# Automobile Service Billing

**A**

**Project Report Submitted to**


**Mandsaur University, Mandsaur**

**Towards Partial Fulfillment for the Award of**

**Bachelor of Computer Application**

**Submitted To:**

**Mr. Hemant Bairagee**

**Asst. Prof. Computer Application Department**

**Submitted By:**

**Shiva Hari V Mohan**

**Enroll. No. 18CSA3BCA1025**

**Department of Computer Application**

**Mandsaur University, Mandsaur**

**2019-20**

**Department of Computer Application**
**Mandsaur University, Mandsaur**

## CERTIFICATE

The Dissertation entitled **"Automobile Service Billing"** being submitted by **Shiva Hari V Mohan (**Enrollment No. **18CSA3BCA1025**) has been examined by us and is hereby approved for the award of degree **Bachelor of Computer Application** for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the dissertation only for the purpose for which it has been submitted.

**Internal Examiner**
**Date:**

**External Examiner**
**Date:**

**Department of Computer Application**
**Mandsaur University, Mandsaur**

## DISSERTATION APPROVAL SHEET

The dissertation work entitled **"Automobile Service Billing"** submitted by **Shiva Hari V Mohan (**Enrollment No. **18CSA3BCA1025**) is approved as partial fulfillment for the award of the **Bachelor of Computer Application** degree by Mandsaur University, Mandsaur (M.P).

**Approved by**

**Prof. Nilesh Jain**
(HOD CA)

**Department of Computer Application**
**Mandsaur University, Mandsaur**

## DECLARATIONS

I hereby declare that the work, which is being presented in the dissertation, entitled **"Automobile Service Billing"** partial fulfillment of the requirements for the award of degree of **Bachelor of Computer Application** submitted in the **Department of Computer Application (Mandsaur University, Mandsaur)** is an authentic record of my own work carried under the guidance of **Mr. Hemant Bairagee**. I have not submitted the matter embodied in this report for award of any other degree.

**Shiva Hari V Mohan**
**(18CSA3BCA1025)**

# ACKNOWLEDGEMENT

I take the opportunity to express my cordial gratitude to **Prof. Nilesh Jain,** HOD Department of Computer Application, Mandsaur University, Mandsaur (M.P.) for the valuable guidance and inspiration throughout the dissertation work. I feel thankful for his innovative ideas, which led to successful completion of this work.

I give special thanks to **Prof. B. K. Sharma,** Assoc. Prof., Department of Computer Application, Mandsaur University, Mandsaur (M.P.) to always being willing to help find solutions to any problems I had with my work.

I would also like to thanks **Prof. Deepak Mehta** Assistant Professor, Department of Computer Application, Mandsaur University, Mandsaur (M.P.) for providing additional guidance and insight into my research work.

I express my gratitude and thanks to all the staff members of Computer Science & Engineering department for their sincere cooperation in furnishing relevant information to complete this dissertation well in time successfully.

Lastly but not least I must express my cordial thank to my parent, family members and friends who gave me the moral support without which it was impossible to complete my project work. With this note I thank everyone for the support.

**Shiva Hari V Mohan**

# ABSTRACT

The Project “Automobile Services Billing” deals with the automation of Automobile’s Workshop. This software will help operators in managing the Records pertaining to his/her customer. The product will help the user to work in a highly effective and efficient environment.

The operators have been recording the customer information in the past and even in the present through their personal manual efforts. And indeed, it consumes their considerable time and energy that could be utilized in the better productive activities. Apart from that, with increasing customer Strength, the task of managing information of each individual customer is indeed a cumbersome task.

There is a lot of reason for the introduction of this project. In the manual System, there are number of inefficiencies that a operator faces. The information retrieval is one of the foremost problems. It is very difficult to gather the overall performance reports of the customer. Large records-books have to be maintained where relevant and irrelevant information has to be stored which is very untidy and clumsy process.

On the other hand, there are many inherent problems that exist in any manual system. Usually, they lack efficiency. Less efficiency has a great impact on the productivity of any human being keeping the data up-to-date.

The automation deals with all such problems and tries to remove them in the best suitable fashion. The new system will cater to the need of the operator of any workshop so that they can manage the system efficiency.

The project “Automobile Service Billing” is developed with the objective of making the system reliable, easier, fast, and more informative.

**Keywords:** Automation, highly effective, efficient environment, reliable, fast, more informative

# CONTENTS



## TABLE OF CONTENTS

## LIST OF FIGURES

# Chapter 1

# INTRODUCTION

## 1.1 Aim of Project

Billing Services send bills to your customers and you want them paid quickly -- that's why efficiency in your billing process is key. Ultimately, the quicker your business can get invoices to customers and clients, the faster your business will be paid, which will have a positive impact on cash flow. Delays in payments can even put your business at stake.

An efficient invoicing system will automatically compile costs and quotes, and then generate invoices from this data. This greatly reduces the number of people required to produce quotes, thereby reducing the chance for human error. This in turn saves your business time, which saves your business money.

The aim of this project is to create an application that should provide service to the Operators and generate invoices.

The main objectives of the project are:

- To illustrate and analyze the basic billing system and the main functionalities that surround the billing system from a business prospective and explains how each interacts to complete the billing cycle.
- To develop a billing system emulator that can bill more quickly and accurately.
- To update customer record
- To develop a billing system that enables customer to view bill information.
- Add and maintain records of available products.
- Add and maintain customer details.
- Add and maintain description of new products.
- Add and maintain new entered category of products.
- Provides economic/financial reports to the owner monthly or weekly and yearly.
- Provides a convenient solution of billing pattern.
- Make an easy to use environment for users and customers.

## 1.2 Statements of Problem

1. **Overcharging: -** The billing system is a very sensitive part and it is faced with a lot of challenges like the overcharging which makes customers/ users complain. This problem may arise from the rating, that is rate given to each call lines and the time by the inaccuracy of the billing system.
2. **Poor customer service**: - Most of the billing systems have poor customer service thereby not given room for customer complaint and attention to their complaints therefore, a Report Generation system will be developed for the user and management of eBilling and Invoicing System. This system will have both details and summary type reports for analysis of the data calculated.
3. **Security and Simplicity:** - Customer, Products, and Billing Generation i.e. it will Automate the current manual bill generation system and maintain the searchable customer, charge database and charge invoice, maintain the data security, user rights.

## 1.3 Future Scope

1. This project will help the Operators of Workshop in fast billing.
2. This project enables Operators of Workshop to maintain a great database of all Customers visited and purchase product from store.
3. Project will enable to see report regarding product and category.
4. It is easy to maintain in future prospect.

# Chapter 2

# ANALYSIS

## 2.1 Software Process Model

A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

### Waterfall Model

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.

## 2.2 Advantages of Model

- It allows for departmentalization and managerial control.
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.

## 2.3 Disadvantages of Model

- It does not allow for much reflection or revision.
- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

## 2.4 Product Perspective

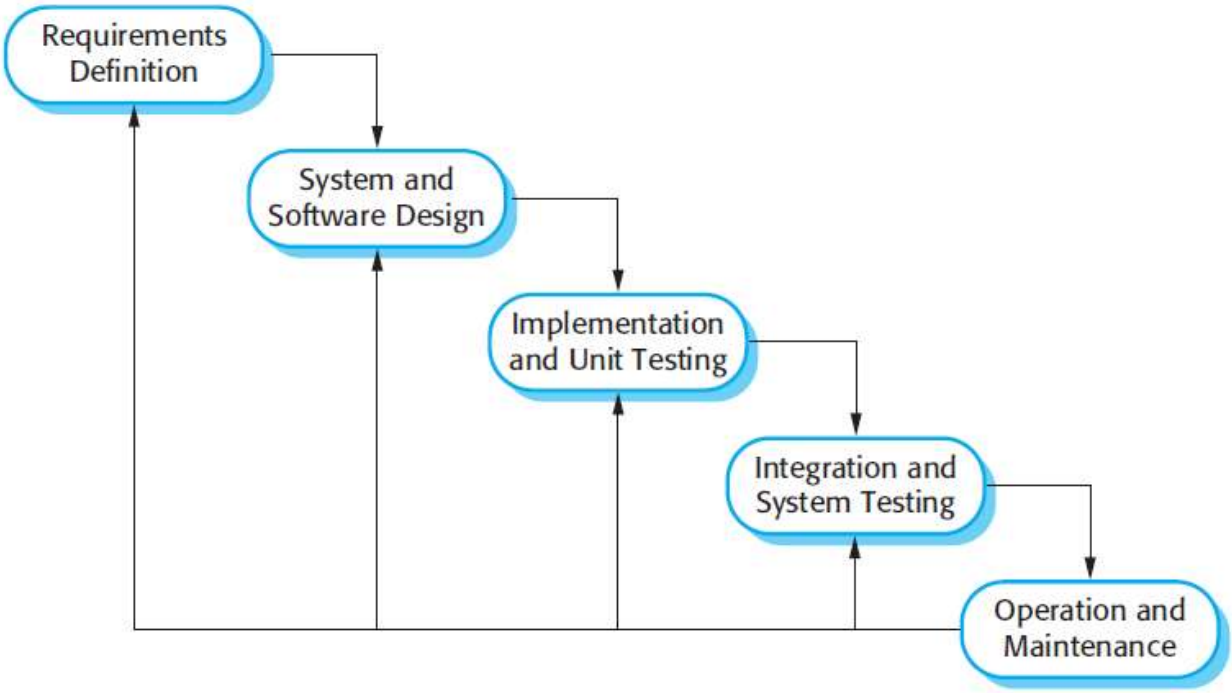


**Fig: - 2.1**

# Chapter 3

# REQUIREMENT SPECIFICATION

## 3.1 Requirement Analysis

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly. Here, we may also use the interaction with the customer to clarify points of confusion and to understand which requirements are more important than others.

## 3.2 Software Specifications

1. Web browser: Google Chrome
2. Languages Used: Python 3.7
3. PyCharm (for compiling and executing the programs)
4. PDF Reader: Google Chrome or Adobe Acrobat Reader (Used as system default PDF reader)

Operating System:

- Windows XP
- Windows 7
- Windows 8

Or any other version

## 3.3 Hardware Specifications

CPU configuration

- AMD processors 4000+ series
- RAM 1 GB DDR2 Monitor
- Any monitor

# Chapter 4
# FEASIBILITY ANALYSIS

The feasibility study is carried out to test whether the proposed system is worth being implemented. Feasibility study is a test of system proposed regarding its work ability, its impact on the organization ability to meet user needs and effective use of resources. It is usually carried out by a small number of people who are familiar with the information system techniques, understand the part of the business or organization that will be involved or effected by the project and are skilled in the system analysis and design process. The key consideration involve in the feasibility study are:

1. Technical
2. Behavioral
3. Economic

## 4.1 Technical Feasibility

Technical feasibility centers on the existing computer system (hardware, Software, etc.) and to what extent it can support the proposed system Addition. For example, if the current system is operating at 70% capacity (an Arbitrary value), then another application could overload the system or require additional hardware. If the budget is serious constrain then the project is judged not feasible.

The technologies and the environment which are used in this project are

**SOFTWARE**

Front End

1. Language used: Python 3.7.

**Operating System:**

Platform: Windows 10.

Our system requires window operating system, which is easily available.

**Hardware:**

Intel based processor-run computer system, which have keyboard and mouse as input devices. This has been decided for its case of availability and up-gradation. The various registers maintained at the different department have enough information recording, which will help in digitizing the available data.

## 4.2 Economic Feasibility

The procedure is to determine the benefits and savings that are expected from a candidate system and compare it with the costs. If a benefit outweighs costs, then the decision is made to design and implement the system.

Otherwise further alterations are made in the proposed system

1. Manpower cost
2. Hardware and software cost.

## 4.3 Behavioral Feasibility

An evaluation of the behavior of the end users, which may effect the Envelopment of the system. People are inherently resistant to change and Computers have to know to facilitate changes and computers have to known To facilitate changes. An estimate should be made of how strong a reaction The user staffs is likely to have towards the development of a computerized System. It is a common knowledge that a computer installation has something to do with turnover, transfer, retraining and changes in employee job status, therefore the introduction of a candidate system requires special effort to educate, sell and train the staff on new ways of conducting business. The personal of the user organization will be affected by the proposed system. As the aim of the system is only to satisfy the information needs, no employees will loose their position by the proposed system. In fact the proposed system will help the organization in reducing the voluminous work involved. Also the involvement of users in every stage of the project is going to increase the success factor.

The staff in not well educated for running a computerized system. They are adamant in perceiving a mechanical process of working as they have long been used to the manual entry system. This aspect needs considerable amount of attention.

Our system is also feasible for organization because it supports of the organization and its strategic plan.

# Chapter 5
# TECHNOLOGY USED

## 5.1 Selection of Platform

**Python 3.7** as a Platform. Python is a cross-platform language: A Python program written on a Macintosh computer will run on a Linux system and vice versa. Python programs can run on a Windows computer, as long as the Windows machine has the Python interpreter installed (most other operating systems come with Python pre-installed).

## 5.2 Tool Selection

**Dependencies used as tools are**: -

1. Tkinter Framework
2. Report lab Library
3. Google Chrome (as PDF reader tool)

## 5.3 Database Used

**MYSQL 8.0** as Database. MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the table. MySQL is open-source and free software under the GNU license. Oracle Company supports it.

# Chapter 6
# PLANNING AND DESIGNING

## 6.1 Planning

A Software Project is the complete methodology of programming advancement from requirement gathering to testing and support, completed by the execution procedures, in a specified period to achieve intended software product.

The Project "Automobile Service Billing" is considered under same circumstances which involve various steps: -

1. Designing Login Page.
2. Designing Bill app page on the basis of user's comfort.
3. Designing the invoice which can be printed or can be shared as PDF.
4. Writing Code in Single Class "Bill App".
5. Making Different Functions for each and every Operations.
6. Making Of only Two Forms.
7. Making Of 10 Different Tables in MySQL Data Base for Management of Data.
8. Functions Like Auto Calculation, Parts Search Operations involved in it with less human errors.

## 6.2 Designing


**Fig: - 6.1**

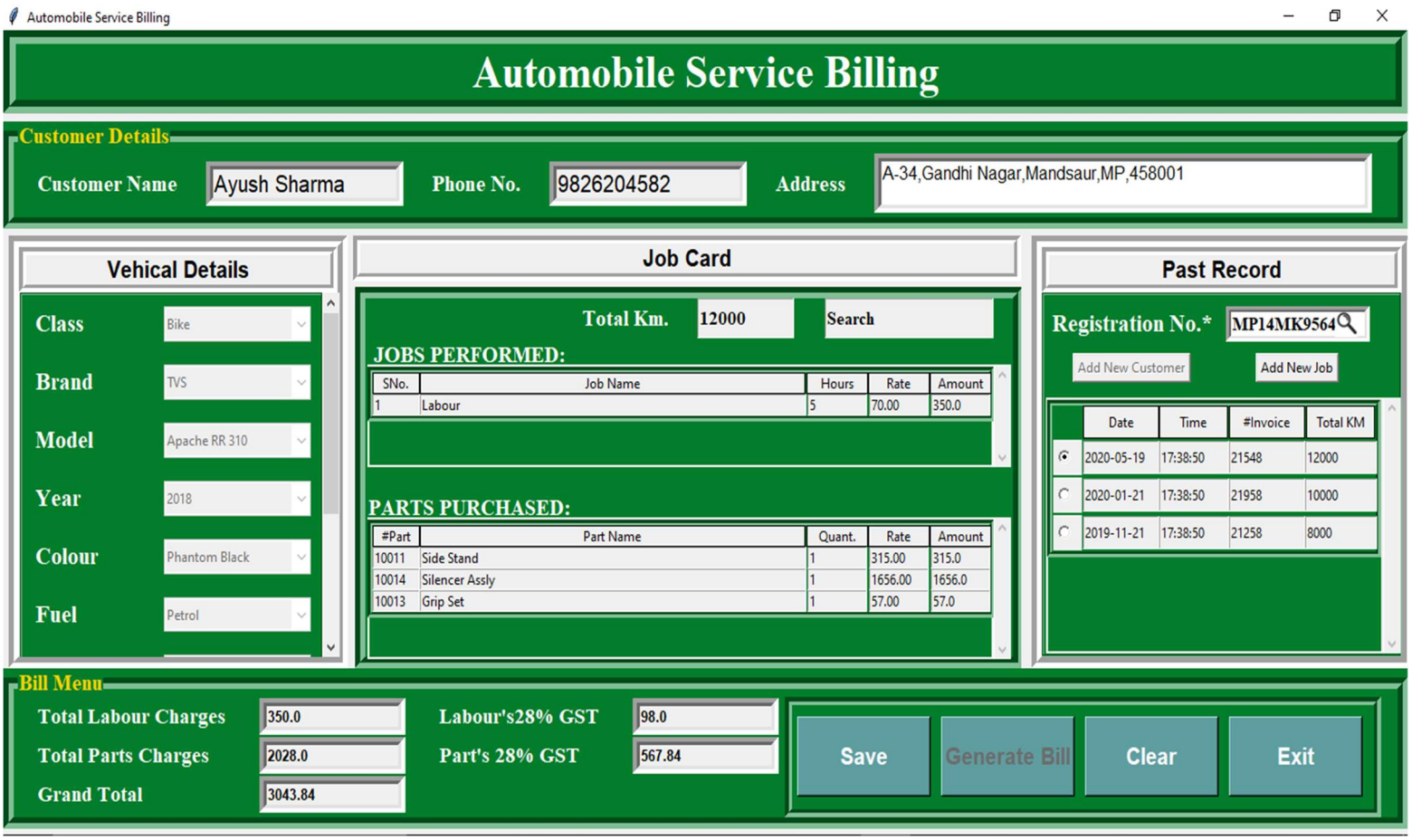
Automobile Service Billing
Automobile Service Billing
Customer Details
Customer Name
Ayush Sharma
Phone No.
9826204582
Address
A-34,Gandhi Nagar,Mandsaur,MP,458001
Vehical Details
Class
Bike
Brand
TVS
Model
Apache RR 310
Year
2018
Colour
Phantom Black
Fuel
Petrol
Job Card
Total Km.
12000
Search
JOBS PERFORMED:
SNo. Job Name Hours Rate Amount
1 Labour 5 70.00 350.0
PARTS PURCHASED:
#Part Part Name Quant. Rate Amount
10011 Side Stand 1 315.00 315.0
10014 Silencer Assly 1 1656.00 1656.0
10013 Grip Set 1 57.00 57.0
Past Record
Registration No.*
MP14MK9564
Add New Customer
Add New Job
Date Time #Invoice Total KM
2020-05-19 17:38:50 21548 12000
2020-01-21 17:38:50 21958 10000
2019-11-21 17:38:50 21258 8000
Bill Menu
Total Labour Charges
350.0
Total Parts Charges
2028.0
Grand Total
3043.84
Labour's28% GST
98.0
Part's 28% GST
567.84
Save
Generate Bill
Clear
Exit

