**A Project (CS405PC) Report**

on

**“CAR SHOWROOM DEALERSHIP”**

Submitted

in the partial fulfilment of the requirements for the award of the degree of

**Bachelor of Technology**

in

**Computer Science and Engineering**

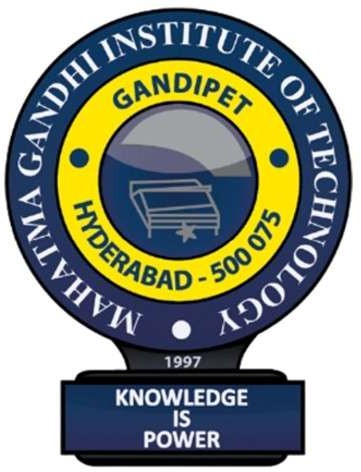
by

**NAIKA NAVANEETH**

**H.T. No: 22261A0535 SHIKA UPNDER**

**H.T. No: 22261A0543**

Under the guidance of **Ms. M. Navya Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING MAHATMA GANDHI INSTITUTE OF TECHNOLOGY**

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

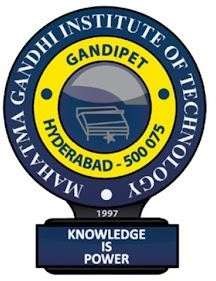
Gandipet, Hyderabad-500075, Telangana (India) [2023-2024]

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad) GANDIPET,

HYDERABAD-500075.Telangana

# CERTIFICATE



This is to certify that the project report (CS802PC) entitled **“Car Showroom Dealership”** is being submitted by **NAIKA NAVAEETH, SHIKA UPENDER** bearing Roll No: **22261A0535,22261A0543** in the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bonafide work carried out under the supervision of **Ms. M. Navya Assistant Professor, Dept. of CSE.** The design and results of the project enclosed in this report have been verified and found satisfactory.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor Head of the Department

**Ms. M. Navya Dr. C.R.K. Reddy**

Asst. Professor, Dept. of CSE Professor, Dept. of CSE

**External Examiner**

# DECLARATION

This is to certify that the work reported in this project entitled **“Car Showroom Dealership”** is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology**,** Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred to in the text. The report is based on the work done entirely by me and not copied from any other source.

**NAIKA NAVANEETH**

**(22261A0535)**

**SHIKA UPENDER**

**(22261A0543)**

# ACKNOWLEDGEMENT

We would like to express my sincere thanks to **Dr. G. Chandra Mohan Reddy, Principal MGIT,** for providing the working facilities in the college.

We wish to express our sincere thanks and gratitude to **Dr. C R K Reddy, Professor and HOD,** Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

We are extremely thankful to **Dr. C R K Reddy**, **Professor, Dr. A. Ratnaraju, Assistant Professor,** Department of CSE, MGIT, project stage-2 coordinators for their encouragement and support throughout the project.

We are extremely thankful and indebted to my internal guide **Ms. M. Navya,** Assistant Professor, Department of CSE, for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped me directly or indirectly for completing this project.

**NAIKA NAVANEETH**

**(22261A0535)**

**SHIKA UPENDAR**

**(22261A0543)**

# TABLE OF CONTENTS

[CERTIFICATE i](#_Toc169301940)

[DECLARATION ii](#_Toc169301941)

[ACKNOWLEDGEMENT iii](#_Toc169301942)

[TABLE OF CONTENTS iv](#_Toc169301943)

[LIST OF TABLES vi](#_Toc169301944)

[LIST OF FIGURES vii](#_Toc169301945)

[ABSTRACT viii](#_Toc169301946)

[1. INTRODUCTION 1](#_Toc169301947)

[1.1 Problem Statement 2](#_Toc169301948)

[1.2 Existing Systems Manual Methods: 2](#_Toc169301949)

[1.3 Proposed System 4](#_Toc169301950)

[1.4 Requirement Specification 5](#_Toc169301951)

[2. LITERATURE SURVEY 6](#_Toc169301952)

[3. DESIGN AND METHODOLOGY 9](#_Toc169301953)

[3.1 Class Diagram 9](#_Toc169301954)

[3.2 Block Diagram 10](#_Toc169301955)

[3.3 Methodology 11](#_Toc169301956)

[3.3.1 Define Objective: 11](#_Toc169301957)

[3.3.2 Data Collection: 11](#_Toc169301958)

[3.3.3 Data Preprocessing: 11](#_Toc169301959)

[3.4 UML Diagrams 12](#_Toc169301960)

[3.4.1 Use Case Diagram 12](#_Toc169301961)

[3.4.2 Activity Diagram 13](#_Toc169301962)

[3.4.3 Sequence Diagram 14](#_Toc169301963)

[4. IMPLEMENTATION AND RESULTS 15](#_Toc169301964)

[4.1 Implementation 15](#_Toc169301965)

[4.2 Results 16](#_Toc169301966)

[4.2.1 Opening Interface 17](#_Toc169301967)

[4.2.2 City Selection 17](#_Toc169301968)

[4.2.3 Search and Showroom Login 18](#_Toc169301969)

[4.2.4 Showroom Login and Success 18](#_Toc169301970)

[4.2.5 Add New Car 19](#_Toc169301971)

[4.2.6 Cars Information 19](#_Toc169301972)

[4.2.7 Developers Login 20](#_Toc169301973)

[4.2.8 Showroom Access Information 20](#_Toc169301974)

[4.2.9 New Showroom 21](#_Toc169301975)

[4.2.10 Search Cars 21](#_Toc169301976)

[4.2.11 Final Output 22](#_Toc169301977)

[4.3 Testing 23](#_Toc169301978)

[4.3.1 User Interface Evaluation: 23](#_Toc169301979)

[4.3.2 Functional Testing: 23](#_Toc169301980)

[4.3.3 Performance Testing: 23](#_Toc169301981)

[4.3.4 Accessibility Testing: 23](#_Toc169301982)

[5. CONCLUSION 24](#_Toc169301983)

[5.1 Conclusion 24](#_Toc169301984)

[5.2 Future Scope 24](#_Toc169301985)

[BIBLIOGRAPHY 25](#_Toc169301986)

[APPENDIX 26](#_Toc169301987)

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **NAME OF THE TABLE** | **PAGENO** |
| 2.1 | Literature survey | 6 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| FIGURE NO | NAME OF THE FIGURE | PAGE NO |
| 3.1 | Class Diagram | 9 |
| 3.2 | Block Diagram | 10 |
| 3.3.1 | Data Preprocessing | 11 |
| 3.4.1 | Use Case Diagram | 12 |
| 3.4.2 | Activity Diagram | 13 |
| 3.4.3 | Sequence Diagram | 14 |
| 4.2.1 | Operating Interface | 17 |
| 4.2.2  4.2.3  4.2.4  4.2.5  4.2.6  4.2.7  4.2.8  4.2.9  4.2.10  4.2.11 | City Selection  Search and Showroom Login  Showroom Login and Success  Add New Car  Cars Information  Developers Login  Showroom Access Information  New Showroom  Search Cars  Final Output | 17  18  18  19  19  20  20  21  21  22 |

# ABSTRACT

The Multi-City Car Showroom Application is a sophisticated Python GUI-based solution designed to streamline the car browsing experience across various cities. This application integrates seamlessly with multiple showrooms, offering users an extensive collection of vehicles to explore and compare. The key feature of this application lies in its powerful filtering mechanism, providing users with a tailored and efficient search experience. Multi-City Car Showroom Application with Comprehensive Filtering Mechanism, contrasting its features and advantages against traditional online car platforms. Developed using Python GUI, this application integrates multiple city showrooms, offering users an extensive selection of vehicles with advanced filters such as color, model, fuel type, mileage, price, brand, seating capacity, and more. The Multi-City Car Showroom Application distinguishes itself from traditional online car platforms through a myriad of key features that redefine the car shopping experience. By seamlessly integrating physical showrooms from various cities, this application establishes a direct link between users and local dealerships, fostering a personalized connection that goes beyond the virtual realm. Real-time updates on local inventories ensure users access the latest information on available cars and prices, a feature absent or less prominent in conventional online car platforms that rely on periodic updates. The user-friendly Python GUI enhances engagement through an intuitive interface, while the emphasis on physical presence and potential test drive opportunities sets the Multi-City Car Showroom Application apart, providing users with a holistic and immersive car shopping journey.

.

# INTRODUCTION

**[1]** The Multi-City Car Showroom Application is a cutting-edge platform that aims to revolutionize the car buying experience for both customers and showroom owners. Designed with a user-friendly search interface, it allows potential buyers to easily browse through an extensive array of car models, detailed specifications, and competitive prices available in multiple cities. Users can apply various filters such as car brand, model, year, price range, and location to quickly find the perfect vehicle that suits their preferences and needs. This intuitive and efficient browsing experience significantly reduces the time and effort involved in searching for a car, making the buying process smoother and more enjoyable for customers.

**[2]** For showroom owners, the application offers robust database access and inventory management tools that streamline their operations. Owners can update their listings in real-time, ensuring that potential buyers have access to the most current information about available cars. The platform allows showroom owners to upload high-quality images, detailed descriptions, and comprehensive specifications for each vehicle, effectively showcasing their offerings and attracting more prospective buyers. By providing these advanced tools, the application enables showroom owners to manage their inventory more efficiently, optimize their sales processes, and respond quickly to market demands and customer inquiries.

**[3]** Moreover, the Multi-City Car Showroom Application enhances the overall car buying process by facilitating better communication and offering additional services. The built-in secure communication system allows buyers to directly interact with showroom representatives, making it easy to inquire about specific models, request test drives, and negotiate prices. This direct line of communication ensures that customers receive timely and accurate responses, improving their buying experience. Additionally, the application provides access to financing options, insurance quotes, and after-sales support, offering a comprehensive solution that addresses all aspects of car purchasing.

