**GPSSimCRM**

PROJECT REPORT

(KCA451)

*Submitted in Partial Fulfilment for the award of the degree*

*of*

**MASTER OF COMPUTER APPLICATIONS**

(M.C.A)

Submitted by

**CHITRA MAURYA**

2000380140012

Under the Supervision of

**Prof. Abhay Kumar Ray**

ASSISTANT PROFESSOR

DEPARTMENT OF IT



**Department of Information Technology**

**Institute of Technology & Science**

Mohan Nagar, Ghaziabad-201007

(2020-2022)

[www.its.edu.in](http://www.its.edu.in)

**DECLARATION**

I hereby declare that this project work entitled “GPSSimCRM” has been prepared by me during the year 2021 – 22 under the guidance of Prof. Abhay Kumar Ray, Department of IT, Institute of Technology and Science, Mohan Nagar Ghaziabad in the partial fulfilment of MCA degree prescribed by the college.

I also declare that this project is the outcome of my own effort, that it has not been submitted to any other university for the award of any degree.

Date: 13 May 2022

CHITRA MAURYA

2000380140012

**CERTIFICATE**

### This is to Certify that **Chitra Maurya** has carried out the project work presented in this report entitled “**GPSSimCRM”** for the award of **Masters of Computer** **Application from Institute of Technology & Science, Mohan Nagar, Ghaziabad** under my supervision. The report embodies result of original work and studies carried out by Student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else.

Date: 13 May 2022, PROF. ABHAY KUMAR RAY

Assistant Professor, Dept-IT

Institute Of Technology & Science

Mohan Nagar, Ghaziabad

**ACKNOWLEDGEMENT**

I take this opportunity to thank our teachers and friends who helped me throughout the project.

First and foremost I would like to thank my guide for the project (PROF. ABHAY KUMAR RAY, ASSISTANT PROFESSOR, DEPARTMENT OF IT) for his valuable advice and time during development of project.

I would also like to thank Dr. Sunil Kumar Pandey, Director-IT for his constant support during the development of the project.

CHITRA MAURYA

Roll No: 2000380140012

**ABSTRACT**

I have developed the GPS sim CRM tool to keep track and solve the problems faced by our consumers. We receive queries of our consumers through tickets raised by them in the support section. Consumers can report the nature of the problem that they are facing and the support staff can get back to them on their issues.

Consumers can also create the defected by filing the Raise defect form. There is a defect form which the consumer needs to fill up in order to report the issues where they fill up details such as the name of the module, module version, severity of problem.

The issues reported by the consumers go to the sales team first and then the support staff and if the issue remains unsolved even at that time, then the issue is reported to the engineer or developer for thorough problem diagnosis.

**INDEX**

Chapter 1: Introduction

* 1. General
  2. Background
  3. Objective

Chapter 2: System Analysis

1. Issues and challenges in Bug tracking system
2. Trends in Bug tracking system
3. Research on other similar current system
4. Requirement analysis
5. Problem Statement
6. Existing Work

Chapter 3: Proposed Approach

1. Introduction
2. Software Specification
3. Role and Responsibility
4. System Development Life Cycle (SDLC)
5. Data Flow Diagram (DFD)
6. Function Specification
7. MVC Architecture

Chapter 4: Implementation

4.1 Use-Case Diagram

4.2 ER Diagram

4.3 Database Design

4.4 Project Architecture

4.5 User interface

Chapter 5: Deployment and Demo

Chapter 6: Test Cases

Chapter 7: Result and Discussion

7.1 Introduction

7.2 Testing

7.3 Maintenance

7.4 Challenges

Chapter 8: Conclusion

Chapter 9: Future Work

Chapter 10: References

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

This project is based on CRM domain for a product GPSSim. Objective of this web-based project is to handle various queries, suggestion, defects and enhancements posted by GPSSim (Product of GPS based on IT Infrastructure Monitoring System) customers. There are 4 major sections in this project: Sales, Customer, GPS Employees, and Administrator.

A software defect is an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result or causes it to behave in unplanned ways. Most bugs arise from mistakes and errors made by people in either a program’s source code or its design.

By using defect tracking tools, we can keep track of reported software bugs with the help of bug tracking tools developer can further refine software design by making changes or upgrading the software. A bug tracking system is a software application that is designed to help quality assurance and programmers keep track of reported software bugs in their work.

Once the bug is added, it is tracked through to completion. Users of bug tracking system are generally clients, software testers, project team members, management members and software developers. It provides a facility to all the users to track and manage multiple projects concurrently. The testing team can use the bug tracking system to record issues progress, until such time that a developer could look into the other issues.

* 1. **BACKGROUND**

The ultimate aim of software maintenance is to not only improve the performance but also fix defects and enhance attributes of the software, which lead to better quality software. Software bugs usually appear during software development process, unfound bugs can lead to the loss of billions of dollars in the software maintenance process for large-scale software programs, software artefacts, especially for bug reports, become an important medium to help developers resolve bugs. Specifically, a user or developer can report the software bug in a fixed format (i.e., bug report) and upload it to a bug tracking system such as Bugzilla1 and FogBugz2. Then, a senior developer is assigned to fix the reported bug according the information shown in this submitted report.

* 1. **OBJECTIVE**

The aim of GPSSimCRM is to review, discuss and improve current bug reporting and maintaining procedures, in order to increase consistency, readability and ease of use. it is possible to divide the bug list into smaller, logically sorted and properly described entries, with minimal amount of upkeep required. We hope to make GPSSimCRM user friendly, especially to project newcomers, future developers and all those that currently feel lost after generating full list of opened bugs.

The aim of this defect tracking tool is to study the bug and the mechanisms to resolve and its life cycle in which it is determined how the bug arises and its consequences.

GPSSimCRM can be defined as a process of detecting defects, reporting on them, and fixing them.

**CHAPTER 2**

**SYSTEM ANALYSIS**

1. **ISSUES AND CHALLENGES BUG TRACKING SYSTEM**

Software development project faces multitude of challenges. One of those challenges is the defects found in the project. Defects or bugs are problems exist in the system and they are normally unavoidable in software development. A software bug is the common term used to describe an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result or causes it to behave in unintended ways.

Most bugs arise from mistakes and errors made by people in either a program’s source code or its technical design. The presence of bugs has caused the system fail to meet customer’s reasonable expectation of quality. Quality is defined as meeting the customer’s requirements and suitable for the use of users. Defect or bug is something that does not allow the product to meet customer requirements, or maybe does not fulfil the aspect of fitness for use. The quality problem causes the customers dissatisfactions or project failure. Defect is not an accident, but it occurs because something has not worked as planned. Since that, people realized that implementing bug tracking process is important in order to find, identify and remove bugs or issues within application and eventually keep the system running properly.

Besides, the purpose of tracking bug is to improve the quality of software products; it is an eternal truth that newly written software packages will contain bugs and it is inevitable. When a system gets a bug, it can cause all sorts of problems. Bugs can cause the system to perform task incorrectly or cause things to occur that should not be. This can be especially annoying to users who may be unable to reach certain areas of the system or to perform certain operations on the system due to a hue in the system. Once a software product is released to customers, all organizations should adhere to some process of managing customers problems for the ultimate purpose of customer satisfaction. A helpful tool is needed to ease the situation stated above.

1. **TRENDS IN BUG TRACKING SYSTEM**

Bug tracking database has been introduced to replace spreadsheet. In recent years, bug tracking systems are widely deployed in the information technology industry, especially in software development. Software and hardware companies use bug tracking systems during the development cycle to track bugs and design issues. Some use it during later phases of the product lifecycle to keep track of defect reports and to obtain quality indicators. The advantage is that this system contains database that stores all issues reported by users.

Previously, many older bug or defect tracking systems are client server based and it caused inconveniences to users at the same time. Users need to install the server, and each user need to install the client software. If external users were involved, it could be problematic because of issues like firewall blocking. Also, it is not always feasible to install client software in every user’s computer. All in all, to track bugs, many organizations still rely on Word documents and Excel spreadsheets, but these tactics are inefficient and error-prone to say the least. A good automated issue-tracking solution should streamline the process of raising, managing and fixing issues because bug tracking system should be useful software which can be designed to help software developers in keeping track of reported bug in their other software products.

