Chapter 1

INTRODUCTION

CAR SHOWROOM MANAGEMENT SYSTEM is a software application to maintain day to day activities in the store, This software help to maintain the record of the bike, customers, workers and sales information Admin manages, adds, updates and deletes the cars, sales persons and admin records. Admin also can add new admins. The software is designed to handle, the daily activities of all the stores. Search the details when required.

1.1 PURPOSE

Provides a simple way of accessing the database. Allows the admin to make search and find certain bikes, customers and sales records which are linked through tin numbers of different stores located at different locations. Provides centralized control of store database. Holds complete information of the customer.

1.2 PROJECT SCOPE

The application is designed and developed so that there will be elimination of storing the records and other details in a hardcopy thereby supporting eco-friendly nature. Bike Showroom Management System application is quiet simple and easy to use. This application has no much Complications. It has a very simple way of maintaining the store records such as storing the bike Details of a certain store, customers visited at a certain store and the sales details of the store that the admin choose to gain access.

1. 3 Project Goal

The aim of this project is to show the shadow implementation of JAVA & SQL which include better storage facility, update, and view of data. The application provides a user friendly interface so that the admin can access with ease.

Chapter 2

REQUIREMENTS SPECIFICATION

2.1 Software Requirements

FRONT END : Java NetBeans IDE 8.2

BACK END : Java NetBeans IDE 8.2, SQL

OPERATING SYSTEM : Windows 10

Java program as a front-end application

Java scripts to create database through JDBC correctly and retrieves data by executing the expected java program.

2.2 Hardware Requirements

This packages has been developed on:

Processor : Intel core i5.

Hard Disk : 20 GB, 80 GB, 160 GB or above.

Monitor : 15.6 VGA color, 1366*768 resolution

RAM : 2 GB or above.

Input device : Keyboard and Mouse.

