A

PROJECT REPORT

ON

"VEHICLE REGISTRATION DETAIL"

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Science (Computer Application)

B.SC (2021-2022)

C.C.S UNIVERSITY, MEERUT

UNDER THE GUIDANCE OF:

Submitted To:	Submitted By:
Mr.Mohit Bansal(Senior Lecturer)	Pulkit Pal
Mr.Amit Sharma(Lecturer)	

D.A.V. (P.G.) COLLEGE,

MUZAFFARNAGAR 251001

DECLARATION

I hereby declare that the project work entitled "VEHICLE REGISTRATION DETAIL" submitted to the DAV College
Muzaffarnagar. This is a record of an original work done by PULKIT PAL . under the guidance of Mr.Mohit Bansal (Senior
Lecturer),Mr.Amit Sharma(Lecturer) and this project work is submitted in the partial fulfilment of the requirements for the award of the B.SC(Computer Application)

Pulkit Pal

ACKNOWLEDGEMENT

We are very pleased to submit this report "VEHICLE REGISTRATION DETAIL" .We have prepared this project with the help of our project leader and colleagues. We are very thankful to them for their support in making this report. I am grateful to all those people who extended their support and completion of my project.

We are deeply indebted to Dr.Lalit Kumar (Principal) and Mr. RAHUL SHARMA (InCharge) S.F.C and Head of the Department of Computer Application and Business Administration for providing us laboratory facilities to work in D.A.V. College ,MuzaffarNagar.

We express our profound thankfulness to Mr. Mohit Bansal (Senior Lecturer) Mr.Amit Sharma (Lecturer) for their excellent guidance and support throughout the project. We would also like to express my gratitude towards all faculty members without whom this project would not have been possible. They have been a motivator & source of inspiration for me to carry out the necessary proceedings for the project to be completed successfully.

We again place on record my gratitude to all those who were instrumental in getting the project work and report completed. We owe our sincere thanks to all faculty members of their supports, understanding and manifold help in every possible way.

Pulkit Pal

CONTENTS

INTRODUCTION

1.1 INTRODUCTION/OBJECTIVES OF THE PROJECT

One must know what the problem is before it can be solved as necessary is the mother of invention & of course, if we don't know where to go, we will not know when we will have reached there. The basis for a candidate system is recognition of a need for improving an information system or a procedure. This need leads to a preliminary survey or an initial investigation. It entails looking into the duplication of efforts, inefficient existing procedures etc.

MAIN OBJECTIVE OF THIS PROJECT:

- To provide security as only an authorized employee can interact with the system.
- To help the staff in working easily and efficiently.
- To help the staff in reducing errors that are encountered frequently during manual operations by concurrently updating the data stored in many places.
- To reduce the workload on the staff by providing automatic calculation.
- To help the staff or manager in tracing the transaction quickly.
- To help the staff in generating various reports at various nick of time as per their requirements e.g. daily, weekly, monthly or annually.
- To reduce the manpower.
- To maintain the data with complete consistency and minimum redundancy.

One must know what the problem is before it can be solved as necessary is the mother of invention & of course, if we don't know where to go, we will not know when we will have reached there. The basis for a candidate system is recognition of a need for improving an information system or a procedure. This need leads to a preliminary survey or an initial investigation. It entails looking into the duplication of efforts, inefficient existing procedures etc.

SYSTEM ANALYSIS

2.1 SYSTEM ANALYSIS

Systems analysis refers to the process of examining a business situation with the intent of improving it through better procedure and methods. Systems development can generally be thought of as having two major components Systems Analysis and Systems Design.

Systems analysis is the process of gathering and interpreting facts, diagnosis problems and using the information to recommend improvement to the system. In brief, we can say that analysis specifies what the system should do.

System analysis is conducted with the following objectives in mind:

1. Identify the customer and supplier members need

2. Evaluate the system concept for feasibility

3. Perform economic and technical analysis

4. Allocate function to hardware, software, people, database and other system elements

5. Establish cost and schedule constraints

6. Create a system definition that forms the foundation for all subsequent engineering

2.4 FEASIBILITY STUDY

The concept of feasibility is to determine whether or not a project is worth doing.
The process followed in making this determination is called feasibility study.
Once it has been determined that a project is feasible, the system analyst can go ahead and prepare the project specification which finalizes project requirements.

Types of feasibility

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility
4. Social feasibility
5. Management Feasibility
6. Legal Feasibility
7. Time Feasibility

Here we describe only few of these in detail:-

2.4.1 TECHNICAL FEASIBILITY

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. Technical needs of the system include:-

- Facility to produce output in a given time
- Response time under certain conditions
- Ability to process a certain volume of transactions at a particular period
- Facility to communicate data to distant location

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware .Configuration should give the complete picture about the system's requirements: how many workststion are required , how these units are interconnected so that they could operate and communicate smoothly. What speeds of input and output should be achieved at paticular quality of printing.

The computers are easily available in almost all the places, even in villages.The hardware needed to carry out this project include workststion with 64 MB of RAM and 2 GB HDD.

The software needed to carry out this project include Java as front end and ms access as backend. So the technology required to carry out the project is easily available and affordable, hence this project is technically feasible.

Due to all these reasons implementation of such system becomes not only feasible but reputed to the organization..

2.4.2 OPERATIONAL FEASIBILITY

This is mainly related to human organization and political aspects. The points to be considered are:-

- What changes will be brought with the system?
- What organizational structures are disturbed?

- What new skills will be required? Do the existing staff members have these skills?

This feasibility study is carried out by a small group of people who are familiar with the informations system techniques, who understand the parts of business that are relevant to the project and are skilled in the system analysis and design process.This project is not developed just for fun. They are developed on demand of the organization for which the system is being developed. Therefore the chances of resistance from the company Staff is almost nil. Any disturbance to the organization if occurs will be advantageous to the organization.Also the time required to carry out a transaction will be required to a large extent, which will make the customers and others happy and cheerful. The operators now will be able to service more customers and staff members than before in same time period. There is no need to recruit new staff cooperate the system .The existing staff of the company can be trained to interact with the system, which is a GUI, based software and is easy to use.

Hence the project is Operationally feasible.

2.4.3 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost-benefit analysis; the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs,a decision is taken to design and implement the system.

Cost-Benefit Analysis

Since cost plays an important role in deciding the new system, it must be identified and estimated properly. Benefits are also of different types and can be grouped on the basis of advantages they provide to the management.

Cost-saving benefit leads to reduction in administrative and operational costs.

Cost-avoidance benefit eliminate future administrative and operational costs.

Improved-service-level benefits are those where the performance of the system is improved by a new computer based method e.g. servicing a customer in two minutes rather than five to ten minutes is an example of this type of benefit.

Improved-information benefits is where computer based methods lead to better information for decision-making.

Direct Or Indirect Costs And Benefits

Direct costs are those which are directly associated with a system. They are applied directly to the operator.

Direct benefits also can be specifically attributable to a given project.

Indirect costs not directly associated with a specific activity in the system. they are often referred to as overhead expenses.

For example cost of space to install a system, maintenance of computer center ,

Heat ,light and air-conditioning are all tangible costs but their proportion is difficult to calculate to a specify activity like report.

Indirect benefits are realized as by –products of another system.

We can define cost-benefit analysis as:-

- That method by which we can estimate the value of the gross benefits of a new system specification.
- That method by which we find and determine the increased operating costs associated with the gross benefits.

- The subtraction of these operating costs from the associated gross benefits to arrive at net benefits.

- That method by which we find and estimate the monetary value of the development costs that produce the benefits.

- Those methods by which we show the time –relationship between net benefits and development costs as they relate to cash flow,payback on investment, and time-in process taking into operation factors such as inflation etc.

2.4.3.1 <u>COST ESTIMATON OF THE PROJECT</u>

The primary reason for cost and schedule estimation is to enable the client or developer to perform a cost-benefit analysis and for project monitoring and control . Cost and schedule estimates are also required to determine the staffing level for a project during different phases.

