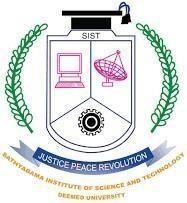
**RentIt – A car rental platform**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

**RAKSHITH.R (Reg. No:39110832)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**March 2022**

 **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC**

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

[**www.sathyabama.ac.in**](http://www.sathyabama.ac.in)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the Bonafide work of **RAKSHITH.R (Reg.No: 39110832)** who carried out the project entitled **“ RentIt – A car rental platform ”** under our supervision from December 2021 to March 2022.

**Internal Guide**

**Dr. Shankari M.**

**Head of the Department**

**Submitted for Viva voce Examination held on**

**Internal Examiner External Examiner**

**DECLARATION**

I **RAKSHITH.R (Reg. No: 39110832)** hereby declare that the Project Report entitled **RentIt – A car rental platform** done by me under the guidance of **Dr. Shankari M.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

|  |  |
| --- | --- |
| **DATE:** |  |
| **PLACE:** CHENNAI | **SIGNATURE OF THE CANDIDATE** |

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J. Albert Mayan, M.E., Ph.D., for** his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

**COURSE COMPLETION CERTIFICATE**

**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 user-friendly 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 sub-directories of the present project and is required to redirect the links.

When the manage.py file is executed through the terminal, it re-routes 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 sub-directory 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**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No.** | **TITLE** | **PAGE No.** |
|  | Abstract | **vi** |
|  | List of figures | **ix** |
|  | List of abbreviations | **xi** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **AIM AND SCOPE OF THE INVESTIGATION** | **3** |
|  | 2.1 Aim | **3** |
|  | 2.2 Background of the project | **3** |
|  | 2.3 Scope | **3** |
|  | 2.4 Development environment software | **4** |
|  | 2.5 Hardware | **6** |
|  | 2.6 Operation environment | **6** |
| **3** | **Methodology and Execution** | **7** |
|  | 3.1 Methodology | **7** |
|  | 3.2 Working of the model | **9** |
|  | 3.3 Execution of the project | **13** |
| **4** | **RESULTS** | **15** |
|  | 4.1 Results and output | **15** |
|  | 4.2 Sample Codes | **25** |
| **5** | **SUMMARY AND CONCLUSION** | **38** |
|  | **REFERENCES** | **38** |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE NAME** | **PAGE NO.** |
| 2.1 | Table for operation environment | 6 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO:** | **FIGURE NAME** | **PAGE NO** |
| 2.4.1 | Django | 4 |
| 2.4.2 | MySQL | 5 |
| 3.2.1 | Project folder | 9 |
| 3.2.2 | App folder | 10 |
| 3.2.3 | Templates folder for customer portal | 11 |
| 3.2.4 | Views.py and urls.py | 11 |
| 3.2.5 | Models.py and apps.py | 12 |
| 3.2.6 | HTML template for dealer’s portal | 12 |
| 3.3.1 | XAMPP | 13 |
| 3.3.2 | Openimg CMD | 13 |
| 3.3.3 | Executing Server | 14 |
| 4.1.1 | Home page | 15 |
| 4.1.2 | Login page | 16 |
| 4.1.3 | Registration page | 16 |
| 4.1.4 | Registration successfull | 17 |
| 4.1.5 | Customer home page | 17 |
| 4.1.6 | Search page | 18 |
| 4.1.7 | Search results | 18 |
| 4.1.8 | Rent page | 19 |
| 4.1.9 | Confirmation page | 19 |
| 4.1.10 | Management page | 20 |
| 4.1.11 | Dealer home page | 20 |
| 4.1.12 | Vehicles page | 21 |
| 4.1.13 | Order page | 21 |
| 4.1.14 | Earnings page | 22 |
| 4.1.15 | Admin login | 22 |
| 4.1.16 | Admin dashboard | 23 |
| 4.1.17 | Edit user | 23 |
| 4.1.18 | Flow Diagram | 24 |

**LIST OF ABBREVIATIONS**

MTV – Model, template, Views

.py - Python extension file

.js – Java script extension file

WP – Web portal

ocrs – Online car rental system

venv – Virtual Environment

IDE - Integrated Development Environment

DFD – Data flow diagram

GUI - Graphical User Interface

DBMS – Database management system

HTML – Hypertext markup language

**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.

On December 31st, 2019, the first confirmed case of the deadly COVID-19virus was in

Wuhan, China. 2019, which was slowly gaining ground and growth at the time. The World

Health Organization (WHO) formally declared the COVID-19 outbreak on the 11th January,

2020 as a pandemic. Since then, the virus has steadily infected more individuals across the

globe. On 13 January 2020, the first case outside China was registered in Thailand (Hui et al,

2020). As the virus continually ravages and cripples many countries, no country has been

spared. Around the world, the economy is steadily declining as a result of the effect of virus.

The pandemic of COVID-19 is a rapidly emerging pandemic and is already an evolving virus.

The virus is transmitted from person to person mainly by coming in contact with the droplets

of an infected person (IAMAT, 2020). Various Steps were listed to avoid the spread, but the

virus continues to increase in numbers.

On December 31st, 2019, the first confirmed case of the deadly COVID-19virus was in

Wuhan, China. 2019, which was slowly gaining ground and growth at the time. The World

Health Organization (WHO) formally declared the COVID-19 outbreak on the 11th January,

2020 as a pandemic. Since then, the virus has steadily infected more individuals across the

globe. On 13 January 2020, the first case outside China was registered in Thailand (Hui et al,

2020). As the virus continually ravages and cripples many countries, no country has been

spared. Around the world, the economy is steadily declining as a result of the effect of virus.

The pandemic of COVID-19 is a rapidly emerging pandemic and is already an evolving virus.

The virus is transmitted from person to person mainly by coming in contact with the droplets

of an infected person (IAMAT, 2020). Various Steps were listed to avoid the spread, but the

virus continues to increase in numbers.

On December 31st, 2019, the first confirmed case of the deadly COVID-19virus was in

Wuhan, China. 2019, which was slowly gaining ground and growth at the time. The World

Health Organization (WHO) formally declared the COVID-19 outbreak on the 11th January,

2020 as a pandemic. Since then, the virus has steadily infected more individuals across the

globe. On 13 January 2020, the first case outside China was registered in Thailand (Hui et al,

2020). As the virus continually ravages and cripples many countries, no country has been

spared. Around the world, the economy is steadily declining as a result of the effect of virus.

The pandemic of COVID-19 is a rapidly emerging pandemic and is already an evolving virus.

The virus is transmitted from person to person mainly by coming in contact with the droplets

of an infected person (IAMAT, 2020). Various Steps were listed to avoid the spread, but the

virus continues to increase in numbers

On December 31st, 2019, the first confirmed case of the deadly COVID-19virus was in

Wuhan, China. 2019, which was slowly gaining ground and growth at the time. The World

Health Organization (WHO) formally declared the COVID-19 outbreak on the 11th January,

2020 as a pandemic. Since then, the virus has steadily infected more individuals across the

globe. On 13 January 2020, the first case outside China was registered in Thailand (Hui et al,

2020). As the virus continually ravages and cripples many countries, no country has been

spared. Around the world, the economy is steadily declining as a result of the effect of virus.

The pandemic of COVID-19 is a rapidly emerging pandemic and is already an evolving virus.

The virus is transmitted from person to person mainly by coming in contact with the droplets

of an infected person (IAMAT, 2020). Various Steps were listed to avoid the spread, but the

virus continues to increase in numbers.

**CHAPTER 2**

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

**2.1 AIM**

The advancement in Information Technology and internet penetration has greatly enhanced various business processes and communication between companies (services provider) and their customers of which car rental industry is not left out. The two main aims for this project are to produce a web-based system that allow customer to register and reserve car online and for the company to effectively manage their car rental business and to ease customer’s task whenever they need to rent a car. Car rental industry, that has not become familiar to this region which must be taken into account.

**2.2 BACKGROUND OF THE PROJECT**

This project traverses a lot of areas ranging from business concept to computing field to be able to achieve the project objectives. The area covers include: Car rental industry, Django technology, front-end development and customer interaction.

**2.3 Scope**

There is enough scope for car rental services in India in the future if they understand the Indian shoppers’ psyche and cater to their needs. The main scope of this project is to develop a user-friendly car rental platform and provide people the access to this huge rental industry with genuine car rental policies and extend the automotive/rental sector.

**2.4 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 nextprogram on the list. This alone does wonders in motivating software developersto 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.

**2.5 Hardware**

2.5.1 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.

2.5.2 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.

**2.6 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**

**3.1 METHODOLOGY**

The continuing spread of COVID-19 disease is increasingly transforming our everyday lives.

We need data collection and analysis with visualizations. In this case, Power BI is also used

as a method to rapidly evaluate data. For worksheets and dashboards, visualizations. Power

BI helps us to build dashboards that provide actionable insights for company which spreads

more rapidly. Products from Power BI are often used in virtualized environments if optimized

with proper hardware and system of operation. You will work with more unordered COVID-

19 with Power BI, Data and build a variety of visualizations using the built-in features

offered by Power BI. In addition, many methods of drilling the COVID-19 data and data

would be able to gain great context. Exploring data in a matter of minutes. Power BI enables

businesses to evaluate future data without any future targets. We investigate visualizations

and observe information from various approaches to COVID-19. With hypothetical

visualizations and a feature of adding components for comparison and analysis. The user-

friendly feature is the major strength of Power BI

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 Power BI Desktop, unless you are confident that your information

is already in the format you need for visualization. Which may be the case if, you prefer to

use a scripting language like R or Python to wrangle your data.