Chapter 3

ER DIAGRAM AND SCHEMA DIAGRAM

3.1 ER DIAGRAM:

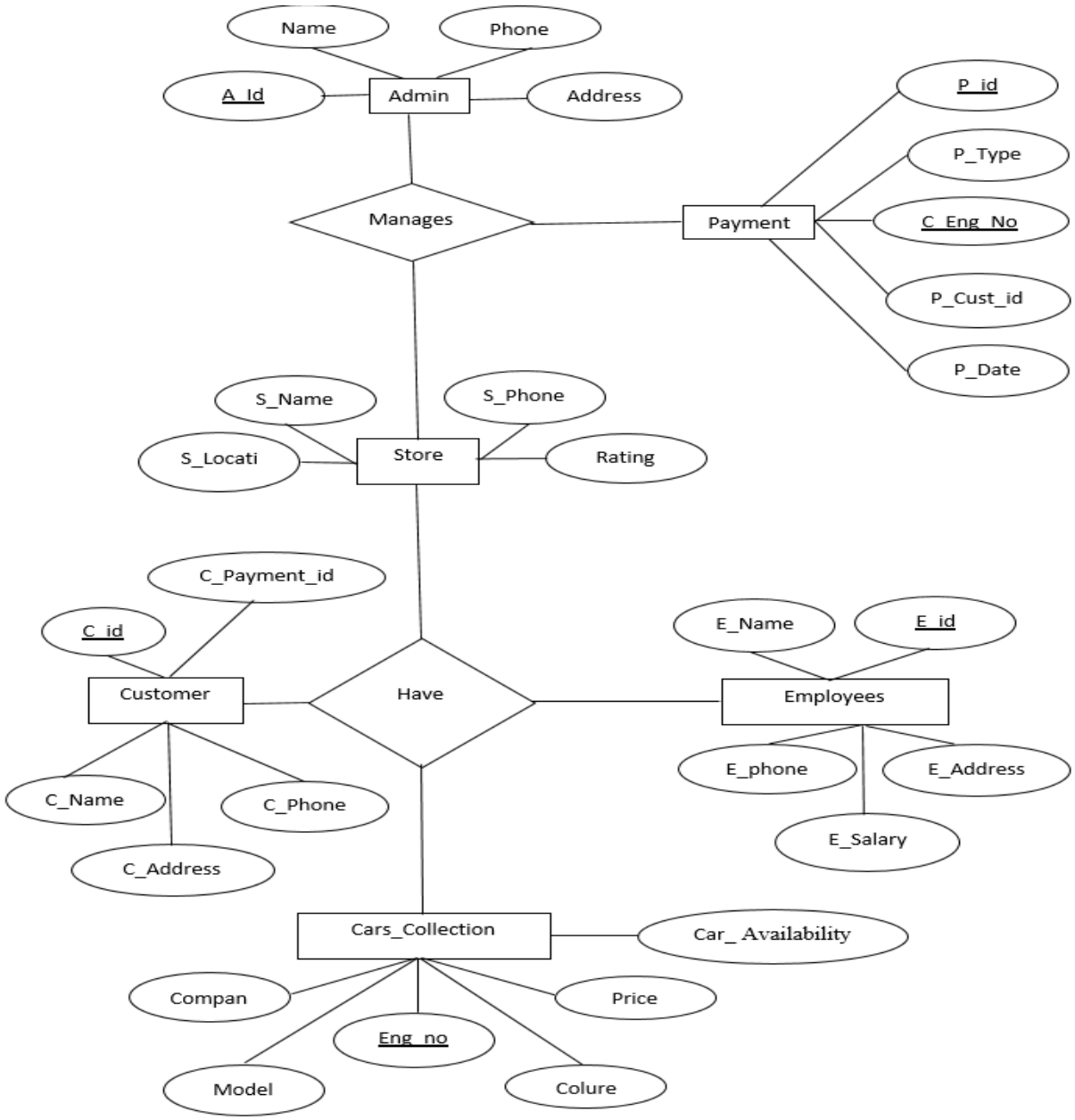

Fig 3.1: ER Diagram

An Entity Relationship Diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of database management system, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, Entity Relationship Diagram shows the complete logical structure of a database.

