Vehicle Service Management System

A PROJECT SUBMIT TO

BHAKTA KAVI NARSINH MEHTA UNIVERSITY

FOR THE DEGREE

OF

BECHLOR OF COMPUTER APPLICATION

IN

COMPUTER SCIENCE

Student Name

DHOKIYA PAYAL

(Enrollment.Number: 20221011073)

Project Guide

MR. JAYDIP RATHOD

Associate Professor

Department Of Computer Science

Shri V.J Modha College Information

&Technology- Porbandar

Table of Contents

1. Project Profile

1.1 Introduction

The Vehicle Service Management System is a software application developed in Python, designed to streamline and automate the operations of vehicle service centers. This system provides an efficient platform to manage customer details, vehicle information, service bookings, and payment records. It eliminates the need for manual processes, ensuring accuracy and enhancing overall productivity.

In today's fast-paced world, vehicle owners often face challenges in keeping track of service schedules and maintaining service records. Similarly, service centers encounter difficulties in managing appointments, tracking service histories, and handling customer data manually. This project addresses these challenges by providing a comprehensive solution that integrates all these functions into a single system.

The system is user-friendly and ensures smooth interaction between customers and service providers. It is scalable, meaning additional features can be integrated in the future to meet growing demands, such as inventory management or online payment gateways.

1.2 Objective

The objectives of the project are to address the challenges faced by vehicle service centers and their customers by creating a comprehensive and efficient software solution. Below are the detailed objectives of the project:

1. Automating the Service Management Process:

Replace traditional, manual methods of handling service requests with a fully automated system.

Minimize human errors in record-keeping and appointment scheduling.

2. Centralized Database for Records Management

Create a centralized repository to store all customer, vehicle, and service data.

Ensure data consistency and security with a robust database system.

3. Customer Convenience and Satisfaction

Enable customers to book service appointments easily through an intuitive interface.

Offer access to detailed service history for better vehicle maintenance tracking.

4. Efficient Service Scheduling and Tracking

Allow service centers to manage multiple appointments effectively.

Allocate resources (e.g., mechanics, tools) based on the service requirements.

5. Generating Accurate Billing and Invoices

Automate the generation of bills based on the services rendered.

Ensure error-free and transparent invoicing for both customers and the service center.

6. Performance Reporting and Analysis

Generate reports on completed services, revenue, and customer trends.

Provide insights into business performance, helping the service center make informed decisions.

Track customer satisfaction and identify areas for improvement.

7. Scalability and Future Integration

Design the system to accommodate additional features like inventory management, online payments, or loyalty programs in the future.

Ensure the system is adaptable to the growing needs of service centers and their customers.

1.3 Scope

The Vehicle Service Management System is designed to streamline and automate the operations of a vehicle service center. Below is a detailed scope of the project:

1. Customer Management

The system will store and manage customer details, including their name, contact information, and vehicle details.

It ensures that customer information is easily retrievable for future interactions, such as repeat services or service history checks.

2. Vehicle Management

Allows the service center to record and maintain vehicle details, such as make, model, registration number, and engine type.

Tracks service history for each vehicle, helping customers and the service center monitor maintenance schedules.

3. Service Scheduling and Management

Allows service providers to manage and schedule appointments efficiently, minimizing overlaps or delays.

4. Billing and Invoicing

Automates the generation of accurate bills and invoices based on the services rendered and parts used.

Maintains a record of all financial transactions for reference and audit purposes.

Ensures transparent billing processes, enhancing customer trust and satisfaction.

5. Reporting and Analytics

Generates detailed reports on various metrics, including completed services, revenue, customer trends, and service center performance.

Provides insights into business operations, helping management identify strengths and areas for improvement.

6. Data Security and Integrity

Ensures that all customer, vehicle, and service data are stored securely in a centralized database.

Implements authentication mechanisms to prevent unauthorized access to the system.

Regular backups of data to avoid loss in case of system failures.

2 Requirements Analysis

2.1 Target Audience

The Vehicle Service Management System is designed to cater to a wide range of users, primarily focusing on service centers and vehicle owners. Below is a detailed description of the target audience:

1. Service Centers

The primary users of this system are small to medium-scale vehicle service centers aiming to improve their operational efficiency and customer satisfaction.

Efficient Workflow Management: Automates appointment scheduling, service tracking, and billing processes.

Improved Record-Keeping: Centralized storage of customer, vehicle, and service data for easy access.

Enhanced Customer Service: Enables faster and more accurate responses to customer inquiries.

Scalability: Allows service centers to expand their operations with minimal adjustments to the system.

2. Vehicle Owners (Customers)

Customers benefit from this system by gaining access to an organized platform for managing their vehicle service needs.

Convenient Service Booking: Ability to book services online or through a user-friendly interface.

Service History Tracking: Access to detailed records of past services, helping them keep track of maintenance schedules.

Transparent Billing: Receives accurate and itemized bills for services rendered.

3. Business Owners/Managers

The system provides business owners or managers with tools to monitor and analyze the performance of their service center.

Operational Insights: Generate reports on revenue, completed services, and customer trends to make informed decisions.

Resource Optimization: Efficient allocation of resources like mechanics, tools, and equipment based on the system's insights.

Customer Retention: Better service and communication lead to improved customer satisfaction and loyalty.

4. Employees of the Service Center

Staff members such as receptionists, mechanics, and supervisors also benefit from the system as it simplifies their day-to-day tasks.

Task Management: Clear assignment of tasks and appointments based on the schedule.

Reduced Workload: Automated processes reduce manual effort in record-keeping and billing.

Improved Communication: Easier coordination between team members for smoother operations.

5. Future Expansion for Other Audiences

The system can also be tailored for additional audiences with future upgrades:

Fleet Management Companies: To manage multiple vehicles efficiently and track their service schedules.

Large Service Chains: To provide a centralized solution for managing multiple branches.

Insurance Companies: To track service history and vehicle condition for claims processing.]

2.2 Hardware and Software

2.2.1 Hardware Requirements

1. For Development Environment

RAM: 8 GB (16 GB recommended for faster performance during development)

Operating System: Windows 10/11 or Linux (64-bit)

Additional Peripherals: Keyboard, mouse, and internet connectivity

2. For Database Server (MySQL)

RAM: 8 GB (16 GB recommended for handling large amounts of data)

Storage: 50 GB for storing database files

Operating System: Windows Server, Linux (Ubuntu, CentOS, or similar)

Network: High-speed internet

2.2.2 Software Requirements

1. Development Environment

Integrated Development Environment (IDE):Visual Studio (Community, Professional, or Enterprise Edition)

Programming Language:Python 3.10 or higher

Database Management System (DBMS):MySQL Server (version 8.0 or higher)

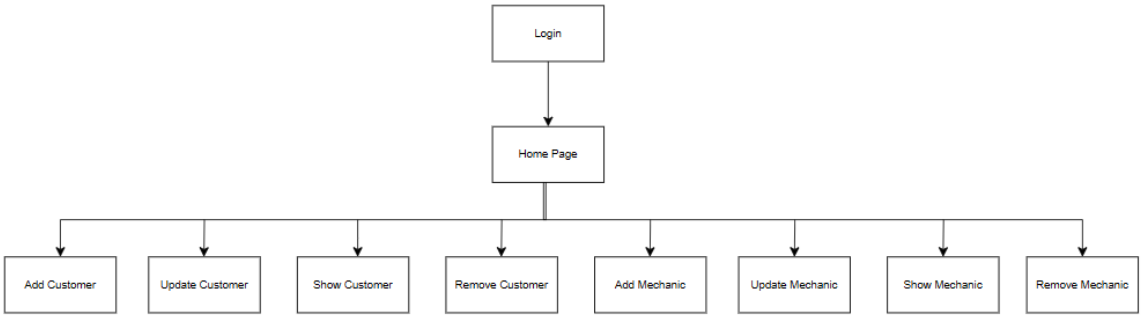
Database Client Tool:MySQL Workbench (for database design and management)

Libraries/Packages:mysql-connector-python (for connecting Python with MySQL)

Additional libraries like tkinter, flask (if GUI or web interface is needed), pandas, and matplotlib (for data analysis and visualization).

3 Website Planning and Structure

3.1 Sitemap



3.2 Key Pages

1. User Login and Registration

Login page for users (staff, admin).

Password reset functionality.

2. Home Page (Dashboard)

Overview of system functionalities.

Quick access to key features like service status, and analytics.

3. Admin Panel

Manage users (customers, employees).

Add/edit/delete service packages.

4. Customer Management

Add new customers.

View and edit customer details.

Search functionality for customer records.

5. Employee Management

Add/edit/delete employee details.

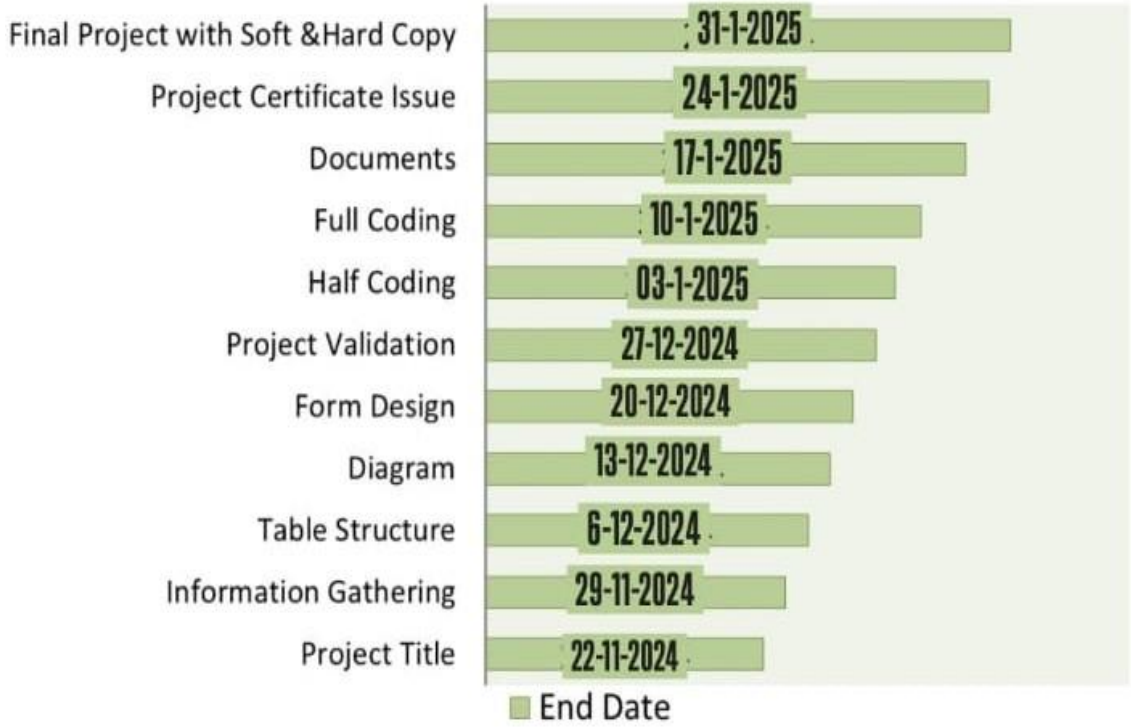
Assign tasks or services to employees.

View employee performance reports.

6. Service History

View past services for customers and vehicles.