## 1.1 Problem Statement

The challenges facing the Multi-City Integrated Showroom Application are multifaceted. Visual accessibility issues hinder user engagement, while inadequate inventory management leads to inefficiencies in stock monitoring and allocation. Limited database access restricts seamless integration and data sharing across multiple locations, exacerbating operational complexities. Moreover, the lack of robust customer interaction features diminishes user satisfaction and loyalty, impacting the application's competitiveness. Additionally, the threat of damaged vehicles in the online environment poses significant risks to both customers and the business, necessitating stringent quality control measures. Addressing these challenges requires a holistic approach encompassing user-centric design, enhanced inventory management systems, improved database accessibility, proactive customer engagement strategies, and rigorous quality assurance protocols to ensure the application's success in the dynamic automotive market

## 1.2 Existing Systems Manual Methods:

**Paper-based Inventory Management**: Showroom owners might maintain car inventory records using physical logs, spreadsheets, or paper forms. This can include details such as car specifications, pricing, and availability. These records are often stored in filing cabinets or binders.

**Manual Listings and Advertising**: Showrooms may advertise their cars through traditional media such as newspapers, magazines, and flyers. This involves manually creating advertisements, distributing brochures, and posting listings on bulletin boards or other physical locations within the showroom.

**In-Person Customer Interaction**: Sales representatives often interact with customers face-to-face, providing them with brochures, conducting test drives, and discussing car details verbally. All customer inquiries and follow-ups are typically handled manually through phone calls, emails, or in-person meetings.

**Car Showroom Management Software:**

The existing system of the car showroom application focuses on providing users with a seamless experience for exploring and purchasing new vehicles exclusively. It ensures an intuitive and visually appealing user interface, facilitating easy navigation and efficient browsing of the latest models available in the showroom. A meticulously managed inventory of new vehicles offers detailed information on specifications, pricing, and dealership details for each listing, empowering buyers with comprehensive data to make informed decisions. Quality assurance measures, including rigorous inspections and certifications, uphold the reliability of the listings. Overall, the car showroom application's system is designed to deliver a convenient and reliable platform for users seeking to explore and purchase new vehicles, enhancing the showroom experience.

**Limitations of Existing Systems:**

* **Lack of Database for the Backup [3]:**

The absence of a comprehensive database backup system highlights a vulnerability in current online trends.

This problem concerns data integrity and security necessitating the implementation of robust backup protocols to safeguard against potential data loss or corruption.

* **Features with limited space [2]:**

Limited spacing in a car showroom application poses a disadvantage as it restricts the presentation of comprehensive car details and diminishes the user experience by potentially overcrowding the interface.

Additionally, it may hinder the effective organization and navigation of information, leading to difficulty in accessing critical features or options.

* **Lack of Inventory Management [1]:**

The absence of inventory management in a car showroom application leads to inefficient tracking of available vehicles, potentially resulting in errors in stock management and

delays in updating vehicle availability for customers.

## 

## 1.3 Proposed System

The proposed system for a multi-city integrated car showroom leverages Python's Tkinter for the user interface and MySQL for the database. Initialization ensures readiness with database connections and server configurations. The user interface begins with a Tkinter welcome page and city selection dropdown. Advanced car searching allows users to input criteria such as type, price, seating, and city, processing this data to query the MySQL database for matches. Showrooms management maintains updated car information, integrating data from multiple cities. Secure user authentication is achieved through ID and password verification against stored credentials. Upon authentication, users access car or financial details, with matched car results displayed or a "no match found" message. The booking process includes displaying details, confirming user information, and updating records, followed by a final booking confirmation and redirection to the home page. The system ensures a robust, user-friendly platform with secure data handling and virtual car showroom accessibility.

**Key features of the proposed system include**:

**User-Friendly Car Search and Booking:** Enable users to easily find and book cars using detailed search criteria. Users can filter searches by car type, price, seating, mileage, brand, and model for efficient and precise results.

**Integrated Multi-City Showroom Management:** Manage car inventories and sales across multiple cities seamlessly within a single platform. The application allows users to select their city and access specific showroom data, ensuring accurate and localized car search results while enabling efficient multi-city inventory management for showroom operators.

**Comprehensive Financial and Car Information Access:** Provide detailed car and financial information. Authenticated users can view extensive car details and financial options, aiding informed purchasing decisions.

**Reliable Car Sales Management:** Accurately manage and present car sales details. The system maintains up-to-date car availability and details, facilitating effective showroom management.

**Secure User Authentication:** Ensure secure access to the application. Users log in with an ID and password, protecting sensitive data and ensuring only authorized access.

**Efficient Data Processing:** Process user input and search results quickly. The system swiftly handles queries, providing fast and accurate results for a smooth user experience.

## 1.4 Requirement Specification

## 

**Software Requirements:**

* **Operating System:** The system should be compatible with commonly used operating systems such as Windows, macOS, and Linux.
* **Python**: The main programming language for developing the application.
* **Tkinter**: The standard GUI library for Python. TKinter is included with most Python installations, but you can install it using pip if needed.
* **MySQL**: The database management system for storing car inventory and other related data. You need MySQL Server installed on your machine. MySQL Workbench or any other MySQL client for database management (optional).
* **MySQL Connector**: The connector for connecting Python to the MySQL database. Install using pip install MySQL-connector-python.

**Hardware Requirements:**

* **Processor:** Any modern multi-core processor (e.g., Intel Core i3, AMD Ryzen 3) should be sufficient for running the application smoothly.
* **Memory (RAM):** At least 2 GB of RAM is recommended for running the application without performance issues.
* **Storage:** The storage requirements for the project are minimal as it primarily consists of source code files, libraries, and documentation. A few hundred megabytes of disk space should be more than adequate.
* **Display:** A standard monitor with a resolution of 1024x768 pixels or higher is recommended for optimal display of the GUI.
* **Input Devices:** A keyboard and mouse (or trackpad) are required for interacting with the application.

# LITERATURE SURVEY

The Project car showroom applications has led to the publication of three significant papers in three different ways.

1. **“A Virtual Car Showroom”**

**Adrian Nowak et al.**, [1] explores the innovative use of virtual reality (VR) and augmented reality (AR) in the automotive industry. The authors describe the development and implementation of a virtual car showroom using the Unity game engine, head-mounted displays (HMDs), and a gesture tracking system. This system allows users to interact with high-fidelity 3D car models in a fully immersive virtual environment, enhancing the processes of prototyping, marketing, and customer engagement. The article highlights the advantages of VR/AR technologies in providing cost-effective, time-efficient, and interactive solutions for car presentation and evaluation.

1. **“Design of Virtual Automotive Showroom with Augmented Reality Technology Using the Smartphone”**

**F. Fahmi et al., [2]** explore the development of a smartphone application utilizing augmented reality (AR) to enhance the automotive showroom experience. This application, developed with Vuforia SDK and Unity 3D for the Android platform, allows users to interactively view and customize 3D car models in real-time. By pointing a smartphone camera at a special marker, users can visualize and change car features, such as colors, providing a dynamic and immersive way to explore different car configurations without the need for physical inventory. This innovation addresses the limitations of traditional showrooms, such as space constraints and the inability to display all available car models and colors. The AR application aims to improve customer engagement and provide a unique promotional tool for car vendors.

1. **Trends In an Online Automobile Market [3]**

**Raheem Ajetol Azeez et al., [3]** presents a comprehensive proposal for an online automobile market system designed to enhance the user experience and streamline the process of buying and selling vehicles. It addresses key issues in existing systems, such as the lack of a structured catalog and limited customization options for automobiles. The proposed system utilizes modern web technologies, including HTML, CSS, JavaScript, AJAX, PHP, and MySQL, to create a responsive and interactive user interface. Users can browse detailed information about automobiles, customize their preferences, and apply for products seamlessly. The system also incorporates efficient database management, error checking, and advanced search functionalities to ensure a smooth and user-friendly experience to traditional on-premises solutions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.**  **No** | **Title** | **Author** | **Proposed System / Algorithm** | **Merits** | **Demerits** |
| **1** | **A Virtual Car Showroom[ 1].** | Adrian Nowak, Jakub Flotyński | * Gesture Tracking System * 3D Car Models * Virtual Showroom Environment * Head-Mounted Display (HMD) * Unity Game Engine * VR technology | * Display of cars In 3D view * Interior Design Of the Car was displayed * Easy and less time for Development or Modification * Enables User configuration in a way that may be   impossible in showroom. | * Web Page application is not available. * new models will be added or customized depending on showroom every time.. * Symantic Representation of 3D Models. |
| **2** | **Design of Virtual Automotive Showroom with Augmented Reality Technology Using The Smartphone [2].** | F Fahmi M Alwy | * Real-time Processing * Interactive Features * Object Display or Rendering * Marker Detection * AR Visualization * User Interaction Handling   . | * AR projection-virtual objects in real time display * Augmented reality display of cars in different features in real-time such as size and colors | * presentation as much information on the cars * features with limited space. * Application Is made for Only for Android smart phones. * It cannot be Executed with in dim Light. * Data Input Requires Many Devices. |
| **3** | **Trends in an Online Automobile Market [ 3].** | Raheem Ajetola Azeez,  Mutiat Ogunrinde | * Automobile Customization * Modern Web Technologies | * Very Specified Structured Catalog. * Good Automobile Customization. * provides Individual Convenience for the User | * security Concerns   -related to data protection.   * No Database like Cloud for Backup of Data |

**Table 2.1** Literature Survey

# 3 DESIGN AND METHODOLOGY

## 3.1 Class Diagram

The design and methodology of the Car Showroom Dealer ship method consists of UML diagrams.