3.2 SCHEMA DIAGRAM

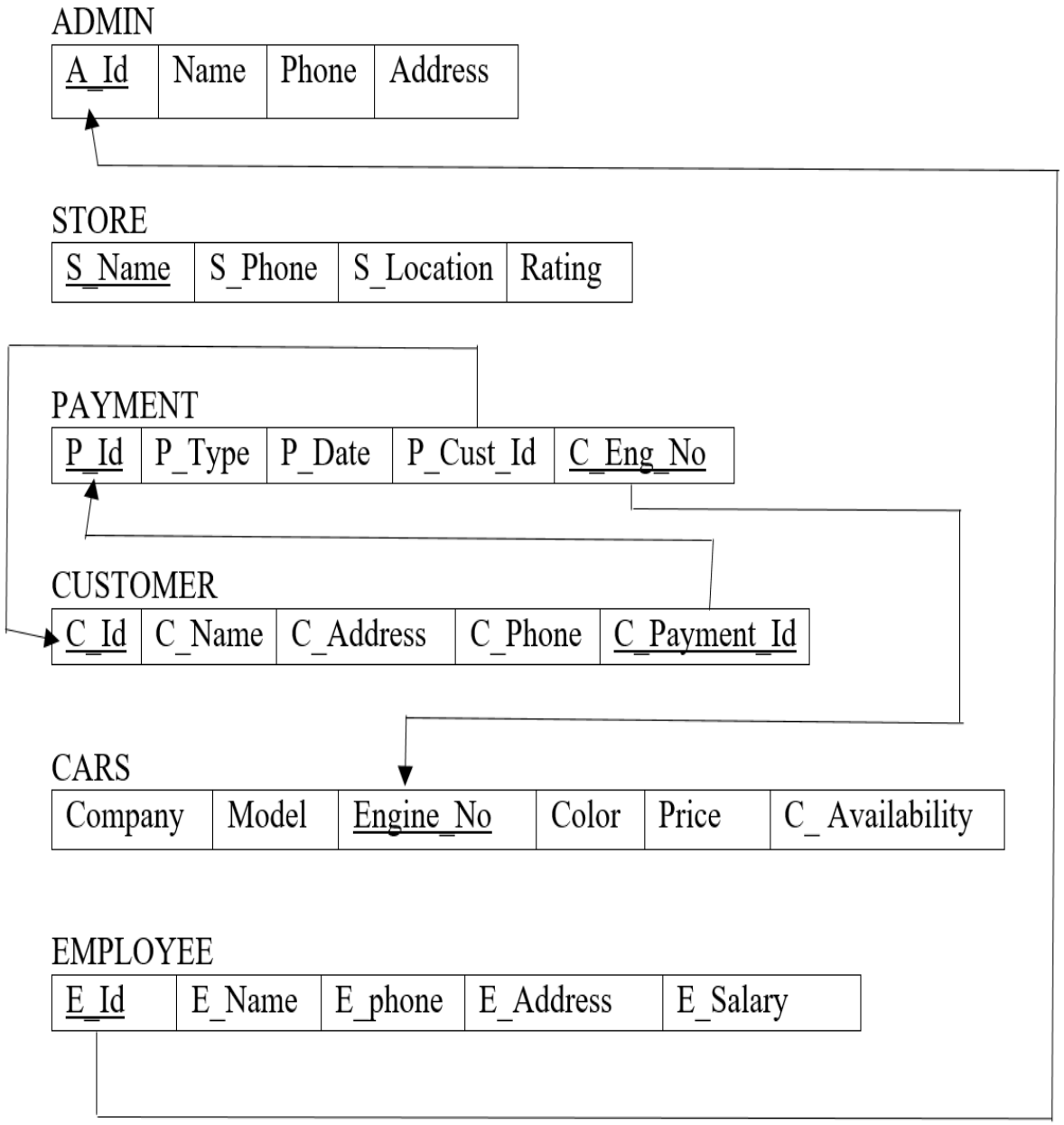

Fig 3.2: Schema Diagram

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers to understand the database and make its useful.

Chapter 4

TESTING

Testing is a dynamic technique of verification and validation. It involves executing an implementation of the software with test data and examining the outputs of the software and its operational behavior to check that it is performing as required.

The following statements serve as the objectives for testing:

1. Testing is a process of executing a program with the intent of finding error
2. A good test case is one that has a high probability of finding an as-yet undiscovered error.
3. A successful test is one that uncovers as-yet undiscovered error.

Verification and validation is intended to show that a system confirms to its specification and that the system meets the expectations of the customer. Verification involves checking that the software confirms to its specification. We should check that the system meets its specified functional and non-functional requirements. Validation ensures that the software meets the expectations of the customer. It goes beyond checking conformance of the system to its specification to showing that the software does what the customer expects as distinct from what has been specified.

The testing process should proceed in stages where testing is carried out incrementally in conjunction with system implementation. System components are tested, the integrated system is tested and, finally, the system is tested with the customer's data.

The stages in the testing process are:

4.1 Unit testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components. Thus the project has been successfully tested.

Table 4.1: Table for unit testing test case and expected output

TEST NO.	TEST CASES	ACTUAL OUTPUT	EXPECTED OUTPUT	REMARKS
1.	Admin Login	Gets logged in	Successfully logged in	Success
2.	Salesperson Login	Gets logged in	Successfully logged in	Success
3.	Insert Car Details	Inserted Successfully	Inserted Successfully	Success
4.	Delete Car Details	Deleted Successfully	Deleted Successfully	Success
5.	Update Car Details	Update Successfully	Update Successfully	Success
6.	View Car Details	Car Details Are Viewed	View Successful	Success
7.	Add / Remove Admins	Admin Added/Removed Successfully	Admin Added/Removed Successfully	Success
8.	Add / Remove Salesperson's	Salespersons Added/Removed Successfully	Salespersons Added/Removed Successfully	Success
9.	View Store Progress	Progressed viewed successfully	Progressed viewed successfully	Success
10.	Generate/View Final Bill	Generated/Viewed Final Bill successfully	Generated/Viewed Final Bill successfully	Success

4.2 Integration testing.

- Main function is design to call many sub functions, where different options are given in the sub functions. Main function in this software is to maintain a music store details like which albums are present in the shop. Albums are of two types- movie album and singer album.
- Now, different functions are included in the main separately and tested for error. Album details, customer details and sales details are viewed properly.
- The software was compiled and tested and desired output was obtained without any error or exception.

4.3 Sub-system testing

This phase involves testing collections of modules which have been integrated into sub-systems. The sub-system test process should concentrate on the detection of module interface errors by rigorously exercising these interfaces.