If you're used to Excel, you might assume that the way to start analyzing your data in Power

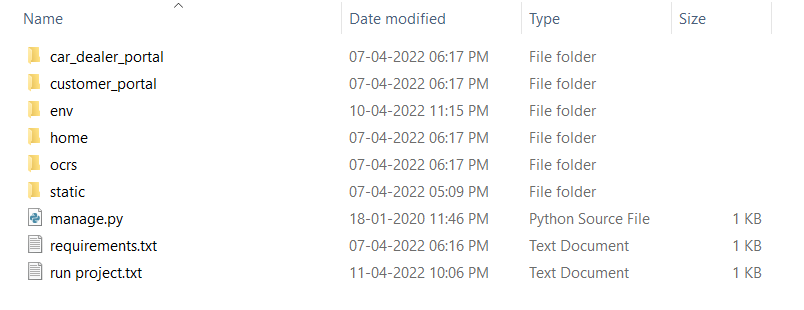
BI is to select File > Open. But you would be wrong — File > Open is only for an existing

Power BI project.

Instead, click the Get Data button on the Home tab to import new data, select your sort of

data source, and then click Link

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.



**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.

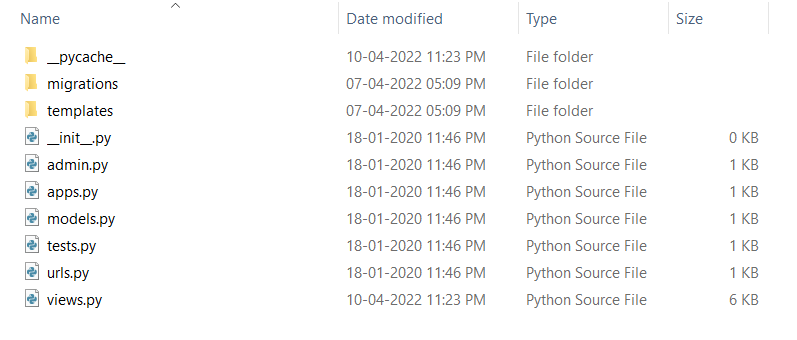
The ‘env’ folder is the folder that contains all the software requirements for the project. Django, mysqlclient source files are all extracted here and is accessed by python.

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.

