AspireDev

CAB BOOKING SYSTEM

PROJECT REPORT

Submitted by

Name	**:**	**A. PRATHESH KUMAR**
ID . No	**:**	**ASPDN1563**
STAFF NAME	**:**	**R. RAMYA**
COURSE NAME	**:**	**PYTHON FULL STACK**
DATE	**:**	**4/03/2023**

ABSTRACT

The Cab Booking System is being developed for customers so that they can book their cabs from any part of the world. It is a web applications made for pre-booking cab for particular period according to customer requirement. Cab Booking System is based on a concept to book cab and taxi online. The rental cab details are added in this web application by the cab rental owner. A customer being registered in the website has the facility to book a vehicle which he requires. The proposed system is a completely integrated online system. It automates manual procedure in an effective and efficient way.This automated system facilities customer and provides to fill up the details according to their requirements. It includes type of vehicle they are trying to hire and location.Here at first the customer has to login or sign up to get access. Then the customer can list, search for cabs according to their needs, check each cabs description, booking prices and book easily with help of various payment methods. The customer can filter search or sort cab by price for searching cabs. The web application also displays reserved cabs list. With the help of this web application, the online cab booking has become easier for the customers.

ACKNOWLEDGEMENT

First of all I thank God almighty for giving us the wisdom, grace and knowledge to complete this project successfully and also for showering his choicest blessing upon us.I express my gratitude to **R. RAMYA** I who co-ordinate and guided us in this project and all through this PYTHON course. I take this pleasant opportunity to convey my thanks to her for her kind co-operation, necessary requirements and access to complete my project work.

I am thankful to all Staffs of ASPIREDEV and to my friend who has given their timely support and helps to bring out our project as a successful one.

Above all I remain greatly thankful to my beloved Parents who inspired and guided us in the right path and for being the backbone of all the successful endeavours in our life.

A. PRATHESH KUMAR
(ASPDN 1563)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Nowadays, especially in metro cities, taxicabs play a crucial role in bridging the gap between public (e.g., state transport buses, Indian railways etc.) and private transportation. A taxicab is a vehicle service, hired along with the driver for either single or in a shared pool manner. It conveys passengers between the selected locations. Unlike public transport modes, where the pickup and drop-off locations are predefined by the service provider, taxis provide a hybrid bus/taxi mode. Indian regulations make it mandatory for all taxicabs to install fare-meter. The taxicabs are very popular means of transportation because it provides the comfortable, quicker and secure journey.

Worldwide, taxis fulfil the same goal but differ significantly in its style of booking, i.e., manual, automatic, or via brokers. This has gradually evolved with time and has surged from manual to a phone call or online mode. Manual cab booking system requires the passenger to physically book the cab through booking office which therefore provides the liberty and scope of bargaining at both ends. While in online mode, clients carry out booking either through phone or the Internet. This indeed is quicker and faster but hinders bargaining.

Cab Booking/Hiring is a method that may be utilized for a price for a limited length of time. Peoplewho do not have access to their own personal automobile or do not own one at all can travel aroundby renting a cab. In online booking system database of booked/available taxisis fed to the central server. All the bookings are query based which is directed to the central server and followed by acknowledgment by the server. Individuals who wish to hire or rent an automobile must first contact the desired vehicle's cab rental business. This may be done over the internet. This individual must now provide certain information, such as the rental dates and automobile type. Following the completion of these data, the person renting the automobile must produce a valid identification card. The majority of enterprises in the sector earn based on the sort of automobiles they sell. Customers can use an online booking system to rent cabs.

Customers may use this online system to find available taxis, register cabs, see profiles, and book cabs. Taxi booking is a typical kind of transportation that is offered by a number of different transportation firms in a particular city. The bulk of people rely on taxi services for their daily transportation needs. The company must be registered and fulfil all of the transportation department's requirements and security requirements. The Online Cab Booking System is a web-based platform that allows your customers to order taxis and executive cabs from their own home or office.

The objective of this project is to develop a web app to automate the manual cab booking system to overcome the drawbacks that are faced in the existing systems. The main objective of the cab Rental System is to manage the details of cab, orders, Customer and driver details. To design a user friendly system that enables client check for availability of vehicle and book or reserve a vehicle online.

CHAPTER 2

SYSTEM ANALYSIS

Systems design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

Analysis consists of two sub phases :

- Planning
- Requirements definition

During planning phase, cost estimation and work schedules will be planned. Requirement definition is a specification that describes the processing environment, the required software functions, performance constraints (size, speed, machine configuration) are exception handling.

2.1 EXISTING SYSTEM

In the existing system, there is no website or web application for cab booking system, all the processes are done in a manual way. The customer who needs to lease a vehicle, first contact the vehicle rental organization for making decision of vehicle. Existing system, which is the traditional system, in which the customer needs to go the Cab office and book manually, which require lots of physical and mental efforts. The organisation uses a manual system for reserving and renting a cab through phone calls placed to the cab operators by intending users. Under this manual method, there is little or no record of all the rental activities and customer information. During cab booking, the customer records are stored in paper format and are filed together. Existing system provide manual paper work or excel sheet to track the booking and registered cabs details. The user has to go in the office where the user can get the cab on rent and book their cab. Most of the time user does not get a sight of the cab in which he is planning to travel. Which results in compromising the travel comfort.

Drawbacks of Existing System

- Time consuming process
- Requires more human intervention
- More cost
- Unreliable
- Manual system does not allow client to booking online and hard to keep track on the record of rental cabs.

2.2 PROPOSED SYSTEM

Cab Booking System project will enable the user to rent a vehicle. In the proposed system, the admin i.e., the cab rental owner has the overall control over the web application. The admin login into the web application and add the cab details along with the rental price, seat and location details in the web application. The driver details including their contact number are added in the web application by the admin. The customer who wishes to rent a cab can make us of this web application, new user needs to register their details in the system and already existing customer can make use of this web application by login into the web application through login page. After login the customer can search for the available cabs by entering the city name, then the list of available cab details will be displayed to the customer from which the customer can book for the needed cab. The cab can be booked by entering the total number of days, after ordering the ordered details including rental amount, vehicle name, duration, cab dealer and contact number will be displayed to the customer. The added cab details can be viewed and modified by the rental owner and the rental owner can view the vehicle orders list and can update the status. Report regarding the order details and cab details can be generated and viewed by the rental owner.

Advantages

- The user may rent a cab online and look up the driver's details in easier way.
- Customer can order taxis from the convenience of their own homes or companies, which is handy for them.
- It's incredibly secure since only logged-in users may book a cab, preventing attackers from viewing sensitive data.
- It increases the efficiency of the system in offering high-quality services to its customers.

CHAPTER 3

SYSTEM SPECIFICATION

Systems implementation is the process of: defining how the information system should be built (i.e., physical system design), ensuring that the information system is operational and used, ensuring that the information system meets quality standard (i.e., quality assurance). The implementation phase involves putting the project plan into action. It's here that the project manager will coordinate and direct project resources to meet the objectives of the project plan. As the project unfolds, it's the project manager's job to direct and manage each activity, every step of the way.

3.1 HARDWARE SPECIFICATION

Processor	:	Intel i3 2.93 GHZ
RAM	:	2 GB
Floppy	:	1.44 MB
Mouse	:	Optical Mouse
Monitor	:	SVGA
Key board	:	104 Keys Standard
Hard disk	:	500 GB

3.2 SOFTWARE SPECIFICATION

Operating System	:	Windows XP or Higher
Front End	:	Python
Back End	:	My SQL

CHAPTER 4

LANGUAGE SPECIFICATION

4.1 PYTHON

Python is a high-linterpretedevel, general-purpose programming language. Created by Guidovan Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council

Python interpreters are available for many operating systems. A global community of programmers develops and maintains C Python, A non-profit organization, the Python Software Foundation, manages and directs resources for Python and C Python development.

Features of Python:

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming. Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at official web address. The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python OOPs Concepts

Like other general purpose languages, python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. In Python, we can easily create and use classes and objects.

Object

The object is an entity that has state and behaviour. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc. Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute __doc__, which returns the doc string defined in the function source code.

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Inheritance

Inheritance is the most important aspect of object-oriented programming which simulates the real world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object. By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.It provides re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many and Morphs means form, shape. By polymorphism, we understand that one task can be performed in different ways. For example a class having animal, and all animals speak. But they speak differently. Here, the "speak" behaviour is polymorphic in the sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an important aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonym because data abstraction is achieved through encapsulation. Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.Data abstraction is the reduction of a particular body of data to a simplified representation of the whole. Abstraction, in general, is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

Python 3.7

Python 3.7.0 it was released on June 27 my first attempts to run it on WSL (Windows Subsystem for Linux) Ubuntu didn’t quite go as planned. There’s no Debian or Ubuntu distribution of Python 3.7.0 at the moment, just the sources, so I used pyenv,which fetches the sources and build them. There’s a lot of new stuff in Python 3.7.0, much of it quite simple.

Features of Python 3.7.0

- The breakpoint() Built-In.
- Data Classes.
- Customization of Module Attributes.
- Typing Enhancements.
- Timing Precision.

Django 2.2.4

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Features

- Denial-of-service possibility in django.utils.text.Truncator
- Denial-of-service possibility in strip_tags()
- SQL injection possibility in key and index lookups for JSONField/HStoreField
- Potential memory exhaustion in django.utils.encoding.uri_to_iri()

4.2 MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter My, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB. MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

Features

The main features associated with MySQL are,

Open-Source

MySQL is open-source, which means this software can be downloaded, used and modified by anyone. It is free-to-use and easy-to-understand. The source code of MySQL can be studied, and changed based on the requirements. It uses GPL, i.e. GNU General Public license which defines rules and regulations regarding what can and can’t be done using the application.

Quick and Reliable

MySQL stores data efficiently in the memory ensuring that data is consistent, and not redundant. Hence, data access and manipulation using MySQL is quick.