4.4 System testing

The sub-systems are integrated to make up the system. This process is concerned with finding errors that result from unanticipated interactions between sub-systems and sub-system interface problems. It is also concerned with validating that the system meets its functional and non-functional requirements and testing the emergent system properties. The software is designed and developed so that it can be run on a system without any errors. The software is tested for this system and its functionality is achieved.

4.5 Acceptance testing

This is the final stage in the testing process before the system is officially accepted for operational use. The system is tested with data supplied by the system customer rather than simulated test data. Objective is to insert, retrieve, delete, and update salesperson details and customer's details. The main objective of this software is to generate the bill for the customer and print it.

Chapter 5

SNAPSHOTS AND RESULTS

The set of tables is created using the relational database for the identified entities at the design stage. The uniqueness of the data fields in these tables are established using primary key, while the relationships are maintained using foreign keys.

The web pages of the information will guide the use and operation of this system. Figure 5.1 illustrate the login home page for our ERP College Management.

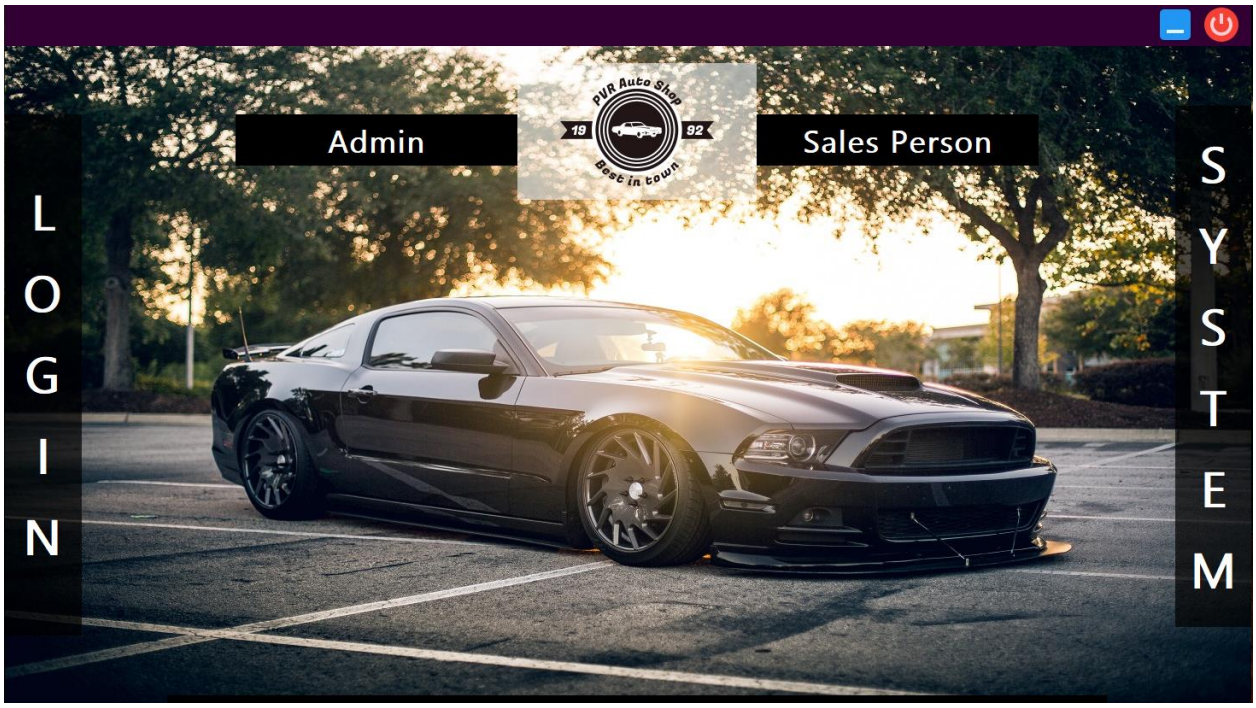

Fig 5.1 Login form

The fig 5.1 shows login page for the CAR Showroom Management System. When Username and password is correct it shows successful and when Username and Password is wrong it shows unsuccessful. The login button is used to enter the next table and continue the process. Then the close button is used to close the Login Table.