**Fig: - 6.2**

Shiva Auto Services

The Art of Performance

# INVOICE

| DATE | 15 May 2020 | INVOICE # | INV-00-12345 |
|---|---|---|---|

| CUSTOMER INFO | | VEHICLE INFO | |
|---|---|---|---|
| NAME: | Ayush Sharma | BRAND | TVS |
| | | MODEL | Apache RR 310 |
| ADDRESS: | A-34 ,Gandhi Nagar janta colony mandsaur Mandhya- | YEAR | 2018 |
| | Pradesh Mandsaur, MP, 458001 | COLOR | Phantom Black |
| | | CC | 312.20 |
| PHONE NO: | 9826204582 | REG# | MP14MK9564 |
| | | KM | 12000 |

| Job Performed | Hours | Rate | Amount |
|---|---|---|---|
| Labour | 2 | 75.00 | 150.00 |
| | | | - |
| | | | - |
| | | | - |
| | | | - |
| | | | - |
| | | | - |
| | | | - |

| | | |
|---|---|---|
| SUBTOTAL | $ | 150.00 |
| GST | 28% | 42.00 |

| PART # | PART NAME | QTY | UNIT PRICE | AMOUNT |
|---|---|---|---|---|
| 12345 | Part Name 1 | 1 | 34.00 | 34.00 |
| 67890 | Part Name 2 | 2 | 17.55 | 35.10 |
| | | | | - |
| | | | | - |
| | | | | - |
| | | | | - |
| | | | | - |
| | | | | - |

| | | |
|---|---|---|
| SUBTOTAL | $ | 69.10 |
| GST | 28% | 6.50 |

COMMENTS

Please include the invoice number as reference when paying online or by check

| | | |
|---|---|---|
| TOTAL LABOUR | $ | 150.00 |
| TOTAL PARTS | $ | 69.10 |
| GST | $ | 13.50 |
| TOTAL | $ | 232.60 |

Thank you for your business!

Make all checks payable to
Shiva Auto Services

Should you have any enquiries concerning this invoice, please contact Mr. Shiva on 9562458585

127, Juhu Street, Indira Nagar, Housing Colony, Mandsaur, M.P, India, 458001

Tel: +91 9654895656 Fax: 2-555-898-2635 Email: info@shivaautoservices.com Web: www.shivaautoservices.com

