**Car Sales Trading Management System**

**PYTHON PROGRAMMING**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**



**Submitted by:-** **Submitted to :**

**Prikshit Singh Thakur** **Mr.Manpreet Singh** (Project Mentor)

**Roll no:- 1701003043**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Solitaire Infosys INC.**

**C-110, Industrial Area, Phase 7, Mohali 160055**

**Rayat-Bahra University,Sahauran,**

**Mohali**

**ACKNOWLEDGEMENT**

We wish to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide. Mr.Manpreet Singh , Institute of Science and Technology, Rayat-Bahra University for his valuable guidance, encouragement and help for completing this work. His useful suggestions for this whole work and co-operative behavior are sincerely acknowledgement .We would like to express our sincere thanks to Mr.Manpreet Sir for giving us this opportunity to undertake this project also our principal who gave the opportunity to do the project on the topic Car Trading Management System.

We would also like to thank Mr.Manpreet Singh for his whole hearted support. With his support and guidance we learnt new things and able to research more information related to our topic .Last but not least, we would like to express our sincere thanks to all our friends and other who helped us either directly or indirectly during this project. The quality time spent with our teacher and friends will remain forever in our heart and minds.

**ABSTRACT**

Car Trading Management System Project provides solution for showrooms for organizing data using software application. This software application helps administrator to update data in step by step process while selling vehicle to customer. Using this system management team can update customer information, vehicle information, payment details, Insurance details, take order etc.

Car Trading Management software is presently used in every car for computerizing entire system. This software package includes inventory management module, daily management details, image uploads, sales reports. There is no other scripting or server end management for this system.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page No.** |
|  |  |  |
| 1 | Introduction | 1-4 |
|  |  |  |
| 2 | SRS | 5-9 |
|  |  |  |
| 3 | Architecture Diagram | 10-14 |
|  |  |  |
| 4 | Project methodology | 15 |
|  |  |  |
| 5 | Screen shots | 16-29 |
|  |  |  |
| 6 | Conclusion and Future Scope  References | 30 |
|  |  |  |

**List of Figures:-**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| **1** | DFD Diagram | **10-11** |
| **2** | Class Diagram | **12** |
| **3** | Use Case Diagram | **13** |
| **4** | Sequencediagram | **14** |

**CHAPTER = 1**

**INTRODUCTION**

Car Trading Management System project main objective is to manage step by step process in purchasing and selling vehicles like cars, bikes from show room. This system automates computerize total process included in show rooms from customer request to end process till vehicle is purchased by customer. This system can helps to records of customer registration details, show room branch details, vehicle booking requests, repaired vehicles information vehicle details and other details. Car Trading Management System is to developing to computerize the process in the Car Trading Management System. In the showrooms, there are many Variety of vehicles based on the models and vehicle type like two wheeler, four wheeler etc. In the management process, a systematic approach is needed to manage the records of vehicles information details, vehicle bookings and vehicle delivery. This project is a complete management system for the vehicle show room management. It is very helpful for show room module, which contains the vehicle details and image with zooming options. Through this module officials can show the vehicle details to the customer before physically viewing the vehicle and booking and delivery , delivery modules are used to track the orders and delivered vehicles in the showroom, This system can also reduces manual work and total number of employees. In this project includes inventory management module, daily management details, image uploads, sales reports. It provides information about customer details, Sales reports .It provides information about customer details, Sales details, Company items and services and also Vehicle Details. It also maintain the records related to purchase, sales, returns, stock updating, cash and bank flows.

**CHAPTER = 2**

**SRS(SOFTWARE REQUIREMENT SPECIFICATION-:**

A Software requirements specification (SRS) document describes the intended purpose, requirements, and nature of software/application/project to be developed.

To prepare an SRS document, you would need to have a functional knowledge of your project or application, knowledge of software/hardware/technology to be used.

**INDEX:**

1) Introduction

 1.1) Purpose

2) Project scope

3) Feasibility Study

3.1) Economic Feasibility

 3.2)Technical Feasibility

3.3) Operational Feasibility

4)Intended Audience and Document Overview

5)Definitions, Acronyms and Abbreviations

6)Product Perspective-

6.1)Design and Implementation constraints

7) Requirements

7.1) Functional Requirements

  7.2) Non- Functional Requirements

**1) Introduction:-**

Vehicle showroom management system project main objective is to manage step by step process in purchasing and selling vehicles like cars, bikes from show room. This system automates computerize total process included in show rooms from customer request to end process till vehicle is purchased by customer.

**1.1) Documentation Purpose:-**

The purpose of this document is to present a detailed description of vehicle management system. It will explain the purpose and feature of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to stimuli. This document is intended for both the stakeholders and the developers of the system. The features discussed in this document are mainly functional requirements.These requirements are described in software development perspective to easily grasp by use. It also covers the non-functional requirements like performance, reliability, availability, security and maintainability.

**3) Project Scope:-**

All the resources essential for the growth of the software as well as the preservation of the software is existing in the organization here we are utilizing the resources which are previously available in the system by using this we can get the maximum usability of the same resources. Even after the development, the organization will not be in a condition to provide more in the organization.

In any it will be designed elements with input and output and must focus on database device or how data should be organized around user requirements. The main objective of software arrangement is to provide a frame work that enables the manger to make practical estimates made within a incomplete time frame at the beginning of the software project should be simplified regularly as the project progresses.

**4.1)Technical Feasibility**

The technology needed for the proposed system that we are going to develop is available. We can work for the project is done with current equipment existing tools like python. We can develop our system still using this technology if needed to upgrade.In future ,if we want to use new technology for our system it is possible .hence the system that we are going to develop will successfully satisfy the needs of the system for technically feasibility

**4.2)Economic Feasibility**

Since the system is developed as a part of project work. There is no manual cost to spend for the proposed system. Also all the resources are already available .it give an indication that the system is economically possible for development economic justification ids generally “the bottom line” consideration for most system. Hence the project that we are going to develop won’t require enormous amount of money to be developed so it will be economically feasible.

**4.3)Operational Feasibility**

The user interface will be user friendly and no training will be required to use the application the solution proposed for our project as operationally workable and most likely conveninent to solve the irrelevant document.

**5)Intended Audience and Document Overview:-**

This document will be the first guide for prospective software developers, project managers, testers and most importantly users of spam mail detector. This document contains software functionality, software and hardware requirements and user documentation.

• Developer:

The developer who wants to read, change, modify or add new requirements into the existing program may need first to consult this document and update the requirements in appropriate manner so as not to change the actual purpose of the system or make the system inconsistent.

• User:

The user of this program reviews the diagram and the specification provided in the document and check to determine whether the software has all the suitable requirements and if the software developer has the implemented all of them. He can also consult the user guide in the event of any confusion for clarifications.

• Tester:

The tester needs this document to prepare his test cases to validate that the initial requirements of this project is actually implemented in the deliverable.

**6) Definitions, Acronyms and Abbreviations:-**

• Python: It’s a very-high-level dynamic object-oriented programming language.It’s designed to be easy to learn, read and program. Python is a general purpose language, which means that many things are made easy like string processing, reading/writing files, sockets, websites, databases, GUI’s. Also it’s dynamic i.e. new code can be executed during runtime without limitations.

• Anaconda: Anaconda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Anaconda quickly installs, runs and updates packages and their dependencies. Anaconda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

**7) Product Perspective:-**

The product is supposed to be an open source, under the (GNU) General Public License. It is a web bases system implementing client – server model. The system provides simple mechanism for users to share and acquire knowledge.

The following are the main features that are include in portal system:-

* User account: The system allows the user to create their accounts in the system and provide features of updating and viewing profiles.
* Cross platform support: Offers operating support for most of the known and commercial operating systems.
* Search: It is simply local search engine based on key words.
* Discussion Forum: Provides users with a platform to discuss and help each other with their problems.

**7.1)Design and Implementation constraints:-**

This system is provisioned to be built on the Django Framwork which is highly flexible Decision regarding which database to use should be taken considering the fact that data being exchanged or stored is large, and the appropriate data management system will yield efficient performance.

**8)Requirement analysis-:**

**8.1) Functional requirement:-**

* Enable a user to view questions and their corresponding answers.
* Enable a logged in user to ask questions.
* Enable a logged in user to post answers.

**8.2) Non-functional requirement:-**

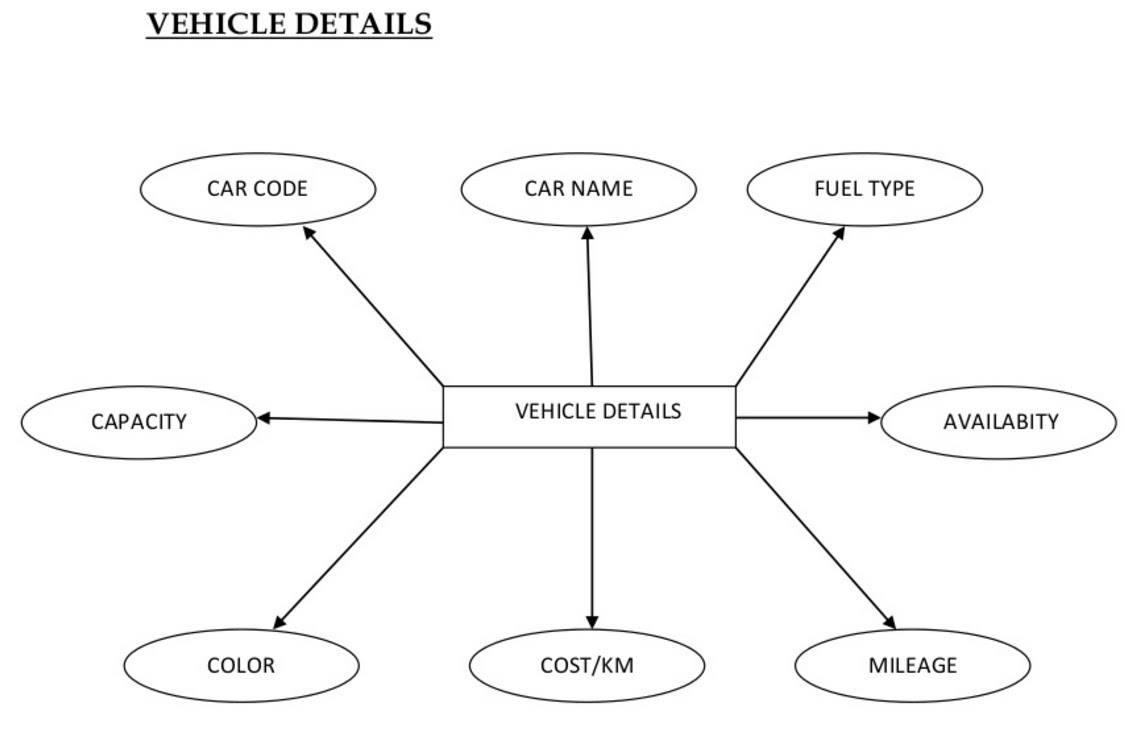
It’s basically relates with the performance, safety, reliabilityand security of the software