Cost in a project is due to the requirements for software,hardware,and human resources . Hardware resources are such things as the computer time,terminal time and memory required for the project , whereas software resources include the tools and the compilers needed during development.The bulk of the cost of software development is due to the human resources needed , and most cost estimation procedures focus on this aspect. Most of the estimates are determined in terms of person-months(PM)

The primary factor that controls cost is the size of the project, i.e. , the larger is the prject , the higher the cost and resorce requirement . Other factors that affect the cost include programmer ability , experience of the developer in the area of interest, complexity of the project , and reliability requirements. The most common approach for estimating effort is to make it a function of a single variable i.e. , the project size . The equation of effort is $\text{EFFORT} = a * \text{SIZE}^{b}$ where a and b are constants . Also,

$\text{EFFORT}=a* \text{SIZE} + b$

2.4.4 <u>SOCIAL FEASIBILITY</u>

It is the determination of whether a proposal project will be acceptable to , the staff members and customers , or not .This determination typically examines the probability of the project

being accepted by the group directly affected the proposed system change. To solve the actual problems in an company setting , a software or a team of engineers must incorporate a development strategy that encompasses the process, methods , and tools layers. This strategy is often refered to as a process model or a software engineering paradigm . A process model for software engineering paradigm .A Process model for software engineering is chosen based on the nature of the project and application , the methods and tools to be used and the controls and deliverables that are required.

SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a document that completely describes what the proposed software should without describing how the software will do it. The basic goal of requirement phase is to produce the SRS, which describes the complete external behavior of the proposed software.

A basic purpose of software requirements specification is to bridge the communication gap between the software developer and the client. SRS is the medium through which the client and user needs are accurately specified; indeed SRS forms the basis of software development. Another important purpose of developing an SRS is helping the clients understand their own needs.

The main phases of Software Requirements Specification are:-

4.1.1 FUNCTIONAL REQUIREMENTS

Functional Requirements specify which outputs should be produced from the given inputs. They describe the relationship between the input and output of the system. For each functional requirement, a detailed description of all the data inputs and their source, the units of measure, and the range of valid inputs must be specified.

All the operations to be performed on the input data to obtain the output should be specified. This includes specifying the validation checks on the input and output data.

4.1.2 PURPOSE OF THIS DOCUMENT

The purpose of this document is to convey the requirements of the project (as specified by the client) to the programmers to ensure that the programmers understand and fulfill the requirements to the expectation of the client.

Secondly, this document is used to ensure that the development team understands the requirements specified by the client. This document will act as the contract for all future development; all development spawns from and adheres to the details in the requirements. The SRS also outlines the performance requirements that may be set and required by the client/user.

This is the Software requirement specification for the “Auto spare parts Management” Project. This document provides a description of the purpose, functionality and interface of the software designed by the developer with input of the project. In this Software Requirement Specification, constraints and other design issues are addressed.

4.1.3 Scope of the Development Project

This project primarily achieves the following objectives on-line:-

1. To help the company staff in working easily and efficiently.
2.

   To help the company staff in providing better service to the customer and staff members in a Reasonable time i.e. to process their requests quickly without wasting time in Forming long queues by customers to get their job done.
3.

   To reduce the load of work on the company staff by providing automatic generation of the reports etc.
4. To help the company staff in generating reports about customers and staff as well.

5. To help the company staff in reducing errors that is encountered frequently during manual operations.

6. To provide security as only an authorized user can interact with the system.

4.1.4 Overview of Document

The remainder of this document describes the intended users that would be expected to interact with the system frequently, and a simple profile of each user type is provided as a sample. This document now will go into more detail on the expected users their interface and interaction with the product and more on the technical approach and considerations to be implemented.

4.1.5 General Description

User Persons and Characteristics

The primary users of this product are the company employees, director perspectives candidates. Most of them already have some experience in using computer components (mouse and keyboard), and are willing to learn and explore under the supervision of their superiors. The employees have adequate knowledge so that they can be trained easily to operate the system.

Overview of Functional Requirements

Our product will be stand-alone and will have an interface, which can be accessed by staff and managing director of institute who are conducting different type of courses in his Company. Our main goal is to present facts on a comprehensive level, and makes it easier as well.

4.1.6 Hardware Interface Requirements:

Our product will require at least a PowerPC Macintosh or a Pentium class PC with 64 MB of RAM (64 + recommended), and color display.

Other Software Components:

Operating System: Windows 98/ME/2000

Detailed Description of Functional Requirements

Template for describing functional requirements

4.1.1 Performance Requirements

The software is inherently designed to handle a user accessing the database system. Single user can use this system at a time and will provide reliability, efficiency, and excellent response time.

DESIGN

FRONT END & BACK END

FRONT END

JAVA has been developed by Sun MicroSystem. Java can use a large number of reference liabraries and components. Visual Basic –the environment-provides integration with a variety of other tools, including source code management ,component reuse tools and data manipulation. Visual Basic truly is much more of a “programming environment” than just a language. using this environment ,a single developer can quickly create a simple application; a team of developers can create a so sophisticated, distributed application.

BACK END

A database is a collection of information that’s related to a particular subject or purpose, such as tracking customer orders or maintaining a music collection .If your database isn’t stored on a computer, or only parts of it are , you may be tracking information from a variety of sources that you have having to coordinate and organize yourself.

For example, suppose the phone numbers of your suppliers are stored in various locations: in a card file containing supplier phone numbers, in product information file cabinet, and in a spreadsheet containing order information .if a supplier’s phone number changes, you might have to update that information in all three places in database, however, you only have to update that information in one place- the supplier’s phone number is automatically updated wherever you use it in database.

Using Ms Access, you can manage all your information from a single database file. within the file, you can use:

- Tables to store your data.
- Queries to find and retrieve just the data you want

- Forms to view, add and update data in tables.
- Reports to analyze or print data in a specific layout.
- Data access pages to view, update, or analyze the database's data from the internet or an intranet.

5.1 DESIGN OBJECTIVES

The primary objects of design are to deliver the requirements as specified in the feasibility report. Following objectives should be kept in mind:

a) Practicality

The system must be stable and can ne operated by people with average intelligence.

b) Efficiency

This involves accuracy, timelines and comprehensiveness of the system output.

c) Cost

It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy all the requirements .

d) Flexibility

The system should be modifiable depending on the changing needs of the user. Such modification should not entail extensive reconstruction o recreation of software. It should also be portable to different computer system.

e) Security

This is very important aspect of the design and should cover areas of hardware reliability, fall back procedures, physical security of data and provision for detection of fraud and abuse.

5.2 INTRODUCTION

The aim of system design, which is sometimes also referred to as top – level design, is to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system

design all the major data structures, file formats, output formats and the major modules in the system and their specifications are needed.

5.3 SOFTWARE DESIGN

Software design in an iterative process through which requirements are translated into a “blue print” for constructing the software. Characteristics that serve as a guide for the evaluation of a good design:-

1. The design must implement all of the explicit requirements contained in the analysis model, and it accommodates all of the implicit requirements desire by the customers and supplier members.

2. The design must be readable, understandable guide for those who generate code and for those who test and subsequently maintain the software.

3. The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

5.4 DESIGN QUALITY CRITERIA

1. A design should exhibit a hierarchical organization that makes intelligent use of control among the elements of software.

2. A design should be modular i.e. the software should be logically partitioned into elements that perform specific functions and sub functions.

3. A design should contain both data and procedural abstraction.

4. A design should lead to modules that exhibit independent functional characteristics.

5. A design should lead to interfaces that reduce the complexity of connections between modules and with

the external environment.

4. A design should be derived using a repeatable method that is driven by information obtained during software requirement analysis. All these things are implemented in our projects using options on the main menu screen. Each option provides a different kind of information , providing modular approach. Data submitted to the database, which gives abstraction to the data using middle tier concepts with fully Object-Oriented programming.

5.5 DESIGN CONCEPTS

A set of fundamental software design concepts has evolved:-

1. What criteria can be used to partition software into individual components?

2. How is function or data structure detail separated from a conceptual representation of the software?

3. Are there uniform criteria that define the technical quality of a software design?

5.5.1 TOP-DOWN AND BOTTOM UP STRATEGIES

A system consists of components, which have the components of their own, indeed a system is a hierarchy of components. The highest-level components correspond to the total system.

A top-down design approach starts with identifying the major components of the system, decomposing them into their low- level components and iterating until desired level of details is achieved. Top-down design methods often result in some form of stepwise refinement. Starting from an abstract design, in each step the design is refined to a more concrete level, until we reach a level where no more refinement is needed and the design can be implemented directly. A bottom-up design approach starts with designing the most basic or primitive components and proceeds to higher-level components. Bottom-up methods work with layer of abstraction. Starting from the very bottom, operations that provide a layer of abstraction are implemented. The operations of this layer are then used to implement more powerful operations and a still higher layer of abstraction, until the stage is reached where the operation supported by the layer are those desired by the system.

