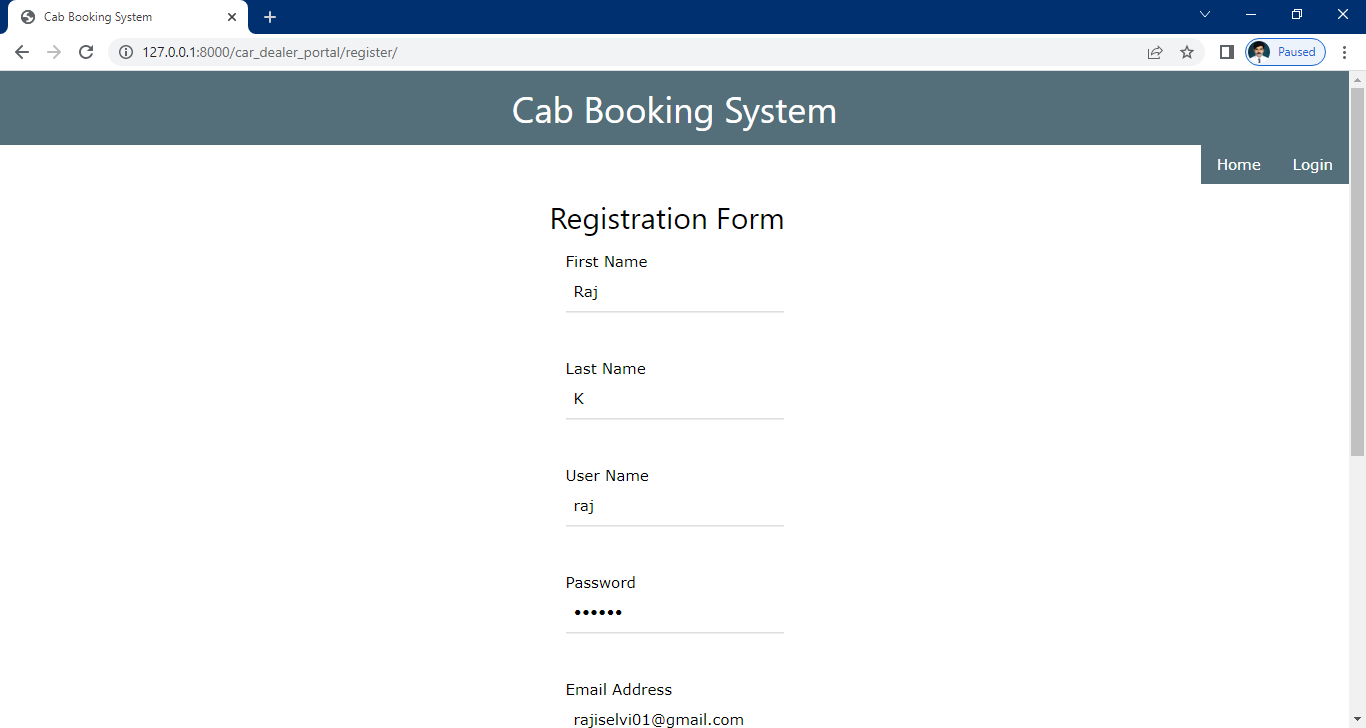
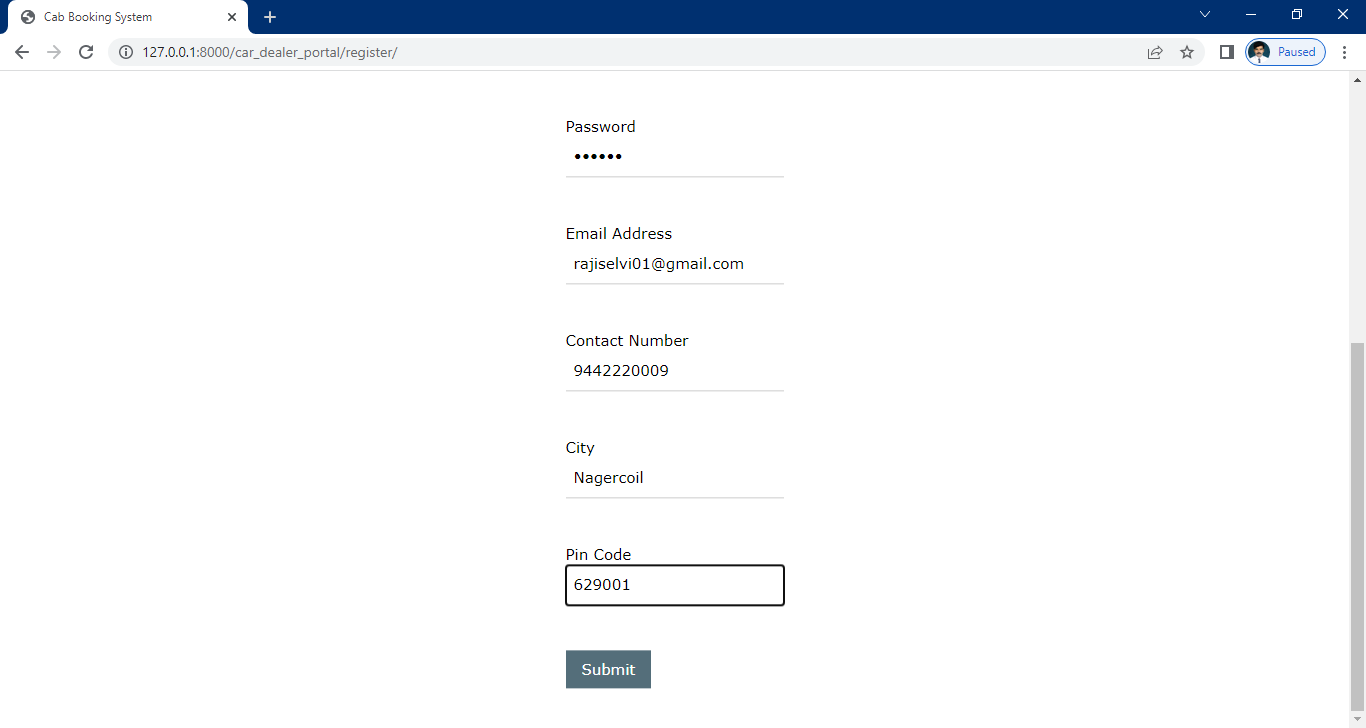
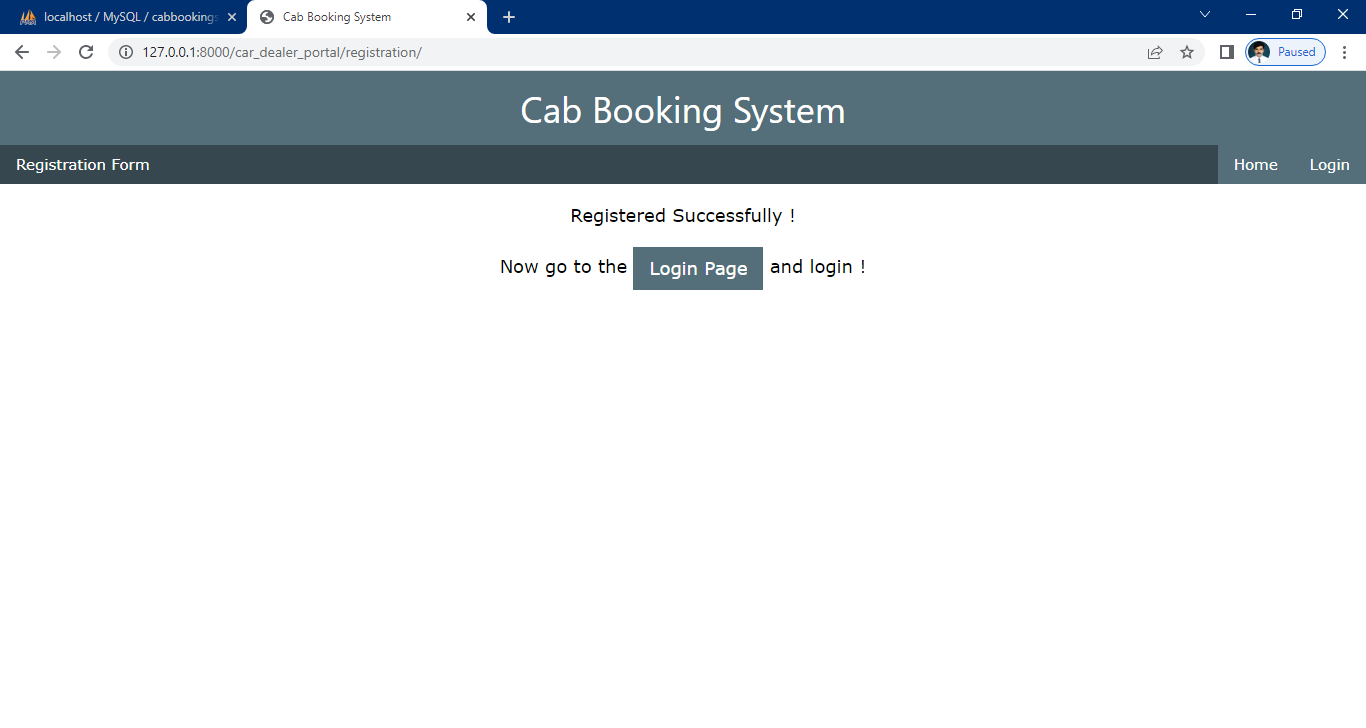