**CHAPTER = 3**

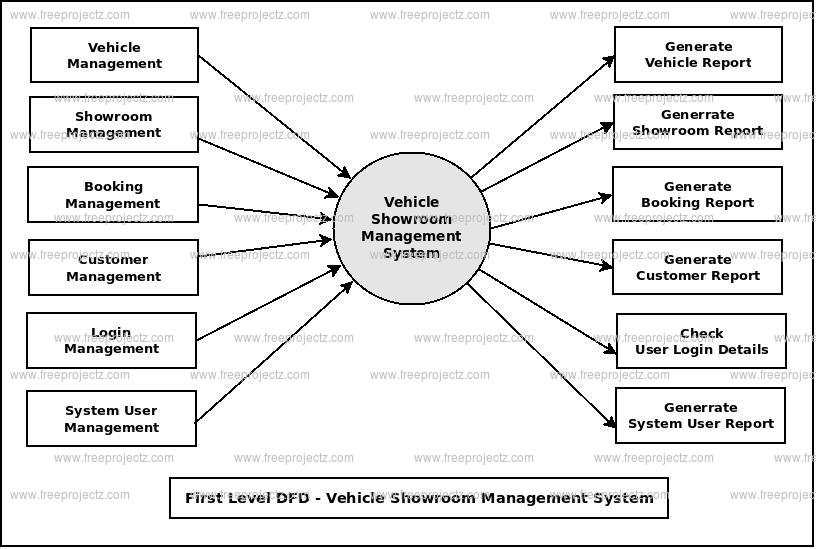
**Data Flow Diagram:-**

It is a directed graph where nodes represents processing activity and arc represent data items transmitted between processing nodes**.**

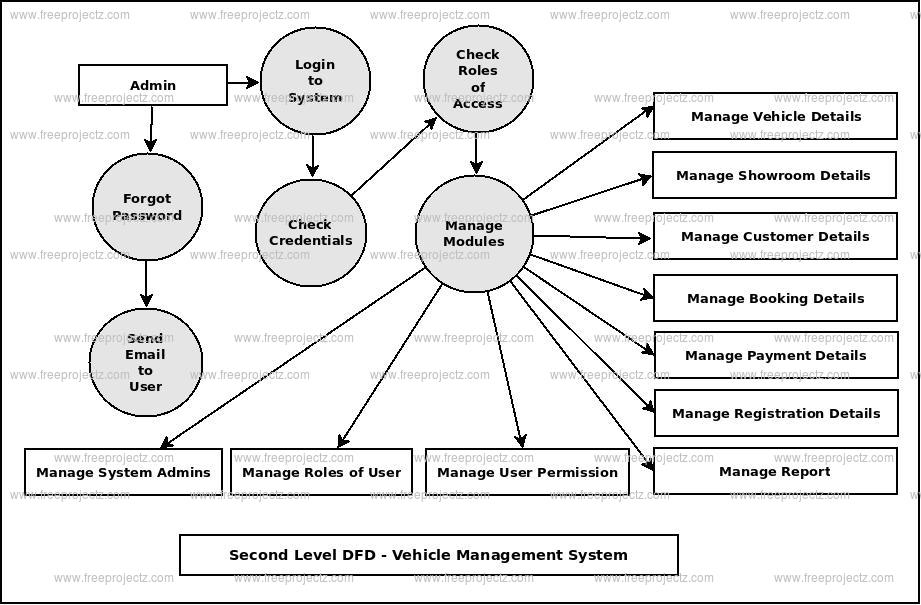
**Level 0**



**Level 1**



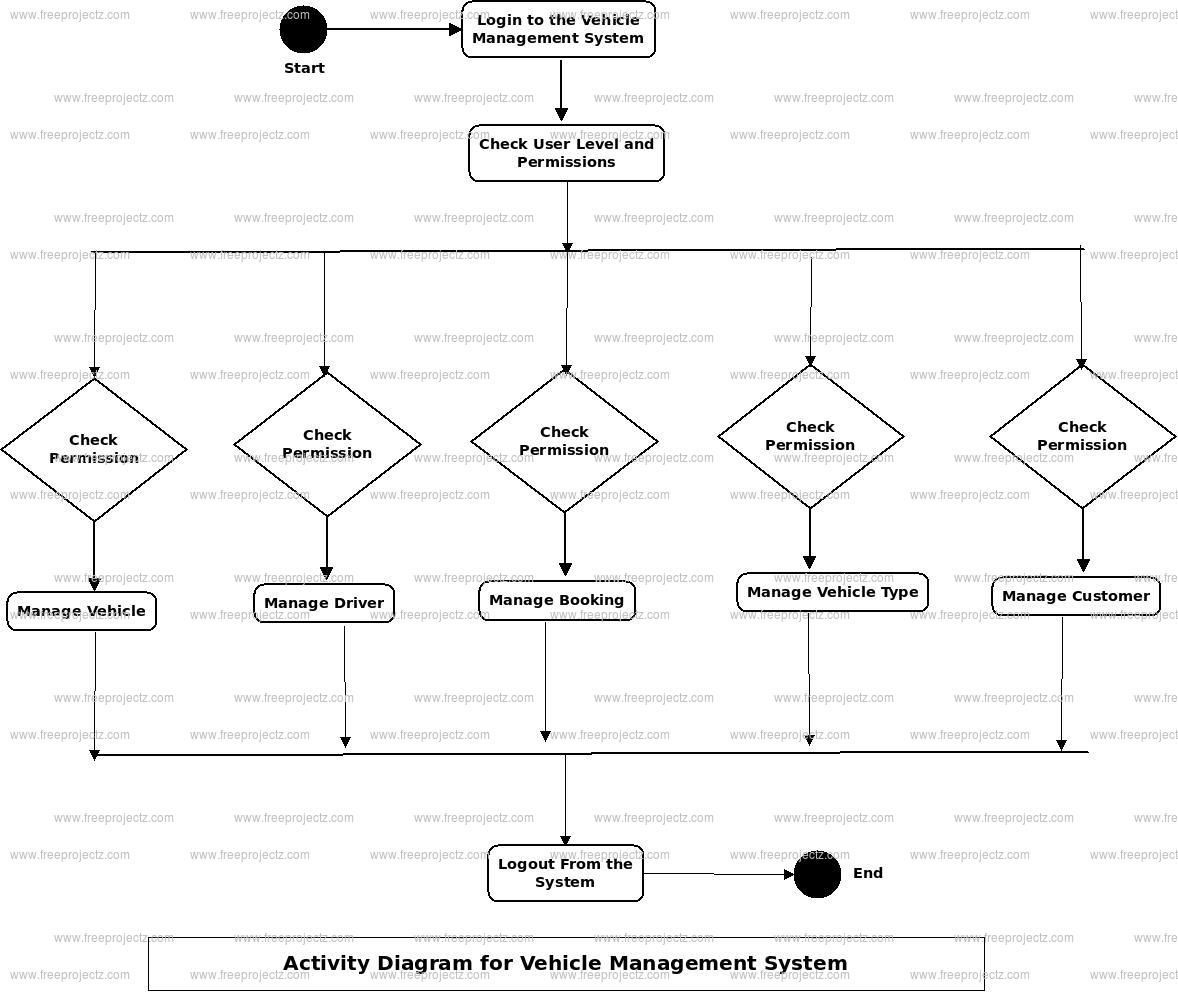
**Level 2**



### UML DIAGRAM:-

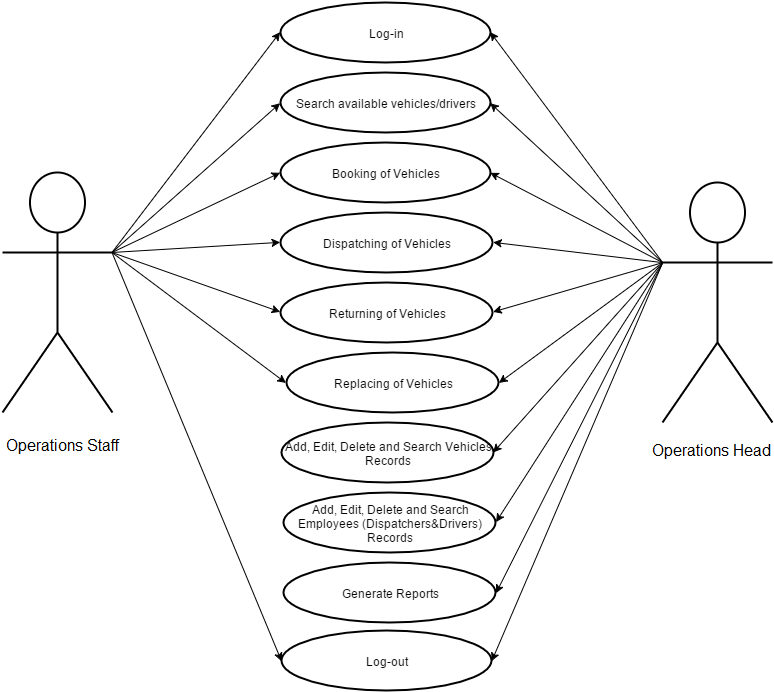
### Classdiagram:

These diagrams are considered as the static structure of an application. The class diagrams include member variables and member functions or methods with the relationship among them. In the Figure member variables and methods are clearly specified for each module.



### User casediagram:-

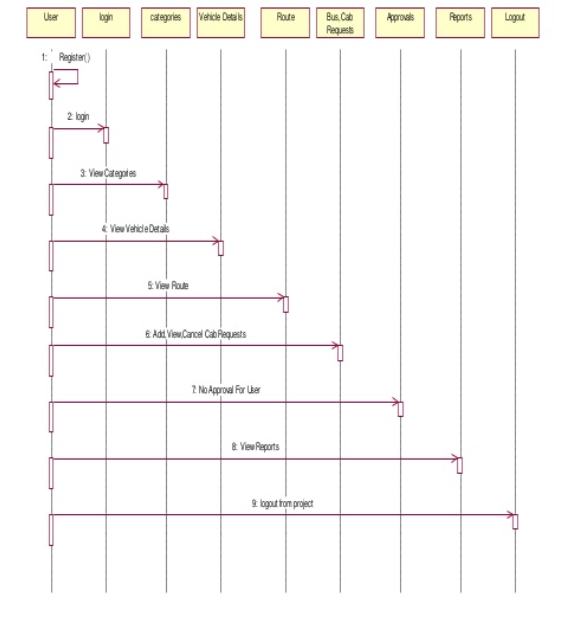
User case diagram typically define the use of a system. It depicts the relationship and communication between the actors and a system to reach the targets of the system. Any external machine or a human can be applicable as an actor. In the below figure, the activities between the master server and among the clients are clearly represented.



