statement.executeQuery(SQL);
        while(rs.next())   count++;
      }
             catch(Exception e)
             {
        System.out.println("Error is: "+ e);
       }
            if(count==0)
                  return false;
     return true;
  }
      ////////////////Function for getting Login Details/////////
  public HashMap getLoginDetails(String login_user,String login_password)
      {
     HashMap resultsArray = new HashMap();
     int count=0;
            try
            {
        String SQL =  "SELECT * FROM login WHERE login_user = '"+login_user+"'
AND login_password = '"+login_password+"'" ;
        statement = connection.createStatement();
        rs = statement.executeQuery(SQL);
        while(rs.next())
                  {
                        resultsArray.put("login_id",rs.getString("login_id"));

      resultsArray.put("login_emp_id",rs.getString("login_emp_id"));
                        resultsArray.put("login_user",rs.getString("login_user"));
```

```
                        resultsArray.put("login_level",rs.getString("login_level"));
                        count++;
          }
                    if(count==0)
                    {
                        resultsArray.put("login_id","");
                        resultsArray.put("login_emp_id","");
                        resultsArray.put("login_user","");
                        resultsArray.put("login_level","");
                    }
        }
                catch(Exception e)
                {
           System.out.println("Error is: "+ e);
         }
       return resultsArray;
    }
    /////////////////Function for getting Login Details/////////
    public int checkUsernameExits(String login_user, int type)
        {
       HashMap resultsArray = new HashMap();
       int exits=0;
                try
                {
                    String SQL = "";
                    if(type == 1) {
                        SQL =  "SELECT * FROM login WHERE login_user =
'"+login_user+"'" ;
```

