# 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='asher123',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.Heading",

background="#444444", # Slightly lighter gray for header

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.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\_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.winfo\_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','Volkswagen','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.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\_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.winfo\_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="Acces 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,time,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,pady=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,pady=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,rely=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,service,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','Volkswagen','Skoda','Honda','Toyota']

company\_dropdown=CTkOptionMenu(form\_frame,values=company,variable=company\_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:

messagebox.showerror("Error","Invalid purchase number")

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,font=("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,font=("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,font=("Arial", 20,"bold"),fg\_color='#990000',hover\_color="#670101").grid(row=12,column=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')

eemail=CTkEntry(tframe)

eemail.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,row=4,sticky='n',padx=5)

signup=CTkButton(fre,text="Create an Account",command=creation,fg\_color="transparent",font=('Arial',13,'underline'))

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

signin()

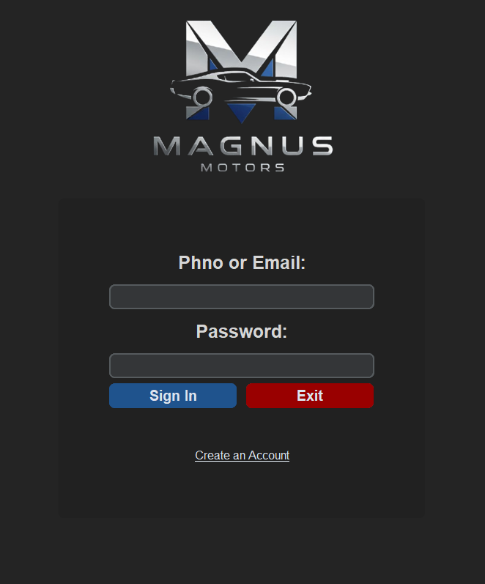
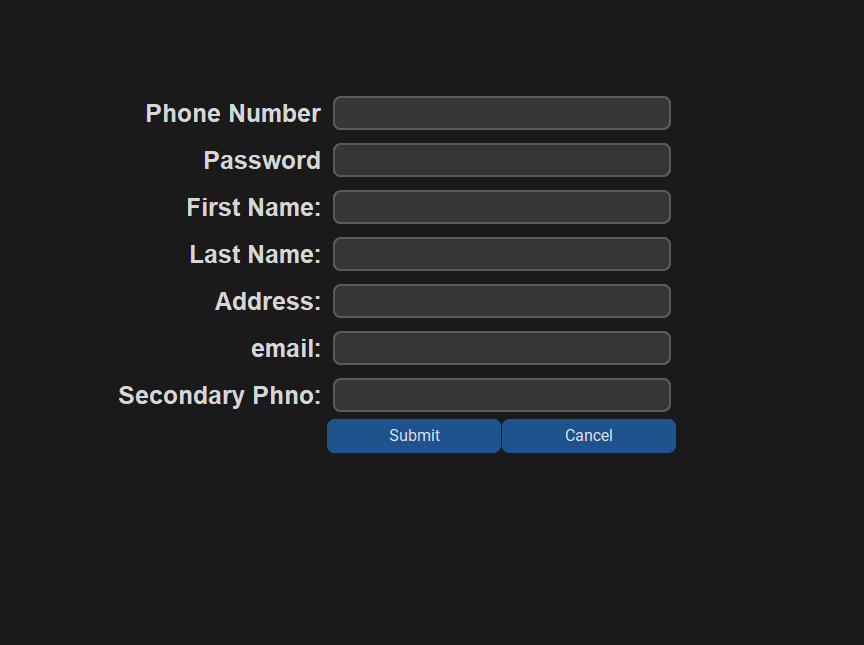
root.mainloop()

mycon.commit()

mycon.close()