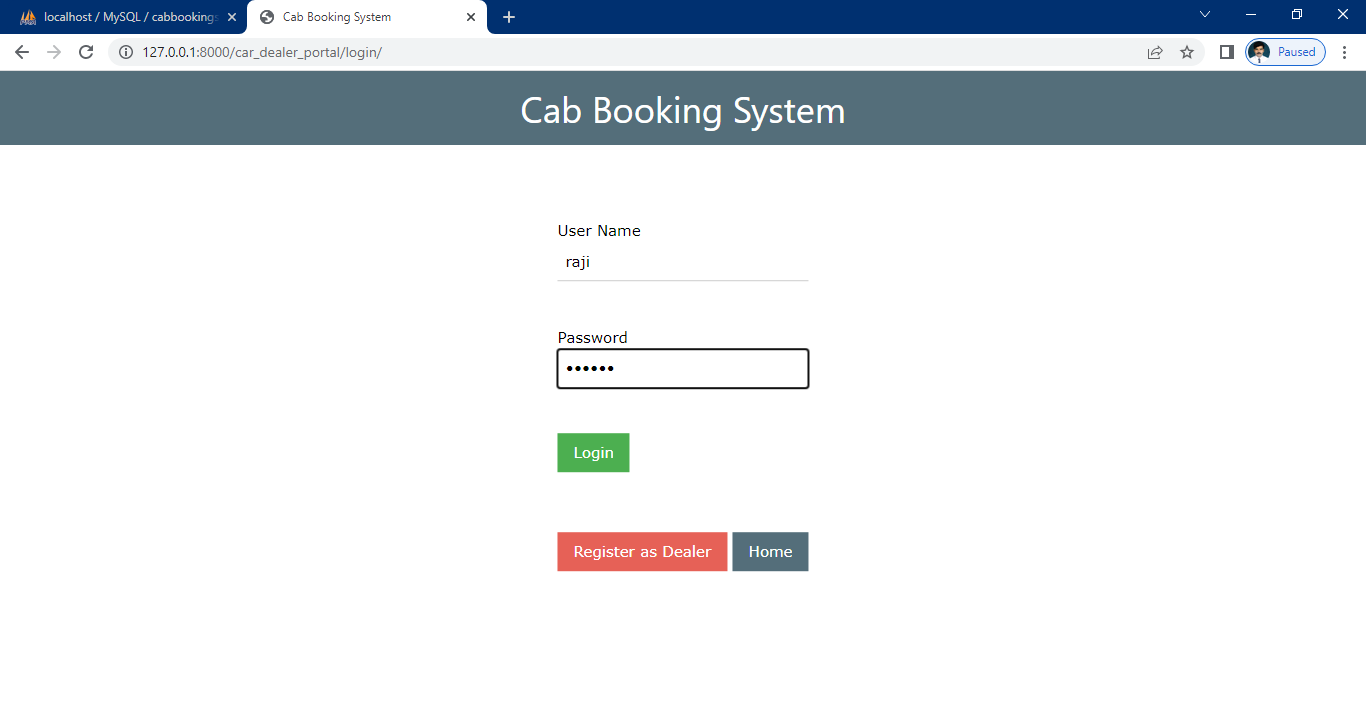
Dealer register



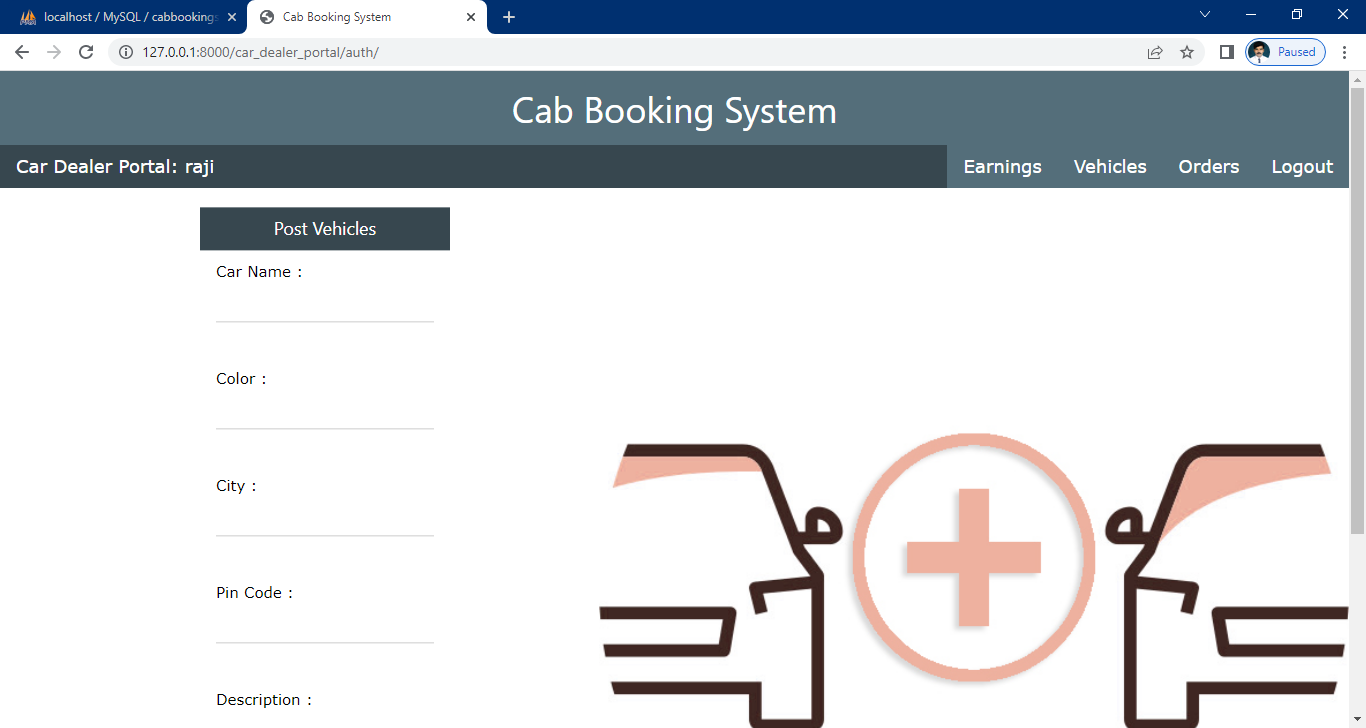




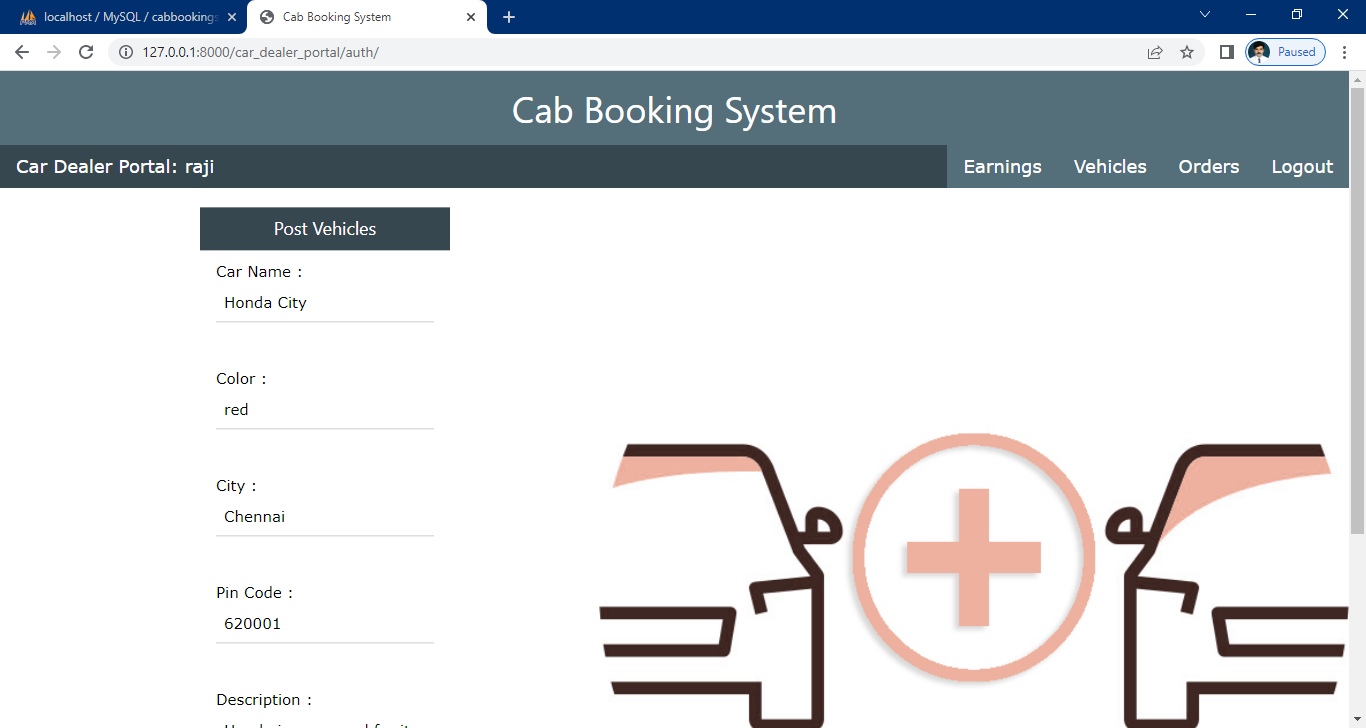