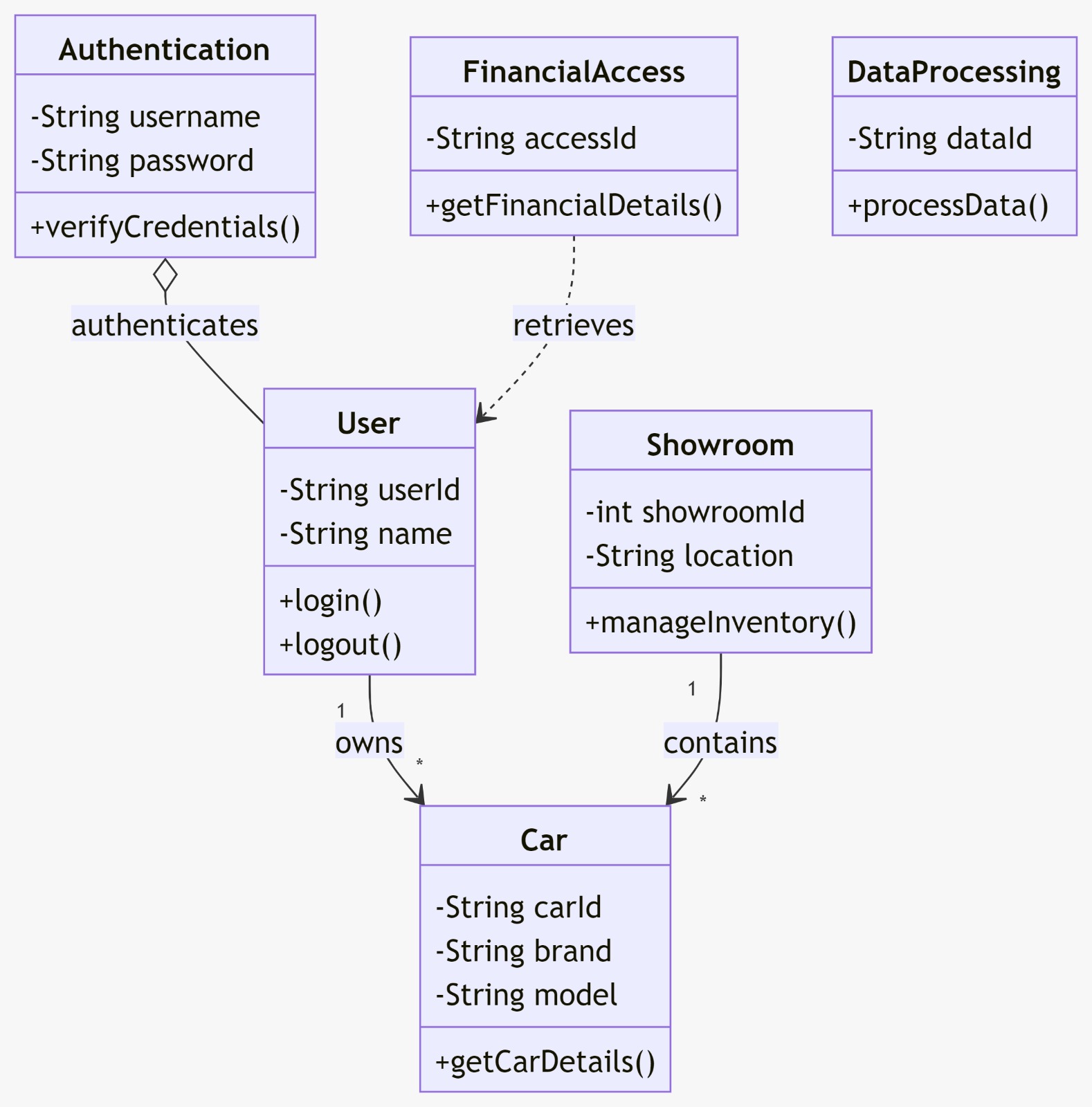


Figure 3.1 Class Diagram

In the above Figure 3.1, the class diagram is depicted about the process of authentication for the user and tells how access is given to the user about the different car model.

## 3.2 Block Diagram

The block diagram depicts the illustration of a system whose components are represented by blocks.

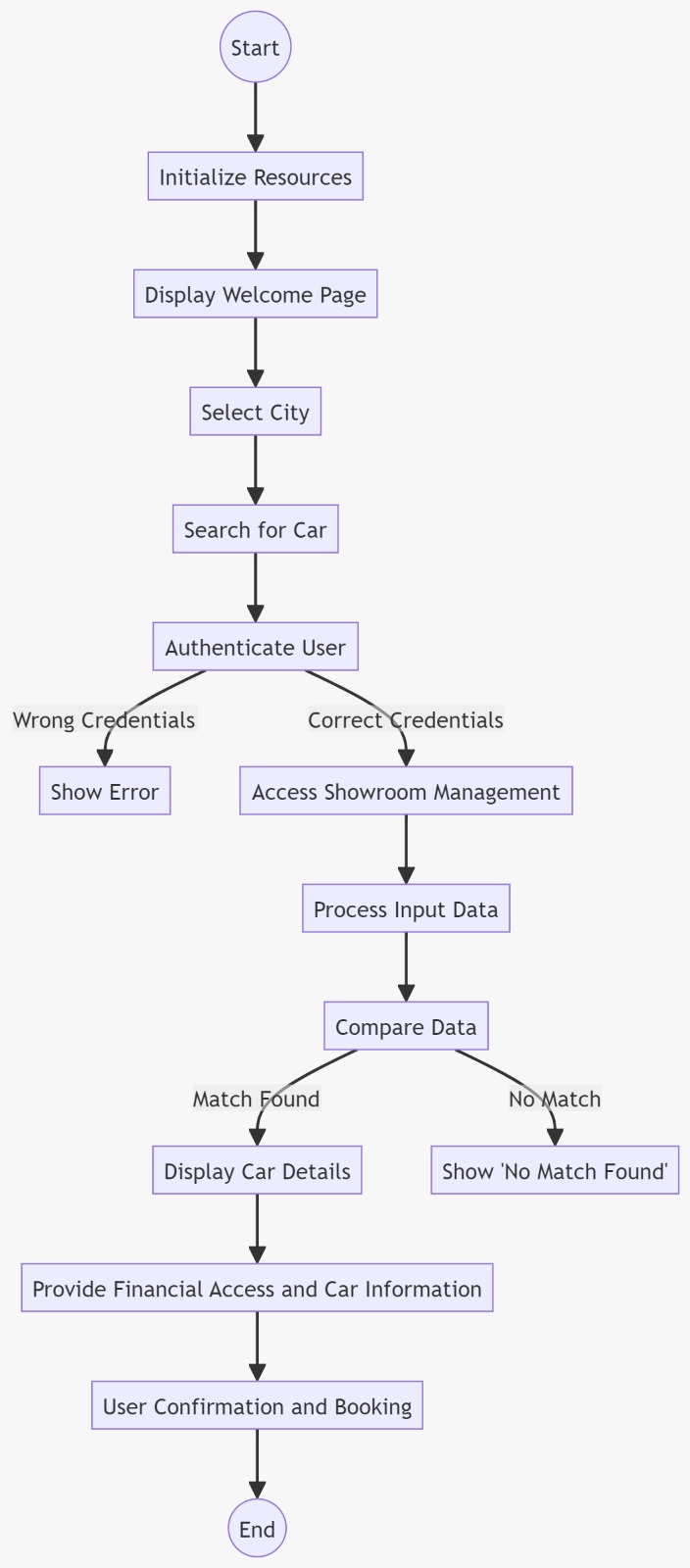


Figure 3.2Block Diagram for Tournament Management System

## 3.3 Methodology

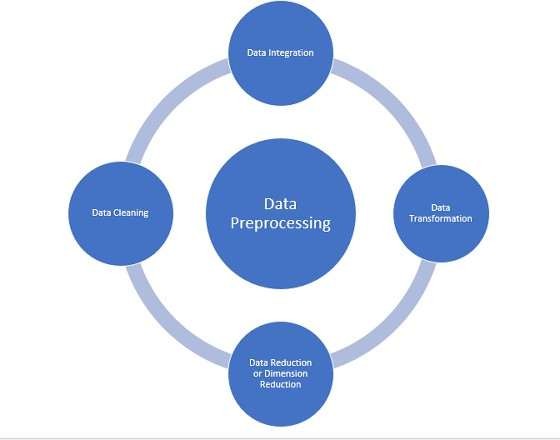
### 3.3.1 Define Objective:

The car showroom management system aims for efficient inventory management, CRM, sales and performance tracking, financial management, and service scheduling. This involves tracking car models and stock levels, maintaining customer records and interactions, analyzing sales data, monitoring financials, and managing vehicle maintenance.

### 3.3.2 Data Collection:

Data collection includes gathering inventory data from databases and suppliers, sales data from POS systems, customer data from CRM systems and surveys, financial data from accounting systems, and service data from scheduling systems. This ensures comprehensive data coverage across all showroom operations.

### 3.3.3 Data Preprocessing:

Preprocessing involves cleaning data to remove errors and standardize formats, transforming it into analyzable formats, integrating data from multiple sources, validating data for quality and consistency, and securely storing it. This structured approach facilitates easy access and analysis, supporting informed decision-making.

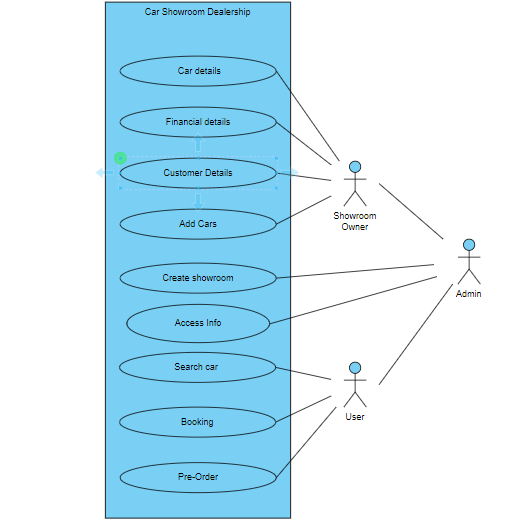
**Figure 3.3.1**: Data Preprocessing

## 3.4 UML Diagrams

UML Diagrams are based on diagrammatic representations of software components. There are two main categories: structure diagrams and behavioral diagrams. Structure diagrams show the things in the modelled system. In a more technical terms, they show different objects in a system. Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

### 3.4.1 Use Case Diagram

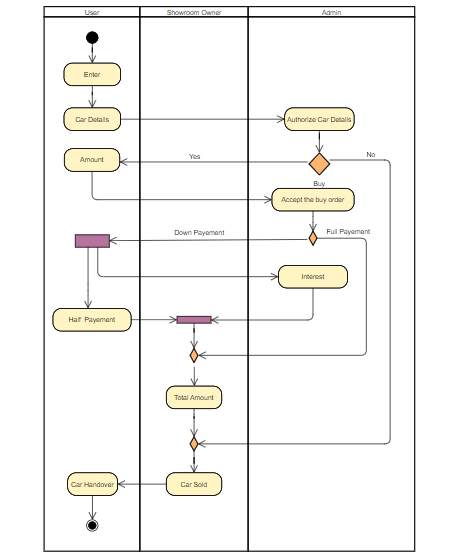
A use case diagram goal is to provide a graphical picture of a system's functionality in terms of actors, goals (represented as use cases), and any dependencies between those cases. A use case diagram's main aim is to indicate which actor performs which system functions.



