

Acknowledgement

I would like to express my sincere gratitude and heartfelt thanks to Ms. Betsy N. Kuriakose, my esteemed computer science teacher. Her valuable suggestions and unwavering guidance were instrumental in the successful completion of my computer science project. She meticulously helped me comprehend the intricate details and critical concepts essential for this endeavour, illuminating paths that might otherwise have remained obscure. Her profound knowledge and pedagogical expertise provided a solid foundation, enabling me to navigate the project's complexities with greater clarity and confidence.

This project provided a significant opportunity to apply the best of my knowledge and practical experience gained during my comprehensive studies and engaging coursework. While I poured my efforts into this undertaking, it is important to acknowledge that the development of sophisticated software systems is inherently a complex and often time-consuming process. It demands not only a systematic and rigorous study but also profound insight, a clear vision, and a consistently professional approach throughout all phases of design and development. The challenges encountered served as valuable learning experiences, reinforcing the importance of meticulous planning and adaptive problem-solving.

Furthermore, I feel deeply indebted to my friends for their consistent support and valuable suggestions throughout the project work. Their collaborative spirit and diverse perspectives offered fresh insights and constructive feedback, significantly contributing to the refinement and enhancement of the project. The exchange of ideas and collaborative troubleshooting sessions proved to be an invaluable asset, enriching the overall development process.

Table Of Contents

Sl no.	Name	Page no.
1	Introduction	3
2	Objective And Scope	4
3	System Implementation: 3.1 The Hardware Used 3.2 The Software Used 3.3 The Hardware Required 3.4 The Software Required	5
4	System Design And Development 4.1 Database Design 4.2 Event Coding	7
5	Output 5.1 User 5.2 Admin	51
6	User Manual	59
7	Conclusion	60
8	References	61

Introduction

This project, Magnus Motors – Car Dealership Management System developed by Geordin Shelly, Vaishnav Prasad and Asher Thomas Viju, is a software application designed to automate and simplify the daily operations of a car dealership. The system allows customers to conveniently book cars, schedule servicing, and request test drives, while ensuring accurate record-keeping and improved dealership efficiency.

The dealership name, Magnus Motors, is an imaginary brand, created to showcase how technology can be applied to solve real-world business challenges. The software emphasises both ease of use for customers and efficient backend management for dealership staff.

The project has been developed using the following technologies:

- Python: For implementing the core logic and overall programming of the system.
- CustomTkinter: For designing a modern, responsive, and user-friendly graphical user interface (GUI).
- MySQL: For reliable database storage and handling of customer details, bookings, cars, and service records.

With these tools, the system provides essential dealership functionalities, including:

- Car booking by model, variant, and price range
- Easy scheduling of test drives with preferred timings
- Vehicle service booking with enquiring about customers' experiences
- Validated storage of customer details (email, phone, etc.)
- Database for efficient retrieval and updates

This project demonstrates how Python-based GUI applications integrated with databases can mirror real-world use cases, making dealership management faster, accurate, and customer-friendly.

Objective And Scope

Project Objectives

The main objectives of the Magnus Motors – Car Dealership Management System are:

- To provide a digital platform for booking cars across different companies and variants.
- To design a user-friendly application for managing car bookings, service scheduling, and test drives.
- To implement a database-driven system that securely stores customer details, car data, and booking/service records.
- To eliminate manual processes by automating dealership operations for efficiency and accuracy.
- To ensure data validation and consistency in customer entries, such as contact details and booking information.
- To demonstrate the practical use of Python programming, CustomTkinter for GUI, and MySQL database integration in building real-world business solutions.

Scope of the Project

The scope of this project extends to covering the core operations of a modern multi-brand car dealership. The features include:

- Car Sales Management: Customers can browse and book cars from different companies, compare available models, and select variants.
- Test Drive Scheduling: Customers can book test drives for cars of their choice with preferred dates and times.
- Service Booking: Existing customers can schedule servicing of purchased vehicles with proper record-keeping for follow-ups.
- Customer Information Management: Secure and validated storage of customer data, including name, email, phone, and booking history.
- Multi-Company Database Integration: The system supports the storage of car details from different companies and their variants, ensuring smooth selection and booking.
- Intuitive GUI: A modern and interactive interface built using customtkinter, ensuring simplicity for customers and dealership staff.

System Implementation

Hardware Used

The hardware used to develop and test the Magnus Motors – Car Dealership Management System includes the following specifications:

- Processor: Intel Core i3-1115G4
- RAM: 8 GB
- Graphics: Intel UHD Graphics
- Display: 1920 x 1080 resolution, 60 Hz refresh rate
- Storage: 1 TB hard disk drive (HDD) and 256 GB solid-state drive (SSD)

This hardware configuration provided sufficient performance and storage capacity to develop, test, and run the application efficiently.

Software Used

The software used to develop and test the Magnus Motors – Car Dealership Management System includes the following specifications:

- Operating System: Windows 11
- Programming Language: Python (3.12.4)
- GUI Library: CustomTkinter (5.2.2)
- Database System: MySQL (8.0.41)
- Database Connector: mysql-connector-python (9.3.0)
- Additional Python Libraries:
 - tkinter
 - tkcalendar
 - datetime
 - Pillow
 - ttk
 - messagebox

These tools and libraries were selected to create a modern, robust, and feature-rich application for managing multiple dealership operations efficiently.

Hardware Required

The minimum hardware specifications required to install and run the Magnus Motors – Car Dealership Management System are:

- Processor: Intel Core i3 (or equivalent dual-core processor)
- RAM: 4 GB
- Storage: 100 GB HDD or SSD (to accommodate the OS, Python, MySQL, software files, and growing database)
- Display: 1280 x 720 or higher resolution monitor
- Input Devices: Standard keyboard and mouse
- Internet Connection: Recommended for software installation, database setup, and updates

Software Required

To install and operate the Magnus Motors – Car Dealership Management System, the following software components are required:

- Python (version 3.7 or higher): The main programming environment in which the application is developed and executed.
- MySQL Community Server (version 5.7 or higher): Relational database system for storing dealership data.
- MySQL Connector/Python: Library to connect Python applications to the MySQL database.
- A Python IDE or code editor: Such as IDLE, PyCharm, VSCode, or any editor for running and modifying code.
- Operating System: Windows 10/11, Linux, or macOS – any recent OS that supports Python and MySQL installations.
- customtkinter: For creating a modern and responsive GUI.
- tkinter: Standard Python GUI package, typically included with Python installations.
- tkcalendar: For DateEntry widgets and calendar functionalities.
- datetime: Standard Python package for date and time operations (comes built-in).
- Pillow (PIL): For image processing and handling image display in the application.
- ttk: Enhanced themed widgets for tkinter (included in the tkinter module).
- messagebox: For showing dialogue boxes, part of tkinter.

All required libraries and packages should be installed before deployment for proper functionality.

System Design And Development

Database Design

Database: dealer_service

The dealer_service database contains the following tables that support the operation of the Magnus Motors dealership system:

users: Stores customer and user details such as phone number, email, password, name, and address.

cars: Contains car inventory information like company, model, and variant-wise prices.

car_purchase: Records details of car purchases linked to customer phone numbers and includes car details and payment information.

car_service: Manages car servicing information including customer, car, service type, problem description, and service dates.

car_test: Handles test drive bookings including customer phone number, car, company, scheduled date, and time.

This structure enables efficient management of customers, cars, purchases, services, and test drives in the dealership system.

EVENT CODING


```
import customtkinter as ctk
import tkinter as tk
from customtkinter import *
from tkinter import messagebox,ttk
from tkcalendar import DateEntry
import datetime
from PIL import Image

import mysql.connector
mycon=mysql.connector.connect(host='localhost',user='root',passwd='a
sher123',database='dealer_service')
mycur=mycon.cursor()

#creating main window
root=ctk.CTk()
root.after(0,lambda: root.state('zoomed'))
ctk.set_appearance_mode("dark") #setiing window to dark mode
ctk.set_default_color_theme("dark-blue")
#creating frame for a clean layout

logo=Image.open("C:\\\\Users\\hp\\Downloads\\MagnusMotorsLogo.png")
logo = CTkImage(light_image=logo, dark_image=logo, size=(200, 200))

namelogo_img=Image.open("C:\\\\Users\\hp\\Downloads\\namelogo.png")
namelogo_img = CTkImage(light_image=namelogo_img,
dark_image=namelogo_img, size=(200, 100))

f=font=('MT',20, "bold")
def cancellation(x):
    root.deiconify()
    root.state("zoomed")
    x.destroy()

def destroy(): #cancelling main window fnc
    root.destroy()
```

```

def admin():
    adminwindow=CTkToplevel(root)
    adminwindow.state("zoomed")
    adminwindow.title("Admin Access Terminal")

    style = ttk.Style(root)
    style.theme_use("clam") # Use a theme that supports
    fieldbackground

    # Configure Treeview style for dark theme
    style.configure("Treeview",
                    background="#2d2d2d", # Dark gray background
    for rows
                    fieldbackground="#2d2d2d", # Dark gray
    background for empty area
                    foreground="white", # white text color
                    rowheight=25,
                    font=("Calibri", 12))

    style.configure("Treeview.Hheading",
    for header
                    background="#444444", # slightly lighter gray
                    foreground="white",
                    font=("Calibri", 13, "bold"))

    # Optional: map selected row colors
    style.map("Treeview",
             background=[("selected", "#0078d7")], # Blue selection
             foreground=[("selected", "white")])

def carbooking():

    def delete_booking():
        if delivered.get():
            a=delivered.get()
            delivered.delete(0, 'end')
        elif cancel_booking.get():