Dealer login

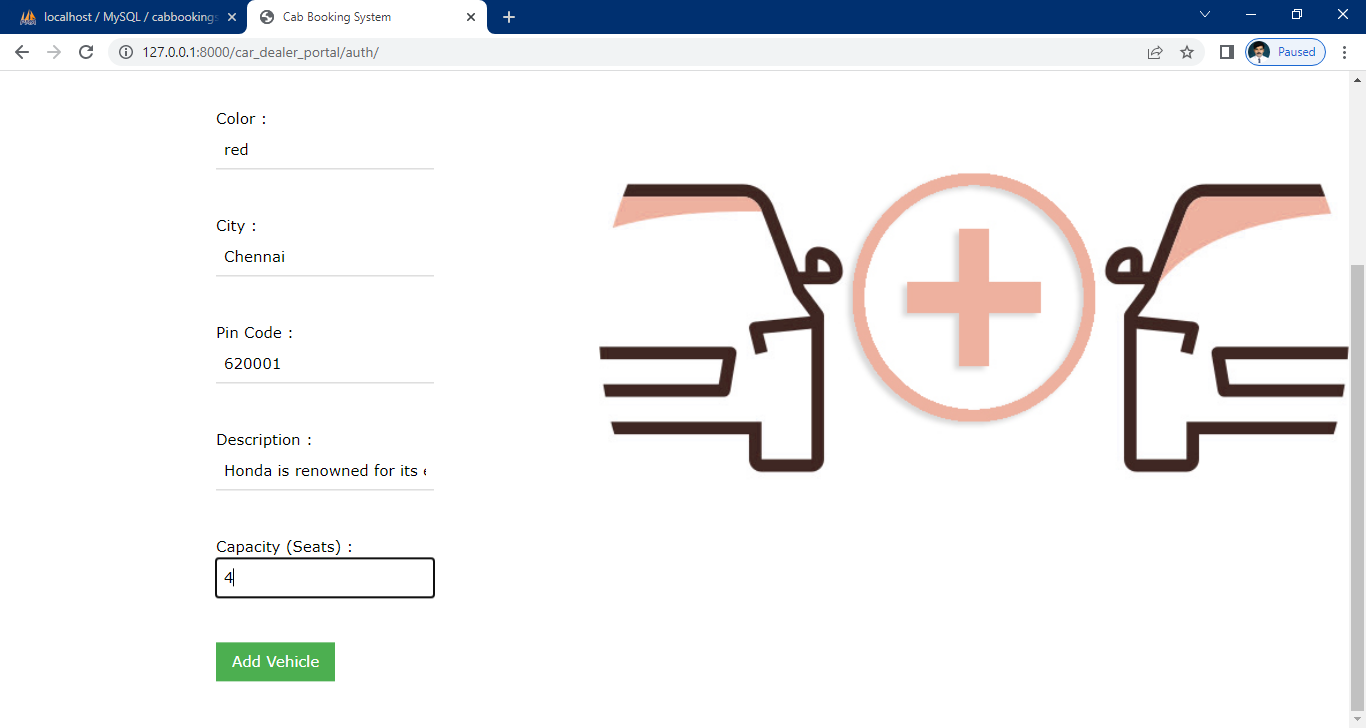


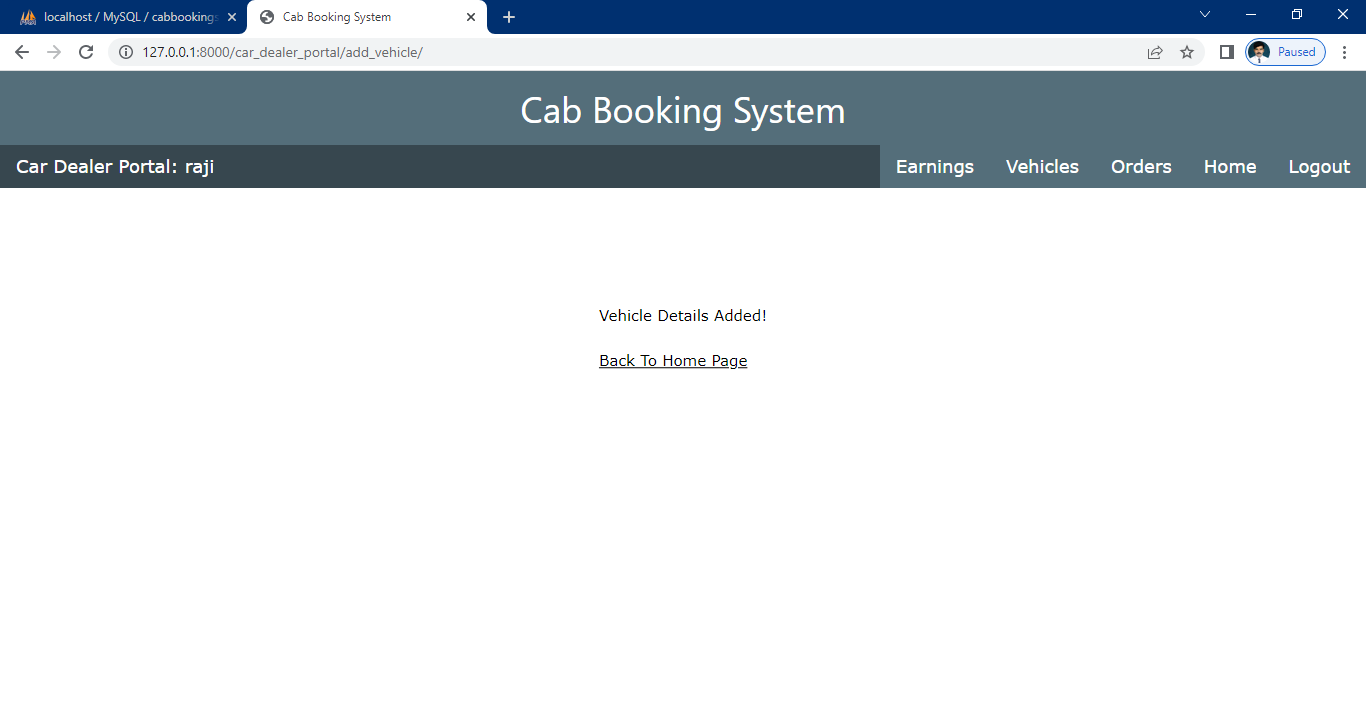
Dealer home



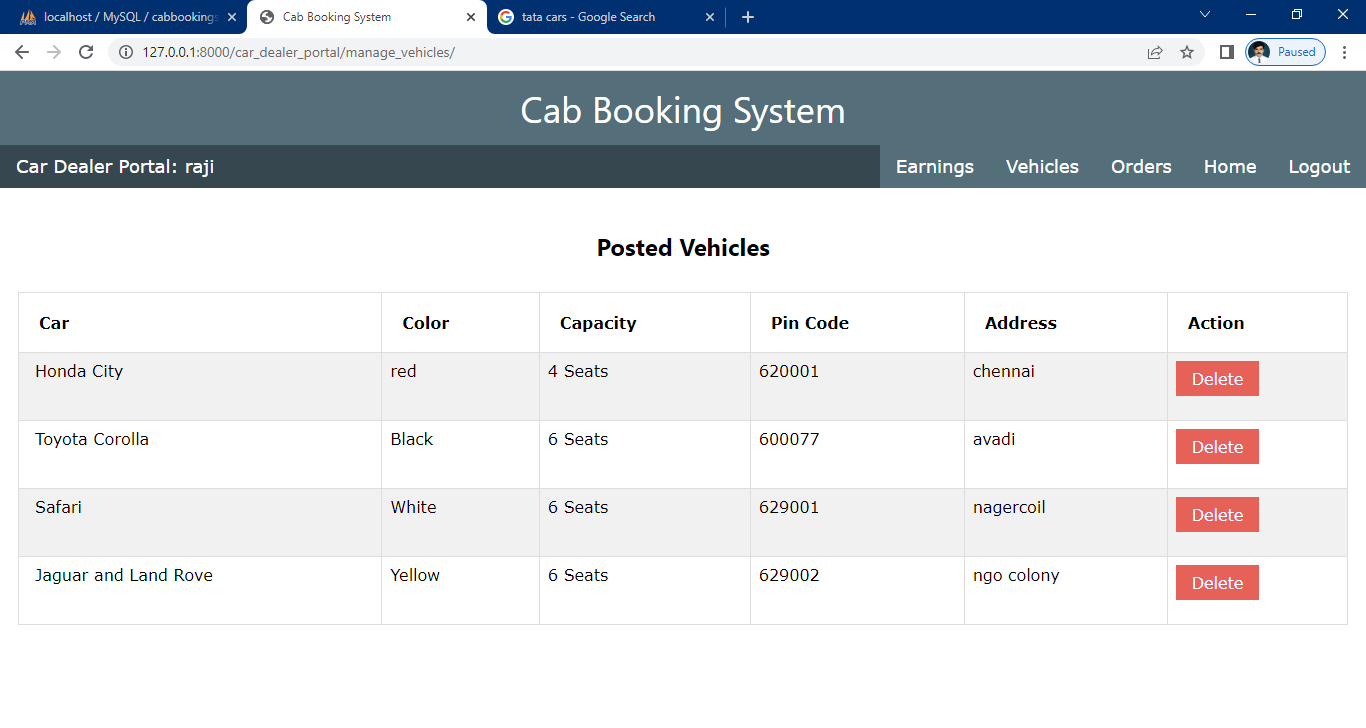