### 

### Sequencediagram:-

Sequence diagram can also be referred as even diagram or timing diagrams. It typically defines the processes including in a system, how they interact with each other and its priorities. The object interactions perform in a specified sequence. It also provides a clear picture of objects and classes along with ordering of messages among the objects for specific function. This figure clearly represents the exchange of messages and activities between the modules.



**CHAPTER = 4**

**PROJECT-METHODOLOGY**

Top of Form

Bottom of Form

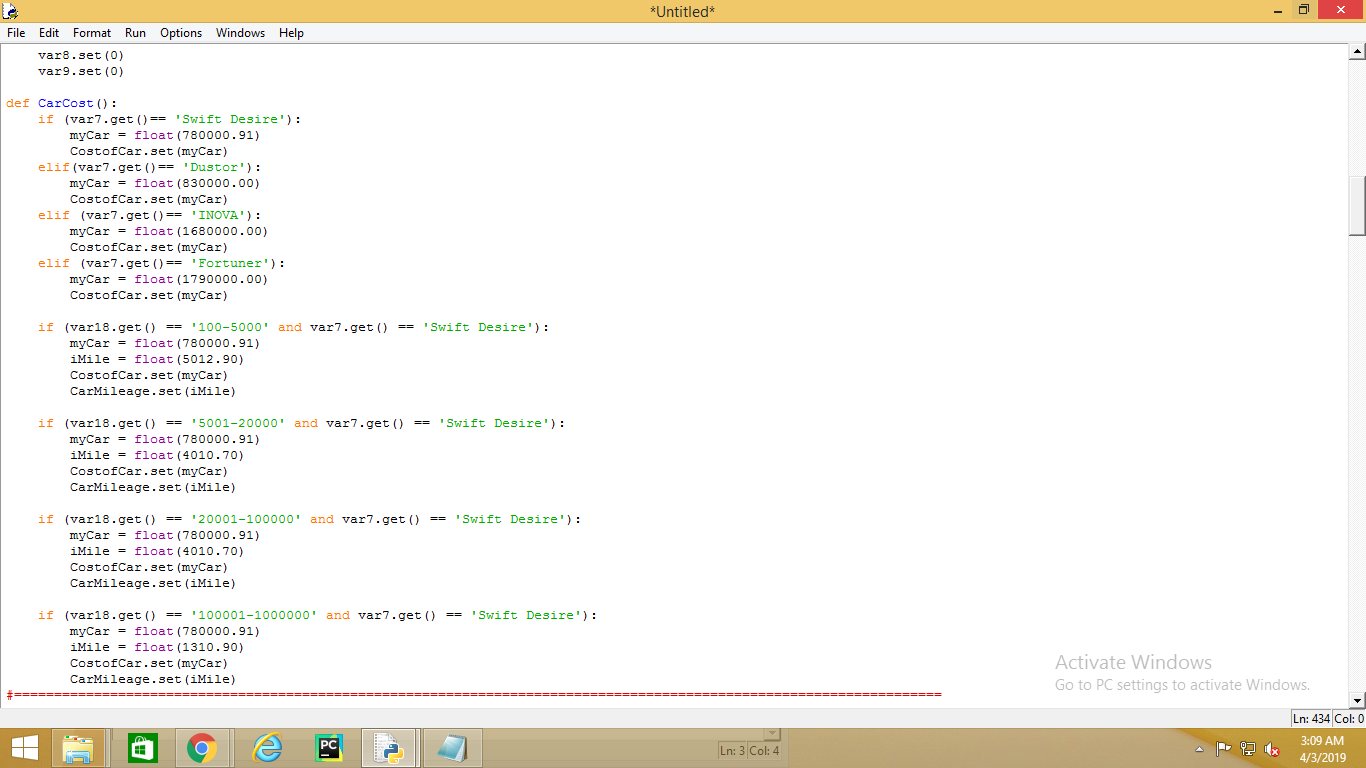
For companies that owns a fleet of vehicles available for use by their employees, the existing process is like to submit requests and be assigned a vehicle manually is a time consuming one. The proposed project is a single centralized framework that would integrate various disparate vehicle management processes such as scheduling, maintenance, sales, insurance tc. The proposed web based vehicle management system helps in streamlining and easing the process of managing company vehicles. The solution also provides the client with the means to monitor vehicle and driver performances.

The solution required to enable client to maintain a database of vehicles. Employees need to be provided with the means to submit and track vehicle requests online. Administration managers will be provided with the means to process such requests and assign vehicles. The facility to track vehicle insurance, fuel consumption, service etc is also required.