Besides, this can be seen as one of the forms of issue tracking system. Actually, bug tracking is a misleading term because it is depending on what kinds of issues need to be tracked. The system can be called as issue tracking as issue is a broad term to describe most of the kinds of tasks might need to track when developing an application. The issue tracked might include bugs and feature-changes instead of tracking bugs or defects only. As it evolved along the time, this system is not just designed for internal teams such as developers and testers, but also for clients or other end users to report issues related to the system. This system enabled clear visibility of the software quality and improved the planning and estimating the research and development maintenance effort internally.

1. **RESEARCH ON OTHER SIMILAR CURRENT SYSTEMS**

Many companies use bug tracking system in developing software products because it is extremely useful and valuable. Bug tracking system is needed to store all related data together because most teams do not adequately maintain their requirements and design documents throughout the life of project. Bug tracking system provides important benefits to software development. Bug tracking database facilitates clear communication about bugs. Well- written, standardized reports tell the story much better than free form emails.

There are numerous bugs tracking tools available both commercially and as open source. Select the best bug tracking tool that fulfil the user’s requirement is not simple.

1. **REQUIREMENT ANALYSIS**

The requirement analysis is done in order to understand the problem the software system is to be solved. The emphasis in requirement analysis is on identifying what is need for the system not how will achieve its goal. The goal of requirement activities is to document requirement in a software requirement specification. There are two major phases in this phase Problem understood or analysis and Requirement specification.

In problem analysis the aim is to understand the problem and its context, the requirement of the system to be develop. Once problem analysis done and requirements understood. Requirements must be specified in requirement specification document. The requirement document must be specifying all function and performance of requirement. Format of input and output and design constraints due to political economic and environmental and security reasons. In other word besides functional specification all the factor the affect and design and proper functionality of the system must be specify in requirement document. Over all in requirement analysis we identify the requirement and constraints of proposal system. We can define requirement as “A condition of capability by a user to solve a problem or achieve an objective”

“A condition or a capability that must be met or possessed by a system…. to satisfy a contract, standard, specification, or other formally imposed document.”

1. **PROBLEM STATEMENT**

It is difficult to established proper communication among GPS Customer & Employees for GPSSIM product.

1. **EXISTING WORK**

Sales teams send mail to respective developers if found any issue during demo Sales teams register new customer and create own new defect.

Developers interact with Sales/Customer via mail chain If any major issue fixed /enhancement done inside GPSSim then there is mail chain send to all of existing customer regarding changes.

There are numerous bugs tracking tools available both commercially and as open source. Select the best bug tracking tool that fulfil the users’ requirement is not simple. Because Open Source tracking tool are not fulfilled all requirement of company customer and user Based on Company requirement commercial tools have more cost and not provide many required feature.

**CHAPTER 3**

**PROPOSED APPROACH**

1. **INTRODUCTION**

Create a central Portal to established proper communications among Sales / Customer / Employees and Admin.

Sales person is a GPS Employee who can creates Customer account & raised any defect during Customer demo activities.

Customers can log their defects & queries after login into this project. They can raise any suggestion & enhancements as well.

GPS Employees are L3 level Support teams who are responsible to interact with Customer to solve their queries / defects related to GPS Sim product.

Administrator can add Sales & Employee team into this project as well as responsible to do maintenance work along with reporting [Monthly reports to list down all defects based on priority (i.e. BLOCKER, CRITICAL, MAJOR, MINOR defects, etc.)]

1. **SOFTWARE SPECIFICATION**

**Table 3.1: Software Specification**

|  |  |
| --- | --- |
| **OS** | Windows 10 & Linux |
| **IDE & Other tools** | Eclipse |
| **Versioning System** | Tortoise SVN |
| **Database** | MySQL 5.x |
| **Web Server** | Tomcat 8.x |
| **Server-side Languages & other tools** | Java 8, JDBC, JSP, Servlet, Design Patterns (MVC & Singleton) |
| **Client-Side Languages & Other tools** | HTML, XML, CSS, JavaScript, jQuery, Bootstrap, JSON, AJAX |

1. **ROLE AND RESPONSIBILITY**

Develop UI Mock up for all modules using Pencil tool. Understand Project Architecture Deployment process. Implemented Sales, Customer, GPS Employee Modules Performance tuning (If required) Bug fixing and enhancements.

1. **SYSTEMS DEVELOPMENT LIFE CYCLE(SDLC)**

The meaning of SDLC is system Design life cycle. SDLC do the one of system design model like (waterfall model, spiral model etc.) the SDLC has some phases to design the software.

**Requirement Analyses.**

**Design**

**Implementation.**

**Testing**

**Deployment**

* **Requirement Analyses**

This the first phases to design the software. In this phase we gather the over idea or software related information from the different source’. After gather the information analyses the information and hold the useful information after gathering the useful send the information to the next phases.

* **Design**

In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model

* **Implementation**

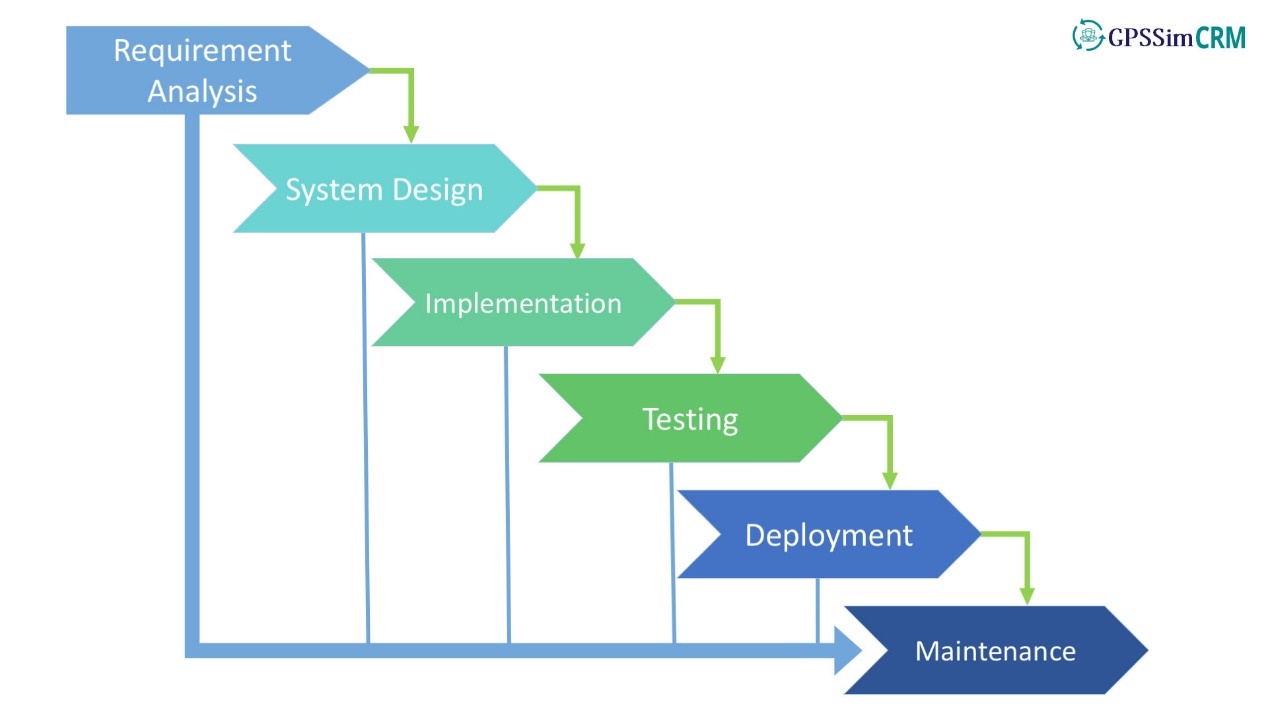
In this phase the code is produced so it is the main focus for the developer. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently. This is the longest phase of the software development life cycle.

* **Testing**

In this phase the testing team test the software by the helps of testing technic the software work properly than the testing team sends the software to the next phases If the software is not working properly than the testing team send the software to the Implementation phases. In the testing phases testing team find all the bug from the software.

* **Deployment**

This is the last phases of SDLC. In this phases deployment team do the demo in front to the customer. After the demo the Deployment team deploy the software to the customer.



**Fig 3.1: SDLC**

1. **DATA FLOW DIAGRAM (DFD)**

DFD (data flow diagrams) known as a bubble chart has the purpose of clarifying system requirement and identifying major transformation that will come programs in system design. So, it is starting point of the design phase that functional decompose the requirement speciation down to lowest level of details. A DFD consist of bubbles joined by the lines. The bubbles represent the data transformation and the lines represent data flow in the system.

