# RENTIT - Online car rental platform

*Submitted in partial fulfillment of the*

*requirement for the award of the degree of*

Bachelor of Computer Science

**By**

T YOGESHWARAN(RA2031005020040)
S MANISHA (RA2031005020017)
T NITHISHKUMAR(RA2031005020019)

**Under the guidance of**

**DR.V.SARAVANAN**
**Prof.&Head CS**
**VP (Academic, S&H)**
**Department of Computer Science and applications.**

**DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS -B.Sc(CS)**
**FACULTY OF SCIENCE AND HUMANITIES**
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**
**Ramapuram, Chennai.**
**April 2023**

# SRM Institute of Science and Technology

**Ramapuram, Campus.**

**Faculty of Science and Humanities**

**Department of Computer Science and Applications (B.Sc-CS)**

**<u>BONAFIDE CERTIFICATE</u>**

Certified that this project report titled **“RentIt – An online car rental platform”** is the bonafide work of **T.Yogeshwaran(RA2031005020040),S.Manisha (RA2031005020017),T.Nithish Kumar (RA2031005020019)** who carried out the project work done under my supervision during the academic year 2022-2023Semester VI.

Course Code : USC20D10L

Course Name : Project Work

Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Signature of Internal Guide** **Signature of Head of the Department**

**Signature of External Examiner**

## ACKNOWLEDGEMENT

I extend my sincere gratitude to the Chancellor **Dr. T.R. PACHAMUTHU** and to Chairman **Dr. R. SHIVAKUMAR** of SRM Institute of Science and Technology, Ramapuram, Chennai for providing me the opportunity to pursue the BSc(CS) degree at this University.

I express my sincere gratitude to **Maj. Dr. M. VENKATRAMANAN,** Dean **(S&H)**, SRM IST, Ramapuram for his support and encouragement for the successful completion of the project.

I record my sincere thanks to **Dr. J. DHILIPAN, Head(MCA) & Vice Principal (Admin, S&H)**, SRM IST, Ramapuram for his continuous support and keen interest to make this project a successful one.

I record my sincere thanks to **Dr. V. SARAVANAN, Vice Principal (Academic, S&H) & Head(B.Sc(CS)**, SRM IST, Ramapuram for his encouragement and keen interest to make this project a successful one.

I find no word to express profound gratitude to my guide**. Dr. V. SARAVANAN, Vice Principal (Academic, S&H)** Assistant Professor, Department of Computer Science and Applications-BSc(CS), SRM IST, Ramapuram for his kind co-operation and encouragement which helped us for the completion of this project.

I thank the almighty who has made this possible. Finally, I thank my beloved family member and friend for their motivation, encouragement and cooperation in all aspect which led me to the completion of this project.

T YOGESHWARAN(RA2031005020040)
S MANISHA (RA2031005020017)
T NITHISH KUMAR (RA2031005020019)

# ABSTRACT

This report describes the project development of **RentIt - A car rental platform** that was developed with several frameworks, languages and algorithms. RentIt, is mainly build upon Python's Django framework and MySQL. It is then accessed by mysqlclient, a python extension to connect MySQL and Django. Django being a flexible and easy to use framework, helps to output the processed data from MySQL to the HTML template as the 'POST' data. It allows data to pass-in through the 'GET' function and feeds it to the database management system. The processed data is finally exported to a live server(local host) through a browser. The output template is a minimal and very userfriendly interface.

Most of the data processing happens on the server side, i.e.) Django. Django has 4 main components for it to run. They are url, view, model, template. The 'template' consists of all the front-end codes of the project. 'Model' consists of the structure of the project and is used to redirect the processing to the required path. 'View' consists of the main code for the project and is the base file. 'url' consists of all the subdirectories of the present project and is required to redirect the links.

When the manage.py file is executed through the terminal, it reroutes the path to settings.py in the main project folder. The settings.py has all the environment settings and requirements to run the project. The path is then redirected to the 'apps' we specified in the settings.py file. Apps are the subdirectory folders of the main project folder. Each app has an MTV(Model, Template, Views) format on its own.

**Keywords**:

RentIt; Rental platform; Django; MySQL; Python frameworks; Python extensions; MTV model;

# Table Of Contents

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

The self-drive online car rental car is categorized as one of the most sought-after modes of commute preferred by many online car rental customers. Self-drive online car rental services provide the customer with full control over the rental vehicle for the stipulated time period.

The prominence of the self-drive rental platform across the markets can be termed as the most impactful factor in the growth of the short-term online car rental service providers. Many platforms offer hourly, daily, or even monthly online car rental services to their customer. This service proves extremely useful for tourists looking to procure vehicles in tourist destinations for a limited time period.

This domain was severely affected by the outbreak of Covid-19. The thing is that car reservation systems are heavily dependent on tourism. Therefore, it comes as no surprise that the travel restrictions harmed this market.

Still, the latest industry reports show that the demand for car booking systems remains pretty high. It is predicted that the car rental market will reach $131 billion by 2026. The cost-effectiveness of the car rental reservation process serves as a key reason for such growth.

The rise of new technologies boosted the market significantly. Their active adoption enables service providers to offer customers better user experience. These enhancements include optimized customer information management and convenient online car booking systems.

With the advent of the Internet, customers started ordering products and services online. That is how online booking systems have gained momentum. The same goes for rental services. If users want to rent a car, they visit car rental websites and book a vehicle there.

North America is home to many prominent online car rental platform developers and also online car rental service providers that are making strategic developments to lure in customers and cement their revenue share in the market. By adding rental services for goods carrier vehicles in their portfolio the companies are striving to widen the scope of their existing customer base in the market.Also, with the rising prices of fossil fuel, we might see a boom in the car rental industry soon enough.

Car rental service combined with a car-pooling service is a definite win for a car rental organization at this time. Sadly, India being one of the top consumers of fossil fuels haven't adapted this trend of car rental services. The two main grounds for this being lack of car rental startups and lack of people knowing car rental platform's advantages.

# CHAPTER 2

# WORKING ENVIRONMENT

## 2.1 DEVELOPMENT ENVIRONMENT SOFTWARE

Operating system: Windows 10

Windows 10 is selected as the developing operating system because Windows has the biggest selection of software available for its platform than any other operating system. The benefit of this is that users get to choose from wider variety of options. This creates healthy "competition" for users, where software developers really have to push boundaries to produce the best program possible. Anything less than the best will result in user's picking the next program on the list. This alone does wonders in motivating software developers to deliver excellent solutions that meet users' needs.

**Software used:**
**DJANGO**

Fig : 2.4.1 Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

With Django, you can take web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. The Django project's stability, performance and community have grown tremendously over the past decade since the framework's creation. Detailed tutorials and good practices are readily available on the web and in books. The framework continues to add significant new functionality such as database migrations with each release.

**MYSQL**

**Fig : 2.4.2 MySQL**

MySQL is very fast reliable and flexible Database Management System. MySQL is open source, i.e., anyone can use it for free. And, anyone can modify the code. It supports all the major platforms like Windows, Linux, Solaris, macOS, and FreeBSD. MySQL is well-known for its ease of use, but definitely an interface is needed. The open nature of MySQL had spawned quite a number of third party front-ends in addition to their own official one.

On the other hand, MySQL developer community is very active and that’s why MySQL gets frequent software updates. MySQL has a client-server architecture and can be use in any networked environment. Every client can make request to server using some network, which is necessary for this project.