Fig 5.2 Admin Home Page

The fig 5.2 shows the details on the admin home page. The admin can access various field such as car store, employee details, and can monitor the progress of the store. They can also view their respective profiles. The admin can also generate bill using the billing system.

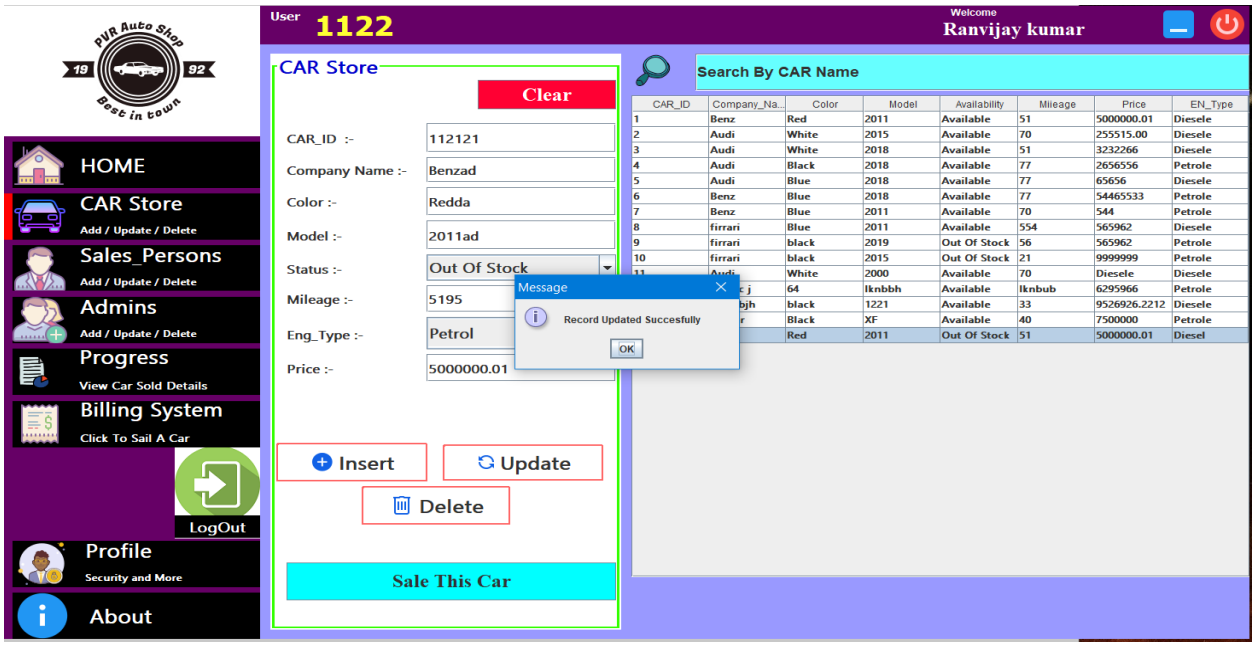

Fig 5.3 Car Store

The fig 5.3 shows the details of the car store. The admin can add information about a new car, can update the information of existing car and delete the information about existing car. By clicking on the “Sell this car”, the admin will be directed to the billing system.