DFD describes what data flow rather than how they are processed. The key question that we are trying to answer is: what major transformation must occur for input to be correctly transformed into output.

**Data Flow Diagram Symbol**

* **External Entity**

External entity

**Fig No. 3.2: Entity**

An external entity is a source or destination of a data flow. Only those entities which originate or receive data are represented on a business process diagram. The symbol used as a square containing a meaningful and unique identifier.

* **Process**

**Fig No. 3.3: Process**

A process shows a transformation or manipulation of data flows within the system. A circle represents a process that transfer incoming data flow into outgoing data flow

* **Dataflow**

**Fig No. 3.4: Data flow**

A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

* **Data store**

Data store

**Fig No. 3.5: Data Store**

A data store is a holding place for information within the system. It is represented by an open-ended narrow rectangle.

**Steps to Construct Data Flow Diagrams**

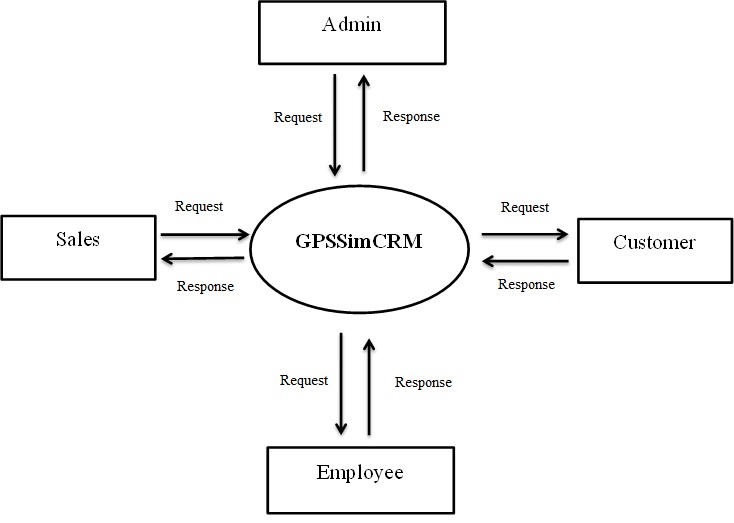
Four steps are commonly used to construct a DFD

* Process should be named and numbered for easy reference. Each name should be representative of the process.
* The direction of flow is from top to bottom and from left to right.
* When a process is exploded into lower-level details they are numbered.
* The names of data stores, sources and destinations are written in capital letters.

**Rules for constructing a Data Flow Diagram**

* Arrows should not cross each other.
* Squares, Circles and files must bear names.
* Decomposed data flow squares and circles can have same names. Choose meaningful names for dataflow.
* Draw all data flows around the outside of the Diagram.
* **Level 0 DFD**

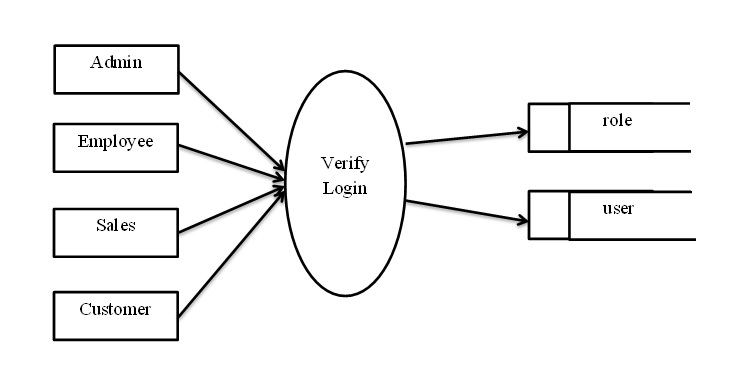
This Level 0 DFD describe the structure of project how to all module get request and responses.



**Fig No. 3.6: 0 Level**

* **Level 1 DFD**

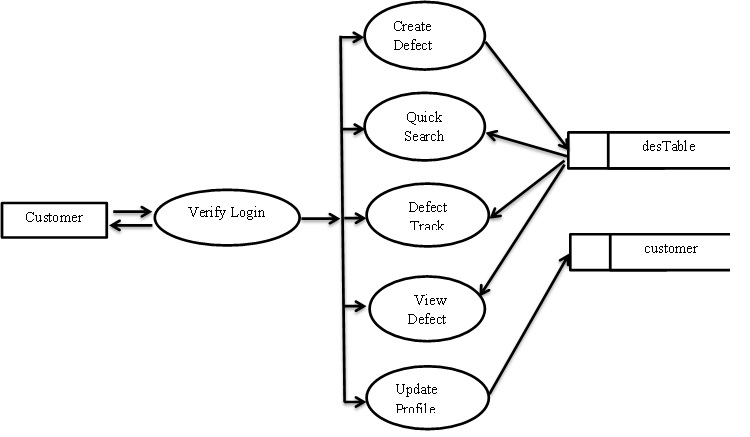
One level DFD defined the how to verify the user.



**Fig No. 3.7: 1 Level**

* **Level 2 DFD**

Two level DFD describe the work flow of every module transition.



**Fig No. 3.8: 2 Level**

1. **FUNCTIONAL SPECIFICATION**

**Customer**

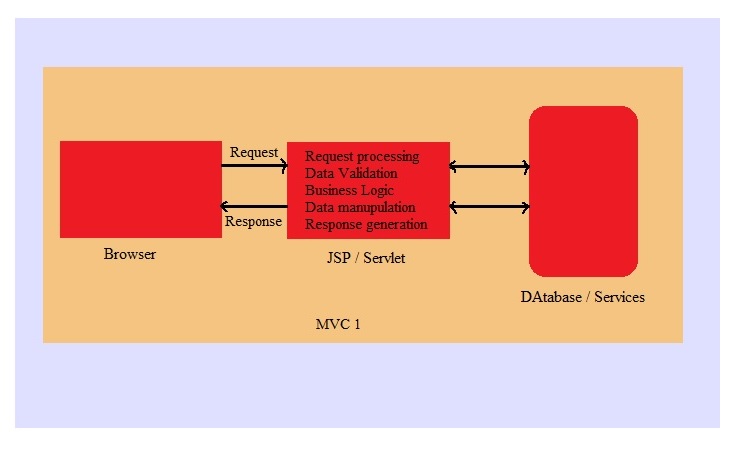
Customer interact with Sales personally or via mail chain. Customer can also interact with employee via mail. The customer can view existing defect list and comment.

If a customer faces the minor defects then customer view old CLOSED defect or raised defect and search the old defect solution. Customer can view the list of all own defect and check all defect status for example defect is OPEN, IN-PROGRESS or CLOSED.

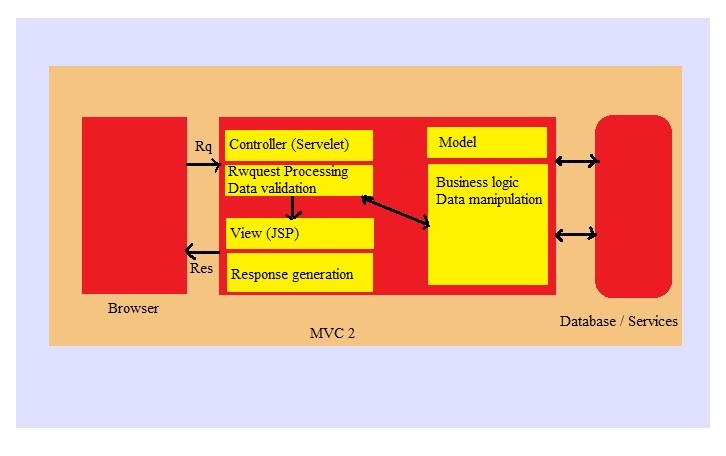
Customer can search the defect by using ID or keyword or ticket number. Customer can contact a salesperson about the product and the version of the product, quality and latest features of the product. Customer can update their profile also customer can change their password by using Email. Customer can raise new defects / suggestion / Enhanced update images while creating / updating defect. Customer can provide feedback about product to sales. Customer give any suggested to sales or employee for improvement of software quality.

1. **MVC Architecture**

MVC architecture defined which approach based our project design like algorithm, method or patterns.