```
                    }
                    if(type == 2) {
                            SQL =  "SELECT * FROM login WHERE login_email =
'"+login_user+"'" ;
                    }
          statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                    {
                            exits++;
          }
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
         }
       return exits;
    }
}
```

Coding of Project Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Project extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public Project()
   {
                Connect.connect_mysql();
   }
        //////////Save User Details /////
        public String saveProject(HashMap projectData)
        {
                String SQL = "INSERT INTO project (project_type_id,
project_manager_id, project_title, project_description, project_frontend,
project_backend, project_client_name) VALUES (?, ?, ?, ?, ?, ?, ?);";
                int record=0;
                String error = "";


                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String) projectData.get("project_type_id"));
```

```
                    pstmt.setString(2,(String)
projectData.get("project_manager_id"));
                    pstmt.setString(3,(String) projectData.get("project_title"));
                    pstmt.setString(4,(String)
projectData.get("project_description"));
                    pstmt.setString(5,(String) projectData.get("project_frontend"));
                    pstmt.setString(6,(String) projectData.get("project_backend"));
                    pstmt.setString(7,(String)
projectData.get("project_client_name"));


                    record = pstmt.executeUpdate();
                    pstmt.close();
                    connection.close();
            }
            catch(Exception e)
            {
                    StringWriter writer = new StringWriter();
                    PrintWriter printWriter = new PrintWriter( writer );
                    e.printStackTrace( printWriter );
                    printWriter.flush();
                    String stackTrace = writer.toString();
                    error+="Error : "+stackTrace;
                    System.out.println(" Error : "+ e.toString());
            }
            return error;
    }
    /////////////////Function for getting Users Details/////////
  public HashMap getProjectDetails(int project_id)
    {
```

```
        HashMap results = new HashMap();
        int count=0;
                try
                {
           String SQL =  "SELECT * FROM `project` WHERE project_id = "+project_id
;
           statement = connection.createStatement();
           rs = statement.executeQuery(SQL);
           while(rs.next())
                        {


results.put("project_type_id",Integer.parseInt(rs.getString("project_type_id")));


results.put("project_manager_id",Integer.parseInt(rs.getString("project_manager_i
d")));
                                results.put("project_title",rs.getString("project_title"));

          results.put("project_description",rs.getString("project_description"));

          results.put("project_frontend",rs.getString("project_frontend"));

          results.put("project_backend",rs.getString("project_backend"));

          results.put("project_client_name",rs.getString("project_client_name"));

                                results.put("project_id",rs.getString("project_id"));

                                count++;

           }
                        if(count==0)
                        {
```

```
                    results.put("project_type_id",0);
                    results.put("project_manager_id",0);
                    results.put("project_title","");
                    results.put("project_description","");
                    results.put("project_frontend","");
                    results.put("project_backend","");
                    results.put("project_client_name","");
                    results.put("project_id","");
                }
        }
            catch(Exception e)
            {
          System.out.println("Error is: "+ e);
         }
       return results;
   }
   /// Update the Project ////
        public String updateProject(HashMap projectData)
        {
            String SQL = "UPDATE project SET project_type_id = ?,
project_manager_id = ?, project_title = ?, project_description = ?,
project_frontend = ?, project_backend = ?, project_client_name = ? WHERE
project_id = ?;";
            String error = "";

            int record=0;

            try
            {
```

```
                pstmt = connection.prepareStatement(SQL);
                pstmt.setString(1,(String) projectData.get("project_type_id"));
                pstmt.setString(2,(String)
projectData.get("project_manager_id"));
                pstmt.setString(3,(String) projectData.get("project_title"));
                pstmt.setString(4,(String)
projectData.get("project_description"));
                pstmt.setString(5,(String) projectData.get("project_frontend"));
                pstmt.setString(6,(String) projectData.get("project_backend"));
                pstmt.setString(7,(String)
projectData.get("project_client_name"));
                pstmt.setString(8,(String) projectData.get("project_id"));
                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
            }
            catch(Exception e)
            {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
        }
```

```
        ////////////////Function for getting all the Airport Details//////////////////
    public ArrayList getAllProject()
        {
                String SQL = "SELECT * FROM `project`,`employee`,`type` WHERE
project_manager_id = employee_id AND project_type_id = type_id";
                int count=0;
        ArrayList resultArray = new ArrayList();
        try
                {
                        statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                        {
                                HashMap results = new HashMap();

         results.put("project_type_id",rs.getString("project_type_id"));


results.put("project_manager_id",Integer.parseInt(rs.getString("project_manager_i
d")));
                                results.put("project_title",rs.getString("project_title"));

         results.put("project_description",rs.getString("project_description"));

         results.put("project_frontend",rs.getString("project_frontend"));

         results.put("project_backend",rs.getString("project_backend"));

         results.put("project_client_name",rs.getString("project_client_name"));

                                results.put("project_type",rs.getString("type_title"));
```

```
                        results.put("project_id",rs.getString("project_id"));


        results.put("employee_name",rs.getString("employee_first_name")+"
"+rs.getString("employee_last_name"));
                        count++;
            resultArray.add(results);
          }
        }
              catch(Exception e)
              {
          System.out.println("Error is: "+ e);
       }
       return resultArray;
    }
        /////Function for Getting the List////////////
        public String getEmployeeOption(Integer SelID)
    {
              int selectedID = SelID.intValue();
        return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");
    }
    /////Function for Getting the List////////////
        public String getProjectTypeOption(Integer SelID)
    {
              int selectedID = SelID.intValue();
        return
Connect.getOptionList("type","type_id","type_title","type_id,type_title",selectedID,"
1");
```

Coding of Project Assignment Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class ProjectAssignment extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public ProjectAssignment()
   {
                Connect.connect_mysql();
   }
        //////////Save User Details /////
        public String saveProjectAssignment(HashMap projectData)
        {
                String SQL = "INSERT INTO project_assignment (pra_emp_id,
pra_prj_id, pra_assgindate) VALUES (?, ?, ?);";
                int record=0;
                String error = "";


                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String) projectData.get("pra_emp_id"));
                        pstmt.setString(2,(String) projectData.get("pra_prj_id"));
```