**Fig: - 6.3**

## 6.3 Entity Relationship Diagram

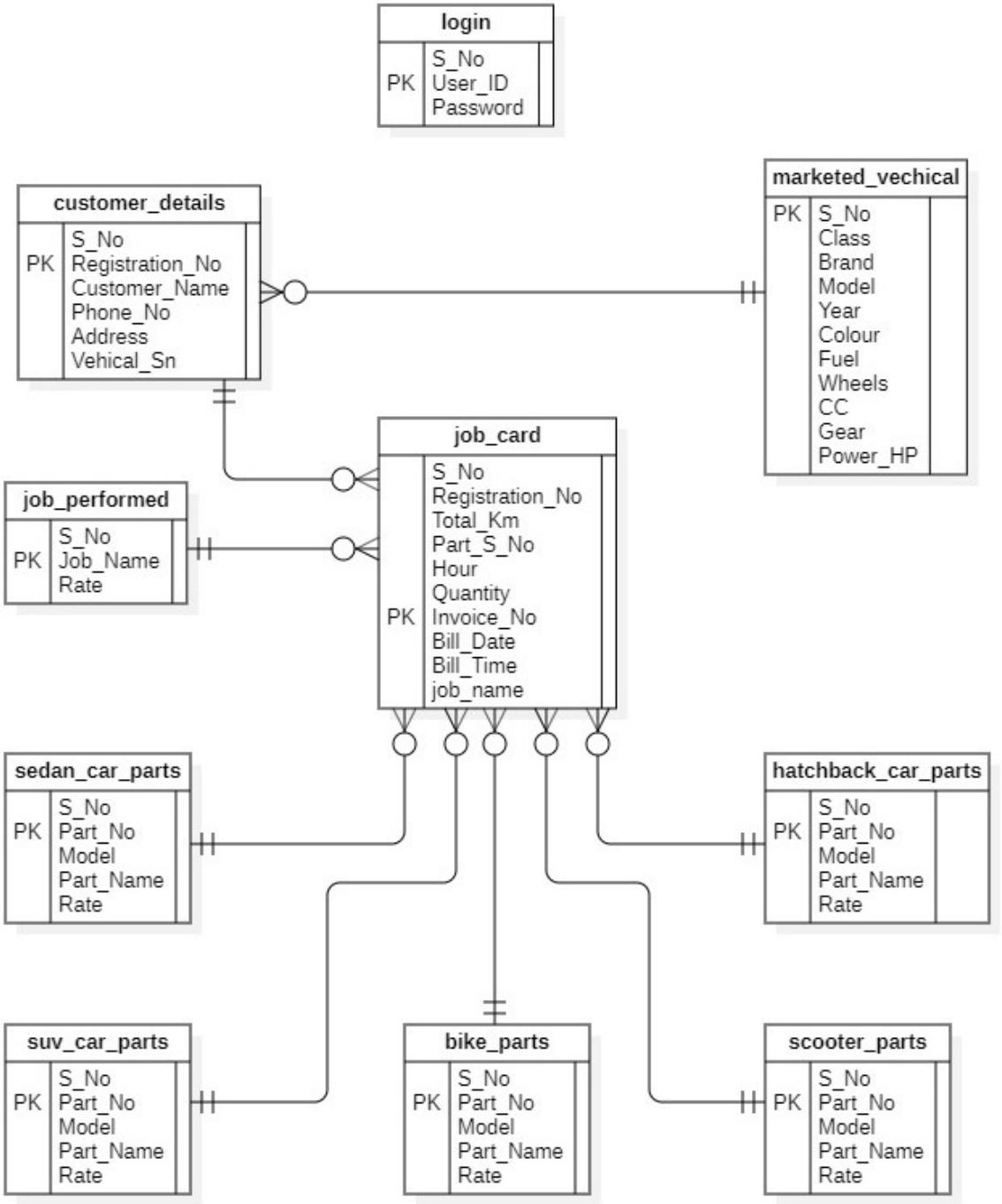


**Fig: - 6.4**

## 6.4 Data Flow Diagram

### 6.4.1DFD Level 0

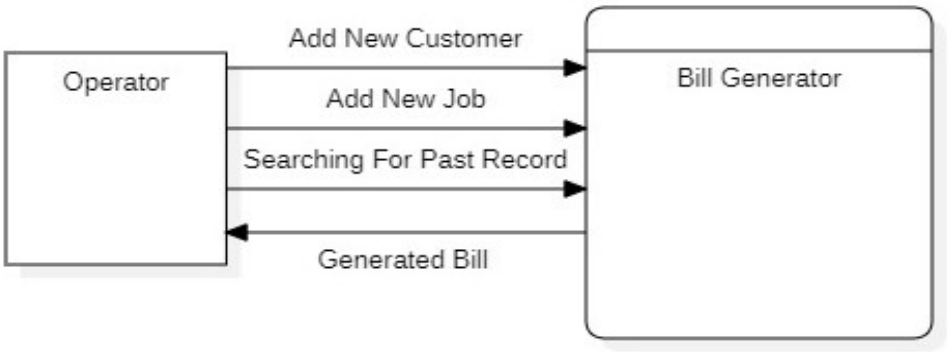


**Fig: - 6.5**

### 6.4.2DFD Level 1

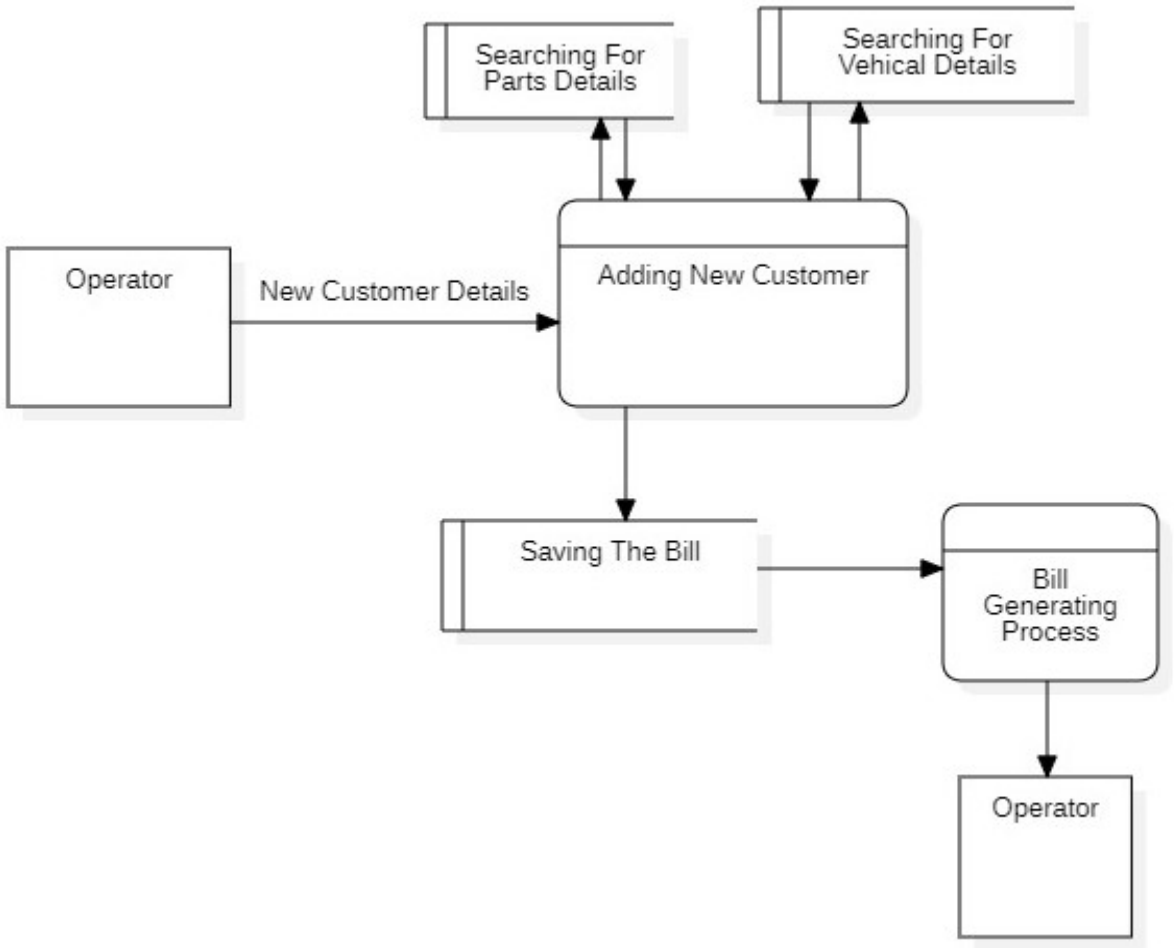


**Fig: -6.6**

## 6.4.3DFD Level 2

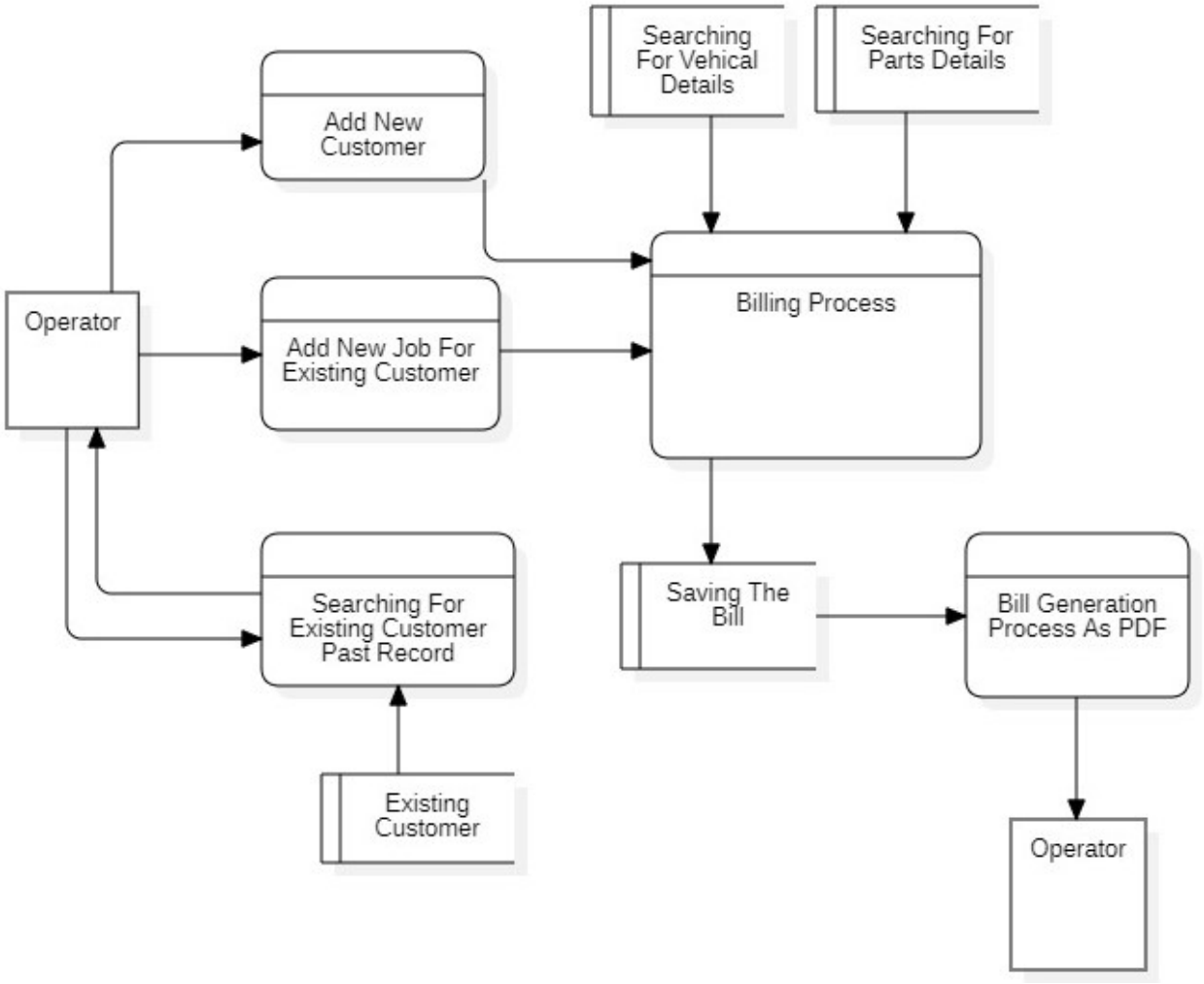


**Fig: -6.7**

## 6.5 Use Case Diagram

### 6.5.1Use Case for Operator

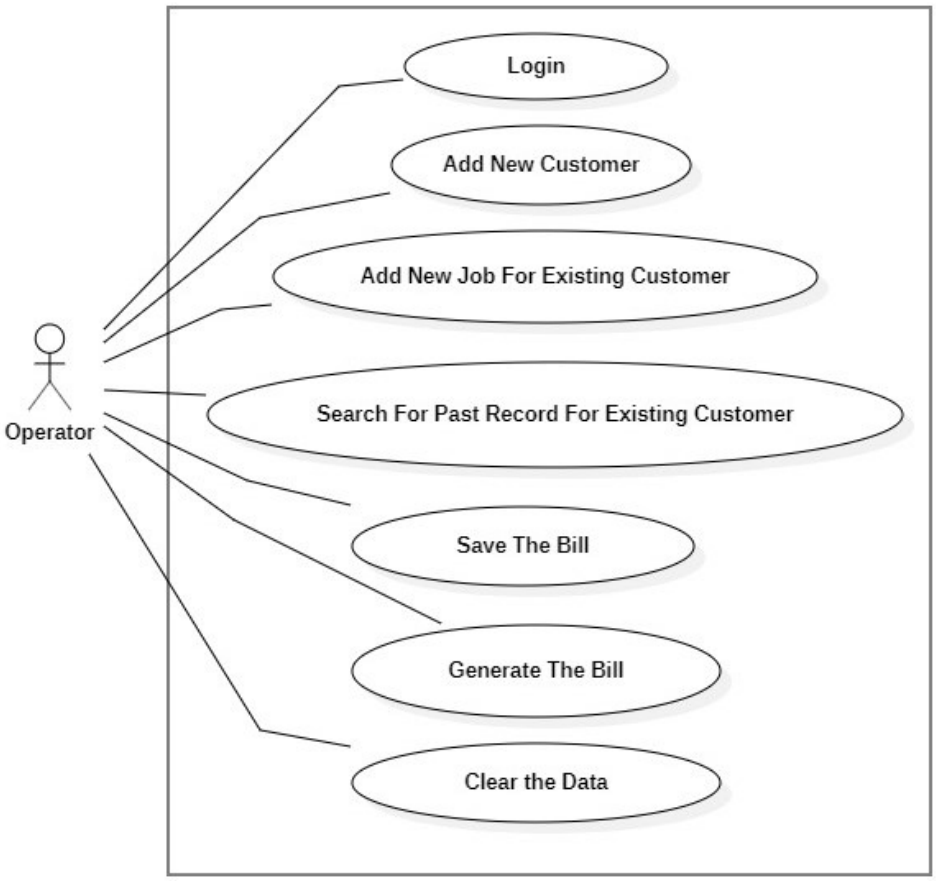


**Fig: - 6.8**

# Chapter 7

# IMPLEMENTATION

## Login Page Code: -

```
from tkinter import *
import mysql.connector
import bill

root1 = Tk()
root1.resizable(False, False)
window_height = 350
window_width = 450
screen_width = root1.winfo_screenwidth()
screen_height = root1.winfo_screenheight()

x_cordinate = int((screen_width/2) - (window_width/2))
y_cordinate = int((screen_height/2) - (window_height/2))

root1.geometry("{}x{}+{}+{}".format(window_width, window_height, x_cordinate, y_cordinate))

def loginfn():
    conn = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
    cursor = conn.cursor()
    global label3
    global label4
    label3.destroy()
    label4.destroy()
    label3 = Label(frame1, text="Try Again!, Incorrect user ID and Password", bg=bg_color, fg="white")
    label4 = Label(frame1, text="Try Again!, Incorrect Password", bg=bg_color, fg="white")
    a = user_id_txt.get()
    b = password_txt.get()
    sql ="SELECT * from login where User_ID = %s;"
    cursor.execute(sql,(a,))
```

```
    result =cursor.fetchall()
    print(result)
    sql2 ="Select SHA1(%s) ;"
    cursor.execute(sql2,(b,))
    result2 =cursor.fetchall()
    if result == []:

        label3.place(x=110, y=250)
    else:
        if result[0][1] == a and result[0][2] != result2[0][0]:
            label4.place(x=130, y=250)
        elif result[0][1] == a and result[0][2]== result2[0][0]:

            root1.destroy()
            bill.root = Tk()
            obj = bill.Bill_App(bill.root)
            bill.root.mainloop()

    conn.close()
    cursor.close()


bg_color = "#027E2C"
frame =LabelFrame(root1,bd=10,relief=GROOVE,font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
frame.place(x=0,y=0,width=450,height=350)
title=Label(frame,text="SIGN IN",bd=8,relief=GROOVE,bg=bg_color,fg="white",font=("times new roman",20,"bold"),pady=2)
title.pack(fill=X)
frame1 =LabelFrame(frame,font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
label0=Label(frame1,text="    ",bg=bg_color).grid(row=0,column=0,pady=10)
user_id=Label(frame1,text="      USER ID",bg=bg_color,fg="white",font=("times new roman",15,"bold"))
user_id.grid(row=1,column=0,padx=10,pady=10)
user_id_txt=Entry(frame1,width=16,font="arial 15",bd=7,relief=SUNKEN)
user_id_txt.grid(row=1,column=1,padx=10,pady=10)
bullet = "\u2022"
password=Label(frame1,text="      PASSWORD",bg=bg_color,fg="white",font=("times new roman",15,"bold"))
password.grid(row=2,column=0,padx=10,pady=10)
password_txt=Entry(frame1,width=16,show=bullet,font="arial 15",bd=7,relief=SUNKEN)
password_txt.grid(row=2,column=1,padx=10,pady=10)

sign_in_btn =Button(frame1,text="SIGN IN",fg="white",font=("times new roman",15,"bold"),command=loginfn,bg=bg_color)
sign_in_btn.place(x=165,y=200)


label3 = Label(frame1)

label4 = Label(frame1)



root1.mainloop()
```

# Bill App Page Code: -

```
from tkinter import *
from tkinter.ttk import Combobox
# noinspection PyUnresolvedReferences
from tkinter.ttk import Notebook
# noinspection PyUnresolvedReferences
from PIL import Image, ImageTk
import mysql.connector
from mysql.connector import errorcode
from mysql.connector import errors
import re
from tkinter import messagebox
import login_page


class Bill_App:
    def __init__(self,root):
        self.root = root
        w, h = self.root.winfo_screenwidth(), self.root.winfo_screenheight()
        self.root.geometry("%dx%d+0+0" % (w, h))
        #self.root.geometry("1366x768+0+0")
        self.root.title("Automobile Service Billing")
        bg_color = "#027E2C"
        title=Label(self.root,text="Automobile Service Billing",bd=12,relief=GROOVE,bg=bg_color,fg="white",font_=("times new roman",30,"bold"),pady=2).pack(fill=X)




#varlable name=========
        #====================Customer Details================
        self.customer=StringVar()
        self.Phone =StringVar()

        #====================Vehical Details================
        self.KM =StringVar()
        self.Class=StringVar()
        self.Brand=StringVar()
        self.model=StringVar()
        self.year=StringVar()
        self.colour=StringVar()
        self.fuel=StringVar()
        self.wheels=StringVar()
        self.cc=StringVar()
        self.gear=StringVar()
        self.power=StringVar()



        #======================Job Card====================


        #====================Registration Search==============
        self.regsearch=StringVar()
        #====================Menu ============================
        self.Totalpart =StringVar()
        self.TotalLabour =StringVar()
```

```
        self.TotalLabour =StringVar()
        self.GstLabour =StringVar()
        self.GstParts = StringVar()
        self.GrandTotal = StringVar()


        #============Customer Detail Frame
        F1 = LabelFrame(self.root,bd=10,relief=GROOVE,text="Customer Details",font=("times new roman",15,"bold"),fg="gold",bg=bg_color)
        F1.place(x=0,y=80,relwidth=1)

        cname_lbl=Label(F1,text="Customer Name",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=0,padx=20,pady=5)
        self.cname_txt=Entry(F1,width=16,state="readonly",textvariable=self.customer,font="arial 15",bd=7,relief=SUNKEN)
        self.cname_txt.grid(row=0,column=1,padx=5,pady=10)

        cphone_lbl=Label(F1,text="Phone No.",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=2,padx=20,pady=5)
        self.cphone_txt=Entry(F1,width=16,state="readonly",textvariable=self.Phone,font="arial 15",bd=7,relief=SUNKEN)
        self.cphone_txt.grid(row=0,column=3,padx=5,pady=10)

        caddress_lbl=Label(F1,text="Address",bg=bg_color,fg="white",font=("times new roman",15,"bold")).grid(row=0,column=4,padx=20,pady=5)
        self.caddress_txt=Text(F1,width=52,font="arial 12",height=2,bd=7,relief=SUNKEN)
        self.caddress_txt.grid(row=0,column=5,padx=5)
        self.caddress_txt.config(state=DISABLED)


        #bill_btn=Button(F1,text="Search",width=10,bd=7,font="arial 12 bold").grid(row=0,column=6,padx=10,pady=10)


#========Vehical Details==
        F2 = Frame(self.root,bd=10, relief=GROOVE)
        bill_title1 = Label(F2, text="Vehical Details", font="arial 15 bold", bd=7, relief=GROOVE).pack(fill=X)
        F2.place(x=5, y=180, width=330, height=380)

        FF2 = Frame(F2, relief=GROOVE, bg=bg_color)
        # FF2.place(x=5, y=180, width=325, height=380)
        canvas = Canvas(FF2, bg=bg_color, width=290)
        scrollbar = Scrollbar(FF2, orient="vertical", command=canvas.yview)
        scrollable_frame = Frame(canvas, bg=bg_color, bd=2, relief=GROOVE)

        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(
                scrollregion=canvas.bbox("all")
            )
        )

        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)

        v=[]
        vclass_lbl = Label(scrollable_frame, text="Class", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=0, column=0, padx=10, pady=10, sticky="w")
        self.vclass_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.Class,values=v)
        self.vclass_txt.set("Select")
        self.vclass_txt.grid(row=0, column=1, padx=10, pady=10,ipady=5)
        self.vclass_txt.bind('<<ComboboxSelected>>',self.classvalues)
```

```
        vbrand_lbl = Label(scrollable_frame, text="Brand", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=1, column=0, padx=10, pady=10, sticky="w")
        self.vbrand_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.Brand,values=v)
        self.vbrand_txt.set("Select")
        self.vbrand_txt.grid(row=1, column=1, padx=10, pady=10,ipady=5)
        self.vbrand_txt.bind('<<ComboboxSelected>>',self.brandvalues)

        vmodel_lbl = Label(scrollable_frame, text="Model", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=2, column=0, padx=10, pady=10, sticky="w")
        self.vmodel_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.model,values=v)
        self.vmodel_txt.set("Select")
        self.vmodel_txt.grid(row=2, column=1, padx=10, pady=10,ipady=5)
        self.vmodel_txt.bind('<<ComboboxSelected>>', self.modelvalues)

        vyear_lbl = Label(scrollable_frame, text="Year", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=3, column=0, padx=10, pady=10, sticky="w")
        self.vyear_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.year,values=v)
        self.vyear_txt.set("Select")
        self.vyear_txt.grid(row=3, column=1, padx=10, pady=10,ipady=5)
        self.vyear_txt.bind('<<ComboboxSelected>>', self.yearvalues)

        vcolour_lbl = Label(scrollable_frame, text="Colour", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=4, column=0, padx=10, pady=10, sticky="w")
        self.vcolour_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.colour,values=v)
        self.vcolour_txt.set("Select")
        self.vcolour_txt.grid(row=4, column=1, padx=10, pady=10,ipady=5)
        self.vcolour_txt.bind('<<ComboboxSelected>>', self.colourvalues)

        vfuel_lbl = Label(scrollable_frame, text="Fuel", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=5, column=0, padx=10, pady=10, sticky="w")
        self.vfuel_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.fuel,values=v)
        self.vfuel_txt.set("Select")
        self.vfuel_txt.grid(row=5, column=1, padx=10, pady=10,ipady=5)
        self.vfuel_txt.bind('<<ComboboxSelected>>', self.fuelvalues)

        vwheels_lbl = Label(scrollable_frame, text="Wheels", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=6, column=0, padx=10, pady=10, sticky="w")
        self.vwheels_txt = Combobox(scrollable_frame, state="disabled", textvariable=self.wheels, values=v)
        self.vwheels_txt.set("Select")
        self.vwheels_txt.grid(row=6, column=1, padx=10, pady=10,ipady=5)
        self.vwheels_txt.bind('<<ComboboxSelected>>', self.wheelsvalues)

        vcc_lbl = Label(scrollable_frame, text="CC", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=7, column=0, padx=10, pady=10, sticky="w")
        self.vcc_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.cc,values=v)
        self.vcc_txt.set("Select")
        self.vcc_txt.grid(row=7, column=1, padx=10, pady=10,ipady=5)
        self.vcc_txt.bind('<<ComboboxSelected>>', self.ccvalues)

        vgear_lbl = Label(scrollable_frame, text="Gear", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=8, column=0, padx=10, pady=10, sticky="w")
        self.vgear_txt = Combobox(scrollable_frame, state="disabled", textvariable=self.gear, values=v)
        self.vgear_txt.set("Select")
        self.vgear_txt.grid(row=8, column=1, padx=10, pady=10,ipady=5)
        self.vgear_txt.bind('<<ComboboxSelected>>', self.gearvalues)

        vpower_lbl = Label(scrollable_frame, text="Power(HP)", font=("times new roman", 16, "bold"), bg=bg_color,fg="white").grid(row=9, column=0, padx=10, pady=10, sticky="w")
        self.vpower_txt = Combobox(scrollable_frame,state="disabled",textvariable=self.power,values=v)
        self.vpower_txt.set("Select")
        self.vpower_txt.grid(row=9, column=1, padx=10, pady=10,ipady=5)
```

```
        FF2.place(x=1, y=40, height=320)
        canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")
        scrollable_frame.place()



#========Job Card==
        F3 = Frame(self.root)
        F3.place(x=340, y=180, width=650, height=380)
        bill_title3 = Label(F3, text="Job Card", font="arial 15 bold", bd=7, relief=GROOVE).pack(fill=X)
        self.FF32 = LabelFrame(F3, bd=10, relief=GROOVE, font=("times new roman", 15, "bold"), fg="gold", bg=bg_color)
        self.FF32.place(x=2, y=45, width=650, height=335)



        lable2 = Label(self.FF32,text="", bg=bg_color).grid(row=0, column=0)
        vkm_lbl = Label(self.FF32, text=" Total Km.     ", font=("times new roman", 15, "bold"), bg=bg_color,fg="white").grid(row=0, column=1, sticky="w")
        self.vkm_txt = Entry(self.FF32, textvariable=self.KM,state="readonly",font=("times new roman", 13, "bold"),width=10)
        self.vkm_txt.grid(row=0, column=2, ipady=6)

        lable2 = Label(self.FF32, text="        ", bg=bg_color).grid(row=0, column=3)

        self.test_list = []
        self.entry = Entry(self.FF32,font=("times new roman", 12, "bold"),width=20)
        self.entry.grid(row=0, column=4,ipady=6)
        self.entry.insert(0,"Search")
        self.entry.config(state="readonly")
        self.entry.bind('<Button-1>', self.listbox_show)
        self.entry.bind('<KeyRelease>', self.on_keyrelease)
        self.listbox = Listbox()
        self.cells7 = {}
        self.cells8 = {}
        self.cells9 = {(0, 0): Entry()}
        self.cells10 = {(0, 0): Entry()}

        lable1 = Label(self.FF32, text="JOBS PERFORMED:", font=("times new roman", 15, "bold"), fg="white",bg=bg_color).grid(row=1, column=0,ipady=0)



        lable2 = Label(self.FF32, bg=bg_color).grid(row=3, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=4, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=5, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=6, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=7, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=8, column=0)
        lable2 = Label(self.FF32, bg=bg_color).grid(row=9, column=0)

        FF3 = Frame(self.FF32, relief=GROOVE, bg="white")
        # FF3.place(x=2, y=60, width=626, height=100)
        canvas1 = Canvas(FF3, bg=bg_color, width=200)
        scrollbar1 = Scrollbar(FF3, orient="vertical", command=canvas1.yview)
        self.scrollable_frame1 = Frame(canvas1, bg=bg_color, bd=5, relief=GROOVE, width=600)