**Fig No. 3.9: MVC1**



**Fig No. 3.10: MVC2**

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 USE CASE DIAGRAM**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

**Basic Use Case Diagram Symbols and Notations**

**System:** Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



**Fig 4.1: System Name**

**Use Case:** Draw use cases using ovals. Label the ovals with verbs that represent the system's functions



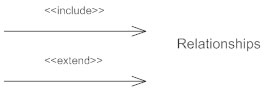
**Fig 4.2: Use Case**

**Actors:** Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



**Fig 4.3: Actor**

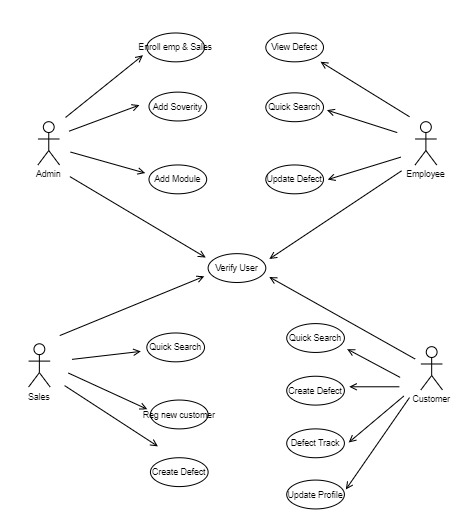
**Relationships:** Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labelled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



**Fig 4.4: Relationships**

**General use case diagram of GPSSimCRM**

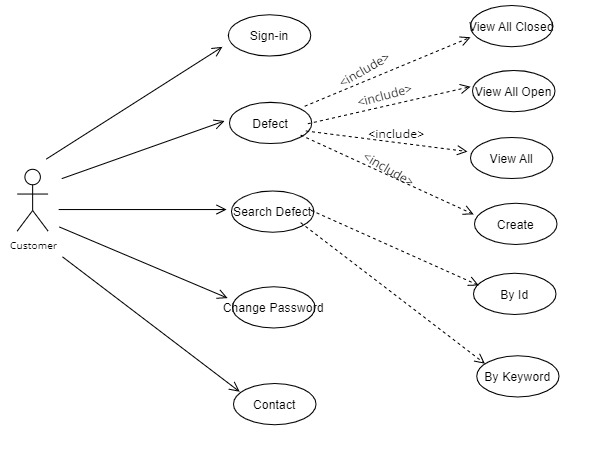
This figure describes the flow of four module. What is the working of All module admin can add enrolled employee can update sales can review of product customer can reset the defect.



**Fig 4.5 General use case diagram of GPSSimCRM**

**Customer work flow use case diagram**

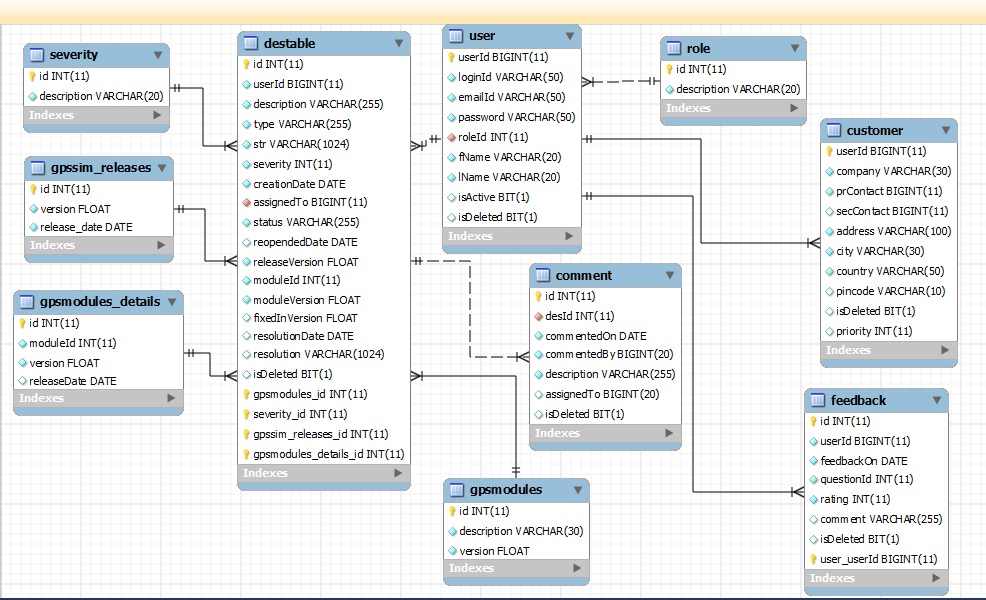
Customer can raise defect and view and update defect search the defect after searching check defect status. Like OPEN, CLOSED or IN-PROGRESS.



**Fig 4.6: Customer use case diagram**

**4.2 ER Diagram**

It describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.



**Fig No. 4.7: ER Diagram**

**4.3 DATABASE DESIGN**

Database design define the database table design in which form data is store

**User**

User table describe the data of GPSSimCRM user

**Table No. 4.1: User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| userId | Bigint (11) | Primary Key | Store userId |
| loginId | Varchar (50) | Not Null | Store loginId |
| Email | Varchar (50) | Not Null | Store Email |
| Password | Varchar (50) | Not Null | Store Password |
| Roll Id | int (11) | Foreign Key | Store Roll Id |
| FName | Varchar (20) | Not Null | Store FName |
| LName | Varchar (20) | Not Null | Store LName |
| IsActive | Boolean | Allow Null | Store Bit Val |
| IsDeleted | Boolean | Allow Null | Store Bit Val |

**Role**

Role table describe the role of user, who user performed which role

**Table No. 4.2: Role**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store id |
| Description | Varchar (20) | Not Null | Store Description |

**Comment**

Comment table define the format of storing of user table.

**Table No. 4.3: Comment**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary key | Store id |
| desId | int (11) | Foreign key | Store desId |
| commentedOn | Date | Not null | Store commentOn |
| commentedBy | Bigint (20) | Not null | Store commentBy |
| Description | Varchar (255) | Not null | Store description |
| assignedTo | Bigint (20) | Allow null | Store assignedTo |
| isDeleted | Boolean | Allow null | Store bit val |

**Feedback**

Feedback table store the details of updating module.

**Table No. 4.4: Feedback**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary key | Store id |
| userId | bigint (11) | Foreign key | Store userId |
| feedbackOn | Date | Not null | Store feedbackOn |
| questionId | int (11) | Not null | Store questionId |
| Rating | Int (11) | Not null | Store rating |
| Cmment | Varchar (255) | Allow null | Store comment |
| isDeleted | Boolean | Allow null | Store bit val |

**Feedback Question**

Feedback question table store feedback question details.

**Table No. 4.5: Feedback Question**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store Id |
| Description | Varchar (255) | Not Null | Store Description |

**DesTable**

DesTable store defect details.

**Table No. 4.6: DesTable**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store Id |
| userId | Bigint (11) | Foreign Key | Store userId |
| Description | Varchar (255) | Not Null | Store description |
| Type | Varchar (255) | Not Null | Store Type |
| Str | Varchar (1024) | Not Null | Store str |
| Severity | int (11) | Not Null | Store Severity |
| Creation Date | Date | Not Null | Store Creation |
| AssignTo | Bigint (20) | Not Null | Store Assign to |
| Status | Varchar (255) | Foreign Key | Store status |
| Reopen Date | Date | Allow Null | Store reopen date |
| Release Version | Float | Not Null | Store release ver |
| moduleId | Int | Not Null | Store module id |

**Gpsmodule**

Gpsmodule table store module details.

**Table No. 4.7: Gpsmodule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store id |
| Description | Varchar (30) | Not Null | Store description |
| Version | Float (4) | Not Null | Store version |

**Gpssim release**

Gpssim release table store release date details

**Table No. 4.9 Gpssim release**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store id |
| Version | Float (4) | Not Null | Store version |
| Release\_date | Date | Not Null | Store release date |

**Severity**

Severity table store the module conditions details.