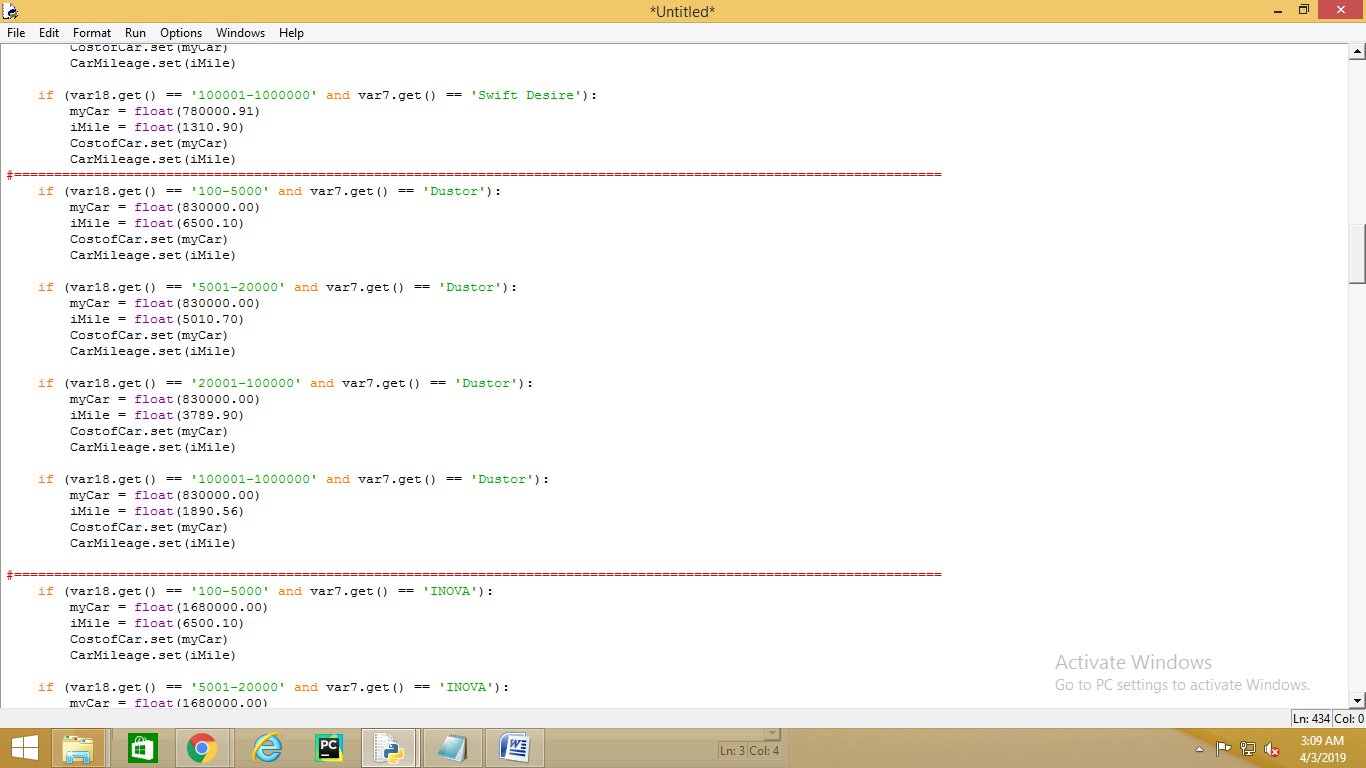
**CHAPTER = 5**

**SCREEN-SHOTS**

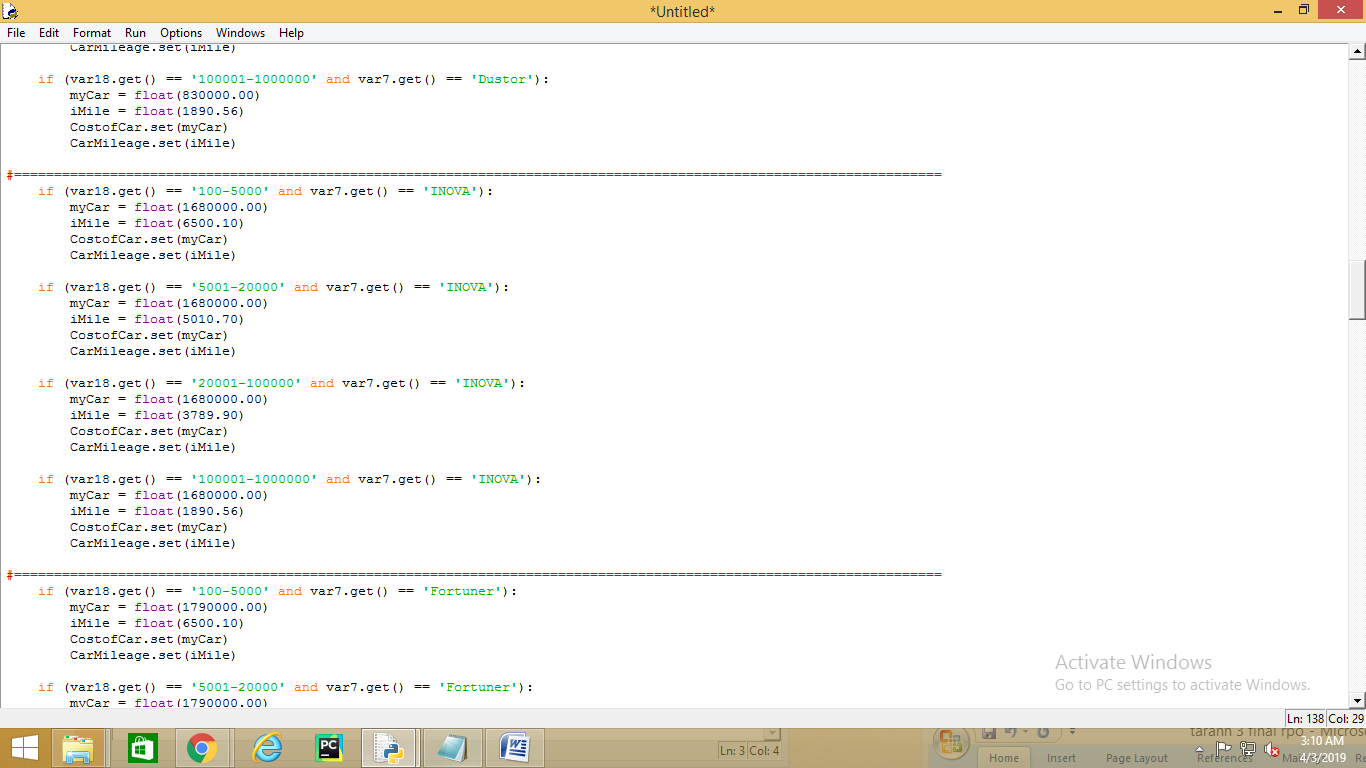
DEFINING CarCostOf SWIFT DEZIRE



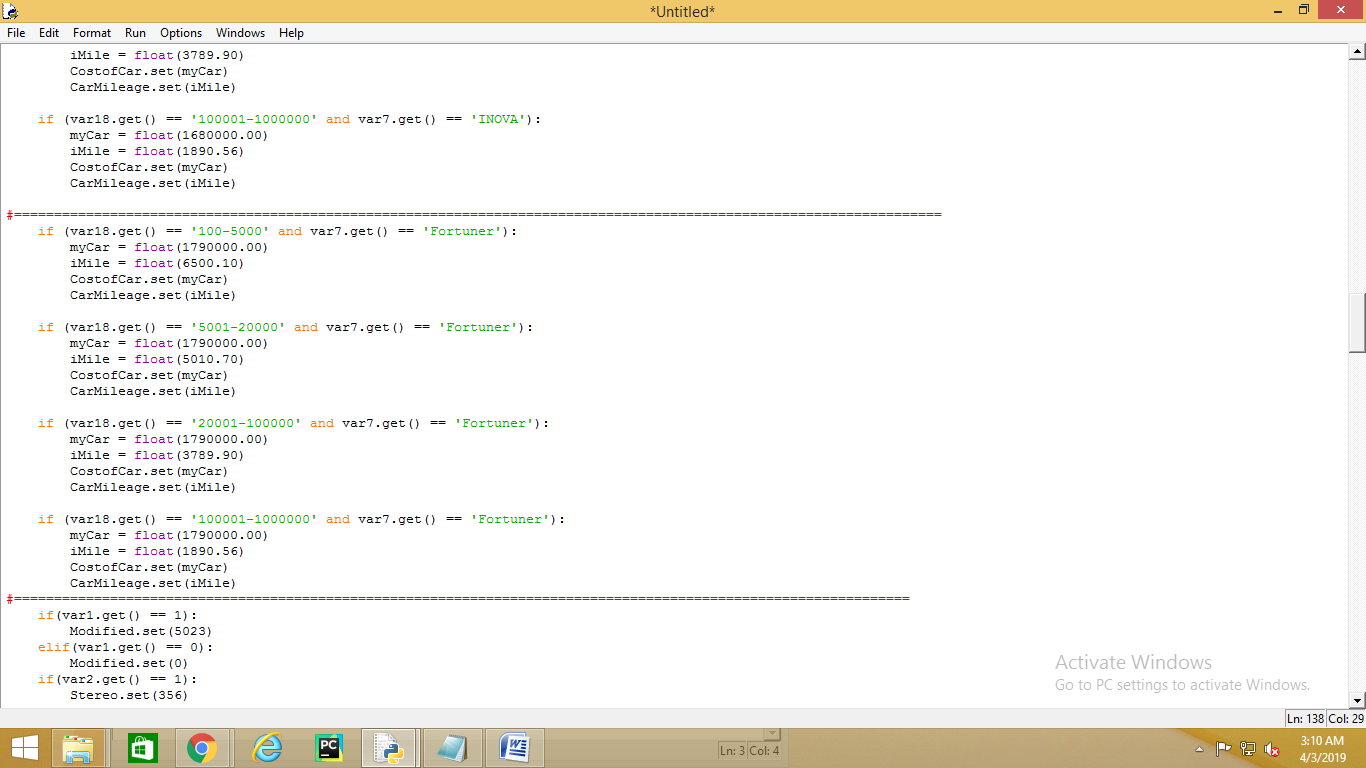
DEFINING CarCostOf DUSTOR

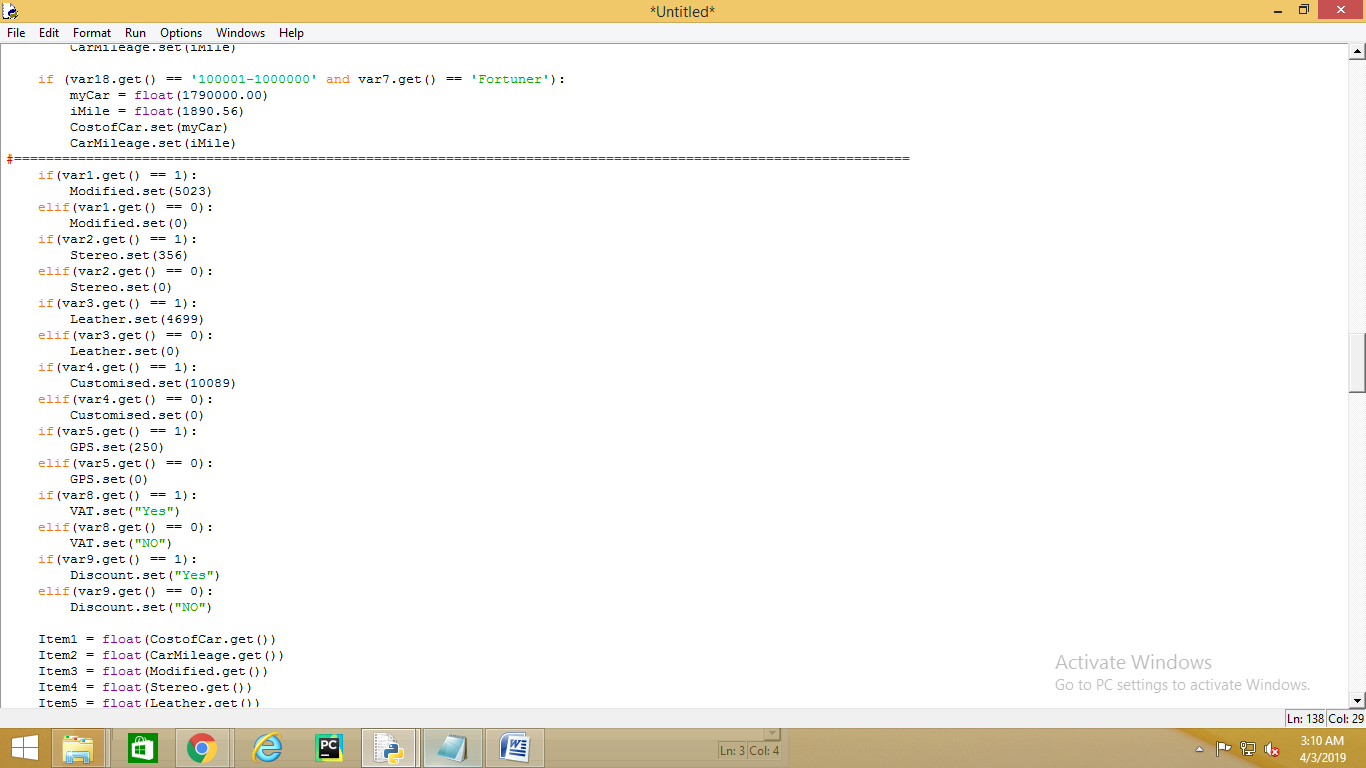


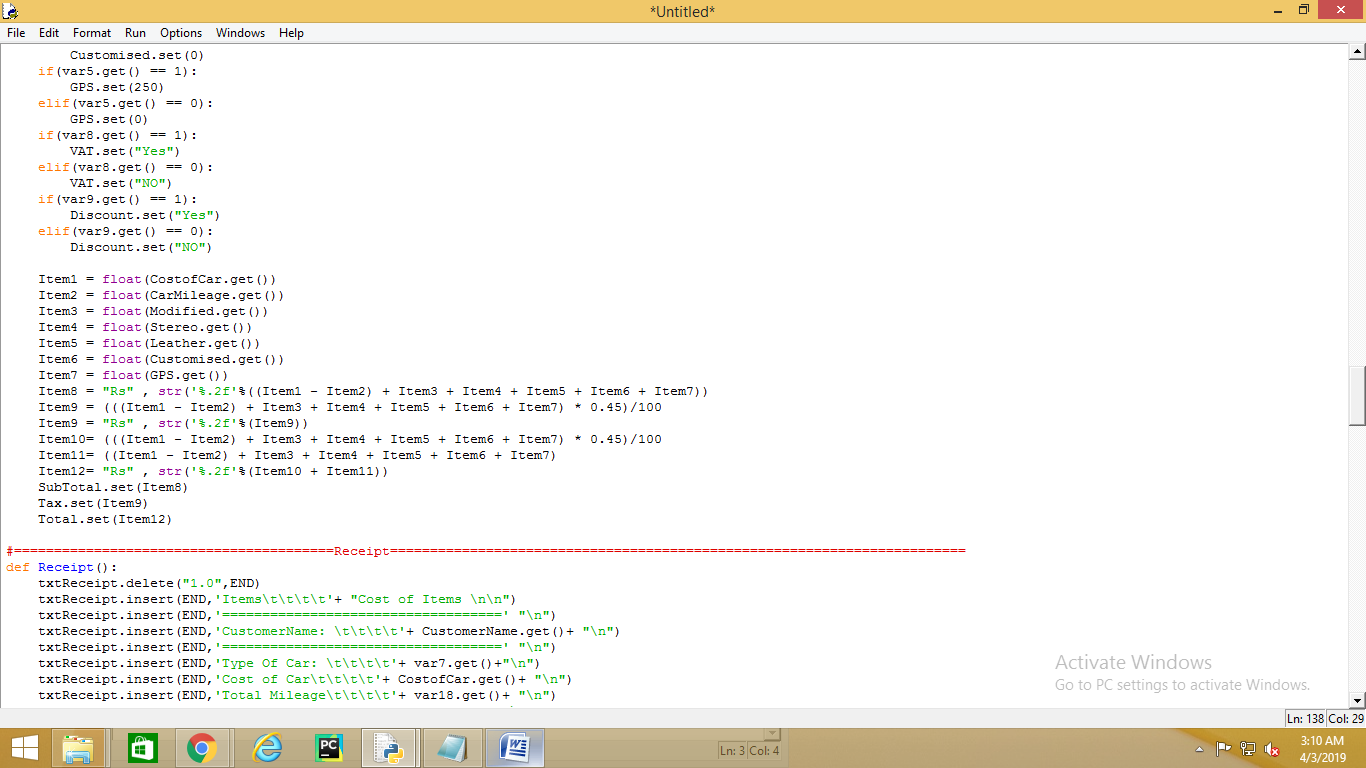