3.3 Timeline and Milestones



4 Data Dictionary

4.1 Tables

1.Login Table

Column Name	Data Type	Description
username	Varchar(50)	Store the name of user
password	Varchar(50)	Store the password of user

2.Customer Table

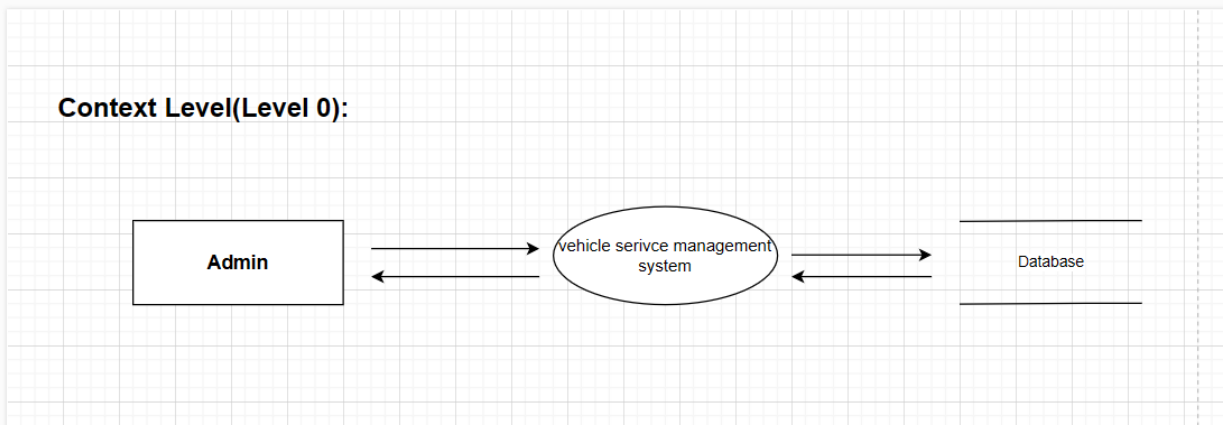
Column Name	Data Type	Description
customer_id	Int(10)	Unique identifier for each customer
Full_Name	Varchar(100)	Describe the name of user
Gender	Varchar(50)	Describe the gender of user
Phone_no	Int(10)	Describe the phone number of user
Email	Varchar(100)	Customer's email address
Address	Varchar(500)	Customer's address
vehicle_type	Varchar(200)	Type of the vehicle (e.g., Sedan, SUV, Truck)
vehicle_name	Varchar(200)	Name of the vehicle
vehicle_number	Int(20)	Registration number of vehicle
vehicle_brand	Varchar(200)	Brand of the vehicle
Service_date	Int(20)	Date when the service was performed
Service_description	Varchar(500)	Description of the service
Service_by	Varchar(200)	Name of the technician who performed the service
Service_cost	Int(20)	Total cost for the service

3. Technician Table

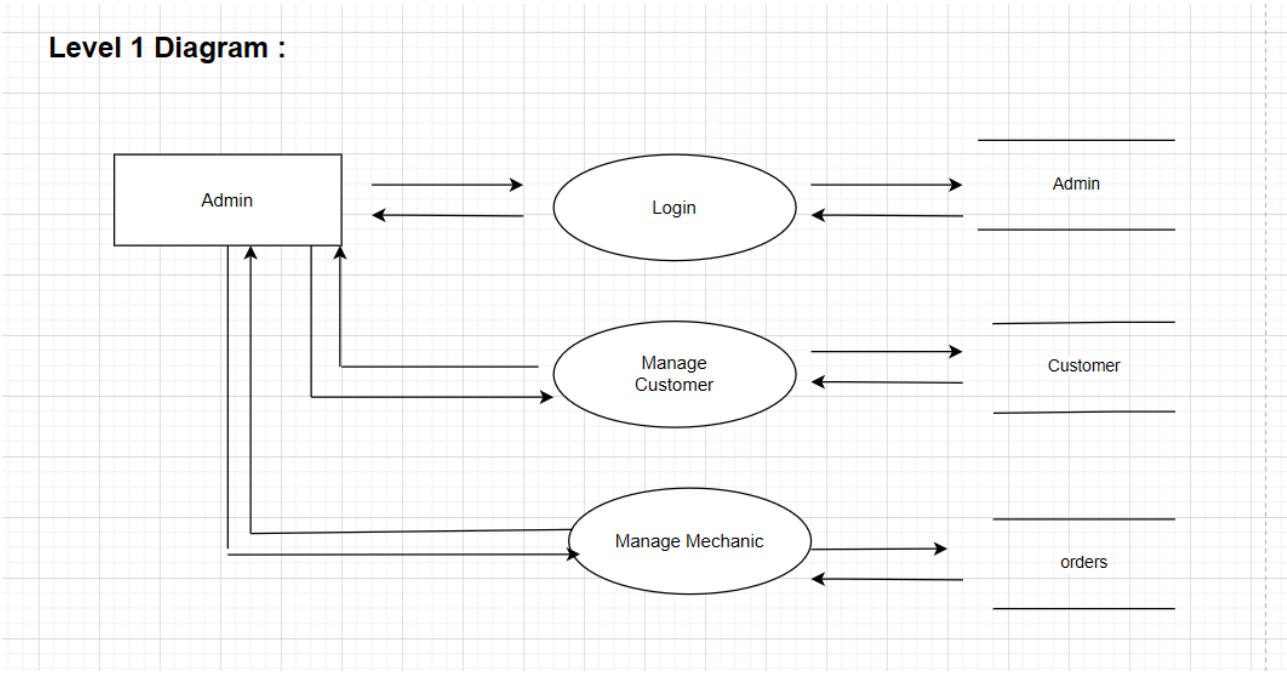
Column Name	Data Type	Description
Technician _id	Int(10)	Unique identifier for each Technician
Name	Varchar(100)	Describe the name of Technician
Gender	Varchar(50)	Describe the gender of user
Phone_no	Int(10)	Describe the phone number of Technician
Email	Varchar(100)	Technician‘s email address
Address	Varchar(500)	Technician 's address
Skills	Varchar(500)	Describe Technician skills
Salary	Int(20)	Salary of the Technician

5 Diagrams

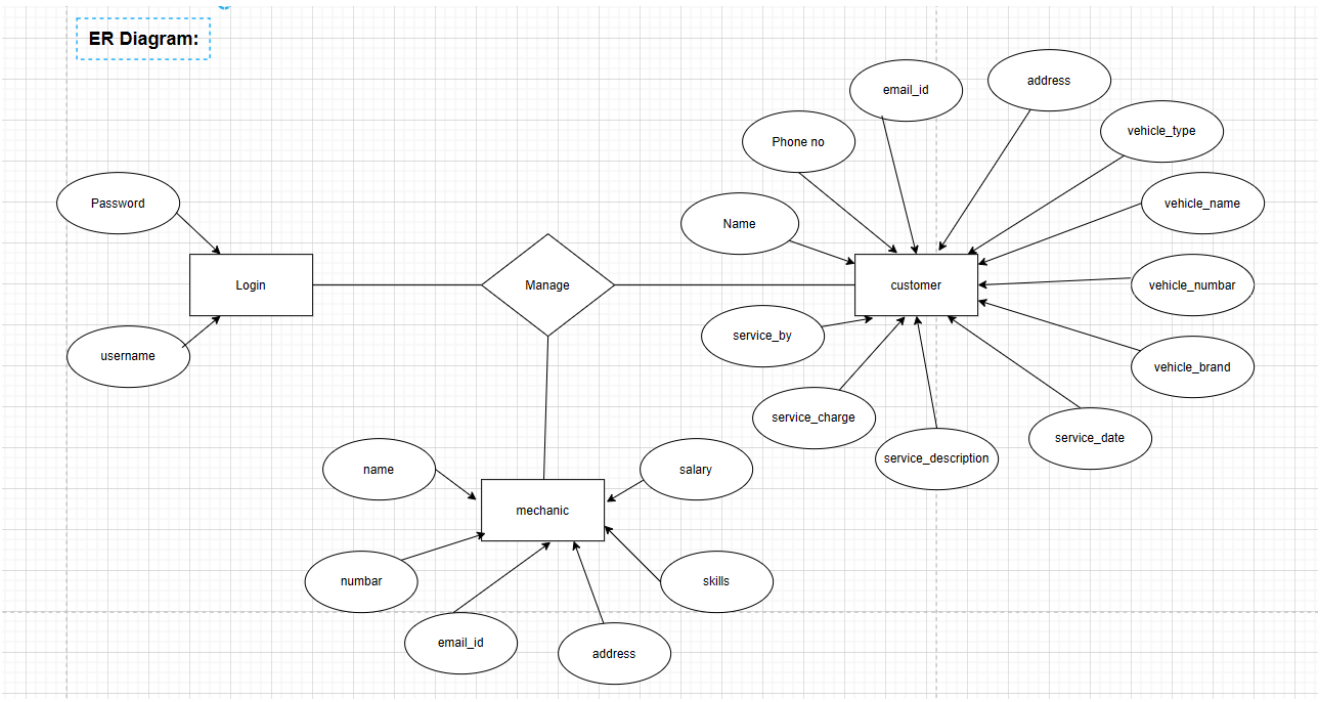
5.1 Context Leval(Leval-o)