Scalable

Scalability refers to the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines, and so on. MySQL server was developed to work with large databases.

Data Types

It contains multiple data types such as unsigned integers, signed integers, float (FLOAT), double (DOUBLE), character (CHAR), variable character (VARCHAR), text, blob, date, time, datetime, timestamp, year, and so on.

Character Sets

It supports different character sets, and this includes latin1 (cp1252 character encoding), German, Ujis, other Unicode character sets and so on.

Secure

It provides a secure interface since it has a password system which is flexible, and ensures that it is verified based on the host before accessing the database. The password is encrypted while connecting to the server.

Support for large databases

It comes with support for large databases, which could contain about 40 to 50 million records, 150,000 to 200,000 tables and up to 5,000,000,000 rows.

Client and Utility Programs

MySQL server also comes with many client and utility programs. This includes Command line programs such as 'mysqladmin' and graphical programs such as 'MySQL Workbench'. MySQL client programs are written in a variety of languages. Client library (code encapsulated in a module) can be written in C or C++ and would be available for clients that have C bindings.

CHAPTER 5

SYSTEM DESIGN

5.1 DATAFLOW DIAGRAM

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.

DFD for Context Level

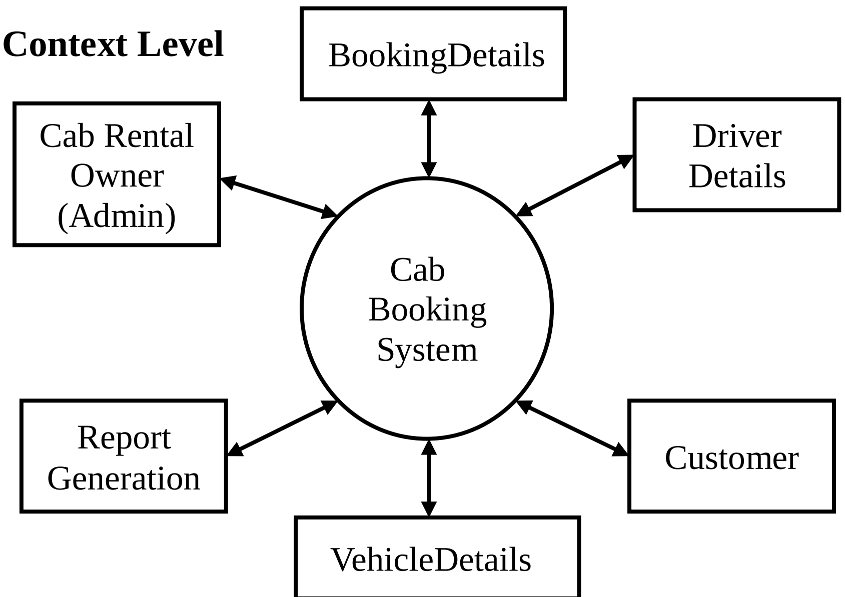


Figure 5.1 Context level DFD

The above fig 5.1 shows the level 0 DFD for context level. A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process. The cabbooking system involves admin, customer, driver details, booking details and vehicle details management.

DFD for Rental Owner

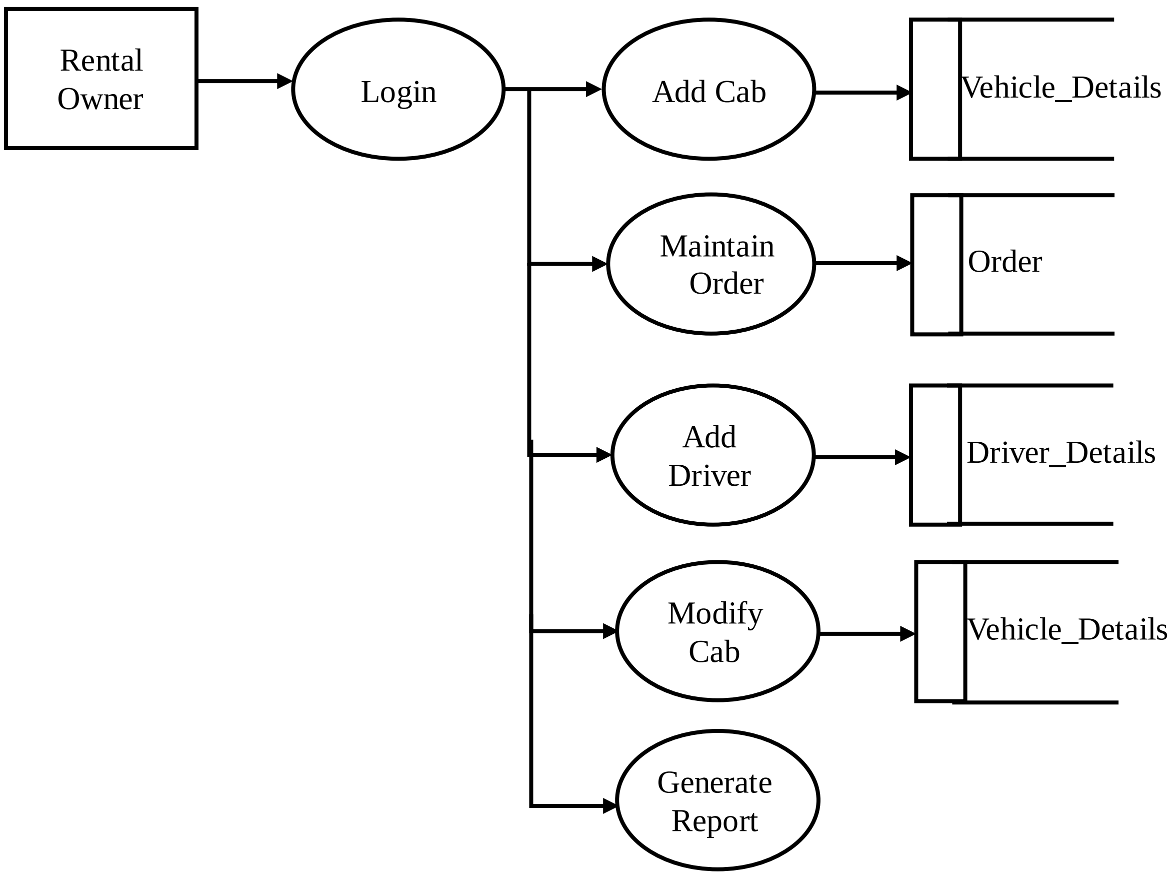


Figure 5.2 DFD for Rental Owner

The above figure 5.2 shows the DFD of admin of the rental shop. The admin will add the cab details in the DB of the web application and can also modify the added cab details. The driver details are added in the web application by the admin. Reports regarding the cab booking detailsare generated by the admin.

DFD for Customer

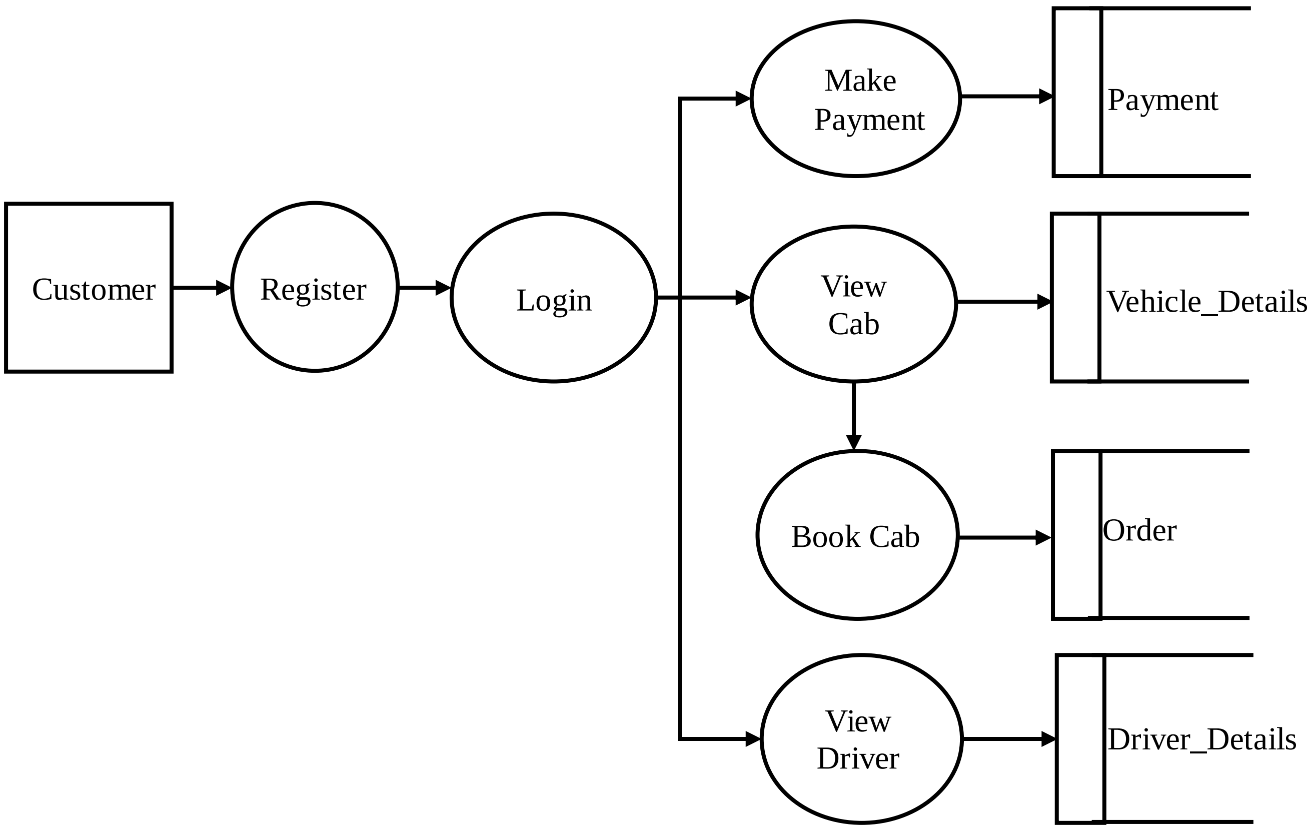


Figure 5.3 DFD for Customer

The above figure shows the DFD for the customer. The customer can view and book cab based on their need by login into the web application.New customer needs to register into the web application and need to create a login credentials. From the web application, the customer can view the driver details. The payment for the cab be made by the customer through online mode.