```

```
216         self.scrollable_frame1.bind(
217             "<Configure>",
218             lambda e: canvas1.configure(
219                 scrollregion=canvas1.bbox("all")
220             )
221         )
222 
223         canvas1.create_window((0, 0), window=self.scrollable_frame1, anchor="nw", width=610)
224         canvas1.configure(yscrollcommand=scrollbar1.set)
225 
226         self.cells6 = {(0,0):Entry()}
227         a1 = Label(self.scrollable_frame1, text="SNo.",bd=1,relief="solid", width=6).grid(row=0,column=0)
228         a2 = Label(self.scrollable_frame1, text="Job Name",bd=1,relief="solid", width=53).grid(row=0, column=1)
229         a3 = Label(self.scrollable_frame1, text="Hours",bd=1,relief="solid", width=8).grid(row=0, column=2)
230         a4 = Label(self.scrollable_frame1, text="Rate",bd=1,relief="solid", width=8).grid(row=0, column=3)
231         a5 = Label(self.scrollable_frame1, text="Amount",bd=1,relief="solid", width=8).grid(row=0, column=4)
232 
233 
234 
235 
236 
237 
238 
239 
240         FF3.place(x=2, y=58, width=626, height=90)
241         canvas1.pack(side="left", fill="both", expand=True)
242         scrollbar1.pack(side="right", fill="y")
243         self.scrollable_frame1.place()
244 
245 
246 
247 
248 
249 
250         lable3 = Label(self.FF32, text="PARTS PURCHASED:", font=("times new roman", 15, "bold"), fg="white",bg=bg_color).grid(row=8, column=0)
251 
252         FFF3 = Frame(self.FF32, relief=GROOVE, bg="white")
253         # FFF3.place(x=2, y=60, width=626, height=100)
254         canvas2 = Canvas(FFF3, bg=bg_color, width=200)
255         scrollbar2 = Scrollbar(FFF3, orient="vertical", command=canvas2.yview)
256         self.scrollable_frame2 = Frame(canvas2, bg=bg_color, bd=5, relief=GROOVE, width=600)
257 
258         self.scrollable_frame2.bind(
259             "<Configure>",
260             lambda e: canvas2.configure(
261                 scrollregion=canvas2.bbox("all")
262             )
263         )
264 
265         canvas2.create_window((0, 0), window=self.scrollable_frame2, anchor="nw", width=610)
266         canvas2.configure(yscrollcommand=scrollbar2.set)
267 
268         self.cells4 = {(0,0):Entry()}
```

```
        self.cells4 = {(0,0):Entry()}
        b1 = Label(self.scrollable_frame2, text="#Part", bd=1, relief="solid", width=6).grid(row=0, column=0)
        b2 = Label(self.scrollable_frame2, text="Part Name", bd=1, relief="solid", width=53).grid(row=0, column=1)
        b3 = Label(self.scrollable_frame2, text="Quant.", bd=1, relief="solid", width=8).grid(row=0, column=2)
        b4 = Label(self.scrollable_frame2, text="Rate", bd=1, relief="solid", width=8).grid(row=0, column=3)
        b5 = Label(self.scrollable_frame2, text="Amount", bd=1, relief="solid", width=8).grid(row=0, column=4)









        FFF3.place(x=2, y=192, width=626, height=125)
        canvas2.pack(side="left", fill="both", expand=True)
        scrollbar2.pack(side="right", fill="y")
        self.scrollable_frame2.place()



#Past Record=================

        F5 = Frame(self.root, bd=10, relief=GROOVE)
        F5.place(x=1000, y=180, width=370, height=380)
        bill_title=Label(F5,text="Past Record",font="arial 15 bold",bd=7,relief=GROOVE).pack(fill=X)
        FF5=LabelFrame(F5,relief=GROOVE, font=("times new roman", 15, "bold"),fg="gold", bg=bg_color)
        FF5.pack(fill=BOTH,expand=1)

        Reg22_lbl = Label(FF5, text="Registration No.*", font=("times new roman", 16, "bold"), bg=bg_color, fg="white").grid(row=0, column=0, padx=5
        self.Reg22_txt = Entry(FF5, width=16,textvariable=self.regsearch,font=("times new roman", 12, "bold"), bd=5,relief=SUNKEN)
        self.Reg22_txt.grid(row=0, column=1, padx=5, pady=10)


        self.iconPath = 'search5.jpg'
        self.icon = ImageTk.PhotoImage(Image.open(self.iconPath))
        self.icon_btn = Button(self.root,command=self.searchers, image=self.icon, width=20, height=20,border=0)
        self.icon_btn.place(x=1295,y=247)



        self.newcustomer =Button(FF5,text="Add New Customer",command=self.newcustomerf,state="disabled")
        self.newcustomer.grid(row=1,column=0)
        self.newjob =Button(FF5,text="Add New Job",command=self.newjobf,state="disabled")
        self.newjob.grid(row=1,column=1)









        FFF5 = Frame(FF5, relief=GROOVE, bg="white")
        # FF3.place(x=2, y=60, width=626, height=100)
        canvas4 = Canvas(FFF5, bg=bg_color, width=200)
```

```
        canvas4 = Canvas(FFF5, bg=bg_color, width=200)
        scrollbar4 = Scrollbar(FFF5, orient="vertical", command=canvas4.yview)
        self.scrollable_frame4 = Frame(canvas4, bg=bg_color, bd=5, relief=GROOVE)

        self.scrollable_frame4.bind(
            "<Configure>",
            lambda e: canvas4.configure(
                scrollregion=canvas4.bbox("all")
            )
        )

        canvas4.create_window((0, 0), window=self.scrollable_frame4, anchor="nw", width=325)
        canvas4.configure(yscrollcommand=scrollbar4.set)

        self.cells2 = {(0,0):Entry()}
        c1 = Label(self.scrollable_frame4,text="",bd=1,relief="solid",width=4,border=0,bg=bg_color)
        c1.grid(row=0,column=0,ipady=5)
        c2= Label(self.scrollable_frame4, text="Date",bd=1,relief="solid", width=10).grid(row=0, column=1, ipady=5)
        c3= Label(self.scrollable_frame4, text="Time",bd=1,relief="solid", width=9).grid(row=0, column=2, ipady=5)
        c4= Label(self.scrollable_frame4, text="#Invoice",bd=1,relief="solid", width=10).grid(row=0, column=3, ipady=5)
        c6= Label(self.scrollable_frame4, text="Total KM",bd=1,relief="solid", width=9).grid(row=0, column=4, ipady=5)
```

```
        FFF5.place(x=2, y=90, width=345, height=225)
        canvas4.pack(side="left", fill="both", expand=True)
        scrollbar4.pack(side="right", fill="y")
        self.scrollable_frame4.place()

    #Menu Button Frame=========

        F6 = LabelFrame(self.root, bd=10, relief=GROOVE, text="Bill Menu", font=("times new roman", 15, "bold"),fg="gold", bg=bg_color)
        F6.place(x=0, y=560, relwidth=1, height=140)

        m1_lbl=Label(F6,text="Total Labour Charges", font=("times new roman", 14, "bold"),bg=bg_color,fg="white").grid(row=0,column=0,padx=20,pady=1,sticky="w")
        m1_txt=Entry(F6,width=18,font="arial 10 bold",state="readonly",textvariable=self.TotalLabour,bd=7,relief=SUNKEN).grid(row=0,column=1,padx=10,pady=1)

        m2_lbl = Label(F6, text="Total Parts Charges", font=("times new roman", 14, "bold"), bg=bg_color,fg="white").grid(row=1, column=0, padx=20, pady=1, sticky="w")
        m2_txt = Entry(F6, width=18, font="arial 10 bold",state="readonly",textvariable=self.Totalpart, bd=7, relief=SUNKEN).grid(row=1, column=1, padx=10, pady=1)

        m3_lbl = Label(F6, text="Grand Total ", font=("times new roman", 14, "bold"), bg=bg_color,fg="white").grid(row=2, column=0, padx=20, pady=1, sticky="w")
        m3_txt = Entry(F6, width=18, font="arial 10 bold",state="readonly",textvariable=self.GrandTotal, bd=7, relief=SUNKEN).grid(row=2, column=1, padx=10, pady=1)



        c1_lbl = Label(F6, text="Labour's28% GST", font=("times new roman", 14, "bold"), bg=bg_color,fg="white").grid(row=0, column=2, padx=20, pady=1, sticky="w")
        c1_txt = Entry(F6, width=18, font="arial 10 bold",state="readonly",textvariable=self.GstLabour, bd=7, relief=SUNKEN).grid(row=0, column=3, padx=10, pady=1)

        c2_lbl = Label(F6, text="Part's 28% GST", font=("times new roman", 14, "bold"), bg=bg_color,fg="white").grid(row=1, column=2, padx=20, pady=1, sticky="w")
        c2_txt = Entry(F6, width=18, font="arial 10 bold",state="readonly",textvariable=self.GstParts, bd=7, relief=SUNKEN).grid(row=1, column=3, padx=10, pady=1)
```

```
        self.invoice_no1 =""

        btn_F=Frame(F6,bd=7,bg=bg_color,relief=GROOVE,pady=5)
        btn_F.place(x=750,width=580,height=105)

        save_btn=Button(btn_F,text="Save",bg="cadetblue",state="normal",fg="white",command=self.savebutton,pady=15,width=10,font="arial 15 bold",bd=2)
        save_btn.grid(row=0,column=0,padx=5,pady=5)
        self.Generate_btn = Button(btn_F, text="Generate Bill",state="disabled",command=self.generatebill, bg="cadetblue", fg="white", pady=15, width=10, font="arial 15 bold",bd=2
        self.Generate_btn.grid(row=0, column=1, padx=5, pady=5)
        Clear_btn = Button(btn_F, text="Clear",command=self.clearall,bg="cadetblue", fg="white", pady=15, width=10, font="arial 15 bold",bd=2)
        Clear_btn.grid(row=0, column=2, padx=5, pady=5)
        Exit_btn = Button(btn_F, text="Exit", bg="cadetblue", fg="white",command=self.exitbutton, pady=15, width=10, font="arial 15 bold",bd=2)
        Exit_btn.grid(row=0, column=3, padx=5, pady=5)

    def searchers(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)

            regg=self.regsearch.get()
            rex = re.compile("^[A-Z]{2}[0-9]{2}[A-Z]{2}[0-9]{4}$")
            if rex.match(regg):
                print("Correct format")
                sql = "SELECT * FROM customer_details where Registration_No = %s ;"
                gerr = (regg,)
                cursor.execute(sql, gerr)
                myresult = cursor.fetchall()
                Regsql=""
                vehsql=""

                for x in myresult:
                    Regsql =x[1]
                    self.customer.set(x[2])
                    self.Phone.set(x[3])
                    self.caddress_txt.config(state=NORMAL)
                    self.caddress_txt.delete(1.0, END)
                    self.caddress_txt.insert(END,x[4])
                    self.caddress_txt.config(state=DISABLED)
                    vehsql=x[5]
                if self.regsearch.get() != Regsql:
                    self.customer.set("No Record Found")
                    self.Phone.set("")
                    self.caddress_txt.config(state=NORMAL)
                    self.caddress_txt.delete(1.0,END)
                    self.caddress_txt.config(state=DISABLED)
                    self.newcustomer.config(state="normal")
```

```
460
461         if self.regsearch.get() == Regsql:
462             sql1 = "SELECT * FROM marketed_vehical where S_No = %s ;"
463             gerr1 = (vehsql,)
464             cursor.execute(sql1, gerr1)
465             myresult1 = cursor.fetchall()
466             self.newjob.config(state="normal")
467             self.newcustomer.config(state="disabled")
468             for y in myresult1:
469                 self.Class.set(y[1])
470                 self.Brand.set(y[2])
471                 self.model.set(y[3])
472                 self.year.set(y[4])
473                 self.colour.set(y[5])
474                 self.fuel.set(y[6])
475                 self.wheels.set(y[7])
476                 self.cc.set(y[8])
477                 self.gear.set(y[9])
478                 self.power.set(y[10])
479             sql2 = "SELECT * FROM job_card where Registration_No = %s ;"
480             jobb1 =(Regsql,)
481             cursor.execute(sql2, jobb1)
482             myresult2 = cursor.fetchall()
483             cells1 = {}
484             i1 = 0
485             for r in myresult2:
486                 cells1[i1] = (myresult2[i1])
487                 i1 += 1
488             print(cells1)
489
490             draww =len(cells1)+1
491             vaa = IntVar()
492             for cellsv in range (1,draww):
493                 ins = cellsv - 1
494                 self.cells2[cellsv,0] = Radiobutton(self.scrollable_frame4, text="", variable=vaa,command=lambda: changeButton(vaa.get()), value=cellsv)
495                 self.cells2[cellsv,0].grid(row=cellsv, column=0, ipady=3, pady=1)
496
497                 self.cells2[cellsv, 1] = Entry(self.scrollable_frame4, text="", width=11)
498                 self.cells2[cellsv,1].grid(row=cellsv, column=1, ipady=5, ipadx=2)
499                 self.cells2[cellsv,1].insert(0,cells1[ins][7])
500                 self.cells2[cellsv,1].config(state="readonly")
501
502                 self.cells2[cellsv,2] = Entry(self.scrollable_frame4, text="", width=10)
503                 self.cells2[cellsv,2].grid(row=cellsv, column=2, ipady=5, ipadx=2)
504                 self.cells2[cellsv,2].insert(0,cells1[ins][8])
505                 self.cells2[cellsv,2].config(state="readonly")
506
507                 self.cells2[cellsv,3] = Entry(self.scrollable_frame4, text="", width=11)
508                 self.cells2[cellsv,3].grid(row=cellsv, column=3, ipady=5, ipadx=2)
509                 self.cells2[cellsv,3].insert(0,cells1[ins][6])
510                 self.cells2[cellsv,3].config(state="readonly")
511
512                 self.cells2[cellsv,4] = Entry(self.scrollable_frame4, text="", width=11)
```

```
                self.cells2[cellsv,4] = Entry(self.scrollable_frame4, text="", width=11)
                self.cells2[cellsv,4].grid(row=cellsv, column=4, ipady=5)
                self.cells2[cellsv,4].insert(0,cells1[ins][2])
                self.cells2[cellsv,4].config(state="readonly")

            print(self.cells2)
    #view RadioButton
            def viewclick(value):

                try:
                    connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
                    if connection.is_connected():
                        db_Info = connection.get_server_info()
                        print("Connected to MySQL Server version ", db_Info)
                        cursor = connection.cursor()
                        cursor.execute("select database();")
                        record = cursor.fetchone()
                        print("You're connected to database: ", record)
                    fg = value-1
                    partsT=cells1[fg][3]
                    quantity=cells1[fg][5]
    #for parts
                    list200 =[]
                    first0 = 0
                    jaa = ","

                    for iaa in partsT:
                        if jaa in iaa:
                            yaa = partsT.index(jaa, first0)
                            list200.append(partsT[first0:yaa])
                            first0 = yaa + 1
    #for quantity
                    list300 = []
                    first1 = 0
                    jaa1 = ","

                    for iaa1 in quantity:
                        if jaa1 in iaa1:
                            yaa1 = quantity.index(jaa1, first1)
                            list300.append(quantity[first1:yaa1])
                            first1 = yaa1 + 1


                    incr =0
                    cells3 = {}
                    for l00 in list200:
                        sql3 = "SELECT * FROM scooter_parts where S_No = %s ;"
                        cursor.execute(sql3, (l00,))
                        myresult3 = cursor.fetchall()
                        cells3[incr] =myresult3
                        incr += 1
                    print(cells3)
                    sizz = len(cells3)+1
```

```
                grnd0 =0
                for sizz0 in range(1,sizz):
                    inc1 = sizz0 - 1
                    self.cells4[sizz0,0] = Entry(self.scrollable_frame2, width=7)
                    self.cells4[sizz0,0].grid(row=sizz0, column=0)
                    self.cells4[sizz0,0].insert(0,cells3[inc1][0][1])
                    self.cells4[sizz0,0].config(state="readonly")

                    self.cells4[sizz0,1] = Entry(self.scrollable_frame2, width=62)
                    self.cells4[sizz0,1].grid(row=sizz0, column=1)
                    self.cells4[sizz0,1].insert(0,cells3[inc1][0][3])
                    self.cells4[sizz0,1].config(state="readonly")

                    self.cells4[sizz0,2] = Entry(self.scrollable_frame2, width=9)
                    self.cells4[sizz0,2].grid(row=sizz0, column=2)
                    self.cells4[sizz0,2].insert(0,list300[inc1])
                    self.cells4[sizz0,2].config(state="readonly")

                    self.cells4[sizz0,3] = Entry(self.scrollable_frame2, width=9)
                    self.cells4[sizz0,3].grid(row=sizz0, column=3)
                    self.cells4[sizz0,3].insert(0,cells3[inc1][0][4])
                    self.cells4[sizz0,3].config(state="readonly")

                    self.cells4[sizz0,4] = Entry(self.scrollable_frame2, width=9)
                    self.cells4[sizz0,4].grid(row=sizz0, column=4)
                    amount1 = float(self.cells4[sizz0,2].get())
                    amount2 = float(self.cells4[sizz0,3].get())
                    total0 = amount1*amount2
                    self.cells4[sizz0,4].insert(0,total0)
                    self.cells4[sizz0,4].config(state="readonly")
                    grnd0 = grnd0 + total0
    #for Job Performed
                JobP = cells1[fg][9]
                HourH = cells1[fg][4]

                #for Job Name
                list400 = []
                first2 = 0
                jaa2 = ","

                for iaa2 in JobP:
                    if jaa2 in iaa2:
                        yaa2 = JobP.index(jaa2, first2)
                        list400.append(JobP[first2:yaa2])
                        first2 = yaa2 + 1
                # for Hour
                list500 = []
                first3 = 0
                jaa3 = ","

                for iaa3 in HourH:
                    if jaa3 in iaa3:
                        yaa3 = HourH.index(jaa3, first3)
                        list500.append(HourH[first3:yaa3])
                        first3 = yaa3 + 1
```

```
                incr3 = 0
                cells5 = {}
                for l100 in list400:
                    sql4 = "SELECT * FROM job_performed where S_No = %s ;"
                    cursor.execute(sql4, (l100,))
                    myresult5 = cursor.fetchall()
                    cells5[incr3] = myresult5
                    incr3 += 1
                print(cells5)
                sizzzz = len(cells5) + 1

                grnd1 =0
                for sizz3 in range(1, sizzzz):
                    inc3 = sizz3 - 1
                    self.cells6[sizz3, 0] = Entry(self.scrollable_frame1, width=7)
                    self.cells6[sizz3, 0].grid(row=sizz3, column=0)
                    self.cells6[sizz3, 0].insert(0, cells5[inc3][0][0])
                    self.cells6[sizz3, 0].config(state="readonly")

                    self.cells6[sizz3, 1] = Entry(self.scrollable_frame1, width=62)
                    self.cells6[sizz3, 1].grid(row=sizz3, column=1)
                    self.cells6[sizz3, 1].insert(0, cells5[inc3][0][1])
                    self.cells6[sizz3, 1].config(state="readonly")

                    self.cells6[sizz3, 2] = Entry(self.scrollable_frame1, width=9)
                    self.cells6[sizz3, 2].grid(row=sizz3, column=2)
                    self.cells6[sizz3, 2].insert(0, list500[inc3])
                    self.cells6[sizz3, 2].config(state="readonly")
                    self.cells6[sizz3, 3] = Entry(self.scrollable_frame1, width=9)
                    self.cells6[sizz3, 3].grid(row=sizz3, column=3)
                    self.cells6[sizz3, 3].insert(0, cells5[inc3][0][2])
                    self.cells6[sizz3, 3].config(state="readonly")

                    self.cells6[sizz3, 4] = Entry(self.scrollable_frame1, width=9)
                    self.cells6[sizz3, 4].grid(row=sizz3, column=4)
                    amount3 = float(self.cells6[sizz3, 2].get())
                    amount4 = float(self.cells6[sizz3, 3].get())
                    total1 = amount3 * amount4
                    self.cells6[sizz3, 4].insert(0, total1)
                    self.cells6[sizz3, 4].config(state="readonly")
                    grnd1 = grnd1 + total1

                gstlabour = float(grnd1*28/100)
                gstparts = float(grnd0*28/100)
                grandttl = float(grnd0+gstparts+grnd1+gstlabour)
                self.Totalpart.set(grnd0)
                self.TotalLabour.set(grnd1)
                self.GstLabour.set(gstlabour)
                self.GstParts.set(gstparts)
                self.GrandTotal.set(grandttl)

                self.KM.set(cells1[fg][2])


            finally:
                if (connection.is_connected()):
```

```
                cursor.close()
                connection.close()
                print("MySQL connection is closed")

            def changeButton(value):
                cell4_len =len(self.cells4)
                cell6_len =len(self.cells6)
                del4_len =cell4_len -1
                del6_len =cell6_len -1
                final_cell4 =int(del4_len/5)
                final_cell6 =int(del6_len/5)
                final_cell4 = final_cell4 + 1
                final_cell6 = final_cell6 + 1
                if final_cell4 > 1:
                    for row in range(1,final_cell4):
                        self.cells4[row,0].destroy()
                        self.cells4[row,1].destroy()
                        self.cells4[row,2].destroy()
                        self.cells4[row,3].destroy()
                        self.cells4[row,4].destroy()

                if final_cell6 > 1:
                    for row1 in range(1,final_cell6):
                        self.cells6[row1, 0].destroy()
                        self.cells6[row1, 1].destroy()
                        self.cells6[row1, 2].destroy()
                if final_cell6 > 1:
                    for row1 in range(1,final_cell6):
                        self.cells6[row1, 0].destroy()
                        self.cells6[row1, 1].destroy()
                        self.cells6[row1, 2].destroy()
                        self.cells6[row1, 3].destroy()
                        self.cells6[row1, 4].destroy()
                    viewclick(value)


            else:
                print("Incorrect")

        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")




        # Get,Set and connectivity
        # Search Button


    def newjobf(self):
```

```
        cell4_len = len(self.cells4)
        cell6_len = len(self.cells6)
        del14_len = cell4_len - 1
        del16_len = cell6_len - 1
        final_cell4 = int(del14_len / 5)
        final_cell6 = int(del16_len / 5)
        final_cell4 =final_cell4 + 1
        final_cell6 =final_cell6 + 1

        if final_cell4 > 1:
            for row in range (1,final_cell4):
                self.cells4[row, 0].destroy()
                self.cells4[row, 1].destroy()
                self.cells4[row, 2].destroy()
                self.cells4[row, 3].destroy()
                self.cells4[row, 4].destroy()

        if final_cell6 > 1:
            for row1 in range (1,final_cell6):
                self.cells6[row1, 0].destroy()
                self.cells6[row1, 1].destroy()
                self.cells6[row1, 2].destroy()
                self.cells6[row1, 3].destroy()
                self.cells6[row1, 4].destroy()
        self.KM.set("")
        self.vkm_txt.config(state="normal")
        self.TotalLabour.set("")
        self.Totalpart.set("")
        self.GstLabour.set("")
        self.GstParts.set("")
        self.GrandTotal.set("")
        self.entry.config(state="normal")
        self.Reg22_txt.config(state="readonly")
        self.icon_btn.config(state="disabled")
        self.newjob.config(state="disabled")



    def newcustomerf(self):
        self.vclass_txt["values"]=["Scooter","Bike","Hatchback Car","Sedan Car","SUV Car"]
        self.customer.set("")
        self.cname_txt.config(state="normal")
        self.cphone_txt.config(state="normal")
        self.caddress_txt.config(state="normal")
        self.newcustomer.config(state="disabled")
        self.vclass_txt.config(state="readonly")
        self.Reg22_txt.config(state="readonly")
        self.icon_btn.config(state="disabled")
        self.entry.config(state="normal")
        self.vkm_txt.config(state="normal")


    def classvalues(self,event):
```

```
780         try:
781             connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
782             if connection.is_connected():
783                 db_Info = connection.get_server_info()
784                 print("Connected to MySQL Server version ", db_Info)
785                 cursor = connection.cursor()
786                 cursor.execute("select database();")
787                 record = cursor.fetchone()
788                 print("You're connected to database: ", record)
789             a = self.vclass_txt.get()
790             v = []
791             if a =="Scooter":
792                 sql = "SELECT DISTINCT Brand FROM marketed_vehical where Class = %s ;"
793                 cursor.execute(sql,("Scooter",))
794                 myresult = cursor.fetchall()
795                 for i in myresult:
796                     v.append(i)
797 
798             elif a =="Bike":
799                 sql1 = "SELECT DISTINCT Brand FROM marketed_vehical where Class = %s ;"
800                 cursor.execute(sql1,("Bike",))
801                 myresult1 = cursor.fetchall()
802                 for j in myresult1:
803                     v.append(j)
804 
805             elif a =="Hatchback Car":
806                 sql2 = "SELECT DISTINCT Brand FROM marketed_vehical where Class = %s ;"
807                 cursor.execute(sql2,("Hatchback Car",))
808                 myresult2 = cursor.fetchall()
809                 for k in myresult2:
810                     v.append(k)
811 
812             elif a =="Sedan Car":
813                 sql3 = "SELECT DISTINCT Brand FROM marketed_vehical where Class = %s ;"
814                 cursor.execute(sql3,("Sedan Car",))
815                 myresult3 = cursor.fetchall()
816                 for q in myresult3:
817                     v.append(q)
818 
819             elif a =="SUV Car":
820                 sql4 = "SELECT DISTINCT Brand FROM marketed_vehical where Class = %s ;"
821                 cursor.execute(sql4,("SUV Car",))
822                 myresult4 = cursor.fetchall()
823                 for w in myresult4:
824                     v.append(w)
825 
826             self.vbrand_txt["values"] = v
827             self.vbrand_txt.config(state="readonly")
828 
829         finally:
830             if (connection.is_connected()):
831                 cursor.close()
832                 connection.close()
```

```
            print("MySQL connection is closed")

    def brandvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)
            a = self.vbrand_txt.get()
            b = self.vclass_txt.get()
            v = []
            sql = "SELECT DISTINCT Model FROM marketed_vehical where Class = %s and Brand = %s ;"
            cursor.execute(sql,(b,a))
            myresult = cursor.fetchall()
            print(myresult)
            for i in myresult:
                v.append(i[0])
            print(v)
            self.vmodel_txt["values"] = v
            self.vmodel_txt.config(state="readonly")
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")


    def modelvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)
            a= self.vmodel_txt.get()
            v = []
            sql = "SELECT DISTINCT Year FROM marketed_vehical where Model = %s ;"
            cursor.execute(sql,(a,))
            myresult = cursor.fetchall()
            print(myresult)
            for i in myresult:
```

```
            print(myresult)
            for i in myresult:
                v.append(i[0])
            print(v)
            self.vyear_txt["values"] = v
            self.vyear_txt.config(state="readonly")

        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")



    def yearvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)
            a = self.vmodel_txt.get()
            b= self.vyear_txt.get()
            v = []
            sql = "SELECT DISTINCT Colour FROM marketed_vehical where Model = %s and Year = %s ;"
            cursor.execute(sql,(a,b))
            myresult = cursor.fetchall()
            print(myresult)
            for i in myresult:
                v.append(i[0])
            print(v)

            self.vcolour_txt["values"] = v
            self.vcolour_txt.config(state="readonly")
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")


    def colourvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)
```

```
            a= self.vmodel_txt.get()
            b= self.vyear_txt.get()
            v = []
            sql = "SELECT DISTINCT Fuel FROM marketed_vehical where Model = %s and Year = %s ;"
            cursor.execute(sql,(a,b))
            myresult = cursor.fetchall()
            print(myresult)
            for i in myresult:
                v.append(i[0])
            print(v)

            self.vfuel_txt["values"] = v
            self.vfuel_txt.config(state="readonly")
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")

    def fuelvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)
            a= self.vmodel_txt.get()
            b= self.vyear_txt.get()
            v = []
            sql = "SELECT DISTINCT Wheels FROM marketed_vehical where Model = %s and Year = %s ;"
            cursor.execute(sql,(a,b))
            myresult = cursor.fetchall()
            print(myresult)
            for i in myresult:
                v.append(i[0])
            print(v)

            self.vwheels_txt["values"] = v
            self.vwheels_txt.config(state="readonly")
        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")

    def wheelsvalues(self,event):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
```

```
            cursor = connection.cursor()
            cursor.execute("select database();")
            record = cursor.fetchone()
            print("You're connected to database: ", record)
        a= self.vmodel_txt.get()
        b= self.vyear_txt.get()
        v = []
        sql = "SELECT DISTINCT CC FROM marketed_vehical where Model = %s and Year = %s ;"
        cursor.execute(sql,(a,b))
        myresult = cursor.fetchall()
        print(myresult)
        for i in myresult:
            v.append(i[0])
        print(v)

        self.vcc_txt["values"] = v
        self.vcc_txt.config(state="readonly")
    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

def ccvalues(self,event):
    try:
        connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
        if connection.is_connected():
            db_Info = connection.get_server_info()
            print("Connected to MySQL Server version ", db_Info)
            cursor = connection.cursor()
            cursor.execute("select database();")
            record = cursor.fetchone()
            print("You're connected to database: ", record)
        a= self.vmodel_txt.get()
        b= self.vyear_txt.get()
        v = []
        sql = "SELECT DISTINCT Gear FROM marketed_vehical where Model = %s and Year = %s ;"
        cursor.execute(sql,(a,b))
        myresult = cursor.fetchall()
        print(myresult)
        for i in myresult:
            v.append(i[0])
        print(v)

        self.vgear_txt["values"] = v
        self.vgear_txt.config(state="readonly")
    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

def gearvalues(self,event):
    try:
```

```
1041        def gearvalues(self,event):
1042            try:
1043                connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
1044                if connection.is_connected():
1045                    db_Info = connection.get_server_info()
1046                    print("Connected to MySQL Server version ", db_Info)
1047                    cursor = connection.cursor()
1048                    cursor.execute("select database();")
1049                    record = cursor.fetchone()
1050                    print("You're connected to database: ", record)
1051                a= self.vmodel_txt.get()
1052                b= self.vyear_txt.get()
1053                v = []
1054                sql = "SELECT DISTINCT Power_HP FROM marketed_vehical where Model = %s and Year = %s ;"
1055                cursor.execute(sql,(a,b))
1056                myresult = cursor.fetchall()
1057                print(myresult)
1058                for i in myresult:
1059                    v.append(i[0])
1060                print(v)
1061
1062                self.vpower_txt["values"] = v
1063                self.vpower_txt.config(state="readonly")
1064            finally:
1065                if (connection.is_connected()):
1066                    cursor.close()
1067                    connection.close()
1068                    print("MySQL connection is closed")
1072        #Job Card Functions and Events for Create Job And Create Customer
1073        def listbox_show(self, j):
1074         if self.entry["state"] == NORMAL:
1075            self.entry.delete(0,END)
1076            try:
1077                connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
1078                if connection.is_connected():
1079                    db_Info = connection.get_server_info()
1080                    print("Connected to MySQL Server version ", db_Info)
1081                    cursor = connection.cursor()
1082                    cursor.execute("select database();")
1083                    record = cursor.fetchone()
1084                    print("You're connected to database: ", record)
1085
1086                # parts table
1087                sql1 = "SELECT * FROM scooter_parts where Model = %s ;"
1088                vehsql = self.vmodel_txt.get()
1089                cursor.execute(sql1, (vehsql,))
1090                myresult1 = cursor.fetchall()
1091                print(myresult1,"this are the values of scooter parts")
1092                inss = 0
1093                for y in myresult1:
1094                    self.cells7[inss] = (y)
1095                    inss += 1
1096                print(self.cells7)
1097                # job performed
```

```
            # job performed
            sql2 = "SELECT * FROM job_performed ;"
            cursor.execute(sql2, )
            myresult2 = cursor.fetchall()
            print(myresult2)
            inss1 = 0
            for zz in myresult2:
                self.cells8[inss1] = (zz)
                inss1 += 1
            print(self.cells8)
            self.test_list = []
            lencell7 = len(self.cells7)
            print(lencell7)
            for x in range(lencell7):
                self.test_list.append(self.cells7[x][3])

            lencell8 = len(self.cells8)
            print(lencell8)
            for tt in range(lencell8):
                self.test_list.append(self.cells8[tt][1])

            print(self.test_list)
            self.listbox.destroy()
            self.listbox = Listbox(self.FF32,width=27)
            self.listbox.place(x=446, y=36)
            # listbox.bind('<Double-Button-1>', on_select)
            self.listbox.bind('<<ListboxSelect>>', self.on_select)
            self.listbox_update(self.test_list)

        finally:
            if (connection.is_connected()):
                cursor.close()
                connection.close()
                print("MySQL connection is closed")

    def on_keyrelease(self, event):

        # get text from entry
        value = event.widget.get()
        value = value.strip().lower()

        # get data from test_list
        if value == '' or value.isspace() == TRUE:
            data = self.test_list
        else:
            data = []
            for item in self.test_list:
                if value in item.lower():
                    data.append(item)

        # update data in listbox
        self.listbox_update(data)

    def listbox_update(self, data):
```

```
1150        def listbox_update(self, data):
1151            # delete previous data
1152            self.listbox.delete(0, 'end')
1153
1154            # sorting data
1155            data = sorted(data, key=str.lower)
1156            # put new data
1157            for item in data:
1158                self.listbox.insert('end', item)
1159
1160        def on_select(self, event):
1161            # display element selected on list
1162            print('(event) previous:', event.widget.get('active'))
1163            print('(event)  current:', event.widget.get(event.widget.curselection()))
1164            value1 = event.widget.get(event.widget.curselection())
1165            print(value1)
1166            lencell9 = len(self.cells9)
1167            mincell9 = lencell9 - 1
1168            totlcell9 = int(mincell9 / 5 + 1)
1169            print(totlcell9)
1170            checkval = 0
1171            for i in self.cells7:
1172                if value1 == self.cells7[checkval][3]:
1173                    self.cells9[totlcell9, 0] = Entry(self.scrollable_frame2, width=7)
1174                    self.cells9[totlcell9, 0].grid(row=totlcell9, column=0)
1175                    self.cells9[totlcell9, 0].insert(0, self.cells7[checkval][1])
1176                    self.cells9[totlcell9, 0].config(state="readonly")
1177
1178                    self.cells9[totlcell9, 1] = Entry(self.scrollable_frame2, width=62)
1179                    self.cells9[totlcell9, 1].grid(row=totlcell9, column=1)
1180                    self.cells9[totlcell9, 1].insert(0, self.cells7[checkval][3])
1181                    self.cells9[totlcell9, 1].config(state="readonly")
1182
1183                    self.cells9[totlcell9, 2] = Entry(self.scrollable_frame2, width=9)
1184                    self.cells9[totlcell9, 2].grid(row=totlcell9, column=2)
1185                    # self.cells9[totlcell9,2].bind('<KeyRelease>',self.autocalculation)
1186                    self.cells9[totlcell9, 2].bind("<KeyRelease>",
1187                                                   lambda event, cellno=9, rowno=totlcell9: self.autocalculation(event, cellno,
1188                                                                                                                 rowno))
1189
1190                    self.cells9[totlcell9, 3] = Entry(self.scrollable_frame2, width=9)
1191                    self.cells9[totlcell9, 3].grid(row=totlcell9, column=3)
1192                    self.cells9[totlcell9, 3].insert(0, self.cells7[checkval][4])
1193                    self.cells9[totlcell9, 3].config(state="readonly")
1194
1195                    self.cells9[totlcell9, 4] = Entry(self.scrollable_frame2, width=9)
1196                    self.cells9[totlcell9, 4].grid(row=totlcell9, column=4)
1197                    self.cells9[totlcell9, 4].config(state="readonly")
```

```
            self.cells9[totlcell9, 4].grid(row=totlcell9, column=4)
            self.cells9[totlcell9, 4].config(state="readonly")
        checkval = checkval + 1

    lencell10 = len(self.cells10)
    mincell10 = lencell10 - 1
    totlcell10 = int(mincell10 / 5 + 1)
    print(totlcell10)
    checkval2 = 0


    for i in self.cells8:
        if value1 == self.cells8[checkval2][1]:
            self.cells10[totlcell10, 0] = Entry(self.scrollable_frame1, width=7)
            self.cells10[totlcell10, 0].grid(row=totlcell10, column=0)
            self.cells10[totlcell10, 0].insert(0, self.cells8[checkval2][0])
            self.cells10[totlcell10, 0].config(state="readonly")

            self.cells10[totlcell10, 1] = Entry(self.scrollable_frame1, width=62)
            self.cells10[totlcell10, 1].grid(row=totlcell10, column=1)
            self.cells10[totlcell10, 1].insert(0, self.cells8[checkval2][1])
            self.cells10[totlcell10, 1].config(state="readonly")

            self.cells10[totlcell10, 2] = Entry(self.scrollable_frame1, width=9)
            self.cells10[totlcell10, 2].grid(row=totlcell10, column=2)
            # self.cells10[totlcell10,2].bind('<KeyRelease>',self.autocalculation)
            self.cells10[totlcell10, 2].bind("<KeyRelease>",lambda event, cellno=10, rowno=totlcell10: self.autocalculation(event,cellno,rowno))

            self.cells10[totlcell10, 3] = Entry(self.scrollable_frame1, width=9)
            self.cells10[totlcell10, 3].grid(row=totlcell10, column=3)
            self.cells10[totlcell10, 3].insert(0, self.cells8[checkval2][2])
            self.cells10[totlcell10, 3].config(state="readonly")

            self.cells10[totlcell10, 4] = Entry(self.scrollable_frame1, width=9)
            self.cells10[totlcell10, 4].grid(row=totlcell10, column=4)
            self.cells10[totlcell10, 4].config(state="readonly")
        checkval2 = checkval2 + 1

    print('---')
    self.entry.delete(0,END)
    self.entry.insert(0,"Search")
    self.listbox.destroy()

def autocalculation(self, event, cellno, rowno):
    value0 = event.widget.get()
    if value0.isspace() == TRUE or value0 == '':
        if cellno == 9:
            self.cells9[rowno, 4].config(state="normal")
            self.cells9[rowno, 4].delete(0, END)
            self.cells9[rowno, 4].config(state="readonly")
        elif cellno == 10:
            self.cells10[rowno, 4].config(state="normal")
            self.cells10[rowno, 4].delete(0, END)
            self.cells10[rowno, 4].config(state="readonly")