**Figure 3.4.1** Use Case Diagram

### 3.4.2 Activity Diagram

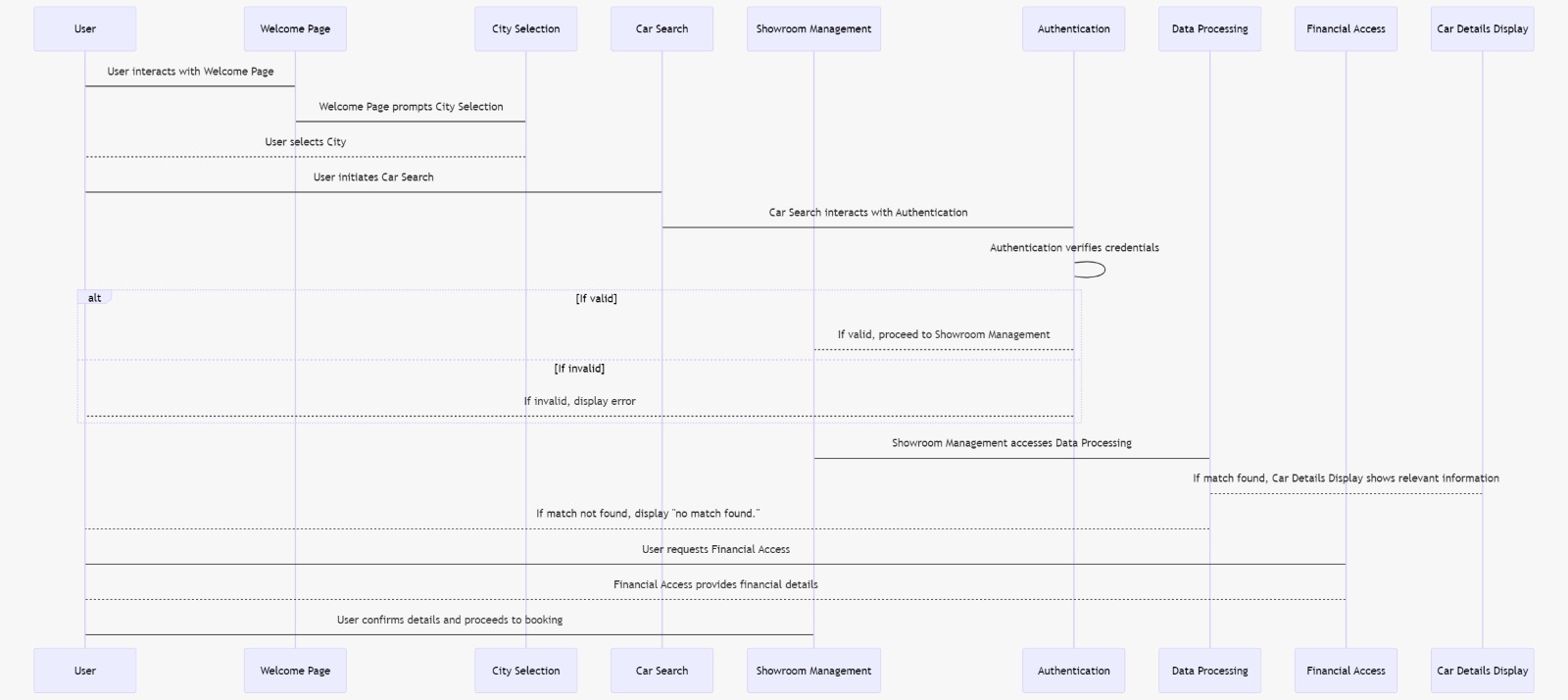
Another major behavioral diagram used in UML diagrams to illustrate the dynamic characteristics of the system is the activity diagram. An activity diagram is a more complex version of a flow chart that depicts the flow of information from one activity to the next.



**Figure 3.4.2** Activity Diagram

### 3.4.3 Sequence Diagram

A sequence diagram is a type of UML diagram that shows the interactions and message flow between objects or components in a system. It represents the chronological order of interactions and how objects collaborate over time.



**Figure 3.4.3** Sequence Diagram

# 4 IMPLEMENTATION AND RESULTS

## 4.1 Implementation

**User Interface (UI):**

The User Interface (UI) of the car showroom application is designed to provide a user-friendly and visually appealing experience for managing showrooms, cars, finances, and user information across multiple cities.

**Key Components:**

1. Dashboard: Offers an overview of showroom operations, financial performance, and inventory status across cities.

2. Forms and Input Fields: Allows adding, editing, and deleting cars, managing financial records, and updating user information.

3. Search and Filter Options: Facilitates efficient car searches based on criteria like make, model, and price range.

4. Navigation Menu: Ensures easy access to different sections such as showroom management, financials, and user profiles.

5. Responsive Design: Adapts seamlessly to various devices and screen sizes, enhancing accessibility.

**Controller:**

The Controller serves as the bridge between the UI (View) and the underlying data (Model), orchestrating the flow of information and actions within the application.

**Key Responsibilities:**

1.Handling User Input: Processes user requests from the UI to trigger appropriate actions within the application.

2.Implementing Business Logic: Executes business rules for operations like inventory management, sales processing, and financial tracking.

3. Routing and Flow Control: Directs data flow between the View and Model components to ensure smooth operation.

4. Error Handling: Manages exceptions and errors, providing feedback to users and logging issues for troubleshooting.

**View:**

The View represents the UI layer that users interact with directly, presenting information and collecting user inputs.

**Key Responsibilities:**

1. Displaying Information: Presents data such as car details, financial records, and user profiles in a clear and organized manner.

2. Collecting User Inputs: Captures user inputs through forms and input fields for actions like car management and user profile updates.

3. Implementing UI Logic: Executes client-side validation and formatting to ensure data integrity and user interaction.

4. Rendering Dynamic Content: Updates content dynamically based on interactions and data retrieved from the Controller and Model.

**Model:**

The Model encapsulates the data and business logic of the car showroom application, defining entities and managing data operations.

**Key Components:**

1. Data Entities: Defines structures for cars, showrooms, financial transactions, and user profiles, including attributes and relationships.

2. Business Logic: Implements rules and algorithms for operations such as inventory management, financial calculations, and user authentication.

3. Database Interaction: Handles data persistence and retrieval, ensuring data integrity and security.

4. Event Handling: Manages notifications and updates triggered by changes in data or user interactions.

**Database:**

The Database serves as the central repository for storing and managing persistent data used by the application.

**Key Responsibilities:**

1. Data Storage: Stores information such as car details, financial records, user profiles, and showroom configurations.

2. Data Retrieval: Provides efficient access to stored data based on queries and requests from the Model layer.

3. Data Security: Ensures data confidentiality, integrity, and availability through access controls, encryption, and backups.

4. Scalability and Performance: Optimizes database performance to handle large volumes of data and concurrent user requests, supporting application scalability.

## 4.2 Results

The car showroom application designed for multiple cities facilitates comprehensive management across various locations. It enables the creation and administration of showrooms in different cities, each equipped with tools for adding, updating, and removing cars from their respective inventories. Financial management capabilities track sales, expenses, and revenue generation per showroom, ensuring financial transparency and performance assessment. User profiles are securely managed, encompassing customer details and preferences, alongside administrative controls for staff. The application features robust search functionalities, allowing users to find cars based on criteria like make, model, and price range across all showrooms. Seamless integration ensures synchronized data across multiple locations, supporting scalability and accessibility for enhanced operational efficiency and customer service.

### A car on the road Description automatically generated4.2.1 Opening Interface

Figure 4.1 Opening Interface

Figure 4.2.1 explains Opening Interface Consists Opening Page and select city

option for selecting various cities.

### A screenshot of a computer Description automatically generated4.2.2 City Selection

Figure 4.2 City Selection

Figure 4.2.2 explains City Selection displays various cities when we click on

select city in Figure 4.2.1

### A screenshot of a computer Description automatically generated4.2.3 Search and Showroom Login

Figure 4.3 Search and Showroom Login

Figure 4.2.3 explains Search and Showroom Login Consists of Explore Cars

Option And Showroom Login Option to login into different Showrooms.

### A screenshot of a computer Description automatically generated4.2.4 Showroom Login and Success

Figure 4.4 Showroom Login and Success

Figure 4.2.4 image shows that message box when the credential of showroom matches with the information in database then allows the user to access the Database.

### 4.2.5 Add New Car

A screenshot of a computer

Description automatically generated

Figure 4.5 Add New Car

Figure 4.2.5 Consists Of Various Data Entry widgets to add a new Car to the Database.

## 

### 4.2.6 Cars Information

Figure 4.6 Cars Information

Figure 4.2.6 explains about the Data present about the cars in the different showrooms.

### A screenshot of a computer Description automatically generated4.2.7 Developers Login

Figure 4.7 Developers Login

Figure 4.2.7 is the interface to login into the developer account for the showroom

creation and access information.

### A screenshot of a computer Description automatically generated4.2.8 Showroom Access Information

Figure 4.8 Showroom Access Information

Figure 4.2.8 is about Showroom Access Information which Consists of data

related to the Ids and Passwords of the showrooms.

### A screenshot of a computer Description automatically generated4.2.9 New Showroom

Figure 4.9 New Showroom

Figure 4.2.9 New Showroom for Adding or Creating new Showroom in the Database.

### 4.2.10 Search Cars

A screenshot of a computer

Description automatically generated

Figure 4.10 Search Cars

Figure 4.2.10 Contains Various Entry Widgets which takes the input from the user and whenever User clicks search it gives the output by checking the information in the Database.

### A screenshot of a computer Description automatically generated4.2.11 Final Output

Figure 4.11 Final Output

Figure 4.2.11 shows the Display of the Information which is matched with the

Database of Showroom.

## 4.3 Testing

### 4.3.1 User Interface Evaluation:

User Interface (UI) Evaluation ensures the system is user-friendly and visually appealing through usability testing, consistency checks, and intuitive navigation assessments. It also involves testing responsiveness across various devices and screen sizes. Accessibility standards are verified to ensure compatibility with screen readers and keyboard navigation.

### 4.3.2 Functional Testing:

Functional Testing ensures the system performs correctly by conducting unit, integration, system, and regression tests. This verifies that all features, including inventory management, sales processing, CRM, financial tracking, and service scheduling, function as intended. Updates and bug fixes are also tested to ensure they do not negatively impact existing functionalities.