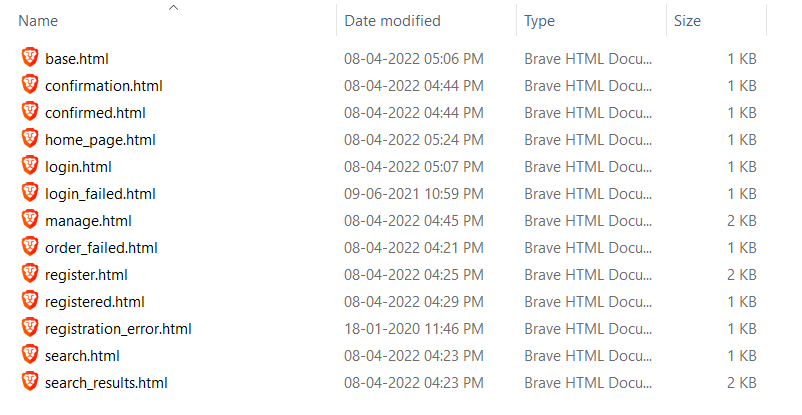
**App folders :**



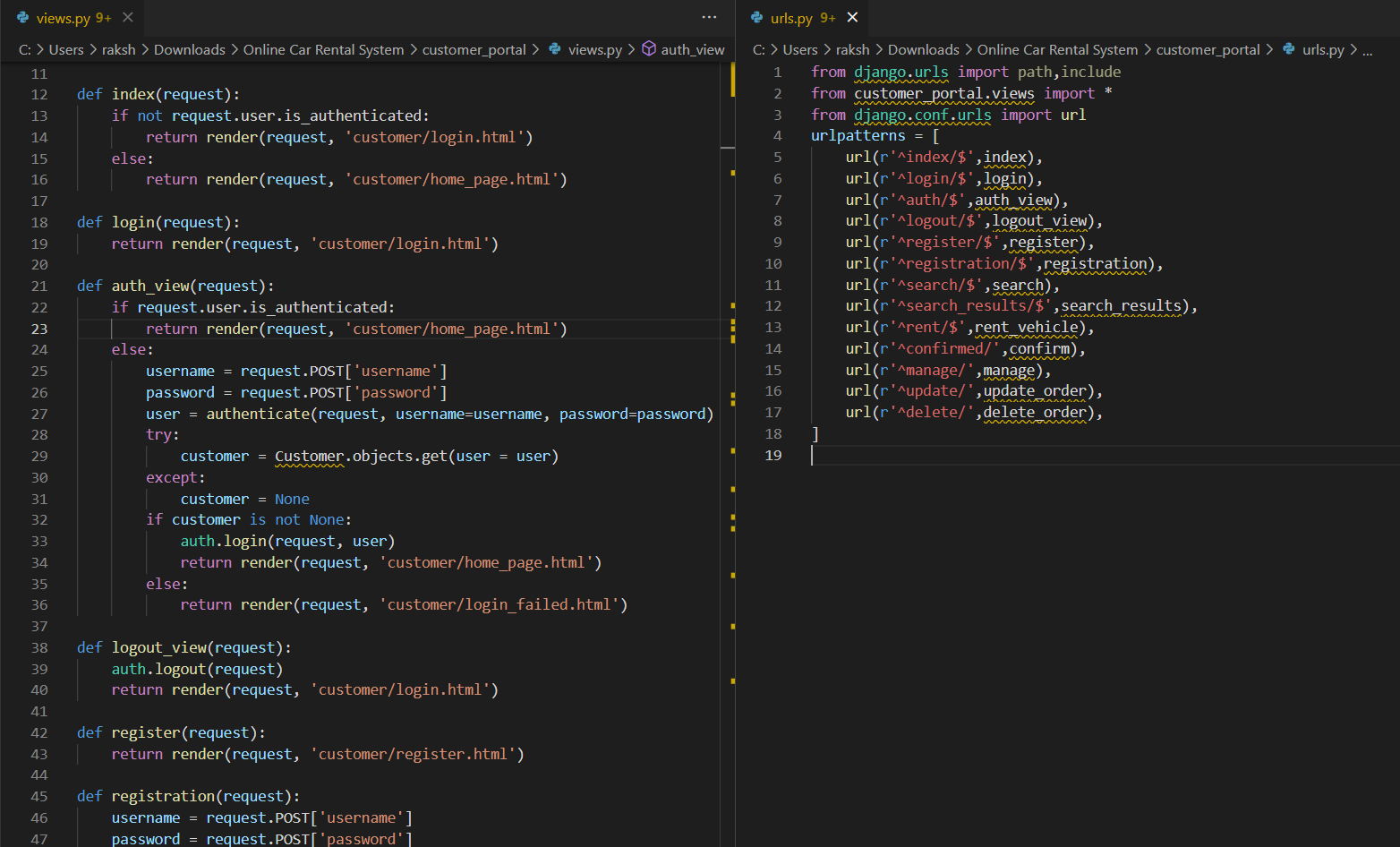
**Fig : 3.2.2: App folder**

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

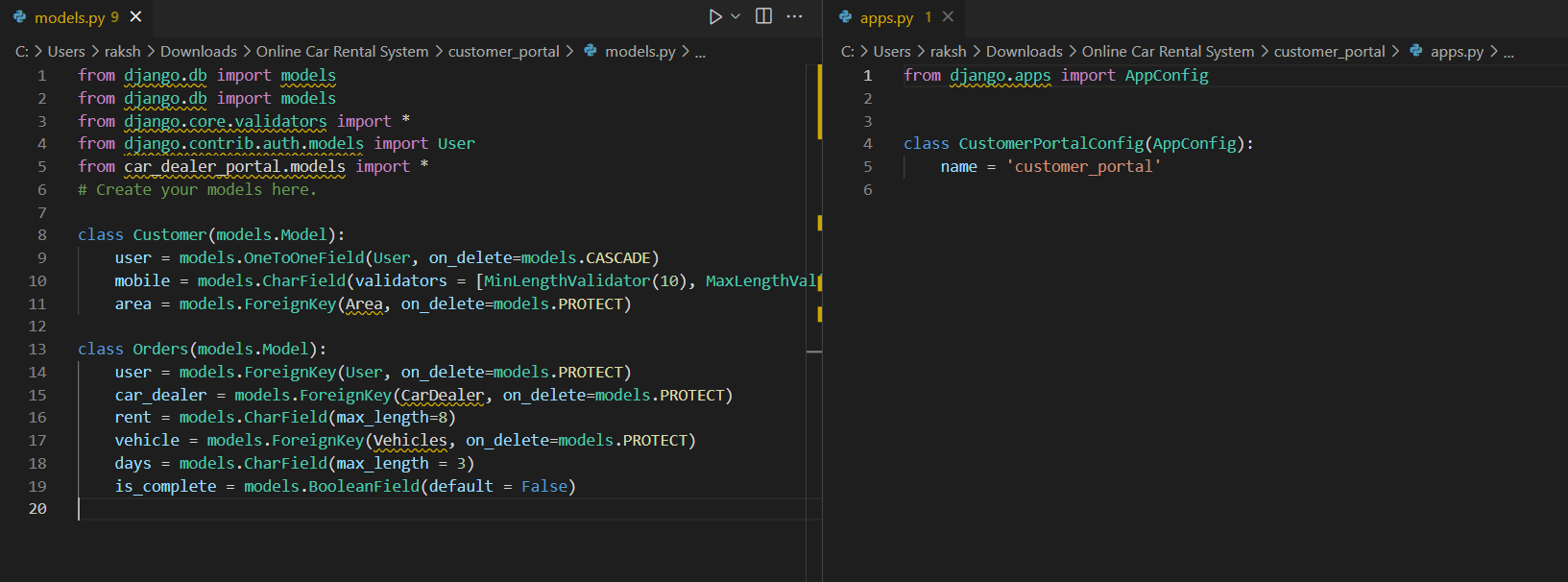
The ‘template’ folder consists of all the front-end codes of the project i.e.) the HTML files of the project. ‘Model’ consists of the structure of the project and is used to redirect the processing to the required path. ‘View’ consists of the main codes for the project and is the base file. It has several 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.



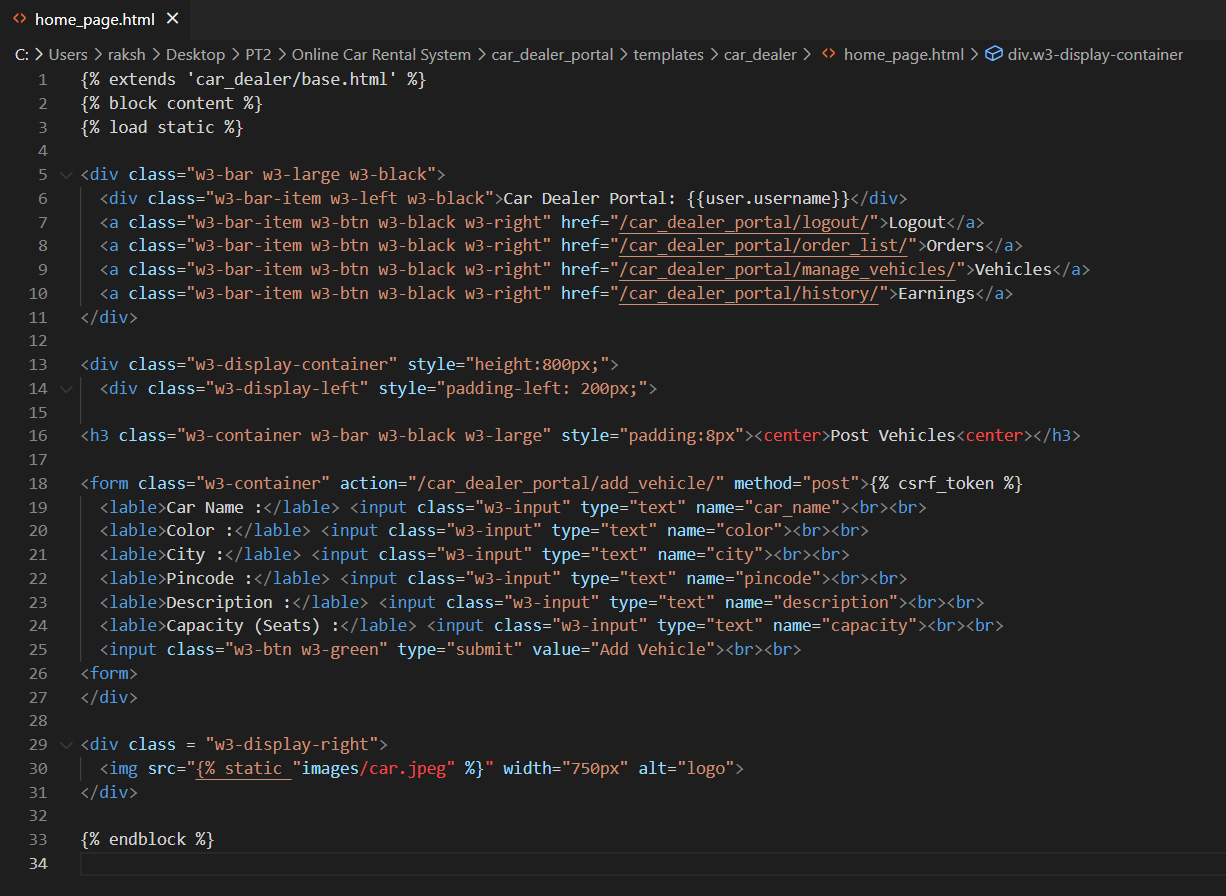
**Fig : 3.2.3: Templates folder of customer portal**

****

**Fig : 3.2.4: Views.py and urls.py**



**Fig : 3.2.5: models.py and apps.py**

****

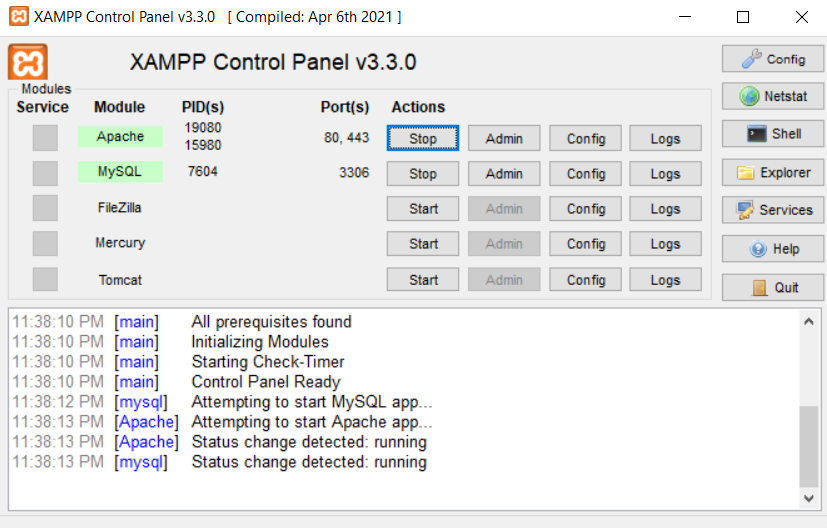
**Fig : 3.2.6: HTML template - Home page of dealer’s portal**

**3.3 EXECUTING THE PROJECT**

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

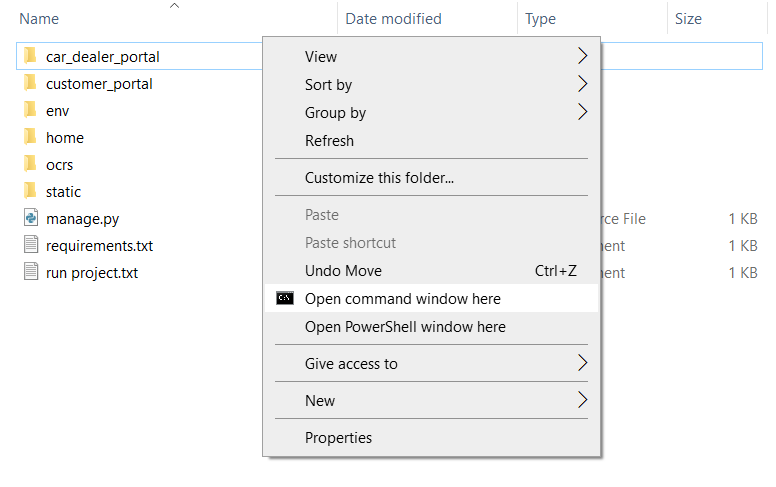
**Step 2:** Click ‘Start’ button next to Apache and MySQL to start MySQL service.

Create a database called ‘ocrsdjango’ through CMD.