```

```
1251        else:
1252            value2 = float(value0)
1253            print(value2)
1254            print(cellno)
1255            print(rowno)
1256            find = re.compile(r"^\d*[.,]?\d*$")
1257
1258            if find.match(value0):
1259                print('TRUE')
1260                if cellno == 9:
1261                    rate0 = self.cells9[rowno, 3].get()
1262                    rate1 = float(rate0)
1263                    print(rate1)
1264                    self.cells9[rowno, 4].config(state="normal")
1265                    self.cells9[rowno, 4].delete(0, END)
1266                    self.cells9[rowno, 4].insert(0, value2 * rate1)
1267                    self.cells9[rowno, 4].config(state="readonly")
1268
1269                    lenthofcell =len(self.cells9)
1270                    lenthminus =lenthofcell-1
1271                    finallenth =int(lenthminus/5+1)
1272                    Sum=0.0
1273                    for i in range(1,finallenth):
1274                        amount = self.cells9[i,4].get()
1275                        if amount.isspace() == TRUE or amount == '':
1276                            pass
1277                        else:
1278                            amount1 =float(amount)
1279                            Sum =Sum+amount1
1280
1281                        self.Totalpart.set(str(Sum))
1282                        gst=(Sum*28/100)
1283                        self.GstParts.set(str(gst))
1284                        labour =self.TotalLabour.get()
1285                        if labour =='':
1286                            labour1=0.0
1287                        else:
1288                            labour1 =float(labour)
1289                        gstlabour=self.GstLabour.get()
1290                        if gstlabour =='':
1291                            gstlabour1=0.0
1292                        else:
1293                            gstlabour1=float(gstlabour)
1294                        finalamount=Sum+gst+labour1+gstlabour1
1295                        self.GrandTotal.set(str(finalamount))
1296
1297
1298                if cellno == 10:
1299                    rate2 = self.cells10[rowno, 3].get()
1300                    rate3 = float(rate2)
1301                    self.cells10[rowno, 4].config(state="normal")
1302                    self.cells10[rowno, 4].delete(0, END)
1303                    self.cells10[rowno, 4].insert(0, value2 * rate3)
1304                    self.cells10[rowno, 4].config(state="readonly")
1305
```

```
                self.cells10[rowno, 4].config(state="readonly")

                lenthofcell1 = len(self.cells10)
                lenthminus1 = lenthofcell1 - 1
                finallenth1 = int(lenthminus1 / 5 + 1)
                Sum1 = 0.0
                for j in range(1,finallenth1):
                    amount2 = self.cells10[j, 4].get()
                    if amount2.isspace() == TRUE or amount2 == '':
                        pass
                    else:
                        amount3 = float(amount2)
                        Sum1 = Sum1 + amount3

                    self.TotalLabour.set(str(Sum1))
                    gst1 = (Sum1 * 28 / 100)
                    self.GstLabour.set(str(gst1))
                    parts2 = self.Totalpart.get()
                    if parts2 =='':
                        parts3 = 0.0
                    else:
                        parts3 = float(parts2)
                    gstparts2 = self.GstParts.get()
                    if gstparts2 =='':
                        gstparts3 =0.0
                    else:
                        gstparts3 = float(gstparts2)
                    finalamount1 = Sum1 + gst1 + parts3 + gstparts3
                        gstparts3 =0.0
                    else:
                        gstparts3 = float(gstparts2)
                    finalamount1 = Sum1 + gst1 + parts3 + gstparts3
                    self.GrandTotal.set(str(finalamount1))

                else:
                    print("False")



    def savebutton(self):
        try:
            connection = mysql.connector.connect(host='localhost', database='bill', user='root', password='login')
            if connection.is_connected():
                db_Info = connection.get_server_info()
                print("Connected to MySQL Server version ", db_Info)
                cursor = connection.cursor()
                cursor.execute("select database();")
                record = cursor.fetchone()
                print("You're connected to database: ", record)

            a = str(self.vclass_txt.get())
            b = str(self.vbrand_txt.get())
            c = str(self.vmodel_txt.get())
            d = str(self.vyear_txt.get())
            e = str(self.vcolour_txt.get())
            f = str(self.vfuel_txt.get())
```

```
        f = str(self.vfuel_txt.get())
        g = str(self.vwheels_txt.get())
        h = str(self.vcc_txt.get())
        i = str(self.vgear_txt.get())
        j = str(self.vpower_txt.get())

        cus_name = str(self.cname_txt.get())
        cus_phone = str(self.cphone_txt.get())
        cus_address = str(self.caddress_txt.get("1.0", END))
        latest_KM1 = self.vkm_txt.get()
        print(latest_KM1)
        latest_KM = str(latest_KM1)
        reg11=re.compile('^[0-9]*$')
        if a == "Select" or b== "Select" or c=="Select" or d=="Select" or e=="Select" or f=="Select" or g=="Select" or h=="Select" or i=="Select" o
            messagebox.showerror("Incomplete","Please Select Or fill all the necessary details Correctly.")
        elif len(cus_address) >144 :
            messagebox.showerror(("Not Valid Address","Please only enter less than 144 Characters including Spaces"))
        elif len(cus_phone) >10 or len(cus_phone) < 10:
            messagebox.showerror("Not Valid Phone Number","Please Enter A Valid 10 digit Number")

        else:
            sql = "SELECT S_No from marketed_vehical where Class = %s and Brand = %s and Model = %s and Year = %s and Colour = %s and Fuel = %s
            cursor.execute(sql, (a, b, c, d, e, f, g, h, i, j))
            myresult = cursor.fetchall()
            vehical_Sno = myresult[0][0]
            print(vehical_Sno)


            sql1 = "SELECT max(S_NO) from customer_details ;"
            cursor.execute(sql1, )
            myresult1 = cursor.fetchall()
            customer_Sno1 = (myresult1[0][0]) + 1
            print(customer_Sno1)
            customer_Sno=str(customer_Sno1)

            sql2 = "SELECT max(S_NO) from job_card ;"
            cursor.execute(sql2, )
            myresult2 = cursor.fetchall()
            jobcard_Sno1 = (myresult2[0][0]) + 1
            print(jobcard_Sno1)
            jobcard_Sno =str(jobcard_Sno1)

            table_check = self.vclass_txt.get()
            if table_check == "Scooter":
                table_name = 'scooter_parts'

            elif table_check == "Bike":
                table_name = 'bike_parts'

            elif table_check == "Hatchback Car":
                table_name = 'hatchback_car_parts'

            elif table_check == "Sedan Car":
                table_name = 'sedan_car_parts'
```

```
            table_name = 'sedan_car_parts'

        elif table_check == "SUV Car":
            table_name = 'suv_car_parts'

        jobperformed_S_no = ""
        jobhours_nos = ""
        cell10len = len(self.cells10)
        if cell10len > 1:
          minuscell10 = cell10len - 1
          finalcell10 = int(minuscell10 / 5 + 1)
          for j in range(1, finalcell10):
            jobname = self.cells10[j, 1].get()
            sql4 = "SELECT S_NO from job_performed where Job_Name = %s ;"
            cursor.execute(sql4, (jobname,))
            myresult4 = cursor.fetchall()
            jobname_Sno = myresult4[0][0]
            jobperformed_S_no = jobperformed_S_no + str(jobname_Sno) + ","

            jobhours1 = self.cells10[j, 2].get()
            jobhours_nos = jobhours_nos + str(jobhours1) + ","

          print(jobperformed_S_no)
          print(jobhours_nos)

          if cell10len > 1:
           if jobhours_nos.find(",,") ==-1:
            if jobhours_nos[0] ==",":
              messagebox.showerror("Empty Field in Job Card","Please Fill the No of Hours")
            else:
              print("not found")
           else:
            messagebox.showerror("Empty Field in Job Card", "Please Fill the No of Hours")
        parts_S_no1 = ""
        quantity_nos = ""
        cell9len = len(self.cells9)
        if cell9len > 1:
            minuscell9 = cell9len - 1
            finalcell9 = int(minuscell9 / 5 + 1)
            for i in range(1, finalcell9):
                partname = self.cells9[i, 1].get()
                sql3 = "SELECT S_NO from {} where Part_Name = %s ;".format(table_name)
                cursor.execute(sql3, (partname,))
                myresult3 = cursor.fetchall()
                partname_Sno = myresult3[0][0]
                parts_S_no1 = parts_S_no1 + str(partname_Sno) + ","

                quantity1 = self.cells9[i, 2].get()
                quantity_nos = quantity_nos + str(quantity1) + ","
                print(parts_S_no1)
                print(quantity_nos)
                if quantity_nos.find(",,") ==-1:
                  if quantity_nos[0] == ",":
                    if cell9len > 1:
```

```
                if quantity_nos.find(",,") ==-1:
                    if quantity_nos[0] == ",":
                        if cell9len > 1:
                            messagebox.showerror("Empty Field in Job Card","Please Fill the No of Quantity")
                        else:
                            print("not found2")
                    else:
                        messagebox.showerror("Empty Field in Job Card","Please Fill the No of Quantity")

            import random
            import math

            data = "0123456789"
            leng = len(data)

            invoice_no = ""

            for i in range(5):
                invoice_no += data[math.floor(random.random() * leng)]

            print(invoice_no)
            self.invoice_no1 = str(invoice_no)

            sql0 = "SELECT Registration_No from customer_details where Registration_No = %s ;"
            research = self.regsearch.get()
            cursor.execute(sql0, (research,))
            myresult0 = cursor.fetchall()
            print(myresult0)
            sql0 = "SELECT Registration_No from customer_details where Registration_No = %s ;"
            research = self.regsearch.get()
            cursor.execute(sql0, (research,))
            myresult0 = cursor.fetchall()
            print(myresult0)
            if len(myresult0) == 0:
                finalresult0 = '0'
            else:
                finalresult0 = myresult0[0][0]

            if finalresult0 != research:
                if messagebox.askyesno("Are You Sure","The Details you entered Once Saved won't be changed. IF Correct then Click on 'Yes' otherwi
                    sql5 = "INSERT INTO customer_details VALUES( %s, %s ,%s , %s , %s , %s );"
                    cursor.execute(sql5,(customer_Sno, research, cus_name, cus_phone, cus_address, vehical_Sno))
                    connection.commit()
                    print(cursor.rowcount, "record5 inserted.")

                    sql6 = "INSERT INTO job_card VALUES(%s , %s , %s , %s, %s , %s , %s ,curdate(),curtime(), %s) ;"
                    cursor.execute(sql6, (jobcard_Sno, research, latest_KM, parts_S_no1, jobhours_nos, quantity_nos,self.invoice_no1, jobperformed
                    connection.commit()
                    print(cursor.rowcount, "record6 inserted.")
                    self.Generate_btn.config(state="normal")

            else:
                sql7 = "INSERT INTO job_card VALUES(%s , %s , %s , %s, %s , %s , %s ,curdate(),curtime(), %s) ;"
                cursor.execute(sql7, (jobcard_Sno, research, latest_KM, parts_S_no1, jobhours_nos, quantity_nos, self.invoice_no1,jobperformed_S_n
                connection.commit()
                print(cursor.rowcount, "record6 inserted.")
```

```
1503                    self.Generate_btn.config(state="normal")
1504
1505                else:
1506                    sql7 = "INSERT INTO job_card VALUES(%s , %s , %s , %s, %s , %s , %s ,curdate(),curtime(), %s) ;"
1507                    cursor.execute(sql7, (jobcard_Sno, research, latest_KM, parts_S_no1, jobhours_nos, quantity_nos, self.invoice_no1,jobperformed_S_
1508                    connection.commit()
1509                    print(cursor.rowcount, "record6 inserted.")
1510                    self.Generate_btn.config(state="normal")
1511
1512
1513
1514
1515
1516
1517        finally:
1518            if (connection.is_connected()):
1519                cursor.close()
1520                connection.close()
1521                print("MySQL connection is closed")
1522
1523
1524
1525        def generatebill(self):
1526            import webbrowser
1527            from reportlab.pdfgen import canvas
1528            from reportlab.platypus import Paragraph, Table, TableStyle, Frame
1529            from reportlab.lib import colors
1530            from reportlab.lib.pagesizes import A4
1525        def generatebill(self):
1526            import webbrowser
1527            from reportlab.pdfgen import canvas
1528            from reportlab.platypus import Paragraph, Table, TableStyle, Frame
1529            from reportlab.lib import colors
1530            from reportlab.lib.pagesizes import A4
1531            from reportlab.lib.styles import getSampleStyleSheet
1532            from reportlab.lib.units import inch
1533            from reportlab.lib.styles import ParagraphStyle
1534            from reportlab.lib.enums import TA_CENTER
1535            from reportlab.pdfbase.ttfonts import TTFont, TTEncoding
1536            pdf = canvas.Canvas("pdf2342.pdf", pagesize=A4)
1537            flow_obj = []
1538            styles = getSampleStyleSheet()
1539
1540            # Company Name
1541            p = ParagraphStyle('test')
1542            p.textColor = 'black'
1543            p.borderColor = 'black'
1544            p.alignment = TA_CENTER
1545            p.fontSize = 15
1546            text1 = Paragraph("Shiva Auto Services", p)
1547            flow_obj.append(text1)
1548            frame1 = Frame(5, 800, 150, 40, showBoundary=0)
1549            frame1.addFromList(flow_obj, pdf)
1550
1551            # Tagline
```

```
1602        t = Table([["DATE", datefromdb, "INVOICE #", "INV-00-"+self.invoice_no1]], colWidths=[2.06 * inch] * 5)
1603        ts = TableStyle([("BACKGROUND", (0, 0), (-1, -1), colors.white),
1604                         ("TEXTCOLOR", (0, 0), (-1, 0), colors.black),
1605                         ("TEXTCOLOR", (0, 0), (0, 0), colors.blue),
1606                         ("TEXTCOLOR", (2, 0), (2, 0), colors.blue),
1607                         ("ALIGN", (2, 0), (2, 0), 'RIGHT')
1608
1609                         ])
1610        t.setStyle(ts)
1611        flow_obj.append(t)
1612        frame1 = Frame(0, 758, 595, 30, showBoundary=0)
1613        frame1.addFromList(flow_obj, pdf)
1614
1615        # Customer info and vehical info header
1616        column_width9 = 2.06 * inch
1617        column_width10 = 3.0 * inch
1618        column_width11 = 2.06 * inch
1619        column_width12 = 2.06 * inch
1620        t = Table([["CUSTOMER INFO", "", "VEHICLE INFO", ""]], colWidths=[2.06 * inch] * 5)
1621        ts = TableStyle([("BACKGROUND", (0, 0), (-1, -1), colors.blue),
1622                         ("TEXTCOLOR", (0, 0), (-1, 0), colors.white)
1623                         ])
1624        t.setStyle(ts)
1625        flow_obj.append(t)
1626        frame1 = Frame(0, 730, 595, 30, showBoundary=0)
1627        frame1.addFromList(flow_obj, pdf)
1628
1629        # Customer info and vehical info
```

```
1629        # Customer info and vehical info
1630
1631        address1 =str(self.caddress_txt.get(1.0,END))
1632        print(type(address1))
1633        length1 = len(address1)
1634        length2 =length1-1
1635        print(length1)
1636        line1 = ""
1637        line2 = ""
1638        line3 = ""
1639        if length1 > 96 and length1 <= 144:
1640            line1 = address1[0:48] + "-"
1641            line2 = address1[49:96] + "-"
1642            line3 = address1[97:length2]
1643        elif length1 > 48 and length1 <= 96:
1644            line1 = address1[0:48] + "-"
1645            line2 = address1[49:length2]
1646        else:
1647            line1 = address1[0:length2]
1648        print(line1)
1649
1650        column_width5 = 2.06 * inch
1651        column_width6 = 3.0 * inch
1652        column_width7 = 2.06 * inch
1653        column_width8 = 2.06 * inch
1654        t = Table([["NAME:", self.customer.get(), "BRAND",self.Brand.get(),],
1655                   ["", "", "MODEL",self.model.get()],
1656                   ["ADDRESS:",line1, "YEAR", self.year.get()],
```

```
1650        column_width5 = 2.06 * inch
1651        column_width6 = 3.0 * inch
1652        column_width7 = 2.06 * inch
1653        column_width8 = 2.06 * inch
1654        t = Table([["NAME:", self.customer.get(), "BRAND",self.Brand.get() ],
1655                   ["", "", "MODEL",self.model.get()],
1656                   ["ADDRESS:",line1, "YEAR", self.year.get()],
1657                   ["",line2, "COLOR", self.colour.get()],
1658                   ["",line3, "CC", self.cc.get()],
1659                   ["Phone No:",self.Phone.get(), "REG#",self.regsearch.get()],
1660                   ["", "", "KM", self.KM.get()]], colWidths=[2.06 * inch] * 5)
1661        ts = TableStyle([("BACKGROUND", (0, 0), (-1, -1), colors.lightblue),
1662                         #("GRID",(0,0),(-1,-1),1,colors.yellow),
1663                        ("TEXTCOLOR", (0, 0), (-1, -1), colors.blue),
1664                        ('ALIGN', (0, 0), (-1, -1), 'RIGHT'),
1665                        ("ALIGN", (1, 0), (1, 6), "LEFT"),
1666                        ("ALIGN", (3, 0), (3, 6), "LEFT"),
1667                        ("TEXTCOLOR", (1, 0), (1, 6), colors.black),
1668                        ("TEXTCOLOR", (3, 0), (3, 6), colors.black),
1669
1670                        ])
1671        t.setStyle(ts)
1672        flow_obj.append(t)
1673        frame1 = Frame(0, 592, 595, 150, showBoundary=0)
1674        frame1.addFromList(flow_obj, pdf)
1675
1676        # Job Performed
1677
1678        jobperformed_data =[["Job Performed", "Hours", "Rate", "Amount"],
1679                   ["", "", "", "-"],
1680                   ["", "", "", "-"],
1681                   ["", "", "", "-"],
1682                   ["", "", "", "-"],
1683                   ["", "", "", "-"],
1684                   ["", "", "", "-"],
1685                   ["", "", "", "-"],
1686                   ["", "", "", "-"], ]
1687        lencell10 = len(self.cells10)
1688        minuscell10 =lencell10-1
1689        finalcell10 =int(minuscell10/5 +1)
1690
1691        if finalcell10>1:
1692            for i in range(1,finalcell10):
1693                jobperformed_data[i][0] = str(self.cells10[i,1].get())
1694                jobperformed_data[i][1] = str(self.cells10[i,2].get())
1695                jobperformed_data[i][2] = str(self.cells10[i,3].get())
1696                jobperformed_data[i][3] = str(self.cells10[i,4].get())
1697
1698
1699        column_width13 = 5.09 * inch
1700        column_width14 = 1.06 * inch
1701        column_width15 = 1.06 * inch
1702        column_width16 = 1.06 * inch
1703        t = Table(jobperformed_data, colWidths=[column_width13, column_width14, column_width15, column_width16] * 5)
```

```
        column_width13 = 5.09 * inch
        column_width14 = 1.06 * inch
        column_width15 = 1.06 * inch
        column_width16 = 1.06 * inch
        t = Table(jobperformed_data, colWidths=[column_width13, column_width14, column_width15, column_width16] * 5)
        ts = TableStyle([("BACKGROUND", (0, 0), (-1, 0), colors.blue),
                         ("TEXTCOLOR", (0, 0), (-1, 0), colors.white),
                         ("GRID", (0, 1), (-1, -1), 1, colors.black),
                         ("ALIGN", (1, 0), (1, 8), "CENTER"),
                         ("ALIGN", (2, 0), (2, 8), "CENTER"),
                         ("ALIGN", (3, 0), (3, 8), "CENTER"),


                         ])
        t.setStyle(ts)
        flow_obj.append(t)
        frame1 = Frame(0, 436, 595, 180, showBoundary=0)
        frame1.addFromList(flow_obj, pdf)

        # Job Performed Calculation
        column_width17 = 5.68 * inch
        column_width18 = 1.0 * inch
        column_width19 = 0.5 * inch
        column_width20 = 1.06 * inch
        rupee_uni = "\u20B9"
        print(rupee_uni)
        t = Table([["", "SUBTOTAL", "$", self.TotalLabour.get()],
                   ["", "GST", "28%", self.GstLabour.get()]],
                  colWidths=[column_width17, column_width18, column_width19, column_width20] * 5)
        ts = TableStyle([("BACKGROUND", (0, 0), (-1, -1), colors.white),
                         ("TEXTCOLOR", (0, 0), (-1, -1), colors.blue),
                         # ("GRID",(0,0),(-1,-1),1,colors.black),
                         ("ALIGN", (0, 0), (-1, -1), "CENTER")
                         ])
        t.setStyle(ts)
        flow_obj.append(t)
        frame1 = Frame(0, 400, 595, 50, showBoundary=0)
        frame1.addFromList(flow_obj, pdf)

        # Parts

        Partspurchaced_data = [["PART #", "PART NAME", "QTY", "UNIT PRICE", "AMOUNT"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"],
                   ["", "", "", "", "-"], ]

        lencell9 = len(self.cells9)
        minuscell9 = lencell9 - 1
        finalcell9 = int(minuscell9 / 5 + 1)
```

```
        minuscell9 = lencell9 - 1
        finalcell9 = int(minuscell9 / 5 + 1)

        if finalcell9 > 1:
            for j in range(1, finalcell9):
                Partspurchaced_data[j][0] = str(self.cells9[j, 0].get())
                Partspurchaced_data[j][1] = str(self.cells9[j, 1].get())
                Partspurchaced_data[j][2] = str(self.cells9[j, 2].get())
                Partspurchaced_data[j][3] = str(self.cells9[j, 3].get())
                Partspurchaced_data[j][4] = str(self.cells9[j, 4].get())


        column_width21 = 0.7 * inch
        column_width22 = 4.5 * inch
        column_width23 = 1.00 * inch
        column_width24 = 1.00 * inch
        column_width25 = 1.06 * inch
        t = Table(Partspurchaced_data,colWidths=[column_width21, column_width22, column_width23, column_width24, column_width25] * 5)
        ts = TableStyle([("BACKGROUND", (0, 0), (-1, 0), colors.blue),
                         ("GRID", (0, 0), (-1, -1), 1, colors.black),
                         ("TEXTCOLOR", (0, 0), (-1, 0), colors.white),
                         ("ALIGN", (2, 0), (2, 8), "CENTER"),
                         ("ALIGN", (3, 0), (3, 8), "CENTER"),
                         ("ALIGN", (4, 0), (4, 8), "CENTER"),
                         ])
        t.setStyle(ts)
        flow_obj.append(t)
        frame1 = Frame(0, 230, 595, 180, showBoundary=0)
        frame1.addFromList(flow_obj, pdf)

        # Parts Calculation

        labourgst =float(self.GstLabour.get())
        partsgst  =float(self.GstParts.get())
        Total_Gst =labourgst+partsgst

        column_width26 = 1.06 * inch
        column_width27 = 0.5 * inch
        column_width28 = 1.06 * inch
        t = Table([["SUBTOTAL", "$", self.Totalpart.get()],
                   ["GST", "28%", self.GstParts.get()],
                   ["", "", ""],
                   ["TOTAL LABOUR", "$",self.TotalLabour.get()],
                   ["TOTAL PARTS", "$", self.Totalpart.get()],
                   ["GST", "$", Total_Gst],
                   ["TOTAL", "$", self.GrandTotal.get()]], colWidths=[column_width26, column_width27, column_width28] * 5)
        ts = TableStyle([("BACKGROUND", (0, 0), (-1, -1), colors.white),
                         ("TEXTCOLOR", (0, 0), (-1, -1), colors.blue),
                         # ("GRID",(0,0),(-1,-1),1,colors.black),
                         ("ALIGN", (0, 0), (-1, -1), "CENTER")
                         ])
        t.setStyle(ts)
        flow_obj.append(t)
        frame1 = Frame(400, 94, 210, 150, showBoundary=0)
```

```
1803        frame1 = Frame(400, 94, 210, 150, showBoundary=0)
1804        frame1.addFromList(flow_obj, pdf)
1805
1806        # COMMENTS FRAME
1807        p3 = ParagraphStyle('test')
1808        p3.textColor = 'blue'
1809        p3.borderColor = 'black'
1810        p3.alignment = TA_CENTER
1811        p3.fontSize = 10
1812        text1 = Paragraph("COMMENTS", p3)
1813        flow_obj.append(text1)
1814        frame = Frame(-10, 185, 100, 30, showBoundary=0)
1815        frame.addFromList(flow_obj, pdf)
1816
1817        # Comment Text Frame
1818        p4 = ParagraphStyle('test')
1819        p4.textColor = 'black'
1820        p4.borderColor = 'black'
1821        p4.alignment = TA_CENTER
1822        p4.fontSize = 8
1823        text1 = Paragraph("Please include the invoice number as reference when paying online or by check", p4)
1824        flow_obj.append(text1)
1825        frame = Frame(-8, 166, 320, 30, showBoundary=0)
1826        frame.addFromList(flow_obj, pdf)
1827
1828        # Make all Line
1829        p5 = ParagraphStyle('test')
1831        p5.borderColor = 'black'
1832        p5.alignment = TA_CENTER
1833        p5.fontSize = 8
1834        text1 = Paragraph("Make all checks payable to", p5)
1835        flow_obj.append(text1)
1836        frame = Frame(373, 80, 150, 30, showBoundary=0)
1837        frame.addFromList(flow_obj, pdf)
1838
1839        # Shiva Auto Services
1840        p6 = ParagraphStyle('test')
1841        p6.textColor = 'blue'
1842        p6.borderColor = 'black'
1843        p6.alignment = TA_CENTER
1844        p6.fontSize = 10
1845        text1 = Paragraph("Shiva Auto Services", p6)
1846        flow_obj.append(text1)
1847        frame = Frame(380, 72, 130, 30, showBoundary=0)
1848        frame.addFromList(flow_obj, pdf)
1849
1850        # Thankq For the Business
1851        p7 = ParagraphStyle('test')
1852        p7.textColor = 'blue'
1853        p7.borderColor = 'black'
1854        p7.alignment = TA_CENTER
1855        p7.fontSize = 12
1856        text1 = Paragraph("Thank you for your business!", p6)
1857        flow_obj.append(text1)
```

```
1910            pdf.setLineWidth(2)
1911            pdf.setStrokeColor(colors.black)
1912            pdf.line(0, 195, 400, 195)
1913
1914            pdf.save()
1915
1916            webbrowser.open_new("pdf2342.pdf")
1917            self.Generate_btn.config(state="normal")
1918
1919        def clearall(self):
1920            cell4_len = len(self.cells4)
1921            cell6_len = len(self.cells6)
1922            dell4_len = cell4_len - 1
1923            dell6_len = cell6_len - 1
1924            final_cell4 = int(dell4_len / 5)
1925            final_cell6 = int(dell6_len / 5)
1926            final_cell4 = final_cell4 + 1
1927            final_cell6 = final_cell6 + 1
1928
1929            if final_cell4 > 1:
1930                for row in range(1, final_cell4):
1931                    self.cells4[row, 0].destroy()
1932                    self.cells4[row, 1].destroy()
1933                    self.cells4[row, 2].destroy()
1934                    self.cells4[row, 3].destroy()
1935                    self.cells4[row, 4].destroy()
1936
1937            if final_cell6 > 1:
1938                for row1 in range(1, final_cell6):
1939                    self.cells6[row1, 0].destroy()
1940                    self.cells6[row1, 1].destroy()
1941                    self.cells6[row1, 2].destroy()
1942                    self.cells6[row1, 3].destroy()
1943                    self.cells6[row1, 4].destroy()
1944
1945            self.cells4.clear()
1946            self.cells4[0,0] =Entry()
1947            self.cells6.clear()
1948            self.cells6[0,0] =Entry()
1949
1950            cell2_len = len(self.cells2)
1951            dell2_len = cell2_len - 1
1952            final_cell2 = int(dell2_len / 5 +1)
1953            if final_cell2 > 1:
1954                for row3 in range(1, final_cell2):
1955                    self.cells2[row3, 0].destroy()
1956                    self.cells2[row3, 1].destroy()
1957                    self.cells2[row3, 2].destroy()
1958                    self.cells2[row3, 3].destroy()
1959                    self.cells2[row3, 4].destroy()
1960
1961            self.cells2.clear()
1962            self.cells2[0,0] =Entry()
1963
```

```
        self.cells7.clear()
        self.cells8.clear()

        cell9_len = len(self.cells9)
        dell9_len = cell9_len - 1
        final_cell9 = int(dell9_len / 5 + 1)
        if final_cell9 > 1:
            for row4 in range(1, final_cell9):
                self.cells9[row4, 0].destroy()
                self.cells9[row4, 1].destroy()
                self.cells9[row4, 2].destroy()
                self.cells9[row4, 3].destroy()
                self.cells9[row4, 4].destroy()

        cell10_len = len(self.cells10)
        dell10_len = cell10_len - 1
        final_cell10 = int(dell10_len / 5 + 1)
        if final_cell10 > 1:
            for row5 in range(1, final_cell2):
                self.cells10[row5, 0].destroy()
                self.cells10[row5, 1].destroy()
                self.cells10[row5, 2].destroy()
                self.cells10[row5, 3].destroy()
                self.cells10[row5, 4].destroy()

        self.cells9.clear()
        self.cells9[0,0] =Entry()
        self.cells9.clear()
        self.cells9[0,0] =Entry()
        self.cells10.clear()
        self.cells10[0,0] =Entry()

        self.KM.set("")
        self.vkm_txt.config(state="readonly")
        self.entry.config(state="readonly")
        self.regsearch.set("")
        self.Reg22_txt.config(state="normal")
        self.icon_btn.config(state="normal")
        self.Generate_btn.config(state="disabled")
        self.customer.set("")
        self.Phone.set("")
        self.caddress_txt.config(state="normal")
        self.caddress_txt.delete(1.0,END)
        self.caddress_txt.config(state="disabled")
        self.cname_txt.config(state="readonly")
        self.cphone_txt.config(state="readonly")
        self.vclass_txt.set("")
        self.vclass_txt.config(state="disabled")
        self.vbrand_txt.set("")
        self.vbrand_txt.config(state="disabled")
        self.vmodel_txt.set("")
        self.vmodel_txt.config(state="disabled")
        self.vyear_txt.set("")
        self.vyear_txt.config(state="disabled")
        self.vcolour_txt.set("")
```

```
2015            self.vyear_txt.config(state="disabled")
2016            self.vcolour_txt.set("")
2017            self.vcolour_txt.config(state="disabled")
2018            self.vfuel_txt.set("")
2019            self.vfuel_txt.config(state="disabled")
2020            self.vwheels_txt.set("")
2021            self.vwheels_txt.config(state="disabled")
2022            self.vcc_txt.set("")
2023            self.vcc_txt.config(state="disabled")
2024            self.vgear_txt.set("")
2025            self.vgear_txt.config(state="disabled")
2026            self.vpower_txt.set("")
2027            self.vpower_txt.config(state="disabled")
2028            self.TotalLabour.set("")
2029            self.Totalpart.set("")
2030            self.GrandTotal.set("")
2031            self.GstLabour.set("")
2032            self.GstParts.set("")
2033            self.newjob.config(state="disabled")
2034            self.newcustomer.config(state="disabled")
2035
2036        def exitbutton(self):
2037            sys.exit(0)
2038
2039
2040
2041
2042
```

