# Table of Contents

|  |  |
| --- | --- |
| **1** | **INTRODUCTION** |
| **2** | **SYSTEM ANALYSIS** |
|  | 2.1 Scope of the Project |
|  | 2.2 Aim of Project |
|  | 2.3 Hardware and Software Requirement |
| **3** | **SYSTEM DESIGN** |
|  | 3.1 System Perspective |
|  | 3.1.1 Architecture Diagram  3.1.2 Data flow Diagram |
|  | 3.1.3 ER Diagram  3.1.4 Use case Diagram |
| **4** | **IMPLEMENTATION.** |
|  | 4.1 Discussion of code segment |
|  | 4.2 Database Connection. |
|  | 4.3 Modules Explanation |
|  | 4.4 Output Design |
|  | 4.5 Software Testing |
| **5 DISCUSSION OF THE RESULTS.** | |
|  | 5.1 Screen Shots |
| **6 CONCLUSION AND FUTURE ENHANCEMENT** | |
|  | **REFERENCES** |

ABSTRACT

As the name specifies “Ride On Two Wheels” is web-application developed for People to reach their Destination in Rush Hours with our Bike Taxi Service (Ride On Two Wheels) instead of Public Transportation. We hope that this Service will be frequently used by People who need it. This particular project deals with the problems of daily Job Holders to reach their destination within the Time,

Identifying the drawbacks of the existing systems this leads us to design of web-application that will be compatible to all the Users Which is more user friendly and more GUI oriented.

It likewise gives an occasion for individuals with a bike (with a proper driving permit) to turn into a rider on the platform and bring in extra additional cash in their extra time. Their definitive vision is to innovate consistent commuting for all, directly at your doorstep.

It is likewise exceptionally safe to travel, even in the evening as everything can be followed on the web. Finally, it is only a two-wheeler taxi where a rider will get you from the required point and drop to the required point.

Chapter 1

**INTRODUCTION**

Ride On Two Wheels is a startup which provides bike taxi service in some of the Districts in the Karnataka. We offers unique features where one can ride solo(Rent a Bike), book a quick Ride(Emergency Ride), Book a Ride with driver. Ride On Two Wheels is the cheapest, fastest, efficient and most convenient way to commute in these cities always packed with traffic.

To clarify the time issue trapped with a step-by-step drive to the workplace, bike taxis have increased over the past years in territories of cities in Karnataka

Ride On Two Wheels is the bike taxi service provider, which connects travellers to drivers for short distance drive. It is fundamentally a bike taxi form of Uber, a lot less expensive and more helpful in rush hour traffic as in Larger Cities. Most individuals need every day travel simpler, more secure, and moderate, there are more motivations to pick Ride On Two Wheels-Rides and reach on time with low fares and efficiently.

Regardless, no couple of such bike taxi start-ups have did suffer, only some of the new organisations in the bike taxi district are set up to suffer and be successfull.

# Chapter 2

**SYSTEM ANALYSIS**

2.1 Scope of the Project:

* An effective Bike Taxi, Rental system is integral to providing quality customer service.
* It helps the customer to reach their destination at the given time safely.
* Customer can contact us through filling the form given in the Contact-Us page when things are not working properly.
* Emergency Ride part will help the customer when the customer is in hurry to reach the destination within the constraints of time.

2.2 Aim of Project:

The aim of project is to make last mile connectivity to the Customers to Tackle the issues that they face every morning in the Rush Hours With Our Services. We offer this option to the People instead of Car and A Public Transportation.

2.3 Hardware and Software Requirement

2.3.1 Hardware Configuration

|  |  |
| --- | --- |
| **Number** | **Description** |
| 1.-Disk | PC with 2 GB |
| 2. RAM | 256 MB |

2.3.2 Software Configuration

|  |  |
| --- | --- |
| **Number** | **Description** |
| 1.Operating System | Windows, Linux |
| 2.Database | Sqlite3 |
| 3. Programming Languages | HTML(with CSS), Python, Django |