**Table No. 4.10: Severity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| Id | int (11) | Primary Key | Store id |
| Description | varchar (20) | Not Null | Store description |

**Customer Table**

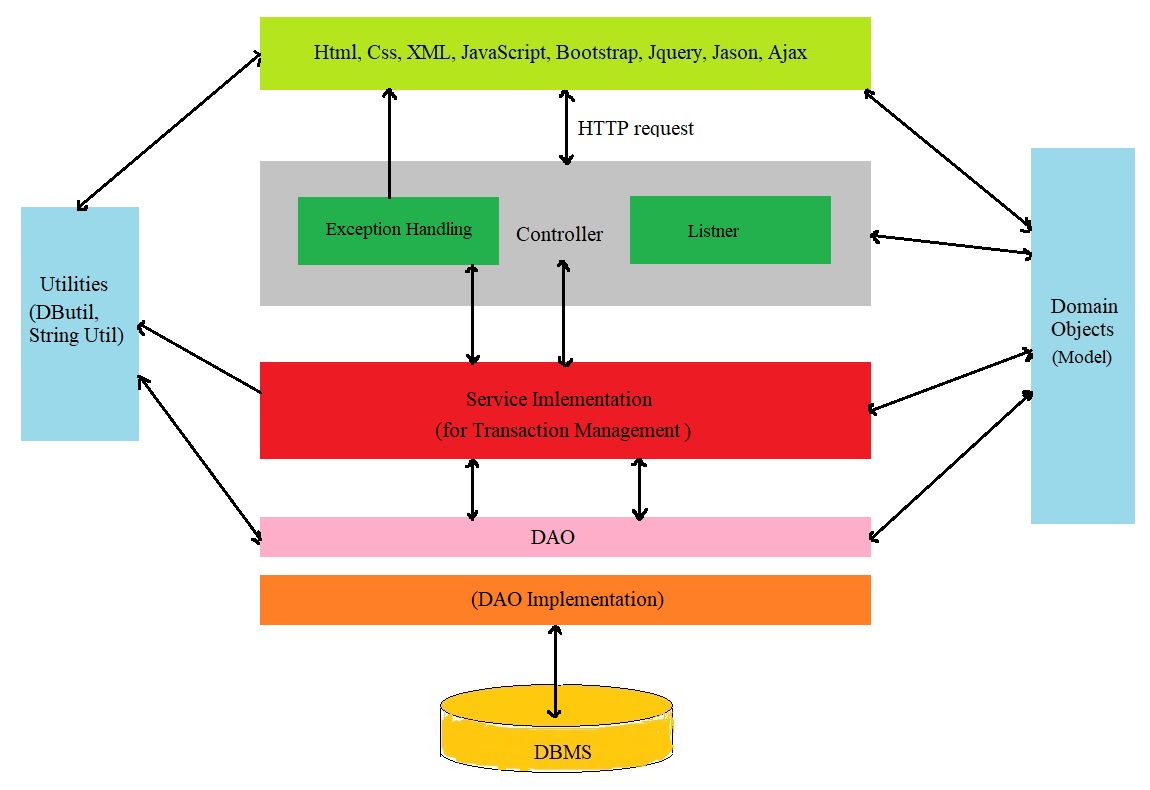
Customer Table store the details of customer.

**Table No. 4.11: Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type (size)** | **Constrains** | **Description** |
| userId | bigint (11) | Foreign key | Store userId |
| Company | Varchar (30) | Not Null | Store company |
| prContact | bigint (11) | Not Null | Store pri contact |
| secContact | bigint (11) | Allow Null | Store sec contact |
| Address | Varchar (100) | Not Null | Store address |
| City | Varchar (30) | Not Null | Store city |
| Country | Varchar (30) | Not Null | Store country |
| Pincode | in (11) | Allow Null | Store pincode |
| isDeleted | Boolean | Allow Null | Store bit val |
| Priority | Int | Allow Null | Store priority |

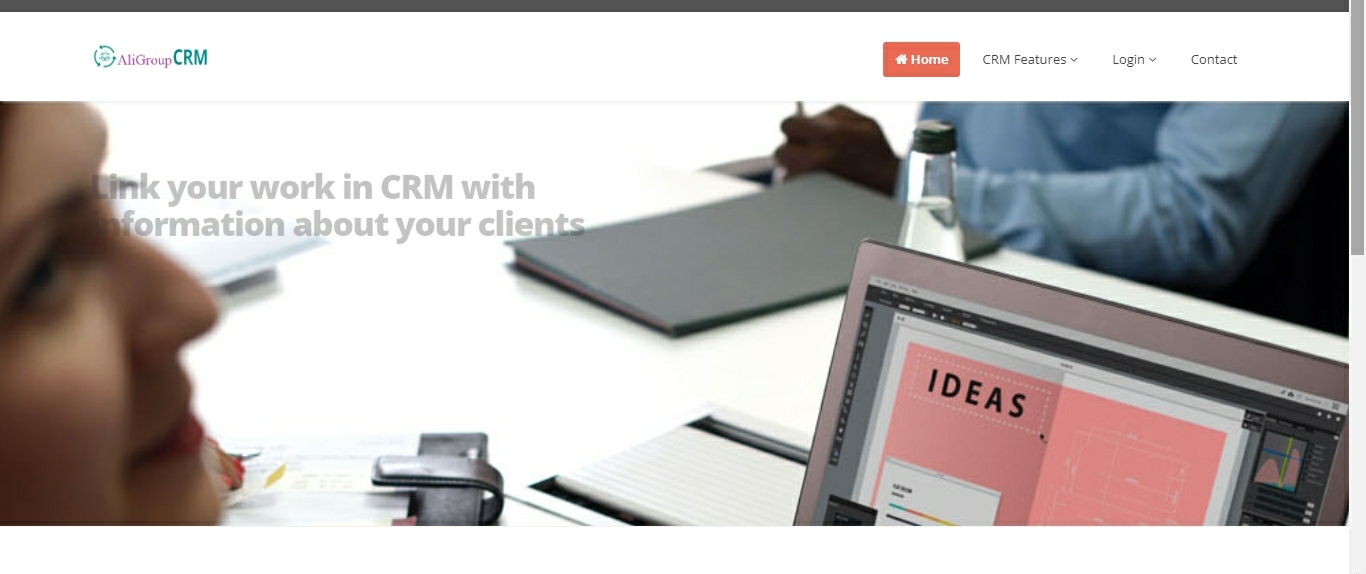
**4.4 PROJECT ARCHITECTURE**

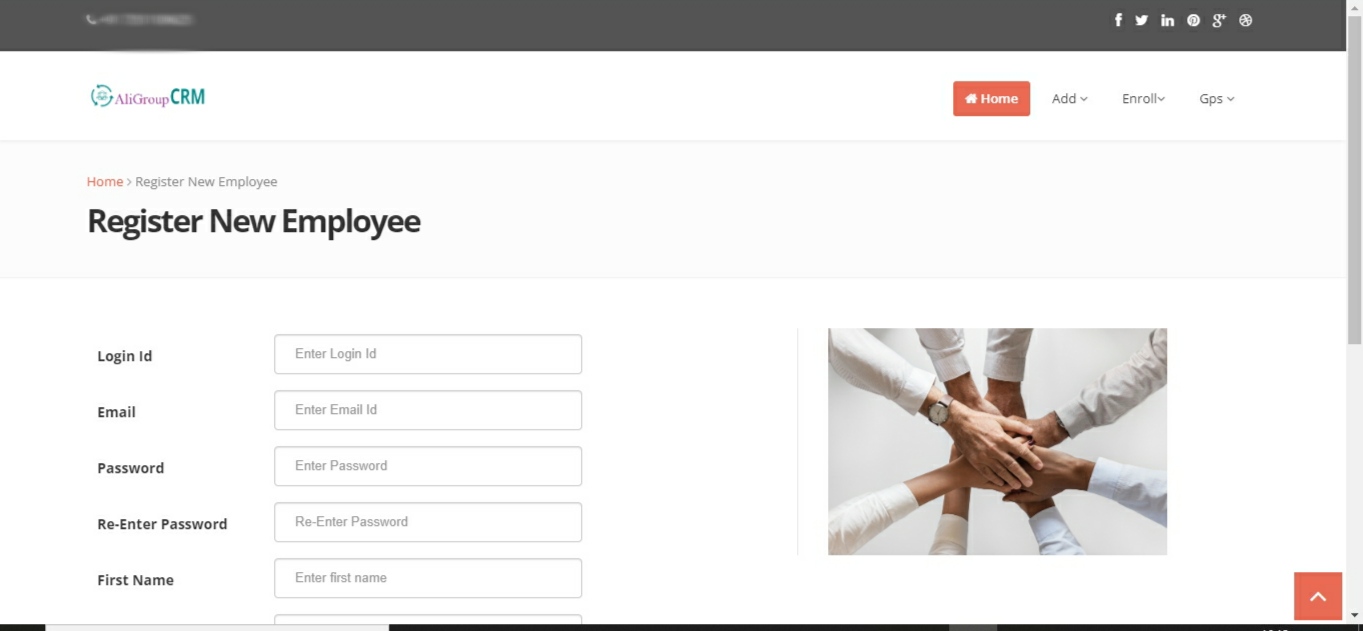
This project architecture defines the structure of the project what is the work flow of the project how to user request to controller or what is the HTTP request and response. The architecture defines how to handle exception handling. This fig defines how to get the data domain architecture.