```
                pstmt.setString(3,(String) projectData.get("pra_assgindate"));


                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
            }
            catch(Exception e)
            {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
        }
        /////////////////Function for getting Users Details/////////
    public HashMap getProjectAssignmentDetails(int pra_id)
        {
       HashMap results = new HashMap();
       int count=0;
            try
            {
          String SQL =  "SELECT * FROM `project_assignment` WHERE pra_id =
"+pra_id ;
          statement = connection.createStatement();
```

```
      rs = statement.executeQuery(SQL);
      while(rs.next())
                  {
    results.put("pra_emp_id",Integer.parseInt(rs.getString("pra_emp_id")));
    results.put("pra_prj_id",Integer.parseInt(rs.getString("pra_prj_id")));
    results.put("pra_assgindate",rs.getString("pra_assgindate"));
                        results.put("pra_id",rs.getString("pra_id"));

                        count++;
      }
                  if(count==0)
                  {
                        results.put("pra_emp_id",0);
                        results.put("pra_prj_id",0);
                        results.put("pra_assgindate","");
                        results.put("pra_id","");
                  }
    }
            catch(Exception e)
            {
      System.out.println("Error is: "+ e);
     }
   return results;
}
/// Update the ProjectAssignment ////
    public String updateProjectAssignment(HashMap projectData)
```

```
	{
		String SQL = "UPDATE project_assignment SET pra_emp_id = ?,
pra_prj_id = ?, pra_assgindate = ? WHERE pra_id = ?;";
		String error = "";

		int record=0;

		try
		{
			pstmt = connection.prepareStatement(SQL);
			pstmt.setString(1,(String) projectData.get("pra_emp_id"));
			pstmt.setString(2,(String) projectData.get("pra_prj_id"));
			pstmt.setString(3,(String) projectData.get("pra_assgindate"));
			pstmt.setString(4,(String) projectData.get("pra_id"));

			record = pstmt.executeUpdate();
			pstmt.close();
			connection.close();
		}
		catch(Exception e)
		{
			StringWriter writer = new StringWriter();
			PrintWriter printWriter = new PrintWriter( writer );
			e.printStackTrace( printWriter );
			printWriter.flush();
			String stackTrace = writer.toString();
			error+="Error : "+stackTrace;
			System.out.println(" Error : "+ e.toString());
```

```
            }
            return error;
        }

        ////////////////Function for getting all the Airport Details//////////////////
    public ArrayList getAllProjectAssignment()
        {
            String SQL = "SELECT * FROM
`project_assignment`,`employee`,`project` WHERE pra_emp_id = employee_id
AND pra_prj_id = project_id";
            int count=0;
        ArrayList resultArray = new ArrayList();
        try
            {
                statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                {
                    HashMap results = new HashMap();

        results.put("pra_emp_id",Integer.parseInt(rs.getString("pra_emp_id")));

        results.put("pra_prj_id",Integer.parseInt(rs.getString("pra_prj_id")));

        results.put("pra_assgindate",rs.getString("pra_assgindate"));
                    results.put("pra_id",rs.getString("pra_id"));
                    results.put("project_title",rs.getString("project_title"));
                    results.put("project_id",rs.getString("project_id"));

        results.put("employee_name",rs.getString("employee_first_name")+"
"+rs.getString("employee_last_name"));
```