DEFINING CarCostOf INOVA



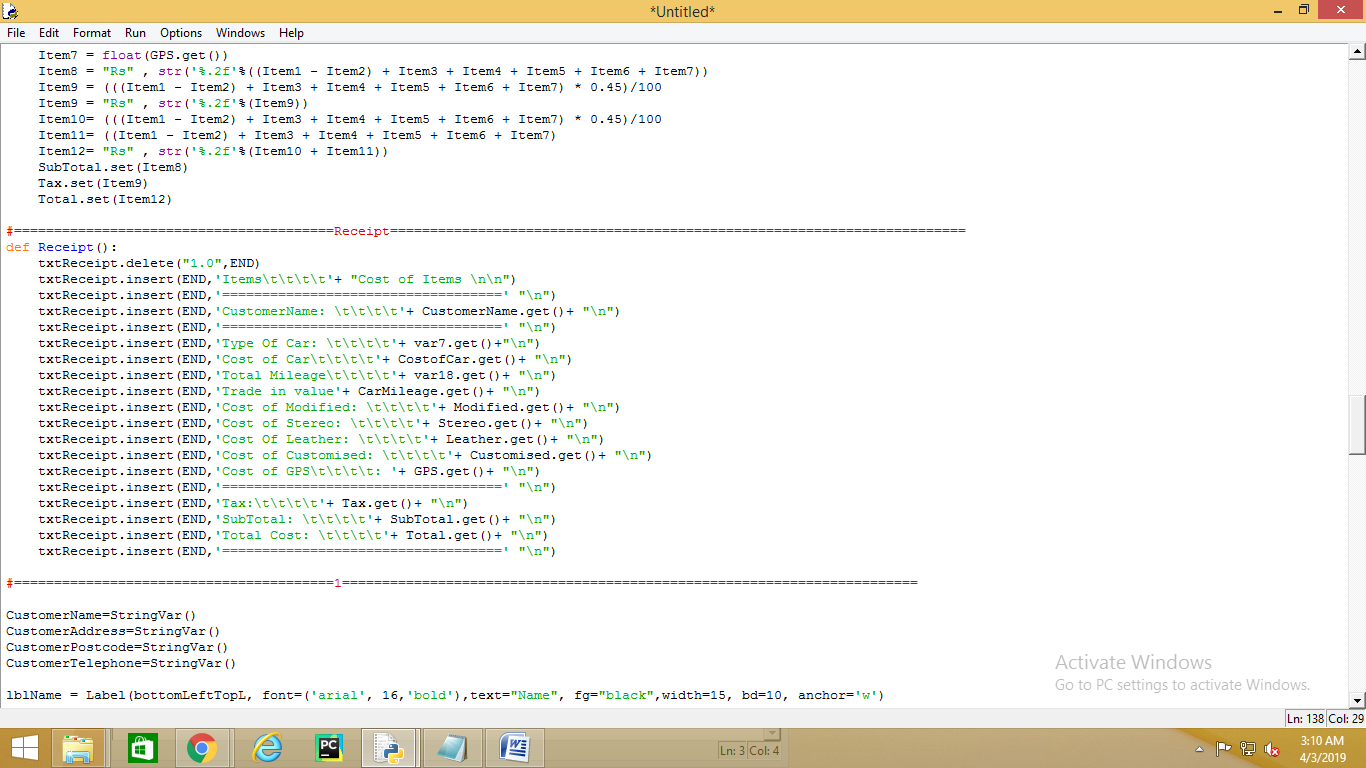
DEFINING CarCostOf Fortuner

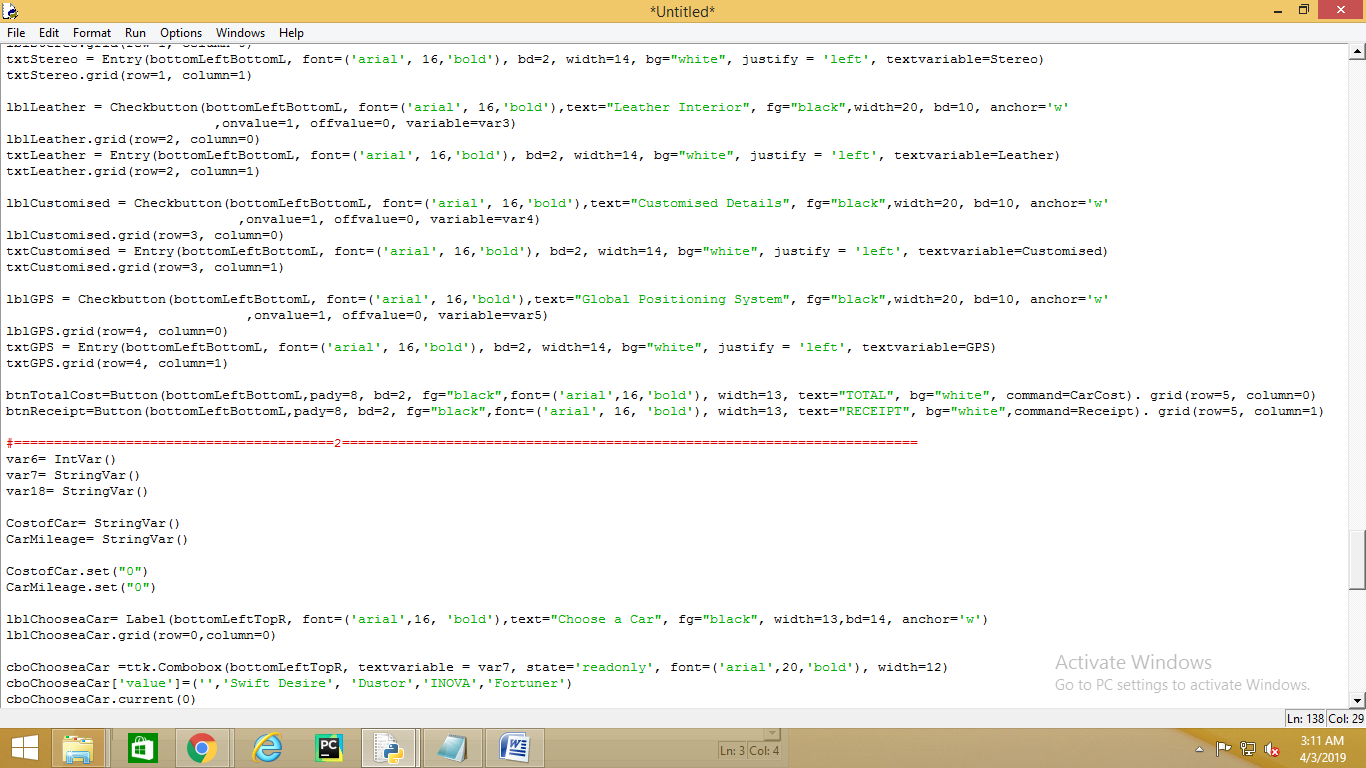




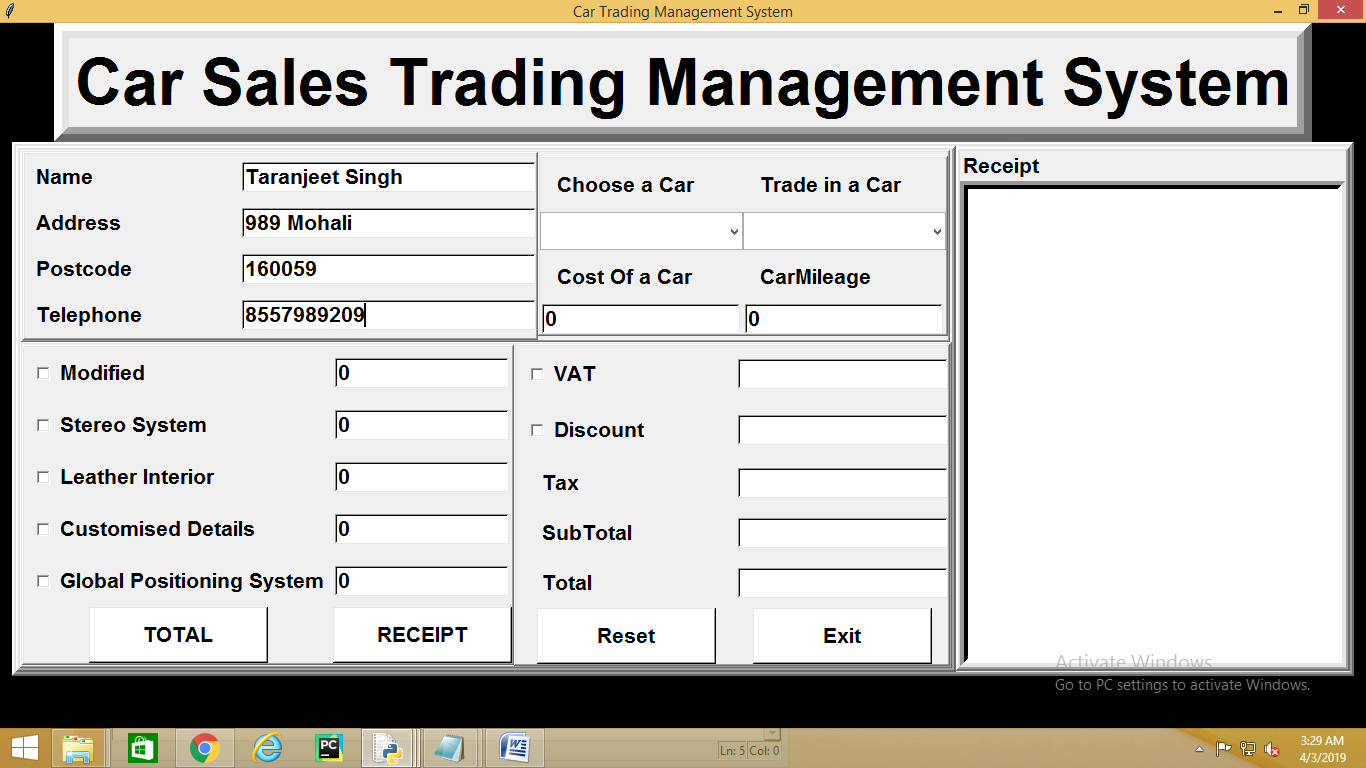


Defined Receipts

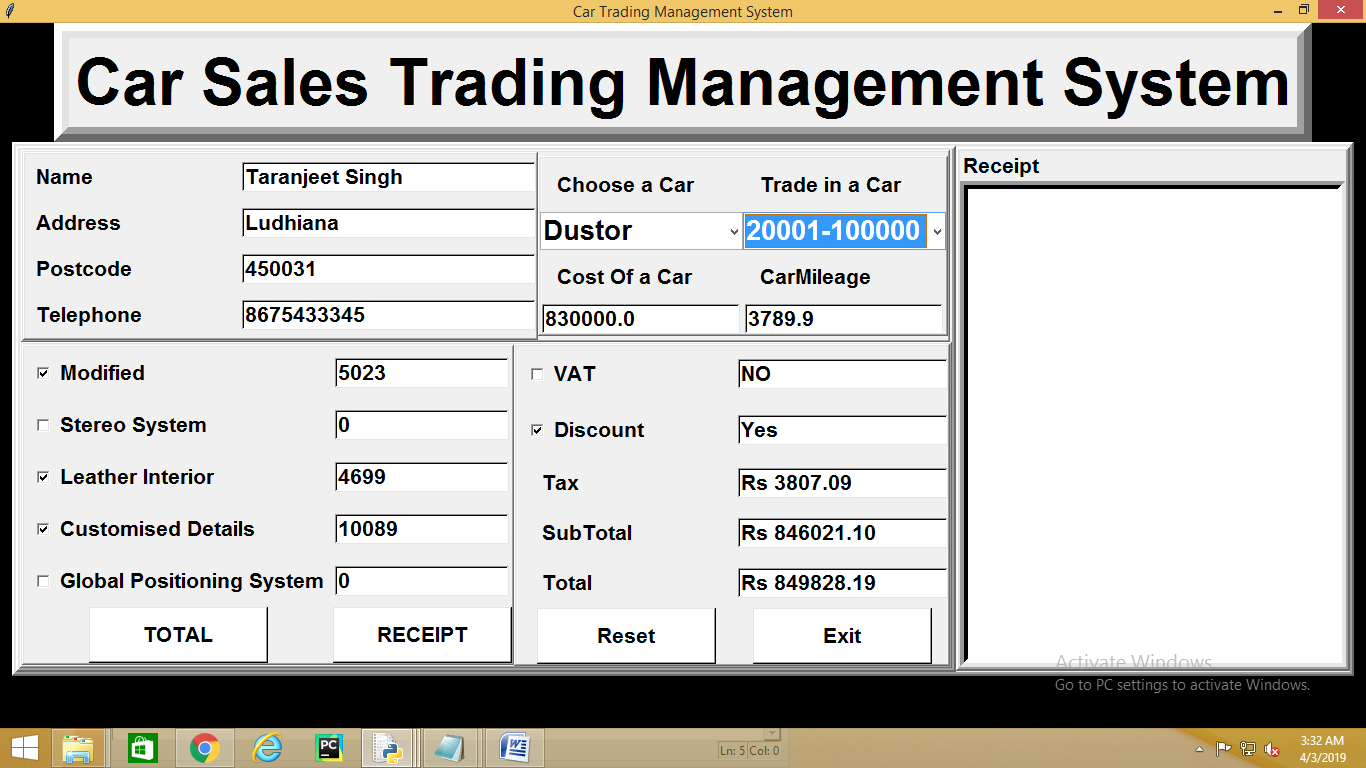




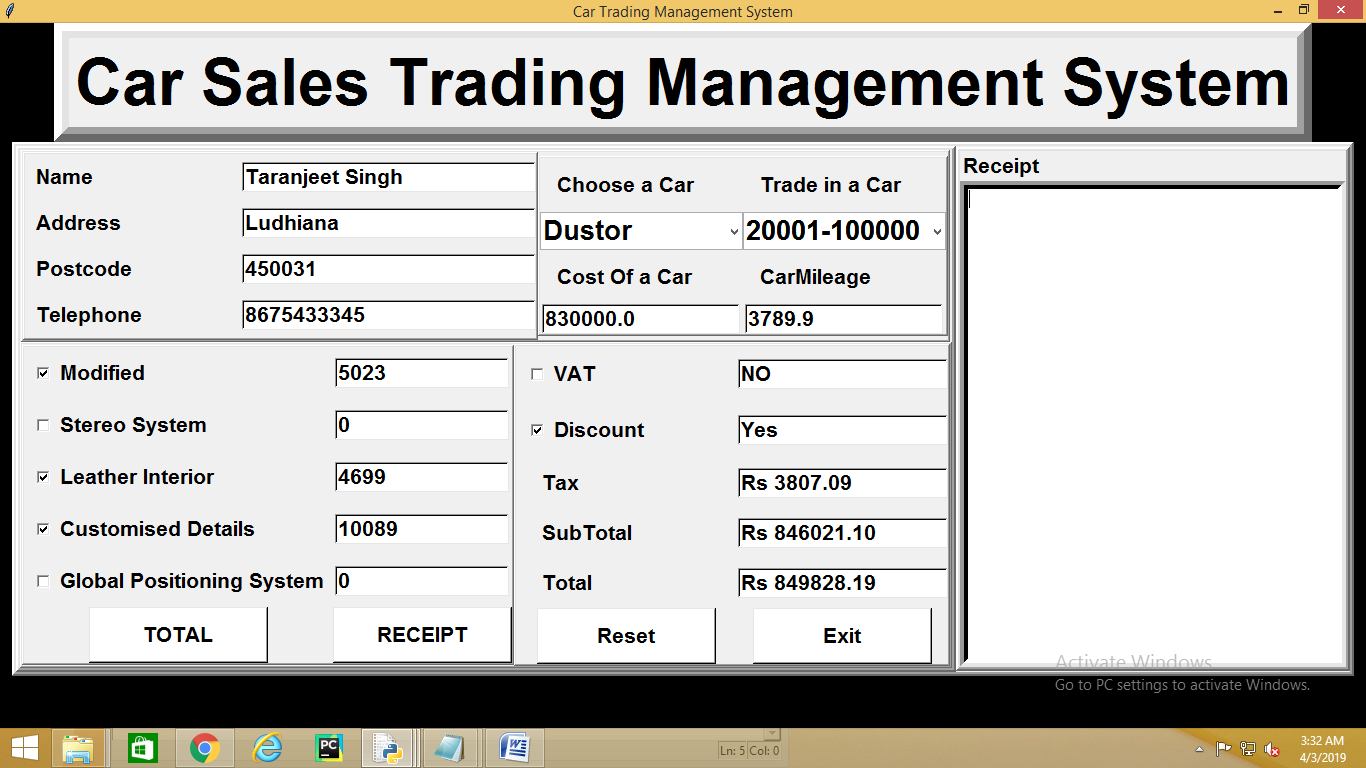