## Data Base Structure and Data: -

```
mysql> use bill;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_bill       |
+----------------------+
| bike_parts           |
| customer_details     |
| hatchback_car_parts  |
| job_card             |
| job_performed        |
| login                |
| marketed_vehical     |
| scooter_parts        |
| sedan_car_parts      |
| suv_car_parts        |
+----------------------+
10 rows in set (0.02 sec)
```

```
mysql> desc customer_details;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| S_no           | int         | YES  |     | NULL    |       |
| Registration_No | varchar(10) | NO   | PRI | NULL    |       |
| Customer_Name  | char(30)    | YES  |     | NULL    |       |
| Phone_No       | bigint      | YES  |     | NULL    |       |
| Address        | varchar(150) | YES  |     | NULL    |       |
| Vehical_Sn     | int         | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
mysql> select * from customer_details;
+------+----------------+-------------------+------------+-----------------------------------------+------------+
| S_no | Registration_No | Customer_Name     | Phone_No   | Address                                 | Vehical_Sn |
+------+----------------+-------------------+------------+-----------------------------------------+------------+
|    6 | MP14MG9875     | Robert Thomas     | 9876543210 | H-764,police colony,mandsaur, m.p, 458001
 |          9 |
|    5 | MP14MH5469     | Gurjar Singh      | 9876543210 | D-09,kityani mandsaur m.p 458001
        |         10 |
|    3 | MP14MJ4578     | Jon Thomas        | 9876543210 | S-67,Pihor Colony, indore M.P 458001
      |         10 |
|    8 | MP14MK2325     | Manoj Reddi       | 9876543210 | A-876,Gandhi Choraha Mandsaur M.P 458001
  |         25 |
|    4 | MP14MK9523     | Jon Watson        | 9876543212 | A-43,pihor colony, mandsaur , m.p 458001
  |         10 |
|    1 | MP14MK9564     | Ayush Sharma      | 9826204582 | A-34,Gandhi Nagar,Mandsaur,MP,458001    |         51 |
|    2 | MP14MK9988     | Jeff Bezos        | 9876543210 | A-32,jagah khedi, mandsaur, m.p 458001
    |         13 |
|    7 | MP14MY5687     | Mahesh Joshi      | 9876543210 | D-56,Ram Tekri, Mandsaur, M.P, 458001
     |         10 |
|    9 | MP65MJ5487     | Shiva Hari V Mohan |          0 | Haveli Mandsaur M.P 458001
              |         13 |
+------+----------------+-------------------+------------+-----------------------------------------+------------+
9 rows in set (0.07 sec)
```

```
mysql> desc login;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| S_No     | int         | YES  |     | NULL    |       |
| User_ID  | varchar(20) | NO   | PRI | NULL    |       |
| password | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select * from login;
+------+---------+------------------------------------------+
| S_No | User_ID | password                                 |
+------+---------+------------------------------------------+
|    1 | login   | 2736fab291f04e69b62d490c3c09361f5b82461a |
+------+---------+------------------------------------------+
1 row in set (0.06 sec)
```

```
mysql> desc marketed_vehical;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| S_No     | int          | NO   | PRI | NULL    |       |
| Class    | char(10)     | YES  |     | NULL    |       |
| Brand    | char(10)     | YES  |     | NULL    |       |
| Model    | varchar(50)  | YES  |     | NULL    |       |
| Year     | int          | YES  |     | NULL    |       |
| Colour   | varchar(50)  | YES  |     | NULL    |       |
| Fuel     | char(10)     | YES  |     | NULL    |       |
| Wheels   | int          | YES  |     | NULL    |       |
| CC       | decimal(6,2) | YES  |     | NULL    |       |
| Gear     | char(10)     | YES  |     | NULL    |       |
| Power_HP | decimal(6,2) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