5.2 DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database management system manages the data accordingly. Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called an ontology.

Table 5.1: Area

NAME	TYPE
Pincode	varchar(10)
City	varchar(100)

Table 5.2: Cabdealer

NAME	TYPE
car_dealer	varchar(100)
Mobile	varchar(50)
Area	varchar(500)
wallet	int(10)

Table 5.3: Customer

NAME	TYPE
User	varchar(50)
Mobile	varchar(20)
Area	varchar(500)

Table 5.4: Orders

NAME	TYPE
User	varchar(50)
car_dealer	varchar(100)
Rent	varchar(10)
Vehicle	varchar(100)

Days	int(10)
is_complete	tinyint(1)

Table 5.5: Vehicles

NAME	TYPE
car_name	varchar(100)
Color	varchar(100)
Dealer	varchar(100)
Area	varchar(500)
Capacity	varchar(100)
is_available	tinyint(1)
Description	varchar(500)

Table 5.6: Driverdetails

NAME	**TYPE**
Did	int(11)
Dname	varchar(100)
Mobile	varchar(50)
car_name	varchar(100)

CHAPTER 6

PROJECT DESCRIPTION

6.1 MODULES

A module is a collection of source files and build settings that allow you to divide your project into discrete units of functionality. The project can have one or many modules and one module may use another module as a dependency. Each module can be independently built, tested, and debugged. The Project Development module includes criteria that span the entire project development process from early planning, alternatives analysis, environmental documentation, preliminary and final design, and construction.The modules used in this project are

- Rental Owner / Admin
- Customer
- Driver
- Booking Details
- Vehicle Details
- Report

6.2 MODULE DESCRIPTION

6.2.1 Rental Owner

Rental owner is the admin of the web application. The admin has the overall control over the web application. The rental owner login into the web application using their login credentials i.e., username and password. After login into the web application, the admin of the rental shop will add the available vehicle details in the web application along with the rental price details. The vehicle details include cab location, cab name, type, seats available and colour of the vehicle. The rental owner also views the booking done by the customer and provide cab rental service to the customer.The admin can modify or delete the added cab details in the database of the web application. The driver details are added in the web application by the admin. The driver details include the contact details of the driver. Reports are generated by the admin. The reports are generated based on the cab booking details and customer.

6.2.2 Customer

The customer or user who wishes to rent a cab can make use of this web application. New user needs to register their details in the web application. At the time of registration the customer needs to create a login credentials i.e., username and password. Already existing user can login into the web application using their username and password. After login into the web application the user can search for the needed cab through the web application. The customer can filter the cabs available in the particular location by entering the location details. When the location is entered, the cabs available in the particular location will be displayed to the user. The user can view the cabs description and can book for the needed cab through this web application. While booking the customer needs to enter the no of days the customer wishes to rent the cab. After booking the rental amount will be calculated and will be displayed to the customer. The customer can make payment for the cab rental through online mode.

6.2.3 Driver

A driver is a person whose job is to take people in a cab to the place they want to go to in return for money. The driver details are added in the web application by the admin. The admin will allot each driver based on the cab booking order. The driver will pick up the customer from the customer location based on the work allotted by the rental owner.

6.2.4 Booking Details

Booking details module is managed by the admin or rental owner. The booking details include the cab number, customer number, cab colour, seats number, rental days and location details. The booking done by customers and added vehicle details are managed in this module.

6.2.5 Vehicle Details

Vehicle details module is managed by the admin or rental owner. The vehicle details are added by the admin of the rental shop. The vehicle details are added along with the vehicle description, colour, name, seats available and price details. The vehicle details include booked vehicle details, remaining vehicle available for rent and rental price.

6.2.6 Report

The report generation module allows you to directly extract all the information you want from the database and either view it directly online or export it in open formats. The admin / rental owner can generate reports based on the booked vehicle details and the customer details. The reports can be generated in weekly, monthly and yearly basis.

CHAPTER 7

CONCLUSION

Cab booking system was implemented successfully using python and MySQLas database. Using this web application, the customer can book for cab in an easier and efficient manner.It also serves an easier way for the users to book any sort of cab for rent from anywhere at any time through this website.Using this web app, the user has the freedom of bookingany cab of his choice as per the occasion. Also this systemprovides a payment entrance victimization that user will build paymenteither by using debit or master card. The proposedapproach is able to deter the single point of failure, and utilizethe local information of the different region of the city to improve the cab availability.

CHAPTER 8

FUTURE ENHANCEMENT

Cab booking system can be extended in future by adding the feature of feedback along with the proposed work. Thereby the customer can give feedback regarding the driver and the cab facility. In the future, this project can also be improved by implementing the project in android app format. Since, applications are usually 1.5 times faster than mobile websites and they perform actions much faster too. Applications store their data locally on the user device. By using apps it improve their processes and increase the level of accessibility their customers have to them. The point of a mobile app is to seamlessly connect and interact with customers, making it a valuable tool for the modern business.