Chapter 3

**SYSTEM DESIGN**

3.1 System Perspective:

3.1.1 Architecture Diagram

1. A user enters URL into browser

2. Browser generates an HTTP request to the appropriate web serve

3. The web server interprets it and forwards it to the corresponding JSP.

4. The JSP processes the request, generates the output and sends it back to the web server.

5. The web server sends the response back to the client. The browser then displays it on the

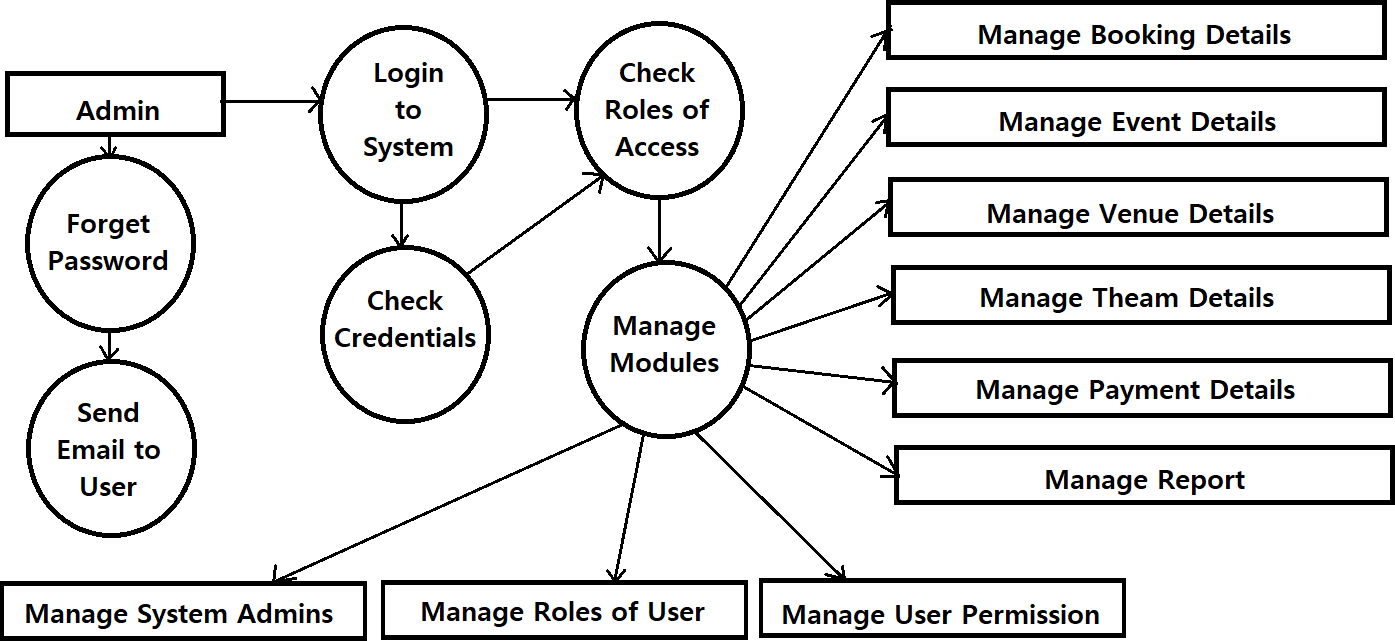
screen.

3.1.2 Data flow Diagram

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether the flow processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).





3.1.3 ER-Diagram

In software engineering, an Entity-Relationship Model (ERM) is an abstract and conceptual presentation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in atop-down fashion.

Entity:- Any real-world object can be represented as an entity about which data can be stored in a database.

ENTITY:

Attribute:- Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc.

ATTRIBUTE:

Relationships:- A relationship is defined as bond or attachment between 2 or more entities.

RELATIONSHIP:

A diamond is used to symbolically represent a relationship in the e-r diagram.

Chapter 4

**IMPLEMENATION**

• HTML:

HTML means Hypertext mark-up language .HTML is method of describing the format of document, which allow them tobe viewed on the computer screen .HTML documents are dis-played by web browser, programs which can navigateacross networks and display a wide variety of types of information.HTML page can be developed to be a simple text or to be complex multimedia containing sound, moving ,images, virtual reality, and java applets. The global publishing format of internet is HTML. It allows authors to use not only

text but also format that text with heading, lists, and tables .Readers can access the pages of information from anywhere in the world at a click of mousebutton.HTML pages can also be used for entering the data as a front end for commercial transactions.