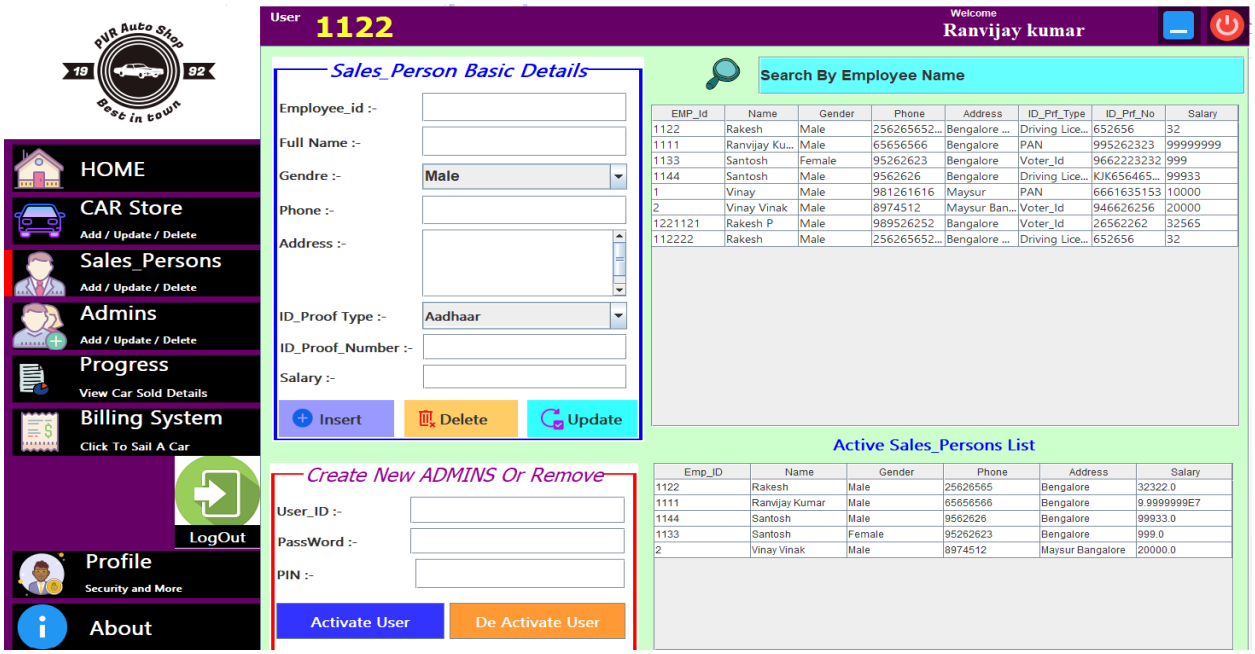

Fig 5.4 Sales Persons Page

The fig 5.4 shows the details of the Sales persons Information. The admin can add information about a sales person, can update the information of sales person and delete the information about sales person. By clicking on “de activate user”, the admin can restrict the sales person to access the system.

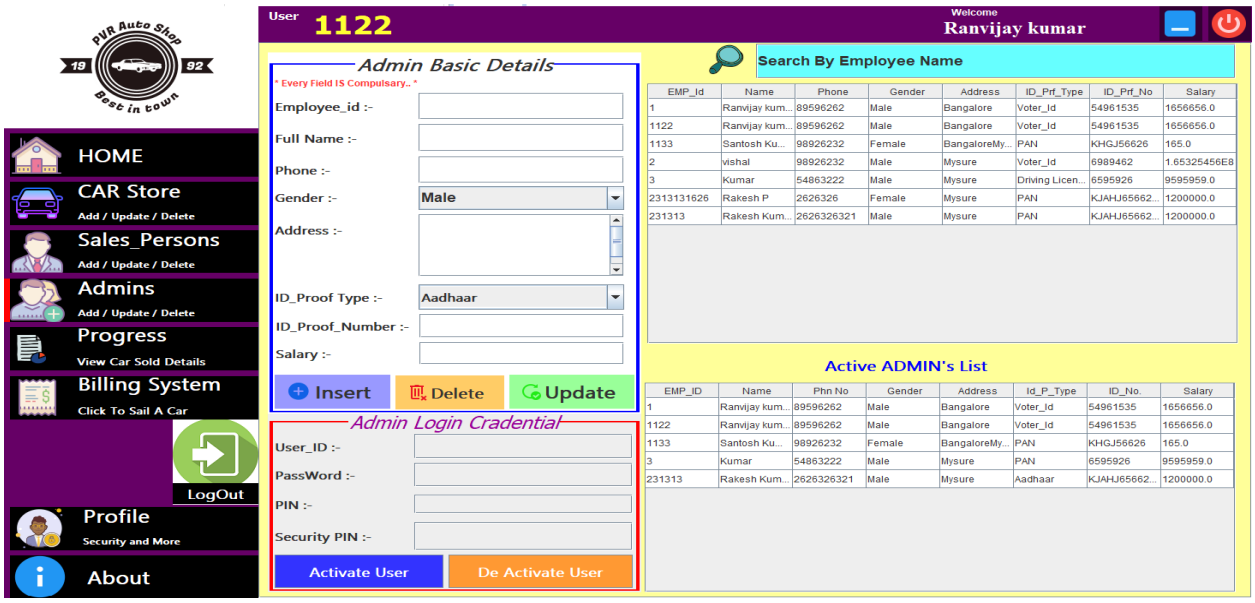

Fig 5.5 Admin Page

The fig 5.5 shows the details of the Existing Admins Information. The admin can add information of new Admins, and can update the information of Existing Admins and delete the information of them. And By clicking on “de activate user”, the admin can restrict the Another Admin to access the system anymore.