# Output

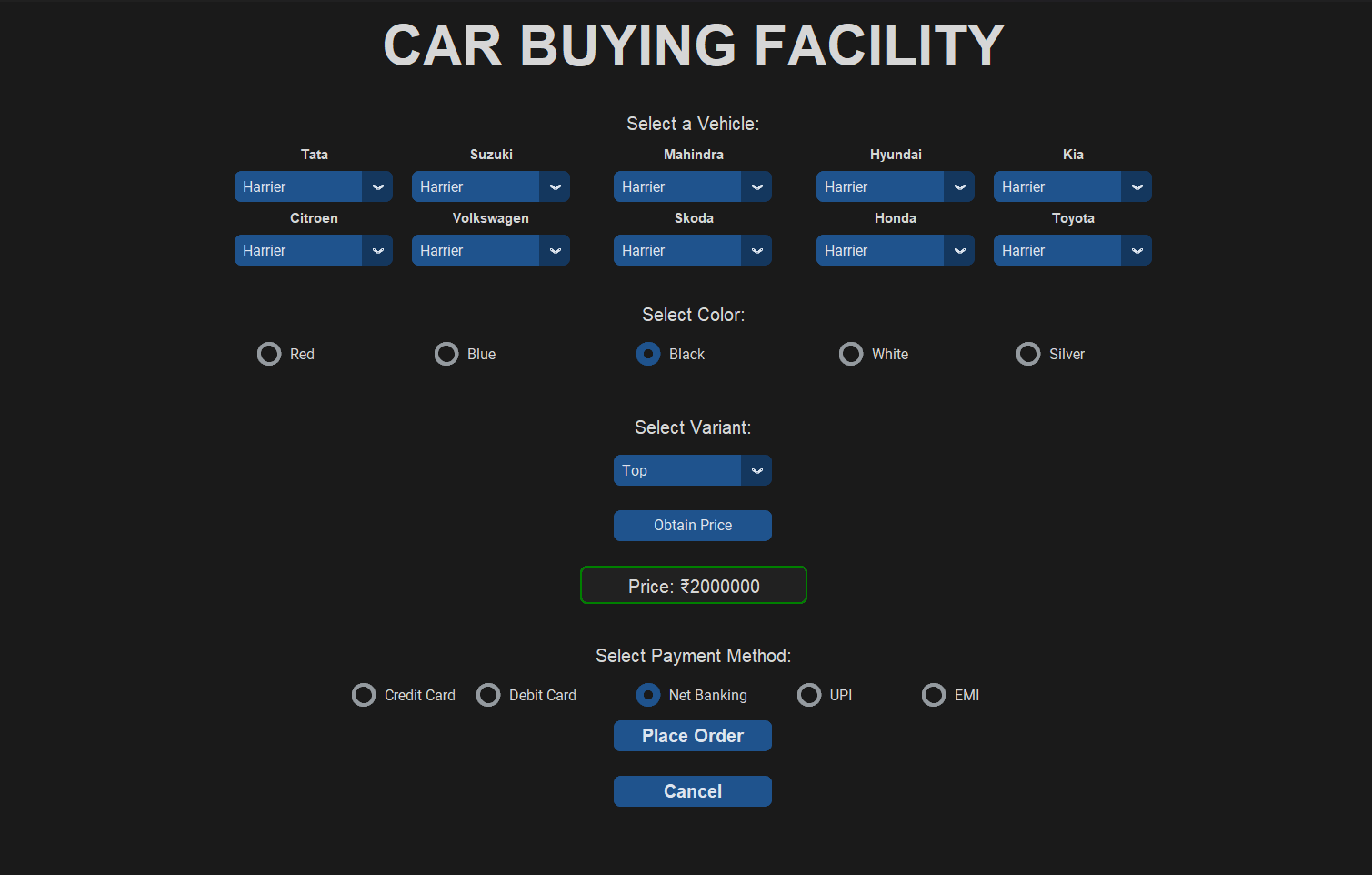
## User

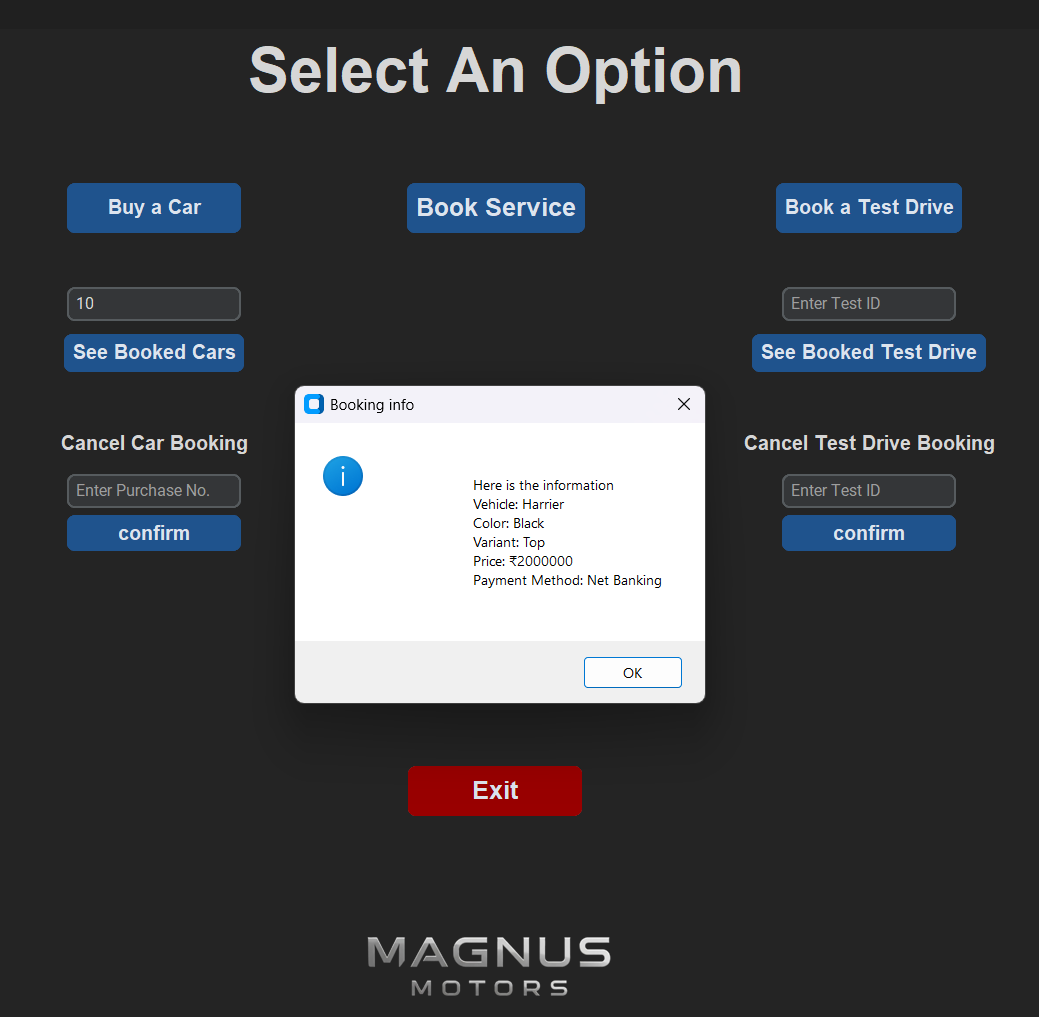
o