```
mysql> select * from marketed_vehical;
+------+---------+-------+--------------------------+------+--------------------------------------------+--------+--------+--------+-----------+----------+
| S_No | Class   | Brand | Model                    | Year | Colour                                     | Fuel   | Wheels | CC     | Gear      | Power_HP |
+------+---------+-------+--------------------------+------+--------------------------------------------+--------+--------+--------+-----------+----------+
|    1 | Scooter | Honda | Activa 6G                | 2020 | Black                                      | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    2 | Scooter | Honda | Activa 6G                | 2020 | Pearl Precious White                       | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    4 | Scooter | Honda | Activa 6G                | 2020 | Matte Axis Grey                            | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    5 | Scooter | Honda | Activa 6G                | 2020 | Pearl Spartan Red                          | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    6 | Scooter | Honda | Activa 6G                | 2020 | Glitter Blue                               | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    7 | Scooter | Honda | Activa 5G                | 2018 | Glitter Blue                               | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    8 | Scooter | Honda | Activa 5G                | 2018 | Pearl Spartan Red                          | Petrol |      2 | 109.00 | Automatic |     7.79 |
|    9 | Scooter | Honda | Activa 5G                | 2018 | Black                                      | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   10 | Scooter | Honda | Activa 5G                | 2018 | Matte Selene Silver                        | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   11 | Scooter | Honda | Activa 5G                | 2018 | Trance Blue Metallic                       | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   12 | Scooter | Honda | Activa 5G                | 2018 | Pearl Precious White and Matte Selene Silver | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   13 | Scooter | Honda | Activa 5G                | 2018 | Majestic Brown Metallic                    | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   14 | Scooter | Honda | Activa 5G                | 2018 | Dazzle Yellow Metallic                     | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   15 | Scooter | Honda | Activa 5G                | 2018 | Pearl Amazing White                        | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   16 | Scooter | Honda | Activa 5G                | 2018 | Silver and black                           | Petrol |      2 | 109.00 | Automatic |     7.79 |
|   17 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Tech Blue                                  | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   18 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Autumn Brown                               | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   19 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Sunlit Ivory                               | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   20 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Volcano Red                                | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   21 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Titanium Grey                              | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   22 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Mystic Gold                                | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   23 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Midnight Black                             | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   24 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Matte Silver                               | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   25 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Matte Blue                                 | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   26 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Starlight Blue                             | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   27 | Scooter | TVS   | Jupiter STD BS6          | 2020 | Royal Wine                                 | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   28 | Scooter | TVS   | Jupiter ZX BS6           | 2020 | Starlight Blue                             | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   29 | Scooter | TVS   | Jupiter ZX BS6           | 2020 | Royal Wine                                 | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   30 | Scooter | TVS   | Jupiter Grande Edition BS6 | 2020 | Tech Blue                                  | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   31 | Scooter | TVS   | Jupiter Grande Edition BS6 | 2020 | Autumn Brown                               | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   32 | Scooter | TVS   | Jupiter Grande Edition BS6 | 2020 | Sunlit Ivory                               | Petrol |      2 | 109.70 | Automatic |     7.99 |
|   33 | Scooter | Hero  | Maestro Edge VX          | 2018 | Techno Blue                                | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   34 | Scooter | Hero  | Maestro Edge VX          | 2018 | Shooting Night Star                        | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   35 | Scooter | Hero  | Maestro Edge VX          | 2018 | Pearl Silver White                         | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   36 | Scooter | Hero  | Maestro Edge VX          | 2018 | Panther Black                              | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   37 | Scooter | Hero  | Maestro Edge VX          | 2018 | Matte Vernier Grey                         | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   38 | Scooter | Hero  | Maestro Edge VX          | 2018 | Matte Blue                                 | Petrol |      2 | 110.90 | Automatic |     8.15 |
|   39 | Scooter | Hero  | Maestro Edge VX          | 2018 | Candy Blazing Red                          | Petrol |      2 | 110.90 | Automatic |     8.15 |
```