APPENDIX A
SOURCE CODE

View.py

```
from django.shortcuts import render

from django.http import HttpResponse

from django.contrib.auth.models import User

from django.contrib.auth import authenticate

from django.contrib import auth

from car_dealer_portal.models import *

from customer_portal.models import *

from django.contrib.auth.decorators import login_required

from django.http import HttpResponseRedirect

# Create your views here.


def index(request):

    if not request.user.is_authenticated:

        return render(request, 'car_dealer/login.html')

    else:

        return render(request, 'car_dealer/home_page.html')


def login(request):
    return render(request, 'car_dealer/login.html')



def auth_view(request):
    if request.user.is_authenticated:
        return render(request, 'car_dealer/home_page.html')
    else:
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        try:
            car_dealer = CarDealer.objects.get(car_dealer = user)
        except:
            car_dealer = None
        if car_dealer is not None:
            auth.login(request, user)
            return render(request, 'car_dealer/home_page.html')
        else:
            return render(request, 'car_dealer/login_failed.html')


def logout_view(request):
    auth.logout(request)
    return render(request, 'car_dealer/login.html')


def register(request):
    return render(request, 'car_dealer/register.html')


def registration(request):
    username = request.POST['username']
    password = request.POST['password']
    mobile = request.POST['mobile']
    firstname = request.POST['firstname']
    lastname = request.POST['lastname']
    email = request.POST['email']
    city = request.POST['city']
    city = city.lower()
    pincode = request.POST['pincode']


    try:
        user = User.objects.create_user(username = username, password =
password, email = email)
        user.first_name = firstname
        user.last_name = lastname
        user.save()
    except:
        return render(request, 'car_dealer/registration_error.html')
    try:
        area = Area.objects.get(city = city, pincode = pincode)
    except:
        area = None
    if area is not None:
        car_dealer = CarDealer(car_dealer = user, mobile = mobile, area=area)
    else:
        area = Area(city = city, pincode = pincode)
        area.save()
        area = Area.objects.get(city = city, pincode = pincode)
        car_dealer = CarDealer(car_dealer = user, mobile = mobile, area=area)
    car_dealer.save()
    return render(request, 'car_dealer/registered.html')

@login_required
def add_vehicle(request):
    car_name = request.POST['car_name']
    color = request.POST['color']
    cd = CarDealer.objects.get(car_dealer=request.user)

    city = request.POST['city']

    city = city.lower()

    pincode = request.POST['pincode']

    description = request.POST['description']

    capacity = request.POST['capacity']

    try:

        area = Area.objects.get(city = city, pincode = pincode)

    except:

        area = None

    if area is not None:

        car = Vehicles(car_name=car_name, color=color, dealer=cd, area = area,
description = description, capacity=capacity)

    else:

        area = Area(city = city, pincode = pincode)

        area.save()

        area = Area.objects.get(city = city, pincode = pincode)

        car = Vehicles(car_name=car_name, color=color, dealer=cd, area =
area,description=description, capacity=capacity)

    car.save()

    return render(request, 'car_dealer/vehicle_added.html')
```

```
@login_required

def manage_vehicles(request):

    username = request.user

    user = User.objects.get(username = username)

    car_dealer = CarDealer.objects.get(car_dealer = user)

    vehicle_list = []

    vehicles = Vehicles.objects.filter(dealer = car_dealer)

    for v in vehicles:

        vehicle_list.append(v)

    return render(request, 'car_dealer/manage.html', 
{'vehicle_list':vehicle_list})


@login_required

def order_list(request):

    username = request.user

    user = User.objects.get(username = username)

    car_dealer = CarDealer.objects.get(car_dealer = user)

    orders = Orders.objects.filter(car_dealer = car_dealer)

    order_list = []

    for o in orders:

        if o.is_complete == False:

            order_list.append(o)
    return render(request, 'car_dealer/order_list.html', {'order_list':order_list})


@login_required

def complete(request):

    order_id = request.POST['id']

    order = Orders.objects.get(id = order_id)

    vehicle = order.vehicle

    order.is_complete = True

    order.save()

    vehicle.is_available = True

    vehicle.save()

    return HttpResponseRedirect('/car_dealer_portal/order_list/')



@login_required

def history(request):

    user = User.objects.get(username = request.user)

    car_dealer = CarDealer.objects.get(car_dealer = user)

    orders = Orders.objects.filter(car_dealer = car_dealer)

    order_list = []

    for o in orders:
        order_list.append(o)
    return render(request, 'car_dealer/history.html', {'wallet':car_dealer.wallet,
'order_list':order_list})


@login_required
def delete(request):
    veh_id = request.POST['id']
    vehicle = Vehicles.objects.get(id = veh_id)
    vehicle.delete()
    return HttpResponseRedirect('/car_dealer_portal/manage_vehicles/')


def index(request):
    if not request.user.is_authenticated:
        return render(request, 'customer/login.html')
    else:
        return render(request, 'customer/home_page.html')




def login(request):
    return render(request, 'customer/login.html')


def auth_view(request):
    if request.user.is_authenticated:
        return render(request, 'customer/home_page.html')
    else:
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        try:
            customer = Customer.objects.get(user = user)
        except:
            customer = None
        if customer is not None:
            auth.login(request, user)
            return render(request, 'customer/home_page.html')
        else:
            return render(request, 'customer/login_failed.html')


def logout_view(request):
    auth.logout(request)
    return render(request, 'customer/login.html')
```

```
def register(request):
    return render(request, 'customer/register.html')



def registration(request):
    username = request.POST['username']
    password = request.POST['password']
    mobile = request.POST['mobile']
    firstname = request.POST['firstname']
    lastname = request.POST['lastname']
    email = request.POST['email']
    city = request.POST['city']
    city = city.lower()
    pincode = request.POST['pincode']
    try:
        user = User.objects.create_user(username = username, password =
password, email = email)
        user.first_name = firstname
        user.last_name = lastname
        user.save()
    except:
        return render(request, 'customer/registration_error.html')
    try:
        area = Area.objects.get(city = city, pincode = pincode)
    except:
        area = None
    if area is not None:
        customer = Customer(user = user, mobile = mobile, area = area)
    else:
        area = Area(city = city, pincode = pincode)
        area.save()
        area = Area.objects.get(city = city, pincode = pincode)
        customer = Customer(user = user, mobile = mobile, area = area)

    customer.save()
    return render(request, 'customer/registered.html')



@login_required
def search(request):
    return render(request, 'customer/search.html')
```

```
@login_required

def search_results(request):

    city = request.POST['city']

    city = city.lower()

    vehicles_list = []

    area = Area.objects.filter(city = city)

    for a in area:

        vehicles = Vehicles.objects.filter(area = a)

        for car in vehicles:

            if car.is_available == True:

                vehicle_dictionary = {'name':car.car_name, 'color':car.color,
'id':car.id, 'pincode':car.area.pincode, 'capacity':car.capacity,
'description':car.description}

                vehicles_list.append(vehicle_dictionary)

    request.session['vehicles_list'] = vehicles_list

    return render(request, 'customer/search_results.html')



@login_required

def rent_vehicle(request):

    id = request.POST['id']
    vehicle = Vehicles.objects.get(id = id)
    cost_per_day = int(vehicle.capacity)*13
    return render(request, 'customer/confirmation.html', {'vehicle':vehicle,
'cost_per_day':cost_per_day})



@login_required
def confirm(request):
    vehicle_id = request.POST['id']
    username = request.user
    user = User.objects.get(username = username)
    days = request.POST['days']
    vehicle = Vehicles.objects.get(id = vehicle_id)
    if vehicle.is_available:
        car_dealer = vehicle.dealer
        rent = (int(vehicle.capacity))*13*(int(days))
        car_dealer.wallet += rent
        car_dealer.save()
        try:
            order = Orders(vehicle = vehicle, car_dealer = car_dealer, user = user,
rent=rent, days=days)
            order.save()
        except:
            order = Orders.objects.get(vehicle = vehicle, car_dealer = car_dealer,
user = user, rent=rent, days=days)
        vehicle.is_available = False
        vehicle.save()
        return render(request, 'customer/confirmed.html', {'order':order})
    else:
        return render(request, 'customer/order_failed.html')



@login_required
def manage(request):
    order_list = []
    user = User.objects.get(username = request.user)
    try:
        orders = Orders.objects.filter(user = user)
    except:
        orders = None
    if orders is not None:
        for o in orders:
            if o.is_complete == False:
```

```
            order_dictionary = {'id':o.id,'rent':o.rent, 'vehicle':o.vehicle, 
'days':o.days, 'car_dealer':o.car_dealer}
            order_list.append(order_dictionary)
    return render(request, 'customer/manage.html', {'od':order_list})



@login_required
def update_order(request):
    order_id = request.POST['id']
    order = Orders.objects.get(id = order_id)
    vehicle = order.vehicle
    vehicle.is_available = True
    vehicle.save()
    car_dealer = order.car_dealer
    car_dealer.wallet -= int(order.rent)
    car_dealer.save()
    order.delete()
    cost_per_day = int(vehicle.capacity)*13
    return render(request, 'customer/confirmation.html', {'vehicle':vehicle}, 
{'cost_per_day':cost_per_day})
```

```
@login_required
def delete_order(request):
    order_id = request.POST['id']
    order = Orders.objects.get(id = order_id)
    car_dealer = order.car_dealer
    car_dealer.wallet -= int(order.rent)
    car_dealer.save()
    vehicle = order.vehicle
    vehicle.is_available = True
    vehicle.save()
    order.delete()
    return HttpResponseRedirect('/customer_portal/manage/')
```

models.py

```
from django.db import models
from django.db import models
from django.core.validators import *
from django.contrib.auth.models import User
```

```
# Create your models here.
class Area(models.Model):
    pincode = models.CharField(validators = [MinLengthValidator(6),
MaxLengthValidator(6)],max_length = 6,unique=True)
    city = models.CharField(max_length = 20)


class CarDealer(models.Model):
    car_dealer = models.OneToOneField(User, on_delete=models.CASCADE)
    mobile = models.CharField(validators = [MinLengthValidator(10),
MaxLengthValidator(13)], max_length = 13)
    area = models.OneToOneField(Area, on_delete=models.PROTECT)
    wallet = models.IntegerField(default = 0)


class Vehicles(models.Model):
    car_name = models.CharField(max_length = 20)
    color = models.CharField(max_length = 10)
    dealer = models.ForeignKey(CarDealer, on_delete = models.PROTECT)
    area = models.ForeignKey(Area, on_delete=models.SET_NULL, null =
True)
    capacity = models.CharField(max_length = 2)
    is_available = models.BooleanField(default = True)
    description = models.CharField(max_length = 100)



class DriverDetails(models.Model):
    did = models.IntegerField(default = 0)
    dname = models.CharField(max_length=100)
    mobile = models.CharField(validators = [MinLengthValidator(10),
MaxLengthValidator(13)], max_length = 13)
    car_name = models.CharField(max_length = 20)

class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    mobile = models.CharField(validators = [MinLengthValidator(10),
MaxLengthValidator(13)], max_length = 13)
    area = models.ForeignKey(Area, on_delete=models.PROTECT)



class Orders(models.Model):
    user = models.ForeignKey(User, on_delete=models.PROTECT)
    car_dealer = models.ForeignKey(CarDealer,
on_delete=models.PROTECT)
```

```
rent = models.CharField(max_length=8)

vehicle = models.ForeignKey(Vehicles, on_delete=models.PROTECT)

days = models.CharField(max_length = 3)

is_complete = models.BooleanField(default = False)
```