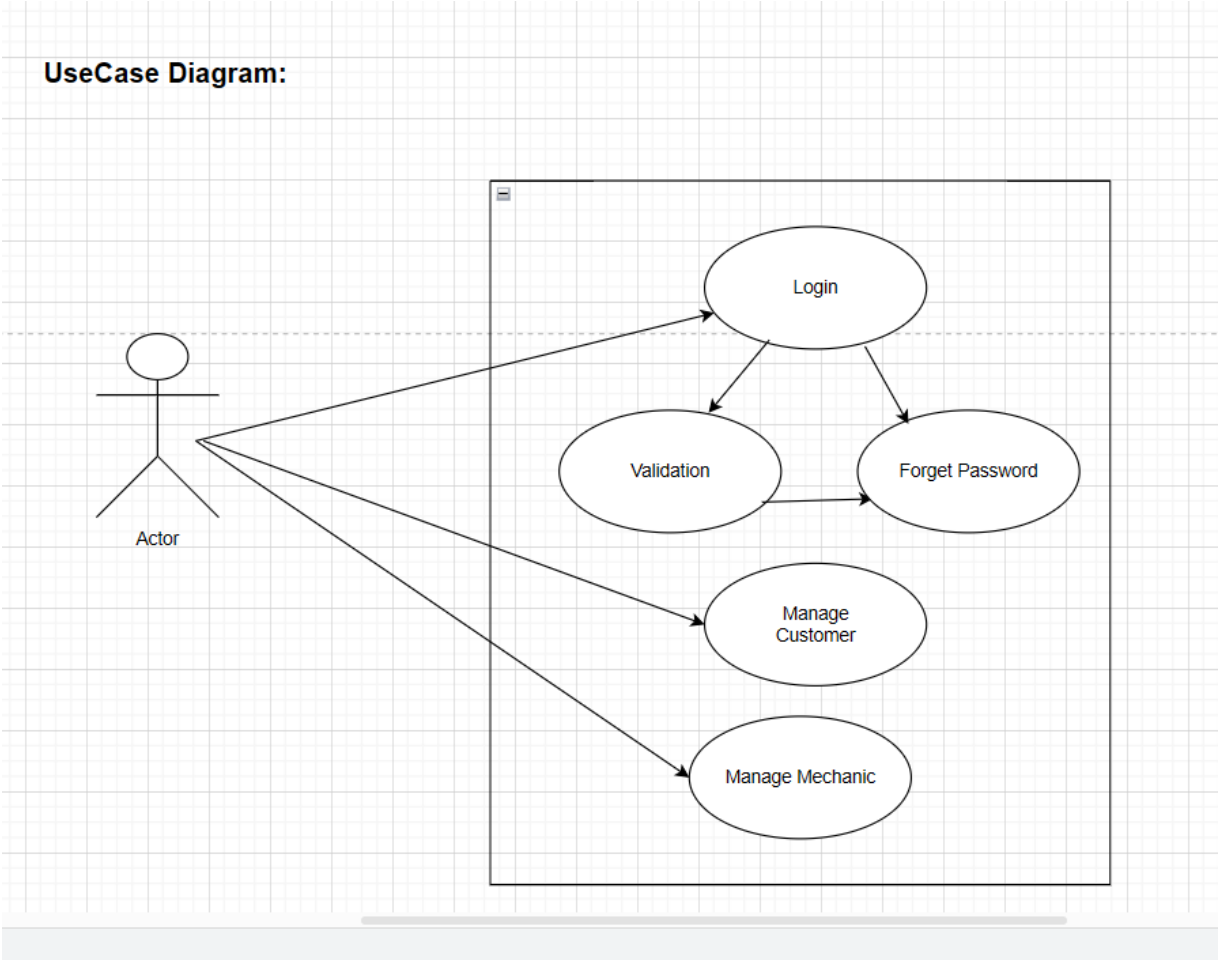
5.2 Leval-1 Diagram



5.3 ER Diagram



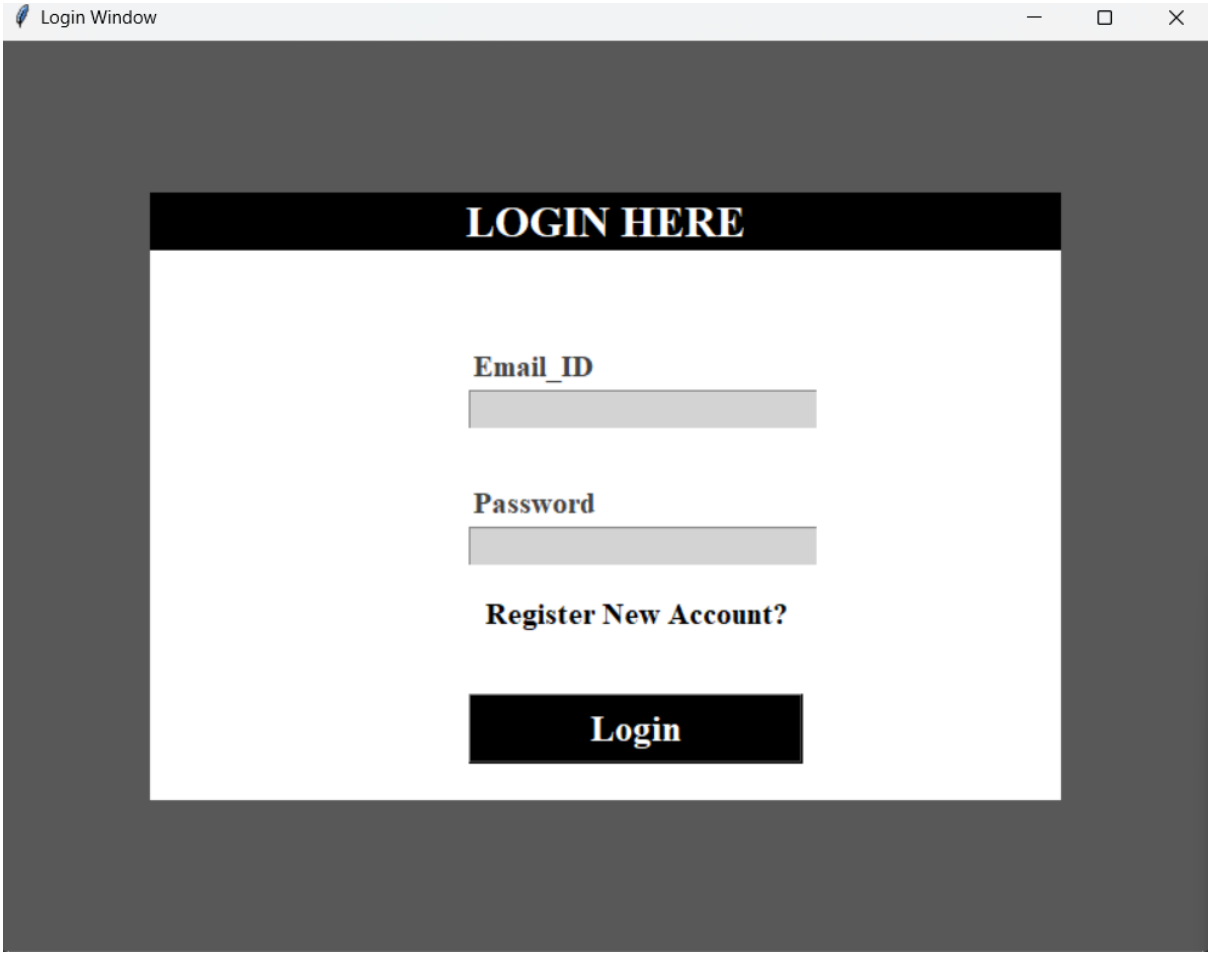
5.4 Use Case Diagram



6 User Interface

6.1 Admin Panel

6.1.1 Login page



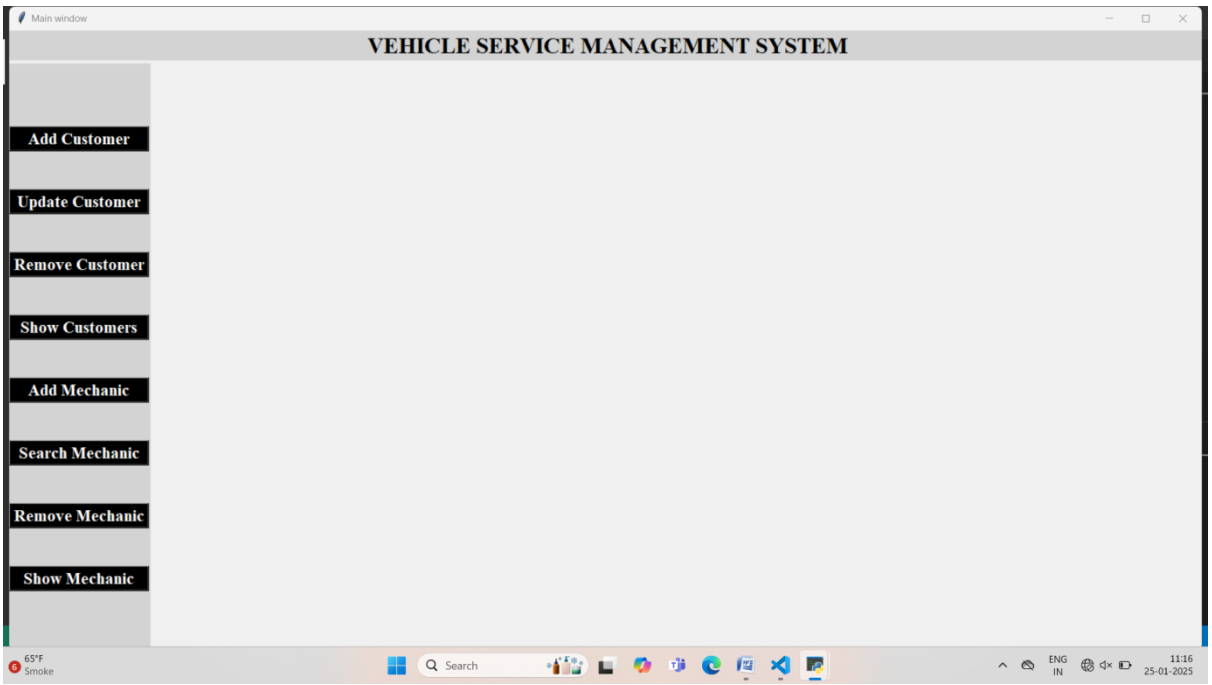
Description:This picture show login form for vehicle service management system.

```
#loginform
class mylogin:
```

```
     def __init__(self,w):

          w.title("Login Window")
          w.geometry("800x600+350+100")
          w.configure(bg="#585858")

          frame1=Frame(w,bg="white")
          frame1.place(x=100,y=100,width=600,height=400)

          title=Label(frame1,text="LOGIN HERE",font=("times",22,"bold"),bg="black",fg="white")
          title.place(x=0,y=0,width=600)
          f_name=Label(frame1,text="Email_ID",font=("times",15,"bold"),bg="white",fg="#404040")
          f_name.place(x=210,y=100)
          a=StringVar()
          b=StringVar()
          txt_name=Entry(frame1,font=("times",15),bg="lightgray",textvariable=a)
          txt_name.place(x=210,y=130,width=230)
          password=Label(frame1,text="Password",font=("times",15,"bold"),bg="white",fg="#404040")
          password.place(x=210,y=190)
          txt_passw=Entry(frame1,font=("times",15),bg="lightgray",textvariable=b)
          txt_passw.place(x=210,y=220,width=230)
# function to check valid email
          def validate_email(txt_name):
            email_pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
            if re.match(email_pattern, txt_name):
              return True
            return False

          def fun():

           c=a.get()
           d=b.get()


           if c=="" or d==="":
             messagebox.showerror("error","please enter username and password")
           email = txt_name.get()
           if not validate_email(email):
                messagebox.showerror("Invalid Input",
                              "Email must contain '@' and '.' characters.")

           else:

                     mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                )
                     mycursor=mydb.cursor()
                     cmd=f"select a_username,a_password from login where a_username=%s and a_password=%s"
                     mycursor.execute(cmd,(c,d))
                     myresult=mycursor.fetchone()
                     mydb.commit()
                     if __name__ == "__main__":
                       if myresult:
                            w2=Tk()
                            obj2=mainform(w2)
                            w2.mainloop()
                            w.destroy()
                       else:
                                          messagebox.showerror("error","please enter valid username and
password")
          def regi():
           if btnlog.selection_get:
               register()
               w.destroy()


          btnlog=Button(frame1,text="Register New
Account?",command=regi,activebackground="black",activeforeground="white",cursor="hand2",width=18,font=("times",1
5,"bold"),fg="black",bg="white",bd=0)
          btnlog.place(x=210,y=260)
```

```
btnlog2=Button(frame1,text="Login",command=fun,activebackground="white",activeforeground="black",width=15,font=(
"times",18,"bold"),bg="black",fg="white",cursor="hand2")
          btnlog2.place(x=210,y=330)

#registerform
def register():
        w1=Tk()
        w1.title("Register Window")
        w1.geometry("800x600+350+100")
        w1.configure(bg="#585858")

        frame2=Frame(w1,bg="white")
        frame2.place(x=100,y=100,width=600,height=400)
        title=Label(frame2,text="REGISTER HERE",font=("times",22,"bold"),bg="black",fg="white")
        title.place(x=0,y=0,width=600)

        f_name=Label(frame2,text="Email_ID",font=("times",15,"bold"),bg="white",fg="#404040")
        f_name.place(x=210,y=100)
        a1=StringVar()
        b1=StringVar()
        txt_name1=Entry(frame2,font=("times",15),bg="lightgray",textvariable=a1)
        txt_name1.place(x=210,y=130,width=230)

        password=Label(frame2,text="Password",font=("times",15,"bold"),bg="white",fg="#404040")
        password.place(x=210,y=190)
        txt_passw1=Entry(frame2,font=("times",15),bg="lightgray",textvariable=b1)
        txt_passw1.place(x=210,y=220,width=230)
# function to check valid email
        def validate_email(txt_name1):
            email_pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
            if re.match(email_pattern, txt_name1):
              return True
            return False

        def registerdemo():
           u = txt_name1.get()
           p = txt_passw1.get()
           if u=="" or p==="":
                 messagebox.showerror("error","please enter username and password")
           email = txt_name1.get()
           if not validate_email(email):
                messagebox.showerror("Invalid Input",
                              "Email must contain '@' and '.' characters.")

           else:

                 mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                 )
                 mycursor=mydb.cursor()
                 sqlinsert=f"insert into login values('','{txt_name1.get()}','{txt_passw1.get()}')"
                 mycursor.execute(sqlinsert)
                 mydb.commit()
                 messagebox.showinfo("valid","success")



btn1=Button(frame2,text="Create",command=registerdemo,activebackground="white",activeforeground="black",width=15
,font=("times",18,"bold"),bg="black",fg="white",cursor="hand2")
        btn1.place(x=210,y=300)
```