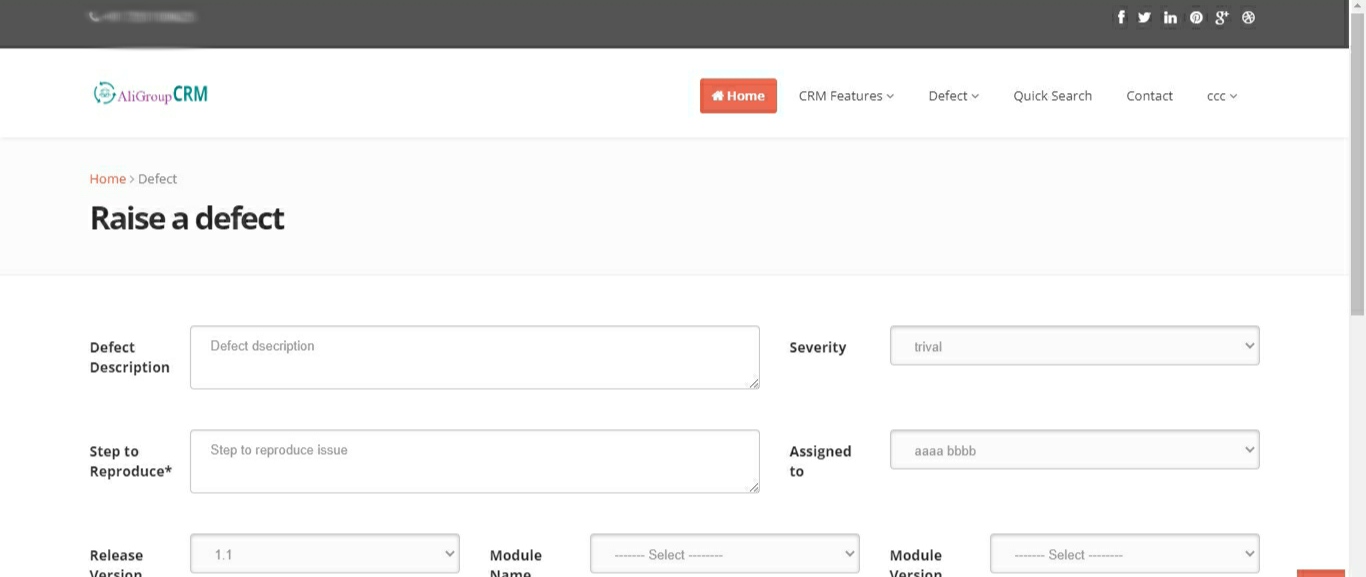


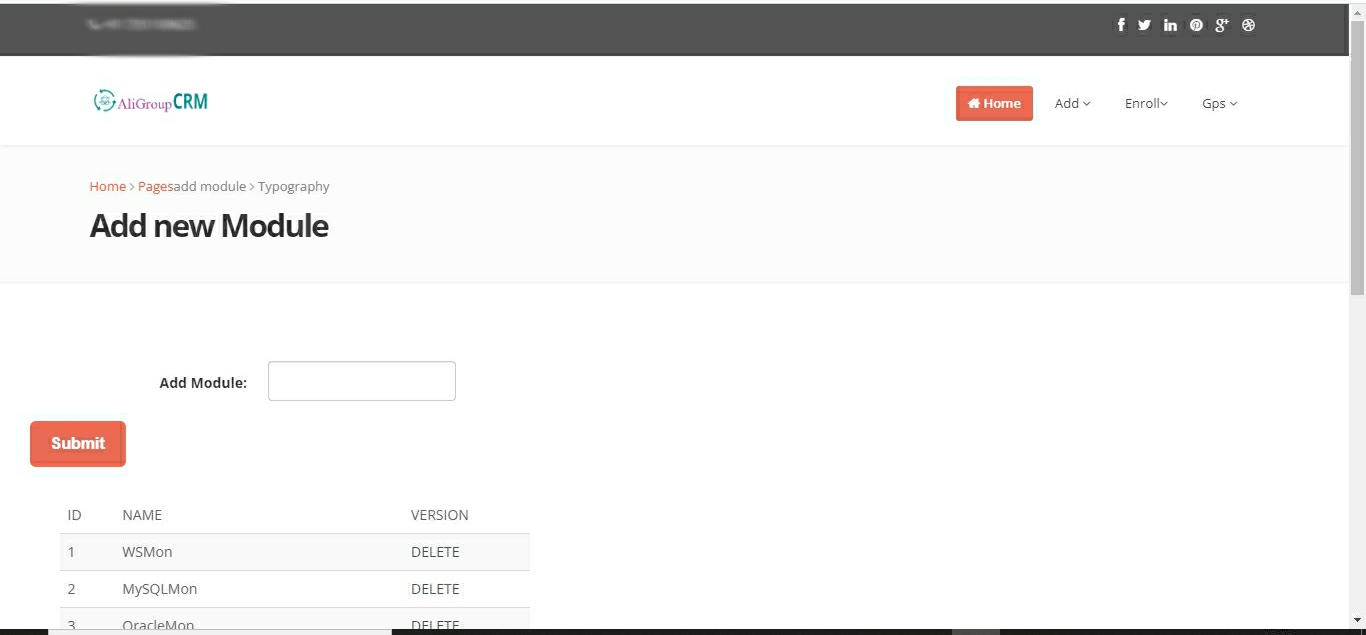
**Fig No. 4.8: Project Architecture**

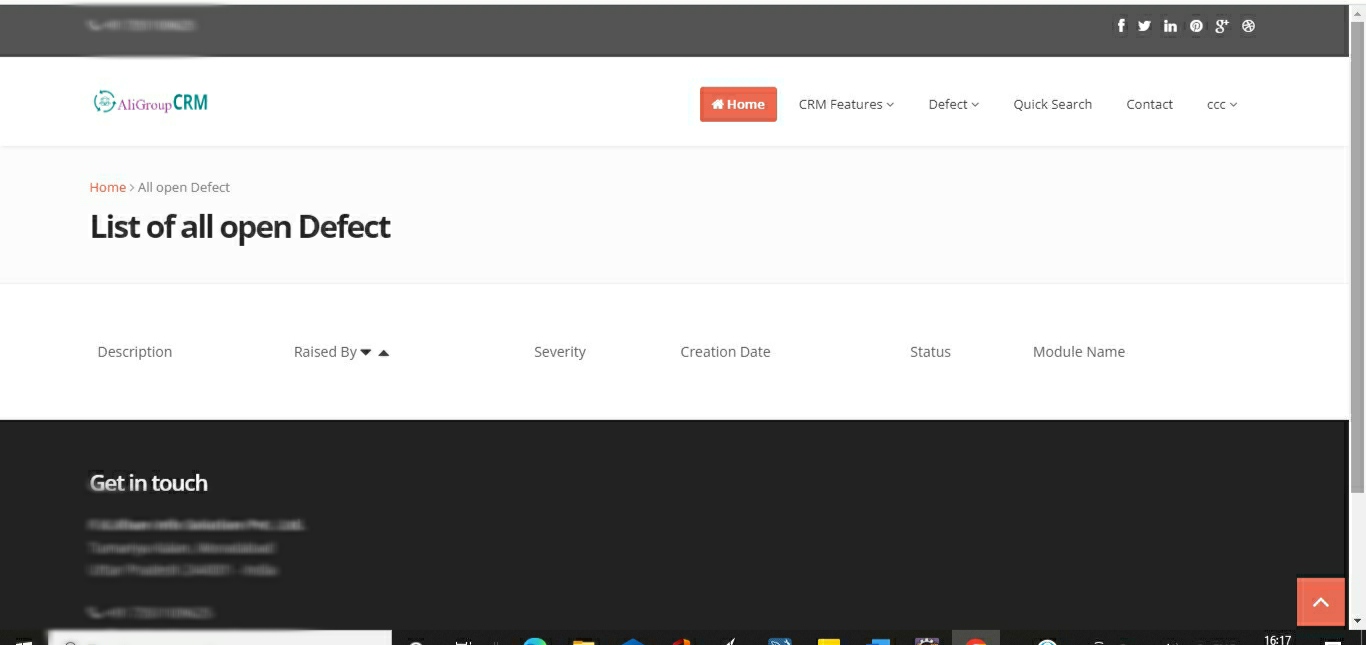
**4.5 User Interface**

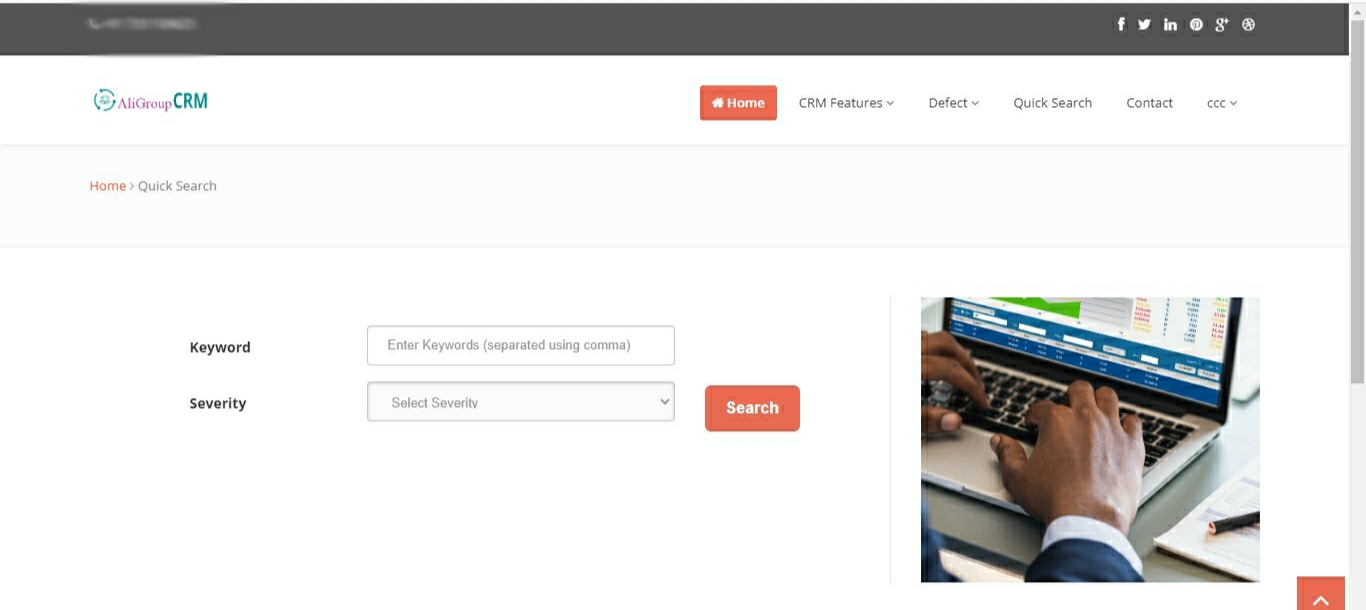


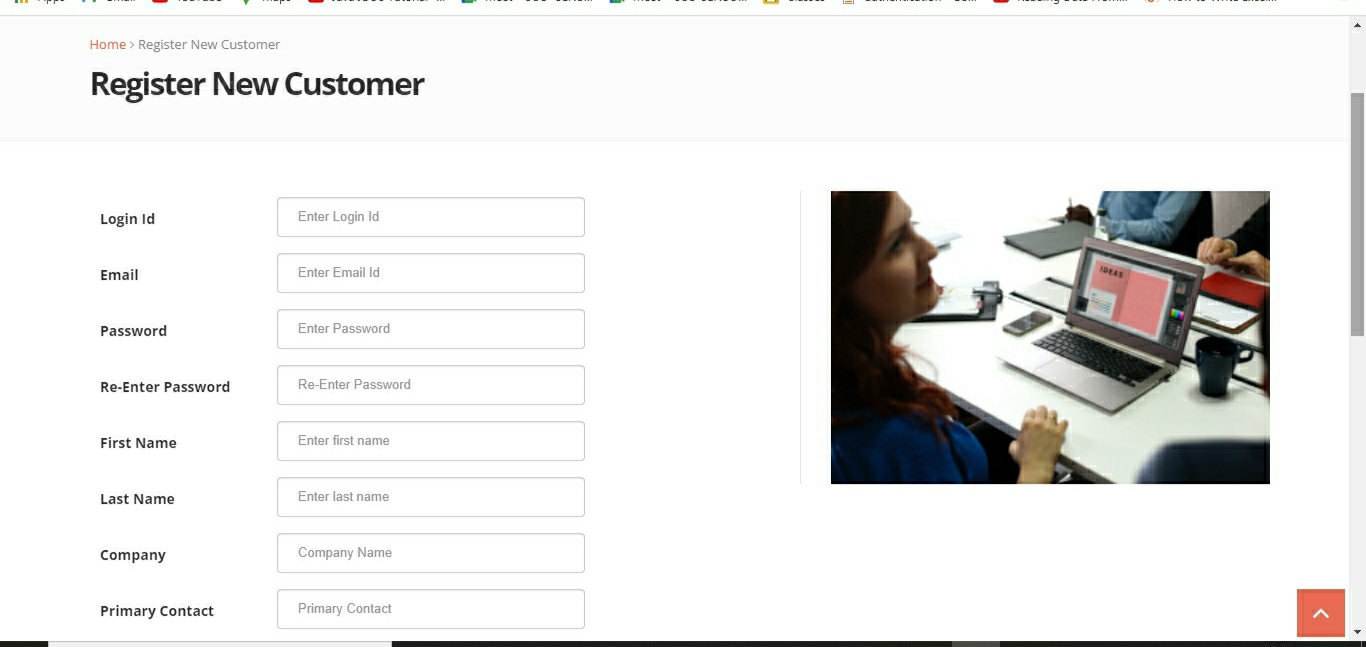


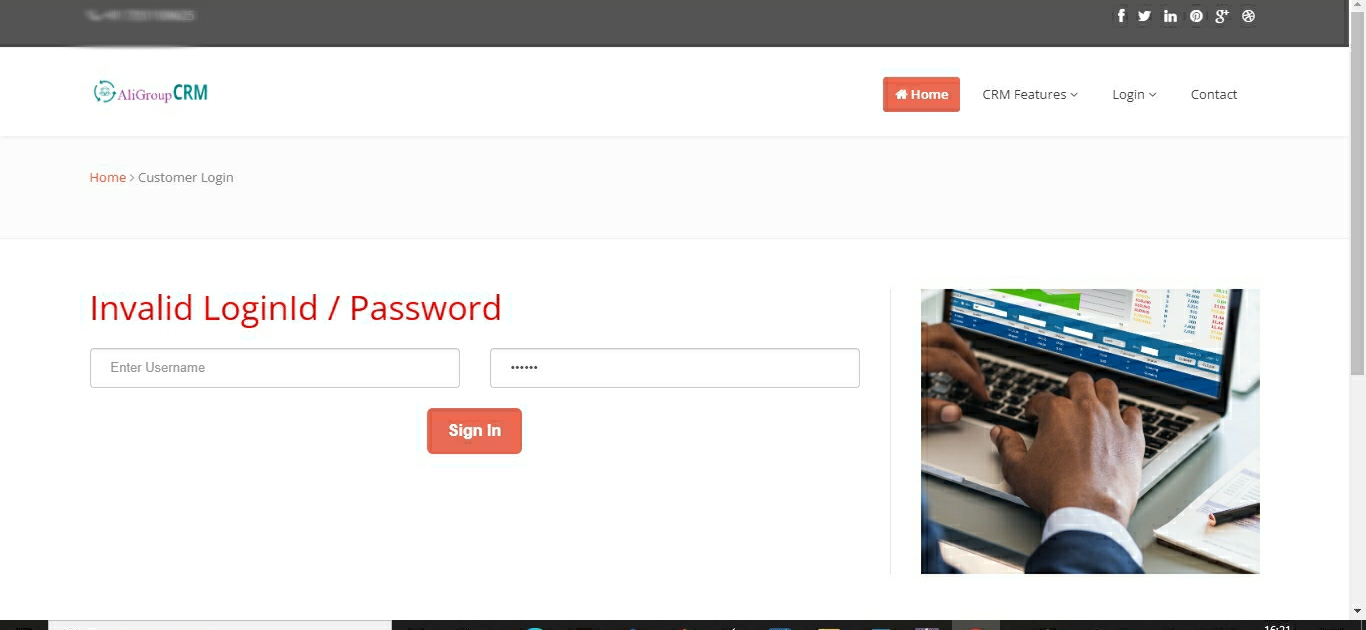












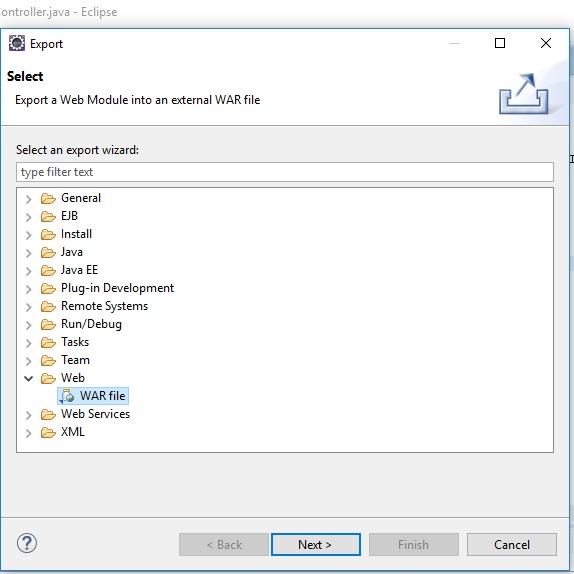
**CHAPTER 5**

**DEPLOYMENT AND DEMO**

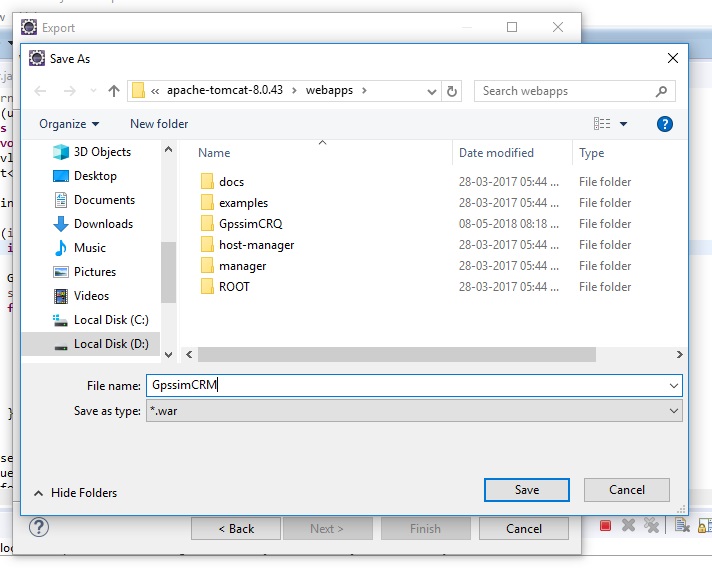
**Deployment and Demo Exporting or deploy the web project as WAR**

After completing the development of a dynamic Web project, you have the option to export it as a WAR file for further usage in your production environment. A Web Archive (WAR) file is a portable, packaged Web application that you can import into your workspace. To export your dynamic Web project:

* Right-click the Web project.
* Select **Export...**
* Choose **Web > WAR file**.



**Fig No. 6.1: WAR file**



**Fig No. 6.2: WAR Name**

Click **Save**

Now open Apache Tomcat Startup window batch file after that open browser and type localhost url and hit Enter.

**CHAPTER 6**

**TEST CASES**

### **Test Cases – Login Page**

| **Sr. No.** | **Test Cases** | **Type- Negative/ Positive Test Case** |
| --- | --- | --- |
| 1 | Verify if a user will be able to login with a valid username and valid password. | Positive |
| 2 | Verify if a user cannot login with a valid username and an invalid password. | Negative |
| 3 | Verify the login page for both, when the field is blank and Submit button is clicked. | Negative |
| 4 | Verify if the data in password field is either visible as asterisk or bullet signs. | Positive |
| 5 | Verify if a user is able to login with a new password only after he/she has changed the password. | Positive |
| 6 | Verify if the login page allows to log in simultaneously with different credentials in a different browser. | Positive |
| 7 | Verify if the ‘Enter’ key of the keyboard is working correctly on the login page. | Positive |

### **Test Cases – Signup**

| **Sr. No.** | **Test Cases** | **Type- Negative/ Positive Test Case** |
| --- | --- | --- |
| 1 | Check user should Register by filling all the required fields | Positive |
| 2 | Check the Email text field that has an Email address without @ symbol. • Check the Email text field that has a random string instead of a real email. • Check the Email text field that has @ symbol written in words. • Check the Email text field that has a missing dot in the email address. | Positive |