5.6 ABSTRACTION

"The psychological notion of " abstraction " permits one to concentrate on a problem at some level of generalization without regard to irrelevant low level details ; use of abstraction also permits one to work with concepts and terms that are familiar in the problem environments without having to transform them to an unfamiliar structure …".

a. <u>Procedural Abstraction</u>

b. Data Abstraction

a) PROCEDURAL ABSTRACTION

It is a named sequence of instruction that has a specific and limited function.

b) DATA ABSTRACTION

It is a named collection of data that describes a data object

1. REFINEMENT

Stepwise Refinement Is a Top-Down Strategy and the architecture of a program is developed by successive refining levels of procedural details "In each step of refinement, one or more instructions of the given program are decomposed into more details instructions. This successive decomposition or refinement of specifications terminates when all instructions are expressed in terms of an underlying computer or programming languages. As tasks are refined, so the data may have to be refined, decomposed, or structured, and it is natural to refine the program and the specification in parallel."

Every solution is always refinable depending on time period and availability of information. This project is also refinable . We can use the same project for online Company i.e. eCompany later on.

5.8 MODULARITY

"Modularity is the single attribute of software that allows a program to be intellectually manageable". Monolithic software can't be easily grasped by a reader. The number of control paths, spam of reference, number of variables, and overall complexity would make understanding close to impossible.

STRUCTURED DESIGN

Structure design methodology views every software system as having some inputs that are converted into the desired outputs by the software system. The software is viewed as a transformation function that transforms the given inputs into the desired outputs, and the central problem of designing this transformation function. Due to this view of software, the structured designed methodology is primarily function oriented and relies heavily on functional abstraction and decomposition. The approach begins with a system specification that identifies inputs and outputs and describes the functional aspects of the system. The next step is the definition of the modules and the relationship with one another in a form called structure chart, using data dictionary and the other structure tools.

Entity- Relationship diagram (ERD)

The E-R diagram enables a software engineer to fully specify the data objects that are input and output to/from a system, the attributes that define the properties of these objects, and the relationship between the objects. The following approach is taken:-

1. During requirements gathering, customerts and suppliermembers are asked to list the “things”

That the application or business process and addresses. These “things” evolve into a list of input and output data, objects as well as external entries that produce or consume information.

2. Taking the objects one at a time, the analyst and costumer define whether or not a connection (unnamed at this stage) exists between the data, object and another object.

3. Wherever a connection exists, the analyst and costumer create one or more objectrelationship pairs.

4. For each object-relationship pair, cardinality and modality are explored.

5. Steps through 4 are continued iteratively until all object-relationship pairs have been defined. It is common to discover omissions as this process continues. New objects are relationships will invariably be added as the number of iterations grows.

6. The attributes of each entity are defined. An entity-relationship diagram is formalized in review.

7. Step I through 6 are repeated until data modeling is complete.

DATA FLOW DAIGRAMS AND DATA DICTIONARY

DFD’s are commonly used during problem analysis. Data flow diagrams are not limited to problem analysis for software requirement specification.

A DFD shows the flow of data through the system. It views a system as a function that that transforms the inputs into desired outputs. The DFD aim to capture the transformations that take place within a system to the input data so that eventually the output data is produced. The agent that performs the transformation of data from one state to another is called a process (or bubble). The processes are shown by named circles and data flows are represented by named arrows entering or leaving the bubbles. A rectangle represents a sources or sink and is a net originator or consumer of data.

It should be pointed out that DFD is not a flowchart. A DFD represents the flow of data, while a flowchart shows the flow of control. A DFD does not represent procedural information. In drawing the DFD the designer has to specify the major transforms in the path of the data flowing from input to output.

DATA DICTIONARY

Data dictionary is a repository of various data flows defined in data flow diagram. The associated data dictionary states precisely the structure of each data flow in DFD. Although the format of dictionaries varies from tool to tool, most consists of the following information:

Name – the primary name of the data or control item, the data store or an external entity.

Alias – other names used for the first entity.

Where – used / how – used – a listing of the processes that used the data or control item and how it is used (E.g. input to the process, output from the process, as a store, as an external entity).

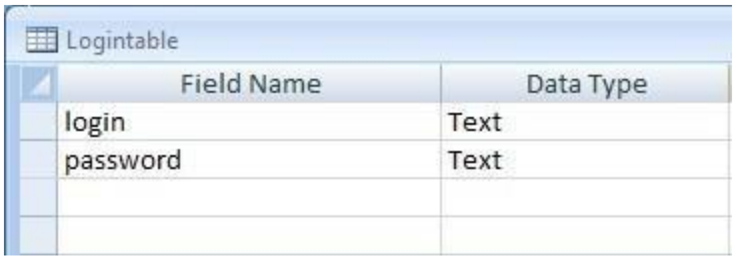
Content description – a notation for representing content.

Supplementary information – other information about data types, present values (if known), restrictions of limitation.

5.1 DETAILED DESIGN

During detailed design, the internal logic of each of these modules specified in system design is decided. During this phase further details of data structures and algorithmic design of each of the modules is specified. The logic of a module is specified in a high-level design descriptional language, which is independent of the target language in which software will eventually be implemented.

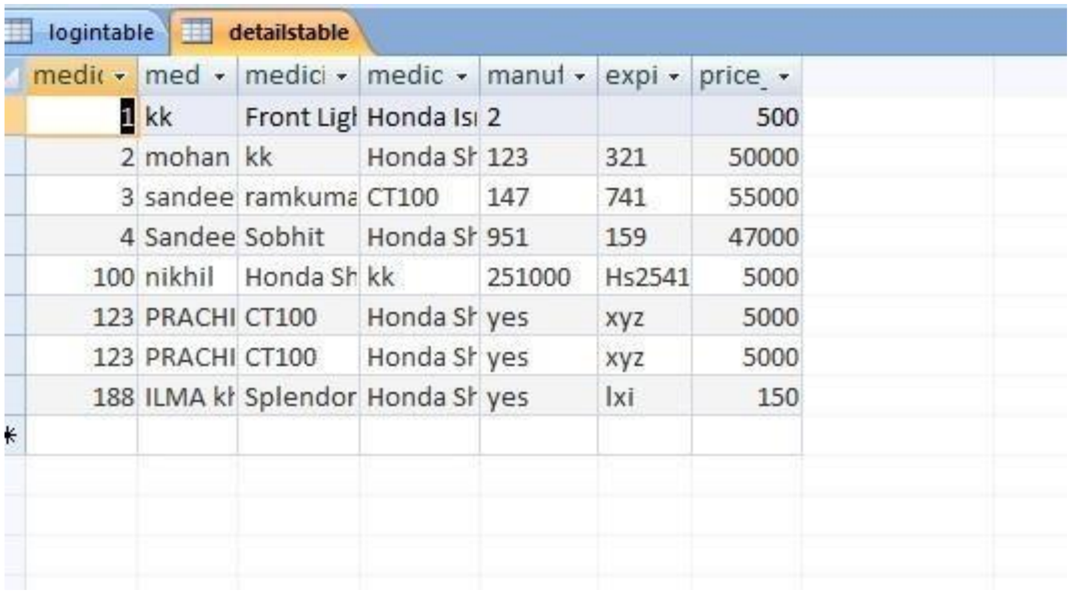
Logintable



Logintable

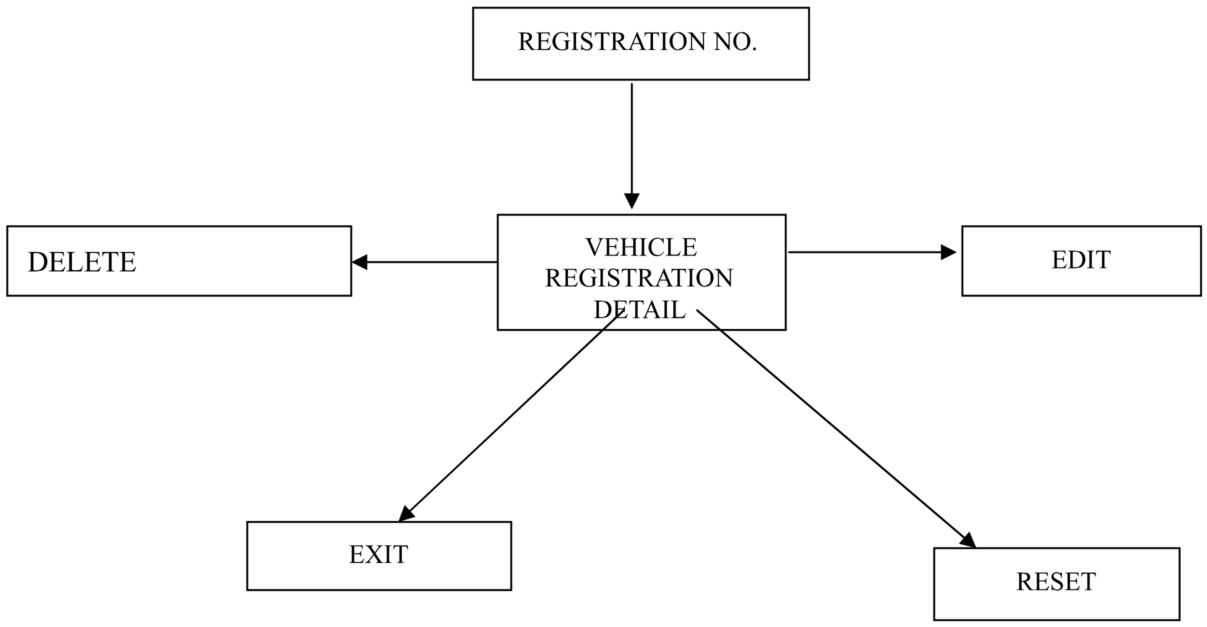
Field Name	Data Type
login	Text
password	Text