• DATABASE : Sqlite3

SQLite is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It’s also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

The sqlite3 module was written by Gerhard Häring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249, and requires SQLite 3

• SERVER : WSGI (web server gateway interface)

The Web Server Gateway Interface (WSGI) is a standard interface between web server software and web applications written in Python. Having a standard interface makes it easy to use an application that supports WSGI with a number of different web servers.

Only authors of web servers and programming frameworks need to know every detail and corner case of the WSGI design. You don’t need to understand every detail of WSGI just to install a WSGI application or to write a web application using an existing framework.

wsgiref is a reference implementation of the WSGI specification that can be used to add WSGI support to a web server or framework. It provides utilities for manipulating WSGI environment variables and response headers, base classes for implementing WSGI servers, a demo HTTP server that serves WSGI applications, and a validation tool that checks WSGI servers and applications for conformance to the WSGI specification

4.1 Discussion of code segment

* User Login:

{% extends 'base.html' %}

{% load static %}

{% block content %}

<div class="container" style="margin: 5%;">

<h1 class="animate\_\_animated animate\_\_bounce text-center" style="font-family: 'IM Fell DW Pica', serif; color: #46B2E0;" >Login Here</h1>

<section class="vh-100">

<div class="container py-5 h-100" style="background-color:antiquewhite">

<div class="row d-flex align-items-center justify-content-center h-100">

<div class="col-md-8 col-lg-7 col-xl-6">

<img src="https://mdbcdn.b-cdn.net/img/Photos/new-templates/bootstrap-login-form/draw2.svg"

class="img-fluid" alt="Phone image">

</div>

<div class="col-md-7 col-lg-5 col-xl-5 offset-xl-1" >

<form method="POST" action="{% url 'login' %}">

{% csrf\_token %}

<!-- Email input -->

<div class="form-outline mb-4">

<input name="username" class="form-control form-control-lg" />

<label class="form-label">User Name</label>

</div>

<!-- Password input -->

<div class="form-outline mb-4">

<input type="password" name="password" class="form-control form-control-lg" />

<label class="form-label">Password</label>

</div>

<!-- Submit button -->

<button type="submit" class="btn btn-warning btn-lg btn-block">Sign in</button>

<div class="text-center" style="margin: 20px;">

OR

</div>

<a class="btn btn-primary btn-lg btn-block" style="background-color:#4285F4 " href="#!"

role="button">

<i class="fa-brands fa-google me-2"></i>Continue with Google

</a>

</form>

</div>

</div>

</div>

</section>

</div>

{% endblock %}

* Main page Views

from django.shortcuts import render,redirect

from django.shortcuts import get\_object\_or\_404

from .models import HomeCard,HomeCurosol,Services,Contact,Careers,AboutJumbo,AboutJumbo2,AboutJumbo3

from django.contrib import messages

# Create your views here.

def homePage(request):

Homecard = HomeCard.objects.all()

data ={

'Homecard':Homecard

}

return render(request ,'webpages/home.html',data)

def homeCurosol(request):

HomeCurosol = HomeCurosol.objects.all()

data ={

'HomeCurosol':HomeCurosol

}

return render(request ,'webpages/home.html',data)

def aboutPage(request):

'AboutJumbo' == AboutJumbo.objects.all()

data ={

'AboutJumbo':AboutJumbo

}

'AboutJumbo2' == AboutJumbo2.objects.all()

data ={

'AboutJumbo2':AboutJumbo2

}

'AboutJumbo3' == AboutJumbo3.objects.all()

data ={

'AboutJumbo3':AboutJumbo3

}

return render(request ,'webpages/about.html')

def servicesPage(request):

services = Services.objects.all()

data ={

'services':services

}

return render(request ,'webpages/services.html',data)

def careersPage(request):

if request.method == 'POST':

fname = request.POST.get('fname')

lname = request.POST.get('lname')

email = request.POST.get('email')

caddress = request.POST.get('caddress')

paddress = request.POST.get('paddress')

phoneno = request.POST.get('phoneno')

aadhar = request.POST.get('aadhar')

dlno = request.POST.get('dlno')

rcno = request.POST.get('rcno')

careers = Careers(fname=fname,lname=lname,email=email, caddress=caddress, paddress=paddress,phoneno=phoneno,aadhar=aadhar,dlno=dlno, rcno=rcno)

careers.save()

messages.success(request, 'We Will Confirm You Within 24 hours, Kindly Wait!')