```
                        count++;
                resultArray.add(results);
              }
            }
                    catch(Exception e)
                    {
              System.out.println("Error is: "+ e);
           }
           return resultArray;
    }
        /////Function for Getting the List////////////
        public String getEmployeeOption(Integer SelID)
    {
                int selectedID = SelID.intValue();
           return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");
    }
    /////Function for Getting the List////////////
        public String getProjectOption(Integer SelID)
    {
                int selectedID = SelID.intValue();
           return
Connect.getOptionList("project","project_id","project_title","project_id,project_title"
,selectedID,"1");
    }
}
```

Coding of Salary Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Salary extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public Salary()
   {
                Connect.connect_mysql();
   }
        //////////Save User Details /////
        public String saveSalary(HashMap salaryData)
        {
                String SQL = "INSERT INTO salary (sal_employe_id, sal_month,
sal_year, sal_amount) VALUES (?, ?, ?, ?);";
                int record=0;
                String error = "";


                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String) salaryData.get("sal_employe_id"));
                        pstmt.setString(2,(String) salaryData.get("sal_month"));
```

```
                pstmt.setString(3,(String) salaryData.get("sal_year"));
                pstmt.setString(4,(String) salaryData.get("sal_amount"));

                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
            }
            catch(Exception e)
            {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
    }
    /////////////////Function for getting Users Details/////////
public HashMap getSalaryDetails(int sal_id)
    {
   HashMap results = new HashMap();
   int count=0;
            try
            {
      String SQL =  "SELECT * FROM salary WHERE sal_id = "+sal_id ;
      statement = connection.createStatement();
```

```
        rs = statement.executeQuery(SQL);
        while(rs.next())
                {
                        results.put("sal_id",rs.getString("sal_id"));

results.put("sal_employe_id",Integer.parseInt(rs.getString("sal_employe_id")));
                        results.put("sal_year",rs.getString("sal_year"));

      results.put("sal_month",Integer.parseInt(rs.getString("sal_month")));
                        results.put("sal_amount",rs.getString("sal_amount"));
                        count++;
        }
                if(count==0)
                {
                        results.put("sal_id","");
                        results.put("sal_employe_id",0);
                        results.put("sal_year","");
                        results.put("sal_month",0);
                        results.put("sal_amount","");
                }
     }
            catch(Exception e)
            {
        System.out.println("Error is: "+ e);
      }
    return results;
  }
```

```
        /////////////////Function for getting Users Details/////////
    public String getMonth(int mon_id)
        {
        HashMap results = new HashMap();
        String SQL = "";
        String value = "";
        int count=0;
                try
                {
          SQL =  "SELECT month_name FROM month WHERE month_id =
"+mon_id ;
          statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                        {
                                value = rs.getString("month_name");
          }
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
         }
        return value;
    }


        public String updateSalary(HashMap salaryData)
        {
                String SQL = "UPDATE salary SET sal_employe_id = ?,sal_year
= ?,sal_month = ?,sal_amount = ? WHERE sal_id = ?";
```

```
        String error = "";

        int record=0;

        try
        {
                pstmt = connection.prepareStatement(SQL);
                pstmt.setString(1,(String) salaryData.get("sal_employe_id"));
                pstmt.setString(2,(String) salaryData.get("sal_year"));
                pstmt.setString(3,(String) salaryData.get("sal_month"));
                pstmt.setString(4,(String) salaryData.get("sal_amount"));
                pstmt.setString(5,(String) salaryData.get("sal_id"));
                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
        }
        catch(Exception e)
        {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
        }
        return error;
}
```

```
	public boolean delete_login(int airline_id)
	{
		return true;
	}

	////////////////Function for getting all the Airport Details//////////////////
   public ArrayList getAllSalary(String emp_id)
	{
		String SQL = "SELECT * FROM salary";
		int count=0;
      ArrayList resultArray = new ArrayList();
      try
		{
			if(!emp_id.equals("0"))
			{
				SQL = "SELECT * FROM salary WHERE sal_employe_id =
"+emp_id;
			}
			statement = connection.createStatement();
         rs = statement.executeQuery(SQL);
         while(rs.next())
			{
				HashMap results = new HashMap();
				results.put("sal_id",rs.getString("sal_id"));

results.put("sal_employe_id",Integer.parseInt(rs.getString("sal_employe_id")));
				results.put("sal_year",rs.getString("sal_year"));
```

```
        results.put("sal_month",Integer.parseInt(rs.getString("sal_month")));
                                results.put("sal_amount",rs.getString("sal_amount"));


                                count++;
                    resultArray.add(results);
                  }
                }
                        catch(Exception e)
                        {
                   System.out.println("Error is: "+ e);
                }
                return resultArray;
           }
                /////Function for Getting the List////////////
                public String getMonthOption(Integer SelID)
           {
                        int selectedID = SelID.intValue();
                return
Connect.getOptionList("month","month_id","month_name","month_id,month_name
",selectedID,"1");
           }
                /////Function for Getting the List////////////
                public String getEmployeeOption(Integer SelID)
           {
                        int selectedID = SelID.intValue();
                return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");
           }
```

```
}
```