## 2.2 HARDWARE REQUIREMENT

### 2.2.1 Processor

Intel core i5 8th Gen Processor provide better processing capabilities and better cooling technology to our CPU. With an Intel processor, we can run our laptop for long time without need to switch off. Besides that, intel processor can help us to boost up the CPU processing power. By using this, we can keep developing the Library Management System without need to worry that the laptop cannot support.

### 2.2.2 RAM : 8 GB

In order to support SQL Server, we use 8Gb Ram to avoid any problem occurred during development phase. Besides that, SQL Server can process faster when running SQL statement with 8Gb ram. It can save a lot of time if total up the process time.

**Operation Environment**

The table shown below is the minimum requirement:

**Table 2.1 Table for operation environment**

| | |
|---|---|
| Processor | Intel Pentium 233Ghz or better performance |
| Operating System | Window 7 and above |
| Memory | 4GB RAM |
| Screen Resolution | Monitor with screen resolution minimum 1024 x 768 |
| Hard disk Space | Minimum 5GB to include database usage for future |

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

The first model of this car rental platform is to define the login authentication model. When the portal is accessed through customer/dealer side, each has a authentication model for it to pass through. If not authenticated, the page is redirected again to the login screen and when authenticated, it is redirected to the home page of accessed portal.

This home page is defined as the ‘auth_view’. Then the registration module is defined. This is activated when the url is redirected to the registration page. The required inputs for registration is received and stored in the database through ‘request’ and ‘post’ functions.

A sub-class is created for ‘auth_view’, called ‘@login_required’. When triggered, only authenticated users are allowed to access the website’s contents.

**Customer Portal :**

The first function for the customer portal is ‘search_results’. This takes the city name as input and passes it thorough the database to find all cars with the city id matching it. It also has a variable called ‘is_available’, for knowing the availability of the car. After all this process the results are posted. The second function is ‘rent_vehicle’. Here is where the cost per day of the cars is calculated. It takes the seat capacity as input and multiplies it by 580 to give out the per day rent.

Then comes the ‘confirm’ function. This saves all the information about the rental transactions and bookings in the database. This function is run only when ‘is_available’ is positive. ‘Manage’ function lets the customers preview their rental cars. This lists all the current rented cars list with the number of days, rent and models.

‘update_order’ gives access to the customers to edit their orders. The order id is the primary key for this function. ‘delete_order’ destroys the whole order with its id and all the data in it from the database and removes it from the dealer’s portal too.

**Dealer Portal :**

The most important function of the dealer's portal is the 'add_vehicle' function. This lets the dealers add multiple cars to the rental platform. This takes inputs like car name, color, city, pin code and capacity and posts it to the database system. All these are given a special and unique car id. 'manage_vehicles' function helps the dealer to manage all the vehicles they posted. This gives them the option to remove, delete or modify the car details they entered previously.

'order_list' from the dealer's end contains all the orders that were placed by customers from the customer portal. This function has a sub-function called 'complete' to make the orders valid and resets the ' is_available' of cars back to positive.

The dealer's portal also has a feature function called 'history' that recalls all the orders in a detailed manner with the dealer's wallet history. This also has a primary key order id. The 'manage_vehicles' function also has a sub-function called 'delete'. This allows the dealer to remove the car from the database, but won't affect the orders that are related to that car previously.

These all functions work together simultaneously one after the other with reference to the url and portal requested. All of them are linked with each other and make the whole eco-system work flawlessly.

## 3.2 WORKING OF THE MODEL

The better place to start is the project folder. This consists of all the apps we created as a separate folder, the main project folder, a static folder and a manage.py file.

| Name | Date modified | Type | Size |
|---|---|---|---|
| car_dealer_portal | 07-04-2022 06:17 PM | File folder | |
| customer_portal | 07-04-2022 06:17 PM | File folder | |
| env | 10-04-2022 11:15 PM | File folder | |
| home | 07-04-2022 06:17 PM | File folder | |
| ocrs | 07-04-2022 06:17 PM | File folder | |
| static | 07-04-2022 05:09 PM | File folder | |
| @ manage.py | 18-01-202011:46 PM | Python Source File | 1 KB |
| Hi requirements.txt | 07-04-2022 06:16 PM | Text Document | 1 KB |
| 3 run project.txt | 11-04-2022 10:06 PM | Text Document | 1 KB |

**Fig : 3.2.1: Project folder**

In the above figure, ‘car_dealer_portal’, ’customer_portal’ and ’home’ are the app folders that were created after the main project folder. These are addressed in the settings.py file for linking them through url.

Here ‘ocrs folder is the main project folder that contains the settings.py file which is the most important file in Django projects. ocrs - online car rental platform.

The ‘env’ folder is the folder that contains all the software requirements for the project. Django, mysqlclient source files are all extracted here and is accessed by python.

The static folder is also one of the important folders in the project. This contains all the images and scripting files for the project including css, js etc. This folder can be accessed by {{static ‘url’}} command anywhere in the python code.

Running manage.py file is the first step of activating the project. This contains the directory of the settings.py file and the import requirements for the project.

requirements.txt file has the third party libraries that needs to be imported to run the project. This project has Django and mysqlclient in the requirements file. This file is accessed by the python terminal and pip install the requirements.

## App folders :

| Name | Date modified | Type | Size |
|---|---|---|---|
| _pycache_ | 10-04-2022 11:23 PM | File folder | |
| migrations | 07-04-2022 05:09 PM | File folder | |
| templates | 07-04-2022 05:09 PM | File folder | |
| H_init_.py | 18-01-2020 11:46 PM | Python Source File | 0 KB |
| @ admin.py | 18-01-2020 11:46 PM | Python Source File | 1 KB |
| ® apps.py | 18-01-2020 11:46 PM | Python Source File | 1 KB |
| @ models.py | 18-01-2020 11:46 PM | Python Source File | 1 KB |
| @ tests, py | 18-01-2020 11:46 PM | Python Source File | 1 KB |
| @ urls.py | 18-01-2020 11:46 PM | Python Source File | 1 KB |
| @ views.py | 10-04-2022 11:23 PM | Python Source File | 6 KB |

**Fig : 3.2.2: App folder**

Each app folder contains all he above elements in it named ‘_init_’, ‘admin’, ‘apps’, ‘models’, ‘tests’, ‘url’ and ‘views.