```

```

        a=cancel_booking.get()
        cancel_booking.delete(0, 'end')
        mycur.execute("DELETE FROM car_purchase where
Purchase_no={}".format(a))
        fetch_data_and_populate()#calling function to
populate data in treeview

    for widget in adminwindow.wininfo_children():
        widget.destroy()

    def fetch_data_and_populate(): #Fetches data from MySQL
and populates the Treeview
        # 1. Execute query and fetch data
        mycur.execute("SELECT * FROM car_purchase")
        rows = mycur.fetchall()

        # 2. Clear existing data in the treeview
        for item in tree.get_children():
            tree.delete(item)

        # 3. Populate the treeview with new data
        for row in rows:
            tree.insert('', tk.END, values=row)

tframe5=CTkFrame(adminwindow,width=400,height=100,fg_color="transparent")

tframe5.place(relx=0.5,rely=0.7,anchor='center')

# --- Frame for Treeview and Scrollbar ---
tree_frame = CTkFrame(adminwindow)
tree_frame.place(anchor='center',relx=0.5,rely=0.2)

# --- Scrollbar ---
tree_scroll = ttk.Scrollbar(tree_frame)
tree_scroll.pack(side="right", fill="y")

# --- Treeview widget ---

```

```

        columns = ('Purchase_no','phno', 'car_name', 'color',
'variant', 'payment','Price')
        tree = ttk.Treeview(
            tree_frame,height=15,
            columns=columns,
            show='headings',
            yscrollcommand=tree_scroll.set)

        # Define headings
        tree.heading('Purchase_no', text='Purchase Number')
        tree.heading('phno', text='Phone number')
        tree.heading('car_name', text='Car Name')
        tree.heading('color', text='Colour')
        tree.heading('variant', text='Variant')
        tree.heading('payment', text='Payment Method')
        tree.heading('Price', text='Price')

        tree.pack(fill="both", expand=True)
        tree_scroll.config(command=tree.yview)

        # --- Load Data Button ---
        load_button = CtkButton(tframe5, text="Load/Refresh Data",
command=fetch_data_and_populate)
        load_button.grid(row=0,column=1)
        exit_btn=CtkButton(adminwindow,text="<--
",command=admin_widgets)
        exit_btn.grid(row=0,column=0)

        CtkLabel(tframe5,text="
").grid(row=1,column=1,rowspan=2,pady=20)
        CtkLabel(tframe5,text="Car Delivery
complete",font=("Arial", 20, "bold")).grid(row=3,column=0,pady=20)
        delivered=CtkEntry(tframe5,placeholder_text="Purchase
Number")
        delivered.grid(row=4,column=0)
        CtkButton(tframe5,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=0,pady=10)

```

```

        CTKLabel(tframe5,text="Car Booking Cancel",font=("Arial",
20, "bold")).grid(row=3,column=2,pady=20)
        cancel_booking=CTKEntry(tframe5,placeholder_text="Purchase
Number")
        cancel_booking.grid(row=4,column=2)
        CTKButton(tframe5,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=2,pady=10)
        fetch_data_and_populate()

```

```
def carlist():
```

```

    for widget in adminwindow.wininfo_children():
        widget.destroy()

```

```
def deletecar():
```

```

    company_name=company_name_variable.get()
    carno=deletion.get()
    delete_query=f"DELETE FROM cars WHERE Car_no=%s"
    mycur.execute(delete_query,(carno,))
    deletion.delete(0,'end')
    fetch_data_and_populate()#calling function to
populate data in treeview

```

```
def addcar():
```

```

    company_name=company_name_variable.get()
    Car_no_query=f"SELECT Car_no FROM cars ORDER BY
Car_no DESC LIMIT 1"
    mycur.execute(Car_no_query)
    last_row =mycur.fetchone()
    if last_row:
        a=last_row[0]+1
    else:
        a=1
    name=car_name.get()
    p1=price1.get()
    p2=price2.get()
    p3=price3.get()

```

```

        addition_query=f"INSERT INTO cars
(Car_no,company,car,price1,price2,price3) VALUES(%s,%s,%s,%s,%s,%s)"

```

```

mycur.execute(addition_query,(a,company_name,name,p1,p2,p3))

```

```

        car_name.delete(0,'end')

```

```

        price1.delete(0,'end')

```

```

        price2.delete(0,'end')

```

```

        price3.delete(0,'end')

```

```

        fetch_data_and_populate()#calling function to
populate data in treeview

```

```

def editcar():

```

```

    company_name=company_name_variable.get()

```

```

    price=price_variable.get()

```

```

    car_no=edit_car_no.get()

```

```

    new_price=edit_price.get()

```

```

    edit_query=f"UPDATE cars SET {price}=%s where
Car_no=%s"

```

```

    mycur.execute(edit_query,(new_price,car_no))

```

```

    edit_car_no.delete(0,'end')

```

```

    edit_price.delete(0,'end')

```

```

    fetch_data_and_populate()#calling function to
populate data in treeview

```

```

def fetch_data_and_populate(): #Fetches data from MySQL
and populates the Treeview

```

```

    company_name=company_name_variable.get()

```

```

    # 1. Execute query and fetch data

```

```

    display="SELECT Car_no,car,price1,price2,price3 FROM
cars WHERE company=%s"

```

```

    mycur.execute(display,(company_name,))

```

```

    rows = mycur.fetchall()

```

```

    # 2. Clear existing data in the treeview

```

```

    for item in tree.get_children():

```

```

        tree.delete(item)

```

```

    # 3. Populate the treeview with new data

```

```

    for row in rows:

```

```

        tree.insert('', tk.END, values=row)

```

```
tframe6=CTkFrame(adminwindow,width=400,height=100,fg_color="transparent")
```

```
    tframe6.place(relx=0.5,rely=0.7,anchor='center')
```

```
    # --- Frame for Treeview and Scrollbar ---
```

```
    tree_frame = CTkFrame(adminwindow)
```

```
    tree_frame.place(anchor='center',relx=0.5,rely=0.2)
```

```
    # --- scrollbar ---
```

```
    tree_scroll = ttk.Scrollbar(tree_frame)
```

```
    tree_scroll.pack(side="right", fill="y")
```

```
    # --- Treeview widget ---
```

```
    columns = ('Car_no','car', 'price1', 'price2', 'price3')
```

```
    tree = ttk.Treeview(
```

```
        tree_frame,height=15,
```

```
        columns=columns,
```

```
        show='headings',
```

```
        yscrollcommand=tree_scroll.set)
```

```
    # Define headings
```

```
    tree.heading('Car_no', text='Car Number')
```

```
    tree.heading('car', text='Car Name')
```

```
    tree.heading('price1', text='Price 1')
```

```
    tree.heading('price2', text='Price 2')
```

```
    tree.heading('price3', text='Price 3')
```

```
    tree.pack(fill="both", expand=True)
```

```
    tree_scroll.config(command=tree.yview)
```

```
company=['Tata','Suzuki','Mahindra','Hyundai','Kia','Citroen','Volks  
wagen','Skoda','Honda','Toyota']
```

```
company_name_variable=ctk.StringVar(value="select a  
Company")
```

```

# --- Load Data Button ---
company_name_entry=CTkOptionMenu(tframe6,
variable=company_name_variable, values=company)
company_name_entry.grid(row=0,column=1,pady=5)
load_button = CTkButton(tframe6, text="Load/Refresh Data",
command=fetch_data_and_populate)
load_button.grid(row=1,column=1,pady=5)
exit_btn=CTkButton(adminwindow,text="<--
",command=admin_widgets)
exit_btn.grid(row=0,column=0)

CTkLabel(tframe6,text="
").grid(row=2,column=1,rowspan=2,pady=20)

CTkLabel(tframe6,text="Delete Car",font=("Arial", 20,
"bold")).grid(row=3,column=0,pady=20)
deletion=CTkEntry(tframe6,placeholder_text="Car Number")
deletion.grid(row=4,column=0,padx=5)
CTkButton(tframe6,text='confirm',font=("Arial", 20,
"bold"),command=deletecar).grid(row=5,column=0,pady=10,padx=5)

CTkLabel(tframe6,text="Add Car",font=("Arial", 20,
"bold")).grid(row=3,column=1,pady=15)
car_name=CTkEntry(tframe6,placeholder_text="Car Name")
car_name.grid(row=4,column=1,pady=5,padx=5)
price1=CTkEntry(tframe6,placeholder_text="price 1")
price1.grid(row=5,column=1,pady=5,padx=5)
price2=CTkEntry(tframe6,placeholder_text="price 2")
price2.grid(row=6,column=1,pady=5,padx=5)
price3=CTkEntry(tframe6,placeholder_text="price 3")
price3.grid(row=7,column=1,pady=5,padx=5)
CTkButton(tframe6,text='confirm',font=("Arial", 20,
"bold"),command=addcar).grid(row=8,column=1,pady=5,padx=5)

CTkLabel(tframe6,text="Edit Price",font=("Arial", 20,
"bold")).grid(row=3,column=2,pady=20)
edit_car_no=CTkEntry(tframe6,placeholder_text="Car
Number")
edit_car_no.grid(row=4,column=2,padx=5)

```



```

        price_variable=ctk.StringVar(value="Choose which Price")
        price_type=CTkOptionMenu(tframe6,variable=price_variable
,values=['Price1','Price2','Price3'])
        price_type.grid(row=5,column=2,padx=5)
        edit_price=CTkEntry(tframe6,placeholder_text="Enter
Price")
        edit_price.grid(row=6,column=2,padx=5)
        CTkButton(tframe6,text='confirm',font=("Arial", 20,
"bold"),command=editcar).grid(row=7,column=2,pady=10,padx=5)