New car add



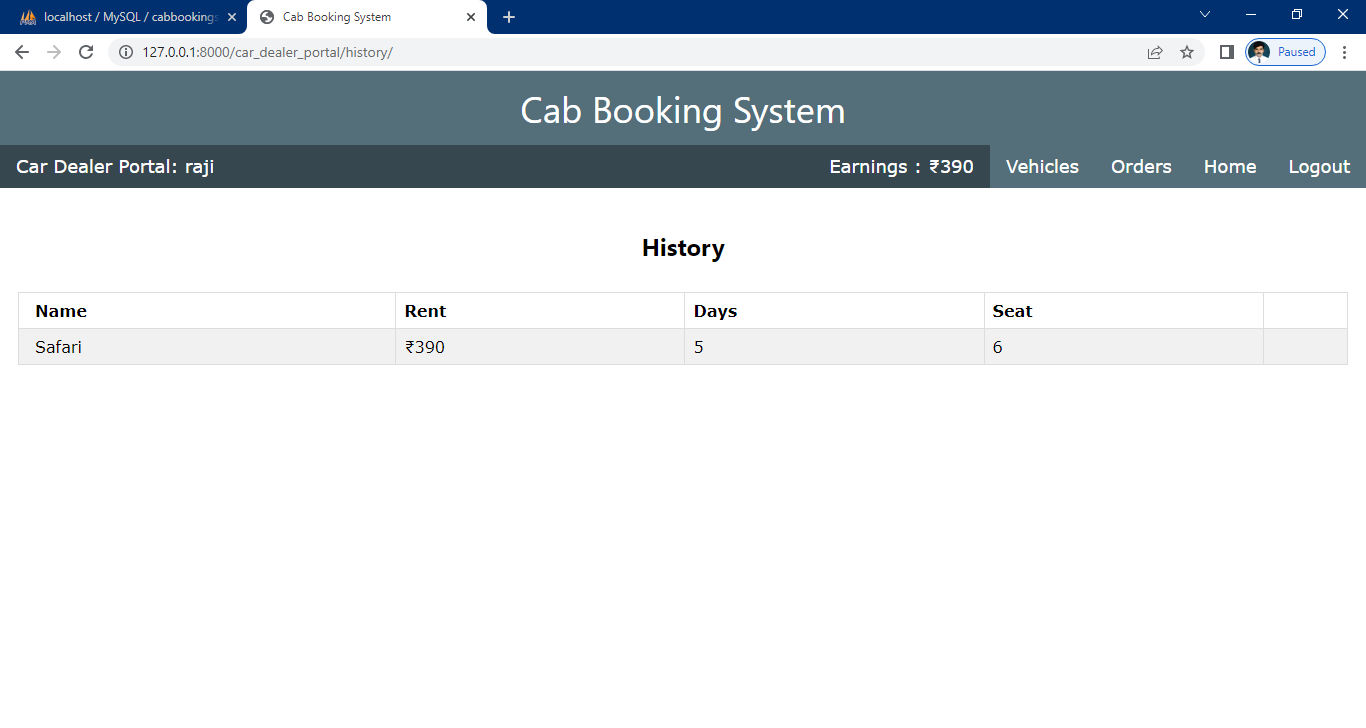




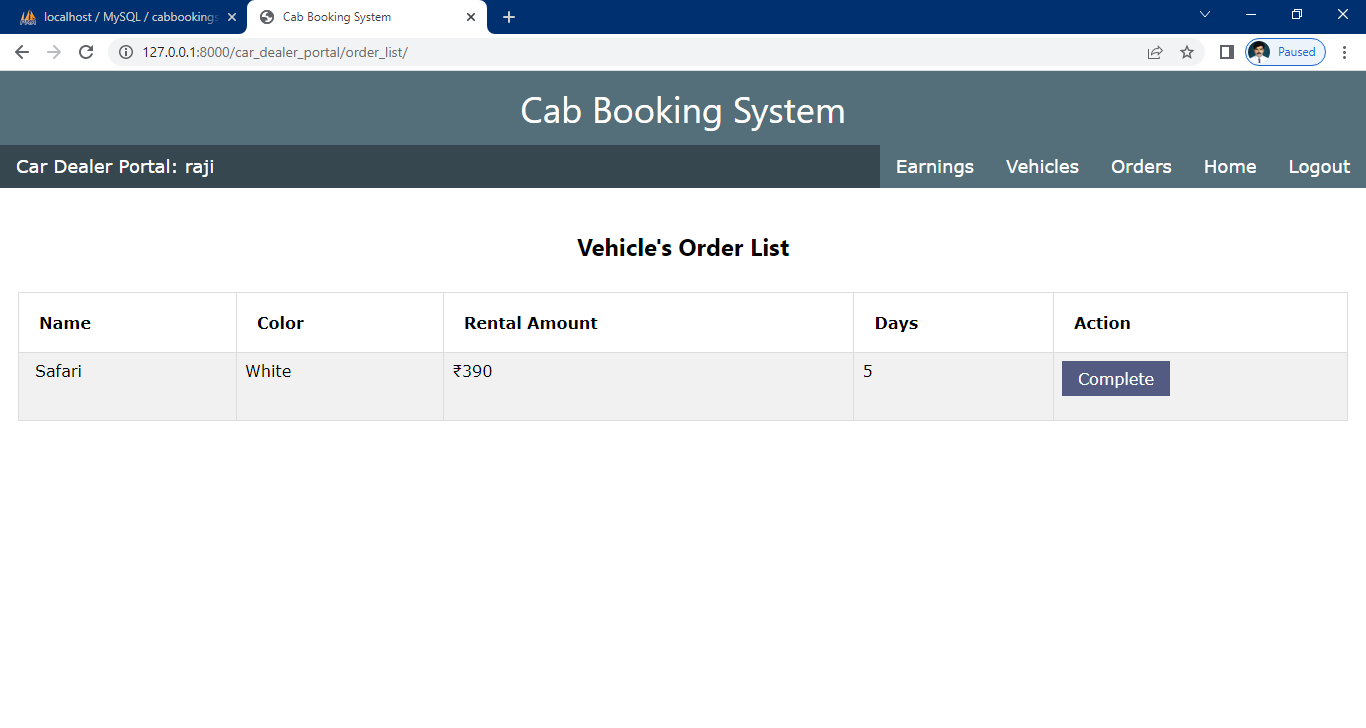
Display vehicles



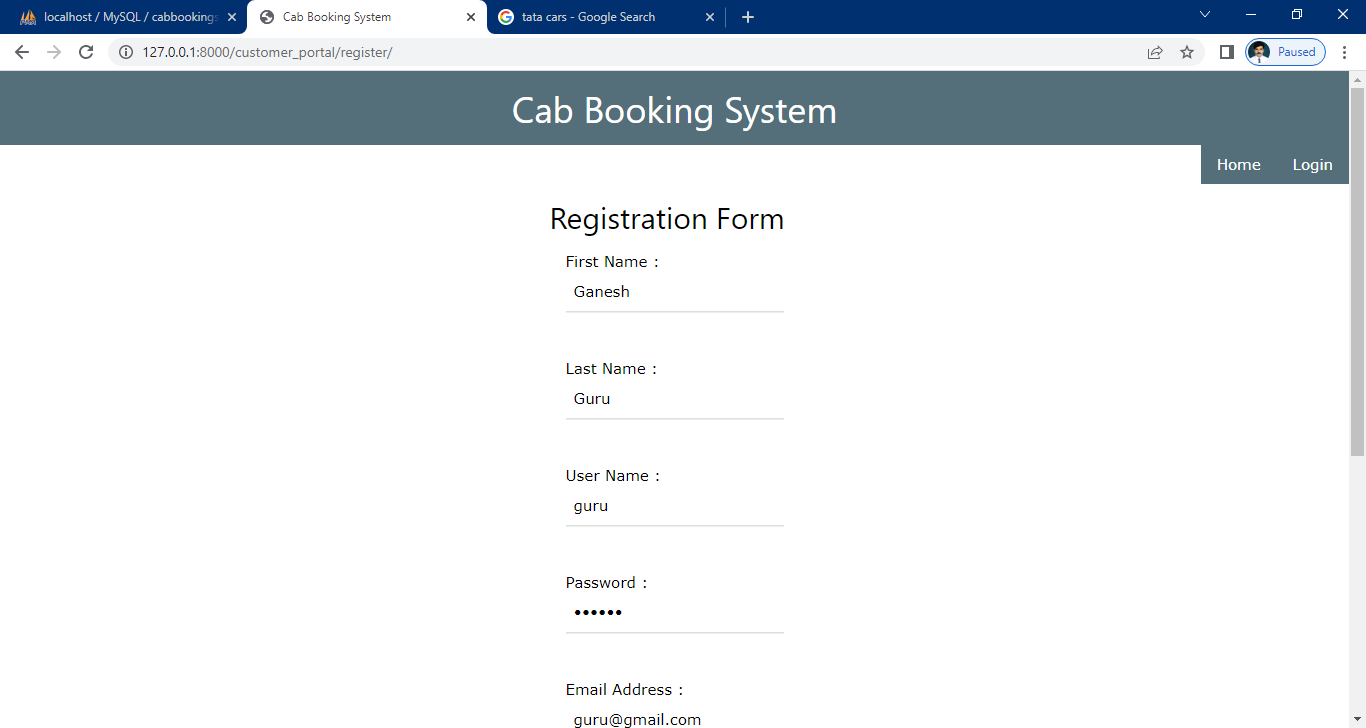