**User Login and Registration**

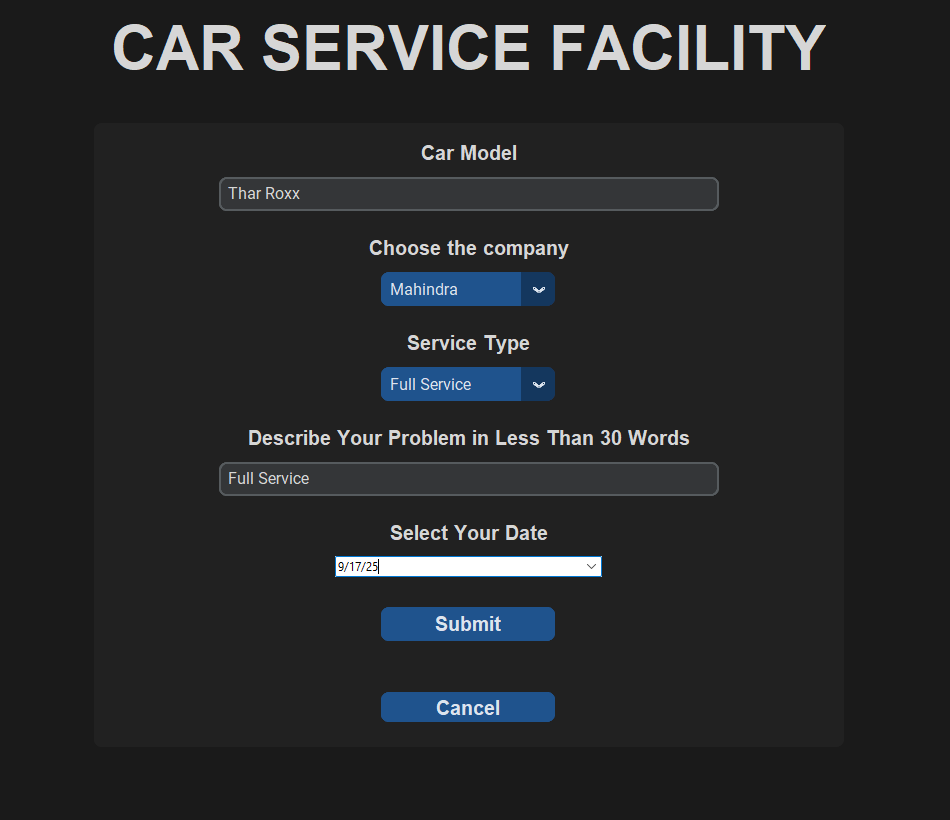
**Home Screen**