The ‘template’ folder consists of all the front-end codes of the project i.e.) the HTML files of the project. ‘Model’ consists of the structure of the project and is used to redirect the processing to the required path. ‘View’ consists of the main codes for the project and is the base file. It has several functions in it that are called during the project. ‘url’ consists of all the sub-directories of the present project and is required to redirect the links. ‘apps’ consists of the directory names of all the user created app.

| Name | Date modified | Type | Size | |
|---|---|---|---|---|
| Q base.html | 08-04-2022 05:06 PM | Brave | HTML Docu... | 1 KB |
| © confirmation.html | 08-04-2022 04:44 PM | Brave | HTML Docu... | 1 KB |
| © confirmed.html | 08-04-2022 04:44 PM | Brave | HTML Docu... | 1 KB |
| © home_page.html | 08-04-2022 05:24 PM | Brave | HTML Docu... | 1 KB |
| & login.html | 08-04-2022 05:07 PM | Brave | HTML Docu... | 1 KB |
| & login_failed.html | 09-06-2021 10:59 PM | Brave | HTML Docu... | 1 KB |
| © manage.html | 08-04-2022 04:45 PM | Brave | HTML Docu... | 2 KB |
| Q order_failed.html | 08-04-2022 04:21 PM | Brave | HTML Docu... | 1 KB |
| Q register.html | 08-04-2022 04:25 PM | Brave | HTML Docu... | 2 KB |
| ^ registered.html | 08-04-2022 04:29 PM | Brave | HTML Docu... | 1 KB |
| © registration_error.html | 18-01-2020 11:46 PM | Brave | HTML Docu... | 1 KB |
| © search.html | 08-04-2022 04:23 PM | Brave | HTML Docu... | 1 KB |
| © search_results.html | 08-04-2022 04:23 PM | Brave | HTML Docu... | 2 KB |

**Fig : 3.2.3: Templates folder of customer portal**

| | | | |
|---|---|---|---|
| •i* views | ,py9₊ X - | *i* urls. | py 9+ X |
| C: > Users > raksh > Downloads > Online Car Rental System > customer_portal > * views.py > © auth_view | | C: > Users > raksh > Downloads > Online Car Rental System > customer_portal > urls.py > ... | |
| 11 | | | from djangoajrls import path,include |
| 12 | def index(request): 1 | 2 | from customer import * |
| 13 | if not request.user.isauthenticated: | 3 | from djangOj^confjjjrl^ import url |
| 14 | return render{request, 'customer/login.html') | 4 | urlpattems = [ |
| 15 | else: | 5 | url(r'^index/$‘,index), |
| 16 | return render{request, ’customer/home_page.html') | 6 | url(r'Alogin/$'.login), |
| 17 | | 7 | u r 1 (r'A a ut h / $', auth^j/iew), |

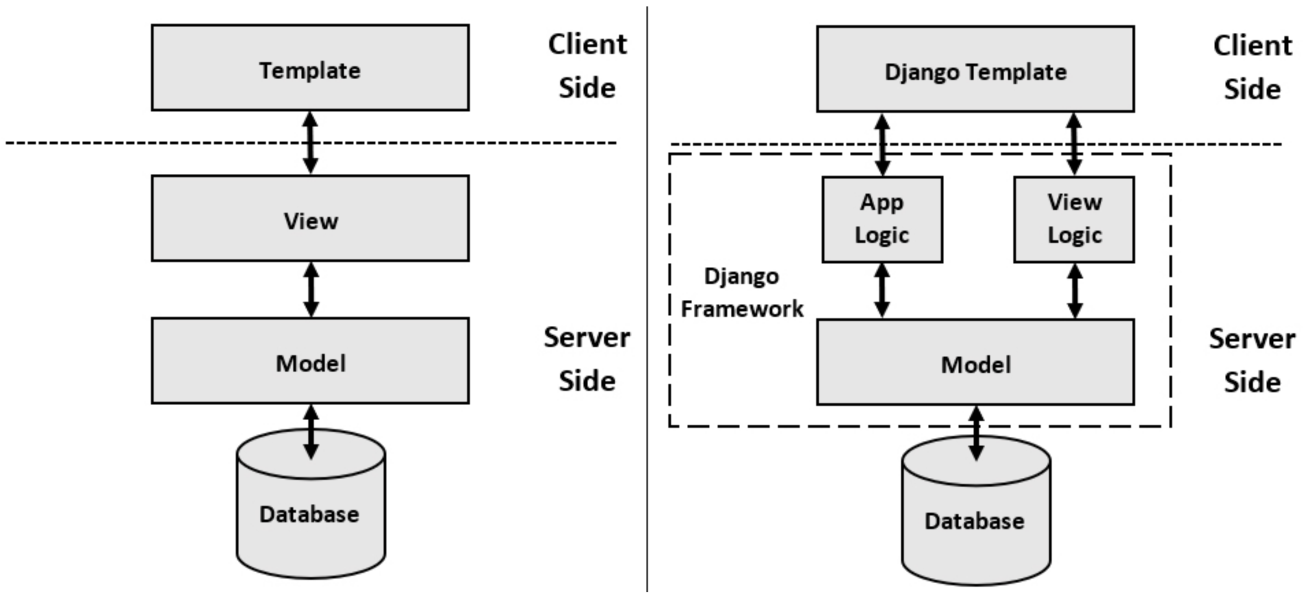
| | | | |
|---|---|---|---|
| 18 | def login(request): | 8 | url(r"Alogout/$',logout^view), |
| 19 | return render(request, 'customer/login.html') | 9 | url(r"Aregister/$",register), |
| 20 | | 10 | url(r'Aregistration/$ ’ ,r^istration), |
| 21 | def authview(request): | 11 | url(r'Asearch/$',search), |
| 22 | if request.user.isauthenticated: | 12 | url(r'Asearch_results/$',search results), |
| 23 | return render(request, 'customer/homepage.html') | 13 | u r 1 (r'A ren t / $', rent^j/ehicle ) , |
| 24 | else: | 14 | url(r'Aconfirmed/',confirm), |
| 25 | username = request,POST['username’] | 15 | ur 1 (n' '■manage/ ' .manage), |
| 26 | password = request.POST[1 password’] ■ | 16 | url(r'Aupdate/',updatejorder), |
| 27 | user = authenticate(request, username=username, password=password) | 17 | url(r'Adelete/',delete order), |
| 28 | try: | 18 | |
| 29 | customer = Customer.objects.get(user = user) | 19 | 1 |
| 30 | except: | | |
| 31 | customer = None | | |
| 32 | if customer is not None: 1 | | |
| 33 | auth.login(request, user) | | |
| 34 | return render(request, "customer/home_page.html') | | |
| 35 | else: | | |
| 36 | return render(request, 'customer/loginfailed.html’) | | |
| 38 | def logoutview(request): | | |
| 39 | auth.logout(request) | | |
| 40 | return render(request, 'customer/login.html') | | |
| 42 | def register(request): | | |
| 43 | return render(request, 'customer/register.html') | | |
| 45 | def registration(request): | | |
| 46 | username = request.POST[’username'] | | |
| 47 | password = request.POSTf’password"1 | | |