Earning income

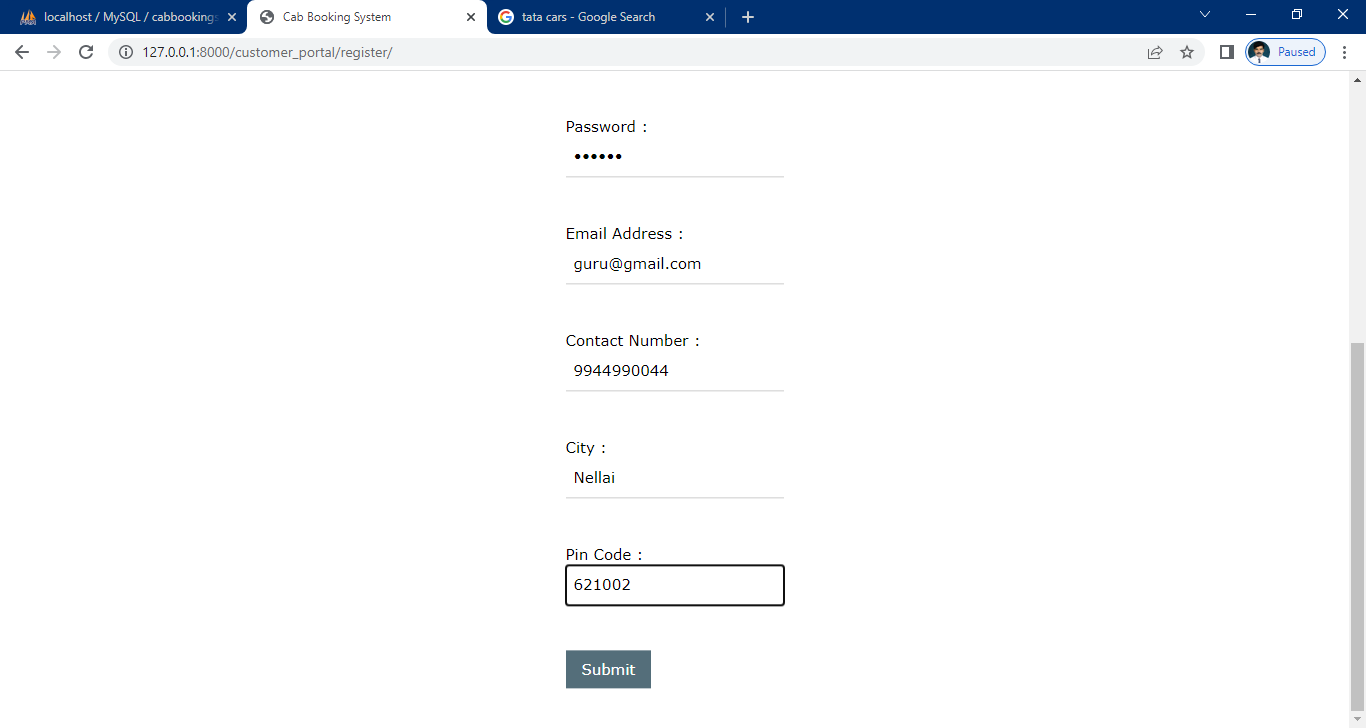


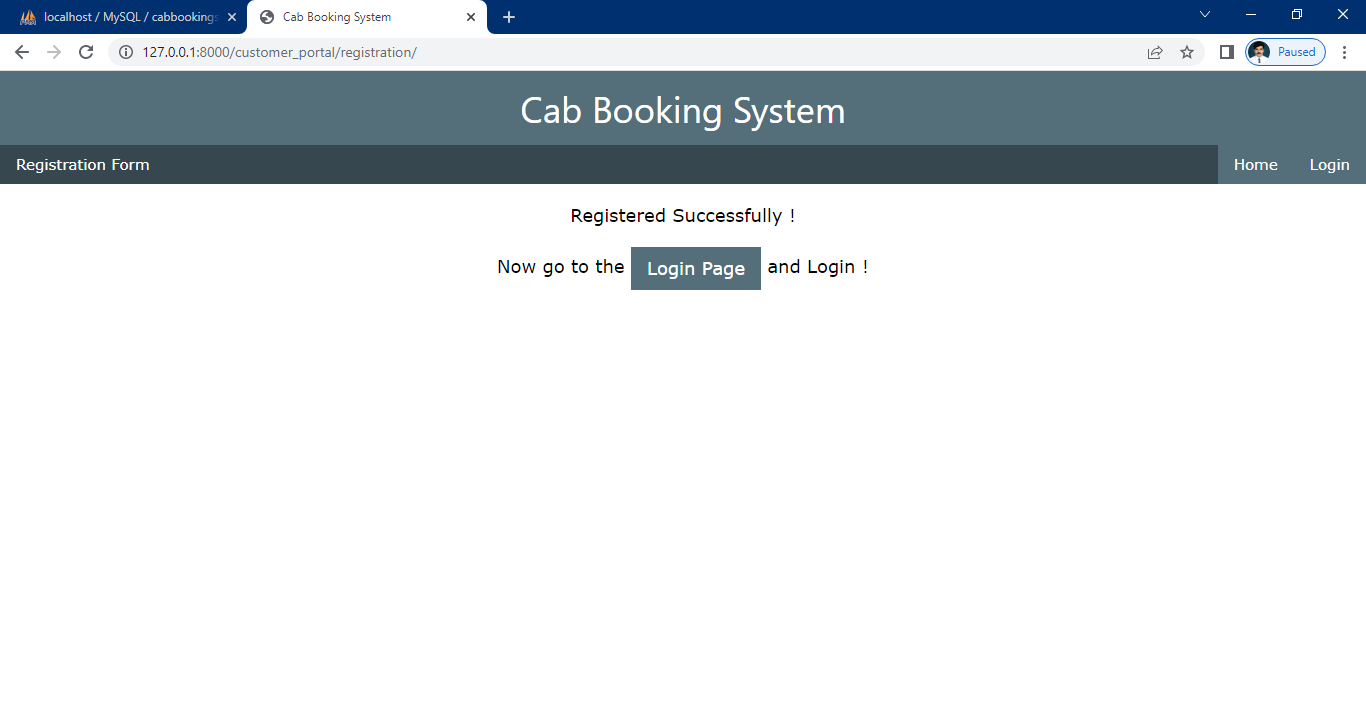
Cab booking history