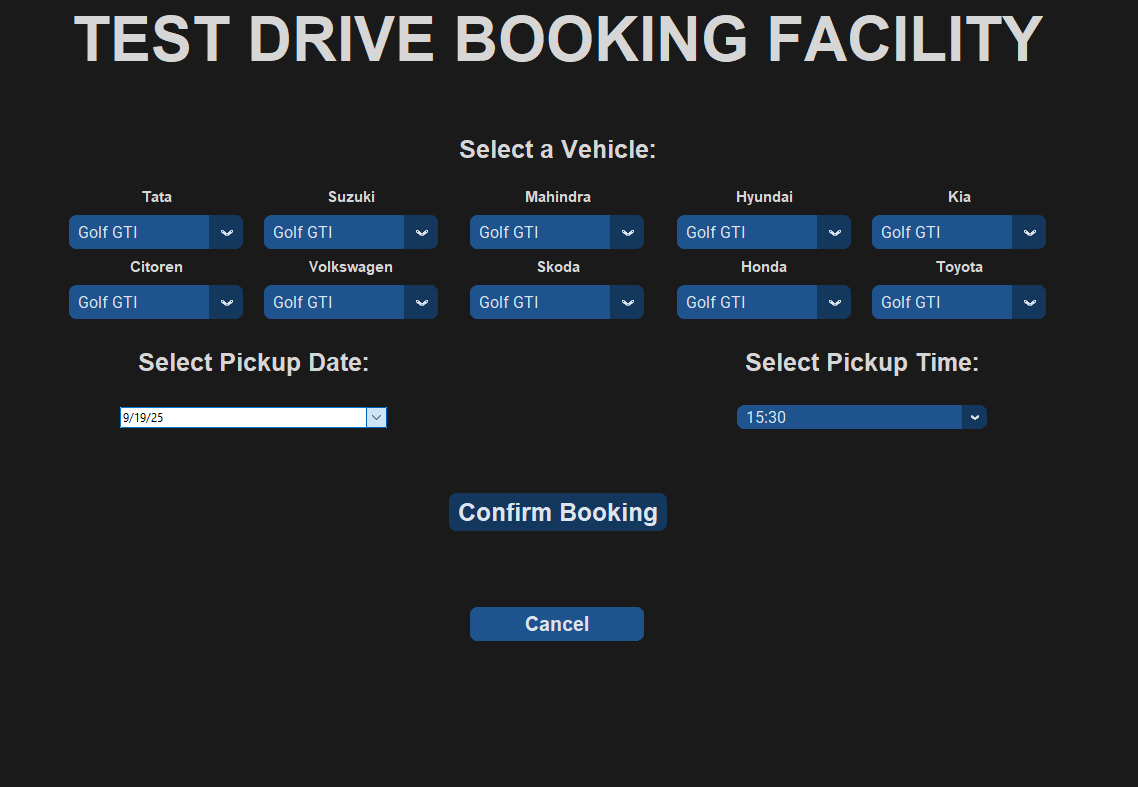


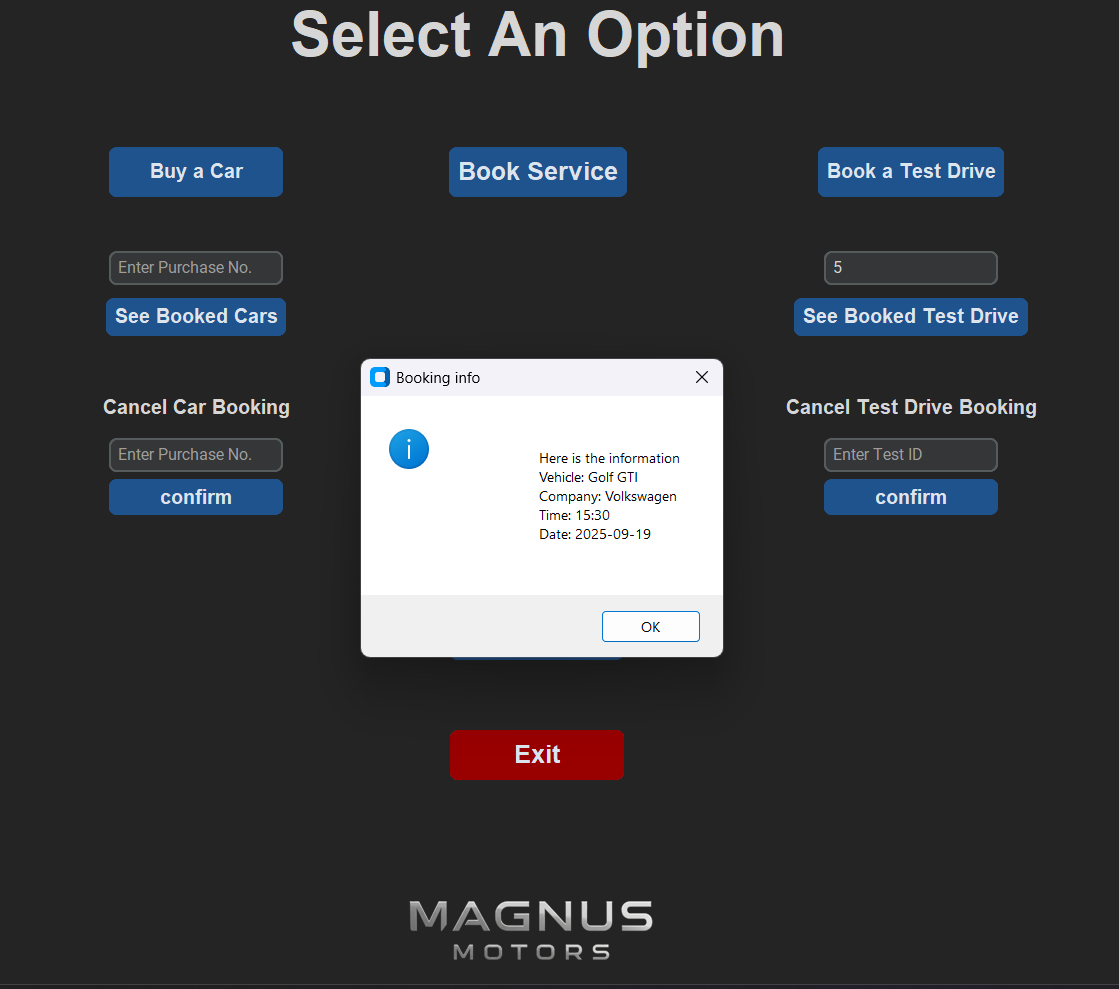
**Viewing Booked Cars**

**Car Buying**



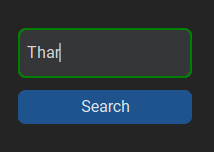
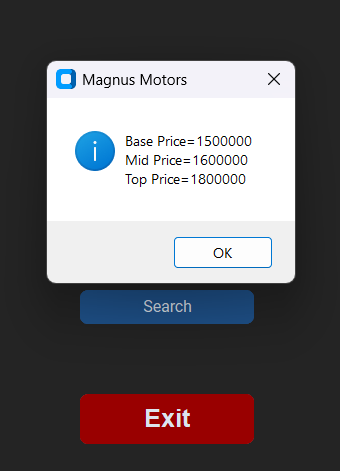
**Booking Service**