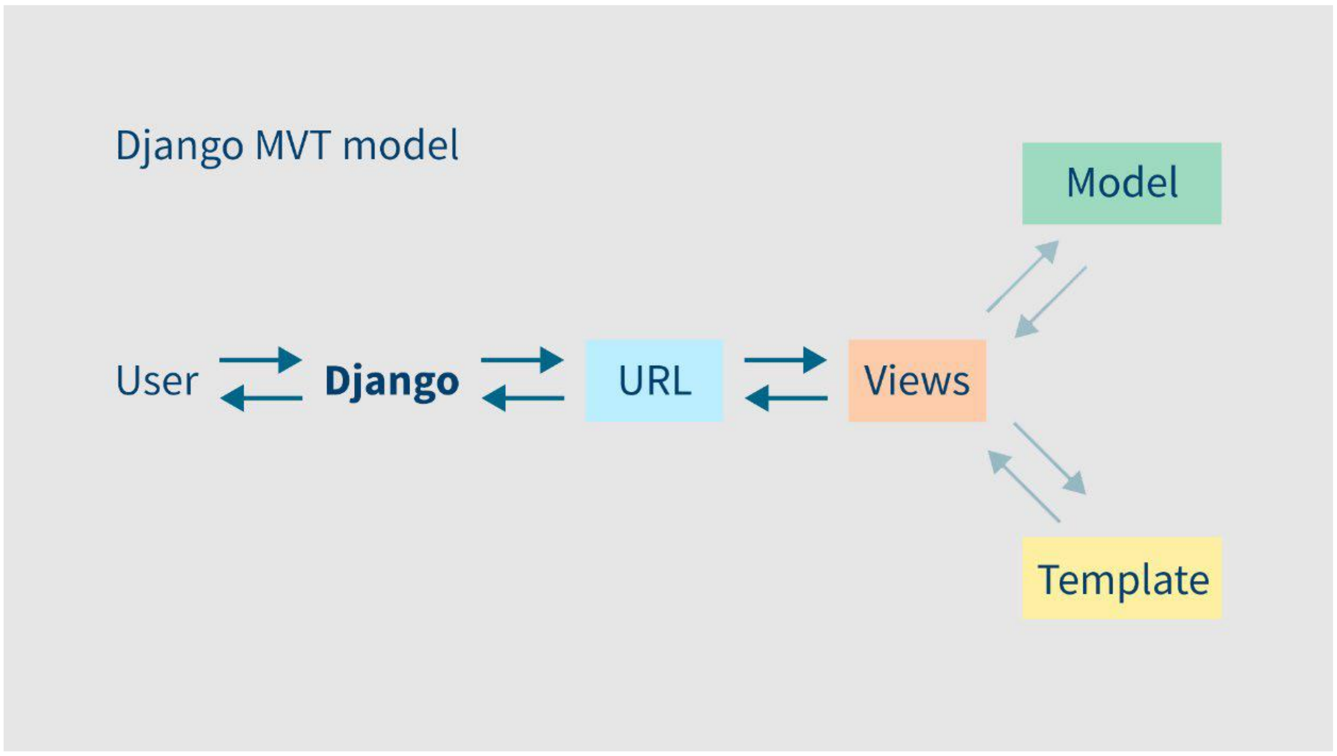
# CHAPTER 4

## System Design

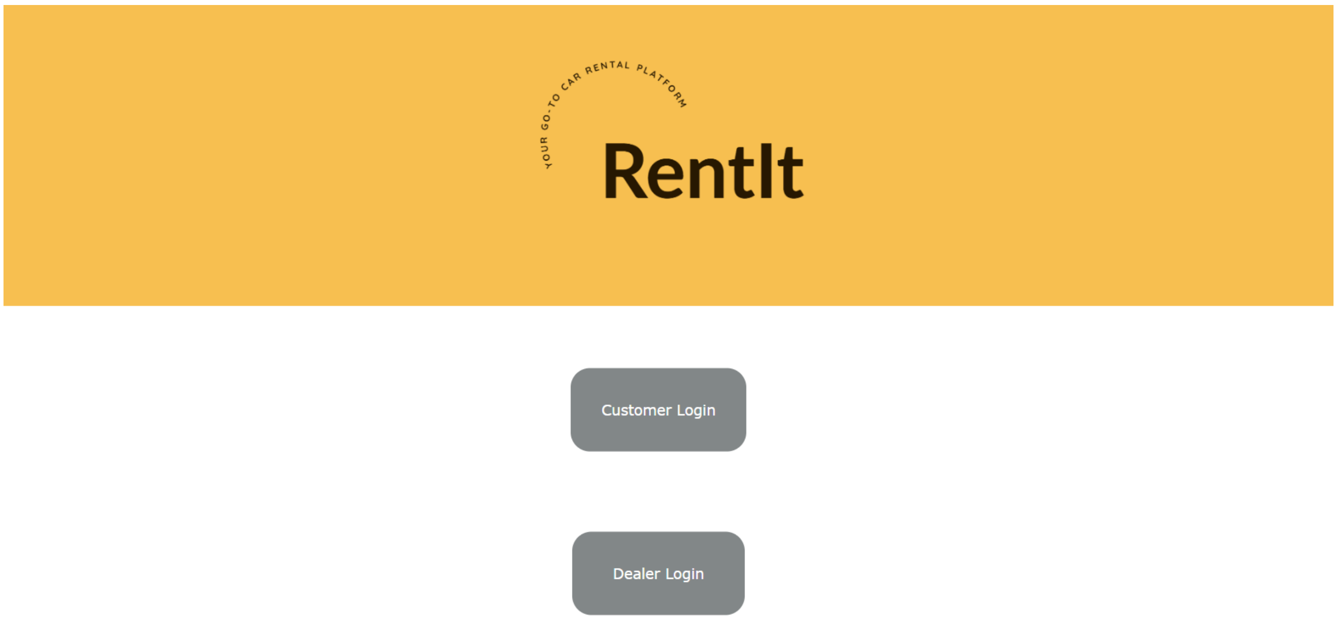
This live server can be accessed through any browser by going to the following port http://127.0.0.1:8000/. Let's discuss the GUI model of the project.

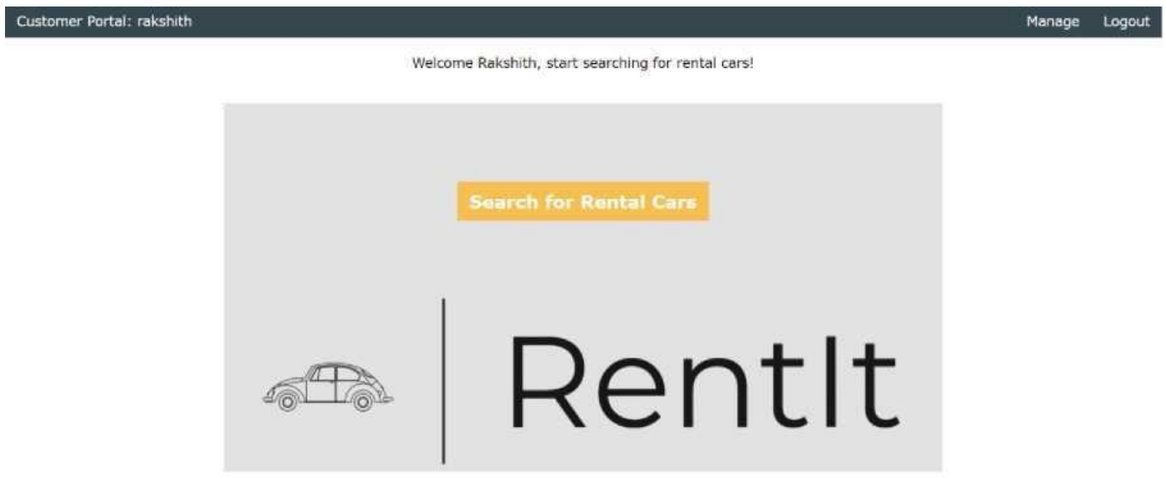
Most of the data processing happens in the server side, i.e.) Django. Django has 4 main components for it to run. They are url, view, model, template. The 'template' consists of all the front-end codes of the project. 'Model' consists of the structure of the project and is used to redirect the processing to the required path. 'View' consists of the main code for the project and is the base file. 'url' consists of all the sub-directories of the present project and is required to redirect the links.