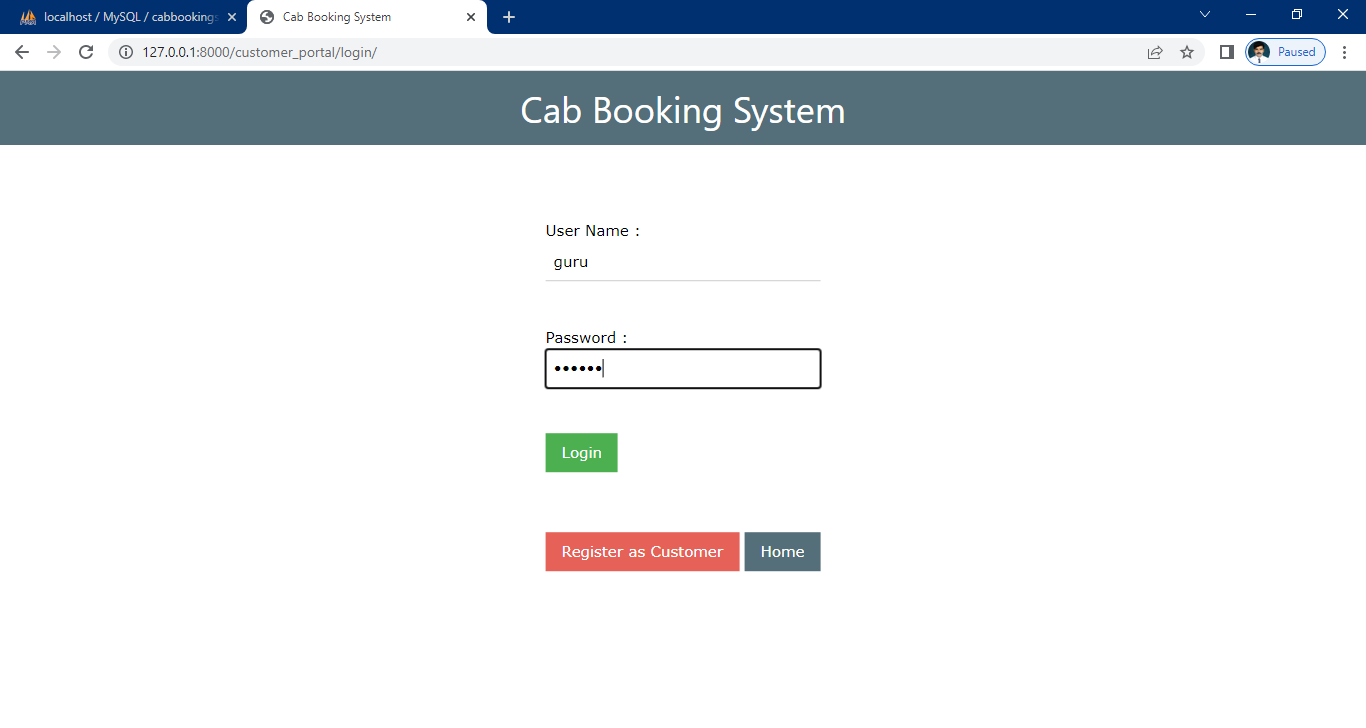
Customer register



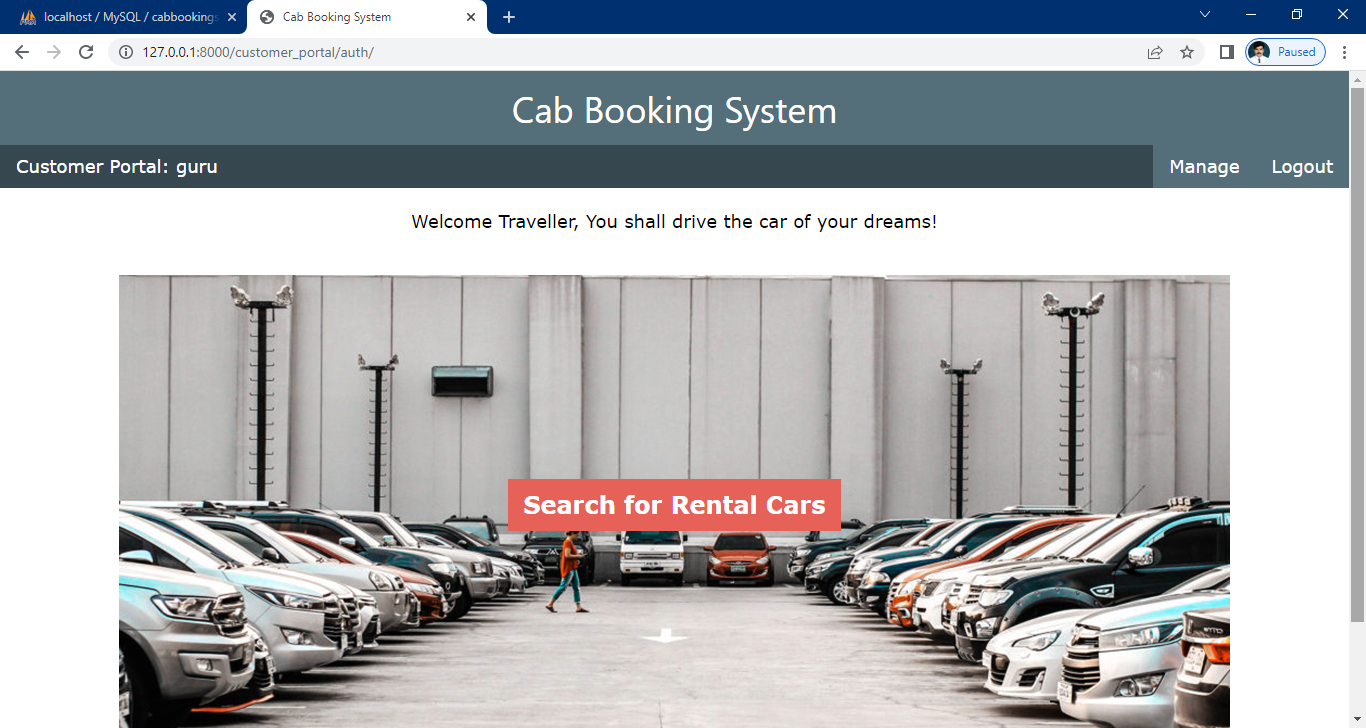




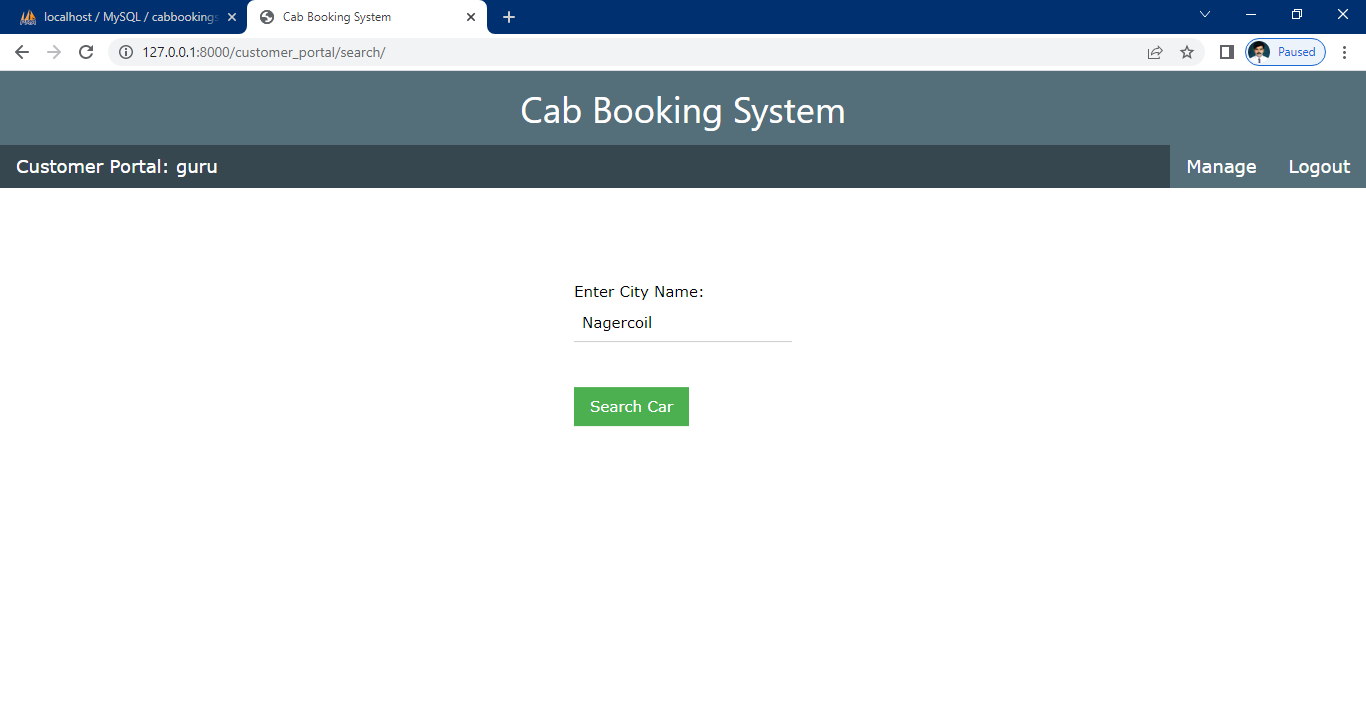
Customer login