APPENDIX B

SCREENSHOTS

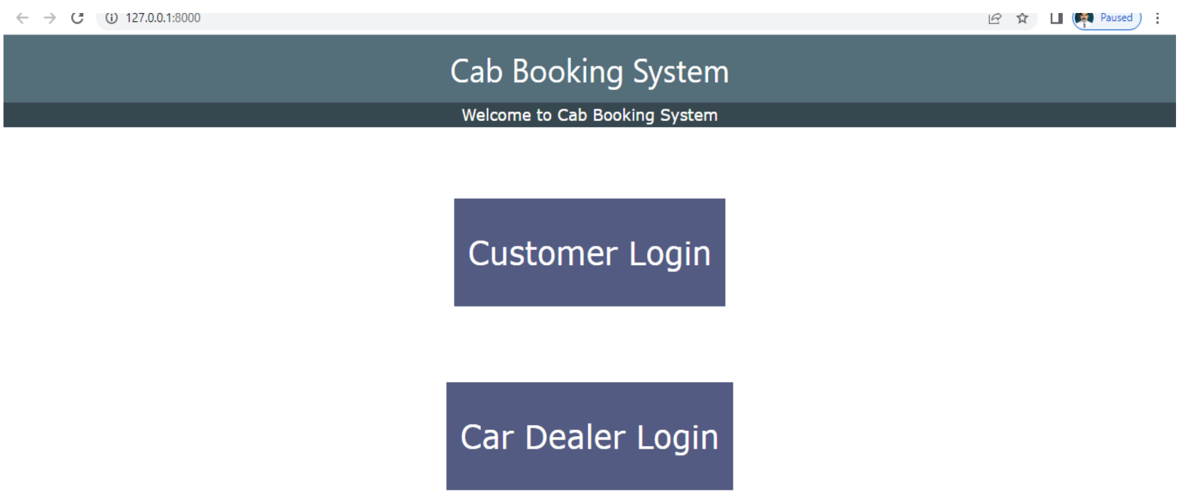
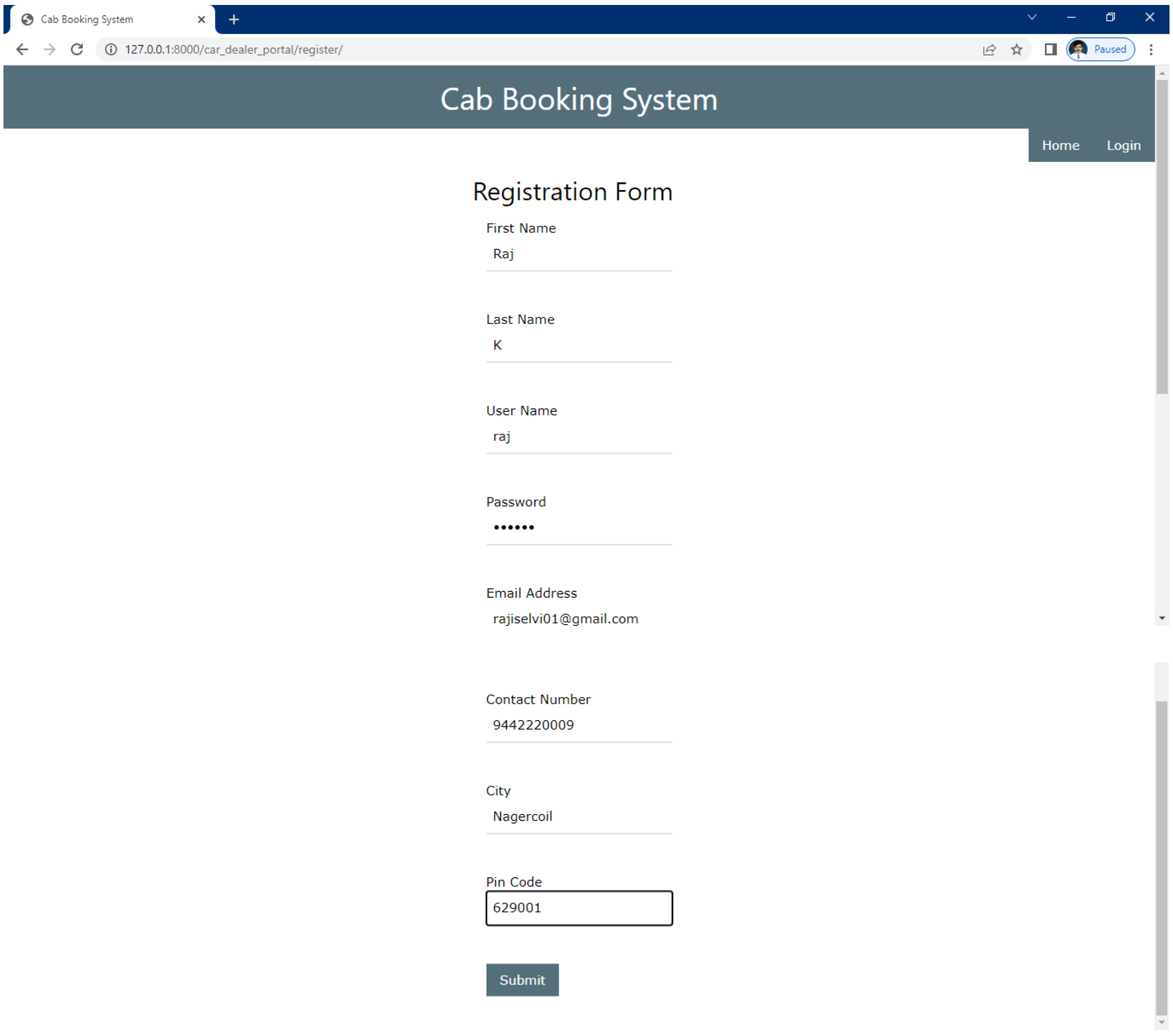


Figure B.1 Login Page


Cab Booking System

127.0.0.1:8000/car_dealer_portal/register/

Cab Booking System

Home Login

Registration Form

First Name: Raj

Last Name: K

User Name: raj

Password: ••••••

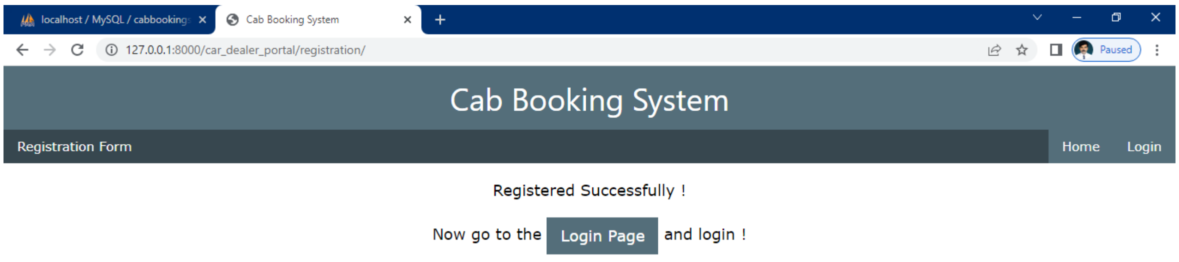
Email Address: rajiselvi01@gmail.com

Contact Number: 9442220009

City: Nagercoil

Pin Code: 629001

Submit


localhost / MySQL / cabbooking

Cab Booking System

127.0.0.1:8000/car_dealer_portal/registration/

Cab Booking System

Registration Form

Home Login

Registered Successfully !

Now go to the Login Page and login !

Figure B.2 Cab Dealer Registration

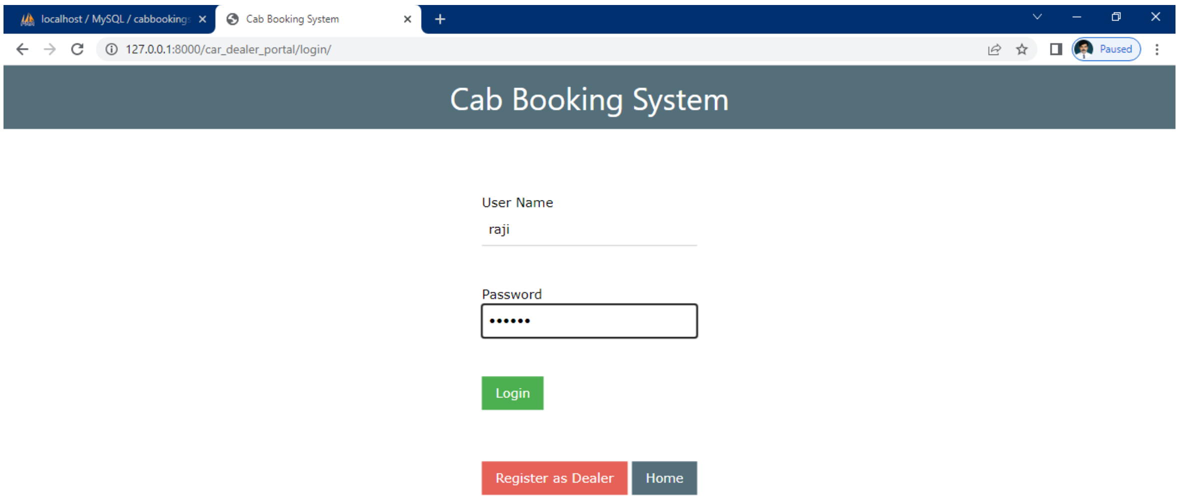


Figure B.3 Dealer Login

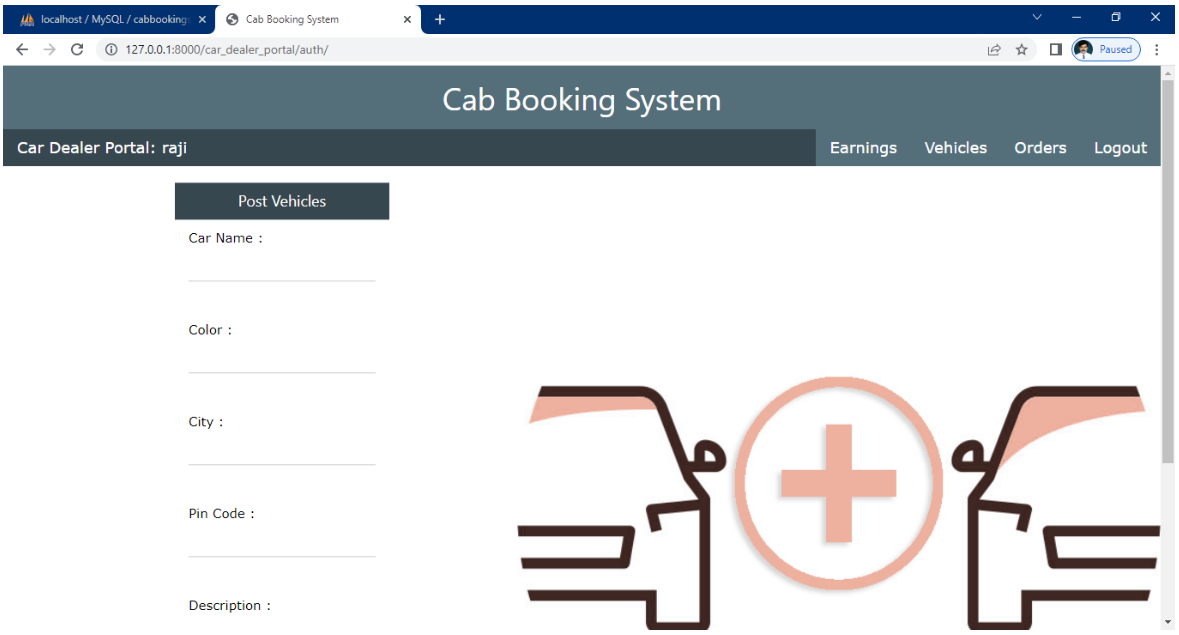
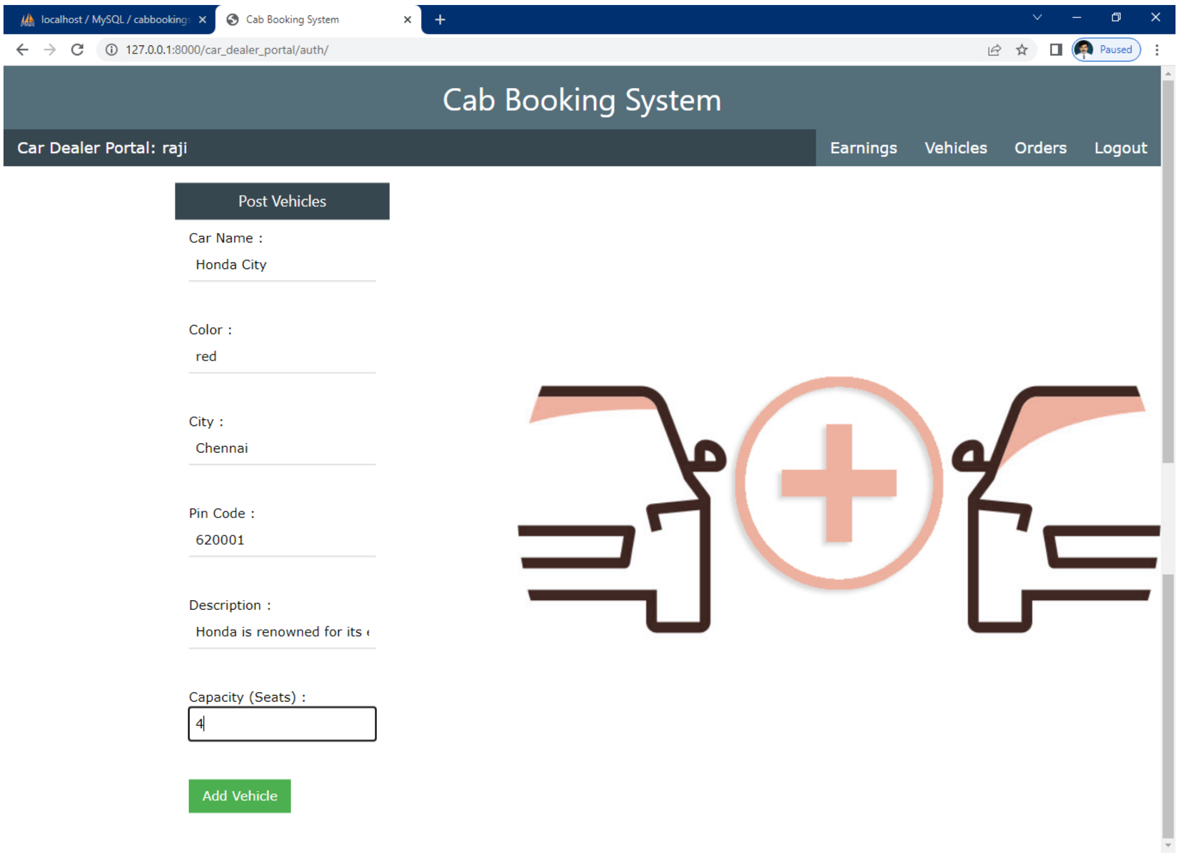