Fig 5.6 Progress Panel

The fig 5.6 shows the progress and the total number of cars sold by the store, this panel contains the information like car id, company Name of the cars, Sales Person Id and Name Of Sales Person , Customer details .

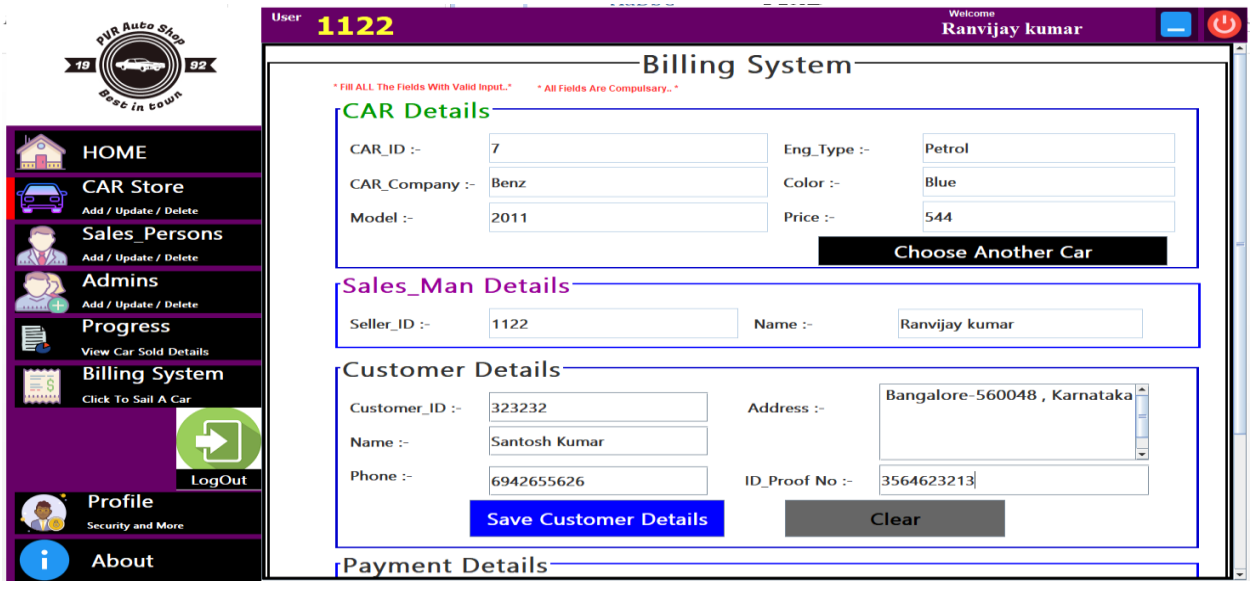

Fig 5.7 Billing System form

The fig 5.7 shows the billing system of the Car Showroom. Here the admin can generate the bills for all the cars that are sold. The bills also include information about the customers and the payment details.

Fig 5.8 Sales Person Home page

The fig 5.8 shows the details on the Sales Person Home page. The admin can access various field such as car store, and can monitor his progress. They can also view their respective profiles. The admin can also generate bill using the billing system.

Chapter 6

CONCLUSION AND FUTURE ENHANACEMENT

6.1 CONCLUSION

To conclude the description about mini project, the project developed in java based on the requirement specification of the user and the analysis of the existing system with flexibility of further enhancement. Thus we have successfully implemented CAR Showroom Management System which helps in administrating the data used for managing the tasks performed in CAR Showroom.

6.2 FUTURE ENHANCEMENT

This mini project which we have developed is a user friendly application but is still some scope of improvement like the following.

- The update and search options can be developed in more efficient way.
- Can be upgraded with new features like SMS Alerts, advance booking of cars, customers feedbacks, service details of cars in every store so that a more detailed information can be stored in the database.
- Car Showroom Management System can be extended with regional language support.