Coding of Project Bug and Ticket Section

```
package Model;


import java.util.*;
import java.sql.*;
import com.*;
import java.io.*;


public class Ticket extends Connect
{
   /////Function for connect to the MySQL Server Database///////////
        public Ticket()
   {
                Connect.connect_mysql();
   }
        //////////Save User Details /////
        public String saveTicket(HashMap ticketData)
        {
                String SQL = "INSERT INTO ticket (ticket_project_id, ticket_type_id,
ticket_employee_id, ticket_title, ticket_description, ticket_start_date,
ticket_due_date, ticket_hours, ticket_status) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";
                int record=0;
                String error = "";


                try
                {
                        pstmt = connection.prepareStatement(SQL);
                        pstmt.setString(1,(String) ticketData.get("ticket_project_id"));
```

```
                pstmt.setString(2,(String) ticketData.get("ticket_type_id"));
                pstmt.setString(3,(String)
ticketData.get("ticket_employee_id"));
                pstmt.setString(4,(String) ticketData.get("ticket_title"));
                pstmt.setString(5,(String) ticketData.get("ticket_description"));
                pstmt.setString(6,(String) ticketData.get("ticket_start_date"));
                pstmt.setString(7,(String) ticketData.get("ticket_due_date"));
                pstmt.setString(8,(String) ticketData.get("ticket_hours"));
                pstmt.setString(9,(String) ticketData.get("ticket_status"));



                record = pstmt.executeUpdate();
                pstmt.close();
                connection.close();
        }
        catch(Exception e)
        {
                StringWriter writer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( writer );
                e.printStackTrace( printWriter );
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
        }
        return error;
    }
    /////////////////Function for getting Users Details/////////
```

```
    public HashMap getTicketDetails(int ticket_id)
        {
       HashMap results = new HashMap();
       int count=0;
              try
              {
          String SQL =  "SELECT * FROM `ticket` WHERE ticket_id = "+ticket_id ;
          statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                       {

results.put("ticket_project_id",Integer.parseInt(rs.getString("ticket_project_id")));

       results.put("ticket_type_id",Integer.parseInt(rs.getString("ticket_type_id")));

results.put("ticket_employee_id",Integer.parseInt(rs.getString("ticket_employee_id
")));
                                results.put("ticket_title",rs.getString("ticket_title"));

       results.put("ticket_description",rs.getString("ticket_description"));

       results.put("ticket_start_date",rs.getString("ticket_start_date"));

       results.put("ticket_due_date",rs.getString("ticket_due_date"));

                                results.put("ticket_hours",rs.getString("ticket_hours"));

       results.put("ticket_status",Integer.parseInt(rs.getString("ticket_status")));
                                results.put("ticket_id",rs.getString("ticket_id"));
```

```
                count++;
        }
                if(count==0)
                {
                        results.put("ticket_project_id",0);
                        results.put("ticket_type_id",0);
                        results.put("ticket_employee_id",0);
                        results.put("ticket_title","");
                        results.put("ticket_description","");
                        results.put("ticket_start_date","");
                        results.put("ticket_due_date","");
                        results.put("ticket_hours","");
                        results.put("ticket_status",0);
                        results.put("ticket_id","");
                }
      }
              catch(Exception e)
              {
        System.out.println("Error is: "+ e);
       }
     return results;
  }
  /// Update the Ticket ////
       public String updateTicket(HashMap ticketData)
       {
              String SQL = "UPDATE ticket SET ticket_project_id = ?, ticket_type_id
= ?, ticket_employee_id = ?, ticket_title = ?, ticket_description = ?,
ticket_start_date = ?, ticket_due_date = ?, ticket_hours = ?, ticket_status = ?