6.1.2 Home Page

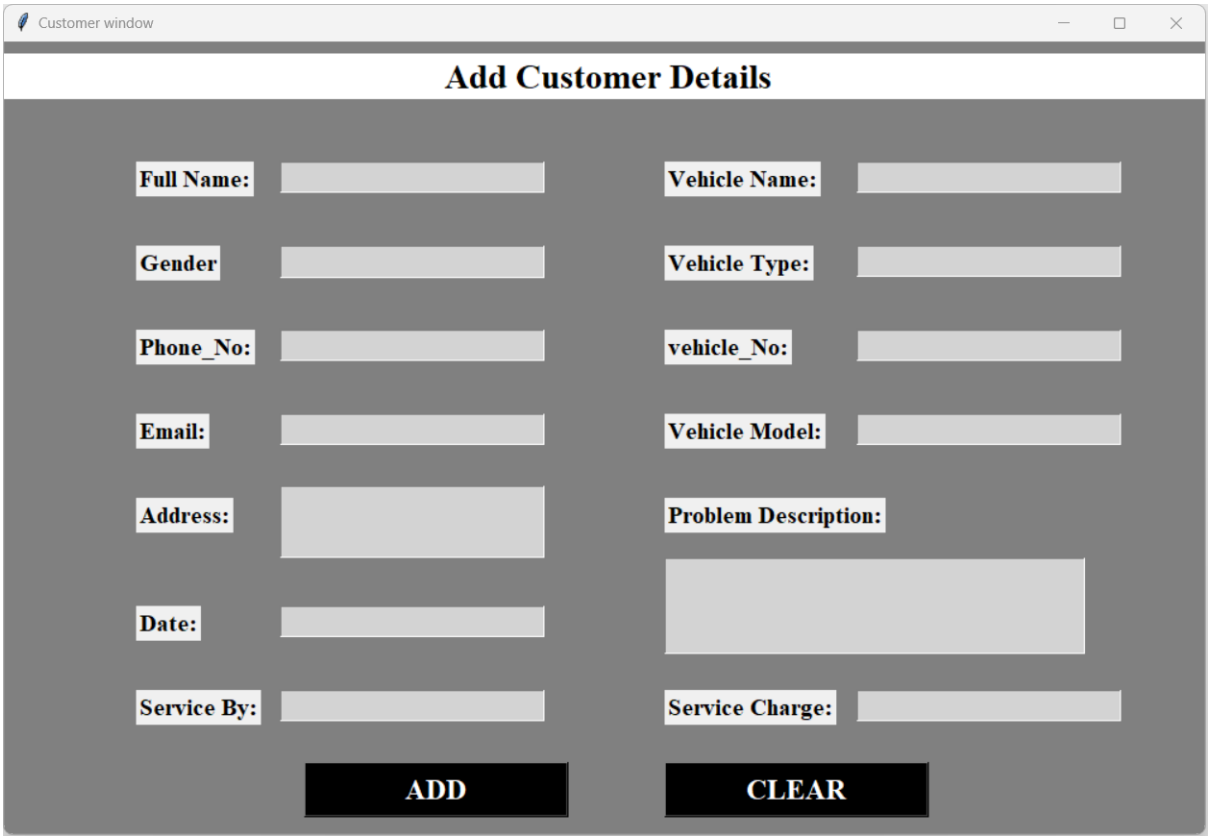


Description:This picture show Main form for vehicle service management system

```
class mainform:
    def __init__(self,w2):
```

```
        w2.title("Main window")
        w2.geometry("1520x790+0+0")
       # w2.configure(bg="#585858")
        title=Label(w2,text=" VEHICLE SERVICE MANAGEMENT SYSTEM",font=("times",22,"bold"),bg="lightgray")
        title.place(x=0,y=0,width=1520)
      # img1=Image.open(r"C:\Users\varun\OneDrive\Documents\payal\image\11.jpg")
      # photoimg1=ImageTk.PhotoImage(img1)
      # lablimg=Label(w2,image=photoimg1)
      # lablimg.place(x=0,y=40)
        frame1=Frame(w2,bg="lightgray",height=800)
        frame1.place(x=0,y=42,width=180)
        def addcustomer():
             new_window=Toplevel(w)
             app=cust_win(new_window)
        def addmech():
             new_window=Toplevel(w)
             app=mech_win(new_window)
        def customsearch():
             new_window=Toplevel(w)
             app=cust_search(new_window)
        def mechsearch():
             new_window=Toplevel(w)
             app=mech_search(new_window)
        def customremove():
             new_window=Toplevel(w)
             app=cust_remove(new_window)
        def mechremove():
             new_window=Toplevel(w)
             app=mech_remove(new_window)
        def customerall():
             new_window=Toplevel(w)
             app=all_cust(new_window)
        def mechall():
             new_window=Toplevel(w)
             app=all_mech(new_window)

        btncus=Button(frame1,text="Add
Customer",command=addcustomer,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold"
),bg="black",fg="white",cursor="hand2")
        btncus.place(x=0,y=80,height=32)
        btnsercus=Button(frame1,text="Update
Customer",command=customsearch,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold
"),bg="black",fg="white",cursor="hand2")
        btnsercus.place(x=0,y=160,height=32)
        btndelcus=Button(frame1,text="Remove
Customer",command=customremove,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold
"),bg="black",fg="white",cursor="hand2")
        btndelcus.place(x=0,y=240,height=32)
        btnallcus=Button(frame1,text="Show
Customers",command=customerall,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold
"),bg="black",fg="white",cursor="hand2")
        btnallcus.place(x=0,y=320,height=32)

        btnmec=Button(frame1,text="Add
Mechanic",command=addmech,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold"),bg
="black",fg="white",cursor="hand2")
        btnmec.place(x=0,y=400,height=32)
        btnsermec=Button(frame1,text="Search
Mechanic",command=mechsearch,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold")
,bg="black",fg="white",cursor="hand2")
        btnsermec.place(x=0,y=480,height=32)
        btndelmec=Button(frame1,text="Remove
Mechanic",command=mechremove,activebackground="white",activeforeground="black",width=14,font=("times",16,"bold")
,bg="black",fg="white",cursor="hand2")
        btndelmec.place(x=0,y=560,height=32)
if __name__ == "__main__":
      w=Tk()
      obj=mylogin(w)
      w.mainloop()
```

6.1.3 Add Customer

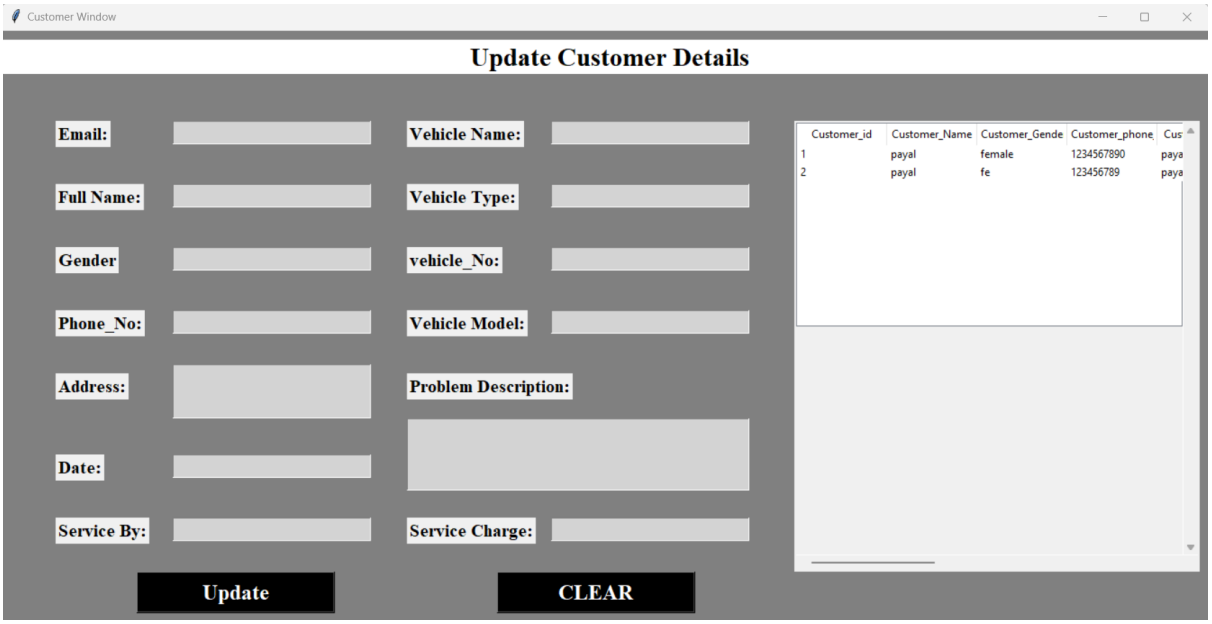


Description:This picture show Add customer form for vehicle service management system

```python
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
from tkinter import ttk
import re
class cust_win:
     def __init__(self,w):

          w.title("Customer window")
          w.geometry("1000x660+230+100")
          w.config(bg="#808080")
          title=Label(w,text=" Add Customer Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1000)

          nm=StringVar()
          b=StringVar()
          pn=StringVar()
          em=StringVar()
          da=StringVar()
          ty=StringVar()
          snm=StringVar()
          vn=StringVar()
          vmo=StringVar()
          sc=StringVar()
          gender_var=StringVar()
          def validate_name(nm):
             if nm.isalpha():
               return True
             return False
# function to check valid phone number
          def validate_phone(txt_phone):
               if txt_phone.isdigit() and len(txt_phone) == 10:
                    return True
               return False
# function to check valid email
          def validate_email(em):
            email_pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
            if re.match(email_pattern, em):
              return True
            return False



          f_name=Label(w,text="Full Name:",font=("times",15,"bold"))
          f_name.place(x=110,y=100)
          txt_name1=Entry(w,font=("times",15),bg="lightgray",textvariable=nm)
          txt_name1.place(x=230,y=100,width=220)

          gender=Label(w,text="Gender",font=("times",15,"bold"))
          gender.place(x=110,y=170)
          radiomale=Entry(w,font=("times",15,"bold"),bg="lightgray")
          radiomale.place(x=230,y=170,width=220)

          phone=Label(w,text="Phone_No:",font=("times",15,"bold"))
          phone.place(x=110,y=240)
          txt_phone=Entry(w,font=("times",15),bg="lightgray",textvariable=pn)
          txt_phone.place(x=230,y=240,width=220)

          email=Label(w,text="Email:",font=("times",15,"bold"))
          email.place(x=110,y=310)
          txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
          txt_email.place(x=230,y=310,width=220)

          addr=Label(w,text="Address:",font=("times",15,"bold"))
          addr.place(x=110,y=380)
          txt_addr=Entry(w,font=("times",15),bg="lightgray")
          txt_addr.place(x=230,y=370,width=220,height=60)

          date=Label(w,text="Date:",font=("times",15,"bold"))
          date.place(x=110,y=470)
          txt_date=Entry(w,font=("times",15),bg="lightgray",textvariable=da)
          txt_date.place(x=230,y=470,width=220)

          s_name=Label(w,text="Service By:",font=("times",15,"bold"))
          s_name.place(x=110,y=540)
          txt_names=Entry(w,font=("times",15),bg="lightgray",textvariable=snm)
          txt_names.place(x=230,y=540,width=220)

          v_name=Label(w,text="Vehicle Name:",font=("times",15,"bold"))
          v_name.place(x=550,y=100)
          txt_namev=Entry(w,font=("times",15),bg="lightgray",textvariable=ty)
          txt_namev.place(x=710,y=100,width=220)

          vtype=Label(w,text="Vehicle Type:",font=("times",15,"bold"))
```

```
          vtype.place(x=550,y=170)
          txt_vtype=Entry(w,font=("times",15),bg="lightgray")
          txt_vtype.place(x=710,y=170,width=220)

          vnumber=Label(w,text="vehicle_No:",font=("times",15,"bold"))
          vnumber.place(x=550,y=240)
          txt_vnumber=Entry(w,font=("times",15),bg="lightgray",textvariable=vn)
          txt_vnumber.place(x=710,y=240,width=220)

          vmodel=Label(w,text="Vehicle Model:",font=("times",15,"bold"))
          vmodel.place(x=550,y=310)
          txt_vmodel=Entry(w,font=("times",15),bg="lightgray",textvariable=vmo)
          txt_vmodel.place(x=710,y=310,width=220)

          scharge=Label(w,text="Service Charge:",font=("times",15,"bold"))
          scharge.place(x=550,y=540)
          txt_scharge=Entry(w,font=("times",15),bg="lightgray",textvariable=sc)
          txt_scharge.place(x=710,y=540,width=220)

          problem=Label(w,text="Problem Description:",font=("times",15,"bold"))
          problem.place(x=550,y=380)
          txt_prob=Entry(w,font=("times",15),bg="lightgray")
          txt_prob.place(x=550,y=430,width=350,height=80)

# function called when Submit is clicked
          def on_submit():
             if nm.get()==""  or pn.get()=="" or em.get()=="" or da.get()=="" or snm.get()=="" or ty.get()=="" or vn.get()=="" or
vmo.get()=="" or sc.get()=="" :
              messagebox.showerror("error","Please valid infomation")
             name = nm.get()
             if not validate_name(name):
              messagebox.showerror("Invalid Input",
                             "Name must contain only alphabets.")
             phone=txt_phone.get()
             if not validate_phone(phone):
                   messagebox.showerror("","Inavlid Phone Number,only 10 digit enter")
             email = em.get()
             if not validate_email(email):
                messagebox.showerror("Invalid Input",
                             "Email must contain '@' and '.' characters.")

             if not re.match("^\\d+$",da.get()):
                   messagebox.showerror("","Inavlid date")
             if not re.match("^\\d+$",vn.get()):
                   messagebox.showerror("","Inavlid vehicle number")
             if not re.match("^\\d+$",sc.get()):
                   messagebox.showerror("","Inavlid service charge")


             else:


                 mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                 )
                 mycursor=mydb.cursor()
                 sql=f"insert into customer
(c_id,c_name,c_gender,c_phone,c_email,c_address,v_type,v_name,v_number,v_brand,s_date,s_description,s_by,s_cost)
values('','{txt_name1.get()}','{radiomale.get()}',{txt_phone.get()},'{txt_email.get()}','{txt_addr.get()}','{txt_vtype.get()}','{t
xt_namev.get()}',{txt_vnumber.get()},'{txt_vmodel.get()}',{txt_date.get()},'{txt_prob.get()}','{txt_names.get()}',{txt_scharge.get
()})"

                 mycursor.execute(sql)
                 mydb.commit()
                 messagebox.showinfo("valid","successfully added customer's data")




btn1=Button(w,text="ADD",activebackground="white",command=on_submit,activeforeground="black",width=15,font=("times",18,"bold"),bg=
"black",fg="white",cursor="hand2")
          btn1.place(x=250,y=600)

btn2=Button(w,text="CLEAR",activebackground="white",activeforeground="black",width=15,font=("times",18,"bold"),bg="black",fg="whit
e",cursor="hand2")
          btn2.place(x=550,y=600)

if __name__ == "__main__":
     w=Tk()
     obj=cust_win(w)
     w.mainloop()
```