Figure B.4 Dealer Homepage


localhost / MySQL / cabbooking
Cab Booking System
127.0.0.1:8000/car_dealer_portal/auth/
Paused
Earnings
Vehicles
Orders
Logout
Add Vehicle

Cab Booking System

Car Dealer Portal: raji

Post Vehicles

Car Name : Honda City

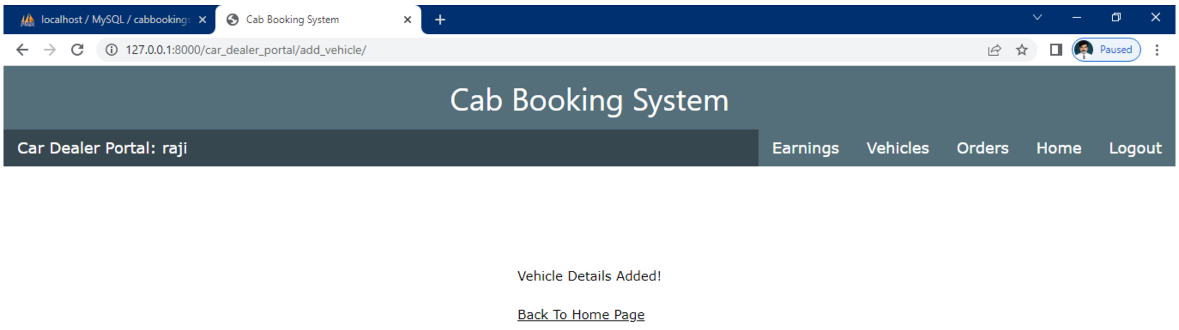
Color : red

City : Chennai

Pin Code : 620001

Description : Honda is renowned for its

Capacity (Seats) : 4

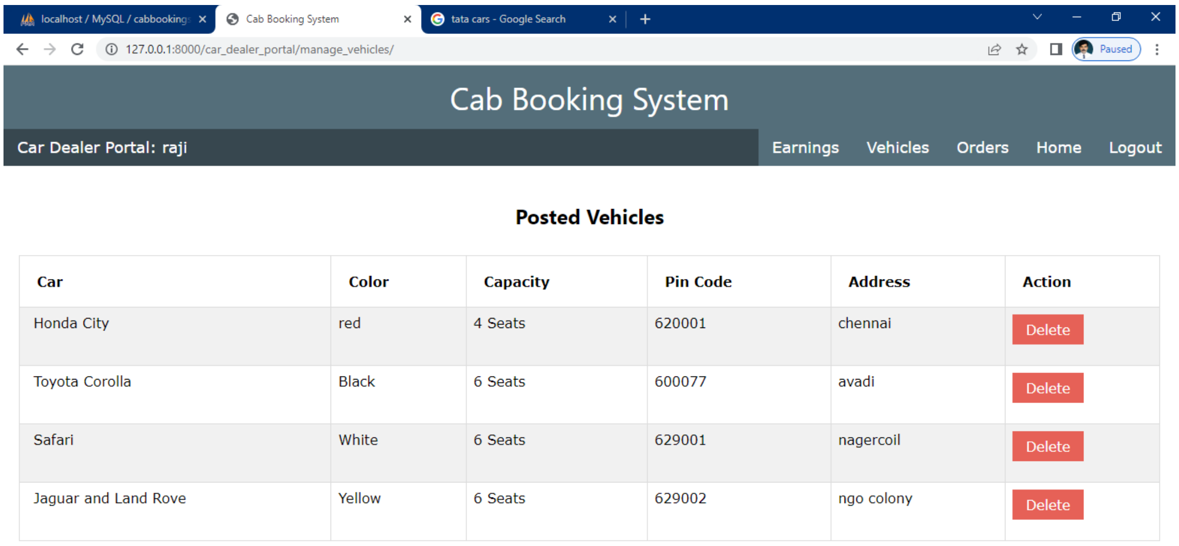

localhost / MySQL / cabbooking
Cab Booking System
127.0.0.1:8000/car_dealer_portal/add_vehicle/
Paused
Earnings
Vehicles
Orders
Home
Logout
Back To Home Page

Cab Booking System

Car Dealer Portal: raji

Vehicle Details Added!

Figure B.5 Adding New Car



localhost / MySQL / cabbooking
Cab Booking System
tata cars - Google Search
127.0.0.1:8000/car_dealer_portal/manage_vehicles/

Cab Booking System

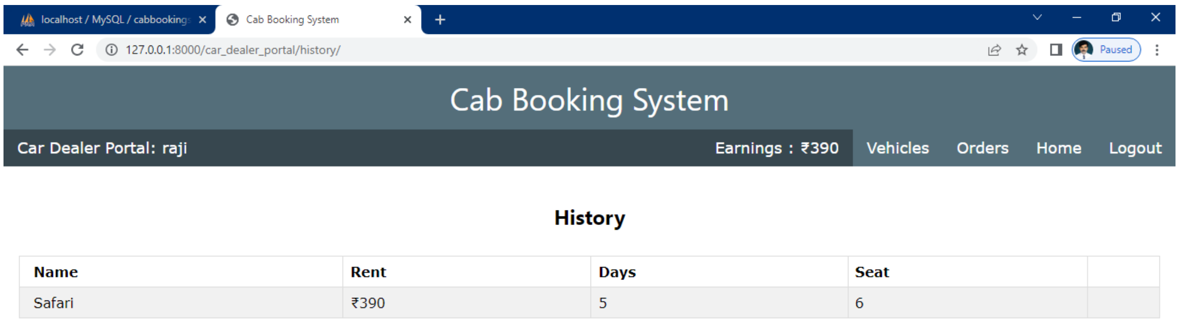
Car Dealer Portal: raji

Earnings Vehicles Orders Home Logout

Posted Vehicles

Car	Color	Capacity	Pin Code	Address	Action
Honda City	red	4 Seats	620001	chennai	Delete
Toyota Corolla	Black	6 Seats	600077	avadi	Delete
Safari	White	6 Seats	629001	nagercoil	Delete
Jaguar and Land Rove	Yellow	6 Seats	629002	ngo colony	Delete

Figure B.5 Display of Vehicles



localhost / MySQL / cabbooking
Cab Booking System
127.0.0.1:8000/car_dealer_portal/history/