| 3 | Verify if the password required rules are not satisfied in the password | Positive |
| 4 | Check the required fields by not filling any data | Negative |
| 5 | Check all the text boxes, radio buttons, buttons, etc. | Positive |

**CHAPTER 7**

**RESULT AND DISCUSSION**

**7.1 INTRODUCTION**

In this chapter we describe the working of project and define the testing phases of projector Maintenance of project. In this phase define the challenge comes at the time of project development time.

**7.2 TESTING**

Testing is the most important aspect of any project. A web application becomes more crucial since the information is to be viewed by thousands of users. It is essential for developer to make sure that the web presentation work well from technical and well as usability side. We have performed the module level testing and content verification before going for integrity and usability testing.

**Manual testing**

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behaviour. To ensure completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases.

**Unit testing**

In computer programmingunit testing is a software testing method by which individual units of source code sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application

**Functional testing**

Functional testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional testing usually describes *what* the system does.

**7.3** **MAINTENANCE**

**Defect and Enhancement**

In our project at the time of testing access deferent user by one id that is the defect of the project in this case the testing team send the project again to the developed or implementation team then implementation team solve this problem and send to the testing team and employee get the data from the data base but they cannot Update the comment this is the big defect comes at the testing. testing team send the defect to the implementation team for solving the defect.

**7.4** **CHALLENGES**

At the time of developing the project we face many challenges such as design initial project architecture as per MVC design pattern understanding the MVC1 and MVC2 architecture.