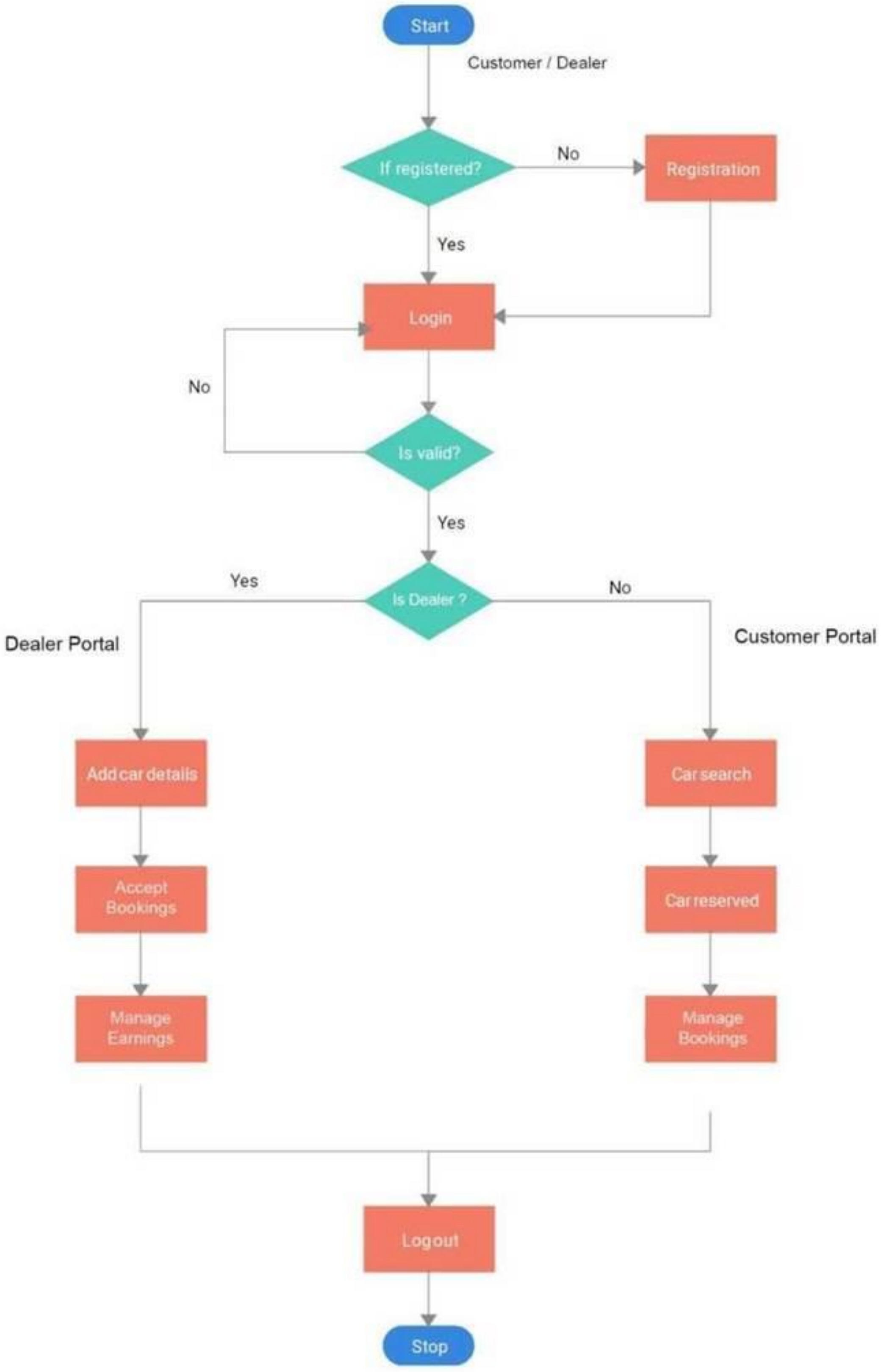


It allows data to pass-in through the ‘GET’ function and feeds it to the database management system. The processed data is finally exported to a live server(local host) through a browser. The output template is a minimal and very user-friendly interface.




Customer Portal: rakshith
Manage
Logout
Welcome Rakshith, start searching for rental cars!
Search for Rental Cars
RentIt

After successfully logging in with valid credentials of a customer, the user will be redirected to the home page of the customer portal. This page welcome the user with their first name. It has 3 buttons for the user to intract with. They are the Search button, Manage button and the logout button. When the user clicks the search button, they will be redirected to the search page. If the manage button is clicked, the user will be redirected to the rental car management page. When clicked ‘logout’, the authentication of the user will be cancelled and the user will be sent to the login page of the portal.

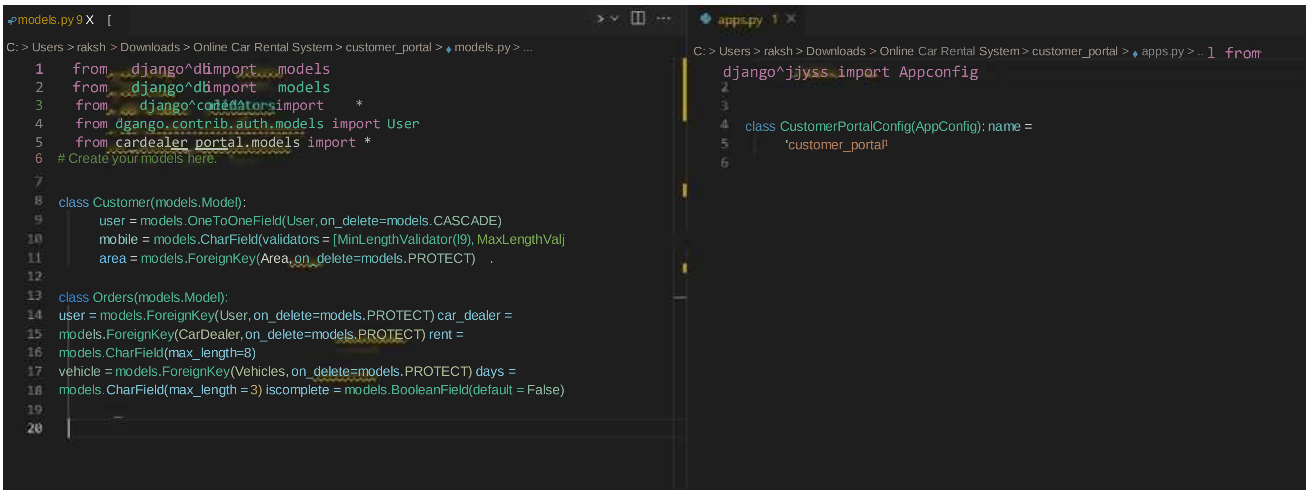

Start
Customer / Dealer
If registered?
No
Registration
Yes
Login
No
Is valid?
Yes
Yes
Is Dealer ?
No
Dealer Portal
Customer Portal
Add car details
Accept Bookings
Manage Earnings
Car search
Car reserved
Manage Bookings
Logout
Stop

# Chapter 5

# CODING AND TESTING

## 5.1CODING

Fig : 5.1: Views.py and urls.py



models.py 9 X [

C: > Users > raksh > Downloads > Online Car Rental System > customer_portal > models.py > ...

```
from   django^dimport   models
from   django^dimport   models
 from    django^corediatorsimport    *
 from dgango.contrib.auth.models import User
 from cardealer portal.models import *
# Create your models here.

class Customer(models.Model):
        user = models.OneToOneField(User, on_delete=models.CASCADE)
        mobile = models.CharField(validators = [MinLengthValidator(l9), MaxLengthValj
        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) iscomplete = models.BooleanField(default = False)
```

apps.py 1 X

C: > Users > raksh > Downloads > Online Car Rental System > customer_portal > apps.py > .. 1 from

```
django^jjyss import Appconfig


class CustomerPortalConfig(AppConfig): name =
        'customer_portal1
```

Fig : 5.2: models.py and apps.py

Views.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 customer_portal.models import *

from django.contrib.auth.decorators import login_required from car_dealer_portal.models import * from django.http import HttpResponseRedirect # Create your views here.