supptable



medic	med	medici	medic	manut	expi	price_
1	kk	Front Ligl	Honda Isı	2		500
2	mohan	kk	Honda Sh	123	321	50000
3	sandee	ramkuma	CT100	147	741	55000
4	Sandee	Sobhit	Honda Sh	951	159	47000
100	nikhil	Honda Sh	kk	251000	Hs2541	5000
123	PRACHI	CT100	Honda Sh	yes	xyz	5000
123	PRACHI	CT100	Honda Sh	yes	xyz	5000
188	ILMA kh	Splendor	Honda Sh	yes	lxi	150

MAIN MODULE DIAGRAM


REGISTRATION NO.
VEHICLE REGISTRATION DETAIL
DELETE
EDIT
EXIT
RESET

DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical tool used to describe and analyze the movement of data through a system - manual or automated including the processes, stores of data and delays in the system. They are central tools and the basis from which other components are developed. It depicts the transformation of data, independent of the physical components, from input to output through logical processes and the interaction between processes.

DFD’s are an excellent mechanism of communicating with the customers and are widely used for representing external and top-level internal design specification. In the later situations, DFD’s are quite valuable for establishing naming conventions and names of system components such as subsystems, files and data links.

In a DFD there are four components:

Sources or Destinations of data such as human, entities that interact with system, outside the system boundary, who form the source and the recipient of information are depicted in the form of a closed rectangle.

Data flow is a packet of data. It identifies data flow. It is a pipeline through which information flows. It is depicted in DFD as an arrow with the pointer pointing in the direction of flow. This connecting symbol connects an entity, process and data stores. This arrow mark also specifies the sender and the receiver.

Process depicts procedure, function or module that transforms input data into output data. It is represented as a circle or a bubble with the procedure name and a unique number inside the circle.

Data stores are the physical areas in the computer’s hard disk where groups of related data are stored in the form of files. They are depicted as an open-ended rectangle. The Data store is used either for storing data into the files or for reference purposes.

ENTITY – RELATIONSHIP DIAGRAM

The entity – relationship (E – R) data model is based on a perception of a real world that consists of a collection of basic objects, called entities, and of relationships among these objects. An entity is a thing or objects in the real world that is distinguishable from other objects. Entities are described in database by a set of attributes.

A relationship is an association among several entities. The set of all entities of the same type and the set of all relationships of the same type are termed as entity set and relationship set, respectively.

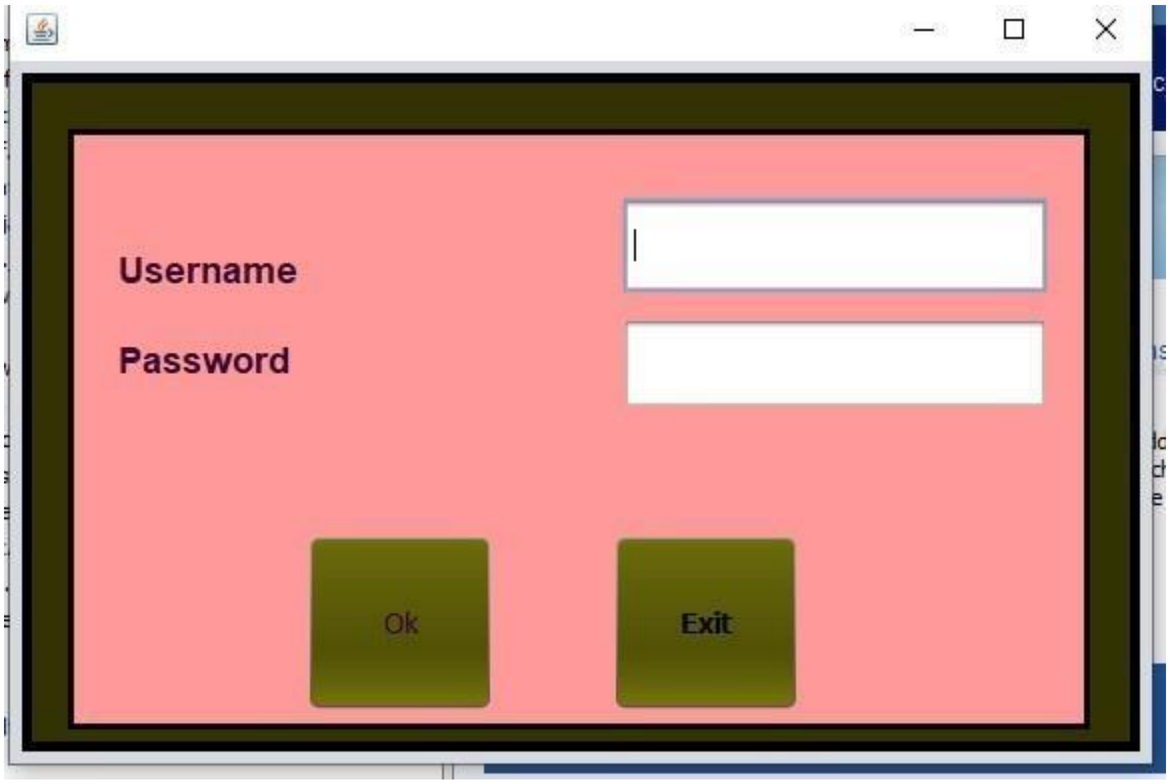
The overall logical structure (schema) of a database can be expressed graphically by an E-R Diagram, which is built up from the following components:

- ❒ Rectangles, which represents entity sets.
- ❒ Ellipses, which represents attributes.
- ❒ Diamonds, which represents relationships among entity sets.

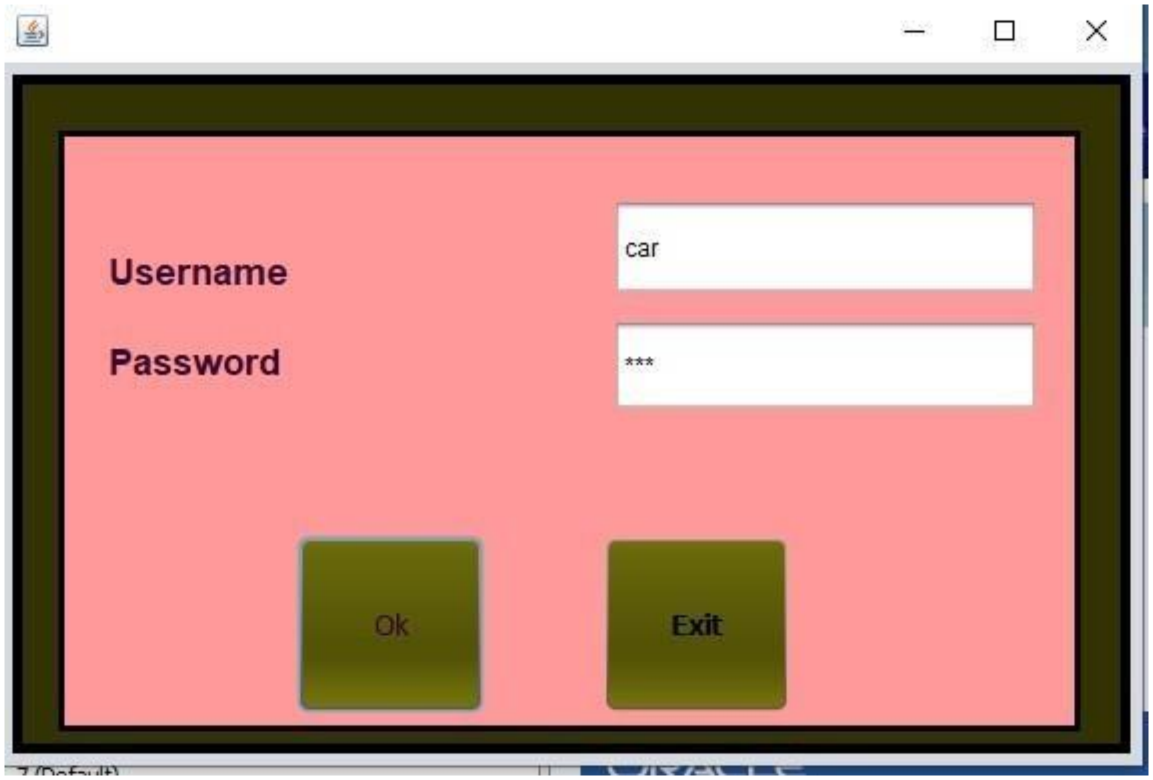
- Lines, which link attributes to entity sets and entity sets to relationships.