```
mysql> desc job_card;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| S_No            | int         | YES  |     | NULL    |       |
| Registration_No | varchar(10) | YES  |     | NULL    |       |
| Total_Km        | int         | YES  |     | NULL    |       |
| Part_S_No       | varchar(20) | YES  |     | NULL    |       |
| Hour            | varchar(20) | YES  |     | NULL    |       |
| Quantity        | varchar(20) | YES  |     | NULL    |       |
| Invoice_NO      | int         | NO   | PRI | NULL    |       |
| Bill_Date       | date        | YES  |     | NULL    |       |
| Bill_Time       | time        | YES  |     | NULL    |       |
| job_name        | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> select * from job_card;
+------+----------------+----------+-----------+------+----------+------------+------------+-----------+----------+
| S_No | Registration_No | Total_Km | Part_S_No | Hour | Quantity | Invoice_NO | Bill_Date  | Bill_Time | job_name |
+------+----------------+----------+-----------+------+----------+------------+------------+-----------+----------+
|    8 | MP14MJ4578      |    15878 | 6,        | 9,   | 1,       |      12569 | 2020-05-19 | 18:55:19  | 1,       |
|    3 | MP14MK9564      |     8000 | 4,1,      | 5,   | 1,1,1,   |      21258 | 2019-11-21 | 17:38:50  | 1,       |
|    1 | MP14MK9564      |    12000 | 1,4,3,    | 5,   | 1,1,1,   |      21548 | 2020-05-19 | 17:38:50  | 1,       |
|    2 | MP14MK9564      |    10000 | 1,4,1,    | 5,   | 1,1,1,   |      21958 | 2020-01-21 | 17:38:50  | 1,       |
|    5 | MP14MK9988      |    19500 |           | 6,   |          |      23222 | 2020-05-19 | 09:18:05  | 1,       |
|   12 | MP14MY5687      |    14658 | 3,1,      | 6,   | 1,1,     |      29374 | 2020-05-19 | 19:38:08  | 1,       |
|   14 | MP65MJ5487      |     5487 | 6,3,      | 10,  | 1,1,     |      31219 | 2020-05-25 | 16:58:56  | 1,       |
|   10 | MP14MH5469      |    14879 | 3,        | 9,   | 1,       |      32536 | 2020-05-19 | 19:08:40  | 1,       |
|   13 | MP14MK2325      |     8254 |           | 5,   |          |      66329 | 2020-05-25 | 16:52:50  | 1,       |
|    6 | MP14MK9988      |    20000 | 1,        | 6,   | 1,       |      75619 | 2020-05-19 | 09:22:18  | 1,       |
|   11 | MP14MG9875      |     2564 | 6,        | 6,   | 1,       |      84722 | 2020-05-19 | 19:24:11  | 1,       |
|    4 | MP14MK9988      |    15480 | 3,1,      | 9,   | 1,1,     |      87824 | 2020-05-19 | 09:12:06  | 1,       |
|    9 | MP14MK9523      |    14769 | 3,        | 9,   | 1,       |      98693 | 2020-05-19 | 18:58:33  | 1,       |
|    7 | MP14MK9988      |    26546 | 9,        | 9,   | 1,       |      99021 | 2020-05-19 | 09:35:14  | 1,       |
+------+----------------+----------+-----------+------+----------+------------+------------+-----------+----------+
14 rows in set (0.04 sec)
```

```
mysql> desc job_performed;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| S_No     | int          | YES  |     | NULL    |       |
| Job_Name | varchar(50)  | NO   | PRI | NULL    |       |
| Rate     | decimal(6,2) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select* from job_performed;
+------+----------+-------+
| S_No | Job_Name | Rate  |
+------+----------+-------+
|    1 | Labour   | 70.00 |
+------+----------+-------+
1 row in set (0.01 sec)
```

```
mysql> desc scooter_parts;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| S_No      | int         | YES  |     | NULL    |       |
| Part_No   | int         | NO   | PRI | NULL    |       |
| Model     | varchar(150)| YES  |     | NULL    |       |
| Part_Name | varchar(90) | YES  |     | NULL    |       |
| Rate      | decimal(6,2)| YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> select * from scooter_parts;
+------+---------+----------+-----------------+---------+
| S_No | Part_No | Model    | Part_Name       | Rate    |
+------+---------+----------+-----------------+---------+
|    1 |   10011 | Activa 5G | Side Stand      |  315.00 |
|    2 |   10012 | Activa 5G | Leg Guard       |  530.00 |
|    3 |   10013 | Activa 5G | Grip Set        |   57.00 |
|    4 |   10014 | Activa 5G | Silencer Assly  | 1656.00 |
|    5 |   10015 | Activa 5G | Rider Foot Rest |   99.00 |
|    6 |   10016 | Activa 5G | No Plate        |  100.00 |
|    7 |   10017 | Activa 5G | Meter Cover     |  284.00 |
|    8 |   10018 | Activa 5G | Inner Cover     | 1117.00 |
|    9 |   10019 | Activa 5G | Horn            |  203.00 |
+------+---------+----------+-----------------+---------+
9 rows in set (0.06 sec)
```

# Chapter 8

# TESTING

## 8.1 Login Page Test Case Diagram

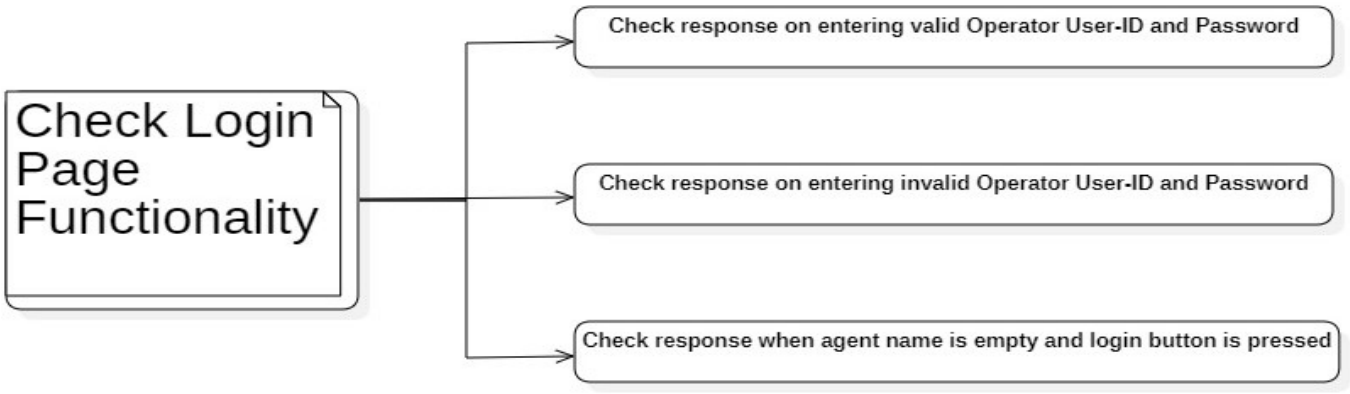


**Fig: -8.1**

## 8.2 Bill Page Test Case Diagram

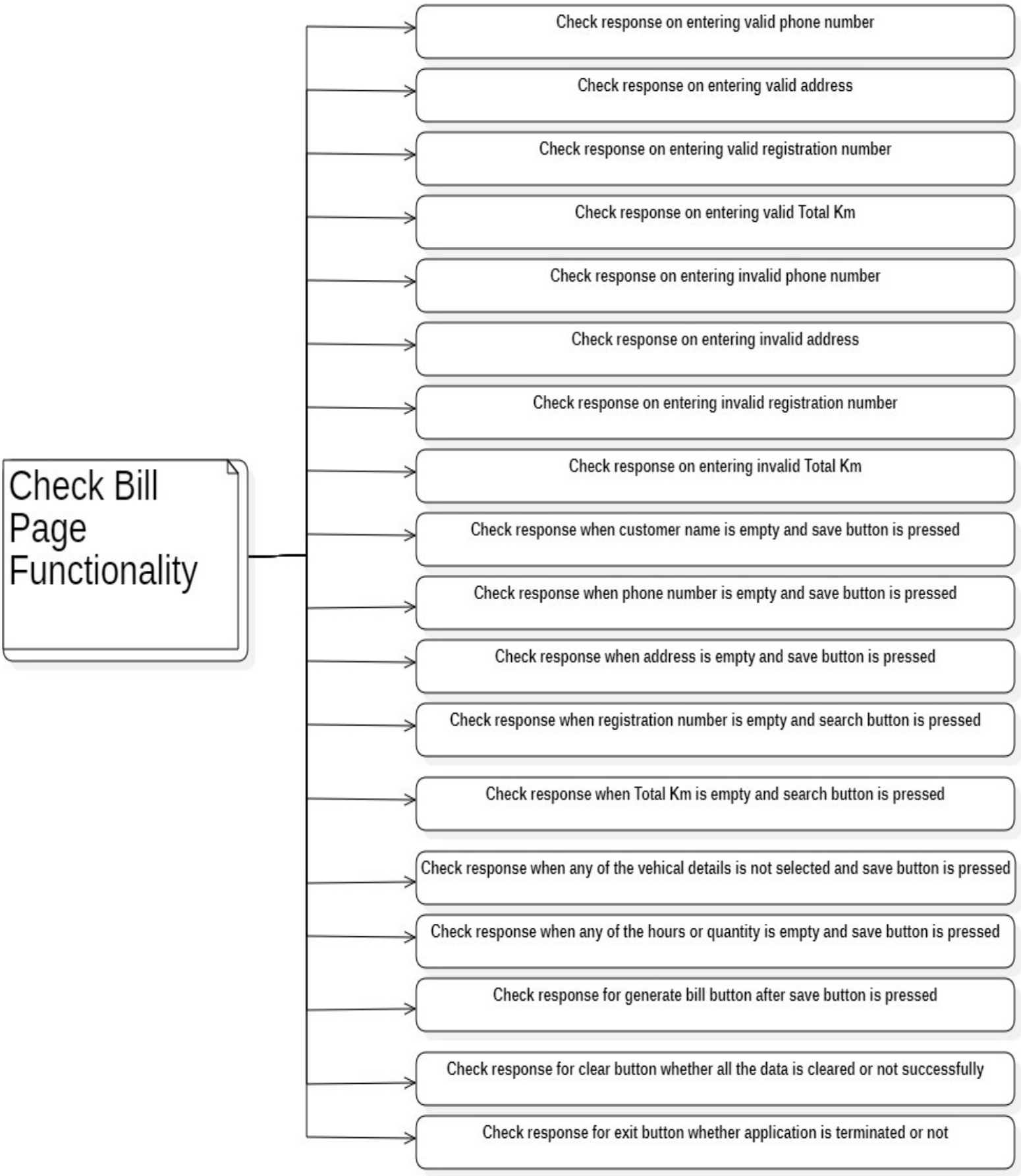


**Fig: - 8.2**

# Chapter 9

# LIMITATIONS OF PROJECT

1. It is only meant for specific Invoice generation i.e. “Automobile Service Billing”.
2. Requires PDF reader most recommended Google Chrome Browser or Adobe Acrobat Reader.
3. Requires Python 3.7 interpreter installed in your system, windows don’t provide it by default.

# Chapter 10

# CONCLUSION

This was my project of System Design about “Billing System”. Development of this System takes a lot of efforts. I think this system gave a lot of satisfaction. Though every task is never said to be perfect in this development field even more improvement may be possible in this system. I learnt so many things and gained a lot of knowledge about development field. I hope this will prove fruitful.

# Chapter 11

# BIBLIOGRAPHY/REFERENCES


1. https://stackoverflow.com/
2. https://github.com/
3. https://www.tutorialspoint.com/
4. https://www.geeksforgeeks.org/
5. https://www.javatpoint.com/
6. https://docs.python.org/3/
7. https://www.w3schools.com/python/
8. https://www.programiz.com/
9. https://www.reportlab.com/docs/reportlab-userguide.pdf