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)

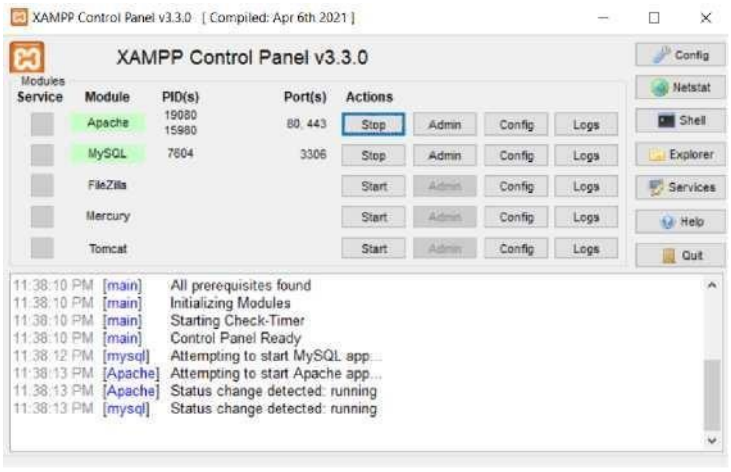
try:

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

## 3.3 EXECUTING THE PROJECT

**Step 1:** Install and Open XAMPP.

**Step 2:** Click ‘Start’ button next to Apache and MySQL to start MySQL service. Create a database called ‘ocrsdjango’ through CMD.



XAMPP Control Panel v3.3.0 [ Compiled: Apr 6th 2021 ]

XAMPP Control Panel v3.3.0

| Service | Module | PID(s) | Port(s) | Actions | | | |
|---|---|---|---|---|---|---|---|
| | Apache | 19080 15980 | 80, 443 | Stop | Admin | Config | Logs |
| | MySQL | 7604 | 3306 | Stop | Admin | Config | Logs |
| | FileZilla | | | Start | Admin | Config | Logs |
| | Mercury | | | Start | Admin | Config | Logs |
| | Tomcat | | | Start | Admin | Config | Logs |

```
11:38:10 PM  [main]    All prerequisites found
11:38:10 PM  [main]    Initializing Modules
11:38:10 PM  [main]    Starting Check-Timer
11:38:10 PM  [main]    Control Panel Ready
11:38:12 PM  [mysql]   Attempting to start MySQL app...
11:38:13 PM  [Apache]  Attempting to start Apache app...
11:38:13 PM  [Apache]  Status change detected: running
11:38:13 PM  [mysql]   Status change detected: running
```

**Fig : 5.4: XAMPP**

**Step 3:** Go to the project directory. Hold shift and right click in the folder. Select open command window here.

Name

car_dealer_portal customer_portal env home ocrs static

@ manage.py 0
requirements.txt 3 run projecttxt

| Date modified | Type |
|---|---|
| View | > |
| Sort by | > |
| Group by | > |
| Refresh | |
| Customize this folder... | |
| Paste | |
| Paste shortcut | |
| Undo Move | Ctrl+Z |
| Q Open command window here | |
| Open PowerShell window here | |
| Give access to | > |
| New | > |
| Properties | |

**Fig :5.5: Opening CMD**

**Step 4**: Install virtualenv, a python plugin to create environments for projects. Enter the following code to install venv.

```
py -m pip install --user virtualenv
```

**Step 5**: Create a virtual environment called ‘env’ by entering the following code.

**py -m venv env**

This creates a folder named ‘env’ in the project directory

**Step 6**: Activate scripts in the environment by entering the following code.

**.\env\Scripts\activate**

**Step 7**: Install the requirements for the project by entering the following code.

**pip install -r requirements.txt**

requirements.txt contains Django and mysqlclient written in it.

**Step 8**: Migrate all the required files and codes by entering the following codes.

**python manage.py makemigrations python manage.py migrate**

**Step 9**: Run the project in a server by entering the following code.

**python manage.py runserver**

```
SB C:\WINDOWS\system32\cmd.exe - python manage.py runserver                                —OX
customer_portal.Orders: (models.W042)Auto-created primary key used when not defining a primarykey type, by default 'django.db.models.AutoField'.
HINT: Configure the DEFAULT_AUTO_FIELD setting or the CustomerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
No changes detected

(env) C:\Users\raksh\Desktop\PT2\0nline Car Rental System>python manage.py migrate System check identified some issues:

WARNINGS:
?: (mysql.W002)MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon insertion, by escalating warnings into errors. It is strongly recommended you activate it. See: htt
ps://docs.djangoproject.eom/en/3.2/ref/databases/#mysql-sql-mode
                                                                                                   car_dealer_p
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the CarDealerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.   ortal.Area:
car_dealer_portal.CarDealer: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.   (models.W04
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the CarDealerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.   2)Auto-
car_dealer_portal.Vehicles: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.   created
HINT: Configure the DEFAULT_AUTO_FIELD setting or the CarDealerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.   primarykey
customer_portal.Customer:                                                                          used when
                                                                                                   not defining a
primarykey type, by default 'django.db.models.AutoField'.



(models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
HINT: Configure the DEFAULT_AUTO_FIELD setting or the CustomerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'. customer_portal.Orders:
(models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
HINT: Configure the DEFAULTAUTOFIELD setting or the CustomerPortalConfig.defaultautofield attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
Operations to perform:
Apply all migrations: admin, auth, car_dealer_portal, contenttypes, customer_portal, sessions Running migrations:

No migrations to apply.

(env) C:\Users\raksh\Desktop\PT2\0nline Car Rental System>python manage.py runserver Watching for file changes with StatReloader Performing system checks...


System check identified some issues:

WARNINGS: car_dealer_portal.Area: (models.W042)Auto-created primarykey used when not defining a primarykey type, by default 'django.db.models.AutoField'.
```

```
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the CarDealerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g    .
car_dealer_portal.CarDealer: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.    'django.db.models.BigAutoField'.
        HINT: Configure the DEFAULTAUTOFIELD setting or the CarDealerPortalConfig.defaultautofield attribute to point to a subclass of AutoField, e.g    .
car_dealer_portal.Vehicles: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.    'django.db.models.BigAutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the CarDealerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g    .
customer_portal.Customer: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.    'django.db.models.BigAutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the CustomerPortalConfig.default_auto_field attribute to point to a subclass of AutoField, e.g.
customer_portal.Orders: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.    'django.db.models.BigAutoField1.

      HINT: Configure the DEFAULTAUTOFIELD setting or the CustomerPortalConfig.defaultautofield attribute to point to a subclass of AutoField, e.g.    'django.db.models.BigAutoField'.

System check identified 5 issues (0 silenced).
April 12, 2022 - 02:44:45
Django version 3.2.9, using settings 'pers.settings' Starting development server at http://127.0.0.1:8000/ Quit the server with CTRL-BREAK.
```

**Fig : 5.6: Executing Server**

# CHAPTER 6

# SUMMARY AND CONCLUSIONS

Car rental business has emerged with a new goody compared to the past experience where every activity concerning car rental business is limited to a physical location only. Even though the physical location has not been totally eradicated; the nature of functions and how these functions are achieved has been reshaped by the power of internet. Nowadays, customers can reserve cars online, rent car online, and have the car brought to their door step once the customer is a registered member or go to the office to pick the car. The web-based car rental system has offered an advantage to both customers as well as Car Rental Company to efficiently and effectively manage the business and satisfies customers' need at the click of a button.