### 4.3.3 Performance Testing:

It assesses system behavior under different conditions, including load testing for peak usage, stress testing for extreme conditions, and scalability testing for handling variable loads. Response time testing is conducted to ensure the system meets performance standards by measuring the time taken to respond to user inputs and complete transactions.

### 4.3.4 Accessibility Testing:

Accessibility Testing ensures the system is usable by people with disabilities by verifying screen reader compatibility, keyboard navigation, and sufficient color contrast and text size.

# 5 CONCLUSION

## 5.1 Conclusion

In conclusion, developing a car showroom application offers significant benefits for enhancing customer experience and operational efficiency. Despite challenges like limited spacing and inventory management gaps, prioritizing user-centric design and data security can ensure success. By integrating advanced features and addressing these challenges, car dealerships can create a robust platform that delivers value to both customers and staff.

## 5.2 Future Scope

The future of car showroom applications is bright, offering opportunities for innovation and growth. Emerging technologies like AR and VR can revolutionize the car buying experience by providing immersive virtual showroom experiences. Advanced data analytics and machine learning enable personalized user experiences, while IoT integration allows real-time monitoring for proactive servicing. Mobile apps, social media integration, subscription models, and sustainability initiatives are also promising areas for development. By embracing these advancements, car showroom applications can continue to deliver exceptional experiences tailored to evolving customer needs.

# BIBLIOGRAPHY

[1]. Adrian Nowak Jakub Flotyński "A VIRTUAL CAR SHOWROOM" Original publication: Conference Paper in Web3D '18 Proceedings of the International ACM Conference Pg. 231-232,2018.

[2]. F Fahmi and M Alwy "DESIGN OF VIRTUAL AUTOMATIVE SHOWROOM WITH AUGUMENTED REALITY TECHNOLOGY USING THE SMARTPHONE " IOP Conf. Ser.: Mater. Sci. Eng, volume 1003, Pg. 145-152, 2020.

[3]. Mutiat A. OGUNRINDE, Raheem A. Azeez, Maryam O. JIMOH "TRENDS IN AN ONLINE AUTOMOBILE MARKET" International Journal of Innovative and Applied Research, Volume2, Issue (8), Pg. 61- 66, 2014.

# APPENDIX

from tkinter import \*

from PIL import Image,ImageTk

import ttkbootstrap as tb

import mysql.connector

from tkinter import ttk

from tkinter import messagebox

root=tb.Window(themename="superhero")

root.title("MY CAR SHOWROOM")

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

root.geometry(f"{screen\_width}x{screen\_height}")

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

def city():

global button1,button2,button3,button4,button5

f1=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f1.place(x=0,y=0)

f1.pack\_propagate(0)

l1=tb.Label(root,text="EXPLORE VARIOUS SHOWROOMS IN DIFFERENT CITIES",bootstyle="light",font=("Helvetica",25))

l1.place(x=250,y=50)

img\_path=r"C:\Users\chand\Downloads\RTP\Kolkata.jpg"

image1=Image.open(img\_path)

image1=ImageTk.PhotoImage(image1)

button1=tb.Button(f1,text="KOLKATA",compound="top",image=image1,bootstyle="info outline",command=Kolkata)

button1.image=image1

button1.place(x=150,y=150)

img\_path=r"C:\Users\chand\Downloads\RTP\Hyderabad.jpg"

image2=Image.open(img\_path)

image2=ImageTk.PhotoImage(image2)

button2=tb.Button(f1,text="HYDERABAD",compound="top",image=image2,bootstyle="info outline",command=Hyderabad)

button2.image=image2

button2.place(x=550,y=150)

img\_path=r"C:\Users\chand\Downloads\RTP\Delhi.jpg"

image3=Image.open(img\_path)

image3=ImageTk.PhotoImage(image3)

button3=tb.Button(f1,text="DELHI",compound="top",image=image3,bootstyle="info outline",command=Delhi)

button3.image=image3

button3.place(x=900,y=150)

img\_path=r"C:\Users\chand\Downloads\RTP\Chennai.jpg"

image4=Image.open(img\_path)

image4=ImageTk.PhotoImage(image4)

button4=tb.Button(f1,text="CHENNAI",compound="top",image=image4,bootstyle="info outline",command=Chennai)

button4.image=image4

button4.place(x=290,y=420)

img\_path=r"C:\Users\chand\Downloads\RTP\Mumbai.jpg"

image5=Image.open(img\_path)

image5=ImageTk.PhotoImage(image5)

button5=tb.Button(f1,text="MUMBAI",compound="top",image=image5,bootstyle="info outline",command=Mumbai)

button5.image=image5

button5.place(x=750,y=420,anchor=NW)

#1#

def LOGIN():

global entry1,entry2

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b3.place(x= 900,y=100 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='750')

c.place(x=320,y=0)

rec1=c.create\_rectangle((0,180,700,400))

label1 = tb.Label(f2, text=" SHOWROOM ID")

label1.place(x=350,y=200,anchor=NW)

l1 = tb.Label(f2, text="SHOWROOM LOGIN")

l1.place(x=500,y=150,anchor=NW)

entry1 = tb.Entry(f2, width="50")

entry1.place(x=500,y=200,anchor=NW)

label2 = tb.Label(f2, text="PASSWORD")

label2.place(x=350,y=260,anchor=NW)

entry2 = tb.Entry(f2,width="50")

entry2.place(x=500,y=255,anchor=NW)

b3=tb.Button(f2,text="LOGIN",bootstyle="sucess outline",command=LOGIN\_VERIFY, width="30")

b3.place(x= 500,y=330 , anchor=NW)

#1#+

def LOGIN\_VERIFY():

Id=entry1.get()

Password=entry2.get()

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

sqlqry="SELECT ID,PASSWORD FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchall()

if(Id=='' or Password==''):

messagebox.showinfo("warning","please enter all the fields")

if result:

if(result[0][0]==Id and result[0][1]== Password):

messagebox.showinfo("Information","Login sucessful")

SHOWROOM\_LOGIN()

else:

messagebox.showinfo("Warning", "Incorrect Credentials")

LOGIN()

#1#1#

def SHOWROOM\_LOGIN():

Id=entry1.get()

Password=entry2.get()

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

sqlqry="SELECT SHOWROOM FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchone()

str1="CARS"

str2="FINANCIALINFO"

str3="USERINFO"

TABLE1 = [str(item) + str1 for item in result]

TABLE2 = [str(item) + str2 for item in result]

TABLE3 = [str(item) + str3 for item in result]

for table\_name1 in TABLE1:

sqlqry1=f"CREATE TABLE IF NOT EXISTS {table\_name1}(MODEL CHAR(10),CAR VARCHAR(20),MILEAGE INT,SEATER VARCHAR(8),FUELTYPE CHAR(7),MINPRICE INT,MAXPRICE INT,QUANTITY INT,COLOR CHAR(10),IMAGE VARCHAR(100))"

cur.execute(sqlqry1)

for table\_name2 in TABLE2:

sqlqry2=f"CREATE TABLE IF NOT EXISTS {table\_name2}(MODEL CHAR(10),CAR VARCHAR(20),AMOUNT INT,COLOR CHAR(10))"

cur.execute(sqlqry2)

for table\_name3 in TABLE3:

sqlqry3=f"CREATE TABLE IF NOT EXISTS {table\_name3}(MODEL CHAR(10),CAR VARCHAR(20),COLOR CHAR(10),NAME CHAR(20),PHONENO INT,ADDRESS VARCHAR(40),GMAIL VARCHAR(20))"

cur.execute(sqlqry3)

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b3=tb.Button(root,text="CARS INFO",bootstyle="primary outline",command=CARS, width="30")

b3.place(x= 200,y=100 , anchor=NW)

b4=tb.Button(root,text="FINANCIAL INFORMATION",bootstyle="primary outline",command=MONEY, width="50")

b4.place(x= 400,y=100 , anchor=NW)

b5=tb.Button(root,text="CUSTOMER INFO",bootstyle="primary outline",command=USER, width="30")

b5.place(x= 600,y=100 , anchor=NW)

b6=tb.Button(root,text="NEW CAR",bootstyle="primary outline",command=ADD\_CAR, width="30")

b6.place(x= 800,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=LOGIN, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

ADD\_CAR()

mydb.commit()

#1#1#1#

def CARS():

#f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

#f2.place(x=0,y=0)

#f2.pack\_propagate(0)

b3=tb.Button(root,text="CARS INFO",bootstyle="primary outline",command=CARS, width="30")

b3.place(x= 200,y=100 , anchor=NW)

b4=tb.Button(root,text="FINANCIAL INFORMATION",bootstyle="primary outline",command=MONEY, width="50")

b4.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="NEW CAR",bootstyle="primary outline",command=ADD\_CAR, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b5=tb.Button(root,text="CUSTOMER INFO",bootstyle="primary outline",command=USER, width="30")

b5.place(x= 800,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=LOGIN, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

def retrieve1\_data():

Id=entry1.get()

Password=entry2.get()

sqlqry="SELECT SHOWROOM FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchone()

str1="CARS"

TABLE11 = [str(item) + str1 for item in result]

for table\_cars1 in TABLE11:

cur.execute(f"SELECT \* FROM {table\_cars1}")

rows = cur.fetchall()

cur.close()

mydb.close()

return rows

def populate\_table1(table1, data):

for i, row in enumerate(data):

table1.insert('', 'end', values=row)

for col in table1["columns"]:

table1.column(col, anchor="center")

table1 = ttk.Treeview(root, columns=("Column 0", "Column 1", "Column 2","column 3","column 4","Column 5", "Column 6", "Column 7","column 8","column 9"))

table1.heading("#0", text="CARS\_INFO")

table1.heading("#1", text="MODEL")

table1.heading("#2", text="CAR\_NAME")

table1.heading("#3", text="MILEAGE")

table1.heading("#4", text="SEATER")

table1.heading("#5", text="FUELTYPE")

table1.heading("#6", text="MINPRICE")

table1.heading("#7", text="MAXPRICE")

table1.heading("#8", text="AVAILABLE\_CARS")

table1.heading("#9", text="COLOR")

table1.heading("#10", text="IMAGE")

data = retrieve1\_data()

populate\_table1(table1, data)

table1.pack(expand=True, fill='both')

b4=tb.Button(root,text="Back",bootstyle="primary outline",command=SHOWROOM\_LOGIN, width="30")

b4.place(x= 1100,y=900 , anchor=NW)