WHERE ticket_id = ?;";
```

```
		String error = "";

		int record=0;

		try
		{
			pstmt = connection.prepareStatement(SQL);
			pstmt.setString(1,(String) ticketData.get("ticket_project_id"));
			pstmt.setString(2,(String) ticketData.get("ticket_type_id"));
			pstmt.setString(3,(String)
ticketData.get("ticket_employee_id"));
			pstmt.setString(4,(String) ticketData.get("ticket_title"));
			pstmt.setString(5,(String) ticketData.get("ticket_description"));
			pstmt.setString(6,(String) ticketData.get("ticket_start_date"));
			pstmt.setString(7,(String) ticketData.get("ticket_due_date"));
			pstmt.setString(8,(String) ticketData.get("ticket_hours"));
			pstmt.setString(9,(String) ticketData.get("ticket_status"));
			pstmt.setString(10,(String) ticketData.get("ticket_id"));


			record = pstmt.executeUpdate();
			pstmt.close();
			connection.close();
		}
		catch(Exception e)
		{
			StringWriter writer = new StringWriter();
			PrintWriter printWriter = new PrintWriter( writer );
			e.printStackTrace( printWriter );
```

```
                printWriter.flush();
                String stackTrace = writer.toString();
                error+="Error : "+stackTrace;
                System.out.println(" Error : "+ e.toString());
            }
            return error;
        }

        ////////////////Function for getting all the Airport Details//////////////////
    public ArrayList getAllTicket()
        {
            String SQL = "SELECT * FROM
`ticket`,`employee`,`project`,`ticket_type` WHERE ticket_employee_id =
employee_id AND ticket_type_id = tt_id AND ticket_project_id = project_id";
            int count=0;
        ArrayList resultArray = new ArrayList();
        try
            {
                statement = connection.createStatement();
          rs = statement.executeQuery(SQL);
          while(rs.next())
                {
                    HashMap results = new HashMap();

results.put("ticket_project_id",Integer.parseInt(rs.getString("ticket_project_id")));

        results.put("ticket_type_id",Integer.parseInt(rs.getString("ticket_type_id")));

results.put("ticket_employee_id",Integer.parseInt(rs.getString("ticket_employee_id
```

```
")));
                        results.put("ticket_title",rs.getString("ticket_title"));
        results.put("ticket_description",rs.getString("ticket_description"));
        results.put("ticket_start_date",rs.getString("ticket_start_date"));
        results.put("ticket_due_date",rs.getString("ticket_due_date"));
                        results.put("ticket_hours",rs.getString("ticket_hours"));

        results.put("ticket_status",Integer.parseInt(rs.getString("ticket_status")));
                        results.put("ticket_id",rs.getString("ticket_id"));
                        results.put("tt_title",rs.getString("tt_title"));
                        results.put("project_title",rs.getString("project_title"));
        results.put("employee_name",rs.getString("employee_first_name")+"
"+rs.getString("employee_last_name"));
                        count++;
            resultArray.add(results);
          }
       }
              catch(Exception e)
              {
          System.out.println("Error is: "+ e);
      }
      return resultArray;
   }
       /////Function for Getting the List///////////
       public String getEmployeeOption(Integer SelID)
```

```
    {
            int selectedID = SelID.intValue();
        return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");
    }
    /////Function for Getting the List/////////////
        public String getTicketTypeOption(Integer SelID)
    {
            int selectedID = SelID.intValue();
        return
Connect.getOptionList("ticket_type","tt_id","tt_title","tt_id,tt_title",selectedID,"1");
    }
    /////Function for Getting the List/////////////
        public String getProjectOption(Integer SelID)
    {
            int selectedID = SelID.intValue();
        return
Connect.getOptionList("project","project_id","project_title","project_id,project_title"
,selectedID,"1");
    }
    /////Function for Getting the List/////////////
        public String getTicketStatusOption(Integer SelID)
    {
            int selectedID = SelID.intValue();
        return
Connect.getOptionList("ticket_status","ts_id","ts_title","ts_id,ts_title",selectedID,"1
");
    }
}
```