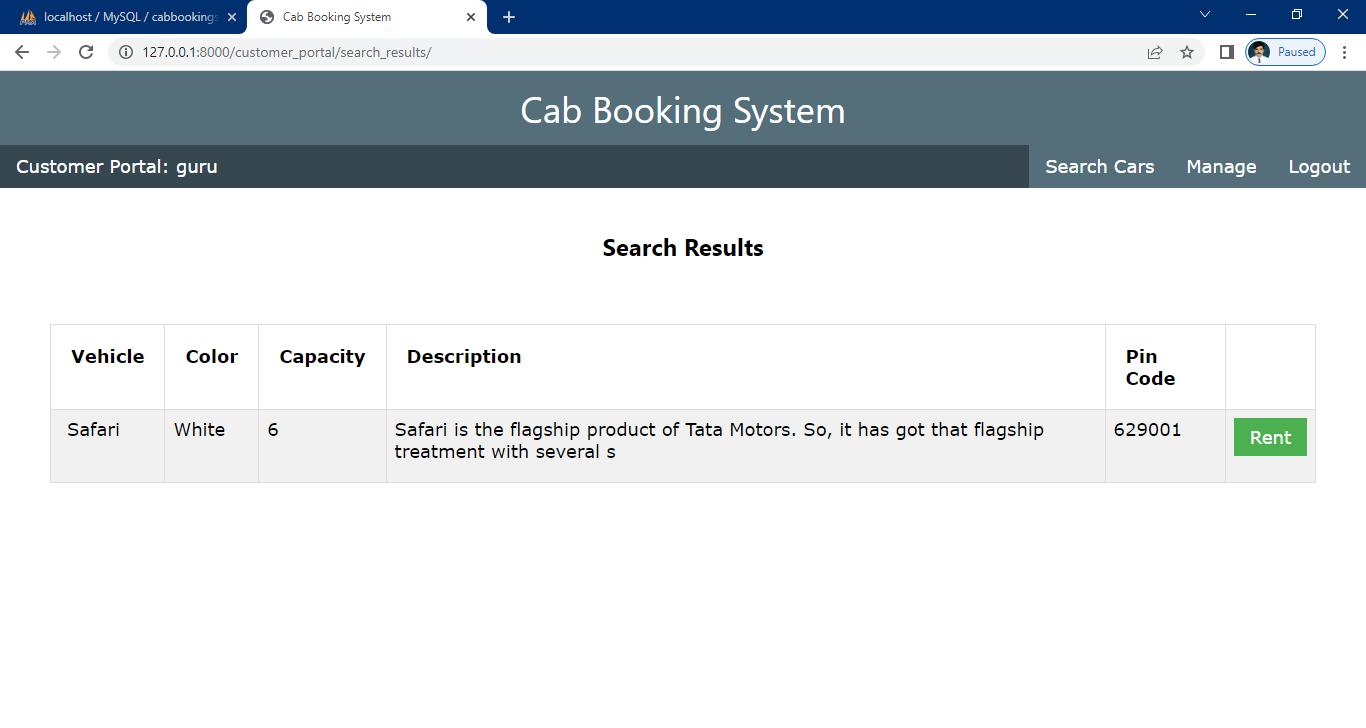


Customer home

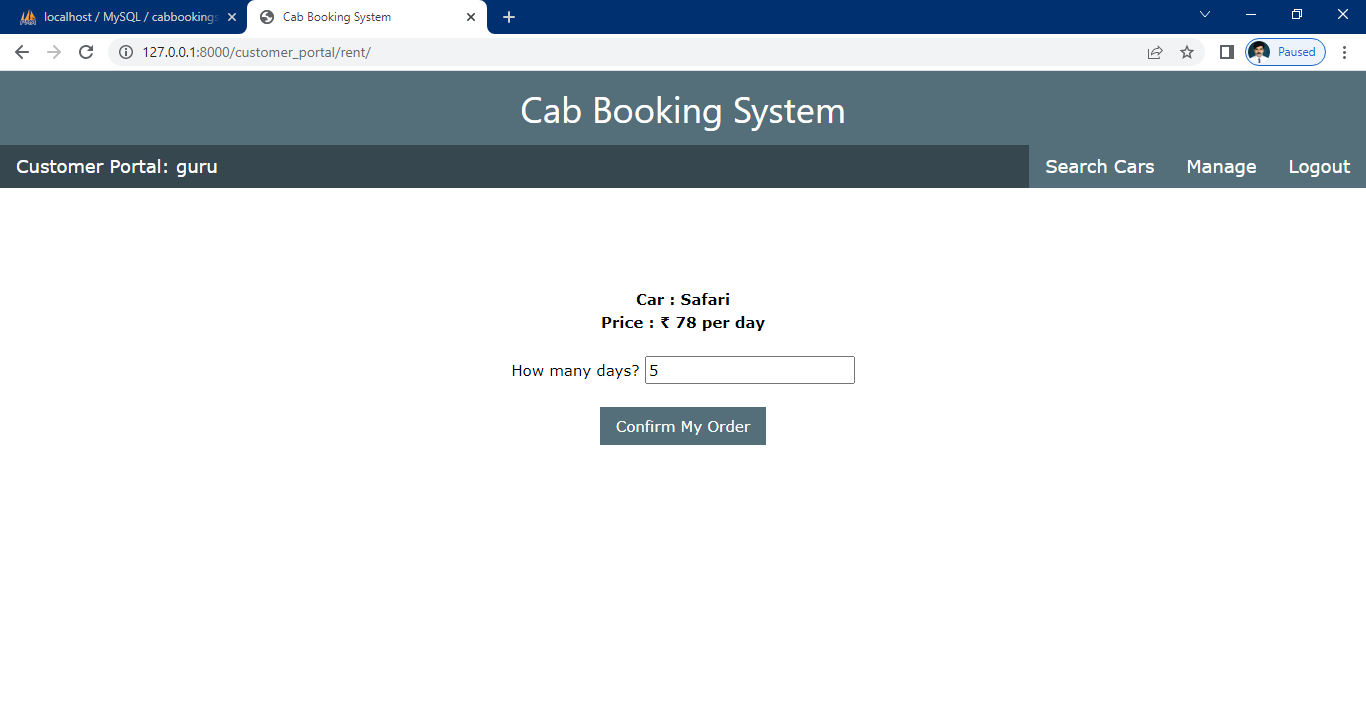


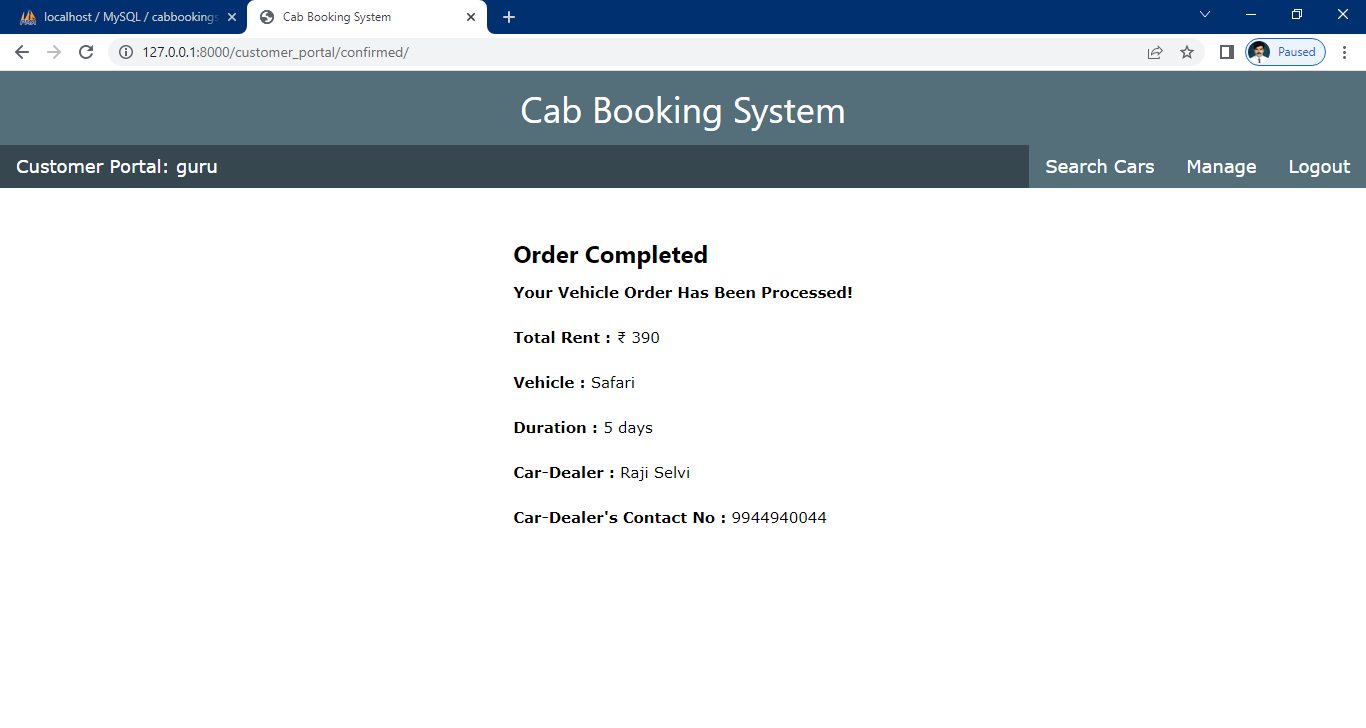