**Fig : 3.3.1: XAMPP**

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



**Fig : 3.3.2: 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.

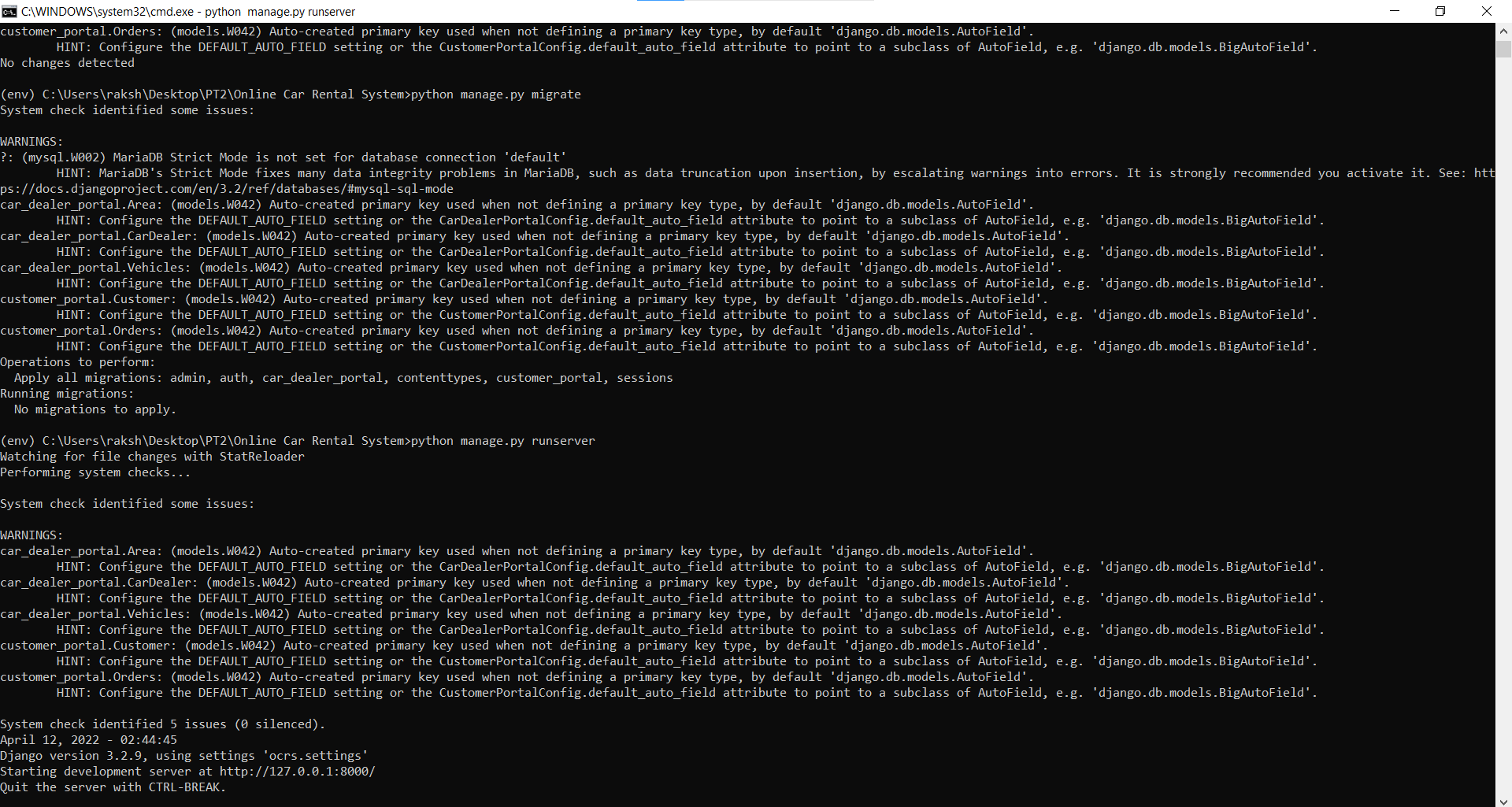
**Step 8**: Migrate all the required files and codes by entering the following codes.

**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**

****

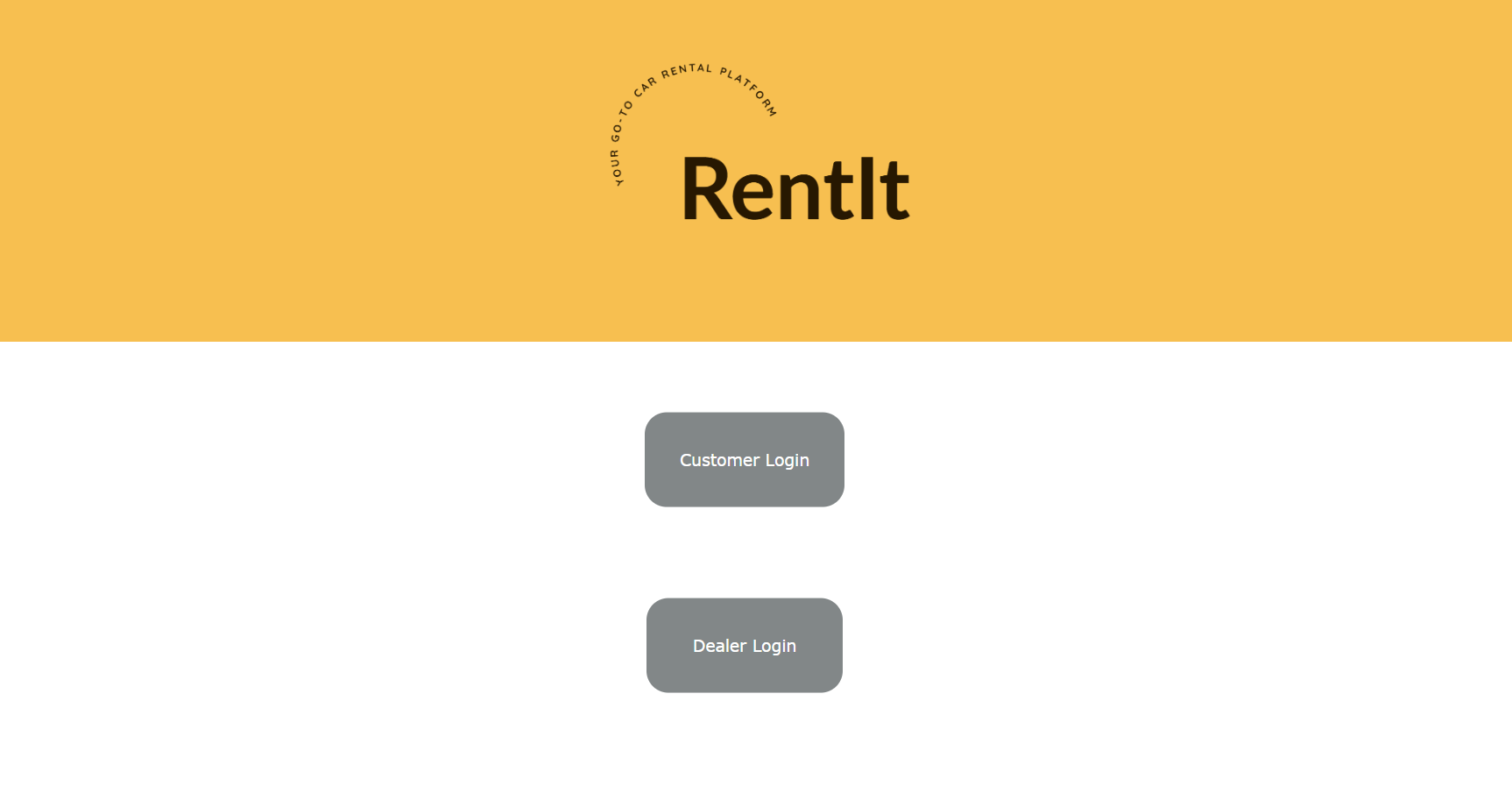
**Fig : 3.3.3: Executing Server**

**Chapter 4 RESULTS**

**4.1 RESULTS AND OUTPUT**

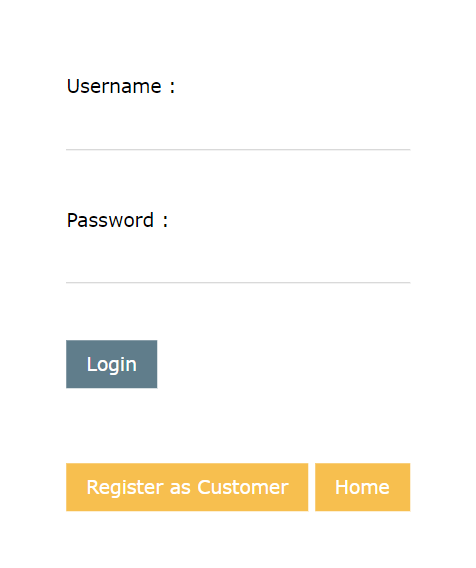
In this Project there are 2 separate portals each showcasing different users’ activities and abilities. The two portals being Customer portal and admin portal, each has its own features to perform and sustain.

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. 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.



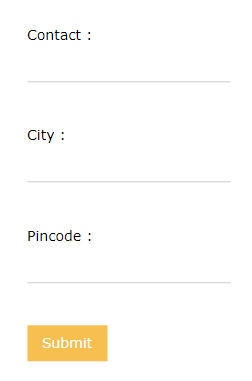
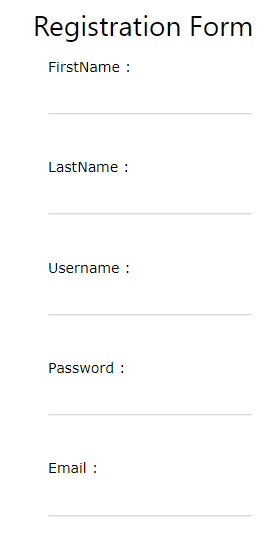
**Fig : 4.1.1: Home page**

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’.