BIBLIOGRAPHY

- Database System Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkanth B. Navathae , 6th edition, Pearson.
- Data Management System, Ramakrishna and Gekhre, 3th edition, 2014, McGraw Hill.
- W3 school.
- YouTube.
- Quora.
- Stack Overflow.

APPENDIX

Source code

LOGIN MODULE

```
public class A_Login_Page extends javax.swing.JFrame {
int x,y;
 public A_Login_Page() {
 initComponents();
 int x,y;
       LoginTx.setBackground(new Color(0,0,0,180));
       SysTx.setBackground(new Color(0,0,0,180));
       //jPanel6.setBackground(new Color(255,0,51,20));
 p2.setBackground(new Color(0,0,0,0));
 p3.setBackground(new Color(0,0,0,0));
 p1.setBackground(new Color(0,0,0,0));
 //jPanel1.setBackground(new Color(255,0,51,150));
  this.p1.setVisible(false);
 this.p2.setVisible(false);
 this.up3.setVisible(false);
 this.up4.setVisible(false);
 //this.ap.setVisible(false);
 //this.a5.setVisible(false);
  this.a1.setVisible(false);
 //this.a2.setVisible(false);
  this.a3.setVisible(false);
   this.aid.setVisible(false);
    this.s2.setVisible(false);
    this.s3.setVisible(false);
```

```
    this.s4.setVisible(false);
    this.s5.setVisible(false);
    this.s6.setVisible(false);
    this.s7.setVisible(false);
    this.up4.setVisible(false)
   this.jLabel8.setVisible(false);
    this.jLabel19.setVisible(false);
    this.jLabel20.setVisible(false);
    this.jLabel18.setVisible(false);
    jLabel21.setBackground(new Color(255,255,255,150));
    }
    private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
int p=this.p3.getX();
p2.setBackground(new Color(255,255,255,0));
p3.setBackground(new Color(255,255,255,180));


   // ap.setVisible(true);
    //a5.setVisible(true);
    if(p>-1)
    {
        Animacion.Animacion.mover_derecha(150,805,1,15,p3);
  }p2.setVisible(false);
     p1.setVisible(true);
   up4.setVisible(true);
    a1.setVisible(false);
```

```
   // a2.setVisible(false);
    a3.setVisible(false);
    aid.setVisible(false);
    aps.setVisible(false);
    a4.setVisible(false);
    a6.setVisible(false);
    s1.setVisible(true);
    s2.setVisible(true);
    s3.setVisible(true);
    s4.setVisible(true);
    s5.setVisible(true);
    a6.setVisible(false);
    s6.setVisible(true);
    s7.setVisible(true);

    }     private void jLabel1MouseClicked(java.awt.event.MouseEvent evt) {
int p=this.p3.getX();
 //jLabel1.setBackground( new Color(255,153,10));
  jLabel2.setBackground(new Color(0,0,0));
//m1.setVisible(true);
p2.setBackground(new Color(255,255,255,0));
p3.setBackground(new Color(255,255,255,180));
    if(p>-1)
```

```
    {
        Animacion.Animacion.mover_izquierda(805,200,1,15,p3);
    }


   // Variables declaration - do not modify
    private javax.swing.JPanel LoginTx;
    private javax.swing.JPanel Mainf;
    private javax.swing.JPanel SysTx;
    private javax.swing.JLabel a1;
    private javax.swing.JLabel a3;
    private javax.swing.JLabel a4;
    private javax.swing.JLabel a5;
    private javax.swing.JLabel a6;
    private javax.swing.JPanel admbtn;
    private javax.swing.JTextField aid;
    private javax.swing.JPasswordField aps;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
```

```
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel p1;
    private javax.swing.JPanel p2;
    private javax.swing.JPanel p3;
    private javax.swing.JLabel s1;
```

```
    private javax.swing.JTextField s2;
    private javax.swing.JPasswordField s3;
    private javax.swing.JLabel s4;
    private javax.swing.JLabel s5;
    private javax.swing.JLabel s6;
    private javax.swing.JLabel s7;
    private javax.swing.JPanel salbtn;
    private javax.swing.JPanel up3;
    private javax.swing.JPanel up4;
    // End of variables declaration
}
```