Cab Booking System

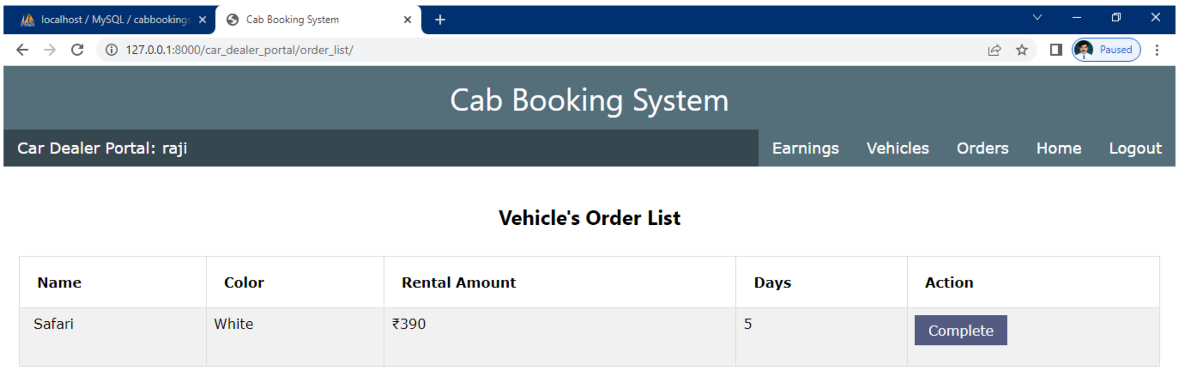
Car Dealer Portal: raji

Earnings : ₹390 Vehicles Orders Home Logout

History

Name	Rent	Days	Seat	
Safari	₹390	5	6	

Figure B.6 Earning History



127.0.0.1:8000/car_dealer_portal/order_list/

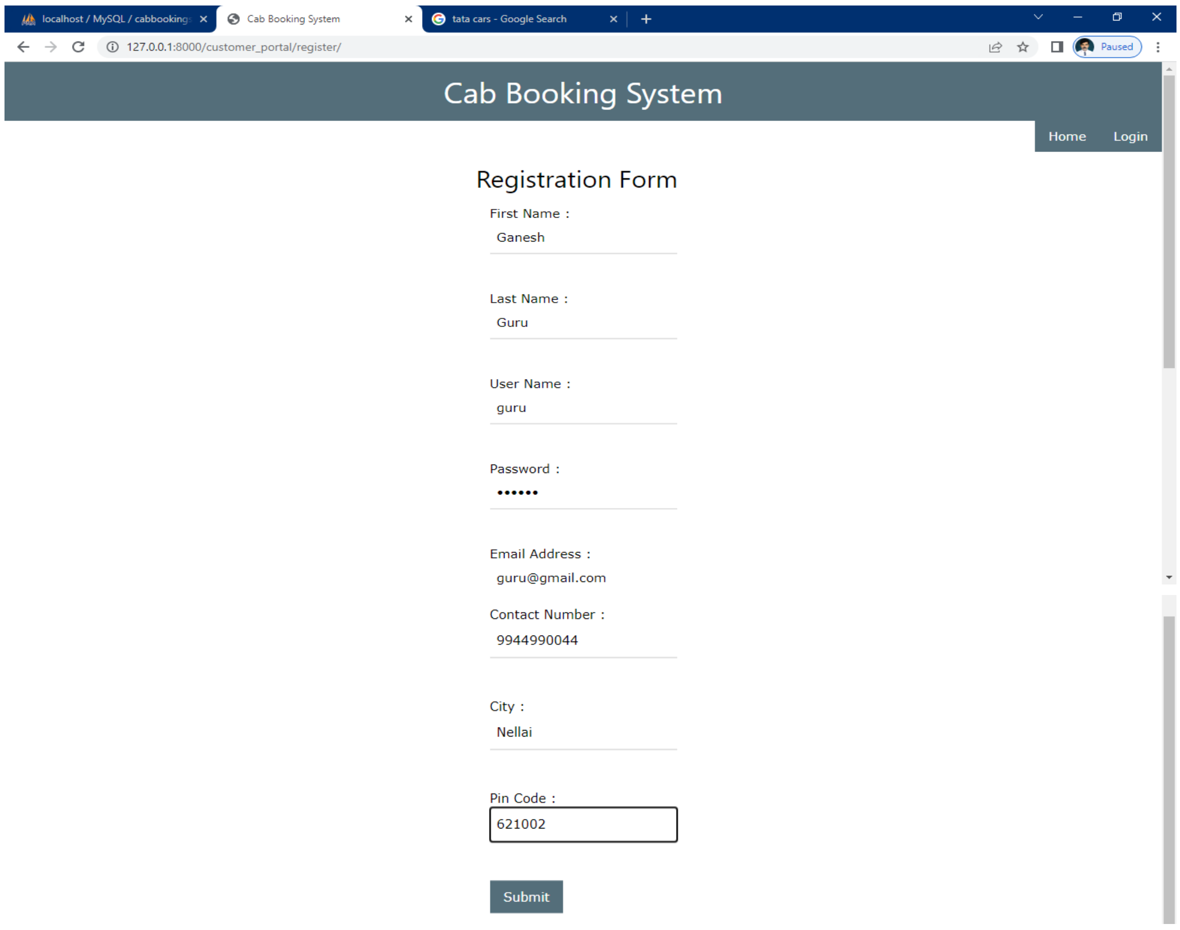
Cab Booking System

Car Dealer Portal: raji

Vehicle's Order List

Name	Color	Rental Amount	Days	Action
Safari	White	₹390	5	Complete

Figure B.7 Cab Booking History



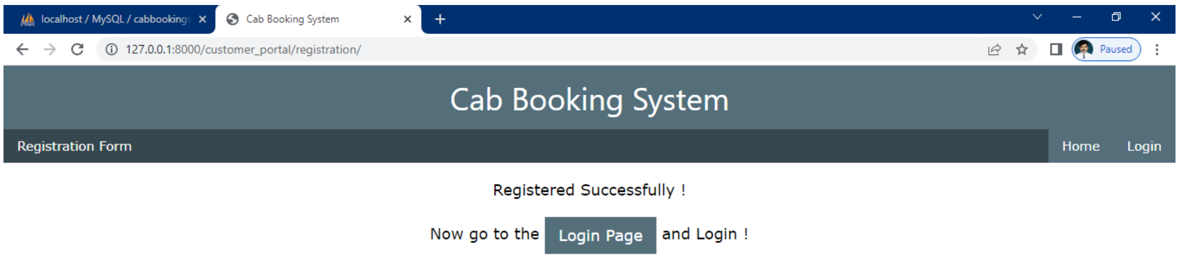
127.0.0.1:8000/customer_portal/register/

Cab Booking System

Registration Form

First Name :	Ganesh
Last Name :	Guru
User Name :	guru
Password :	••••••
Email Address :	guru@gmail.com
Contact Number :	9944990044
City :	Nellai
Pin Code :	621002

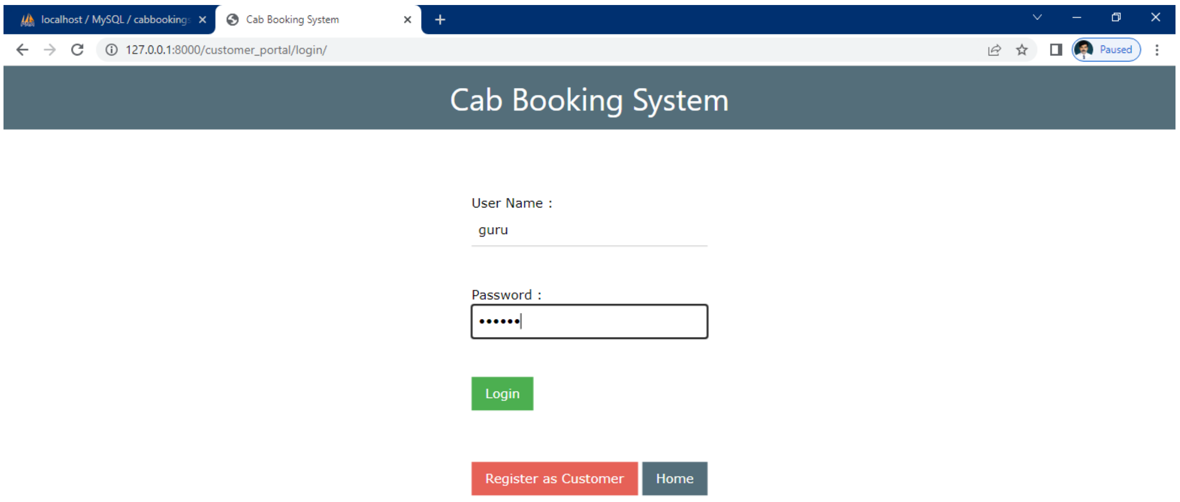
Figure B.8 Customer Registration



Registered Successfully !

Now go to the Login Page and Login !

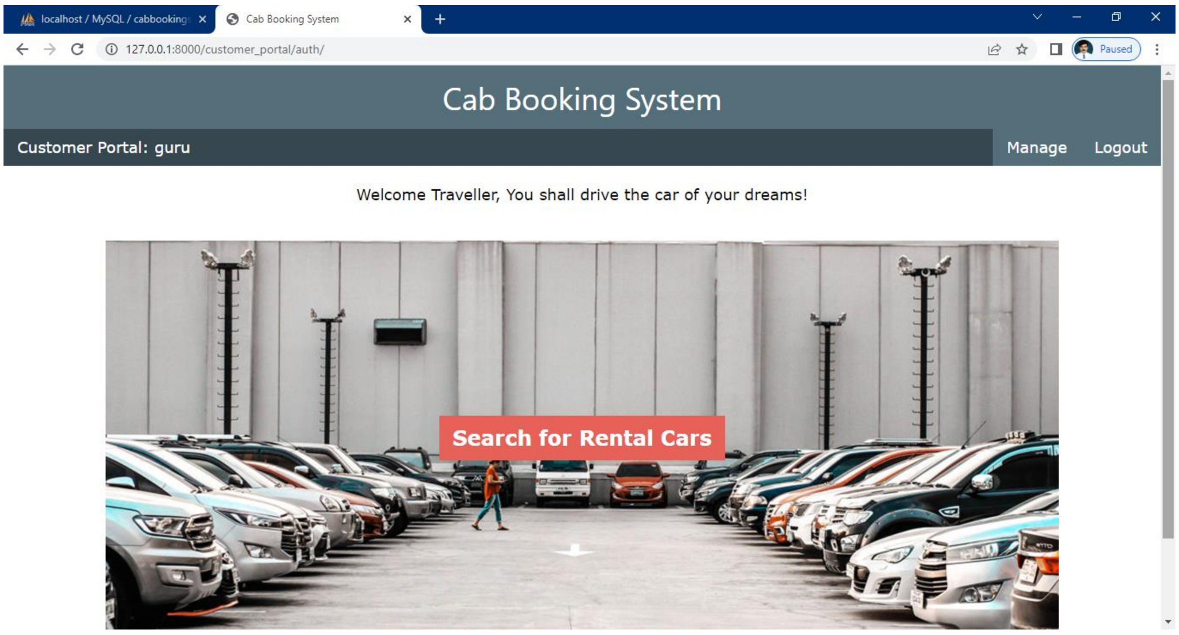
Figure B.9 Customer Successful Registration Notification