SCREENS & REPORTS

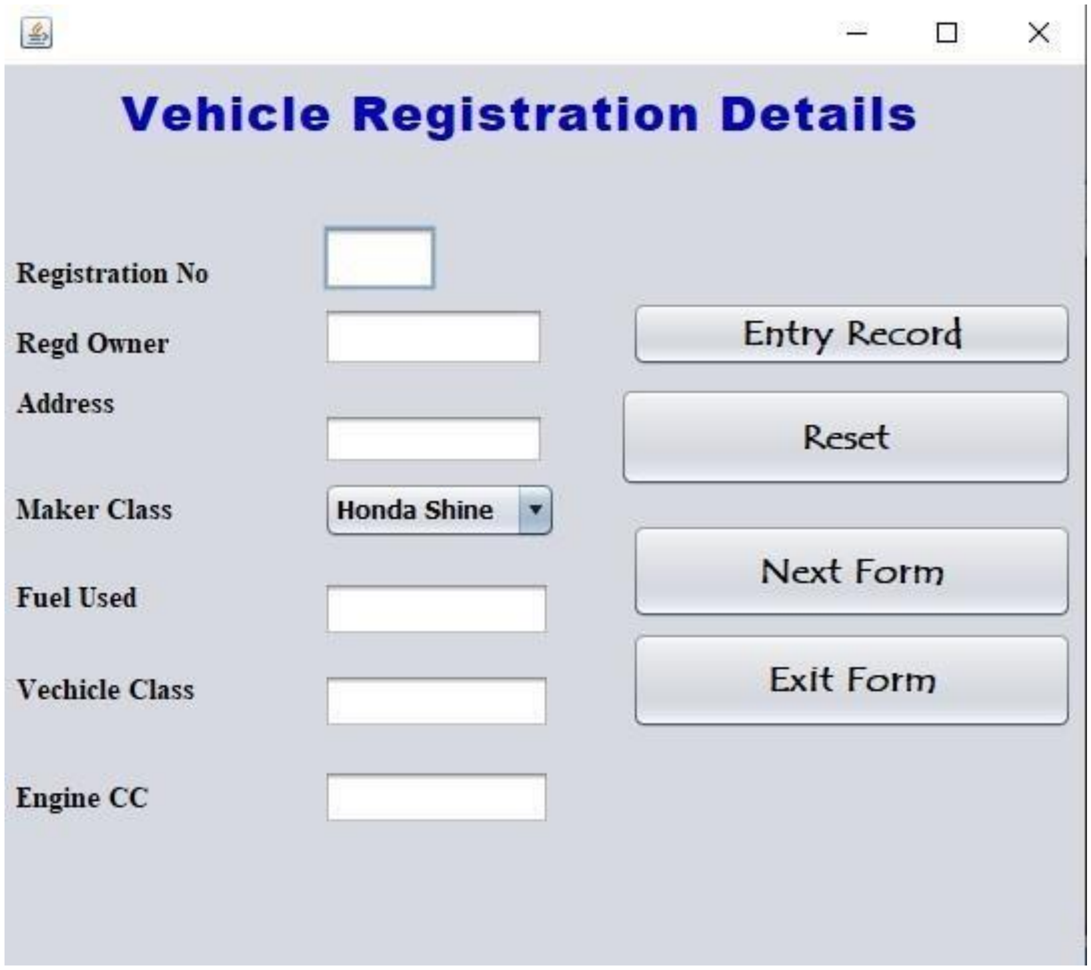
Login Form



Uesr id Form



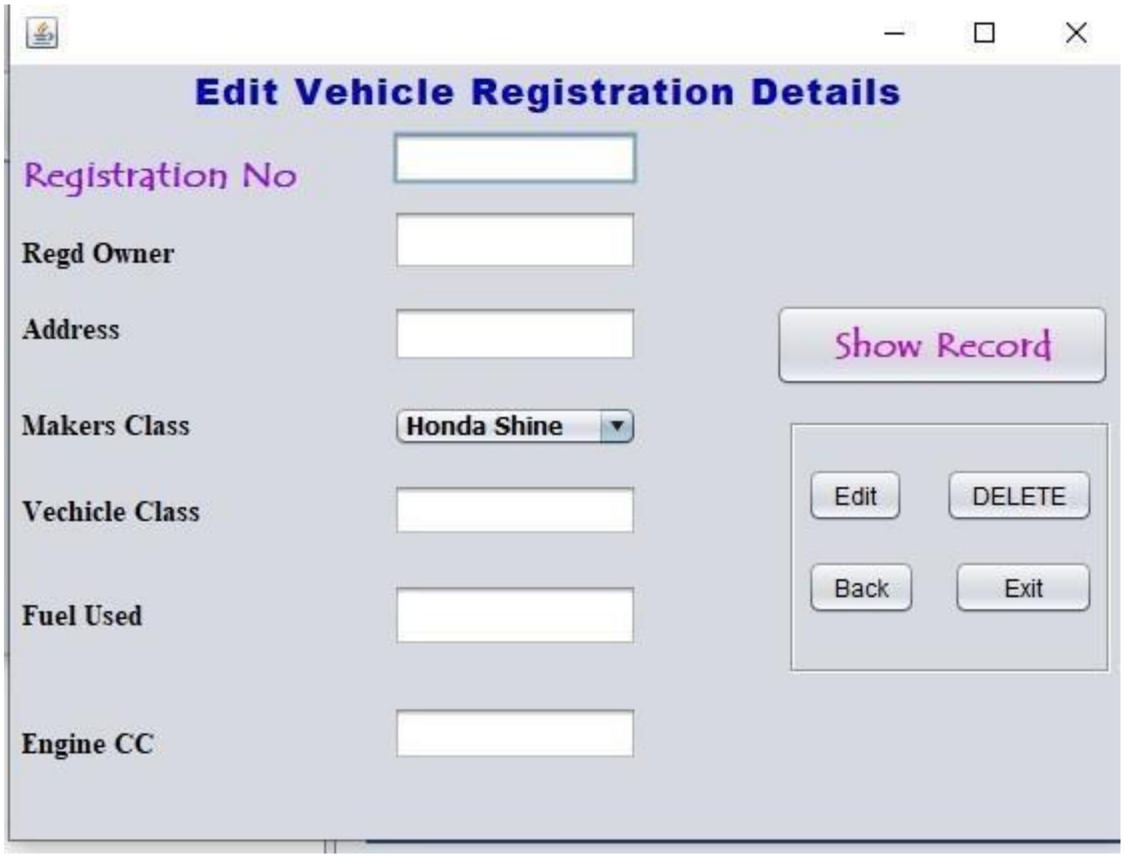
Main Form



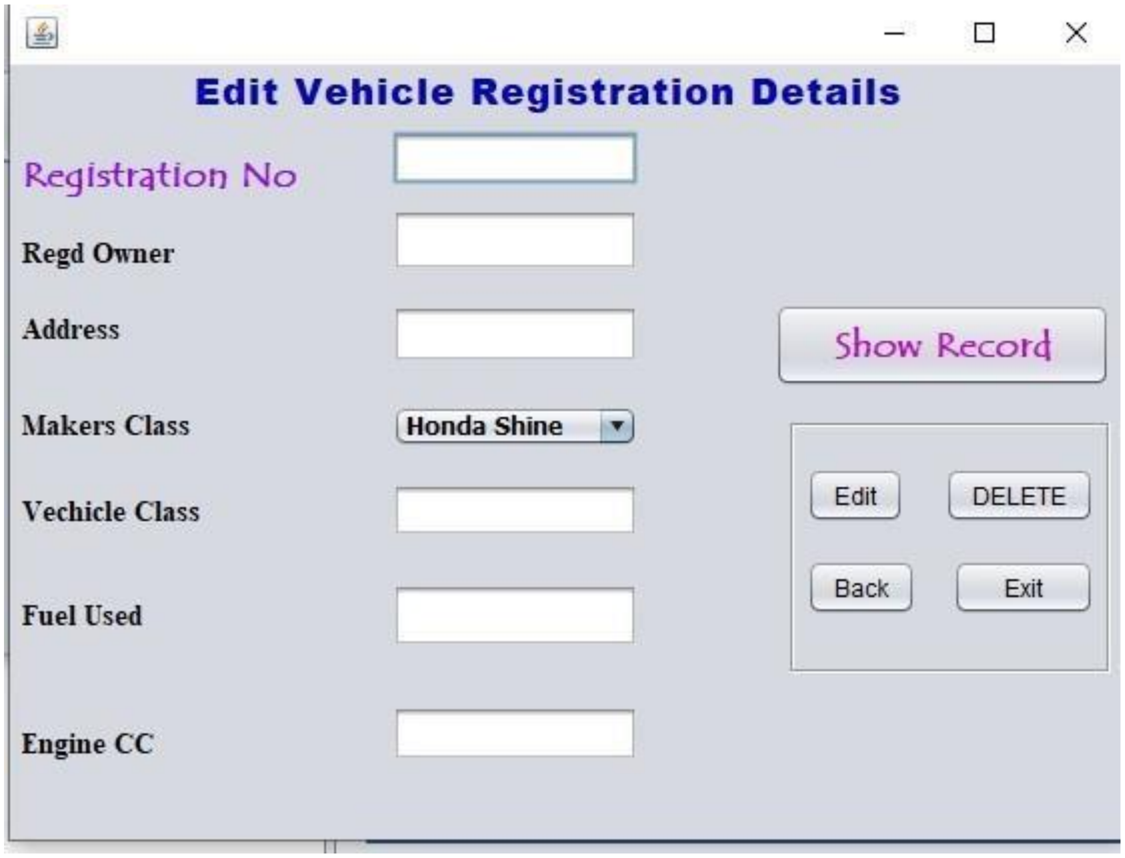
Vehicle Registration Details

Registration No	
Regd Owner	
Address	
Maker Class	Honda Shine
Fuel Used	
Vechicle Class	
Engine CC	

EDIT FORM


Edit Vehicle Registration Details
Registration No
Regd Owner
Address
Makers Class
Honda Shine
Vechicle Class
Fuel Used
Engine CC
Show Record
Edit
DELETE
Back
Exit