return redirect('home')

return render(request ,'webpages/careers.html')

def contactPage(request):

if request.method == 'POST':

name = request.POST['name']

email = request.POST['email']

address = request.POST['address']

phone = request.POST['phone']

message = request.POST.get('message', False)

contact = Contact(name=name,email=email, address=address, phone=phone,message=message)

contact.save()

messages.success(request, 'Thanks for reaching out!')

return redirect('home')

return render(request ,'webpages/contact.html')

* Login page Views

from django.shortcuts import render,redirect

from django.contrib.auth.models import User

from .models import RentingService,BookARide,Emergency,Careers

from django.contrib import messages, auth

from django.contrib.auth.decorators import login\_required

from django.contrib.auth import logout

from django.views.decorators.csrf import csrf\_exempt

import razorpay

# Create your views here.

@login\_required(login\_url='login')

def payment(request):

if request.method == 'POST':

amount = 500

order\_currency = 'INR'

client = razorpay.Client(

auth=('rzp\_test\_i2hZmqLyieAh7Q','zsmy5ehCv6X6igPEeDRvRj6w'))

payment = client.order.create({'amount':amount, 'currency':'INR', 'payment\_capture':'1'})

return render(request,'rider/payment.html')

@csrf\_exempt

def success(request):

return render(request, "rider/success.html")

def login(request):

if request.method=='POST':

username = request.POST['username']

password = request.POST['password']

user = auth.authenticate(username=username, password=password)

if user is not None:

auth.login(request, user)

messages.warning(request,'You are logged IN')

return redirect('home')

else:

messages.warning(request,'Invalid Credentials')

return redirect('login')

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

def signup(request):

if request.method == 'POST':

firstname = request.POST['firstname']

lastname = request.POST['lastname']

username = request.POST['username']

email = request.POST['email']

password = request.POST['password']

confirm\_password = request.POST['confirm\_password']

if password == confirm\_password:

if User.objects.filter(username=username).exists():

messages.warning(request, 'Username exists')

return redirect('signup')

else:

if User.objects.filter(email=email).exists():

messages.warning(request, 'Email already exists')

return redirect('signup')

else:

user = User.objects.create\_user(first\_name=firstname, last\_name=lastname,

username=username, email=email,

password=password)

user.save()

messages.success(request, 'Account Created Successfully ')

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

else:

messages.warning(request, 'Password do not match')

return redirect('signup')

return render(request,'rider/signup.html')

def logout\_user(request):

logout(request)

return redirect('home')

@login\_required(login\_url='login')

def bookRider(request):

if request.method == 'POST':

firstname = request.POST['firstname']

lastname = request.POST['lastname']

email = request.POST['email']

phone = request.POST.get('phone')

gender = request.POST['gender']

city = request.POST['city']

pick\_up = request.POST['pick\_up']

drop\_addr = request.POST['drop\_addr']

bookRider = BookARide(firstname=firstname,lastname=lastname,email=email,phone=phone,gender=gender,city=city,pick\_up=pick\_up,drop\_addr=drop\_addr)

bookRider.save()

messages.success(request, 'Thank You for registration')

return redirect('payment')

return render(request,'rider/bookaride.html')

@login\_required(login\_url='login')

def emergency(request):

emergency = Emergency.objects.all()

data={

'emergency' : emergency,

}

if request.method == 'POST':

firstname = request.POST['firstname']

lastname = request.POST['lastname']

email = request.POST['email']

phone = request.POST['phone']

gender = request.POST.get('gender')

city = request.POST['city']

pick\_up = request.POST['pick\_up']

drop\_addr = request.POST['drop\_addr']

emergency = Emergency(firstname=firstname,lastname=lastname,email=email,phone=phone,gender=gender,city=city,pick\_up=pick\_up,drop\_addr=drop\_addr)

emergency.save()

messages.success(request, 'Thank You for registration')

return redirect('payment')

return render(request,'rider/emergency.html',data)