```

```

def cartest():
    def delete_booking():
        if test_finished.get():
            a=test_finished.get()
            test_finished.delete(0, 'end')
        elif cancel_booking.get():
            a=cancel_booking.get()
            cancel_booking.delete(0, 'end')
        mycur.execute("DELETE FROM car_Test where
Test_id={}".format(a))
        fetch_data_and_populate()#calling function to
populate data in treeview

    for widget in adminwindow.wininfo_children():
        widget.destroy()

    def fetch_data_and_populate(): #Fetches data from MySQL
and populates the Treeview
        # 1. Execute query and fetch data
        mycur.execute("SELECT * FROM car_test")
        rows = mycur.fetchall()

        # 2. Clear existing data in the treeview
        for item in tree.get_children():
            tree.delete(item)

        # 3. Populate the treeview with new data

```

```

        for row in rows:
            tree.insert('', tk.END, values=row)

tframe7=CTkFrame(adminwindow,width=400,height=100,fg_color="transparent")

tframe7.place(relx=0.5,rely=0.7,anchor='center')

# --- Frame for Treeview and Scrollbar ---
tree_frame = CTkFrame(adminwindow)
tree_frame.place(anchor='center',relx=0.5,rely=0.2)

# --- Scrollbar ---
tree_scroll = ttk.Scrollbar(tree_frame)
tree_scroll.pack(side="right", fill="y")

# --- Treeview widget ---
columns = ('Test_id','phno', 'car_name', 'company',
'time', 'date')
tree = ttk.Treeview(
    tree_frame,height=15,
    columns=columns,
    show='headings',
    yscrollcommand=tree_scroll.set)

# Define headings
tree.heading('Test_id', text='Test ID')
tree.heading('phno', text='Phone number')
tree.heading('car_name', text='Car Name')
tree.heading('company', text='Comapny')
tree.heading('time', text='Time')
tree.heading('date', text='Date')

tree.pack(fill="both", expand=True)
tree_scroll.config(command=tree.yview)

# --- Load Data Button ---

```

```

        load_button = CtkButton(tframe7, text="Load/Refresh Data",
                                command=fetch_data_and_populate)
        load_button.grid(row=0,column=1)
        exit_btn=CtkButton(adminwindow,text="<--",
                                command=admin_widgets)
        exit_btn.grid(row=0,column=0)

        CtkLabel(tframe7,text="
").grid(row=1,column=1,rowspan=2,pady=20)
        CtkLabel(tframe7,text="Test Drive complete",font=("Arial",
20, "bold")).grid(row=3,column=0,pady=20)
        test_finished=CtkEntry(tframe7,placeholder_text="Test ID")
        test_finished.grid(row=4,column=0)
        CtkButton(tframe7,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=0,pady=10)
        CtkLabel(tframe7,text="Cancel Test Drive
Booking",font=("Arial", 20, "bold")).grid(row=3,column=2,pady=20)
        cancel_booking=CtkEntry(tframe7,placeholder_text="Test
ID")
        cancel_booking.grid(row=4,column=2)
        CtkButton(tframe7,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=2,pady=10)
        fetch_data_and_populate()

def servicebooking():

    def delete_booking():
        if serviced.get():
            a=serviced.get()
            serviced.delete(0, 'end')
        elif cancel_booking.get():
            a=cancel_booking.get()
            cancel_booking.delete(0, 'end')
        mycur.execute("DELETE FROM car_service where
service_id={}".format(a))
        fetch_data_and_populate()#calling function to
        populate data in treeview

    for widget in adminwindow.winfo_children():

```

```
        widget.destroy()

    def fetch_data_and_populate(): #Fetches data from MySQL
and populates the Treeview
        # 1. Execute query and fetch data
        mycur.execute("SELECT * FROM car_service")
        rows = mycur.fetchall()

        # 2. Clear existing data in the treeview
        for item in tree.get_children():
            tree.delete(item)

        # 3. Populate the treeview with new data
        for row in rows:
            tree.insert('', tk.END, values=row)

tframe8=CTkFrame(adminwindow,width=400,height=100,fg_color="transparent")

tframe8.place(relx=0.5,rely=0.7,anchor='center')

# --- Frame for Treeview and Scrollbar ---
tree_frame = CTkFrame(adminwindow)
tree_frame.place(anchor='center',relx=0.5,rely=0.2)

# --- Scrollbar ---
tree_scroll = ttk.Scrollbar(tree_frame)
tree_scroll.pack(side="right", fill="y")

# --- Treeview widget ---
columns = ('service_id','phno', 'car_name', 'company',
'service_type', 'problem','date')
tree = ttk.Treeview(
    tree_frame,height=15,
    columns=columns,
    show='headings',
    yscrollcommand=tree_scroll.set)

# Define headings
```

```

tree.heading('service_id', text='Service ID')
tree.heading('phno', text='Phone number')
tree.heading('car_name', text='Car Name')
tree.heading('company', text='Company')
tree.heading('service_type', text='Service Type')
tree.heading('problem', text='Problem')
tree.heading('date', text='Date')

tree.pack(fill="both", expand=True)
tree_scroll.config(command=tree.yview)

# --- Load Data Button ---
load_button = CtkButton(tframe8, text="Load/Refresh Data",
command=fetch_data_and_populate)
load_button.grid(row=0,column=1)
exit_btn=CtkButton(adminwindow,text="<--
",command=admin_widgets)
exit_btn.grid(row=0,column=0)

CtkLabel(tframe8,text="
").grid(row=1,column=1,rowspan=2,pady=20)
CtkLabel(tframe8,text="Car Service
Complete",font=("Arial", 20, "bold")).grid(row=3,column=0,pady=20)
serviced=CtkEntry(tframe8,placeholder_text="Service ID")
serviced.grid(row=4,column=0)
CtkButton(tframe8,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=0,pady=10)
CtkLabel(tframe8,text="Car Service Cancel",font=("Arial",
20, "bold")).grid(row=3,column=2,pady=20)
cancel_booking=CtkEntry(tframe8,placeholder_text="Service
ID")
cancel_booking.grid(row=4,column=2)
CtkButton(tframe8,text='confirm',font=("Arial", 20,
"bold"),command=delete_booking).grid(row=5,column=2,pady=10)
fetch_data_and_populate()

def admin_widgets():
    for widget in adminwindow.wininfo_children():

```

```
widget.destroy()
```

```
tframe4=CTkFrame(adminwindow,width=400,height=400,border_width=5,border_color="#FFFFFF")
```

```
tframe4.place(relx=0.5,rely=0.4,anchor='center')
```

```
innerframe=CTkFrame(tframe4,width=400,height=100,fg_color="transparent")
```

```
innerframe.place(relx=0.5,rely=0.5,anchor='center')
```

```
CTkButton(innerframe,text="Access Carbooking",command=carbooking,font=("Arial", 16, "bold")).grid(row=0,column=0,pady=5)
```

```
CTkButton(innerframe,text="Access Car List",command=carlist,font=("Arial", 16, "bold")).grid(row=1,column=0,pady=5)
```

```
CTkButton(innerframe,text="Access Test Drive Booking",command=cartest,font=("Arial", 16, "bold")).grid(row=2,column=0,pady=5)
```

```
CTkButton(innerframe,text="Access Service Booking",command=servicebooking,font=("Arial", 16, "bold")).grid(row=3,column=0,pady=5)
```

```
CTkButton(innerframe,text="Exit",command=lambda: cancellation(adminwindow),font=("Arial", 16, "bold"),fg_color='#990000',hover_color="#670101").grid(row=4,column=0,pady=5)
```

```
CTkLabel(adminwindow,text='',image=namelogo_img).pack(side='bottom')
```

```
admin_widgets()
```

```
def testcar():
```

```
    root.withdraw()
```

```
    root2=ctk.CTkTopLevel(root)
```

```
    root2.state("zoomed")
```

```
    root2.title("Test Drive Booking Portal")
```

```
    #creating a transparent frame to arrange widgets
```

```
tframe3=CTkFrame(root2,width=400,height=100,fg_color="transparent")
```

```
tframe3.place(relx=0.5,rely=0.4,anchor='center')
```

```
#sample data of list of cars
```

#1.Tata

```
mycur.execute("SELECT Car FROM cars WHERE company='Tata'")
```

```
z1 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Tata = [row[0] for row in z1]
```

#2.Suzuki

```
mycur.execute("SELECT Car FROM cars WHERE company='Suzuki'")
```

```
z2 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Suzuki = [row[0] for row in z2]
```

#3.Mahindra

```
mycur.execute("SELECT Car FROM cars WHERE company='Mahindra'")
```

```
z3 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Mahindra = [row[0] for row in z3]
```

#4.Hyundai

```
mycur.execute("SELECT Car FROM cars WHERE company='Hyundai'")
```

```
z4 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Hyundai = [row[0] for row in z4]
```

#5.Kia

```
mycur.execute("SELECT Car FROM cars WHERE company='Kia'")
```

```
z5 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Kia = [row[0] for row in z5]
```