MVC1 in which we understand the working of model, view, and controller and in MVC2 architecture we work on the layers such as presentation layer (In this layer we developed the presentation logic for user interface), service layer (In this layer we write services for each module), Data Access layer by the help of Database access layer we access the data in the database all the data related queries written within this layer.

Understand the working of Jason, jQuery, Servlate etc. understand the working of all the modules use in the project like as admin, employee, customer, sales. how to raise the defects for customer.

Understand the working of many software such as pencil software, Eclipse IDE, Apache tomcat etc.

**CHAPTER 8**

**CONCLUSION**

The goal of our project GPSSimCRM is to bring together all of the company’s relevant departments so they can work together to maintain customer relationships. The Customer Relationship Management has simplified the handling of customers in many industries.

Our project does work properly for tracking the defect and helping us to deliver the solutions of defects.

Our Project enables a company to align its strategy with the needs of the customer in order to best meet those needs and thus ensure long-term customer loyalty.

CRM is a systematic approach towards using information and ongoing dialogue to built long lasting mutually beneficial customer relationship.

**CHAPTER 9**

**FUTURE WORK**

**FUTURE ENHANCEMENT**

* Apply Data encryption like in Password using DES algo with Base 64 api from Apache.
* Apply unique key generation for user id using MD5 digest from Java Security
* Enable Https port.
* Add online help support manual.
* Integrate GPSSimCRM with SVN repository so that anyone can easily see all changes against any fixes.
* Create android web view project to deploy same as Android App.
* Apply central checks using Servlet Filters for Authentication and Authorization
* Add various filter criteria to view different types of defects.

**CHAPTER 10**

**REFERENCES**

1. Java Developers' sites, in particular, http://java.net.

2. The online Java tutorial http://docs.oracle.com/javase/tutorial/.

3. Javatpoint http://javatpoint.com

4. ECMAScript (JavaScript) Specification: "Standard ECMA-262 ECMAScript Language Specification 5.1“

5. jQuery Tutorial @ https://learn.jquery.com.

6. MySQL Mother Site @ www.mysql.com.

7. MySQL Employees Sample Database @ http://dev.mysql.com/doc/employee/en/index.html.