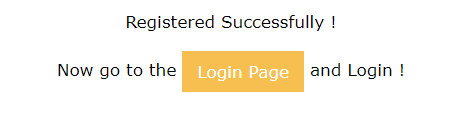
**Fig : 4.1.2: Login Page**

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.

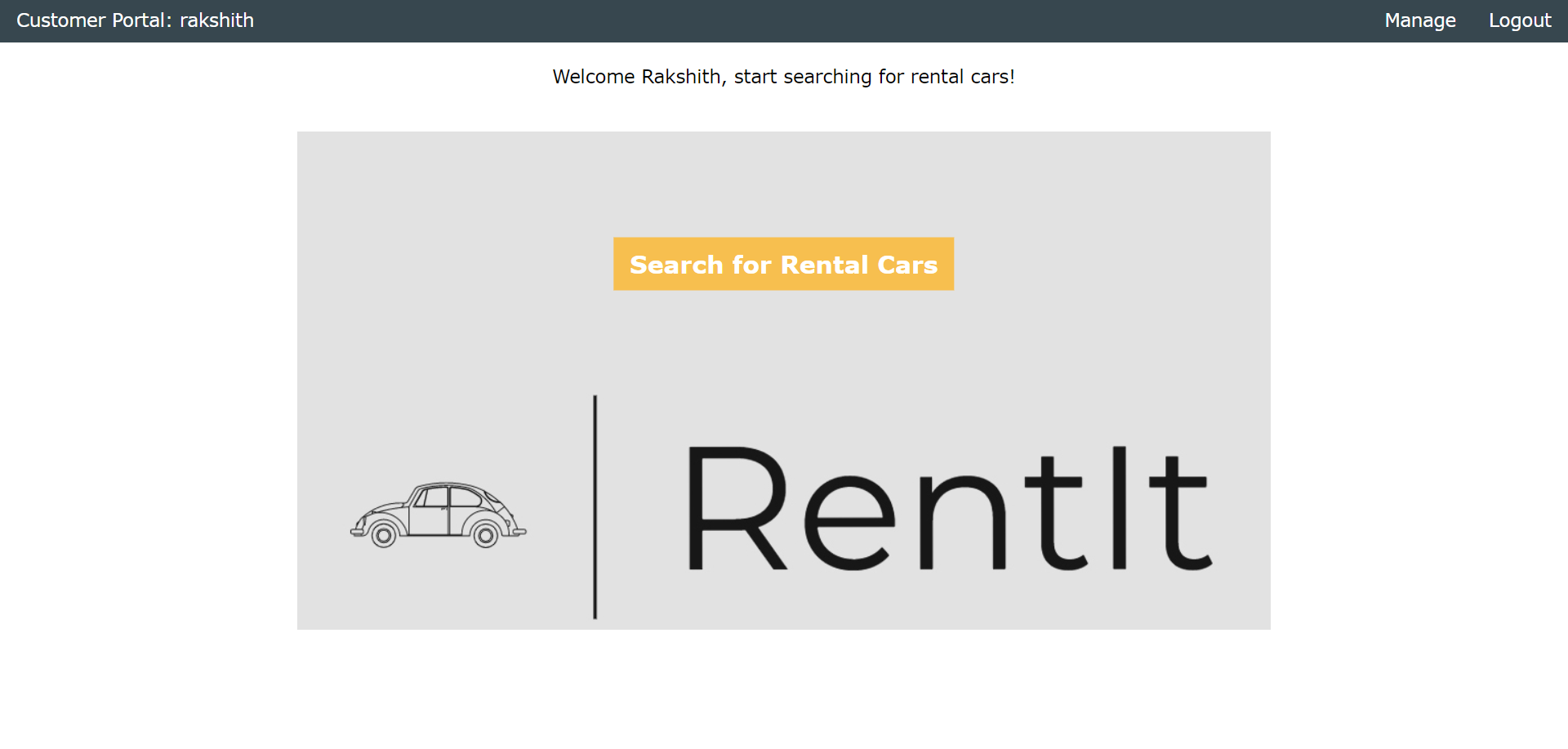


**Fig : 4.1.3: Registration Page**

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

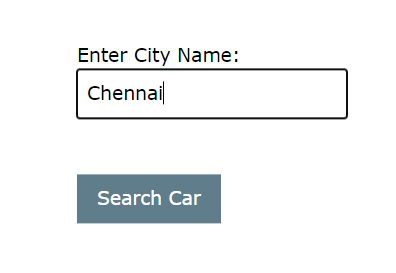


**Fig : 4.1.4: Registration successful**



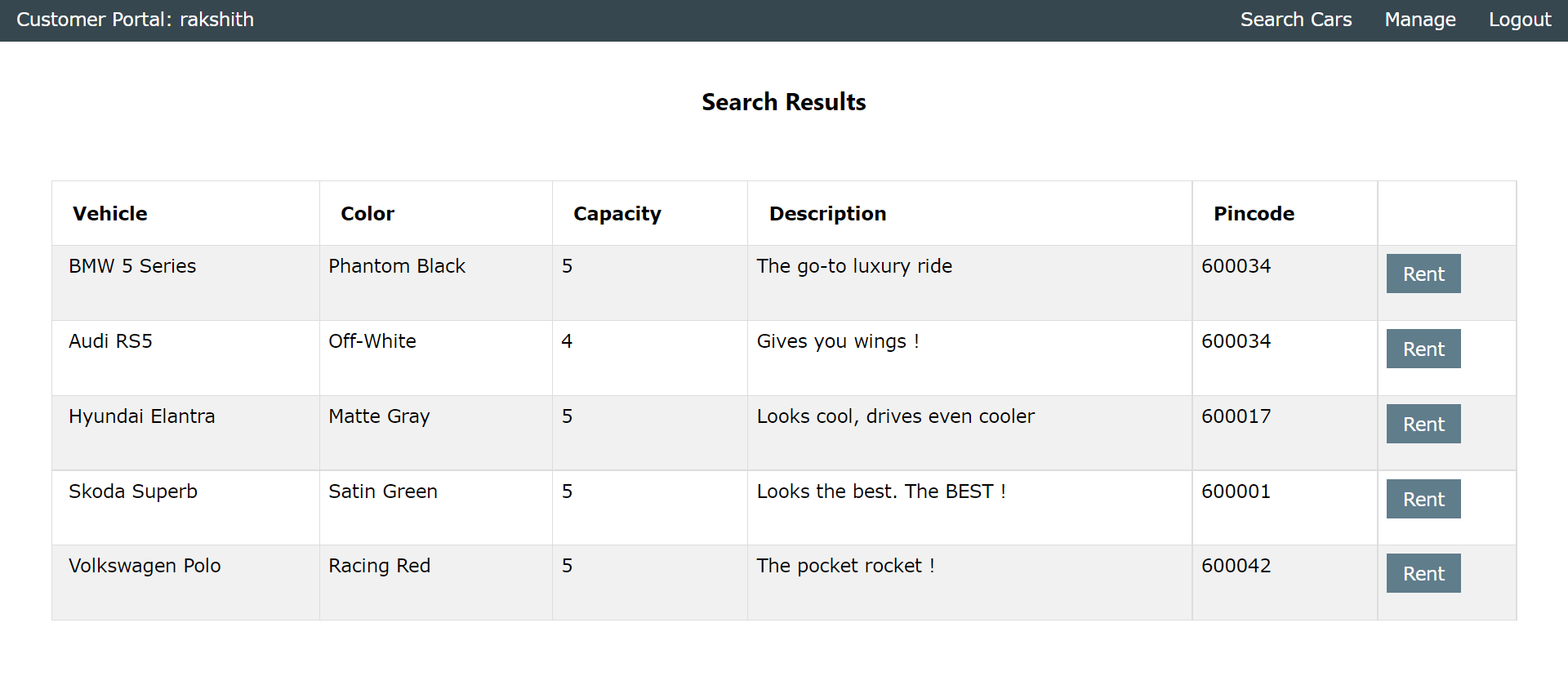
**Fig : 4.1.5: Customer Home page**

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.



**Fig : 4.1.6: Search page**

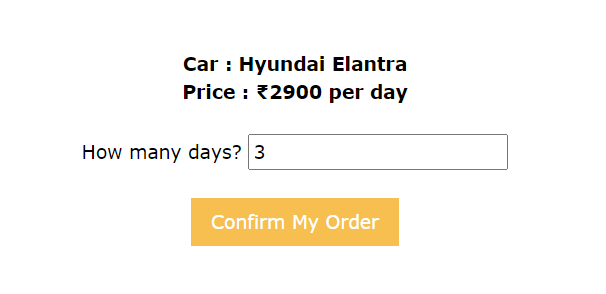
This page prompts the user to enter a city name and sends it to the database to get matching cars available in that city.



**Fig : 4.1.7: Search Results**

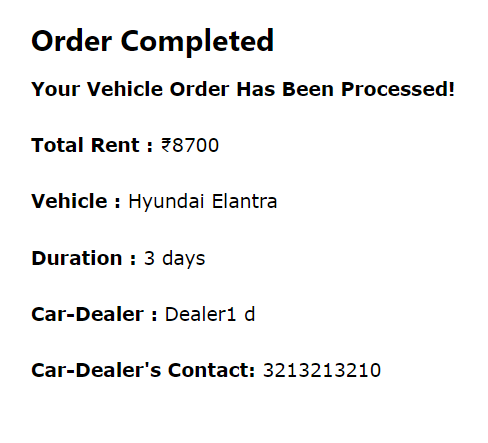
After the user enters a city name, the cars that match that city name and is available to rent will be sent from the database. That output will be listed in the search results page.