**First frame:-**

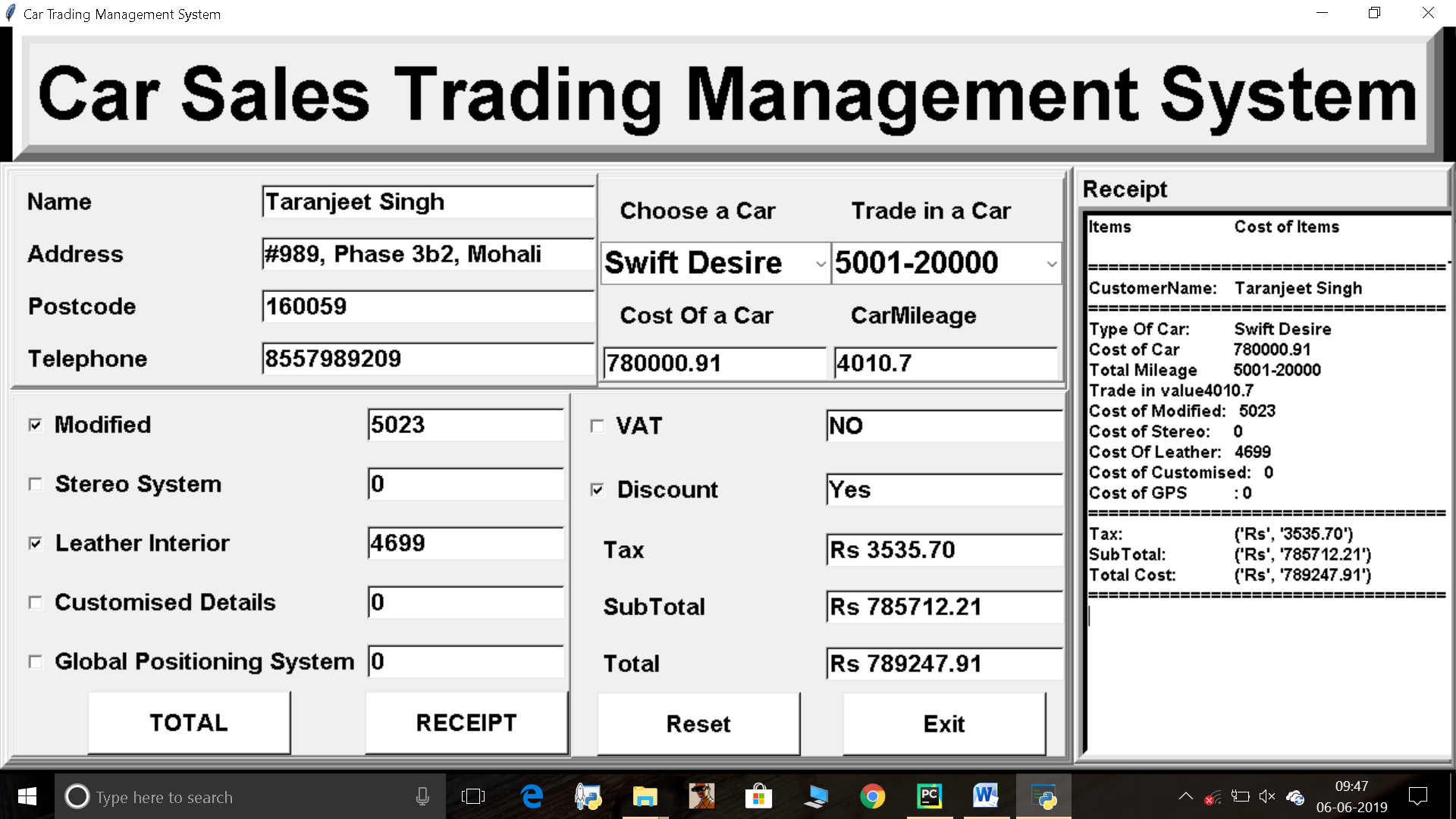


**All frames are in one:-**



**Receipt added with Other Frames**





Coding:-

from tkinter import\*

from tkinter import ttk

root = Tk()

root.geometry("1352x650+0+0")

root.title("Car Trading Management System")

root.configure(background='black')