#6.Citroen

```
mycur.execute("SELECT Car FROM cars WHERE company='Citroen'")
```

```
z3 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Citroen = [row[0] for row in z3]
```

```
#7.Volkswagen
```

```
mycur.execute("SELECT Car FROM cars WHERE  
company='Volkswagen'")
```

```
z7 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Volkswagen = [row[0] for row in z7]
```

```
#8.Skoda
```

```
mycur.execute("SELECT Car FROM cars WHERE company='Skoda'")
```

```
z8 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Skoda = [row[0] for row in z8]
```

```
#9.Honda
```

```
mycur.execute("SELECT Car FROM cars WHERE company='Honda'")
```

```
z8 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Honda = [row[0] for row in z8]
```

```
#10.Toyota
```

```
mycur.execute("SELECT Car FROM cars WHERE company='Toyota'")
```

```
z10 = mycur.fetchall()
```

```
# Extract car names as a flat list
```

```
Toyota = [row[0] for row in z10]
```

```
selected_company = ctk.StringVar(value="Tata")
```

```
companychange=0
```

```
def select_from_tata(car):
```

```
    selected_company.set("Tata")
```

```
def select_from_suzuki(car):
```

```
    selected_company.set("Suzuki")
```

```
def select_from_mahindra(car):
```

```
    selected_company.set("Mahindra")
```



```
def select_from_hyundai(car):
    selected_company.set("Hyundai")
def select_from_kia(car):
    selected_company.set("Kia")
def select_from_citroen(car):
    selected_company.set("Citroen")
def select_from_volkswagen(car):
    selected_company.set("Volkswagen")
def select_from_skoda(car):
    selected_company.set("Skoda")
def select_from_honda(car):
    selected_company.set("Honda")
def select_from_toyota(car):
    selected_company.set("Toyota")

mycur.execute("SELECT Test_id FROM car_test ORDER BY Test_id
DESC LIMIT 1")
last_row = mycur.fetchone()
if last_row:
    a=last_row[0]+1
else:
    a=1

def generate_times():
    times = []
    start = datetime.datetime.strptime("09:00", "%H:%M")
    for i in range(17):
        times.append((start +
datetime.timedelta(minutes=30*i)).strftime("%H:%M"))
    return times

def confirm_booking():
    phno=my_var.get()
    vehicle = vehicle_var.get()
    date    = date_entry.get_date()
    time    = time_var.get()
```

```

company = selected_company.get()
if vehicle and date and time:
    msg = (
        f"Test Drive Booked!\n\n"
        f"Vehicle: {vehicle}\n"
        f>Date: {date.strftime('%Y-%m-%d')}\n"
        f"Time: {time}\n"
        f"Company: {company}\n"
        f"Test ID: {a}\n"
        "Kindly Remember the Test ID"
    )
    messagebox.showinfo("Booking Confirmed", msg)
else:
    messagebox.showwarning("Incomplete", "Please select
all options.")
    mycur.execute("INSERT INTO car_test
(Test_id,phno,car_name,company,time,date)
VALUES({},'{}','{}','{}','{}','{}')".format(a,phno,vehicle,company,t
ime,date))
    root.deiconify()      # Show the root window again
    root.state('zoomed')
    root2.destroy()

# Title
title = CtkLabel(root2, text="TEST DRIVE BOOKING FACILITY",
font=("Arial", 50, "bold"))
title.pack(pady=10)

# vehicle selection
CtkLabel(tframe3, text="Select a vehicle:", font=("Arial", 20,
"bold")).grid(row=0,column=2,pady=10)
vehicle_var = ctk.StringVar(value=" ")

#Tata
ctk.CtkLabel(tframe3, text="Tata", font=("Arial",
12,"bold")).grid(row=1,column=0)
Test_menu = CtkOptionMenu(tframe3, variable=vehicle_var,
values=Tata,command=select_from_tata)
Test_menu.grid(row=2,column=0,padx=8)

```

```
#Suzuki
    ctk.CTkLabel(tframe3, text="Suzuki", font=("Arial",
12,"bold")).grid(row=1,column=1)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Suzuki,command=select_from_suzuki)
    Test_menu.grid(row=2,column=1,padx=8)

#Mahindra
    ctk.CTkLabel(tframe3, text="Mahindra", font=("Arial",
12,"bold")).grid(row=1,column=2)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Mahindra,command=select_from_mahindra)
    Test_menu.grid(row=2,column=2,padx=8)

#Hyundai
    ctk.CTkLabel(tframe3, text="Hyundai", font=("Arial",
12,"bold")).grid(row=1,column=3)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Hyundai,command=select_from_hyundai)
    Test_menu.grid(row=2,column=3,padx=8)

#Kia
    ctk.CTkLabel(tframe3, text="Kia", font=("Arial",
12,"bold")).grid(row=1,column=4)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Kia,command=select_from_kia)
    Test_menu.grid(row=2,column=4,padx=8)

#Citroen
    ctk.CTkLabel(tframe3, text="Citoren", font=("Arial",
12,"bold")).grid(row=3,column=0)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Citroen,command=select_from_citroen)
    Test_menu.grid(row=4,column=0,padx=8)

#Volkswagen
    ctk.CTkLabel(tframe3, text="Volkswagen", font=("Arial",
12,"bold")).grid(row=3,column=1)
```

```

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Volkswagen,command=select_from_volkswagen)

    Test_menu.grid(row=4,column=1,padx=8)

    #Skoda

    ctk.CTkLabel(tframe3, text="Skoda", font=("Arial",
12,"bold")).grid(row=3,column=2)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Skoda,command=select_from_skoda)

    Test_menu.grid(row=4,column=2,padx=8)

    #Honda

    ctk.CTkLabel(tframe3, text="Honda", font=("Arial",
12,"bold")).grid(row=3,column=3)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Honda,command=select_from_honda)

    Test_menu.grid(row=4,column=3,padx=8)

    #Toyota

    ctk.CTkLabel(tframe3, text="Toyota", font=("Arial",12
,"bold")).grid(row=3,column=4)

    Test_menu = CTkOptionMenu(tframe3, variable=vehicle_var,
values=Toyota,command=select_from_toyota)

    Test_menu.grid(row=4,column=4,padx=8)

    # Date selection

    CTkLabel(tframe3, text="Select Pickup
Date:",font=("Arial",20,"bold")).grid(row=5,column=0,columnspan=2,pa
dy=20)

    date_entry = DateEntry(tframe3,width=40)

    date_entry.grid(row=6,column=0,columnspan=2)

    # Time selection

    CTkLabel(tframe3, text="Select Pickup
Time:",font=("Arial",20,"bold")).grid(row=5,column=3,columnspan=2,pa
dy=20)

    time_var = ctk.StringVar(value=generate_times()[0])

    time_combo = CTkOptionMenu(tframe3, variable=time_var,
values=generate_times(),width=200,height=20)

    time_combo.grid(row=6,column=3,columnspan=2)

```

```
# Confirm Button
confirm_btn = CtkButton(tframe3, text="Confirm Booking",
command=confirm_booking, font=("Arial", 20, "bold"))
confirm_btn.grid(row=9, column=2, pady=50)

#cancel button
ctk.CtkButton(tframe3, text="Cancel", command=lambda:
cancellation(root2), font=("Arial",
16, "bold")).grid(row=10, column=2, pady=10)

def buycar():
    root.withdraw()
    root1=ctk.CtkToplevel(root)
    root1.title("Car Buying Portal")
    root1.state("zoomed")

#sample data of list of cars

#1.Tata
mycur.execute("SELECT Car FROM cars WHERE company='Tata'")
z1 = mycur.fetchall()
# Extract car names as a flat list
Tata = [row[0] for row in z1]

#2.Suzuki
mycur.execute("SELECT Car FROM cars WHERE company='Suzuki'")
z2 = mycur.fetchall()
# Extract car names as a flat list
Suzuki = [row[0] for row in z2]

#3.Mahindra
mycur.execute("SELECT Car FROM cars WHERE company='Mahindra'")
z3 = mycur.fetchall()
# Extract car names as a flat list
Mahindra = [row[0] for row in z3]

#4.Hyundai
```

```
mycur.execute("SELECT Car FROM cars WHERE company='Hyundai'")
z4 = mycur.fetchall()
# Extract car names as a flat list
Hyundai = [row[0] for row in z4]

#5.Kia
mycur.execute("SELECT Car FROM cars WHERE company='Kia'")
z5 = mycur.fetchall()
# Extract car names as a flat list
Kia = [row[0] for row in z5]

#6.Citroen
mycur.execute("SELECT Car FROM cars WHERE company='Citroen'")
z3 = mycur.fetchall()
# Extract car names as a flat list
Citroen = [row[0] for row in z3]

#7.Volkswagen
mycur.execute("SELECT Car FROM cars WHERE
company='volkswagen'")
z7 = mycur.fetchall()
# Extract car names as a flat list
Volkswagen = [row[0] for row in z7]

#8.Skoda
mycur.execute("SELECT Car FROM cars WHERE company='Skoda'")
z8 = mycur.fetchall()
# Extract car names as a flat list
Skoda = [row[0] for row in z8]

#9.Honda
mycur.execute("SELECT Car FROM cars WHERE company='Honda'")
z8 = mycur.fetchall()
# Extract car names as a flat list
Honda = [row[0] for row in z8]
```

```
#10.Toyota
mycur.execute("SELECT Car FROM cars WHERE company='Toyota'")
z10 = mycur.fetchall()
# Extract car names as a flat list
Toyota = [row[0] for row in z10]

selected_company = ctk.StringVar(value="Tata")
def select_from_tata(car):
    selected_company.set("Tata")

def select_from_suzuki(car):
    selected_company.set("Suzuki")

def select_from_mahindra(car):
    selected_company.set("Mahindra")

def select_from_hyundai(car):
    selected_company.set("Hyundai")

def select_from_kia(car):
    selected_company.set("Kia")

def select_from_citroen(car):
    selected_company.set("Citroen")

def select_from_volkswagen(car):
    selected_company.set("Volkswagen")

def select_from_skoda(car):
    selected_company.set("Skoda")

def select_from_honda(car):
    selected_company.set("Honda")
```

```

def select_from_toyota(car):
    selected_company.set("Toyota")

colors = ["Red", "Blue", "Black", "White", "Silver"]
variants = ["Base", "Mid", "Top"]

payment_methods = ["Credit Card", "Debit Card", "Net Banking",
"UPI", "EMI"]

# variables
selected_vehicle = ctk.StringVar(value=" ")
selected_color = ctk.StringVar(value=colors[0])
selected_variant = ctk.StringVar(value=variants[0])
selected_payment = ctk.StringVar(value=payment_methods[0])

# Title
ctk.CTkLabel(root1, text="CAR BUYING FACILITY",
font=("Helvetica", 50, "bold")).pack(pady=10)

#creating a transparent frame to arrange vehices,colors and
variant

tframe1=CTkFrame(root1,width=400,height=100,fg_color="transparent")
tframe1.place(relx=0.5,rely=0.4,anchor='center')

#creating a transparent frame to arrange payment

tframe2=CTkFrame(root1,width=400,height=100,fg_color="transparent")
tframe2.place(relx=0.5,rely=0.8,anchor='center')

# Vehicle List
ctk.CTkLabel(tframe1, text="Select a Vehicle:", font=("Arial",
16)).grid(row=0,column=2)

ctk.CTkLabel(tframe1, text="Tata", font=("Arial",
12,"bold")).grid(row=1,column=0)