6.1.4 Update Customer



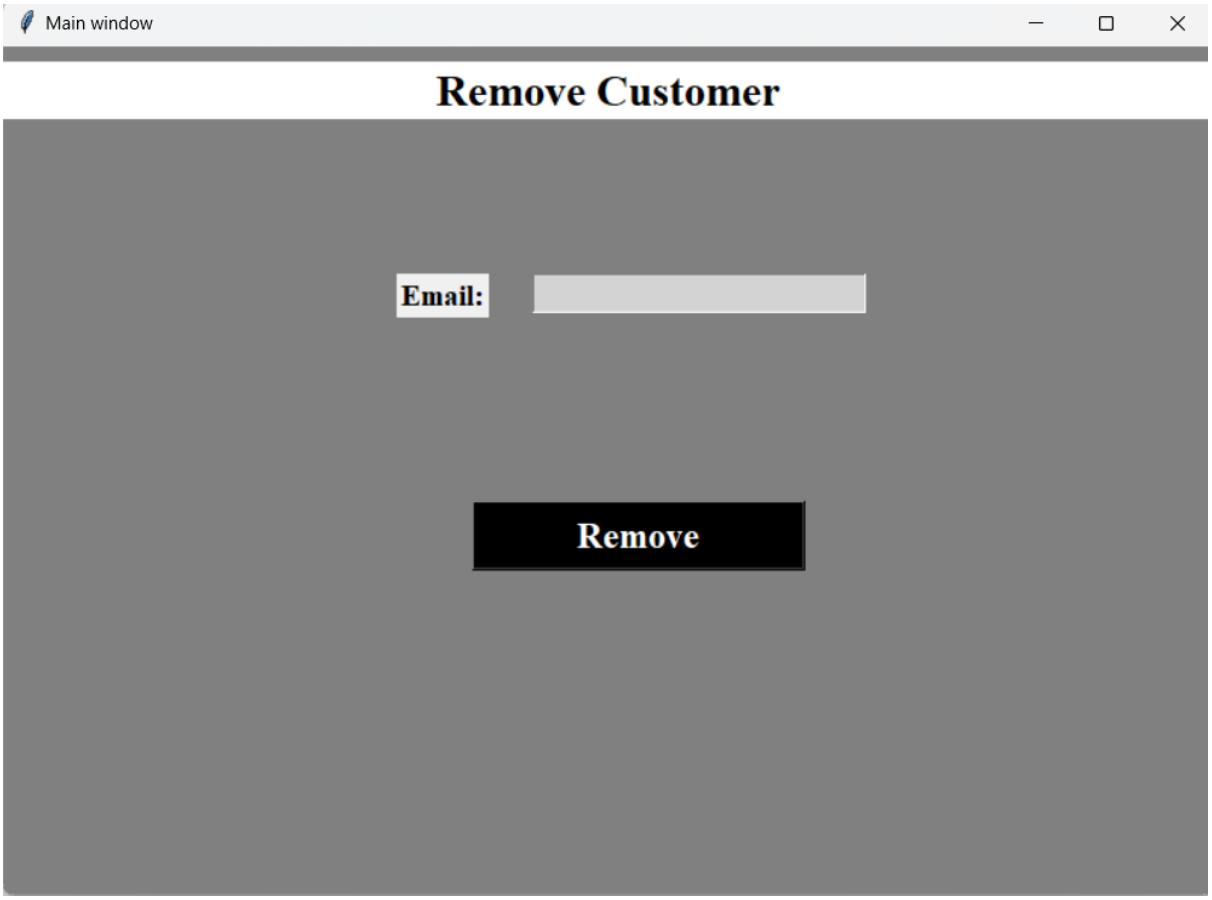
Description:This picture show Update Customer form for vehicle service management system

```
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
from tkinter import ttk
class cust_search:
     def __init__(self,w):

          w.title("Customer Window")
          w.geometry("1340x660+160+100")
          w.config(bg="#808080")
          title=Label(w,text="Update Customer Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1350)

          nm=StringVar()
          b=StringVar()
          pn=StringVar()
          em=StringVar()
          da=StringVar()
          ty=StringVar()
          snm=StringVar()
          vn=StringVar()
          vmo=StringVar()
          sc=StringVar()
          f_name=Label(w,text="Full Name:",font=("times",15,"bold"))
          f_name.place(x=60,y=170)
          txt_name1=Entry(w,font=("times",15),bg="lightgray",textvariable=nm)
          txt_name1.place(x=190,y=170,width=220)

          gender=Label(w,text="Gender",font=("times",15,"bold"))
          gender.place(x=60,y=240)
          txt_gender=Entry(w,font=("times",15),bg="lightgray",textvariable=b)
          txt_gender.place(x=190,y=240,width=220)

          phone=Label(w,text="Phone_No:",font=("times",15,"bold"))
          phone.place(x=60,y=310)
          txt_phone=Entry(w,font=("times",15),bg="lightgray",textvariable=pn)
          txt_phone.place(x=190,y=310,width=220)

          email=Label(w,text="Email:",font=("times",15,"bold"))
          email.place(x=60,y=100)
          txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
          txt_email.place(x=190,y=100,width=220)

          addr=Label(w,text="Address:",font=("times",15,"bold"))
          addr.place(x=60,y=380)
          txt_addr=Entry(w,font=("times",15),bg="lightgray")
          txt_addr.place(x=190,y=370,width=220,height=60)

          date=Label(w,text="Date:",font=("times",15,"bold"))
          date.place(x=60,y=470)
          txt_date=Entry(w,font=("times",15),bg="lightgray",textvariable=da)
          txt_date.place(x=190,y=470,width=220)

          s_name=Label(w,text="Service By:",font=("times",15,"bold"))
          s_name.place(x=60,y=540)
          txt_names=Entry(w,font=("times",15),bg="lightgray",textvariable=snm)
          txt_names.place(x=190,y=540,width=220)

          v_name=Label(w,text="Vehicle Name:",font=("times",15,"bold"))
          v_name.place(x=450,y=100)
          txt_namev=Entry(w,font=("times",15),bg="lightgray",textvariable=ty)
          txt_namev.place(x=610,y=100,width=220)

          vtype=Label(w,text="Vehicle Type:",font=("times",15,"bold"))
          vtype.place(x=450,y=170)
          txt_vtype=Entry(w,font=("times",15),bg="lightgray")
          txt_vtype.place(x=610,y=170,width=220)

          vnumber=Label(w,text="vehicle_No:",font=("times",15,"bold"))
          vnumber.place(x=450,y=240)
          txt_vnumber=Entry(w,font=("times",15),bg="lightgray",textvariable=vn)
          txt_vnumber.place(x=610,y=240,width=220)

          vmodel=Label(w,text="Vehicle Model:",font=("times",15,"bold"))
          vmodel.place(x=450,y=310)
          txt_vmodel=Entry(w,font=("times",15),bg="lightgray",textvariable=vmo)
          txt_vmodel.place(x=610,y=310,width=220)

          scharge=Label(w,text="Service Charge:",font=("times",15,"bold"))
          scharge.place(x=450,y=540)
          txt_scharge=Entry(w,font=("times",15),bg="lightgray",textvariable=sc)
          txt_scharge.place(x=610,y=540,width=220)

          problem=Label(w,text="Problem Description:",font=("times",15,"bold"))
```

```
        problem.place(x=450,y=380)
        txt_prob=Entry(w,font=("times",15),bg="lightgray")
        txt_prob.place(x=450,y=430,width=380,height=80)

        details_f=Frame(w,bd=2)
        details_f.place(x=880,y=100,width=450,height=500)
        scroll_x=ttk.Scrollbar(details_f,orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar(details_f,orient=VERTICAL)

cust_table=ttk.Treeview(details_f,columns=("c_id","c_name","c_gender","c_phone","c_email","c_address","v_type","v_name","v_number"
,"v_brand","s_date","s_description","s_by","s_cost"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)
        scroll_x.config(command=cust_table.xview)
        scroll_y.config(command=cust_table.yview)
        cust_table.heading("c_id",text="Customer_id")
        cust_table.heading("c_name",text="Customer_Name")
        cust_table.heading("c_gender",text="Customer_Gender")
        cust_table.heading("c_phone",text="Customer_phone_no")
        cust_table.heading("c_email",text="Customer_email")
        cust_table.heading("c_address",text="Customer_Address")
        cust_table.heading("v_type",text="vehicle_type")
        cust_table.heading("v_name",text="vehicle_Name")
        cust_table.heading("v_number",text="vehicle_Number")
        cust_table.heading("v_brand",text="vehicle_Brand")
        cust_table.heading("s_date",text="service_date")
        cust_table.heading("s_description",text="probelm")
        cust_table.heading("s_by",text="Sarvice_By")
        cust_table.heading("s_cost",text="Service_cost")

        cust_table["show"]="headings"

        cust_table.column("c_id",width=100)
        cust_table.column("c_name",width=100)
        cust_table.column("c_gender",width=100)
        cust_table.column("c_phone",width=100)
        cust_table.column("c_email",width=100)
        cust_table.column("c_address",width=100)
        cust_table.column("v_type",width=100)
        cust_table.column("v_name",width=100)
        cust_table.column("v_number",width=100)
        cust_table.column("v_brand",width=100)
        cust_table.column("s_date",width=100)
        cust_table.column("s_description",width=100)
        cust_table.column("s_by",width=100)
        cust_table.column("s_cost",width=100)

        cust_table.pack(fill=BOTH)
        cust_table.bind("<ButtonRelease-1>")


        mydb=mysql.connector.connect(
                host="localhost",
                user="root",
                password="",
                database="vehicle"
            )
        mycursor=mydb.cursor()
        mycursor.execute("select * from customer")
        myresult=mycursor.fetchall()
        if len(myresult)!=0:

                for i in myresult:
                    cust_table.insert("",END,values=i)

                mydb.commit()
                mydb.close()
        def get_cursor(event=""):
            cursor_row=cust_table.focus()
            content=cust_table.item(cursor_row)
            row=content["values"]

            txt_name1.set(row[1])
            txt_gender.set(row[2])
            txt_phone.set(row[3])
            txt_email.set(row[4])
            txt_addr.set(row[5])
            txt_date.set(row[6])
            txt_names.set(row[7])
            txt_namev.set(row[8])
            txt_vtype.set(row[9])
            txt_vnumber.set(row[10])
            txt_vmodel.set(row[11])
            txt_prob.set(row[12])
            txt_scharge.set(row[13])
```

```
        def check():
            if nm.get()=="" or b.get()=="" or pn.get()=="" or em.get()=="" or da.get()=="" or snm.get()=="" or ty.get()=="" or
vn.get()=="" or vmo.get()=="" or sc.get()=="" :
                messagebox.showerror("error","Please valid infomation")
            else:

              mydb=mysql.connector.connect(
                  host="localhost",
                  user="root",
                  password="",
                  database="vehicle"
              )
              mycursor=mydb.cursor()
              sql=f"update  customer set
c_name='{txt_name1.get()}',c_gender='{txt_gender.get()}',c_phone={txt_phone.get()},c_address='{txt_addr.get()}',v_type='{txt_vtype
.get()}',v_name='{txt_namev.get()}',v_number={txt_vnumber.get()},v_brand='{txt_vmodel.get()}',s_date={txt_date.get()},s_descriptio
n='{txt_prob.get()}',s_by='{txt_names.get()}',s_cost={txt_scharge.get()} where c_email='{txt_email.get()}'"

              mycursor.execute(sql)
              mydb.commit()
              messagebox.showinfo("valid","successfully updated customer's data")


btn3=Button(w,text="Update",activebackground="white",command=check,activeforeground="black",width=15,font=("times",18,"bold"),bg="
black",fg="white",cursor="hand2")
        btn3.place(x=150,y=600)


btn2=Button(w,text="CLEAR",activebackground="white",activeforeground="black",width=15,font=("times",18,"bold"),bg="black",fg="whit
e",cursor="hand2")
        btn2.place(x=550,y=600)
if __name__ == "__main__":
    w=Tk()
    obj=cust_search(w)
    w.mainloop()
```