@login\_required(login\_url='login')

def rentBike(request):

if request.method == 'POST':

firstname = request.POST['firstname']

lastname = request.POST['lastname']

email = request.POST['email']

phone = request.POST.get('phone')

gender = request.POST['gender']

Occupation = request.POST.get('Occupation')

driving\_lic\_no = request.POST.get('driving\_lic\_no')

city = request.POST.get('city')

curr\_address = request.POST.get('curr\_address')

perm\_address = request.POST.get('perm\_address')

rent\_hrs = request.POST.get('rent\_hrs')

rent\_chars = request.POST.get('rent\_chars')

rentBike = RentingService(firstname=firstname,lastname=lastname,email=email,phone=phone,gender=gender,Occupation=Occupation,driving\_lic\_no=driving\_lic\_no,city=city,curr\_address=curr\_address,perm\_address=perm\_address,rent\_hrs=rent\_hrs,rent\_chars=rent\_chars)

rentBike.save()

messages.success(request, 'Thank You for registration')

return redirect('payment')

return render(request,'rider/rentabike.html')

def careersPage(request):

if request.method == 'POST':

fname = request.POST['fname']

lname = request.POST['lname']

email = request.POST['email']

caddress = request.POST.get('caddress')

paddress = request.POST.get('paddress')

phoneno = request.POST.get('phoneno')

aadhar = request.POST['aadhar']

dlno = request.POST.get('dlno')

rcno = request.POST.get('rcno')

careers = Careers(fname=fname,lname=lname,email=email, caddress=caddress, paddress=paddress,phoneno=phoneno,aadhar=aadhar,dlno=dlno, rcno=rcno)

careers.save()

messages.success(request, 'We Will Confirm You Within 24 hours, Kindly Wait!')

return redirect('home')

return render(request ,'webpages/careers.html')

4.2 Database connection

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

4.3 Modules explanation

**Administrator**

1. The Administrator can manage all the user and their Ride Data Entry in the Database.

2. He can access all the info and change the database of the existing user.

3. He can go through the Complaints or Queries posted by the user using Contact Page and solve them .

A process of converting user originated inputs to a computer-based format. Input design is an important part of development process since inaccurate input data are the most common cause of errors in data processing. Erroneous entries can be controlled by input design. It consists of developing specifications and procedures for entering data into a system and must be in simple format**.** The goal of input data design is to make data entry as easy, logical and free from errors as possible. In input data design, we design the source document that capture the data and then select the media used to enter them into the computer.

There are two major approaches for entering data in to the computer. They are-

* Menus.
* Dialog Boxes.

**Menus**

A menu is a selection list that simplifies computer data access or entry. Instead of remembering what to enter, the user chooses from a list of options. A menu limits a user choice of response but reduce the chances for error in data entry.

**Dialog Box**

Dialog boxes are windows and these windows are mainly popup, which appear in response to certain conditions that occur when a program is run. It allows the display of bitmaps and pictures. It can have various controls like buttons, text boxes, list boxes and combo boxes. Using these controls we can make a ‘dialog’ with the program.

The proposed system has three major inputs. They are Machine Registration, Machine Scheduling and Request Form

4.5 Software Testing

4.5.1 Testing:

For testing of the system “online counselling website” and for its proper working and reliability, two phases have to be incorporated:

1. Alpha testing
2. Beta testing

* Alpha Testing:

Some of the things that could be tested in the development phase of the product can be given as:

1. Check for the queries being written to retrieve information from the database, whether they produce

appropriate results/output or not.

1. Check that the data integrity and atomicity is maintained by performing some of the transaction and see if they successfully complete.
2. Check for the load on the server and if the server is able to withstand the given amount of load which is there on it.
3. Check for the PHP if they produce forms and web pages in the desired manner.
4. Check for if the parameters are being passed from one web page to another where it is required. As in the case where the user enters the information on a web form, which are then passed to the application sever to retrieve from here.