DELETE FORM



Edit Vehicle Registration Details

Registration No

Regd Owner

Address

Makers Class: Honda Shine

Vechicle Class

Fuel Used

Engine CC

Edit Vehicle Registration Details

Registration No

Regd Owner

Address

Makers Class: Honda Shine

Vechicle Class

Fuel Used

Engine CC

CODING

```
/*  * To change this template, choose Tools | Templates  * and open the template in the editor.
 */
package Autopartss;

import java.sql.*; import
javax.swing.*;


/**
 *
*   @author student
 */
public class Partsdetails extends javax.swing.JFrame {

   /**
*   Creates new form Partsdetails
    */
   public Partsdetails() {         initComponents();
   }

   /**
*   This method is called from within the constructor to initialize the form.  * WARNING: Do
    NOT modify this code. The content of this method is always  * regenerated by the Form
    Editor.
    */
   @SuppressWarnings("unchecked")                   //    <editor-fold    defaultstate="collapsed"
desc="Generated
Code">                              private void initComponents() {

      jLabel1 = new javax.swing.JLabel();         jLabel2 = new
javax.swing.JLabel();         btndisplay = new javax.swing.JButton();         jLabel3
= new javax.swing.JLabel();         txtname = new
javax.swing.JTextField();         jLabel4 = new javax.swing.JLabel();
txtcompany = new javax.swing.JTextField();         jLabel5 = new
javax.swing.JLabel();         txtid = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();         jLabel7 = new
javax.swing.JLabel();         txtmfd = new javax.swing.JTextField();         jLabel8
= new javax.swing.JLabel();         txtexp = new
javax.swing.JTextField();         jLabel9 = new javax.swing.JLabel();
txtprice = new javax.swing.JTextField();         jPanel1 = new
javax.swing.JPanel();         btndel = new javax.swing.JButton();
```

```
    btnupdate = new javax.swing.JButton();         btnexit = new
    javax.swing.JButton();        jButton1 = new javax.swing.JButton();
    jComboBox1 = new javax.swing.JComboBox();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_C                    LOSE);
    setBackground(new java.awt.Color(51, 0, 0));

        jLabel1.setBackground(new java.awt.Color(51, 0, 51));

        jLabel2.setBackground(new java.awt.Color(0, 0, 255));        jLabel2.setFont(new
    java.awt.Font("Arial Black", 1,
    18));  //  NOI18N                     jLabel2.setForeground(new   java.awt.Color(0,   0,   204));
jLabel2.setText("Edit Vehicle
                                          Registration Details");

        btndisplay.setFont(new   java.awt.Font("Tempus   Sans   ITC",   1,   18));  //   NOI18N
    btndisplay.setForeground(new java.awt.Color(204, 0,
    204));              btndisplay.setText("Show
    Record");
    btndisplay.addActionListener(new
    java.awt.event.ActionListener() {            public void
    actionPerformed(java.awt.event.ActionEvent evt) {
    btndisplayActionPerformed(evt);
          }
        });

        jLabel3.setFont(new java.awt.Font("Times New Roman",
    1, 14)); // NOI18N
        jLabel3.setText("Regd Owner");

        txtname.addActionListener(new
    java.awt.event.ActionListener() {            public void
    actionPerformed(java.awt.event.ActionEvent evt) {              txtnameActionPerformed(evt);
          }
        });

        jLabel4.setFont(new java.awt.Font("Times New Roman",
    1, 14)); // NOI18N
        jLabel4.setText("Address");

        jLabel5.setFont(new java.awt.Font("Tempus Sans ITC",
1, 18)); // NOI18N             jLabel5.setForeground(new
java.awt.Color(153, 0,
255));
        jLabel5.setText("Registration No");

        jLabel6.setFont(new java.awt.Font("Times New Roman",
1, 14)); // NOI18N
        jLabel6.setText("Makers Class");

        jLabel7.setFont(new java.awt.Font("Times New Roman",
1, 14)); // NOI18N
        jLabel7.setText("Vechicle Class");

        jLabel8.setFont(new java.awt.Font("Times New Roman",
1, 14)); // NOI18N
        jLabel8.setText("Fuel Used");

        jLabel9.setFont(new java.awt.Font("Times New Roman",
1, 14)); // NOI18N
        jLabel9.setText("Engine CC");

jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder
());
        btndel.setText("DELETE");
btndel.addActionListener(new
java.awt.event.ActionListener() {             public void
actionPerformed(java.awt.event.ActionEvent evt) {                btndelActionPerformed(evt);
            }
        });

        btnupdate.setText("Edit");        btnupdate.addActionListener(new
java.awt.event.ActionListener() {             public void
actionPerformed(java.awt.event.ActionEvent evt) {
btnupdateActionPerformed(evt);
            }
        });

        btnexit.setText("Exit");        btnexit.addActionListener(new
java.awt.event.ActionListener() {             public void
actionPerformed(java.awt.event.ActionEvent evt) {                btnexitActionPerformed(evt);
            }
        });

        jButton1.setText("Back");
jButton1.addActionListener(new
java.awt.event.ActionListener() {            public void
actionPerformed(java.awt.event.ActionEvent evt) {
jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig            nment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING, false)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                    .addComponent(jButton1)
                    .addGap(18, 18, 18)
                    .addComponent(btnexit,   javax.swing.GroupLayout.PREFERRED_SIZE,   71,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                    .addComponent(btnupdate)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(btndel)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))                                              );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig nment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()              .addGap(20, 20, 20)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                .addComponent(btnupdate)
                .addComponent(btndel))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                .addComponent(jButton1)
                .addComponent(btnexit))
            .addContainerGap(27, Short.MAX_VALUE))
      );
      jComboBox1.setFont(new java.awt.Font("Tahoma", 1,
12)); // NOI18N
      jComboBox1.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Honda Shine",
"Honda Dream", "CT100", "Honda Ismart", "Splendor" }));
jComboBox1.addActionListener(new java.awt.event.ActionListener() {
public void
actionPerformed(java.awt.event.ActionEvent evt) {
jComboBox1ActionPerformed(evt);
         }
      });

      javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);       layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
          .addGroup(layout.createSequentialGroup()              .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A lignment.LEADING)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A      lignment.TRAILING,
false)
                    .addComponent(jLabel7,      javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel6,      javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A      lignment.TRAILING,
false)
                .addComponent(jLabel9,     javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel8,    javax.swing.GroupLayout.Alignment.LEADING))
.addComponent(jLabel5))
            .addGap(48, 48, 48)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A       lignment.LEADING,
false)
              .addComponent(txtcompany)
              .addComponent(txtmfd)
              .addComponent(txtexp)
              .addComponent(txtprice)
              .addComponent(jComboBox1, 0, 123,
Short.MAX_VALUE)
              .addComponent(txtname)
              .addComponent(txtid))
                txtid.setText("");
txtname.setText("");        txtcompany.setText("");
//txttype.setText("");              txtprice.setText("");
txtmfd.setText("");             txtexp.setText("");
    }
    catch(Exception e)
    {}
   }

   private void
btndelActionPerformed(java.awt.event.ActionEvent evt) {                        int mid;
mid=Integer.parseInt(txtid.getText());       try{
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");                          Connection
con=DriverManager.getConnection("jdbc:odbc:auto");
    Statement stmt=con.createStatement();
    String query="delete * from detailstable where medicine_id="+mid;
stmt.executeUpdate(query);     txtid.setText("");          txtname.setText("");
txtmfd.setText("");          txtexp.setText("");              txtprice.setText("");