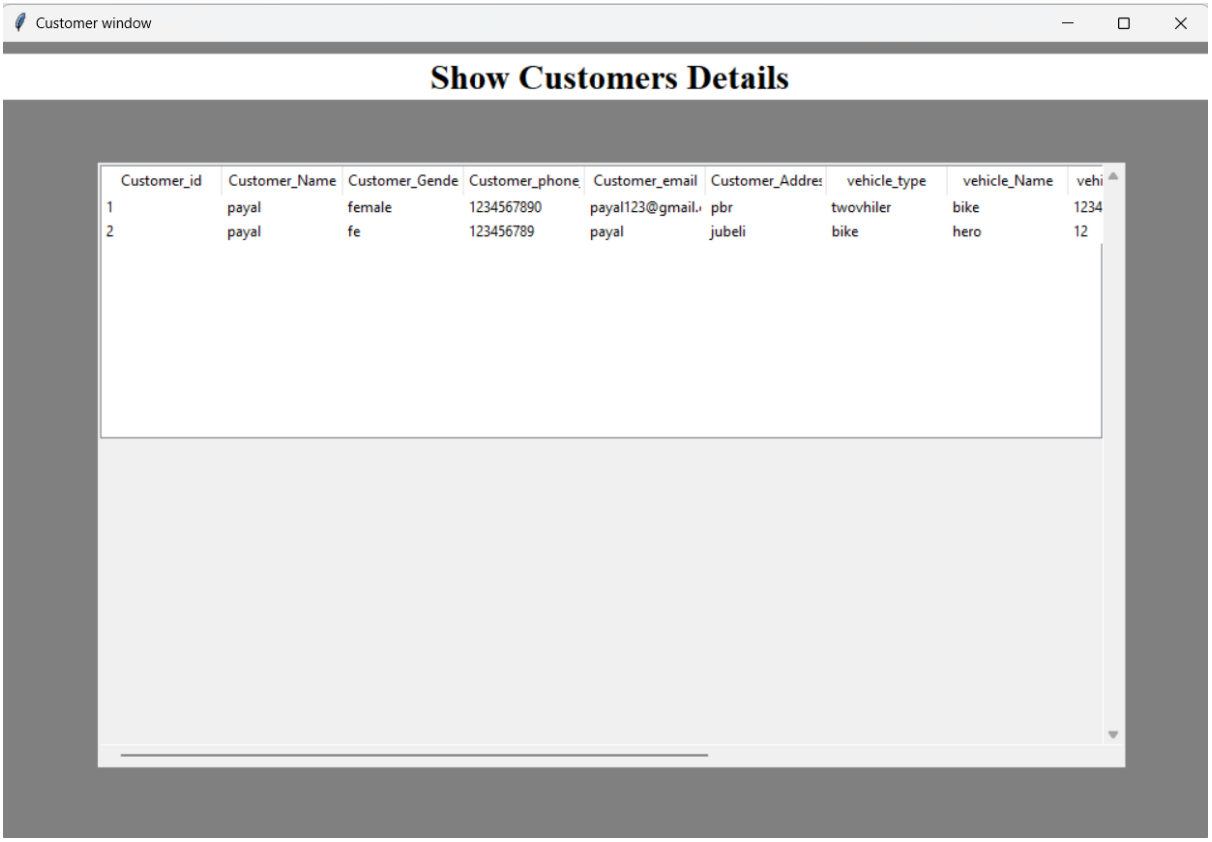
6.1.5 Remove Customer



Description:This picture show Remove cutomer form for vehicle service management system

```python
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector

class cust_remove:
     def __init__(self,w):

          w.title("Main window")
          w.geometry("800x560+380+100")
          w.config(bg="#808080")
          title=Label(w,text=" Remove Customer ",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=800)


          em=StringVar()


          email=Label(w,text="Email:",font=("times",15,"bold"))
          email.place(x=260,y=150)
          txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
          txt_email.place(x=350,y=150,width=220)


          def check():
               if  em.get()==""  :
                    messagebox.showerror("error","Please valid infomation")
               else:
                    mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                  )
                    mycursor=mydb.cursor()
                    sql="delete  from customer where c_email=%s"
                    value=(txt_email.get(),)
                    mycursor.execute(sql,value)
                    mydb.commit()
                    messagebox.showinfo("valid","successfully deleted customer's data")


btn3=Button(w,text="Remove",activebackground="white",command=check,activeforeground="black",width=15,font=("times",18,"b
old"),bg="black",fg="white",cursor="hand2")
          btn3.place(x=310,y=300)

if __name__ == "__main__":
     w=Tk()
     obj=cust_remove(w)
     w.mainloop()
```

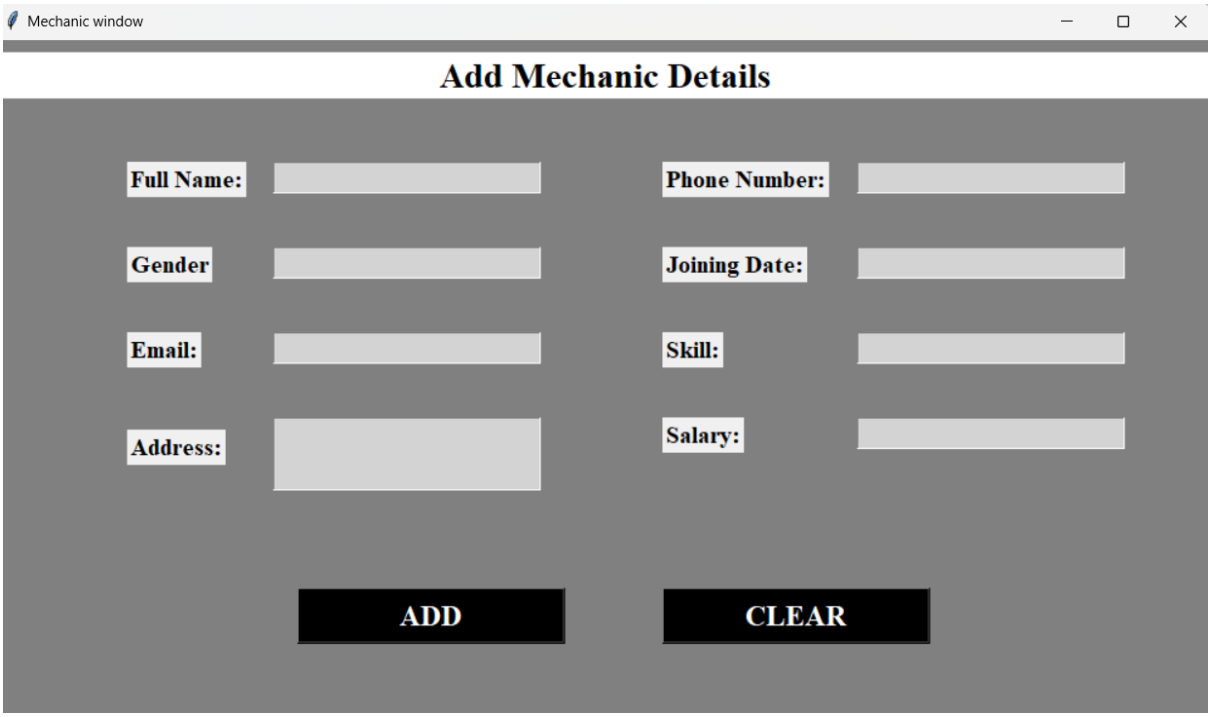
6.1.6 Show Customer



Description:This picture show display all customer's details form for vehicle service management system

```
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
from tkinter import ttk
class all_cust:
     def __init__(self,w):
          w.title("Customer window")
          w.geometry("1000x660+230+100")
          w.config(bg="#808080")
          title=Label(w,text=" Show Customers Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1000)
          details_f=Frame(w,bd=2)
          details_f.place(x=80,y=100,width=850,height=500)
          scroll_x=ttk.Scrollbar(details_f,orient=HORIZONTAL)
          scroll_y=ttk.Scrollbar(details_f,orient=VERTICAL)

cust_table=ttk.Treeview(details_f,columns=("c_id","c_name","c_gender","c_phone","c_email","c_address","v_type","v_name",
"v_number","v_brand","s_date","s_description","s_by","s_cost"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
          scroll_x.pack(side=BOTTOM,fill=X)
          scroll_y.pack(side=RIGHT,fill=Y)
          scroll_x.config(command=cust_table.xview)
          scroll_y.config(command=cust_table.yview)
          cust_table.heading("c_id",text="Customer_id")
          cust_table.heading("c_name",text="Customer_Name")
          cust_table.heading("c_gender",text="Customer_Gender")
          cust_table.heading("c_phone",text="Customer_phone_no")
          cust_table.heading("c_email",text="Customer_email")
          cust_table.heading("c_address",text="Customer_Address")
          cust_table.heading("v_type",text="vehicle_type")
          cust_table.heading("v_name",text="vehicle_Name")
          cust_table.heading("v_number",text="vehicle_Number")
          cust_table.heading("v_brand",text="vehicle_Brand")
          cust_table.heading("s_date",text="service_date")
          cust_table.heading("s_description",text="probelm")
          cust_table.heading("s_by",text="Sarvice_By")
          cust_table.heading("s_cost",text="Service_cost")

          cust_table["show"]="headings"

          cust_table.column("c_id",width=100)
          cust_table.column("c_name",width=100)
          cust_table.column("c_gender",width=100)
          cust_table.column("c_phone",width=100)
          cust_table.column("c_email",width=100)
          cust_table.column("c_address",width=100)
          cust_table.column("v_type",width=100)
          cust_table.column("v_name",width=100)
          cust_table.column("v_number",width=100)
          cust_table.column("v_brand",width=100)
          cust_table.column("s_date",width=100)
          cust_table.column("s_description",width=100)
          cust_table.column("s_by",width=100)
          cust_table.column("s_cost",width=100)

          cust_table.pack(fill=BOTH)
          mydb=mysql.connector.connect(
                    host="localhost",
                    user="root",
                    password="",
                    database="vehicle"
                )
          mycursor=mydb.cursor()
          mycursor.execute("select * from customer")
          myresult=mycursor.fetchall()
          if len(myresult)!=0:
                         for i in myresult:
                               cust_table.insert("",END,values=i)
                         mydb.commit()
                         mydb.close()

if __name__ == "__main__":
     w=Tk()
     obj=all_cust(w)
     w.mainloop()
```