#1#1#2#

def MONEY():

f3=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f3.place(x=0,y=0)

f3.pack\_propagate(0)

b3=tb.Button(root,text="CARS INFO",bootstyle="primary outline",command=CARS, width="30")

b3.place(x= 200,y=100 , anchor=NW)

b4=tb.Button(root,text="FINANCIAL INFORMATION",bootstyle="primary outline",command=MONEY, width="50")

b4.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="NEW CAR",bootstyle="primary outline",command=ADD\_CAR, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b5=tb.Button(root,text="CUSTOMER INFO",bootstyle="primary outline",command=USER, width="30")

b5.place(x= 800,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=LOGIN, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

def retrieve2\_data():

Id=entry1.get()

Password=entry2.get()

sqlqry="SELECT SHOWROOM FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchone()

str2="FINANCIALINFO"

TABLE12 = [str(item) + str2 for item in result]

for table\_finance1 in TABLE12:

cur.execute(f"SELECT \* FROM {table\_finance1}")

rows = cur.fetchall()

cur.close()

mydb.close()

return rows

def populate\_table2(table2, data):

for i, row in enumerate(data):

table2.insert('', 'end', values=row)

for col in table2["columns"]:

table2.column(col, anchor="center")

table2 = ttk.Treeview(root, columns=("Column 0", "Column 1", "Column 2","column 3"))

table2.heading("#0", text="FINANCIAL\_INFO")

table2.heading("#1", text="MODEL")

table2.heading("#2", text="CAR\_NAME")

table2.heading("#3", text="AMOUNT")

table2.heading("#4", text="COLOR")

data = retrieve2\_data()

populate\_table2(table2, data)

table2.pack(expand=True, fill='both')

b4=tb.Button(root,text="Back",bootstyle="primary outline",command=SHOWROOM\_LOGIN, width="30")

b4.place(x= 1100,y=900 , anchor=NW)

#1#1#3#

def USER():

f3=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f3.place(x=0,y=0)

f3.pack\_propagate(0)

b3=tb.Button(root,text="CARS INFO",bootstyle="primary outline",command=CARS, width="30")

b3.place(x= 200,y=100 , anchor=NW)

b4=tb.Button(root,text="FINANCIAL INFORMATION",bootstyle="primary outline",command=MONEY, width="50")

b4.place(x= 400,y=100 , anchor=NW)

b5=tb.Button(root,text="CUSTOMER INFO",bootstyle="primary outline",command=USER, width="30")

b5.place(x= 800,y=100 , anchor=NW)

b6=tb.Button(root,text="NEW CAR",bootstyle="primary outline",command=ADD\_CAR, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=LOGIN, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

def retrieve3\_data():

Id=entry1.get()

Password=entry2.get()

sqlqry="SELECT SHOWROOM FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchone()

str3="USERINFO"

TABLE13 = [str(item) + str3 for item in result]

for table\_user1 in TABLE13:

cur.execute(f"SELECT \* FROM {table\_user1}")

rows = cur.fetchall()

cur.close()

mydb.close()

return rows

def populate\_table3(table3, data):

for i, row in enumerate(data):

table3.insert('', 'end', values=row)

for col in table3["columns"]:

table3.column(col, anchor="center")

table3 = ttk.Treeview(root, columns=("Column 0", "Column 1", "Column 2","column 3","Column 4", "Column 5","Column 6"))

table3.heading("#0", text="FINANCIAL\_INFO")

table3.heading("#1", text="MODEL")

table3.heading("#2", text="CAR\_NAME")

table3.heading("#3", text="COLOR")

table3.heading("#4", text="USER\_NAME")

table3.heading("#5", text="PHONE\_NUMBER")

table3.heading("#6", text="ADDRESS")

table3.heading("#7", text="GMAIL")

data = retrieve3\_data()

populate\_table3(table3, data)

table3.pack(expand=True, fill='both')

b4=tb.Button(root,text="Back",bootstyle="primary outline",command=SHOWROOM\_LOGIN, width="30")

b4.place(x= 1100,y=900 , anchor=NW)

#1#1#4#

def ADD\_CAR():

global selected\_option,selected\_option1,selected\_option2,selected\_option3,e111,e113,e114,e115,e117,e118

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b3=tb.Button(root,text="CARS INFO",bootstyle="primary outline",command=CARS, width="30")

b3.place(x= 200,y=100 , anchor=NW)

b4=tb.Button(root,text="FINANCIAL INFORMATION",bootstyle="primary outline",command=MONEY, width="50")

b4.place(x= 400,y=100 , anchor=NW)

b5=tb.Button(root,text="CUSTOMER INFO",bootstyle="primary outline",command=USER, width="30")

b5.place(x= 800,y=100 , anchor=NW)

b6=tb.Button(root,text="NEW CAR",bootstyle="primary outline",command=ADD\_CAR, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=LOGIN, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='900')

c.place(x=300,y=0)

rec1=c.create\_rectangle((10,180,800,650))

l6 = tb.Label(f2, text="NEW ENTRY")

l6.place(x=600,y=150,anchor=NW)

label1 = tb.Label(f2, text=" MODEL")

label1.place(x=350,y=200,anchor=NW)

options = ["BRAND","HYUNDAI", "HONDA", "KIA","MARUTHISUZUKI","AUDI","MG","TATA","MAHINDRA"]

selected\_option = tb.StringVar()

combobox = tb.Combobox(f2,textvariable=selected\_option, values=options,bootstyle="info")

combobox.place(x=435,y=190)

combobox.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l1 = tb.Label(f2, text="CAR NAME")

l1.place(x=350,y=260,anchor=NW)

e111=tb.Entry(f2,width="60")

e111.place(x=435,y=250,anchor=NW)

label2 = tb.Label(f2, text="SEATER")

label2.place(x=350,y=500,anchor=NW)

options3 = ["SEATER","4 SEATER", "5 SEATER", "6 SEATER","7 SEATER","8 SEATER"]

selected\_option3 = tb.StringVar()

combobox3 = tb.Combobox(f2,textvariable=selected\_option3, values=options3,bootstyle="info")

combobox3.place(x=435,y=500 )

combobox3.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l2 = tb.Label(f2, text=" FUEL TYPE")

l2.place(x=350,y=320,anchor=NW)

options1 = ["FUEL","PETROL", "DIESEL", "CNG"]

selected\_option1 = tb.StringVar()

combobox1 = tb.Combobox(f2,textvariable=selected\_option1, values=options1,bootstyle="info")

combobox1.place(x=435,y=310)

combobox1.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l6 = tb.Label(f2, text="COLOR")

l6.place(x=700,y=320,anchor=NW)

options2 = ["COLOR","WHITE", "GREY", "RED","SILVER","BLUE","BLACK","GREEN","VIOLET","YELLOW"]

selected\_option2 = tb.StringVar()

combobox2 = tb.Combobox(f2,textvariable=selected\_option2, values=options2,bootstyle="info")

combobox2.place(x=785,y=310)

combobox2.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l3 = tb.Label(f2, text="MILEAGE")

l3.place(x=350,y=380,anchor=NW)

e113=tb.Entry(f2,width="30")

e113.place(x=435,y=370,anchor=NW)

l7 = tb.Label(f2, text="QUANTITY")

l7.place(x=700,y=380,anchor=NW)

e117=tb.Entry(f2,width="30")

e117.place(x=785,y=370,anchor=NW)

l4 = tb.Label(f2, text="MIN PRICE")

l4.place(x=350,y=440,anchor=NW)

e114=tb.Entry(f2,width="30")

e114.place(x=435,y=430,anchor=NW)

l5 = tb.Label(f2, text="MAX PRICE")

l5.place(x=700,y=440,anchor=NW)

e115=tb.Entry(f2,width="30")

e115.place(x=785,y=430,anchor=NW)

l8=tb.Label(f2,text="IMAGE")

l8.place(x=350,y=560,anchor=NW)

e118=tb.Entry(f2,width="80")

e118.place(x=435,y=550,anchor=NW)

b8=tb.Button(f2,text="SUBMIT",bootstyle="sucess outline",command=NEW\_DATA, width="30")

b8.place(x=580 ,y=610 , anchor=NW)

#1#1#4#+

def NEW\_DATA():

Id=entry1.get()

Password=entry2.get()

model = selected\_option.get()

car\_name = e111.get()

seater = selected\_option3.get()

fuel\_type = selected\_option1.get()

color = selected\_option2.get()

mileage = e113.get()

quantity = e117.get()

min\_price = e114.get()

max\_price = e115.get()

image=e118.get()

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

sqlqry="SELECT SHOWROOM FROM DE WHERE ID=%s AND PASSWORD=%s"

cur.execute(sqlqry,(Id,Password))

result=cur.fetchone()

str1="CARS"

TABLE1 = [str(item) + str1 for item in result]

for table\_car1 in TABLE1:

sqlqry1=f"INSERT INTO {table\_car1}(MODEL ,CAR,MILEAGE,SEATER ,FUELTYPE,MINPRICE,MAXPRICE,QUANTITY,COLOR,IMAGE) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

value=(model,car\_name,mileage,seater,fuel\_type,min\_price,max\_price,quantity,color,image)

cur.execute(sqlqry1,value)

mydb.commit()

e111.delete(0,END)

e117.delete(0,END)

e113.delete(0,END)

e114.delete(0,END)

e115.delete(0,END)

e118.delete(0,END)

selected\_option.set('')

selected\_option1.set('')

selected\_option2.set('')

selected\_option3.set('')

l5 = tb.Label(root, text="DATA ENTRED SUCCESFULLY")

l5.place(x=600,y=680,anchor=NW)

#2#

def DEVELOPER():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='750')

c.place(x=320,y=0)

rec1=c.create\_rectangle((0,180,550,400))

label1 = tb.Label(f2, text=" ID")

label1.place(x=350,y=200,anchor=NW)

l1 = tb.Label(f2, text="DEVELOPER LOGIN")

l1.place(x=500,y=150,anchor=NW)

entry1 = tb.Entry(f2, width="50")

entry1.place(x=450,y=200,anchor=NW)

label2 = tb.Label(f2, text="PASSWORD")

label2.place(x=350,y=260,anchor=NW)

entry2 = tb.Entry(f2,width="50")

entry2.place(x=450,y=255,anchor=NW)

b3=tb.Button(f2,text="LOGIN",bootstyle="sucess outline",command=DEVELOPERS\_LOGIN, width="30")

b3.place(x= 450,y=330 , anchor=NW)