Search car



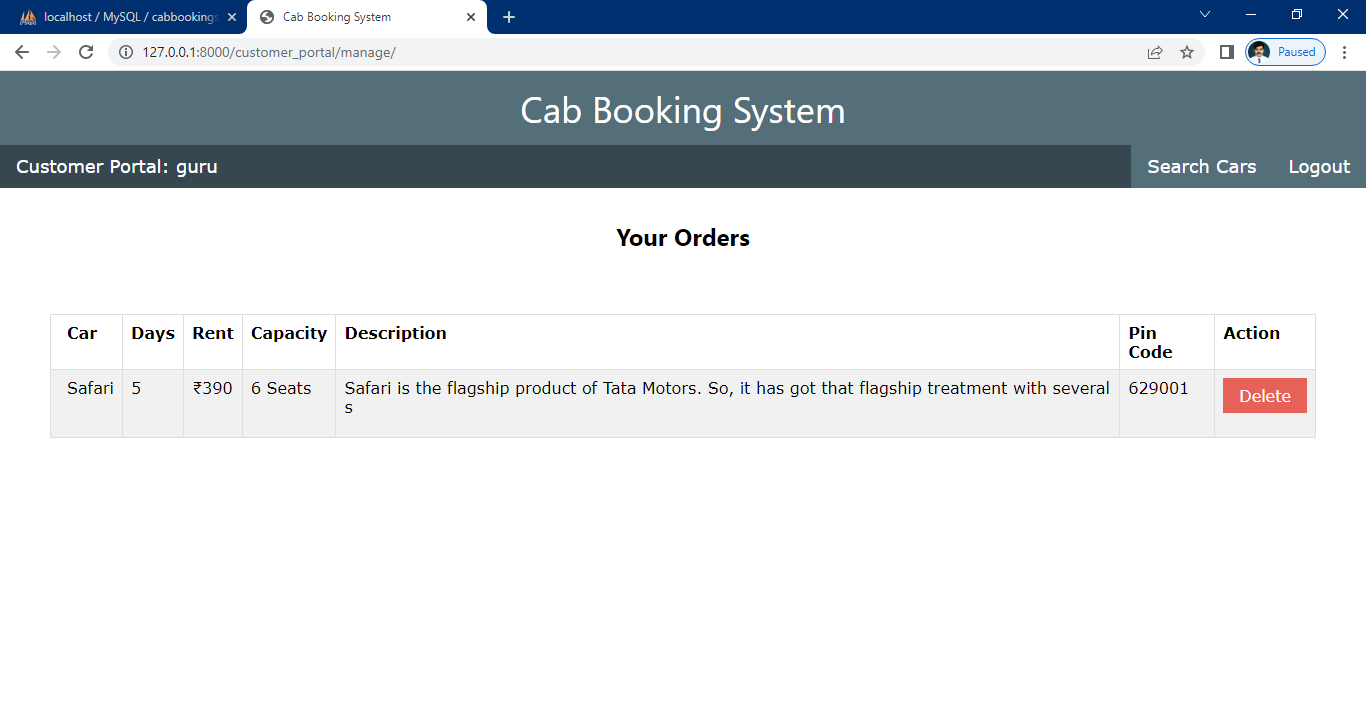


Cab booking





Booking reports



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)