**Viewing Booked Test Drive**

**Test Drive Booking**

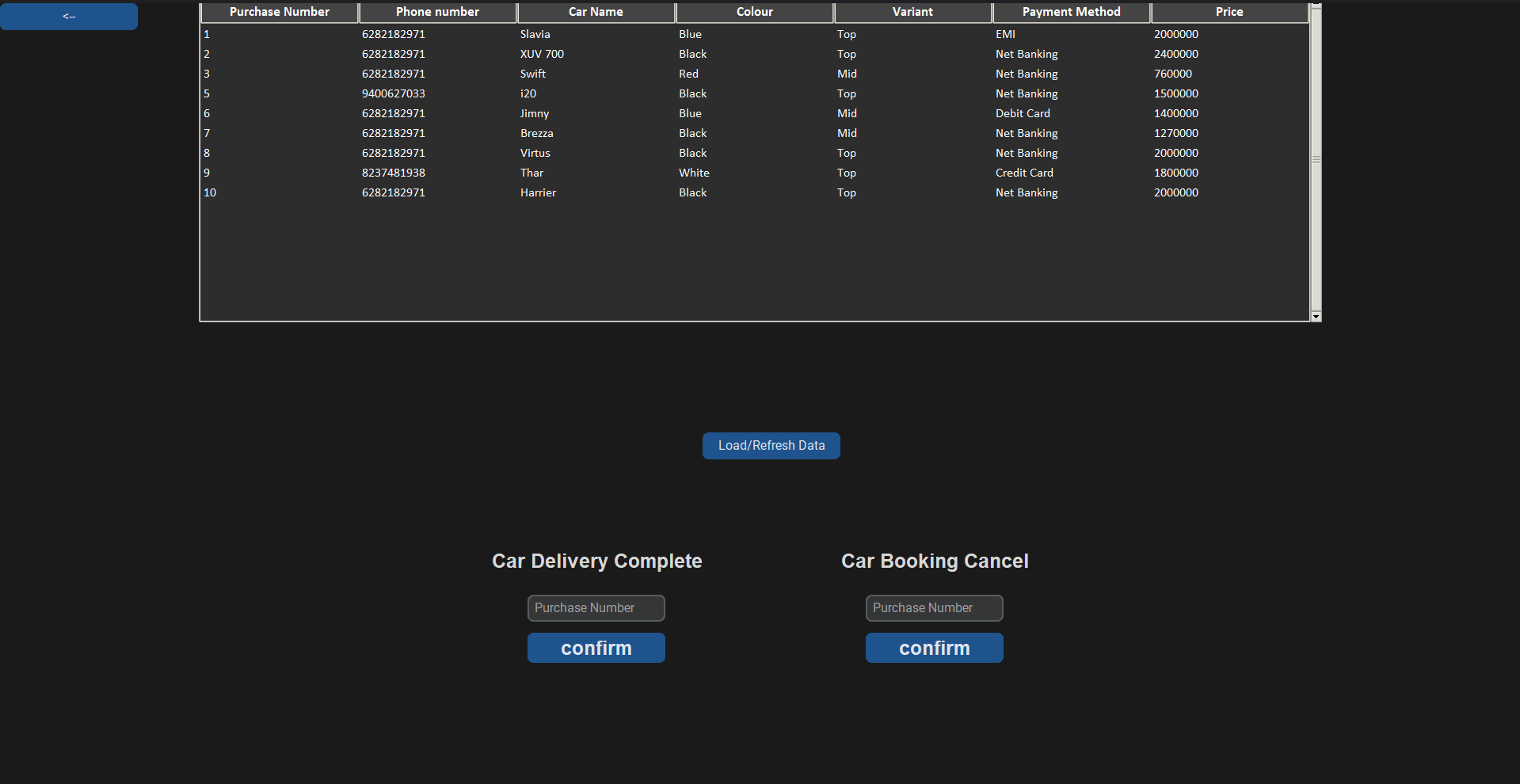


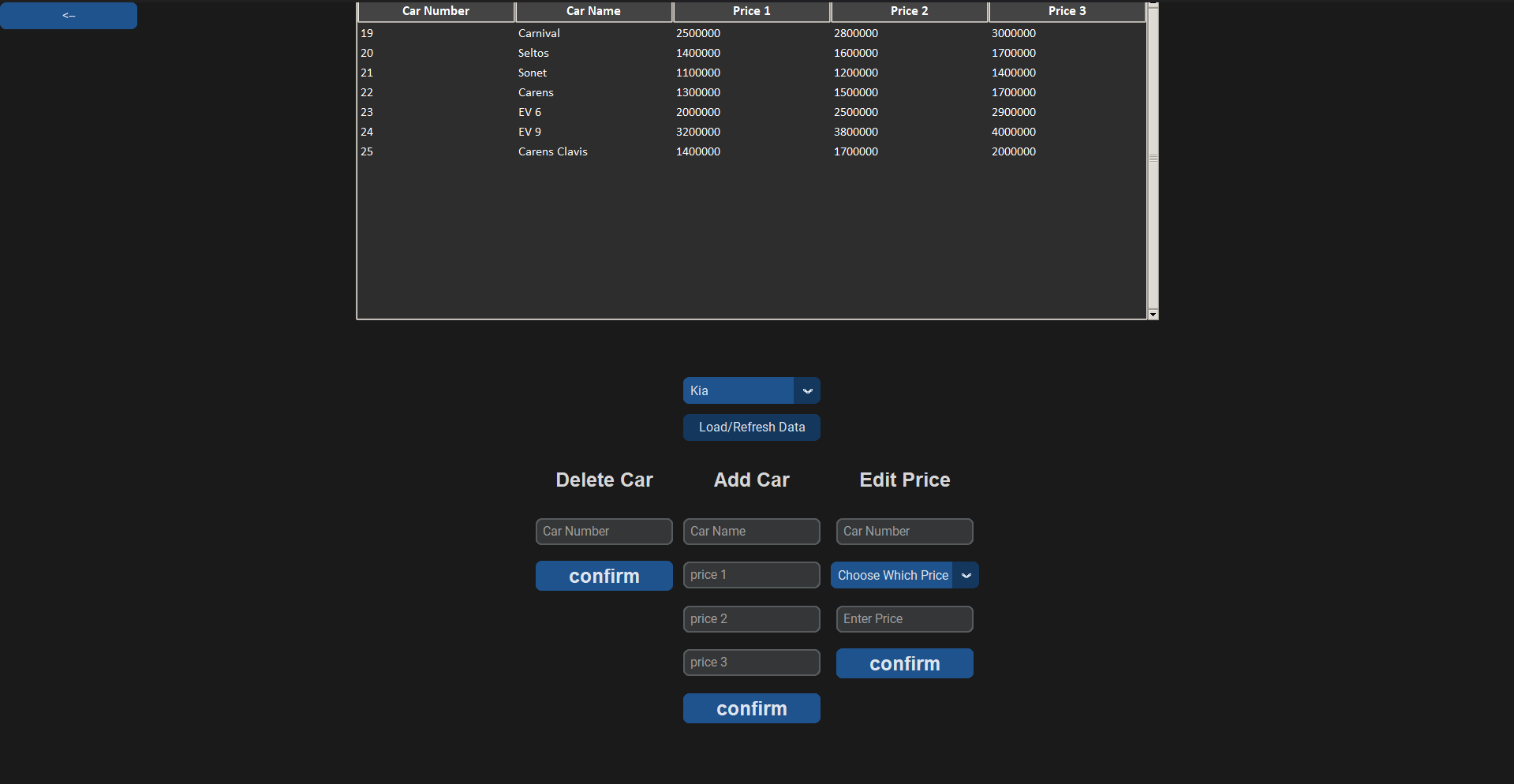
**Searching For Price Range of a Desired Car**