Tops = Frame(root, width=1350, height=100, bd=15, relief="raise")

Tops.pack(side=TOP)

lblInfo = Label(Tops, font=('arial', 49,'bold'),text="Car Sales Trading Management System", bd=5, anchor='w')

lblInfo.grid(row=0, column=0)

bottom = Frame(root, width=1350, height=600, bd=4, relief="raise")

bottom .pack(side=TOP)

bottomLeft = Frame(bottom, width=1000, height=600, bd=4, relief="raise")

bottomLeft .pack(side=LEFT)

#=================================================================================================================

bottomLeftTop = Frame(bottomLeft, width=1000, height=300, bd=2, relief="raise")

bottomLeftTop .pack(side=TOP)

bottomLeftTopL = Frame(bottomLeftTop, width=500, height=200, bd=2, relief="raise")

bottomLeftTopL .pack(side=LEFT)

bottomLeftTopR = Frame(bottomLeftTop, width=500, height=200, bd=2, relief="raise")

bottomLeftTopR .pack(side=RIGHT)

#=================================================================================================================bottomLeftBottomL

bottomLeftBottom = Frame(bottomLeft, width=1000, height=300, bd=2, relief="raise")

bottomLeftBottom .pack(side=BOTTOM)

bottomLeftBottomL = Frame(bottomLeftBottom, width=500, height=400, bd=2, relief="raise")

bottomLeftBottomL .pack(side=LEFT)

bottomLeftBottomR = Frame(bottomLeftBottom, width=500, height=400, bd=2, relief="raise")

bottomLeftBottomR .pack(side=RIGHT)

#=================================================================================================================

bottomRight = Frame(bottom, width=350, height=600, bd=4, relief="raise")

bottomRight .pack(side=RIGHT)

#========================================Exit========================================================================

def iExit():

iExit=messagebox.askyesno("Car Showroom System", "Confirm if you want to exit")

if iExit> 0:

root.destroy ()

return;

#========================================Reset========================================================================

def Reset():

Modified.set("0")

Stereo.set("0")

Leather.set("0")

Customised.set("0")

GPS.set("0")

CostofCar.set("0")

CarMileage.set("0")

CustomerName.set("")

CustomerAddress.set("")

CustomerPostcode.set("")

CustomerTelephone.set("")

VAT.set("")

Discount.set("")

Tax.set("")

SubTotal.set("")

Total.set("")

lblReceipt.delete("1.0",END)

var1.set(0)

var2.set(0)

var3.set(0)

var4.set(0)

var5.set(0)

var6.set(0)

var7.set(0)

var8.set(0)

var9.set(0)

def CarCost():

if (var7.get()== 'Swift Desire'):

myCar = float(780000.91)

CostofCar.set(myCar)

elif(var7.get()== 'Dustor'):

myCar = float(830000.00)

CostofCar.set(myCar)

elif (var7.get()== 'INOVA'):

myCar = float(1680000.00)

CostofCar.set(myCar)

elif (var7.get()== 'Fortuner'):

myCar = float(1790000.00)

CostofCar.set(myCar)

if (var18.get() == '100-5000' and var7.get() == 'Swift Desire'):

myCar = float(780000.91)

iMile = float(5012.90)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '5001-20000' and var7.get() == 'Swift Desire'):

myCar = float(780000.91)

iMile = float(4010.70)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '20001-100000' and var7.get() == 'Swift Desire'):

myCar = float(780000.91)

iMile = float(4010.70)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '100001-1000000' and var7.get() == 'Swift Desire'):

myCar = float(780000.91)

iMile = float(1310.90)

CostofCar.set(myCar)

CarMileage.set(iMile)

#====================================================================================================================

if (var18.get() == '100-5000' and var7.get() == 'Dustor'):

myCar = float(830000.00)

iMile = float(6500.10)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '5001-20000' and var7.get() == 'Dustor'):

myCar = float(830000.00)

iMile = float(5010.70)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '20001-100000' and var7.get() == 'Dustor'):

myCar = float(830000.00)

iMile = float(3789.90)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '100001-1000000' and var7.get() == 'Dustor'):

myCar = float(830000.00)

iMile = float(1890.56)

CostofCar.set(myCar)

CarMileage.set(iMile)

#====================================================================================================================

if (var18.get() == '100-5000' and var7.get() == 'INOVA'):

myCar = float(1680000.00)

iMile = float(6500.10)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '5001-20000' and var7.get() == 'INOVA'):

myCar = float(1680000.00)

iMile = float(5010.70)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '20001-100000' and var7.get() == 'INOVA'):

myCar = float(1680000.00)

iMile = float(3789.90)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '100001-1000000' and var7.get() == 'INOVA'):

myCar = float(1680000.00)

iMile = float(1890.56)

CostofCar.set(myCar)

CarMileage.set(iMile)

#====================================================================================================================

if (var18.get() == '100-5000' and var7.get() == 'Fortuner'):

myCar = float(1790000.00)

iMile = float(6500.10)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '5001-20000' and var7.get() == 'Fortuner'):

myCar = float(1790000.00)

iMile = float(5010.70)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '20001-100000' and var7.get() == 'Fortuner'):

myCar = float(1790000.00)

iMile = float(3789.90)

CostofCar.set(myCar)

CarMileage.set(iMile)

if (var18.get() == '100001-1000000' and var7.get() == 'Fortuner'):

myCar = float(1790000.00)

iMile = float(1890.56)

CostofCar.set(myCar)

CarMileage.set(iMile)

#================================================================================================================

if(var1.get() == 1):

Modified.set(5023)

elif(var1.get() == 0):

Modified.set(0)

if(var2.get() == 1):

Stereo.set(356)

elif(var2.get() == 0):

Stereo.set(0)

if(var3.get() == 1):

Leather.set(4699)

elif(var3.get() == 0):

Leather.set(0)

if(var4.get() == 1):

Customised.set(10089)

elif(var4.get() == 0):

Customised.set(0)

if(var5.get() == 1):

GPS.set(250)

elif(var5.get() == 0):

GPS.set(0)

if(var8.get() == 1):

VAT.set("Yes")

elif(var8.get() == 0):

VAT.set("NO")

if(var9.get() == 1):

Discount.set("Yes")

elif(var9.get() == 0):

Discount.set("NO")

Item1 = float(CostofCar.get())

Item2 = float(CarMileage.get())

Item3 = float(Modified.get())

Item4 = float(Stereo.get())

Item5 = float(Leather.get())

Item6 = float(Customised.get())

Item7 = float(GPS.get())

Item8 = "Rs" , str('%.2f'%((Item1 - Item2) + Item3 + Item4 + Item5 + Item6 + Item7))

Item9 = (((Item1 - Item2) + Item3 + Item4 + Item5 + Item6 + Item7) \* 0.45)/100

Item9 = "Rs" , str('%.2f'%(Item9))

Item10= (((Item1 - Item2) + Item3 + Item4 + Item5 + Item6 + Item7) \* 0.45)/100

Item11= ((Item1 - Item2) + Item3 + Item4 + Item5 + Item6 + Item7)

Item12= "Rs" , str('%.2f'%(Item10 + Item11))

SubTotal.set(Item8)

Tax.set(Item9)

Total.set(Item12)

#========================================Receipt========================================================================

def Receipt():

txtReceipt.delete("1.0",END)

txtReceipt.insert(END,'Items\t\t\t\t'+ "Cost of Items \n\n")

txtReceipt.insert(END,'===================================' "\n")

txtReceipt.insert(END,'CustomerName: \t\t\t\t'+ CustomerName.get()+ "\n")

txtReceipt.insert(END,'===================================' "\n")

txtReceipt.insert(END,'Type Of Car: \t\t\t\t'+ var7.get()+"\n")

txtReceipt.insert(END,'Cost of Car\t\t\t\t'+ CostofCar.get()+ "\n")

txtReceipt.insert(END,'Total Mileage\t\t\t\t'+ var18.get()+ "\n")

txtReceipt.insert(END,'Trade in value'+ CarMileage.get()+ "\n")

txtReceipt.insert(END,'Cost of Modified: \t\t\t\t'+ Modified.get()+ "\n")

txtReceipt.insert(END,'Cost of Stereo: \t\t\t\t'+ Stereo.get()+ "\n")

txtReceipt.insert(END,'Cost Of Leather: \t\t\t\t'+ Leather.get()+ "\n")

txtReceipt.insert(END,'Cost of Customised: \t\t\t\t'+ Customised.get()+ "\n")

txtReceipt.insert(END,'Cost of GPS\t\t\t\t: '+ GPS.get()+ "\n")

txtReceipt.insert(END,'===================================' "\n")

txtReceipt.insert(END,'Tax:\t\t\t\t'+ Tax.get()+ "\n")

txtReceipt.insert(END,'SubTotal: \t\t\t\t'+ SubTotal.get()+ "\n")

txtReceipt.insert(END,'Total Cost: \t\t\t\t'+ Total.get()+ "\n")

txtReceipt.insert(END,'===================================' "\n")

#========================================1========================================================================

CustomerName=StringVar()

CustomerAddress=StringVar()

CustomerPostcode=StringVar()