#2#1#

def DEVELOPERS\_LOGIN():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b4=tb.Button(root,text="NEW SHOWROOM",bootstyle="primary outline",command=ADD\_SHOWROOM, width="30")

b4.place(x= 200,y=100 , anchor=NW)

b5=tb.Button(root,text="FEEDBACKS",bootstyle="primary outline",command=USER\_FEEDBACK, width="30")

b5.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="SHOWROOM-ACCESS INFO",bootstyle="primary outline",command=ID\_PASSWORD, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=DEVELOPER, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

#2#1#1#

def ADD\_SHOWROOM():

global e11,e12,e13,e14,e15

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b4=tb.Button(root,text="NEW SHOWROOM",bootstyle="primary outline",command=ADD\_SHOWROOM, width="30")

b4.place(x= 200,y=100 , anchor=NW)

b5=tb.Button(root,text="FEEDBACKS",bootstyle="primary outline",command=USER\_FEEDBACK, width="30")

b5.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="SHOWROOM-ACCESS INFO",bootstyle="primary outline",command=ID\_PASSWORD, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=DEVELOPER, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='900')

c.place(x=320,y=0)

rec1=c.create\_rectangle((10,180,850,630))

l1=tb.Label(f2,text="SHOWROOM NAME")

l1.place(x=350,y=230,anchor=NW)

e11=tb.Entry(f2,width="60")

e11.place(x=500,y=225,anchor=NW)

l2=tb.Label(f2,text="CITY")

l2.place(x=350,y=290,anchor=NW)

e12=tb.Entry(f2,width="30")

e12.place(x=500,y=285,anchor=NW)

l3=tb.Label(f2,text="NEW ID")

l3.place(x=350,y=350,anchor=NW)

e13=tb.Entry(f2,width="30")

e13.place(x=500,y=345,anchor=NW)

l4=tb.Label(f2,text="NEW PASSWORD")

l4.place(x=350,y=410,anchor=NW)

e14=tb.Entry(f2,width="30")

e14.place(x=500,y=405,anchor=NW)

b7=tb.Button(root,text="NEW SHOWROOM",bootstyle="success outline",command=NEW\_SHOWROOM\_DATA, width="30")

b7.place(x= 550,y=530 , anchor=NW)

l5=tb.Label(f2,text="ADDRESS")

l5.place(x=350,y=470,anchor=NW)

e15=tb.Entry(f2,width="80")

e15.place(x=500,y=465,anchor=NW)

#ID PASSWORD TABLE

cur.execute("CREATE TABLE IF NOT EXISTS DE(ID VARCHAR(20),PASSWORD VARCHAR(20),SHOWROOM VARCHAR(30),CITY CHAR(10),ADDRESS VARCHAR(40))")

#2#1#1#+

def NEW\_SHOWROOM\_DATA():

a=e13.get()

b=e14.get()

c=e11.get()

d=e12.get()

e=e15.get()

sqlqry=("INSERT INTO DE (ID,PASSWORD,SHOWROOM,CITY,ADDRESS) VALUES(%s,%s,%s,%s,%s)")

value=(a,b,c,d,e)

cur.execute(sqlqry,value)

mydb.commit()

e11.delete(0,END)

e12.delete(0,END)

e13.delete(0,END)

e14.delete(0,END)

e15.delete(0,END)

l4=tb.Label(root,text="SHOWROOM DATABASE CREATED")

l4.place(x=550,y=650,anchor=NW)

#2#1#2#

def USER\_FEEDBACK():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b4=tb.Button(root,text="NEW SHOWROOM",bootstyle="primary outline",command=ADD\_SHOWROOM, width="30")

b4.place(x= 200,y=100 , anchor=NW)

b5=tb.Button(root,text="FEEDBACKS",bootstyle="primary outline",command=USER\_FEEDBACK, width="30")

b5.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="SHOWROOM-ACCESS INFO",bootstyle="primary outline",command=ID\_PASSWORD, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=DEVELOPER, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

#2#1#3#

def ID\_PASSWORD():

mydb=mysql.connector.connect(

host="localhost",

user="root",

password="",

database="DB1")

cur=mydb.cursor()

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b4=tb.Button(root,text="NEW SHOWROOM",bootstyle="primary outline",command=ADD\_SHOWROOM, width="30")

b4.place(x= 200,y=100 , anchor=NW)

b5=tb.Button(root,text="FEEDBACKS",bootstyle="primary outline",command=USER\_FEEDBACK, width="30")

b5.place(x= 400,y=100 , anchor=NW)

b6=tb.Button(root,text="SHOWROOM-ACCESS INFO",bootstyle="primary outline",command=ID\_PASSWORD, width="30")

b6.place(x= 600,y=100 , anchor=NW)

b7=tb.Button(root,text="LOGOUT",bootstyle="primary outline",command=DEVELOPER, width="30")

b7.place(x= 1000,y=100 , anchor=NW)

def retrieve\_data():

cur.execute("SELECT \* FROM DE")

rows = cur.fetchall()

cur.close()

mydb.close()

return rows

def populate\_table(table, data):

for i, row in enumerate(data):

table.insert('', 'end', values=row)

for col in table["columns"]:

table.column(col, anchor="center")

tb.Label(root,text=" ").pack()

tb.Label(root,text=" ").pack()

tb.Label(root,text=" ").pack()

tb.Label(root,text=" ").pack()

table = ttk.Treeview(root, columns=("Column 0", "Column 1", "Column 2","column 3","column 4"))

table.heading("#0", text="ID\_PASSWORD")

table.heading("#1", text="ID")

table.heading("#2", text="PASSWORD")

table.heading("#3", text="SHOWROOM")

table.heading("#4", text="CITY")

table.heading("#5", text="ADDRESS")

data = retrieve\_data()

populate\_table(table, data)

table.pack(expand=True, fill='both')

b4=tb.Button(root,text="Back",bootstyle="primary outline",command=DEVELOPERS\_LOGIN, width="30")

b4.place(x= 20,y=10 , anchor=NW)

#3#

def CONTACT():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=70 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=70 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='750')

c.pack()

rec1=c.create\_rectangle((10,180,520,400))

label1 = tb.Label(f2, text="PROBLEM")

label1.place(x=350,y=200,anchor=NW)

l1 = tb.Label(f2, text="CONTACT US")

l1.place(x=500,y=150,anchor=NW)

entry1 = tb.Entry(f2, width="50")

entry1.place(x=430,y=200,anchor=NW)

label2 = tb.Label(f2, text="EMAIL")

label2.place(x=350,y=260,anchor=NW)

entry2 = tb.Entry(f2, width="50")

entry2.place(x=430,y=255,anchor=NW)

label3 = tb.Label(f2, text="PHONE NO")

label3.place(x=350,y=320,anchor=NW)

entry3 = tb.Entry(f2, width="50")

entry3.place(x=430,y=315,anchor=NW)

b4=tb.Button(f2,text="SUBMIT",bootstyle="sucess outline",command=FEEDBACK, width="30")

b4.place(x= 480,y=365 , anchor=NW)

#3#1#

def FEEDBACK():

l1 = tb.Label(root, text="FEEDBACK SENT")

l1.place(x=520,y=415,anchor=NW)

#

def Hyderabad():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

image\_path = r"C:\Users\chand\Downloads\RTP\BackGround1.jpg"

image = Image.open(image\_path)

image = image.resize((860, 600))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(f2, image=photo)

image\_label.image = photo

image\_label.place(x=100,y=135)

b1=tb.Button(root,text="EXPLORE CARS",bootstyle="success outline",command=SEARCH\_CARS, width="30")

b1.place(x= 400,y=650 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

b4=tb.Button(root,text="DEVELOPER LOGIN",bootstyle="primary outline",command=DEVELOPER, width="30")

b4.place(x= 500,y=100 , anchor=NW)

b5=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b5.place(x= 700,y=100 , anchor=NW)

#

def Chennai():

variable=Chennai

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

image\_path = r"C:\Users\chand\Downloads\RTP\BackGround1.jpg"

image = Image.open(image\_path)

image = image.resize((860, 600))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(f2, image=photo)

image\_label.image = photo

image\_label.place(x=100,y=135)

b1=tb.Button(root,text="EXPLORE CARS",bootstyle="success outline",command=SEARCH\_CARS, width="30")

b1.place(x= 400,y=650 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

b4=tb.Button(root,text="DEVELOPER LOGIN",bootstyle="primary outline",command=DEVELOPER, width="30")

b4.place(x= 500,y=100 , anchor=NW)

b5=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b5.place(x= 700,y=100 , anchor=NW)

#

def Delhi():

image\_path = r"C:\Users\chand\Downloads\RTP\BackGround1.jpg"

image = Image.open(image\_path)

image = image.resize((860, 600))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(f2, image=photo)

image\_label.image = photo

image\_label.place(x=100,y=135)

b1=tb.Button(root,text="EXPLORE CARS",bootstyle="success outline",command=SEARCH\_CARS, width="30")

b1.place(x= 400,y=650 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

b4=tb.Button(root,text="DEVELOPER LOGIN",bootstyle="primary outline",command=DEVELOPER, width="30")

b4.place(x= 500,y=100 , anchor=NW)

b5=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b5.place(x= 700,y=100 , anchor=NW)

#

def Kolkata():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

image\_path = r"C:\Users\chand\Downloads\RTP\BackGround1.jpg"

image = Image.open(image\_path)

image = image.resize((860, 600))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(f2, image=photo)

image\_label.image = photo

image\_label.place(x=100,y=135)

b1=tb.Button(root,text="EXPLORE CARS",bootstyle="success outline",command=SEARCH\_CARS, width="30")

b1.place(x= 400,y=650 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

b4=tb.Button(root,text="DEVELOPER LOGIN",bootstyle="primary outline",command=DEVELOPER, width="30")

b4.place(x= 500,y=100 , anchor=NW)

b5=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b5.place(x= 700,y=100 , anchor=NW)