## Admin



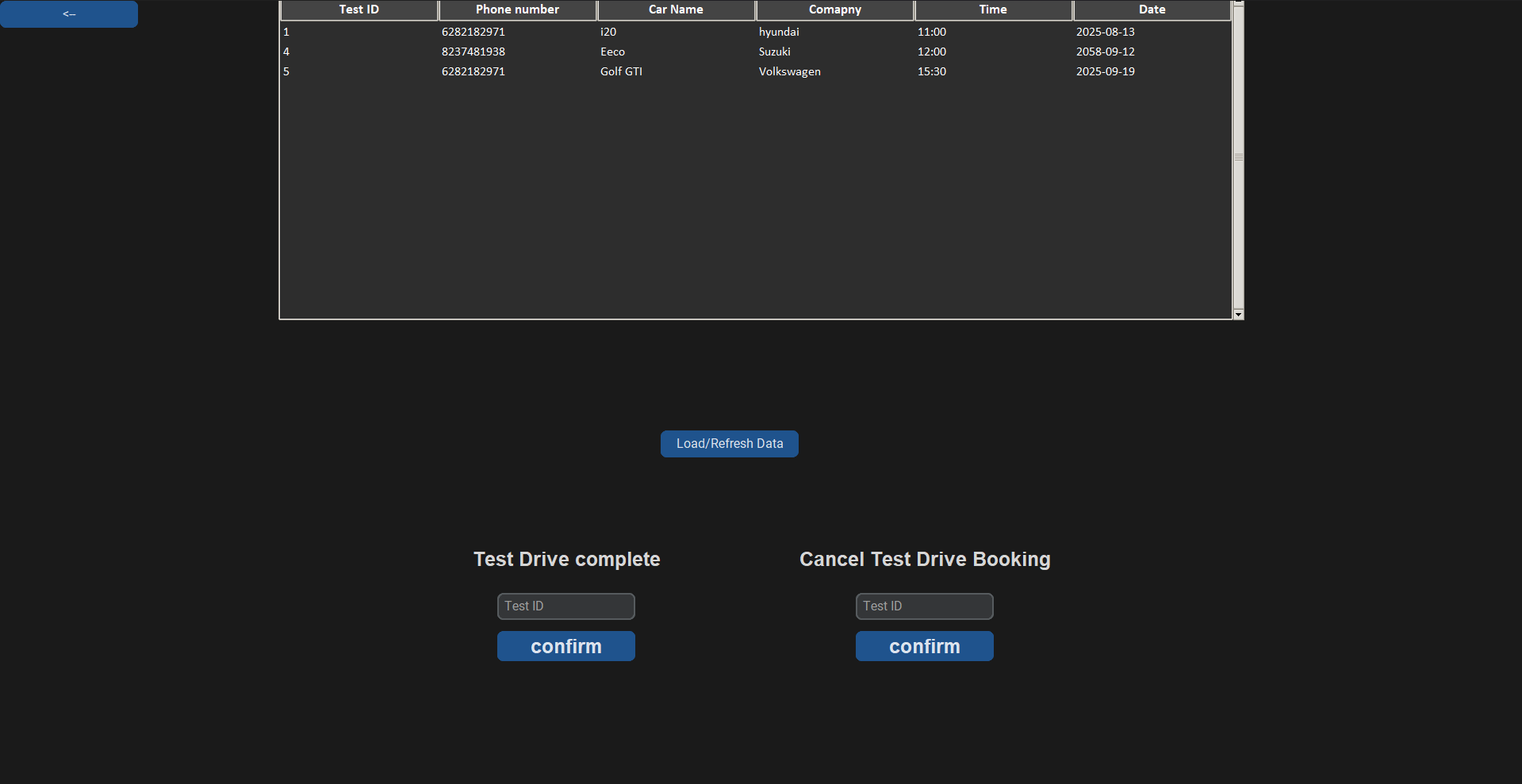
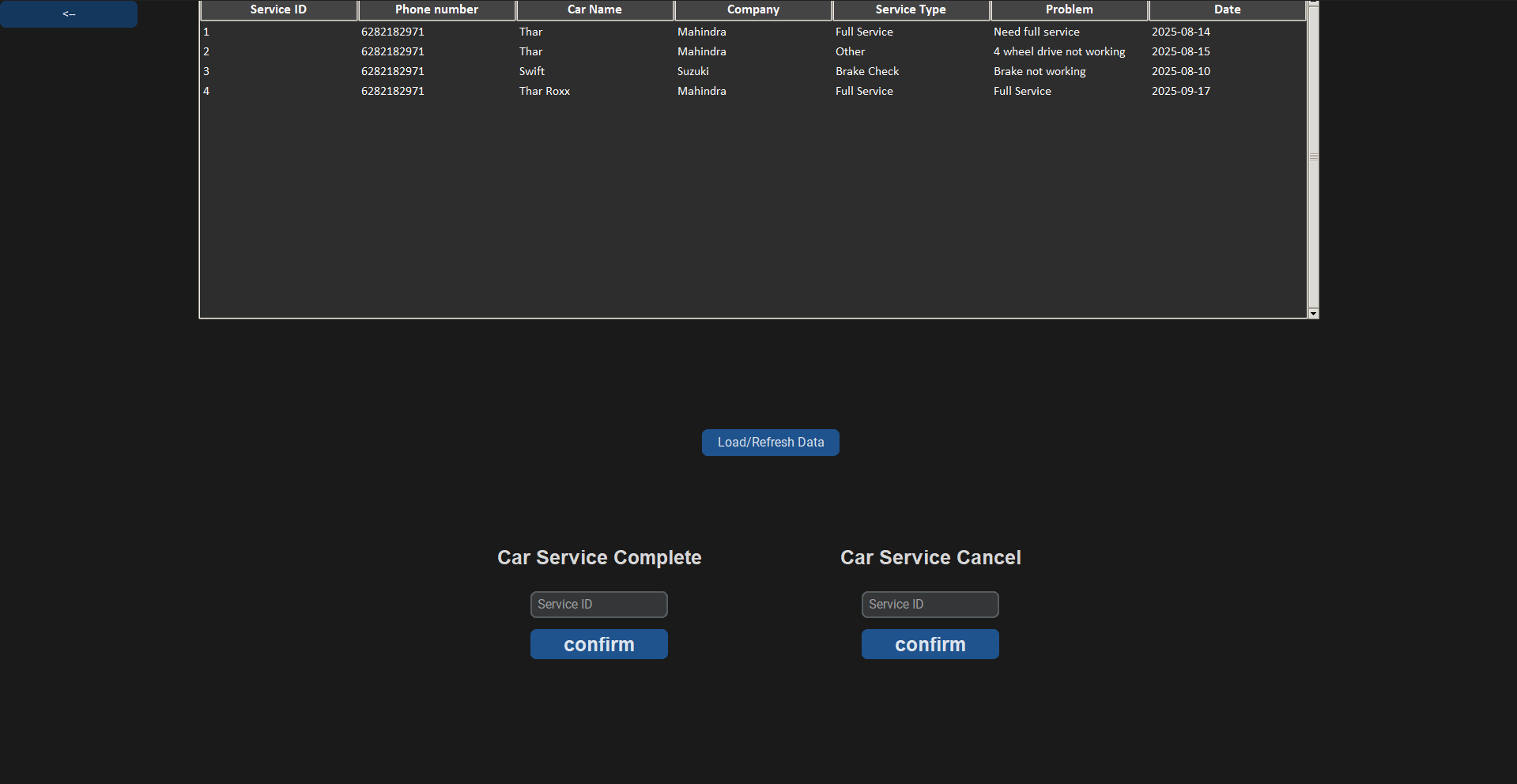
**Admin Home Page**





**Car List**

**Accessing Car Booking**



**Accessing Service Booking**

**Accessing Test Drive 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