```



```
        vehicle_menu =
        ctk.CTkOptionMenu(tframe1, values=Tata, variable=selected_vehicle, command=select_from_tata)
        vehicle_menu.grid(row=2, column=0, padx=8)

        ctk.CTkLabel(tframe1, text="Suzuki", font=("Arial", 12, "bold")).grid(row=1, column=1)
        vehicle_menu =
        ctk.CTkOptionMenu(tframe1, values=Suzuki, variable=selected_vehicle, command=select_from_suzuki)
        vehicle_menu.grid(row=2, column=1, padx=8)

        ctk.CTkLabel(tframe1, text="Mahindra", font=("Arial", 12, "bold")).grid(row=1, column=2)
        vehicle_menu =
        ctk.CTkOptionMenu(tframe1, values=Mahindra, variable=selected_vehicle, command=select_from_mahindra)
        vehicle_menu.grid(row=2, column=2, padx=8)

        ctk.CTkLabel(tframe1, text="Hyundai", font=("Arial", 12, "bold")).grid(row=1, column=3)
        vehicle_menu =
        ctk.CTkOptionMenu(tframe1, values=Hyundai, variable=selected_vehicle, command=select_from_hyundai)
        vehicle_menu.grid(row=2, column=3, padx=8)

        ctk.CTkLabel(tframe1, text="Kia", font=("Arial", 12, "bold")).grid(row=1, column=4)
        vehicle_menu =
        ctk.CTkOptionMenu(tframe1, values=Kia, variable=selected_vehicle, command=select_from_kia)
        vehicle_menu.grid(row=2, column=4, padx=8)

        ctk.CTkLabel(tframe1, text="Citroen", font=("Arial", 12, "bold")).grid(row=3, column=0)
        vehicle_menu = ctk.CTkOptionMenu(tframe1, values=Citroen, variable=selected_vehicle, command=select_from_citroen)
        vehicle_menu.grid(row=4, column=0, padx=8)

        ctk.CTkLabel(tframe1, text="Volkswagen", font=("Arial", 12, "bold")).grid(row=3, column=1)
```

```

        vehicle_menu = ctk.CTkOptionMenu(tframe1, values=Volkswagen,
variable=selected_vehicle,command=select_from_volkswagen)
        vehicle_menu.grid(row=4, column=1, padx=8)

```

```

        ctk.CTkLabel(tframe1, text="Skoda", font=("Arial", 12,
"bold")).grid(row=3, column=2)
        vehicle_menu = ctk.CTkOptionMenu(tframe1, values=Skoda,
variable=selected_vehicle,command=select_from_skoda)
        vehicle_menu.grid(row=4, column=2, padx=8)

```

```

        ctk.CTkLabel(tframe1, text="Honda", font=("Arial", 12,
"bold")).grid(row=3, column=3)
        vehicle_menu = ctk.CTkOptionMenu(tframe1, values=Honda,
variable=selected_vehicle,command=select_from_honda)
        vehicle_menu.grid(row=4, column=3, padx=8)

```

```

        ctk.CTkLabel(tframe1, text="Toyota", font=("Arial", 12,
"bold")).grid(row=3, column=4)
        vehicle_menu = ctk.CTkOptionMenu(tframe1, values=Toyota,
variable=selected_vehicle,command=select_from_toyota)
        vehicle_menu.grid(row=4, column=4, padx=8)

```

```

#creating a transparent label
ctk.CTkLabel(tframe1, text=" ").grid(row=5,column=2)

```

```

# Color Options
ctk.CTkLabel(tframe1, text="Select color:", font=("Arial",
16)).grid(row=6,column=2)
n=0
for color in colors:
    clr=ctk.CTkRadioButton(tframe1, text=color,
variable=selected_color, value=color)
    clr.grid(row=7,column=n,pady=10)
    n=n+1

```

```

#creating a transparent label
ctk.CTkLabel(tframe1, text=" ").grid(row=8,column=2)

```

```

# Variant Options
    ctk.CTkLabel(tframe1, text="Select Variant:", font=("Arial",
16)).grid(row=9,column=2)
    variant_menu =
    ctk.CTkOptionMenu(tframe1,values=variants,variable=selected_variant)
    variant_menu.grid(row=10,column=2,pady=10)

#Final Price
def price():
    variant=selected_variant.get()
    company=selected_company.get()
    vehicle=selected_vehicle.get()
    if variant=="Base":
        price_column="Price1"
    elif variant=="Mid":
        price_column="Price2"
    elif variant=="Top":
        price_column="Price3"

    sql = f"SELECT {price_column} FROM cars WHERE Car = %s and
company= %s"
    mycur.execute(sql,(vehicle,company))
    row = mycur.fetchone()
    global price_value
    price_value = row[0] if row else "N/A"
    price_label.configure(text=f"Price: ₹{price_value}")

    obtain_price=CTkButton(tframe1,text="Obtain
Price",command=price)
    obtain_price.grid(row=11,column=2,pady=10)

price_box=CTkFrame(tframe1,border_width=2,border_color="green",height=35,width=200)
    price_box.grid(row=12,column=2,pady=10)
    price_label=CTkLabel(price_box,text=" ",font=("Arial", 16))
    price_label.place(anchor="center",relx=0.5,relx=0.5)

```

```

# Payment Options
ctk.CTkLabel(tframe2, text="Select Payment Method:",
font=("Arial", 16)).grid(row=0,column=2,pady=10)
m=0
for method in payment_methods:
    payment_=ctk.CTkRadioButton(tframe2, text=method,
variable=selected_payment, value=method)
    payment_.grid(column=m,row=1,padx=5)
    m=m+1

mycur.execute("SELECT Purchase_no FROM car_purchase ORDER BY
Purchase_no DESC LIMIT 1")
last_row = mycur.fetchone()
if last_row:
    a=last_row[0]+1
else:
    a=1

# Submit Function
def submit():
    summary = f"""
    vehicle: {selected_vehicle.get()}
    color: {selected_color.get()}
    variant: {selected_variant.get()}
    Payment Method: {selected_payment.get()}
    Price: ₹{price_value}
    Purchase number: {a}
    Kindly Remember the purchase number for later
clarification
    """
    messagebox.showinfo("Purchase Summary", summary)
    x=a
    x1=my_var.get()
    x2=selected_vehicle.get()
    x3=selected_color.get()
    x4=selected_variant.get()
    x5=selected_payment.get()

```

```

        x6=price_value
        insert = "INSERT INTO car_purchase (Purchase_no,phno,
car_name, color, variant, payment,Price)
VALUES({},'{}','{}','{}','{}','{}',{})".format(x,x1,x2,x3,x4,x5,x6)
        mycur.execute(insert)
        root.deiconify()      # Show the root window again
        root.state('zoomed')
        root1.destroy()

```

```

# Submit Button

```

```

        ctk.CTkButton(tframe2, text="Place Order",
command=submit,font=("Arial",
16,"bold")).grid(row=2,column=2,pady=10)

```

```

#cancel button

```

```

        ctk.CTkButton(tframe2, text="Cancel", command=lambda:
cancellation(root1),font=("Arial",
16,"bold")).grid(row=3,column=2,pady=10)

```

```

def carservice():

```

```

    root.withdraw()
    root3=CTkTopLevel(root)
    root3.state("zoomed")
    root3.title("Service Portal")
    def _add_entry(label_text):
        label = ctk.CTkLabel(form_frame, text=label_text,
font=("Arial", 16, "bold"))
        label.pack(pady=(10, 0))
        entry = ctk.CTkEntry(form_frame, width=400)
        entry.pack(pady=5)
        return entry

```

```

        mycur.execute("SELECT service_id FROM car_service ORDER BY
service_id DESC LIMIT 1")
        last_row = mycur.fetchone()
        if last_row:
            a=last_row[0]+1
        else:

```

```

a=1

def _add_widget(label_text, widget):
    label = ctk.CTkLabel(form_frame, text=label_text,
font=("Arial", 16, "bold"))
    label.pack(pady=(10, 0))
    widget.pack(pady=5)

def submit_form():
    x=my_var.get()
    model = model_entry.get()
    prob= problem.get()
    company=company_type.get()
    service=service_type.get()
    date=date_entry.get_date()
    if not all([model,prob]):
        messagebox.showwarning("Missing Info", "Please fill in
all fields.")
        return
    summary = f"Car model: {model}\nCompany:
{company}\nService Type: {service}\nProblem: {prob}\nService ID:
{a}\nKindly Remember the Service ID"
    messagebox.showinfo("Booking Confirmed", summary)
    insert="INSERT INTO car_service
(service_id,phno,car_name,company,service_type,problem,date)
VALUES({},'{}','{}','{}','{}','{}','{}')".format(a,x,model,company,s
ervice,prob,date)
    mycur.execute(insert)
    root.deiconify()
    root.state("zoomed")
    root3.destroy()

# Title Label
title_label = CTkLabel(root3, text="CAR SERVICE FACILITY",
font=("Arial", 50, "bold"))
title_label.pack(pady=20)
# Frame for input fields
form_frame = CTkFrame(root3,height=500,width=600)