#

def Mumbai():

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

image\_path = r"C:\Users\chand\Downloads\RTP\BackGround1.jpg"

image = Image.open(image\_path)

image = image.resize((860, 600))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(f2, image=photo)

image\_label.image = photo

image\_label.place(x=100,y=135)

b1=tb.Button(root,text="EXPLORE CARS",bootstyle="success outline",command=SEARCH\_CARS, width="30")

b1.place(x= 400,y=650 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

b3=tb.Button(root,text="SHOWROOM",bootstyle="primary outline",command=LOGIN, width="30")

b3.place(x= 300,y=100 , anchor=NW)

b4=tb.Button(root,text="DEVELOPER LOGIN",bootstyle="primary outline",command=DEVELOPER, width="30")

b4.place(x= 500,y=100 , anchor=NW)

b5=tb.Button(root,text="CONTACT US",bootstyle="primary outline",command=CONTACT, width="30")

b5.place(x= 700,y=100 , anchor=NW)

def SEARCH\_CARS():

global e21,e22,selected\_option11,selected\_option22,selected\_option33,selected\_option44,selected\_option55,selected\_option66

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

c=Canvas(f2,bg='white',height='950',width='750')

c.place(x=300,y=0)

rec1=c.create\_rectangle((10,180,700,600))

l6 = tb.Label(f2, text="NEW ENTRY")

l6.place(x=600,y=150,anchor=NW)

label1 = tb.Label(f2, text=" MODEL")

label1.place(x=350,y=200,anchor=NW)

options11 = ["BRAND","HYUNDAI", "HONDA", "KIA","MARUTHISUZUKI","AUDI","MG","TATA","MAHINDRA"]

selected\_option11 = tb.StringVar()

combobox11 = tb.Combobox(f2,textvariable=selected\_option11, values=options11,bootstyle="info")

combobox11.place(x=435,y=190)

combobox11.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l1 = tb.Label(f2, text="SEATER")

l1.place(x=350,y=260,anchor=NW)

options22 = ["SEATER","4 SEATER", "5 SEATER", "6 SEATER","7 SEATER","8 SEATER"]

selected\_option22 = tb.StringVar()

combobox22 = tb.Combobox(f2,textvariable=selected\_option22, values=options22,bootstyle="info")

combobox22.place(x=435,y=250)

combobox22.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l2 = tb.Label(f2, text=" FUEL TYPE")

l2.place(x=350,y=320,anchor=NW)

options33 = ["FUEL","PETROL", "DIESEL", "CNG"]

selected\_option33 = tb.StringVar()

combobox33 = tb.Combobox(f2,textvariable=selected\_option33, values=options33,bootstyle="info")

combobox33.place(x=435,y=310)

combobox33.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l6 = tb.Label(f2, text="COLOR")

l6.place(x=700,y=320,anchor=NW)

options44 = ["COLOR","WHITE", "GREY", "RED","SILVER","BLUE","BLACK","GREEN","VIOLET","YELLOW"]

selected\_option44 = tb.StringVar()

combobox44 = tb.Combobox(f2,textvariable=selected\_option44, values=options44,bootstyle="info")

combobox44.place(x=785,y=310)

combobox44.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l7 = tb.Label(f2, text="CITY")

l7.place(x=700,y=380,anchor=NW)

options66 = ["CITY","Hyderabad", "Kolkata", "Delhi","Mumbai","Chennai"]

selected\_option66 = tb.StringVar()

combobox66 = tb.Combobox(f2,textvariable=selected\_option66, values=options66,bootstyle="info")

combobox66.place(x=785,y=370)

combobox66.current(0)

selected\_text = tb.StringVar()

selected\_label = tb.Label(f2, textvariable=selected\_text)

l3 = tb.Label(f2, text="MILEAGE")

l3.place(x=350,y=380,anchor=NW)

e21=tb.Entry(f2,width="30")

e21.place(x=435,y=370,anchor=NW)

l4 = tb.Label(f2, text="AMOUNT")

l4.place(x=350,y=440,anchor=NW)

e22=tb.Entry(f2,width="30")

e22.place(x=435,y=430,anchor=NW)

b8=tb.Button(f2,text="SEARCH",bootstyle="sucess outline",command=NEW\_SEARCH, width="30")

b8.place(x=580 ,y=550 , anchor=NW)

b2=tb.Button(root,text="HOME",bootstyle="primary outline",command=city, width="30")

b2.place(x= 100,y=100 , anchor=NW)

def NEW\_SEARCH():

global j

A=selected\_option66.get()

sqlqry3="SELECT SHOWROOM FROM DE WHERE CITY='"+A+"'"

cur.execute(sqlqry3)

res=cur.fetchall()

for jj in res:

SEARCH(jj)

#sqlqry1="SELECT COUNT(\*) FROM DE WHERE CITY='"+A+"'"

#cur.execute(sqlqry1)

#res1=cur.fetchone()

#count\_result = res1[0]

#res2 = count\_result + 1

#for aa in range(0,res2):

# SEARCH(res[aa])

def SEARCH(showroom\_name):

global row, col

str1="CARS"

S1 = [str(item) + str1 for item in showroom\_name]

a=(e21.get())

b=(e22.get())

c=selected\_option11.get()

d=selected\_option22.get()

e=selected\_option33.get()

f=selected\_option44.get()

g=b

# Convert the list into a string with elements separated by a comma

print(S1)

separator = ', '

s1 = separator.join(S1)

print(s1)

model ='MODEL'

seater='SEATER'

color='COLOR'

mileage='MILEAGE'

fuel='FUELTYPE'

minprice='MINPRICE'

maxprice='MAXPRICE'

QUR=("SELECT MODEL, CAR, SEATER, FUELTYPE, MINPRICE, MILEAGE, IMAGE,COUNT(\*),QUANTITY FROM {} where {}=%s and {}=%s and {}=%s and {}=%s and {}<=%s and {}>=%s and {}=%s".format(s1,model,mileage,seater,fuel,minprice,maxprice,color))

cur.execute(QUR,(c,a,d,e,b,g,f))

ans= cur.fetchall()

print(ans)

A=selected\_option66.get()

sqlqry4="SELECT COUNT(\*) FROM DE WHERE CITY='"+A+"'"

cur.execute(sqlqry4)

ans1=cur.fetchall()

max\_col = 3

max\_row = 15

i = 0

j = 0

k = 0

for j in range(0, max\_row):

if j >= len(ans):

break

for k in range(0, ans[j][7]):

if ans[k][8] > 0:

t = ans[k][1]

if i >= max\_col:

i = 0

j += 1

if j >= max\_row:

break

fji = tb.Frame(root, width=f"{screen\_width}",height=f"{screen\_height}")

fji.grid(row=j, column=i)

image\_path = rf"C:\Users\chand\Downloads\RTP\CARS1\CARS\{c}\{t}\{f}.jpg"

print(image\_path)

try:

image = Image.open(image\_path)

image = image.resize((250, 200))

photo = ImageTk.PhotoImage(image)

image\_label = tb.Label(fji, image=photo)

image\_label.image = photo

image\_label.place(x=15, y=15)

except Exception as e:

print(f"Error loading image: {e}")

lji1 = tb.Label(fji, text="Mileage-")

lji1.place(x=10, y=250, anchor=tb.NW)

lji2 = tb.Label(fji, text=ans[k][1])

lji2.place(x=10, y=210, anchor=tb.NW)

lji3 = tb.Label(fji, text=ans[k][2])

lji3.place(x=10, y=230, anchor=tb.NW)

lji4 = tb.Label(fji, text=ans[k][3])

lji4.place(x=100, y=230, anchor=tb.NW)

lji5 = tb.Label(fji, text=ans[k][4])

lji5.place(x=100, y=250, anchor=tb.NW)

lji6 = tb.Label(fji, text=ans[k][5])

lji6.place(x=55, y=250, anchor=tb.NW)

lji7 = tb.Label(fji, text="ShowRoom-")

lji7.place(x=10, y=270, anchor=tb.NW)

lji8 = tb.Label(fji, text=showroom\_name)

lji8.place(x=95, y=270, anchor=tb.NW)

lji9 = tb.Label(fji, text="₹")

lji9.place(x=90, y=250, anchor=tb.NW)

bji1 = tb.Button(fji, text="BOOK", bootstyle="success outline", command=lambda: BOOKING(), width=7)

bji1.place(x=60, y=290, anchor=tb.NW)

i += 1

def BOOKING():

global h1,h2,h3,h4

f2=Frame(root,width=f"{screen\_width}",height=f"{screen\_height}")

f2.place(x=0,y=0)

f2.pack\_propagate(0)

b2=tb.Button(root,text="BACK",bootstyle="primary outline",command=SEARCH\_CARS, width="30")

b2.place(x= 100,y=70 , anchor=NW)

c=Canvas(f2,bg='white',height='950',width='750')

c.pack()

rec1=c.create\_rectangle((10,180,520,400))

label1 = tb.Label(f2, text="CAR BOOKING")

label1.place(x=350,y=200,anchor=NW)

l1 = tb.Label(f2, text="CUSTOMER NAME")

l1.place(x=500,y=150,anchor=NW)

h1 = tb.Entry(f2, width="50")

h1.place(x=430,y=200,anchor=NW)

label2 = tb.Label(f2, text="PHONE NUMBER")

label2.place(x=350,y=260,anchor=NW)

h2 = tb.Entry(f2, width="50")

h2.place(x=430,y=255,anchor=NW)

label3 = tb.Label(f2, text="EMAIL")

label3.place(x=350,y=320,anchor=NW)

h3 = tb.Entry(f2, width="50")

h3.place(x=430,y=315,anchor=NW)

label4 = tb.Label(f2, text="ADDRESS")

label4.place(x=350,y=380,anchor=NW)

h4 = tb.Entry(f2, width="50")

h4.place(x=430,y=375,anchor=NW)

b4=tb.Button(f2,text="SUBMIT",bootstyle="sucess outline",command=USER\_ENTRY, width="30")

b4.place(x= 480,y=365 , anchor=NW)

def USER\_ENTRY():

a=h1.get()

b=h2.get()

c=h3.get()

d=h4.get()

e=m

b1=tb.Button(root,text="SELECT CITY",bootstyle="success outline",command=city, width="30")

b1.place(x= 580,y=480 , anchor=NW)

root.mainloop()