txtcompany.setText("");                    // txttype.setText("");

JOptionPane.showMessageDialog(null,"Successfully Deleted");
```

```
        }
        catch(Exception e)
        {}




    }

    private    void       btnexitActionPerformed(java.awt.event.ActionEvent    evt)    {
dispose();
    }

    private void  jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
 Informationform obj=new Informationform();          obj.show();
    }

    private void
jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    }



    /**
* @param args the command line arguments
     */
    public static void main(String args[]) {         /* Set the Nimbus look
and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">         /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.  * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel
/plaf.html             */          try {               for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {               if
("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());                  break;
              }
          }
        } catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(Partsdetails.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Partsdetails.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);         } catch
(IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Partsdetails.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        }           catch           (javax.swing.UnsupportedLookAndFeelException           ex)           {
java.util.logging.Logger.getLogger(Partsdetails.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */          java.awt.EventQueue.invokeLater(new
Runnable() {            public void run() {                 new
Partsdetails().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify                        private
javax.swing.JButton btndel;     private javax.swing.JButton btndisplay;
private javax.swing.JButton btnexit;     private javax.swing.JButton
btnupdate;     private javax.swing.JButton jButton1;     private
javax.swing.JComboBox jComboBox1;     private javax.swing.JLabel
jLabel1;     private javax.swing.JLabel jLabel2;     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.JTextField
txtcompany;     private javax.swing.JTextField
txtexp;     private javax.swing.JTextField txtid;     private
javax.swing.JTextField txtmfd;     private javax.swing.JTextField txtname;
private javax.swing.JTextField txtprice;
    // End of variables declaration                    }
/*  * To change this template, choose Tools | Templates  * and open the template in the editor.
 */
package Autopartss; import
java.sql.*; import javax.swing.*;
/**
```

```
 *
*   @author student
 */
public class login extends javax.swing.JFrame {

   /**
*   Creates new form login
    */    public login() {        initComponents();
   }

   /**
*   This method is called from within the constructor to initialize the form.  * WARNING: Do
    NOT modify this code. The content of this method is always  * regenerated by the Form
    Editor.
    */
   @SuppressWarnings("unchecked")                   //    <editor-fold    defaultstate="collapsed"
desc="Generated
Code">                              private void initComponents() {

       jPanel1 = new javax.swing.JPanel();        jPanel2 = new
javax.swing.JPanel();        jLabel1 = new javax.swing.JLabel();
txtuser = new javax.swing.JTextField();        jLabel2 = new
javax.swing.JLabel();        txtpwd = new javax.swing.JPasswordField();
btnsign = new javax.swing.JButton();        btnexit = new
javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_C LOSE);
       jPanel1.setBackground(new               java.awt.Color(51,               51,               0));
jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(n ew java.awt.Color(0, 0,
0), 5));

       jPanel2.setBackground(new java.awt.Color(255, 153,  153));

jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(n ew java.awt.Color(0, 0,
0), 3));

       jLabel1.setFont(new       java.awt.Font("Tunga",       1,       18));       //       NOI18N
jLabel1.setForeground(new java.awt.Color(51, 0, 51));
jLabel1.setText("Username");

       txtuser.addActionListener(new
java.awt.event.ActionListener() {          public void
```

```
actionPerformed(java.awt.event.ActionEvent evt) {                txtuserActionPerformed(evt);
           }
        });

        jLabel2.setFont(new        java.awt.Font("Tunga",        1,        18));        //        NOI18N
jLabel2.setForeground(new java.awt.Color(51, 0, 51));
jLabel2.setText("Password");

        btnsign.setBackground(new java.awt.Color(51, 51, 0));        btnsign.setFont(new
java.awt.Font("Tahoma", 0, 14));
// NOI18N
        btnsign.setForeground(new  java.awt.Color(51,  0,  51));              btnsign.setText("Ok");
btnsign.addActionListener(new java.awt.event.ActionListener() {            public void
actionPerformed(java.awt.event.ActionEvent evt) {                btnsignActionPerformed(evt);
           }
        });

        btnexit.setBackground(new java.awt.Color(51, 51, 0));        btnexit.setFont(new
java.awt.Font("Tahoma", 1, 14));
// NOI18N
        btnexit.setText("Exit");        btnexit.addActionListener(new
java.awt.event.ActionListener() {            public void
actionPerformed(java.awt.event.ActionEvent evt) {                btnexitActionPerformed(evt);
           }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alig nment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
              .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                     .addGap(22, 22, 22)
```

```
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
                    .addComponent(jLabel1,     javax.swing.GroupLayout.PREFERRED_SIZE,
105, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
106, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, 135, Short.MAX_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING, false)
                    .addComponent(txtpwd,  javax.swing.GroupLayout.DEFAULT_SIZE,  213,
Short.MAX_VALUE)
                    .addComponent(txtuse
```

```
/*
* To change this template, choose Tools | Templates  * and
  open the template in the editor.
 */
package Autopartss;

/**
 *
* @author student
 */
public class VehicaSales {

   /**
* @param args the command line arguments
    */
   public static void main(String[] args) {
// TODO code application logic here
   }
}
```

TESTING

TESTING

During system testing, the system is used experimentally to ensure that the software does not fail. Specific test data are input for processing, and the results examined. It is desirable to discover any surprises before the organization implements the system and depends on it.

10.1 TESTING OBJEC TIVES:

1. Testing is a process of executing a program with the intent of finding an 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 an as yet undiscovered error.

10.2 TESTING PRINCIPLES:

1. All tests should be traceable to the customer and suppliers requirements.

2. Tests should be planned long before the testing begins.
3. Testing should begin “in the small” and progress towards “in the large”.
4. Exhaustive testing is not possible.
5. To be most effective, testing should be conducting by an independent third party.

10.3 TESTING FUNDAMENTALS:

In software development project, errors can be injected at any stage during development. Because code is the only product that can be executed frequently and who’s actual behavior can be observed testing is the phase where the remaining errors from all the previous phases must be detected. Testing performs a critical role for quality assurance and for ensuring the reliability of the software. During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as expected. Due to this approach, dynamic testing can only ascertain the presence of errors in the program; the exact nature of errors is not usually decided by testing.

Testing a large system is a complex activity, so for a project incremental testing is generally performed, in which components and subsystems of the system are tested separately before integrating them to form the system for system testing. This form of testing introduces new issues of how to select components and how to combine them to from systems and subsystems

ERROR

It refers to the discrepancy between a computed, observed, or measured value and true, specified or theoretically correct value i.e. it refers to the difference between the actual output of the software and the correct value.