CustomerTelephone=StringVar()

lblName = Label(bottomLeftTopL, font=('arial', 16,'bold'),text="Name", fg="black",width=15, bd=10, anchor='w')

lblName.grid(row=0, column=0)

txtName = Entry(bottomLeftTopL, font=('arial', 16,'bold'), bd=2, width=24, bg="white", justify = 'left', textvariable=CustomerName)

txtName.grid(row=0, column=1)

lblName = Label(bottomLeftTopL, font=('arial', 16,'bold'),text="Name", fg="black",width=15, bd=10, anchor='w')

lblName.grid(row=0, column=0)

txtName = Entry(bottomLeftTopL, font=('arial', 16,'bold'), bd=2, width=24, bg="white", justify = 'left', textvariable=CustomerName)

txtName.grid(row=0, column=1)

lblAddress = Label(bottomLeftTopL, font=('arial', 16,'bold'),text="Address", fg="black",width=15, bd=10, anchor='w')

lblAddress.grid(row=1, column=0)

txtAddress = Entry(bottomLeftTopL, font=('arial', 16,'bold'), bd=2, width=24, bg="white", justify = 'left', textvariable=CustomerAddress)

txtAddress.grid(row=1, column=1)

lblPostcode = Label(bottomLeftTopL, font=('arial', 16,'bold'),text="Postcode", fg="black",width=15, bd=10, anchor='w')

lblPostcode.grid(row=2, column=0)

txtPostcode = Entry(bottomLeftTopL, font=('arial', 16,'bold'), bd=2, width=24, bg="white", justify = 'left', textvariable=CustomerPostcode)

txtPostcode.grid(row=2, column=1)

lblTelephone = Label(bottomLeftTopL, font=('arial', 16,'bold'),text="Telephone", fg="black",width=15, bd=10, anchor='w')

lblTelephone.grid(row=3, column=0)

txtTelephone = Entry(bottomLeftTopL, font=('arial', 16,'bold'), bd=2, width=24, bg="white", justify = 'left', textvariable=CustomerTelephone)

txtTelephone.grid(row=3, column=1)

#========================================3========================================================================

Modified=StringVar()

Stereo=StringVar()

Leather=StringVar()

Customised=StringVar()

GPS=StringVar()

Modified.set("0")

Stereo.set("0")

Leather.set("0")

Customised.set("0")

GPS.set("0")

var1= IntVar()

var2= IntVar()

var3= IntVar()

var4= IntVar()

var5= IntVar()

lblModified = Checkbutton(bottomLeftBottomL, font=('arial', 16,'bold'),text="Modified", fg="black",width=20, bd=10, anchor='w'

,onvalue=1, offvalue=0, variable=var1)

lblModified.grid(row=0, column=0)

txtModified = Entry(bottomLeftBottomL, font=('arial', 16,'bold'), bd=2, width=14, bg="white", justify = 'left', textvariable=Modified)

txtModified.grid(row=0, column=1)

lblStereo = Checkbutton(bottomLeftBottomL, font=('arial', 16,'bold'),text="Stereo System", fg="black",width=20, bd=10, anchor='w'

,onvalue=1, offvalue=0, variable=var2)

lblStereo.grid(row=1, column=0)

txtStereo = Entry(bottomLeftBottomL, font=('arial', 16,'bold'), bd=2, width=14, bg="white", justify = 'left', textvariable=Stereo)

txtStereo.grid(row=1, column=1)

lblLeather = Checkbutton(bottomLeftBottomL, font=('arial', 16,'bold'),text="Leather Interior", fg="black",width=20, bd=10, anchor='w'

,onvalue=1, offvalue=0, variable=var3)

lblLeather.grid(row=2, column=0)

txtLeather = Entry(bottomLeftBottomL, font=('arial', 16,'bold'), bd=2, width=14, bg="white", justify = 'left', textvariable=Leather)

txtLeather.grid(row=2, column=1)

lblCustomised = Checkbutton(bottomLeftBottomL, font=('arial', 16,'bold'),text="Customised Details", fg="black",width=20, bd=10, anchor='w'

,onvalue=1, offvalue=0, variable=var4)

lblCustomised.grid(row=3, column=0)

txtCustomised = Entry(bottomLeftBottomL, font=('arial', 16,'bold'), bd=2, width=14, bg="white", justify = 'left', textvariable=Customised)

txtCustomised.grid(row=3, column=1)

lblGPS = Checkbutton(bottomLeftBottomL, font=('arial', 16,'bold'),text="Global Positioning System", fg="black",width=20, bd=10, anchor='w'

,onvalue=1, offvalue=0, variable=var5)

lblGPS.grid(row=4, column=0)

txtGPS = Entry(bottomLeftBottomL, font=('arial', 16,'bold'), bd=2, width=14, bg="white", justify = 'left', textvariable=GPS)

txtGPS.grid(row=4, column=1)

btnTotalCost=Button(bottomLeftBottomL,pady=8, bd=2, fg="black",font=('arial',16,'bold'), width=13, text="TOTAL", bg="white", command=CarCost). grid(row=5, column=0)

btnReceipt=Button(bottomLeftBottomL,pady=8, bd=2, fg="black",font=('arial', 16, 'bold'), width=13, text="RECEIPT", bg="white",command=Receipt). grid(row=5, column=1)

#========================================2========================================================================

var6= IntVar()

var7= StringVar()

var18= StringVar()

CostofCar= StringVar()

CarMileage= StringVar()

CostofCar.set("0")

CarMileage.set("0")

lblChooseaCar= Label(bottomLeftTopR, font=('arial',16, 'bold'),text="Choose a Car", fg="black", width=13,bd=14, anchor='w')

lblChooseaCar.grid(row=0,column=0)

cboChooseaCar =ttk.Combobox(bottomLeftTopR, textvariable = var7, state='readonly', font=('arial',20,'bold'), width=12)

cboChooseaCar['value']=('','Swift Desire', 'Dustor','INOVA','Fortuner')

cboChooseaCar.current(0)

cboChooseaCar.grid(row=1, column=0)

lblCostofCar= Label(bottomLeftTopR,font=('arial', 16,'bold'),text="Cost Of a Car", fg="black",width=13, bd=14, anchor='w')

lblCostofCar.grid(row=2, column=0)

txtCostofCar = Entry(bottomLeftTopR, font=('arial', 16,'bold'), bd=2, width=16, bg="white", justify = 'left', textvariable=CostofCar)

txtCostofCar.grid(row=3, column=0)

lblTradeInaCar= Label(bottomLeftTopR, font=('arial', 16,'bold'),text="Trade in a Car", fg="black",width=13, bd=14, anchor='w')

lblTradeInaCar.grid(row=0, column=1)

cboTradeInaCar =ttk.Combobox(bottomLeftTopR, textvariable = var18, state='readonly', font=('arial',20,'bold'), width=12)

cboTradeInaCar['value']=('','100-5000','5001-20000','20001-100000','100001-1000000')

cboTradeInaCar.current(0)

cboTradeInaCar.grid(row=1,column=1)

lblCarMileage= Label(bottomLeftTopR, font=('arial', 16,'bold'),text="CarMileage", fg="black",width=13, bd=14, anchor='w')

lblCarMileage.grid(row=2, column=1)

txtCarMileage = Entry(bottomLeftTopR, font=('arial', 16,'bold'), bd=2, width=16, bg="white", justify = 'left', textvariable=CarMileage)

txtCarMileage.grid(row=3, column=1)

#========================================4===========================================================================

VAT= StringVar()

Discount= StringVar()

Tax= StringVar()

SubTotal= StringVar()

Total= StringVar()

var8= IntVar()

var9= IntVar()

lblVAT = Checkbutton(bottomLeftBottomR, font=('arial', 16,'bold'),text="VAT", fg="black",width=13, bd=12, anchor='w',onvalue=1, offvalue=0, variable=var8)

lblVAT.grid(row=0, column=0)

txtVAT = Entry(bottomLeftBottomR, font=('arial', 16,'bold'), bd=2, width=17, bg="white", justify = 'left', textvariable=VAT)

txtVAT.grid(row=0, column=1)

lblDiscount = Checkbutton(bottomLeftBottomR, font=('arial', 16,'bold'),text="Discount", fg="black",width=13, bd=12, anchor='w',onvalue=1, offvalue=0, variable=var9)

lblDiscount.grid(row=1, column=0)

txtDiscount = Entry(bottomLeftBottomR, font=('arial', 16,'bold'), bd=2, width=17, bg="white", justify = 'left', textvariable=Discount)

txtDiscount.grid(row=1, column=1)

lblTax = Label(bottomLeftBottomR, font=('arial', 16,'bold'),text="Tax ", fg="black",width=13, bd=12, anchor='w')

lblTax.grid(row=2, column=0)

txtTax = Entry(bottomLeftBottomR, font=('arial', 16,'bold'), bd=2, width=17, bg="white", justify = 'left', textvariable=Tax)

txtTax.grid(row=2, column=1)

lblSubTotal = Label(bottomLeftBottomR, font=('arial', 16,'bold'),text="SubTotal", fg="black",width=13, bd=12, anchor='w')

lblSubTotal.grid(row=3, column=0)

txtSubTotal = Entry(bottomLeftBottomR, font=('arial', 16,'bold'), bd=2, width=17, bg="white", justify = 'left', textvariable=SubTotal)

txtSubTotal.grid(row=3, column=1)

lblTotal = Label(bottomLeftBottomR, font=('arial', 16,'bold'),text="Total", fg="black",width=13, bd=12, anchor='w')

lblTotal.grid(row=4, column=0)

txtTotal = Entry(bottomLeftBottomR, font=('arial', 16,'bold'), bd=2, width=17, bg="white", justify = 'left', textvariable=Total)

txtTotal.grid(row=4, column=1)

btnTotalCost=Button(bottomLeftBottomR,pady=8, bd=2, fg="black",font=('arial',16,'bold'), width=13, text="Reset", bg="white", command=Reset). grid(row=5, column=0)

btnReceipt=Button(bottomLeftBottomR,pady=8, bd=2, fg="black",font=('arial', 16, 'bold'), width=13, text="Exit", bg="white", command=iExit). grid(row=5, column=1)

#========================================4============================================================================

lblReceipt= Label(bottomRight, font=('arial', 16,'bold'),text="Receipt",bd=2, anchor='w')

lblReceipt.grid(row=0, column=0, sticky=W)

lblReceipt = Text(bottomRight, font=('arial', 11,'bold'), bd=8,width=46, height=26, bg='white')

lblReceipt.grid(row=1, column=0)

root.mainloop()

**CHAPTER = 6**

**CONCLUSION AND FUTURE SCOPE**

The Corporate Company’s Car Trading Management System application has rich user interface so that novice users can access easily. This application provides the management reports like Approval status report, to track the usage of the company’s transport facility. Our project is only a humble venture to satisfy the needs in a library. Several user friendly. Integrating database technologies in vehicle management system is considered controversial by many. The general conception of database systems is that they are highly resource demanding both with respect to memory consumption and computational overhead. Furthermore, database systems are considered too non-deterministic to be used in vehicle management systems. This is true for many general purpose database management systems. However, commercially embedded solutions exist today, with database engines as small as a few kilobytes. Furthermore, many years of research in real-time database management systems shows that these systems can be made deterministic. Vehicle management system is beneficial for the management and the public. It has a big future scope.