User Name : guru

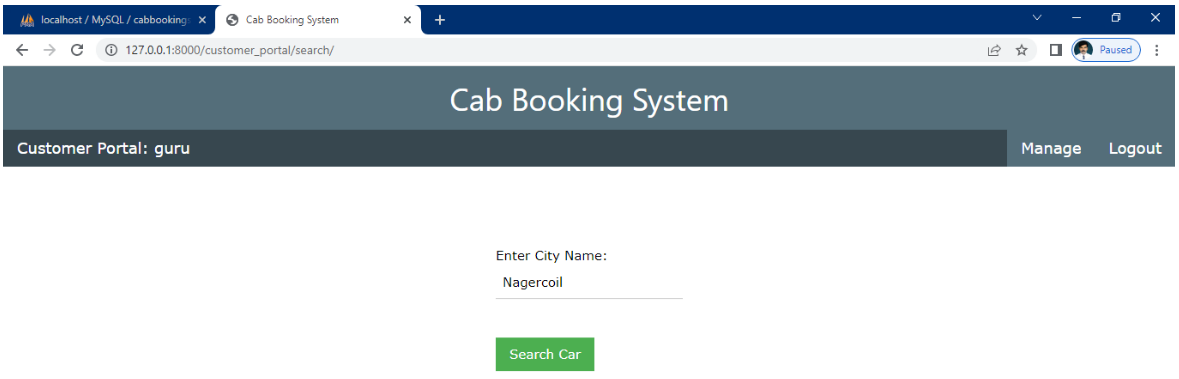
Password : ••••••

Figure B.10 Customer Login



Welcome Traveller, You shall drive the car of your dreams!

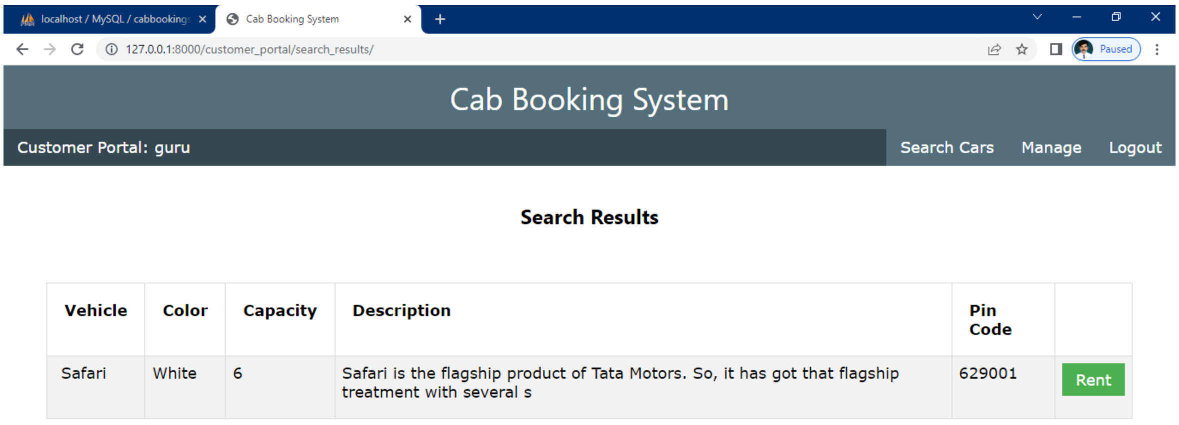
Figure B.11 Customer Homepage



Enter City Name:

Nagercoil

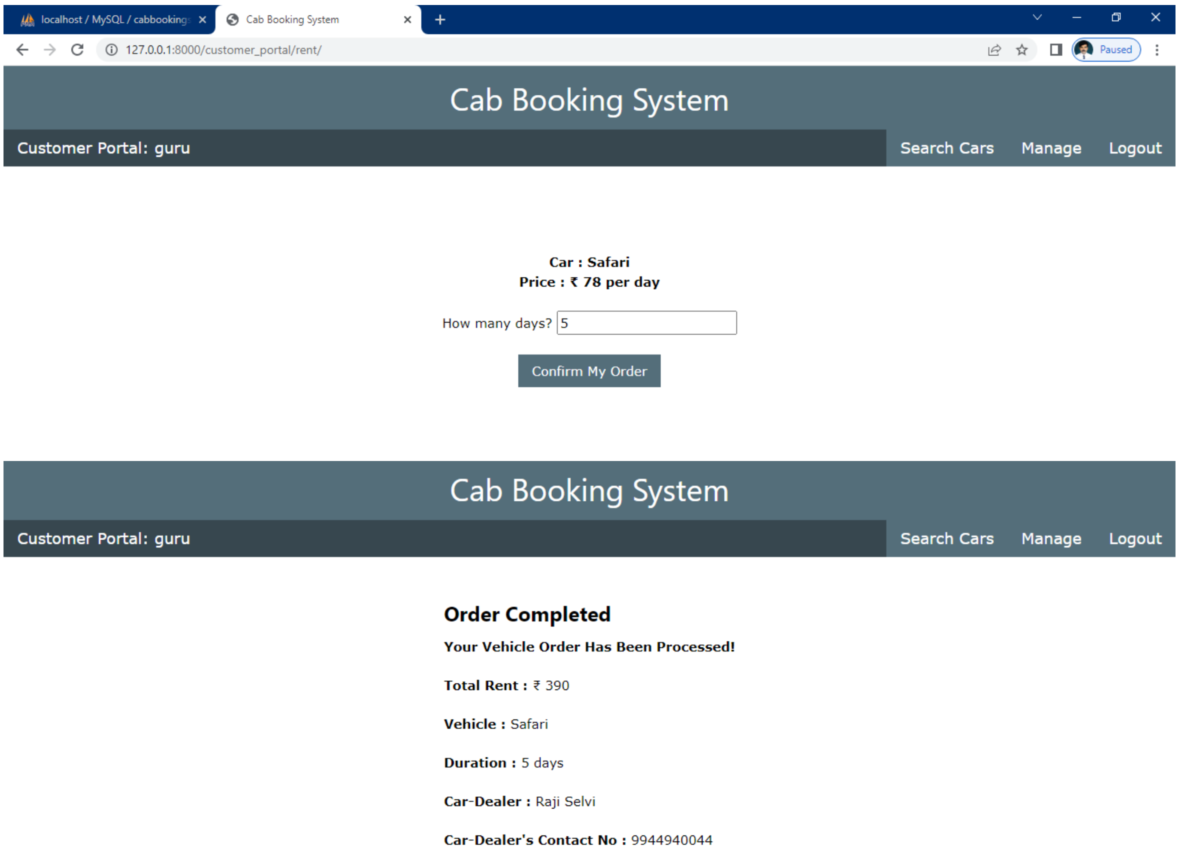
Figure B.12 Cab Search



Search Results

Vehicle	Color	Capacity	Description	Pin Code	
Safari	White	6	Safari is the flagship product of Tata Motors. So, it has got that flagship treatment with several s	629001	Rent

Figure B.13 Cab Search Results



Car : Safari
Price : ₹ 78 per day

How many days? 5

Order Completed

Your Vehicle Order Has Been Processed!

Total Rent : ₹ 390

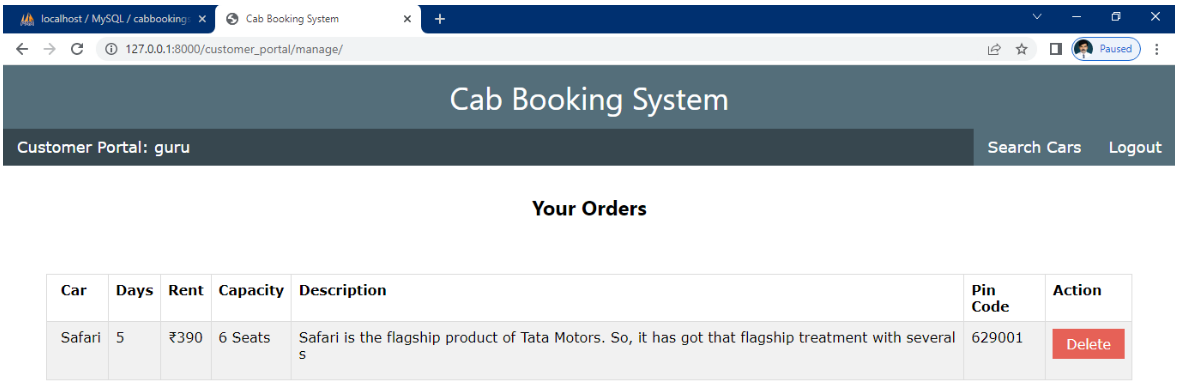
Vehicle : Safari

Duration : 5 days

Car-Dealer : Raji Selvi

Car-Dealer's Contact No : 9944940044

Figure B.14 Cab Booking



Your Orders

Car	Days	Rent	Capacity	Description	Pin Code	Action
Safari	5	₹390	6 Seats	Safari is the flagship product of Tata Motors. So, it has got that flagship treatment with several s	629001	Delete

Figure B.15 Booking Reports