FAULT

It is a condition that causes system to fail in performing its required functions. A fault is the basic reason for software null function and is synonymous with the commonly used term bug.

FAILURE

It is the inability of a system or components to perform a require functions according to its specification. A software Failure occurs if the behavior of the software is different from the specified behavior. Faults have the potential to cause failures and their presence is a necessary but not sufficient condition for failure to occur.

10.4 TEST CASES AND TEST CRITERIA

Ideally, we would like to determine a set of test cases such that successful execution of all of them implies that there are no errors in the program. This ideal goal can't usually be achieved due to practical and theoretical constrains. Each test case costs money, as effort is needed to generate the test case, machine time is needed to execute the program for that test case, and more effort is needed to evaluate the results.
An ideal test case set is one that succeeds if there are no errors in the program. One possible ideal set of test cases is one that includes all the possible inputs to the program. This is often called exhaustive testing, however it is impractical and infeasible.

For a given program P and its specification S, a test selection criterion specifies the condition that must be satisfied by a set of test cases. T for example, if the criterion is that all statements in the program be executed at least one during testing, then a set of test cases T satisfied this criterion for a program P is the execution of P with T ensures that each statement in P is executed at least once.

There are two fundamental properties for a testing criterion: reliability and validity. A criterion is reliable if all the set of test cases that satisfied the criterion detect the same errors. A criterion is valid if for any error in the program there is some set satisfying the criterion that will reveal the error.

Testing can be mainly of two types:

When we know the specified function that a product has been designed to perform, test can be conducted that demonstrates each function is fully operational while at the same time searching for errors in each function. A black box test examines some fundamental aspects of a system with little regard for the internal logical structure of the software. Black box testing also called behavioral testing, focuses on the functional requirements of the software. Black box testing attempts to find errors in the following categories:

1. Incorrect or missing functions.
2. Interface errors.
3. Errors in data structures or external database access
4. Behavior or performance errors. 5. Initialization and termination errors.

10.6 WHITE BOX TESTING

When we know the internal workings of a product, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components have been adequately exercised. This is testing is sometimes called as glass box testing.

Using white box testing methods, the software engineer can derive test cases that…

1. Guarantee that all independent paths within a modules have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds.
4. Exercise internal data structures to ensure their validity.

In this project our main emphasis is on white box testing.

In order to test loops, we used the loop testing technique which is a white box testing technique. Most of loops used in this project belong to the category of simple loops.

We applied the following set of tests to loops, where n is the maximum number of allowable passes through the loop.

1. Skipped the loop entirely.
2. Allowed only one pass through the loop.
3. Allowed to passed through the loop.
4. Allowed m passed through the loop where m<n.
5. Allowed n-1, n, n+1 passes through the loop.

In order to test the control flow structures such as if conditions etc. we used the condition testing technique . The condition testing method focuses on testing each condition in the program. Condition testing strategies have two advantages. First, measurement of test coverage of a condition is simple. Second, the test coverage of conditions in program guidance for the generation of additional tests for the system. Branch testing is the simplest condition testing strategy. For a compound condition C which is composed of two or more simple conditions, Boolean operators and parenthesis, the true and false branches of C and every simple condition in C need to be executed at least once.

10.7 BOUNDARY VALUE ANALYSIS:

For reason that are not completely clear, a greater number of errors tend to occur at the boundaries of the input domain rather than in the 'center'. It is for this reason that boundary value analysis has been developed as a testing technique. Boundary value analysis leads to a selection of test cases that exercise bounding values. We used this approach to test several control statement in our project.

Guidelines for boundary value analysis are as follows:

If an input condition specifies a range bounded by values a and b, test cases should be designed with values a and b just above and below a and b.

If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and below minimum and maximum are also tested.

Apply the same guidelines to output conditions.

If internal program data structures have prescribe boundaries, be certain to design a test cases to exercise the data structures at its boundary.

10.8 FUNCTIONAL TESTING:

In functional testing the structure of the program is not considered. Test cases are decided solely on the basis of the requirements or specifications of the program or module, and the internal of module or the program are not considered for selection of test cases. Due to its nature, functional testing is often called " black box testing ".

10.9 STRUCTURAL TESTING:

In the structural approach, test cases are generated based on the actual code of the program or module to be tested. This structural approach is sometimes called “glass box testing”.

10.10 UNIT TESTING:

In this different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for the verification of the code produced during the coding phase and its goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Structural testing is best suited for this level.

In our project each form works like a unit. Some forms are used to display the data stored in the database and some forms are user to submit data to the database. Before submitting the data to the database, it is tested individually using front end capabilities. The tests that occurred as part of the unit tests are as follows:

The module interface was tested to ensure that information properly flows into and out of the program unit under test.

The local data structures were examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution.

Boundary conditions were tested to ensure that the module operates properly and boundaries established to limit or restrict processing.

All independent paths through the control structure were exercised to ensure that all statements in a module have been executed at least once.

Finally, all errors handling paths were tested.

10.10.2 INTEGRATION TESTING:

Integration technique is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associating with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

Integration can be either top down or bottom down.

Top down integration testing is an incremental approach to construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth first or breadth first manner.

Bottom up integration testing begins construction and testing with atomic modules (i.e. components at the lowest levels in the program structure). Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated.

In our project, we have used the bottom up approach. When all the forms were tested independently, they were integrated to form a module.

10.10.3 REGRESSION TESTING:

Each time a new module is added as part of integration testing, the software changes. New data flow paths are established, new I/o may occur, and new control logic is invoked. These changes cause problems with functions that previously worked flawlessly. In the context of an integration test strategy, regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

Regression testing was used frequently in our project.

VALIDATION TESTING:

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software testsvalidation testing –may begin. Validation can be defined in many ways. But a simple definition is that validation succeeds when software function in a manner that can be reasonably expected by the customer and supplier.

MAINTENANCE

MAINTENANCE

The maintenance phase is the last phase of the system development life cycle . Yet ,a life cycle is circular in that last activity leads back to the first. This means that the process of maintaining an information system is the process of returning to the beginning of the system development life cycle and repeating development steps until the change is implemented.

Four major activities occur within maintenance:

- Obtaining maintenance requests
- Transforming requests into changes
- Designing changes
- Implementing changes

Obtaining maintenance request that a formal process be established whereby user can submit change requests. Most companies have some sort of document to request new development, to report problem, or to request new system features with an existing system.

Once a request is received , analysis must be conducted to gain an understanding of the scope of the request. It must be determined how the request will affect the current system and duration of such project. Next , a change request can be transformed into a formal design change, which can be fed into the maintenance implementation phase.

There are several types of maintenance that you can perform on an information system.
These are :

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

CORRECTIVE MAINTENANCE

Corrective maintenance refers to the changes made to repair defects in the design, coding , or implementation of the system . Most corrective maintenance problems surface soon after installation. When the corrective maintenance problem surface , they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities.

ADAPTIVE MAINTENANCE:

Adaptive maintenance involves making changes to an information system to evolve its functionality to changing business needs or to migrate it toA different operating environment . Contrary to Corrective maintenance , adaptive maintenance is generally a small part of an organization's maintenance effort but does add value to the organization.

PERFECTIVE MAINTENANCE:

Perfective maintenance involves making enhancement to improve processing performance, interface usability or to add desired, but not necessarily required , system features.

PREVENTIVE MAINTENANCE

Preventive maintenance involves changes made to a system to reduce the chance of future system failure . Preventive maintenance might be used to increase the number of records that a system can process far beyond what is currently needed.

SCOPE OF FUTURE APPLICATION

The scope of future application for this system is:

A large product related database can be accommodated in our application in the future.

1.
2.

The related aspects like placement cell, staff details and customer data can be maintained more easily.

3.

In the purposed application examination monitoring, so that in future an increased number of customers can be accommodated by the system.

Vehicle registration details currently takes care of vehicle

4.
5.

The necessary steps have been taken in the proposed system to accommodate the forthcoming changes regarding the course scheduling and the batches thereof.

6.

The format of different types of reports can be suit the requirements in future though major modifications will be required but the base of the project will remain the same and all the working will remain same in this project.

BIBLOGRAPHY

- Complete Ref.

Author	Harbert Sheil
Publisher	TataMcGraw-Hil

Black Book of java

- Microsoft Office

- Software Engineering Apractitioner’s Approach

Author	Roger S. Pressman
Publication	Tata McGraw-Hill

- Fundamentals of Software Engineering

Author	Carlo Ghezzi, Mehdi Jazayeri, DinoMandrioli
Publisher	Prentice Hall of India