Coding of Timesheet Section

```
package Model;


import java.util.*;

import java.sql.*;

import com.*;

import java.io.*;


public class Timesheet extends Connect

{

   /////Function for connect to the MySQL Server Database///////////

        public Timesheet()

   {

                Connect.connect_mysql();

   }

        //////////Save User Details /////

        public String saveTimesheet(HashMap timesheetData)

        {

                String SQL = "INSERT INTO timesheet (timesheet_employee_id,
timesheet_project_id, timesheet_work_title, timesheet_description,
timesheet_hours, timesheet_date) VALUES (?, ?, ?, ?, ?, ?);";

                int record=0;

                String error = "";


                try

                {

                        pstmt = connection.prepareStatement(SQL);

                        pstmt.setString(1,(String)
timesheetData.get("timesheet_employee_id"));
```

```
                    pstmt.setString(2,(String)
timesheetData.get("timesheet_project_id"));

                    pstmt.setString(3,(String)
timesheetData.get("timesheet_work_title"));

                    pstmt.setString(4,(String)
timesheetData.get("timesheet_description"));

                    pstmt.setString(5,(String)
timesheetData.get("timesheet_hours"));

                    pstmt.setString(6,(String)
timesheetData.get("timesheet_date"));


                    record = pstmt.executeUpdate();

                    pstmt.close();

                    connection.close();

            }

            catch(Exception e)

            {

                    StringWriter writer = new StringWriter();

                    PrintWriter printWriter = new PrintWriter( writer );

                    e.printStackTrace( printWriter );

                    printWriter.flush();

                    String stackTrace = writer.toString();

                    error+="Error : "+stackTrace;

                    System.out.println(" Error : "+ e.toString());

            }

            return error;

    }

    /////////////////Function for getting Users Details/////////

  public HashMap getTimesheetDetails(int timesheet_id)

    {
```

```
HashMap results = new HashMap();
int count=0;
        try
        {
    String SQL =  "SELECT * FROM `timesheet` WHERE timesheet_id =
"+timesheet_id ;
    statement = connection.createStatement();
    rs = statement.executeQuery(SQL);
    while(rs.next())
                {
                        results.put("timesheet_id",rs.getString("timesheet_id"));

results.put("timesheet_employee_id",Integer.parseInt(rs.getString("timesheet_emp
loyee_id")));

results.put("timesheet_project_id",Integer.parseInt(rs.getString("timesheet_project
_id")));

    results.put("timesheet_work_title",rs.getString("timesheet_work_title"));

    results.put("timesheet_description",rs.getString("timesheet_description"));

    results.put("timesheet_hours",rs.getString("timesheet_hours"));

    results.put("timesheet_date",rs.getString("timesheet_date"));

                        count++;
    }
                if(count==0)
                {
                        results.put("timesheet_id","");
```

```
                    results.put("timesheet_employee_id",0);
                    results.put("timesheet_project_id",0);
                    results.put("timesheet_work_title","");
                    results.put("timesheet_description","");
                    results.put("timesheet_hours","");
                    results.put("timesheet_date","");
                }
        }
            catch(Exception e)
            {
          System.out.println("Error is: "+ e);
         }
       return results;
    }
    /// Update the Timesheet ////
        public String updateTimesheet(HashMap timesheetData)
        {
            String SQL = "UPDATE timesheet SET timesheet_employee_id = ?,
timesheet_project_id = ?, timesheet_work_title = ?, timesheet_description = ?,
timesheet_hours = ?, timesheet_date = ? WHERE timesheet_id = ?;";
            String error = "";

            int record=0;

            try
            {
                pstmt = connection.prepareStatement(SQL);
                pstmt.setString(1,(String)