## 6.1 FUTURE ENHANCEMENT

The car rental industry is growing very fast and demand for this industry has increased globally in the past few years. The global car rental industry looks good and seems so many opportunities for this sector in the future. With the increase of national and international tourists in different countries, the industry is expecting more growth and bright car rental business opportunities.

There are several factors that are responsible for the growth and development of the Car Rental Industry. Here are some of the most prominent reasons behind the success of the Car Rental Industry-

# CHAPTER 7

# APPENDIX

## 7.1 SCREENSHOT

In this Project there are 2 separate portals each showcasing different users' activities and abilities. The two portals being Customer portal and admin portal, each has its own features to perform and sustain.

Both have a user-friendly and minimal interface for easy access. This webpage is screened through a live server in the local host through Django's framework.

RentIt
Customer Login
Dealer Login

**Fig : 4.1.1: Home page**

The landing page / Home page of this project prompts the user to select any of the two options provided. The user can either login in as a customer by clicking 'Customer login' or login in as a dealer by choosing 'Dealer login'.

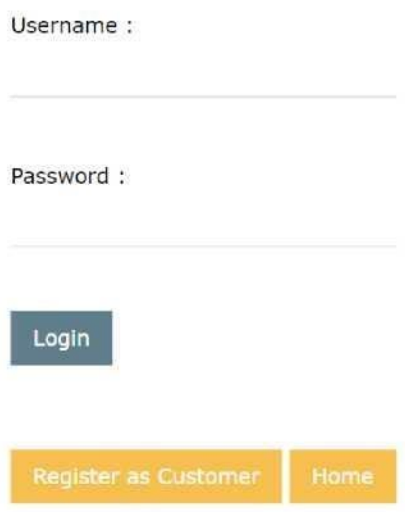


**Fig : 7.1: Login Page**

When the user clicks any one of the previous options, they will be directed to the login page of that respective portal. This page prompts the user to enter a user name and a password. This also has a registration button and a button to redirect to the landing page of the project.

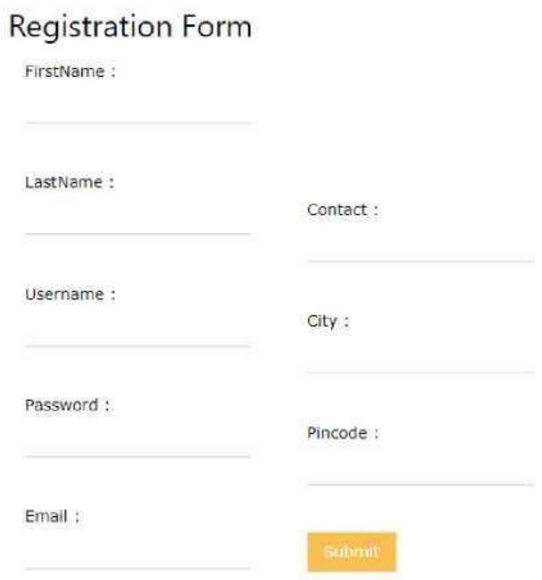
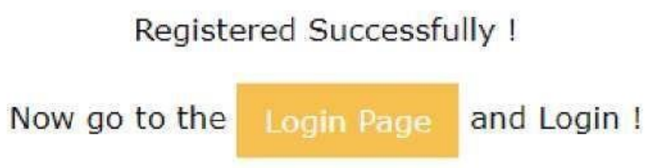


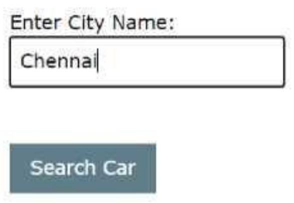
**Fig : 7.2: Registration Page**

This page prompts the user to enter multiple details to get registered to the website inducing Name, username, password, mail, city , contact and pincode. This registration form is common for both customers and dealers.


Registered Successfully !

Now go to the Login Page and Login !

Fig : 7.3: Registration successful


Enter City Name:

Chennai

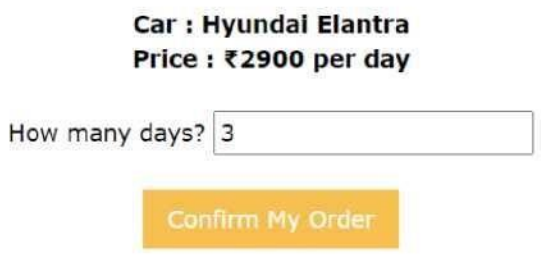
Fig : 7.4: Search page

This page prompts the user to enter a city name and sends it to the database to get matching cars available in that city.

| | | | | | |
|---|---|---|---|---|---|
| Customer Portal: rakshith | | | | Search Cars | Manage Logout |
| | | | Search Results | | |
| Vehicle | Color | Capacity | Description | Pincode | |
| BMW 5 Series | Phantom Black | 5 | The go-to luxury ride | 600034 | |
| Audi RS5 | Off-White | 4 | Gives you wings ! | 600034 | |
| Hyundai Elantra | Matte Gray | 5 | Looks cool, drives even cooler | 600017 | |
| Skoda Superb | Satin Green | 5 | Looks the best. The BEST ! | 600001 | |
| Volkswagen Polo | Racing Red | 5 | The pocket rocket! | 600042 | |

Fig : 7.5: Search Results

After the user enters a city name, the cars that match that city name and is available to rent will be sent from the database. That output will be listed in the search results page.
The output consists of all the details of the car that is given by the dealer. This page allows the user to select the any one car to be rented by prompting the user to click the 'Rent' button on the right end of each row.



**Car : Hyundai Elantra**
**Price : ₹2900 per day**

How many days? 3

Confirm My Order

**Fig: 7.6: Rent page**

After the user selects a car and click on 'Rent', the user will be redirected to this page. This page again summarizes the details of the car and prompts the user to enter the number of days the car needs to be rented. It also displays the rent per day for the user to be aware of.

## Order Completed

**Your Vehicle Order Has Been Processed!**

**Total Rent : ?8700**

**Vehicle :** Hyundai Elantra

**Duration :** 3 days

**Car-Dealer :** Dealerl d

**Car-Dealer's Contact: 3213213210**

**Fig : 7.7: Confirmation page**

When the 'Confirm My order' button is clicked, the order is generated and processed by the backend. The user is then redirected to this page confirming their order. This page displays the total rent amount, vehicle name, duration, and the car dealer's details including their name and contact number.

Your Orders

| Car | Days | Rent | Capacity | Description | Pincode | Action |
|---|---|---|---|---|---|---|
| Hyundai Elantra | 3 | ?8700 | 5 5eats | Looks cool, drives even cooler | 600017 | |

**Fig : 4.1.10: Management page**

After the order is placed, the order details can be viewed again the Management page by clicking the ‘Manage’ button in the navigation bar above. This page allows the user to look into the details of their orders and also allows them to delete the booking they have made by clicking the ‘Delete’ button of the right end of the row.