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.



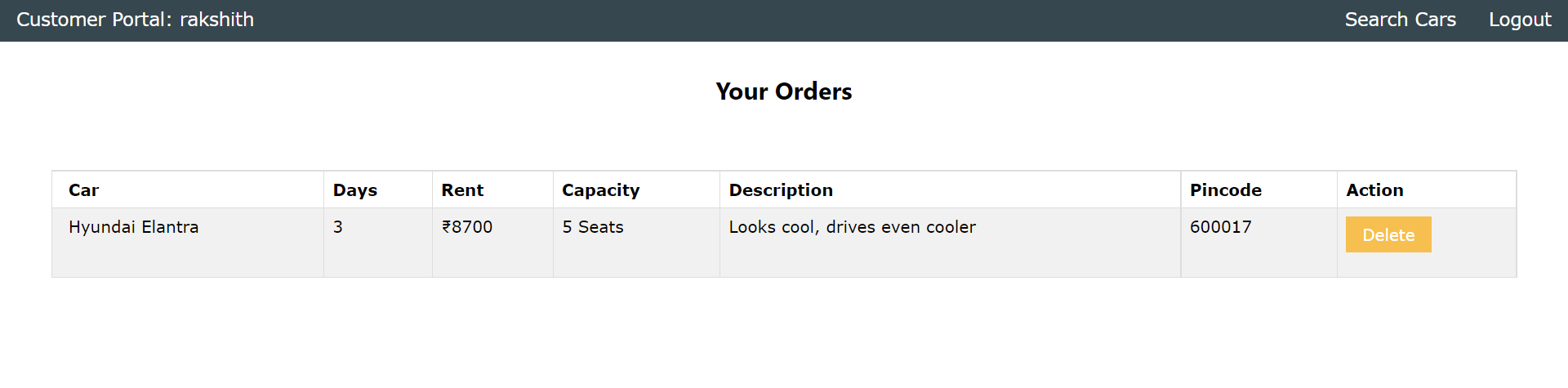
**Fig : 4.1.8: Rent page**

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.



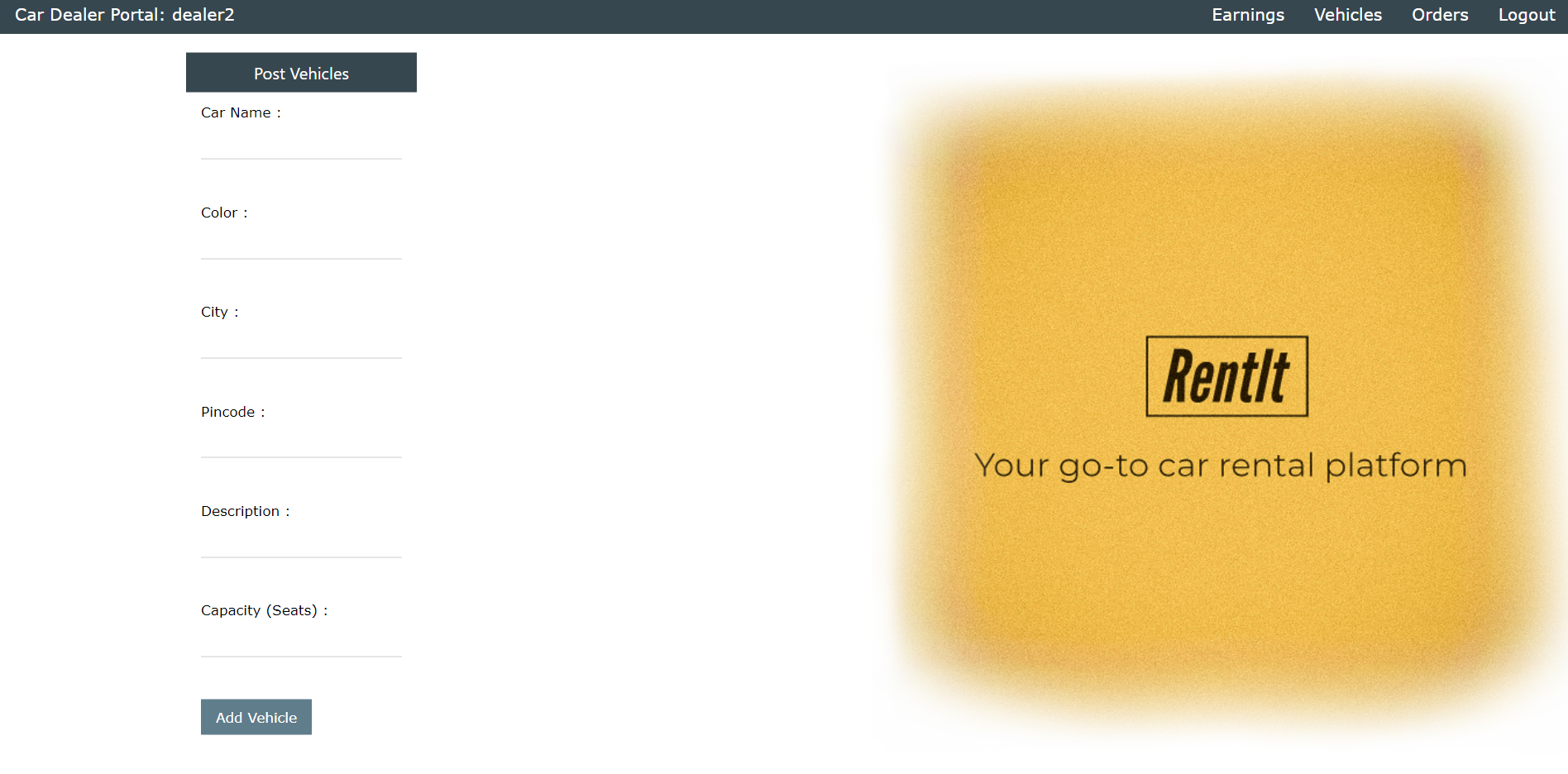
**Fig : 4.1.9: Confirmation page**

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.



**Fig : 4.1.10: Management page**

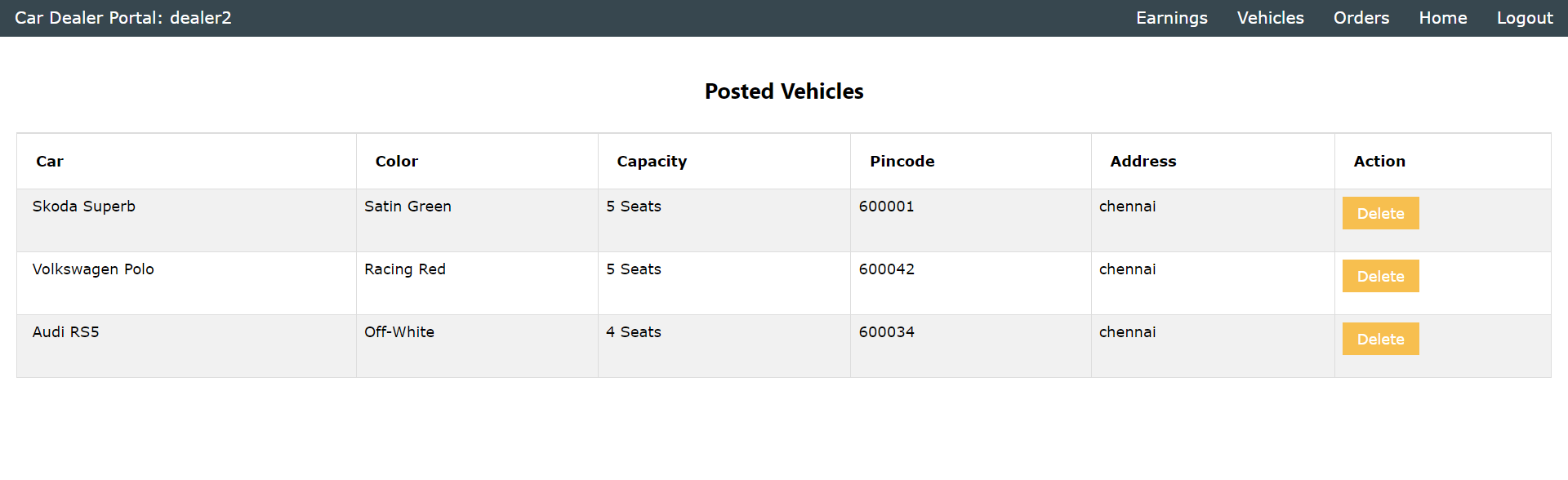
After the order is placed, the order details can be viewed again the Management page by clicking the ‘Manage’ button in the navigation bar above. This page allows the user to look into the details of their orders and also allows them to delete the booking they have made by clicking the ‘Delete’ button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