6.1.7 Add Mechanic



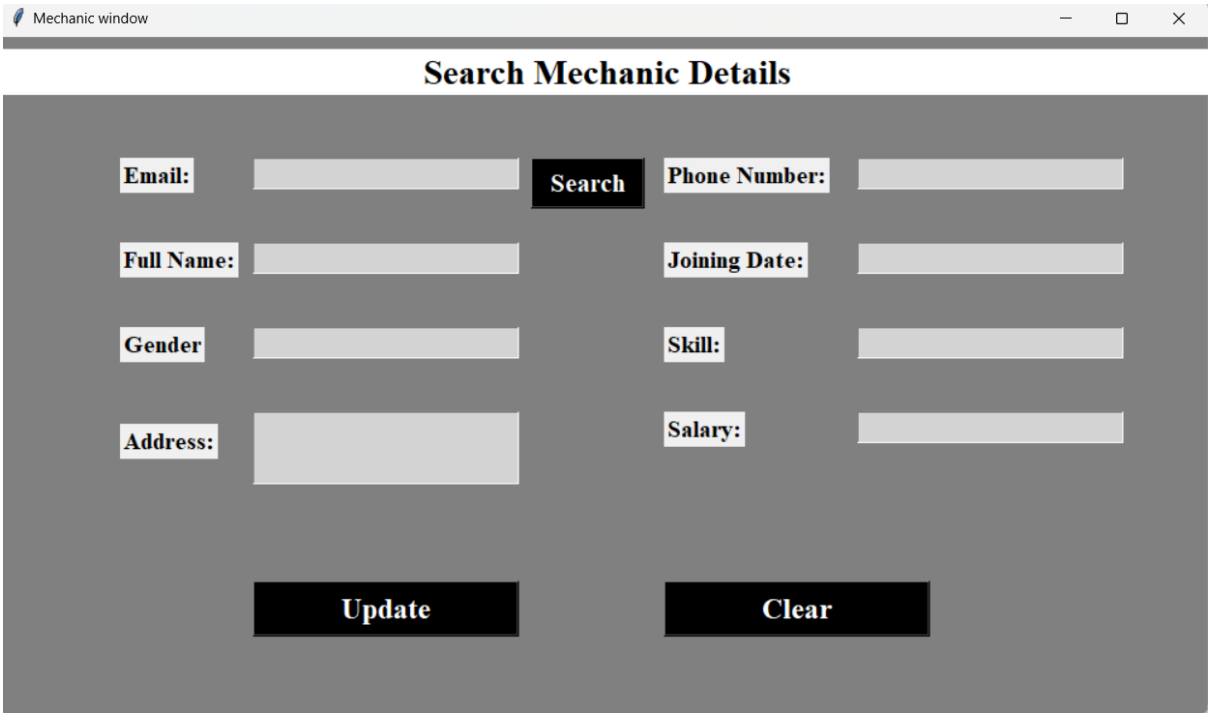
Description:This picture show add mechanic's data form for vehicle service management system

```
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
```

```
import re
class mech_win:
     def __init__(self,w):

          w.title("Mechanic window")
          w.geometry("1000x560+250+140")
          w.config(bg="#808080")
          title=Label(w,text=" Add Mechanic Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1000)

          nm=StringVar()
          b=StringVar()
          pn=StringVar()
          em=StringVar()

          jd=StringVar()

          sk=StringVar()
          sa=StringVar()

          def validate_name(nm):
             if nm.isalpha():
               return True
             return False
# function to check valid phone number
          def validate_phone(txt_phone):
               if txt_phone.isdigit() and len(txt_phone) == 10:
                    return True
               return False
# function to check valid email
          def validate_email(em):
            email_pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
            if re.match(email_pattern, em):
              return True
            return False

# function called when Submit is clicked
          def on_submit():
             if nm.get()=="" or b.get()=="" or pn.get()=="" or em.get()==""  or jd.get()=="" or sk.get()=="" or sa.get()=="" :
                  messagebox.showerror("error","Please valid infomation")

             name = nm.get()
             if not validate_name(name):
              messagebox.showerror("Invalid Input",
                             "Name must contain only alphabets.")
             phone=pn.get()
             if not validate_phone(phone):
                   messagebox.showerror("","Inavlid Phone Number,only 10 digit enter")
             email = em.get()
             if not validate_email(email):
                messagebox.showerror("Invalid Input",
                             "Email must contain '@' and '.' characters.")

             if not re.match("^\\d+$",jd.get()):
                   messagebox.showerror("","Inavlid date")
             if not re.match("^\\d+$",sa.get()):
                   messagebox.showerror("","Inavlid salary")


             else:

                 mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                 )
                 mycursor=mydb.cursor()
                 sql=f"insert into mechanic(t_id,t_name,t_gender,t_phone,t_date,t_email,t_address,t_skills,t_salary)
values('','{txt_name1.get()}','{txt_gender.get()}',{txt_phone.get()},{txt_jdate.get()},'{txt_email.get()}','{txt_addr.get()}','{tx
t_skill.get()}',{txt_salary.get()})"
                 mycursor.execute(sql)
                 mydb.commit()
                 messagebox.showinfo("valid","successfully added Mechanic's data")

          f_name=Label(w,text="Full Name:",font=("times",15,"bold"))
          f_name.place(x=110,y=100)
          txt_name1=Entry(w,font=("times",15),bg="lightgray",textvariable=nm)
          txt_name1.place(x=230,y=100,width=220)

          gender=Label(w,text="Gender",font=("times",15,"bold"))
          gender.place(x=110,y=170)
          txt_gender=Entry(w,font=("times",15),bg="lightgray",textvariable=b)
          txt_gender.place(x=230,y=170,width=220)
```

```
        email=Label(w,text="Email:",font=("times",15,"bold"))
        email.place(x=110,y=240)
        txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
        txt_email.place(x=230,y=240,width=220)

        addr=Label(w,text="Address:",font=("times",15,"bold"))
        addr.place(x=110,y=320)
        txt_addr=Entry(w,font=("times",15),bg="lightgray")
        txt_addr.place(x=230,y=310,width=220,height=60)



        phone=Label(w,text="Phone Number:",font=("times",15,"bold"))
        phone.place(x=550,y=100)
        txt_phone=Entry(w,font=("times",15),bg="lightgray",textvariable=pn)
        txt_phone.place(x=710,y=100,width=220)

        jdate=Label(w,text="Joining Date:",font=("times",15,"bold"))
        jdate.place(x=550,y=170)
        txt_jdate=Entry(w,font=("times",15),bg="lightgray",textvariable=jd)
        txt_jdate.place(x=710,y=170,width=220)

        skill=Label(w,text="Skill:",font=("times",15,"bold"))
        skill.place(x=550,y=240)
        txt_skill=Entry(w,font=("times",15),bg="lightgray",textvariable=sk)
        txt_skill.place(x=710,y=240,width=220)

        salary=Label(w,text="Salary:",font=("times",15,"bold"))
        salary.place(x=550,y=310)
        txt_salary=Entry(w,font=("times",15),bg="lightgray",textvariable=sa)
        txt_salary.place(x=710,y=310,width=220)



btn1=Button(w,text="ADD",activebackground="white",command=on_submit,activeforeground="black",width=15,font=("times",18,"bold"),bg=
"black",fg="white",cursor="hand2")
        btn1.place(x=250,y=450)

btn2=Button(w,text="CLEAR",activebackground="white",activeforeground="black",width=15,font=("times",18,"bold"),bg="black",fg="whit
e",cursor="hand2")
        btn2.place(x=550,y=450)

if __name__ == "__main__":
    w=Tk()
    obj=mech_win(w)
    w.mainloop()
```

6.1.8 Update Mechanic

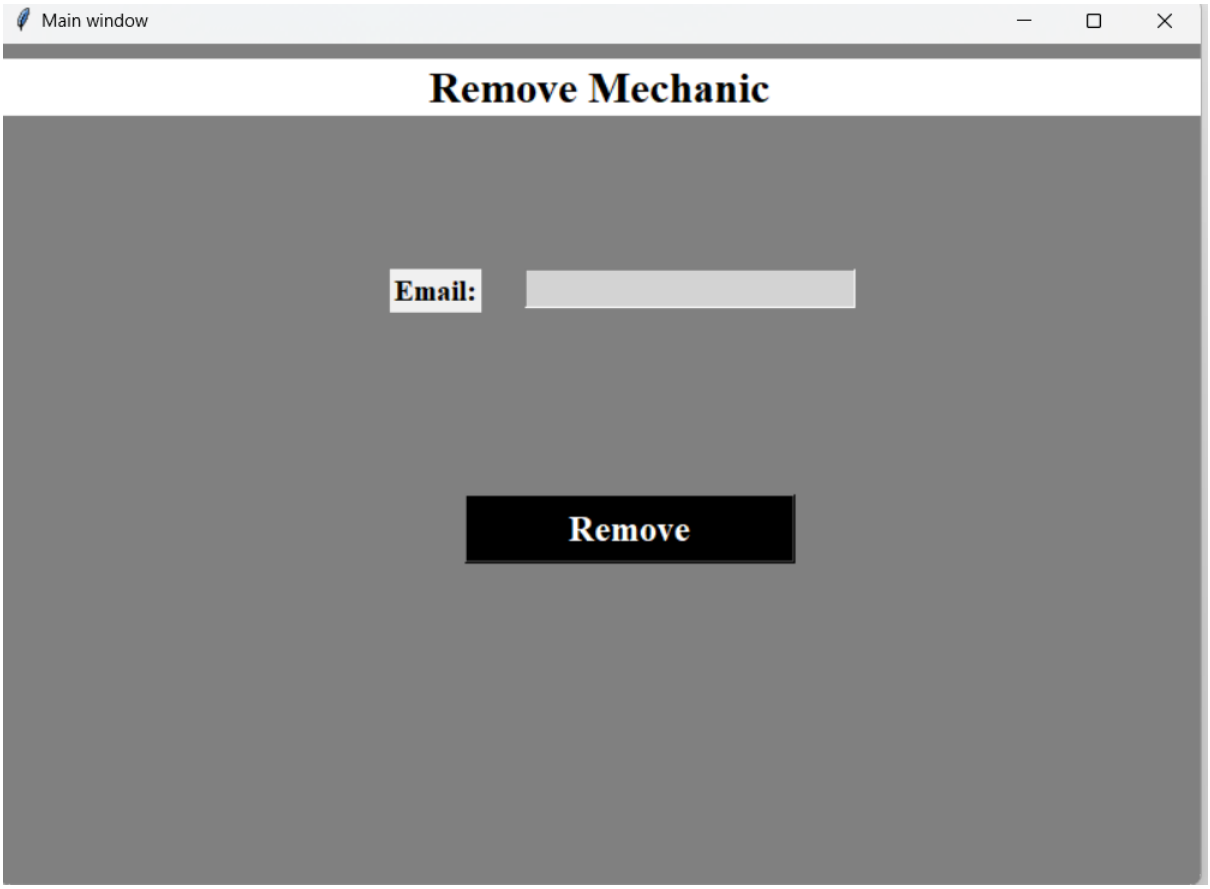


Description:This picture show update mechanic's data form for vehicle service management system

```python
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector

class mech_search:
     def __init__(self,w):

          w.title("Mechanic window")
          w.geometry("1000x560+250+140")
          w.config(bg="#808080")
          title=Label(w,text=" Search Mechanic Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1000)

          nm=StringVar()
          b=StringVar()
          pn=StringVar()
          em=StringVar()

          jd=StringVar()

          sk=StringVar()
          sa=StringVar()

          f_name=Label(w,text="Full Name:",font=("times",15,"bold"))
          f_name.place(x=100,y=170)
          txt_name1=Entry(w,font=("times",15),bg="lightgray",textvariable=nm)
          txt_name1.place(x=210,y=170,width=220)

          gender=Label(w,text="Gender",font=("times",15,"bold"))
          gender.place(x=100,y=240)
          txt_gender=Entry(w,font=("times",15),bg="lightgray",textvariable=b)
          txt_gender.place(x=210,y=240,width=220)


          email=Label(w,text="Email:",font=("times",15,"bold"))
          email.place(x=100,y=100)
          txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
          txt_email.place(x=210,y=100,width=220)

          addr=Label(w,text="Address:",font=("times",15,"bold"))
          addr.place(x=100,y=320)
          txt_addr=Entry(w,font=("times",15),bg="lightgray")
          txt_addr.place(x=210,y=310,width=220,height=60)



          phone=Label(w,text="Phone Number:",font=("times",15,"bold"))
          phone.place(x=550,y=100)
          txt_phone=Entry(w,font=("times",15),bg="lightgray",textvariable=pn)
          txt_phone.place(x=710,y=100,width=220)

          jdate=Label(w,text="Joining Date:",font=("times",15,"bold"))
          jdate.place(x=550,y=170)
          txt_jdate=Entry(w,font=("times",15),bg="lightgray",textvariable=jd)
          txt_jdate.place(x=710,y=170,width=220)

          skill=Label(w,text="Skill:",font=("times",15,"bold"))
          skill.place(x=550,y=240)
          txt_skill=Entry(w,font=("times",15),bg="lightgray",textvariable=sk)
          txt_skill.place(x=710,y=240,width=220)

          salary=Label(w,text="Salary:",font=("times",15,"bold"))
          salary.place(x=550,y=310)
          txt_salary=Entry(w,font=("times",15),bg="lightgray",textvariable=sa)
          txt_salary.place(x=710,y=310,width=220)
          def check():
               if nm.get()=="" or b.get()=="" or pn.get()=="" or em.get()==""  :
                    messagebox.showerror("error","Please valid infomation")
               else:

                 mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                 )
                 mycursor=mydb.cursor()
                 sql=f"update  mechanic set
t_name='{txt_name1.get()}',t_gender='{txt_gender.get()}',t_phone={txt_phone.get()},t_date={txt_jdate.get()},t_address='{txt_addr.g
et()}',t_skills='{txt_skill.get()}',t_salary={txt_salary.get()} where t_email='{txt_email.get()}'"
```

```
                mycursor.execute(sql)
                mydb.commit()
                messagebox.showinfo("valid","successfully updated Mechanic's data")



btn3=Button(w,text="Update",activebackground="white",command=check,activeforeground="black",width=15,font=("times",18,"bold"),bg="
black",fg="white",cursor="hand2")
          btn3.place(x=210,y=450)


btn2=Button(w,text="Clear",activebackground="white",activeforeground="black",width=15,font=("times",18,"bold"),bg="black",fg="whit
e",cursor="hand2")
          btn2.place(x=550,y=450)

if __name__ == "__main__":
     w=Tk()
     obj=mech_search(w)
     w.mainloop()
```