```

```

form_frame.pack(pady=10, padx=20,)
form_frame.pack_propagate("False")

def validate_length(P):
    return len(P)<=30
vcmd=(form_frame.register(validate_length),'%P')

# Car Model Entry
model_entry = _add_entry("Car Model")
#car company entry
company_type=ctk.StringVar(value='Tata')

company=['Tata', 'Suzuki', 'Mahindra', 'Hyundai', 'Kia', 'Citroen', 'Volks
wagen', 'Skoda', 'Honda', 'Toyota']

company_dropdown=CTkOptionMenu(form_frame, values=company, variable=co
mpany_type)
    _add_widget("Choose the company", company_dropdown)


# Service Type Dropdown
service_type = ctk.StringVar(value="Oil Change")
service_dropdown = CTkOptionMenu(form_frame, values=["Oil
Change", "Brake Check", "Full Service", "Engine Light ON", "Wheel
Alignment", "Tyre Change", "Other"], variable=service_type)
    _add_widget("Service Type", service_dropdown)
# problem entry
problem =
CTkEntry(form_frame, validate='key', validatecommand=vcmd, width=400)
    _add_widget("Describe Your Problem in Less Than 30
words", problem)
# Date selection
date_entry = DateEntry(form_frame, width=40)
    _add_widget("Select Your Date", date_entry)


# Submit Button
submit_button = CTkButton(form_frame, text="Submit",
command=submit_form, font=("Arial", 16, "bold"))
submit_button.pack(pady=20)

```

```

cancel_button = CTKButton(form_frame,text="Cancel",
command=lambda: cancellation(root3),font=("Arial", 16, "bold"))
cancel_button.pack(pady=20)
root.mainloop()

```

```

def sub(): #function for receiving phno and pass

```

```

    a=e1.get()
    b=e2.get()
    select='SELECT * FROM user WHERE phno="{' OR email="{' AND
passwd="{'".format(a,a,b)
    mycur.execute(select)
    login=mycur.fetchone()

    if a=="00" and b=="admin":
        root.withdraw()
        admin()
    elif login!=None:
        print("successfull")
        #Destroys all widgets in a given frame
        for widget in root.winfo_children():
            widget.destroy()
        root.title("Selection")
        #if signed in with email obtaining the phone number
        phno_check=0
        d=my_var.get()
        if '@' in d:
            phno_check=1
        if phno_check==1:
            email_query="Select phno from user where email=%s"
            mycur.execute(email_query,(d,))
            row=mycur.fetchone()
            my_var.set(value=row[0])

```



```

def msgbox1():
    purchase_no=entry_purchase_no.get()
    mycur.execute("SELECT * FROM car_purchase where
Purchase_no={}".format(purchase_no))
    row=mycur.fetchone()
    userphno=my_var.get()
    if row is None:
        messagebox.showerror("Error","Invalid purchase
number")
    else:
        phno=row[1]
        if userphno==phno:
            car_name=row[2]
            color=row[3]
            variant=row[4]
            price=row[6]
            payment=row[5]
            summary = f"""
            Here is the information
            Vehicle: {car_name}
            Color: {color}
            Variant: {variant}
            Price: ₹{price}
            Payment Method: {payment}
            """
            messagebox.showinfo("Booking info",summary)
        else:
            messagebox.showerror("Error","Invalid
purchase number")
    entry_purchase_no.delete(0,'end')
def msgbox2():
    test_id=entry_test_id.get()
    mycur.execute("SELECT * FROM car_test where
Test_id={}".format(test_id))
    row=mycur.fetchone()
    userphno=my_var.get()
    if row is None:

```

```

number")
        messagebox.showerror("Error","Invalid purchase
else:
    phno=row[1]
    if userphno==phno:
        car_name=row[2]
        company=row[3]
        time=row[4]
        date=row[5]
        summary = f"""
        Here is the information
        Vehicle: {car_name}
        Company: {company}
        Time: {time}
        Date: {date}
        """
        messagebox.showinfo("Booking info",summary)
    else:
        messagebox.showerror("Error","Invalid Test
ID")

    entry_test_id.delete(0,'end')

def cancel_carbuy():
    userphno=my_var.get()
    purchase_no=purchase_no_entry_cancel.get()
    mycur.execute("SELECT phno FROM car_purchase where
Purchase_no={}".format(purchase_no))
    row=mycur.fetchone()
    if row is None:
        messagebox.showerror("Error","Invalid purchase
number")
    else:
        phno=row[0]
        if phno==userphno:
            response=messagebox.askokcancel("Cancel
purchase?","Are You Sure")
            if response:

```

```

                                mycur.execute("DELETE FROM
car_purchase WHERE Purchase_no={}".format(purchase_no))
                                else:
                                    messagebox.showerror("Error", "Purchase
not cancelled")
                                    purchase_no_entry_cancel.delete(0, 'end')
                                else:
                                    messagebox.showerror("Error", "Purchase_no
not valid")

```

```

def cancel_cartest():
    userphno=my_var.get()
    test_id=test_id_entry_cancel.get()
    mycur.execute("SELECT phno FROM car_test where
Test_id={}".format(test_id))
    row=mycur.fetchone()
    if row is None:
        messagebox.showerror("Error", "Invalid purchase
number")
    else:
        phno=row[0]
        if phno==userphno:
            response=messagebox.askokcancel("Cancel
Test Drive?", "Are You Sure")
            if response:
                mycur.execute("DELETE FROM car_test
WHERE Test_id={}".format(test_id))
            else:
                messagebox.showerror("Error", "Test
Drive not cancelled")
                test_id_entry_cancel.delete(0, 'end')
            else:
                messagebox.showerror("Error", "Test ID Not
valid")

```

```

def cancel_carservice():
    userphno=my_var.get()
    service_id=service_id_entry_cancel.get()

```

```

        mycur.execute("SELECT phno FROM car_service where
service_id={}").format(service_id))
        row=mycur.fetchone()
        if row is None:
            messagebox.showerror("Error","Invalid purchase
number")
        else:
            phno=row[0]
            if phno==userphno:
                response=messagebox.askokcancel("Cancel
service?","Are You Sure")
                if response:
                    mycur.execute("DELETE FROM car_service
WHERE service_id={}").format(service_id))
                else:
                    messagebox.showerror("Error","service
not cancelled")
                    service_id_entry_cancel.delete(0,'end')
            else:
                messagebox.showerror("Error","Service ID
Not Valid")

def search():
    car_name=search_price.get()
    car_name=car_name.title()
    mycur.execute("SELECT price1,price2,price3 FROM cars
WHERE car='{}'").format(car_name))
    row=mycur.fetchone()
    if row:
        price_summary=f'Base Price={row[0]}\nMid
Price={row[1]}\nTop Price={row[2]}'
        messagebox.showinfo("Magnus
Motors",price_summary)
    else:
        messagebox.showerror("Error","Car Not Found")

#creating a transparent frame to arrange widgets

tframe3=CTkFrame(root,fg_color="transparent")

```

```

tframe3.place(relx=0.52, rely=0.4, anchor='center')

CTkLabel(tframe3, text="Select An Option", font=("Arial",
50, "bold")).grid(row=0, column=1, pady=50)

carbuy=CTkButton(tframe3, text="Buy a
Car", command=buycar, height=40, font=("Arial",
16, "bold")).grid(row=1, column=0)

CTkLabel(tframe3, text="
", fg_color="transparent").grid(row=2, column=1)

seecarbook=CTkButton(tframe3, text="See Booked
Cars", command=msgbox1, height=30, font=("Arial",
16, "bold")).grid(row=4, column=0, pady=5)

entry_purchase_no=CTkEntry(tframe3, placeholder_text="Enter
Purchase No.")

entry_purchase_no.grid(row=3, column=0, pady=5)

cartest=CTkButton(tframe3, text="Book a Test
Drive", command=testcar, height=40, font=("Arial",
16, "bold")).grid(row=1, column=2)

seetestbook=CTkButton(tframe3, text="See Booked Test
Drive", command=msgbox2, height=30, font=("Arial",
16, "bold")).grid(row=4, column=2, pady=5)

entry_test_id=CTkEntry(tframe3, placeholder_text="Enter
Test ID")

entry_test_id.grid(row=3, column=2, pady=5)

car_service=CTkButton(tframe3, text="Book
Service", command=carservice, height=40, font=("Arial",
20, "bold")).grid(row=1, column=1, pady=10)

CTkLabel(tframe3, text="
", fg_color="transparent").grid(row=5, column=1)

CTkLabel(tframe3, text='Cancel Car Booking', font=("Arial",
16, "bold")).grid(row=6, column=0, pady=10)

purchase_no_entry_cancel=CTkEntry(tframe3, placeholder_text="Enter
Purchase No.")

purchase_no_entry_cancel.grid(row=7, column=0)

CTkButton(tframe3, text="confirm", command=cancel_carbuy, height=30, fon
t=("Arial", 16, "bold")).grid(row=8, column=0, pady=5)