timesheetData.get("timesheet_employee_id"));
```

```
                    pstmt.setString(2,(String)
timesheetData.get("timesheet_project_id"));

                    pstmt.setString(3,(String)
timesheetData.get("timesheet_work_title"));

                    pstmt.setString(4,(String)
timesheetData.get("timesheet_description"));

                    pstmt.setString(5,(String)
timesheetData.get("timesheet_hours"));

                    pstmt.setString(6,(String)
timesheetData.get("timesheet_date"));

                    pstmt.setString(7,(String) timesheetData.get("timesheet_id"));


                    record = pstmt.executeUpdate();

                    pstmt.close();

                    connection.close();

            }

            catch(Exception e)

            {

                    StringWriter writer = new StringWriter();

                    PrintWriter printWriter = new PrintWriter( writer );

                    e.printStackTrace( printWriter );

                    printWriter.flush();

                    String stackTrace = writer.toString();

                    error+="Error : "+stackTrace;

                    System.out.println(" Error : "+ e.toString());

            }

            return error;

    }


    ////////////////Function for getting all the Airport Details//////////////////
```

```
    public ArrayList getAllTimesheet(String emp_id, String managerID)

        {

                String SQL = "SELECT * FROM `timesheet`,`employee` WHERE
timesheet_employee_id = employee_id";

                int count=0;

        ArrayList resultArray = new ArrayList();

        try

                {

                        if(!managerID.equals("0"))

                        {

                                SQL = "SELECT * FROM `timesheet`,`employee` WHERE
timesheet_employee_id = employee_id AND employee_manager_id =
"+managerID;

                        }


                        if(!emp_id.equals("0"))

                        {

                                SQL = "SELECT * FROM `timesheet`,`employee` WHERE
timesheet_employee_id = employee_id AND employee_id = "+emp_id;

                        }

                        statement = connection.createStatement();

            rs = statement.executeQuery(SQL);

            while(rs.next())

                        {

                                HashMap results = new HashMap();

                                results.put("timesheet_id",rs.getString("timesheet_id"));


results.put("timesheet_employee_id",Integer.parseInt(rs.getString("timesheet_emp
loyee_id")));
```

```
results.put("timesheet_project_id",Integer.parseInt(rs.getString("timesheet_project
_id")));

        results.put("timesheet_work_title",rs.getString("timesheet_work_title"));

        results.put("timesheet_description",rs.getString("timesheet_description"));

        results.put("timesheet_hours",rs.getString("timesheet_hours"));

        results.put("timesheet_date",rs.getString("timesheet_date"));

        results.put("employee_name",rs.getString("employee_first_name")+"
"+rs.getString("employee_last_name"));
                                count++;
            resultArray.add(results);
          }
        }
                catch(Exception e)
                {
          System.out.println("Error is: "+ e);
      }
      return resultArray;
   }
        /////Function for Getting the List////////////
        public String getEmployeeOption(Integer SelID)
   {
                int selectedID = SelID.intValue();
        return
Connect.getOptionList("employee","employee_id","employee_id","employee_id,emp
loyee_id",selectedID,"1");
   }
```

```
    /////Function for Getting the List/////////////

        public String getProjectOption(Integer SelID)

    {

                int selectedID = SelID.intValue();

        return
Connect.getOptionList("project","project_id","project_id","project_id,project_id",sel
ectedID,"1");

    }

}
```

References and Bibliography:

- http://www.bluedart.com/
- http://www.wampserver.com/en/
- http://www.JSP.net/
- http://www.tutorialspoint.com/mysql/
- httpd.apache.org/docs/2.0/misc/tutorials.html