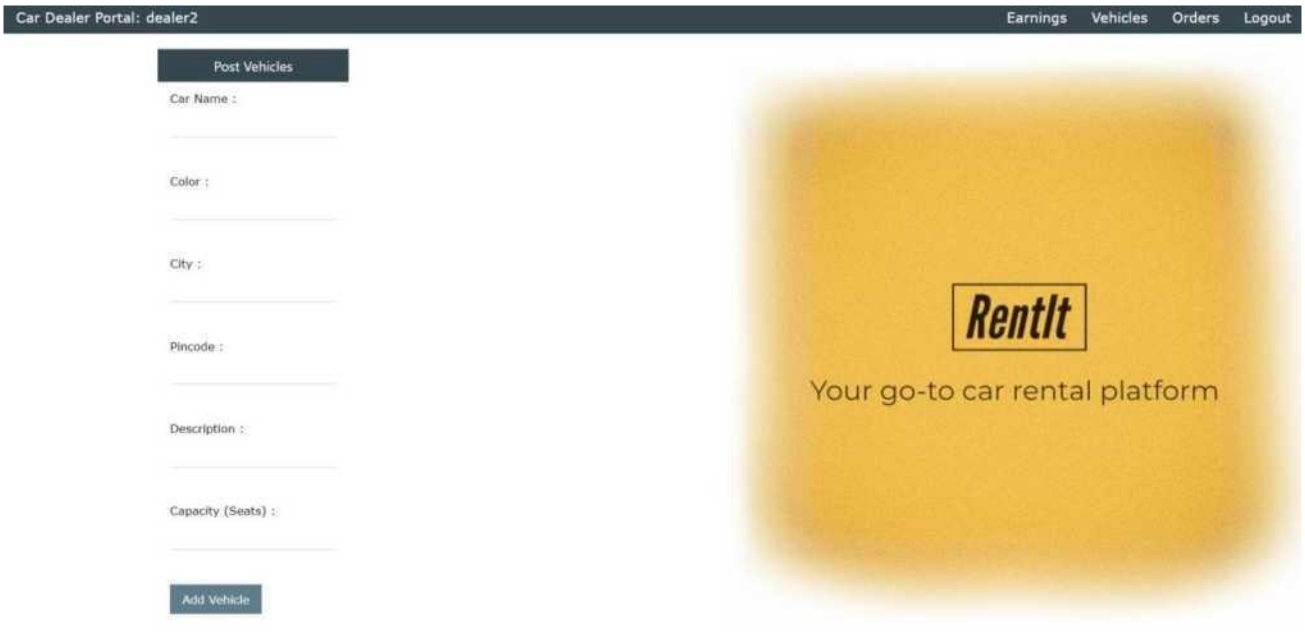


**Fig : 4.1.11: Dealer home page**

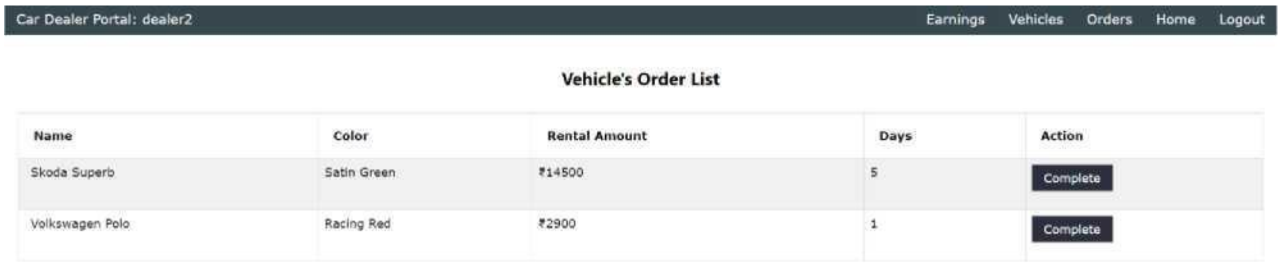
After successfully logging in with valid credentials of a dealer, the user will be redirected to the home page of the dealer portal. This page allows the dealer to add vehicle details and sumbit them to add a new car to the rental management system. It has 4 buttons in the navigation bar for the user to intract with. They are the earnings button, vehicles button, orders button and the logout button.

When the user clicks the earning button, they will be redirected to the earnings page. If the vehicle button is clicked, the user will be redirected to the vehicle management page. When ‘orders’ button is clicked, the user will be redirected to the order list page. When clicked ‘logout’, the authentication of the user will be cancelled and the user will be sent to the login page of the portal.

| Car Dealer Portal: dealer2 | | | | Earnings Vehicles | Orders Home Logout |
|---|---|---|---|---|---|
| | | Posted Vehicles | | | |
| Car | Color | Capacity | Pincode | Address | Action |
| Skoda Superb | Satin Green | 5 Seats | 600001 | chennai | n |
| Volkswagen Polo | Racing Red | 5 Seats | 600042 | chennai | n |
| Audi RS5 | Off-White | 4 Seats | 600034 | chennai | |

**Fig : 4.1.12: Vehicles page**

After clicking the ‘Vehicle’ button, the user is redirected to the vehicles page where the list of all vehicles posted by the dealer is displayed. The dealer has the access to remove any car he added by clicking the ‘Delete’ button.


Car Dealer Portal: dealer2 Earnings Vehicles Orders Home Logout

Vehicle's Order List

| Name | Color | Rental Amount | Days | Action |
|---|---|---|---|---|
| Skoda Superb | Satin Green | ₹14500 | 5 | Complete |
| Volkswagen Polo | Racing Red | ₹2900 | 1 | Complete |

**Fig : 4.1.13: Order page**

If the user clicks the ‘Orders’ button, they will be redirected to the orders page. Here the user can take a look of the all the orders they received from the customers. This page displays the car details, the number of days car needs to be rented and the total amount of the order. The order is validated only when the dealer clicks on the ‘Complete’ button on the right end of each order.

History

Name

Audi RS5 Skoda Superb

Volkswagen Polo

| Rent | Days | Seat |
|---|---|---|
| ?6960 | 3 | 4 |
| ?14500 | 5 | 5 |
| ?2900 | 1 | 5 |

**Fig : 4.1.14: Earnings page**

The earnings page is displayed when the user clicks the ‘Earnings’ button on the navigation bar. This page lists all the order history that the dealer has received in an ordered manner, with the rent, and number of days. This page also sums up the total earnings of the dealer and displays it in the navbar.

# CHAPTER 8

# BIBLIOGRAPHY AND REFERENCE

- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction
- https://capablemachine.com/
- https://www.w3schools.com/django/index.php
- https://stfalcon.com/en/blog/post/car-rent-website-development
- https://www.globenewswire.com/newsrelease/2021/11 /25/2340991/28124/en/India-Car-Rental-Market-Report-2021-Marketis-Proiected-to-Grow-at-a-CAGR-of-9-83-to-Reach-2-030-Billion-by-2026-from-1 053-Billion-in-2019.html
- https://www.forbes.com/advisor/travel-rewards/car-rental-prices-are-up-what-are-thealternatives/
- https://www.youtube.com/watch?v=PtQiiknWUcI
- https://www.educba.com/diango-architecture/