**Fig : 4.1.11: Dealer home page**

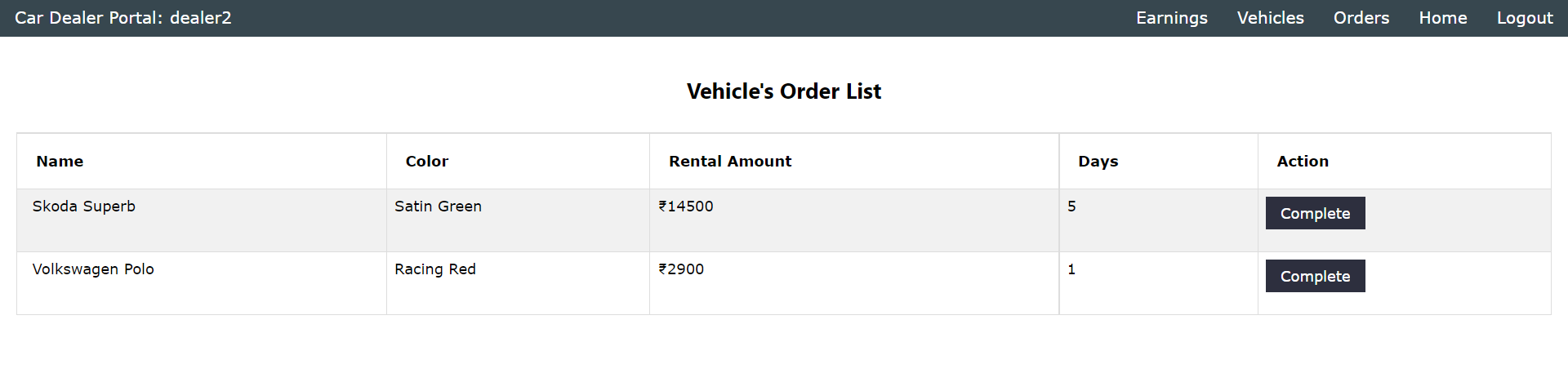
After successfully logging in with valid credentials of a dealer, the user will be redirected to the home page of the dealer portal. This page allows the dealer to add vehicle details and sumbit them to add a new car to the rental management system. It has 4 buttons in the navigation bar for the user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



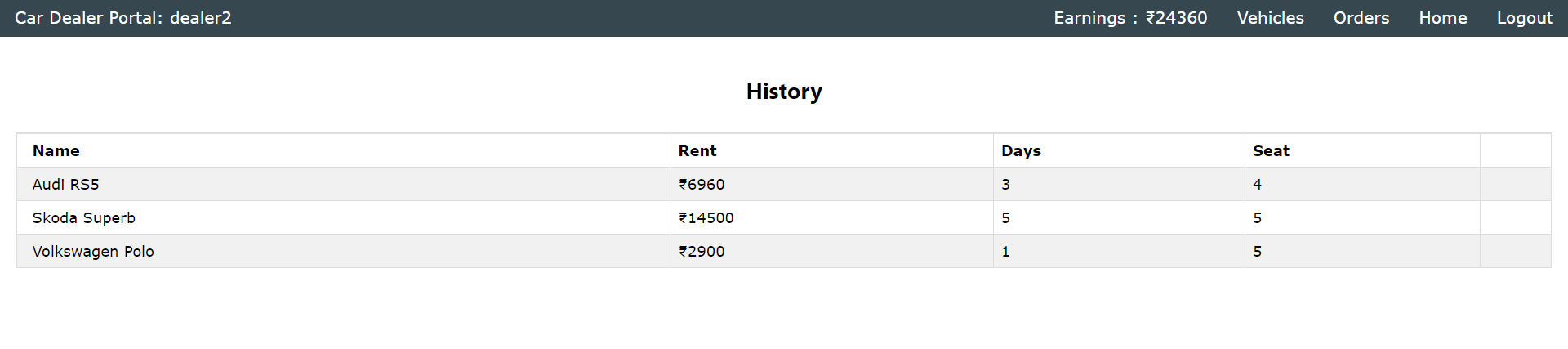
**Fig : 4.1.12: Vehicles page**

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.



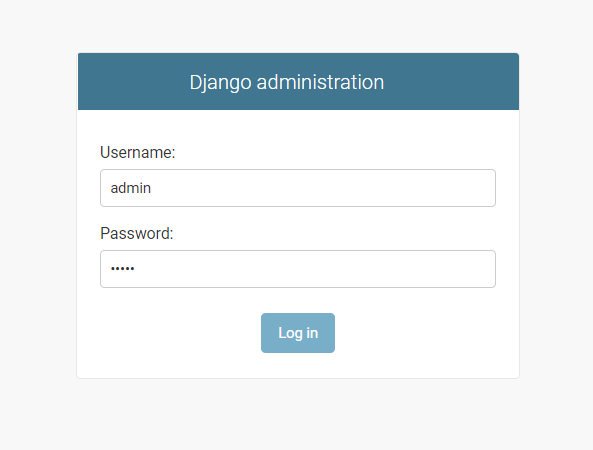
**Fig : 4.1.13: Order page**

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.



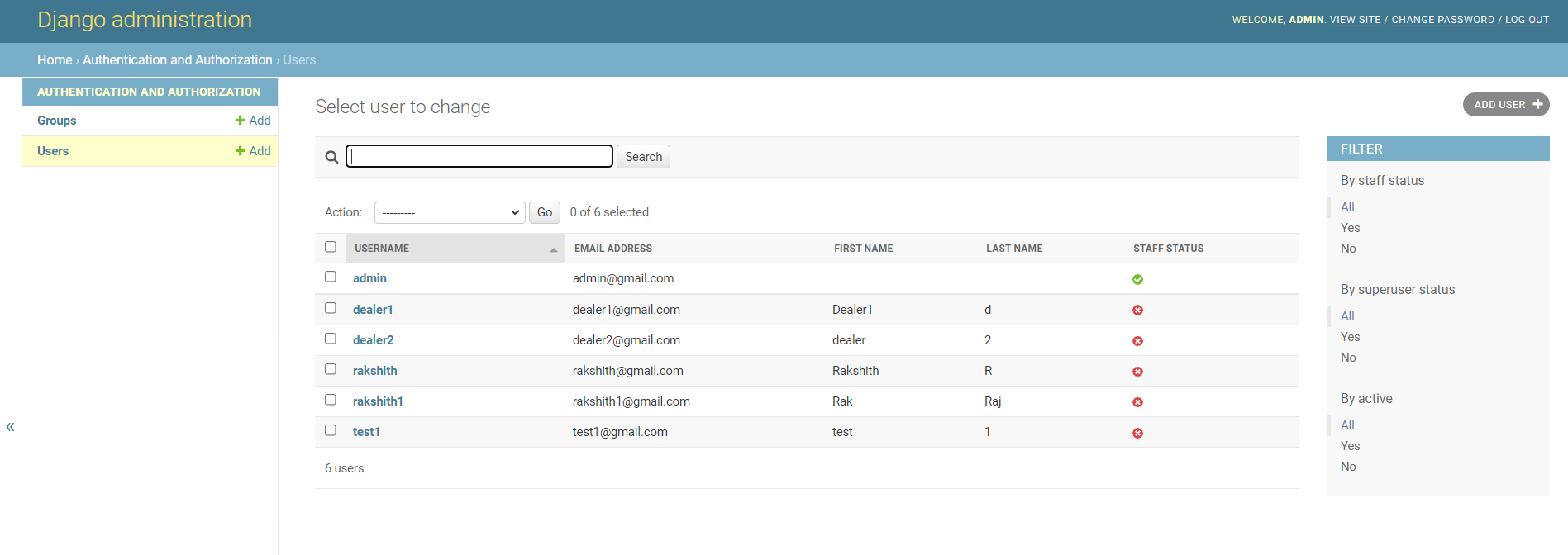
**Fig : 4.1.14: Earnings page**

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.