```

```

        CTKLabel(tframe3,text='Cancel Test Drive
Booking',font=("Arial", 16,"bold")).grid(row=6,column=2,pady=10)

test_id_entry_cancel=CTKEntry(tframe3,placeholder_text="Enter Test
ID")

        test_id_entry_cancel.grid(row=7,column=2)

CTKButton(tframe3,text="confirm",command=cancel_cartest,height=30,fo
nt=("Arial", 16,"bold")).grid(row=8,column=2,pady=5)


        CTKLabel(tframe3,text='Cancel Service
Booking',font=("Arial", 16,"bold")).grid(row=6,column=1,pady=10)

service_id_entry_cancel=CTKEntry(tframe3,placeholder_text="Enter
Service ID")

        service_id_entry_cancel.grid(row=7,column=1)

CTKButton(tframe3,text="confirm",command=cancel_carservice,height=30
,font=("Arial", 16,"bold")).grid(row=8,column=1,pady=5)


        CTKLabel(root,text='',image=namelogo_img).place(relx=0.5,
rely=1.0, anchor='s')

        back_btn=CTKButton(root,text="<--
",command=signin,font=("Arial", 20,"bold")).place(anchor='nw')

        CTKLabel(tframe3,text="
",fg_color="transparent").grid(row=9,column=1)

        search_price=CTKEntry(tframe3,placeholder_text="Search a
Car's Price",height=40,border_width=2,border_color="green")

        search_price.grid(row=10,column=1,pady=5)

CTKButton(tframe3,command=search,text="Search").grid(row=11,column=1
,pady=5)


exit_btn=CTKButton(tframe3,text="Exit",command=destroy,height=40,fon
t=("Arial",
20,"bold"),fg_color='#990000',hover_color="#670101").grid(row=12,col
umn=1,pady=50)

        else:

            messagebox.showinfo("Error","Incorrect Phone Number,Email
or Password")

```

```

def creation(): #new registering window for new comers
    root.withdraw()
    window1=ctk.CTkToplevel(root)
    window1.state('zoomed')
    window1.title("Create An Account")

def destroy1(): #cancelling window fnc
    root.deiconify()      # Show the root window again
    root.state('zoomed')
    window1.destroy()

def go_back():
    x1=ephno.get()
    x2=epasswd.get()
    x3=efname.get()
    x4=elname.get()
    x5=eaddress.get()
    x6=esec_phno.get()
    x7=eemail.get()
    x='insert into
user(phno,passwd,First_Name,Last_Name,address,sec_phno,email)
values("{}","{}","{}","{}","{}","{}","{}")'.format(x1,x2,x3,x4,x5,x6
,x7)

    mycur.execute(x)
    window1.destroy()      # Close the new window
    root.deiconify()      # Show the root window again
    root.state('zoomed')

tframe=CTkFrame(window1,width=400,height=200,fg_color="transparent")
#transparent frame created to make the inner widgets into one unit
    tframe.place(relx=0.5,rely=0.3,anchor="center")

btn_back=CTkButton(tframe,text="Submit",command=go_back).grid(column
=1,row=7,sticky='n')

```

```
btn_cancel=CTkButton(tframe,text="Cancel",command=destroy1).grid(column=2,row=7,sticky='wn')
```

```
lphno=CTkLabel(tframe,text="Phone Number",font=f)
lphno.grid(column=0,row=0,padx=5,pady=5,sticky='e')
ephno=CTkEntry(tframe)
```

```
ephno.grid(column=1,row=0,columnspan=2,padx=5,pady=5,sticky='we')
```

```
lpasswd=CTkLabel(tframe,text="Password",font=f)
lpasswd.grid(column=0,row=1,padx=5,pady=5,sticky='e')
epasswd=CTkEntry(tframe)
```

```
epasswd.grid(column=1,row=1,columnspan=2,padx=5,pady=5,sticky='we')
```

```
lfname=CTkLabel(tframe,text="First Name:",font=f)
lfname.grid(column=0,row=2,padx=5,pady=5,sticky='e')
efname=CTkEntry(tframe)
```

```
efname.grid(column=1,row=2,columnspan=2,padx=5,pady=5,sticky='we')
```

```
llname=CTkLabel(tframe,text="Last Name:",font=f)
llname.grid(column=0,row=3,padx=5,pady=5,sticky='e')
elname=CTkEntry(tframe)
```

```
elname.grid(column=1,row=3,columnspan=2,padx=5,pady=5,sticky='we')
```

```
laddress=CTkLabel(tframe,text="Address:",font=f)
laddress.grid(column=0,row=4,padx=5,pady=5,sticky='e')
eaddress=CTkEntry(tframe)
```

```
eaddress.grid(column=1,row=4,columnspan=2,padx=5,pady=5,sticky='we')
```

```
lemail=CTkLabel(tframe,text="email:",font=f)
lemail.grid(column=0,row=5,padx=5,pady=5,sticky='e')
email=CTkEntry(tframe)
```



```

email.grid(column=1,row=5,columnspan=2,padx=5,pady=5,sticky='we')

lsec_phno=CTkLabel(tframe,text="Secondary Phno:",font=f)
lsec_phno.grid(column=0,row=6,padx=5,pady=5,sticky='e')
esec_phno=CTkEntry(tframe)

esec_phno.grid(column=1,row=6,columnspan=2,padx=5,pady=5,sticky='we'
)
def signin():
    root.title("Sign Up")
    global my_var
    my_var=ctk.StringVar()

    for widget in root.winfo_children():
        widget.destroy()

    fre=CTkFrame(root,width=400,height=350)
    fre.place(relx=0.5,rely=0.5,anchor='center')
    #transparent frame created to make the inner widgets into one
unit
    fr=CTkFrame(fre,width=400,height=200,fg_color="transparent")
    fr.place(anchor='center',relx=0.5,rely=0.4)

    logo_img=CTkLabel(root,text=' ',image=logo)
    logo_img.place(anchor='n',relx=0.5)

    global e1
    global e2
    l1=CTkLabel(fr,text="Phno or Email:",font=f)
    l1.grid(column=0,row=0,columnspan=2,padx=5,pady=5,sticky='wes')
    e1=CTkEntry(fr,textvariable=my_var)
    e1.grid(column=0,row=1,columnspan=2,padx=5,pady=5,sticky='wse')

    l2=CTkLabel(fr,text="Password:",font=f)

```

```
l2.grid(column=0,row=2,columnspan=2,padx=5,pady=5,sticky='wen')
e2=CTkEntry(fr,show='*')
e2.grid(column=0,row=3,columnspan=2,padx=5,pady=5,sticky='wne')

b1=CTkButton(fr,text="Sign In",command=sub,font=("Arial",
16,"bold")).grid(column=0,row=4,sticky='n',padx=5)

b2=CTkButton(fr,text="Exit",command=destroy,font=("Arial",
16,"bold"),fg_color='#990000',hover_color="#670101").grid(column=1,r
ow=4,sticky='n',padx=5)

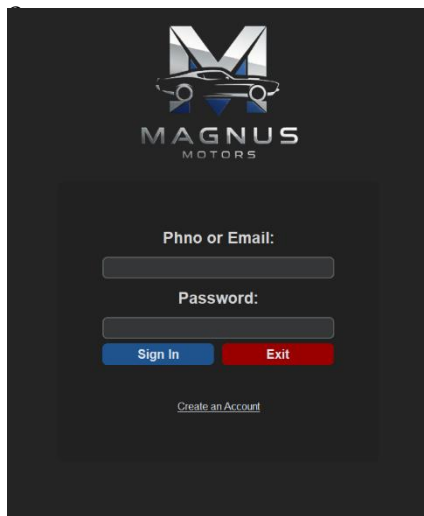
signup=CTkButton(fre,text="Create an
Account",command=creation,fg_color="transparent",font=('Arial',13,'u
nderline'))

signup.place(anchor='center',relx=0.5,rely=0.8)

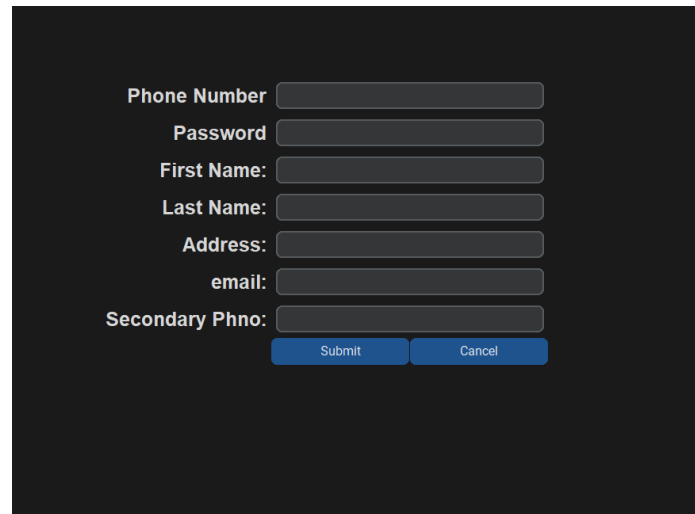
signin()
root.mainloop()
mycon.commit()
mycon.close()
```

Output

User



The login form features the Magnus Motors logo at the top. Below it, there are two input fields: 'Phno or Email:' and 'Password:'. Each field has a corresponding text input box. At the bottom of the form, there are two buttons: 'Sign In' (blue) and 'Exit' (red). A link 'Create an Account' is located below the 'Exit' button.



The registration form contains several input fields for user details: 'Phone Number', 'Password', 'First Name:', 'Last Name:', 'Address:', 'email:', and 'Secondary Phno:'. Each field is followed by a text input box. At the bottom right, there are two buttons: 'Submit' (blue) and 'Cancel' (blue).

User Login and Registration



The home screen is titled 'Select An Option'. It features a grid of buttons for various actions: 'Buy a Car', 'Book Service', 'Book a Test Drive', 'See Booked Cars', 'See Booked Test Drive', 'Cancel Car Booking', 'Cancel Service Booking', and 'Cancel Test Drive Booking'. Each button is accompanied by an input field for a specific ID or number. At the bottom, there is a 'Search' button and an 'Exit' button. The Magnus Motors logo is displayed at the very bottom.