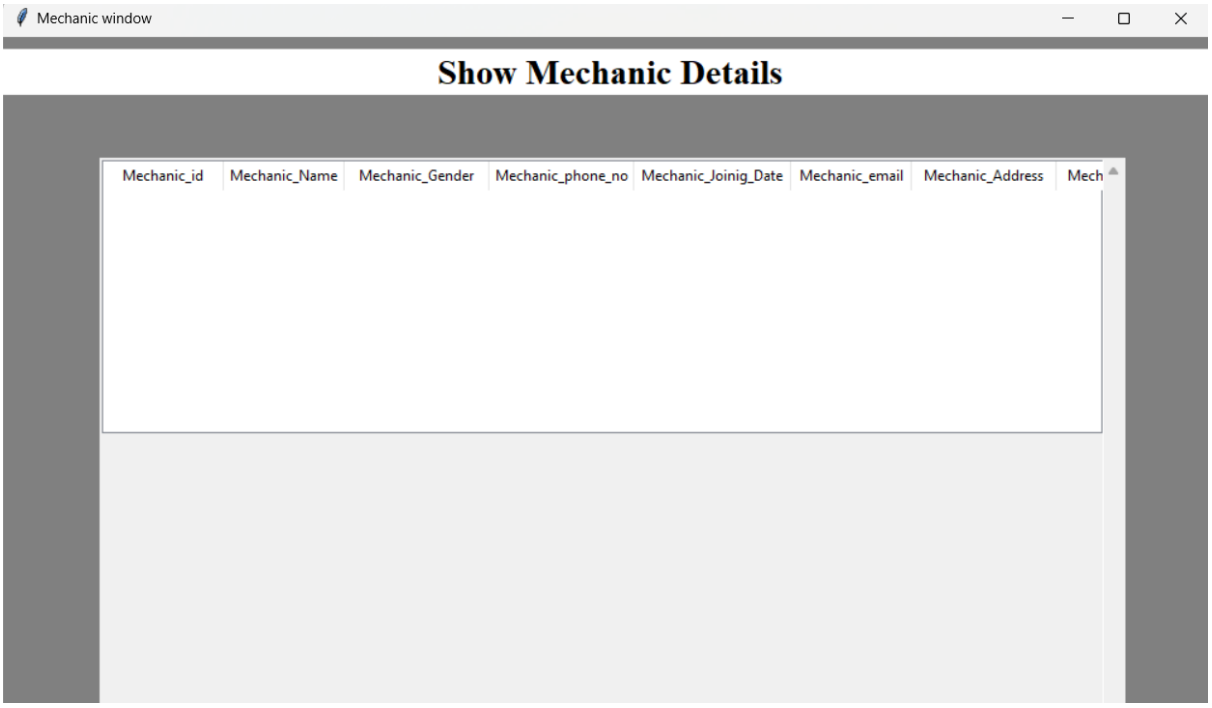
6.1.9 Remove Mechanic



Description:This picture show remove mechanic's data form for vehicle service management system

```
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector

class mech_remove:
     def __init__(self,w):

          w.title("Main window")
          w.geometry("800x560+380+100")
          w.config(bg="#808080")
          title=Label(w,text=" Remove Mechanic ",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=800)


          em=StringVar()


          email=Label(w,text="Email:",font=("times",15,"bold"))
          email.place(x=260,y=150)
          txt_email=Entry(w,font=("times",15),bg="lightgray",textvariable=em)
          txt_email.place(x=350,y=150,width=220)



          def check():
               if  em.get()==""  :
                    messagebox.showerror("error","Please valid infomation")
               else:
                    mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                  )
                    mycursor=mydb.cursor()
                    sql="delete  from mechanic where c_email=%s"
                    value=(txt_email.get(),)
                    mycursor.execute(sql,value)
                    mydb.commit()
                    messagebox.showinfo("valid","successfully deleted customer's data")




btn3=Button(w,text="Remove",activebackground="white",command=check,activeforeground="black",width=15,font=("times",18,"bold"),bg="
black",fg="white",cursor="hand2")
          btn3.place(x=310,y=300)


if __name__ == "__main__":
     w=Tk()
     obj=mech_remove(w)
     w.mainloop()
```

6.1.10 Show Mechanic



Description:This picture show all mechanic's data form for vehicle service management system

```
from tkinter import*
```

```
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
from tkinter import ttk
class all_mech:
     def __init__(self,w):

          w.title("Mechanic window")
          w.geometry("1000x560+250+140")
          w.config(bg="#808080")
          title=Label(w,text=" Show Mechanic Details",font=("times",22,"bold"),bg="white",fg="black")
          title.place(x=0,y=10,width=1000)
          details_f=Frame(w,bd=2)
          details_f.place(x=80,y=100,width=850,height=500)
          scroll_x=ttk.Scrollbar(details_f,orient=HORIZONTAL)
          scroll_y=ttk.Scrollbar(details_f,orient=VERTICAL)

cust_table=ttk.Treeview(details_f,columns=("t_id","t_name","t_gender","t_phone","t_date","t_email","t_address","t_skills","t_salar
y"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
          scroll_x.pack(side=BOTTOM,fill=X)
          scroll_y.pack(side=RIGHT,fill=Y)
          scroll_x.config(command=cust_table.xview)
          scroll_y.config(command=cust_table.yview)
          cust_table.heading("t_id",text="Mechanic_id")
          cust_table.heading("t_name",text="Mechanic_Name")
          cust_table.heading("t_gender",text="Mechanic_Gender")
          cust_table.heading("t_phone",text="Mechanic_phone_no")
          cust_table.heading("t_date",text="Mechanic_Joinig_Date")

          cust_table.heading("t_email",text="Mechanic_email")
          cust_table.heading("t_address",text="Mechanic_Address")
          cust_table.heading("t_skills",text="Mechanic_skills")
          cust_table.heading("t_salary",text="Mechanic_Salary")

          cust_table["show"]="headings"

          cust_table.column("t_id",width=100)
          cust_table.column("t_name",width=100)
          cust_table.column("t_gender",width=120)
          cust_table.column("t_phone",width=120)
          cust_table.column("t_email",width=100)
          cust_table.column("t_date",width=130)

          cust_table.column("t_address",width=120)
          cust_table.column("t_skills",width=100)
          cust_table.column("t_salary",width=100)

          cust_table.pack(fill=BOTH)



          mydb=mysql.connector.connect(
                     host="localhost",
                     user="root",
                     password="",
                     database="vehicle"
                )
          mycursor=mydb.cursor()
          mycursor.execute("select * from mechanic")
          myresult=mycursor.fetchall()
          if len(myresult)!=0:

                         for i in myresult:
                               cust_table.insert("",END,values=i)

                         mydb.commit()
                         mydb.close()
if __name__ == "__main__":
     w=Tk()
     obj=all_mech(w)
     w.mainloop()
```

7.Budget and Financial Plan

7.1Cost Estimation

Cost Item	Assumption (For Student Project)	INR Amount

Development Time	3 months (part-time)	0 (Student effort)
Hardware & Software	- Personal Computer (assumed available) - Python (free) IDE (free options like VS Code, PyCharm Community) Database (free options like SQLite, PostgreSQL Community)	0 (Assumed minimal or readily available)
Cloud Hosting (Optional)	- If deploying online (Heroku, AWS free tier, etc.) - Can be minimal or avoided for student projects	0 - 500 INR per month (if applicable)
Domain Name & Hosting (Optional)	- If deploying a public website - Can be minimal or avoided for student projects	0 - 500 INR per year (if applicable)
Other Costs	- Internet access (assumed available) - Books/Online Courses (optional)	0 - 5000 INR (depending on resources used)
Total Estimated Cost		0 - 6000 INR (Approximate)

7.2 Financial Planning

1. Development Costs:

Software:Python libraries and frameworks (Django/Flask, SQLAlchemy): Free and open-source.

IDE (VS Code, PyCharm Community): Free or have free versions.

Hardware:

Development machines: Cost can vary significantly depending on existing equipment.

Development Time:

Developer salaries (if hiring) or opportunity cost of developer time (if self-funded): This is a major cost factor.

Third-party services:Payment gateways, SMS/email APIs, cloud storage, etc.: Costs vary depending on usage.

2. Ongoing Costs:

Maintenance:Bug fixes, security updates, feature enhancements: Ongoing developer time required.

Support:

Customer support (if applicable): Costs associated with providing assistance to users.

Marketing and Sales:

3. Financial Sustainability Strategies:

Cost Optimization:Utilize free and open-source tools whenever possible.

Optimize development processes for efficiency.

Revenue Diversification:

Explore multiple revenue streams to reduce reliance on a single income source.

Customer Acquisition:Implement effective marketing and sales strategies to acquire and retain customers.

Customer Satisfaction:Provide excellent customer support to build long-term relationships.

Continuous Improvement:Regularly update and enhance the VSMS to meet evolving market demands and maintain a competitive advantage.

8 Future Enhancement

1. User Interface (UI) & User Experience (UX)

Mobile App Version: Develop a mobile app version for easy access by users on the go.

Improved Design: Make the UI more user-friendly with modern design practices, clearer navigation, and responsive layout.

Dark Mode/Accessibility Features: Implement features for accessibility, like text size adjustment, high-contrast themes, etc.

2. Integration with Other Systems

Payment Gateway Integration: Add options for online payments via credit/debit cards, UPI, or wallets for vehicle service payments.

Third-party APIs: Integrate with third-party APIs like Google Maps for location-based services or SMS/Email notifications for service reminders.

Inventory Management Integration: Automatically track spare parts and inventory for repairs.

3. Advanced Features

Customer Loyalty Program: Implement a loyalty or reward system for repeat customers.

AI & Predictive Analytics: Use machine learning to predict maintenance schedules based on vehicle history and usage patterns.

Maintenance History Tracking: Provide customers with detailed reports on their vehicle's service history and future needs.

Service Recommendations: Automatically suggest additional services based on vehicle data, age, or performance.

4. Performance Improvements

Faster Data Processing: Optimize database queries for faster access and updates.

Cloud Hosting for Better Performance: Migrate from local hosting to cloud services for improved performance and reliability.

5. Marketing & CRM Features

Customer Feedback System: Implement a feedback system to get reviews from customers and improve services.

Marketing Automation: Allow automated marketing features like promotional emails or SMS for offers and updates.

Referral Program: Introduce a referral program where customers can earn discounts or rewards by referring friends.

6. Security Enhancements

Advanced Encryption: Implement stronger encryption for sensitive data, such as customer payment information.

Two-Factor Authentication (2FA): Add 2FA for better user authentication and account security.

Regular Security Audits: Set up periodic audits and updates to identify and fix potential vulnerabilities.

7. Reporting & Analytics

Customizable Reports: Allow users to generate detailed reports on services, revenue, inventory, etc.

Data Visualization: Use graphs and charts to represent business data, making it easier to analyze trends.

9. Conclusion

In conclusion, the Vehicle Service Management System is a critical tool for efficiently managing vehicle services, enhancing customer experience, and streamlining operations. This project provides a comprehensive solution for tracking vehicle maintenance, managing inventory, and ensuring timely communication between service providers and customers. Through careful financial planning and development, the system has the potential to reduce operational costs, improve service quality, and increase customer satisfaction.

By focusing on key areas such as software tools, hosting, development resources, and security, the project has laid the foundation for successful execution. Future enhancements, including mobile app development, AI integration, and cloud scalability, will ensure the system remains innovative and responsive to the evolving needs of both service providers and customers.

The financial planning laid out throughout this document ensures that the project can be managed within a reasonable budget, with room for growth and necessary adjustments as the system develops. By keeping track of expenses, optimizing resource allocation, and setting clear objectives, the project will be on track for success.

Furthermore, ongoing improvements and enhancements will ensure that the system stays relevant and competitive in the future, while also expanding its features to accommodate new trends and technologies.

10.References

Websites:

W3Schools. :https://www.w3schools.com/sql/

Online Tools:

Visual Studio Team. (2024). Visual Studio Code. Microsoft. https://code.visualstudio.com/

Youtube

Our Professor :**DR.JAYDIP RATHOD**

Thank you for always being approachable and for your dedication to helping me succeed. I truly appreciate the time and effort you have invested in helping me enhance my skills.