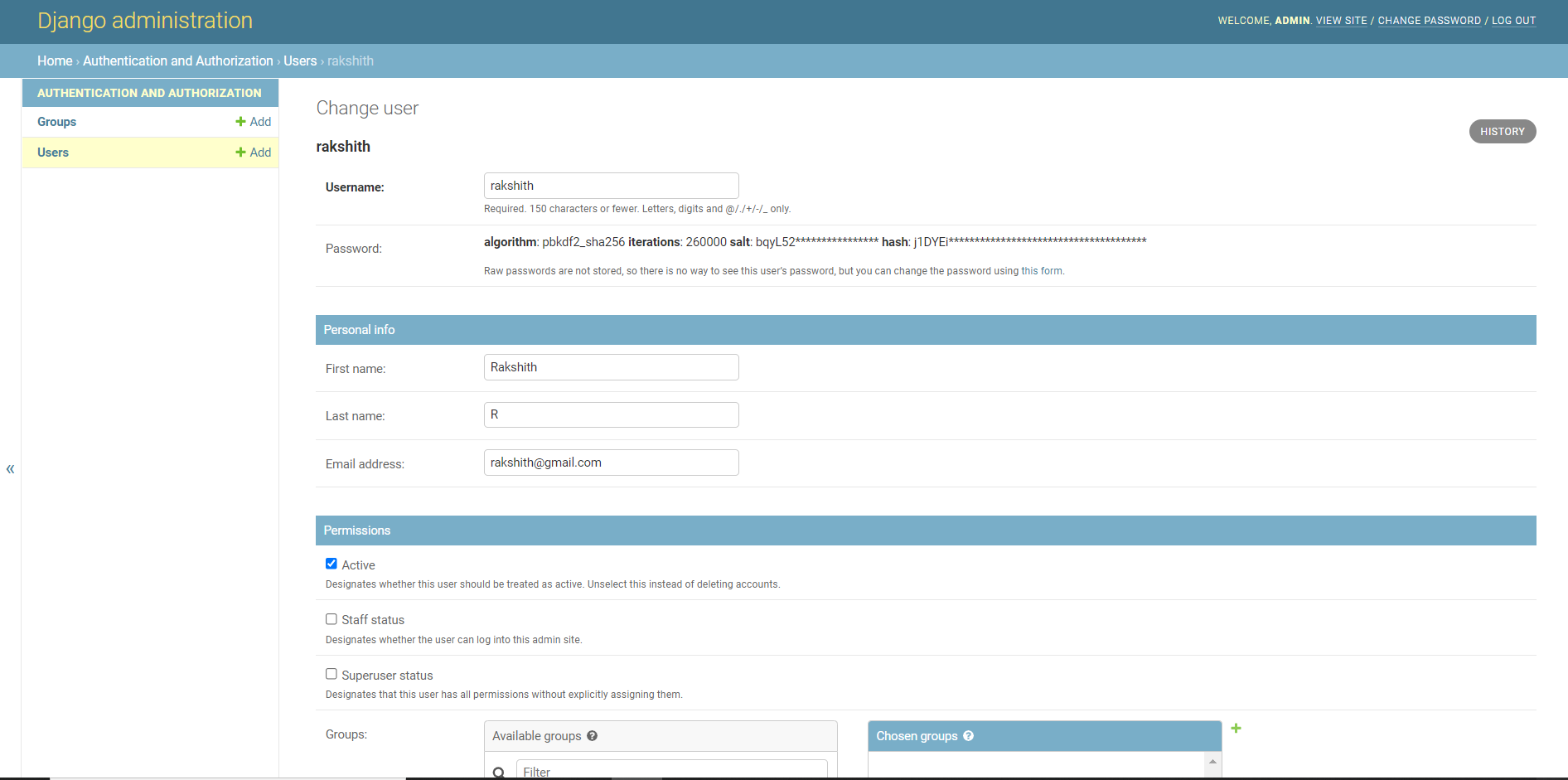
**Fig : 4.1.15: Admin login**

This page can be accessed by going to <http://127.0.0.1:8000/admin/> . This page promps the user to enter the username and password of an admin. This can be created through CMD in the project directory. Enter the following code to create a super user in Django.

**python manage.py createsuperuser**

**Fig : 4.1.16: Admin dashboard**

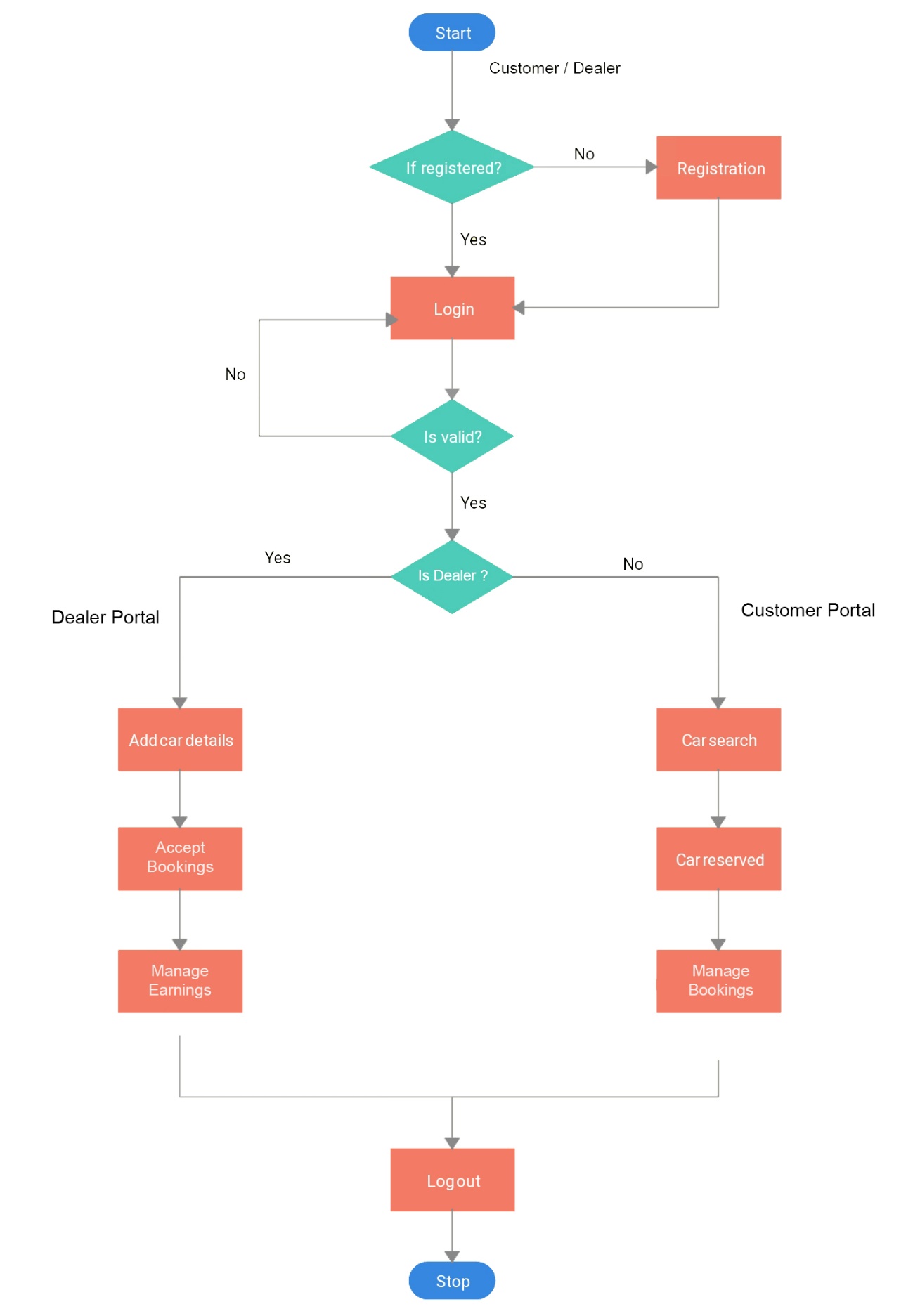
The admin dashboard shows all the active users in the database. It also displays the username, mail and their names. This page gives admin, the access to delete or add new users to the database.

****

**Fig : 4.1.17: Edit user**

This page further allows the admin to edit the user, including username, password, mail, name, status, etc. This page can be accesses by select any one user and clicking on them in the admin dashboard.

**Flow Diagram :**

****

**Fig : 4.1.18: Flow Diagram**

**4.2 SAMPLE SOURCE CODES**

**Costumer Portal:**

**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')

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)\*580

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))\*580\*(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)\*580

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

from car\_dealer\_portal.models import \*

# Create your models here.

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)

**urls.py**

from django.urls import path,include

from customer\_portal.views import \*

from django.conf.urls import url

urlpatterns = [

url(r'^index/$',index),

url(r'^login/$',login),

url(r'^auth/$',auth\_view),

url(r'^logout/$',logout\_view),

url(r'^register/$',register),

url(r'^registration/$',registration),

url(r'^search/$',search),

url(r'^search\_results/$',search\_results),

url(r'^rent/$',rent\_vehicle),

url(r'^confirmed/',confirm),

url(r'^manage/',manage),

url(r'^update/',update\_order),

url(r'^delete/',delete\_order),

]

**Dealer Portal:**

**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 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/')

**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)

**urls.py**

from django.urls import path,include

from car\_dealer\_portal.views import \*

from django.conf.urls import url

urlpatterns = [

url(r'^index/$',index),

url(r'^login/$',login),

url(r'^auth/$',auth\_view),

url(r'^logout/$',logout\_view),

url(r'^register/$',register),

url(r'^registration/$',registration),

url(r'^add\_vehicle/$',add\_vehicle),

url(r'^manage\_vehicles/$',manage\_vehicles),

url(r'^order\_list/$',order\_list),

url(r'^complete/$',complete),

url(r'^history/$',history),

url(r'^delete/$',delete),

]

**CHAPTER 5**

**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.

Through this project, the analysis on COVID-19 data has been performed successfully. The

analysis on this pandemic spread has been done and compared between different countries. The

analysis of confirmed cases, active cases, recovered cases and deaths are done separately to

give a clear look on how the virus is spreading, which countries are getting affected mostly and

how different countries are recovering.

**REFERENCES**

* <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/news-release/2021/11/25/2340991/28124/en/India-Car-Rental-Market-Report-2021-Market-is-Projected-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-the-alternatives/>
* <https://www.youtube.com/watch?v=PtQiiknWUcI>
* <https://www.educba.com/django-architecture/>