Home Screen

CAR BUYING FACILITY

Select a Vehicle:

Tata	Suzuki	Mahindra	Hyundai	Kia
Harrier	Harrier	Harrier	Harrier	Harrier
Citroen	Volkswagen	Skoda	Honda	Toyota
Harrier	Harrier	Harrier	Harrier	Harrier

Select Color:

☐ Red ☐ Blue ☒ Black ☐ White ☐ Silver

Select Variant:

Top

Obtain Price

Price: ₹2000000

Select Payment Method:

☐ Credit Card ☐ Debit Card ☒ Net Banking ☐ UPI ☐ EMI

Place Order

Cancel

Car Buying

Select An Option

Buy a Car

10

See Booked Cars

Cancel Car Booking

Enter Purchase No.

confirm

Book Service

Booking info

Here is the information
Vehicle: Harrier
Color: Black
Variant: Top
Price: ₹2000000
Payment Method: Net Banking

OK

Book a Test Drive

Enter Test ID

See Booked Test Drive

Cancel Test Drive Booking

Enter Test ID

confirm

Exit

MAGNUS
MOTORS

Viewing Booked Cars

CAR SERVICE FACILITY

Car Model

Choose the company

Mahindra ▾

Service Type

Full Service ▾

Describe Your Problem in Less Than 30 Words

Select Your Date

9/17/25 ▾

Submit

Cancel

Booking Service

TEST DRIVE BOOKING FACILITY

Select a Vehicle:

Tata	Suzuki	Mahindra	Hyundai	Kia
Golf GTI	Golf GTI	Golf GTI	Golf GTI	Golf GTI
Citoren	Volkswagen	Skoda	Honda	Toyota
Golf GTI	Golf GTI	Golf GTI	Golf GTI	Golf GTI

Select Pickup Date: 9/19/25

Select Pickup Time: 15:30

Confirm Booking

Cancel

Test Drive Booking

Select An Option

Buy a Car

Enter Purchase No.

See Booked Cars

Cancel Car Booking

Enter Purchase No.

confirm

Book Service

5

See Booked Test Drive

Cancel Test Drive Booking

Enter Test ID

confirm

Booking info

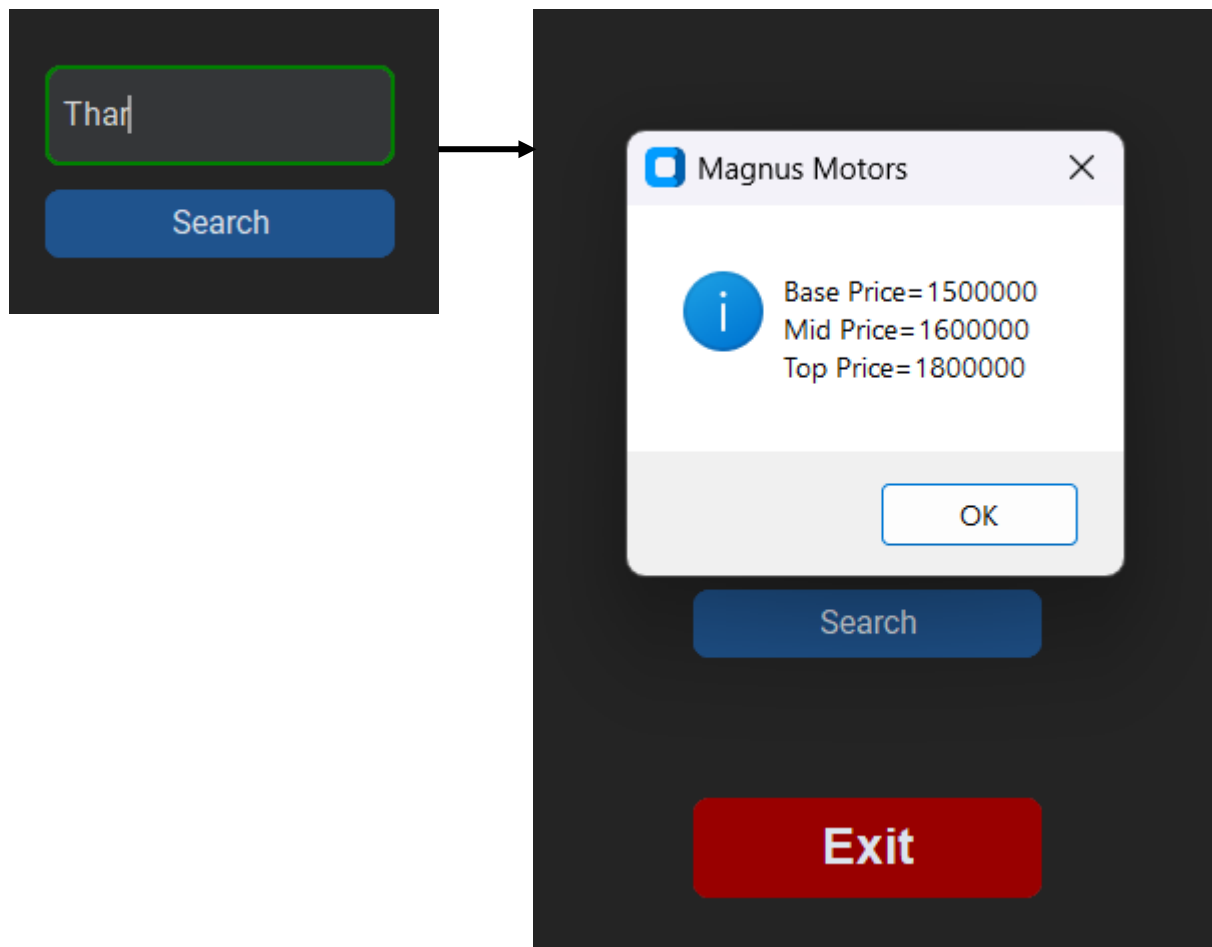
Here is the information
Vehicle: Golf GTI
Company: Volkswagen
Time: 15:30
Date: 2025-09-19

OK

Exit

MAGNUS
MOTORS

Viewing Booked Test Drive



Searching For Price Range of a Desired Car

Admin



Admin Home Page

	Purchase Number	Phone number	Car Name	Colour	Variant	Payment Method	Price
1	6282182971	Slavia	Blue	Top	EMI	2000000	
2	6282182971	XUV 700	Black	Top	Net Banking	2400000	
3	6282182971	Swift	Red	Mid	Net Banking	760000	
5	9400627033	I20	Black	Top	Net Banking	1500000	
6	6282182971	Jimmy	Blue	Mid	Debit Card	1400000	
7	6282182971	Brezza	Black	Mid	Net Banking	1270000	
8	6282182971	Virtus	Black	Top	Net Banking	2000000	
9	8237481938	Thar	White	Top	Credit Card	1800000	
10	6282182971	Harrier	Black	Top	Net Banking	2000000	

Load/Refresh Data

Car Delivery Complete

Purchase Number

confirm

Car Booking Cancel

Purchase Number

confirm

Accessing Car Booking

	Car Number	Car Name	Price 1	Price 2	Price 3
19	Carnival	2500000	2800000	3000000	
20	Seltos	1400000	1600000	1700000	
21	Sonnet	1100000	1200000	1400000	
22	Carens	1300000	1500000	1700000	
23	EV 6	2000000	2500000	2900000	
24	EV 9	3200000	3800000	4000000	
25	Carens Clavis	1400000	1700000	2000000	

Kia

Load/Refresh Data

Delete Car

Add Car

Edit Price

Car Number

Car Name

Car Number

confirm

price 1

Choose Which Price

price 2

Enter Price

price 3

confirm

confirm

Car List

	Test ID	Phone number	Car Name	Company	Time	Date
1		6282182971	I20	hyundai	11:00	2025-08-13
4		8237481938	Eeco	Suzuki	12:00	2025-09-12
5		6282182971	Golf GTI	Volkswagen	15:30	2025-09-19

Load/Refresh Data

Test Drive complete

Test ID

confirm

Cancel Test Drive Booking

Test ID

confirm

Accessing Test Drive Booking

	Service ID	Phone number	Car Name	Company	Service Type	Problem	Date
1		6282182971	Thar	Mahindra	Full Service	Need full service	2025-08-14
2		6282182971	Thar	Mahindra	Other	4 wheel drive not working	2025-08-15
3		6282182971	Swift	Suzuki	Brake Check	Brake not working	2025-08-10
4		6282182971	Thar Roxx	Mahindra	Full Service	Full Service	2025-09-17

Load/Refresh Data

Car Service Complete

Service ID

confirm

Car Service Cancel

Service ID

confirm

Accessing Service Booking

User Manual

Registration

- Click **Create an Account** on the login screen.
- Fill in phone number, password, name, address, email, and secondary phone (optional).
- Click **Submit** to register.

Login

- Enter phone number or email and password.
- Click **Sign In** to access the main dashboard.
- Admin login: Phone 00, Password admin.

Main Dashboard

- **Buy a Car:** Choose car brand, model, color, variant, and payment method. Place the order and note purchase number.
- **Book a Test Drive:** Select vehicle, date, and time. Confirm and save the test ID.
- **Book Service:** Provide car model, service type, describe issues, and select service date. Save the service ID.
- **View Bookings:** Check car purchases or test drives by entering their IDs.
- **Cancel Bookings:** Cancel car purchase, test drive, or service using corresponding IDs.
- **Search Car Prices:** Enter car name to view base, mid, and top-tier prices.
- **Exit:** Close the application

Admin Panel Features

- Manage car purchases (view, confirm delivery, cancel).
- Manage car inventory (add, edit prices, delete cars).
- Manage test drive bookings (view, complete, cancel).
- Manage service bookings (view, complete, cancel).

Important: Always note down the unique numbers (purchase number, test id and service id)

Conclusion

I would like to express my heartfelt gratitude to everyone who has supported me throughout this project. A special thanks to my family for their endless encouragement and understanding, and to my friends for their constant motivation, help and feedback which helped me to make the best version of this project. I am deeply grateful to my teachers, whose guidance and knowledge have been invaluable in completing this work. I also thank my school for providing the resources and environment that fostered my learning and growth. Finally, I extend my appreciation to all who contributed directly or indirectly to making this project a reality. This achievement would not have been possible without your support.

References