* Beta Testing:

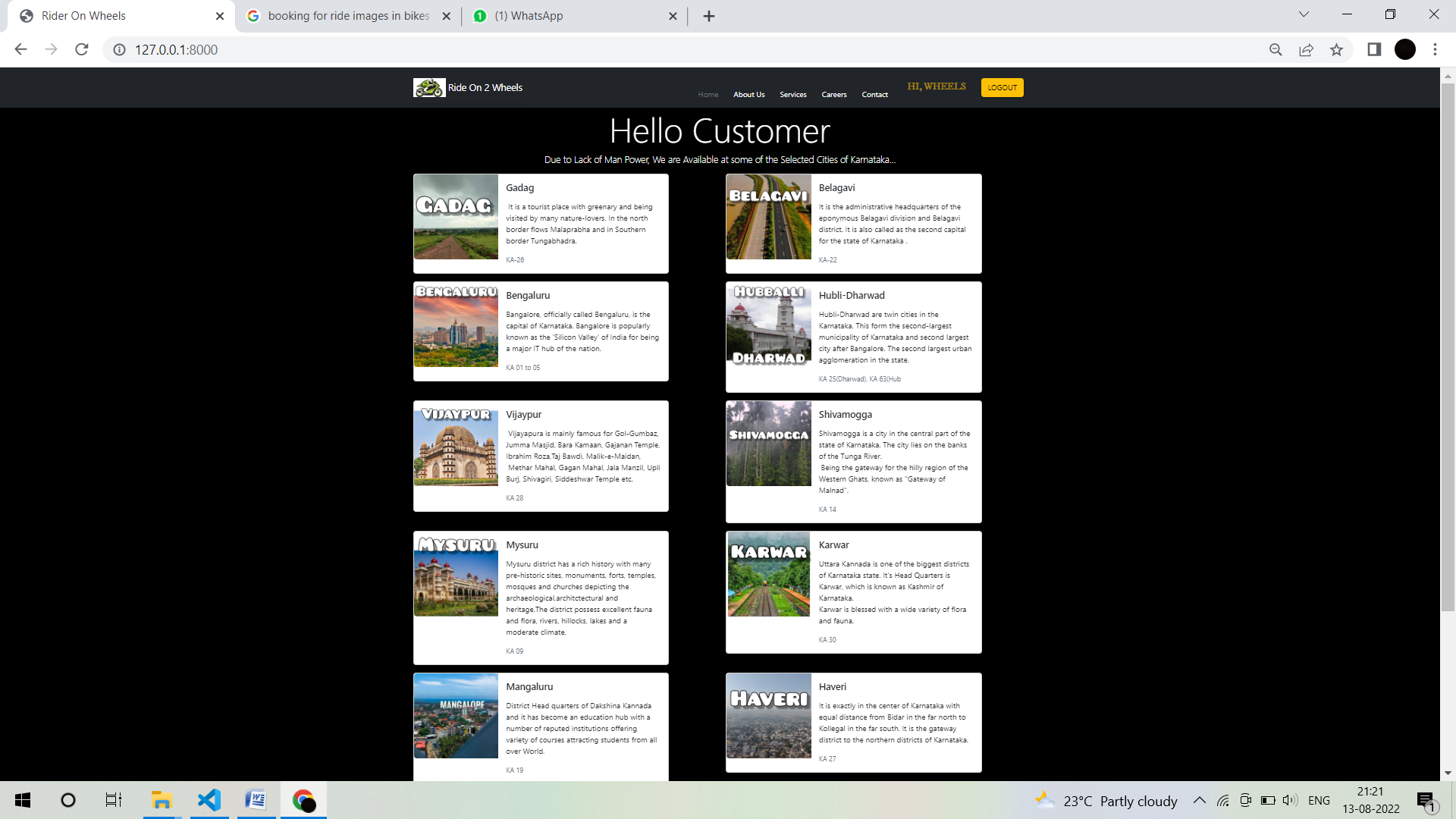
1. Check for the validity of the system and the result returned by the queries hitting at the database. The way is enter the user id and the password and see the web page being produce at the o/p. check this and with the information available on paper.
2. Same can be done for different actor of the system as college or administrator. Check this and verify this with the information available on paper.
3. Check that any updation to the database is being properly incorporated at suitable place.

Chapter 5

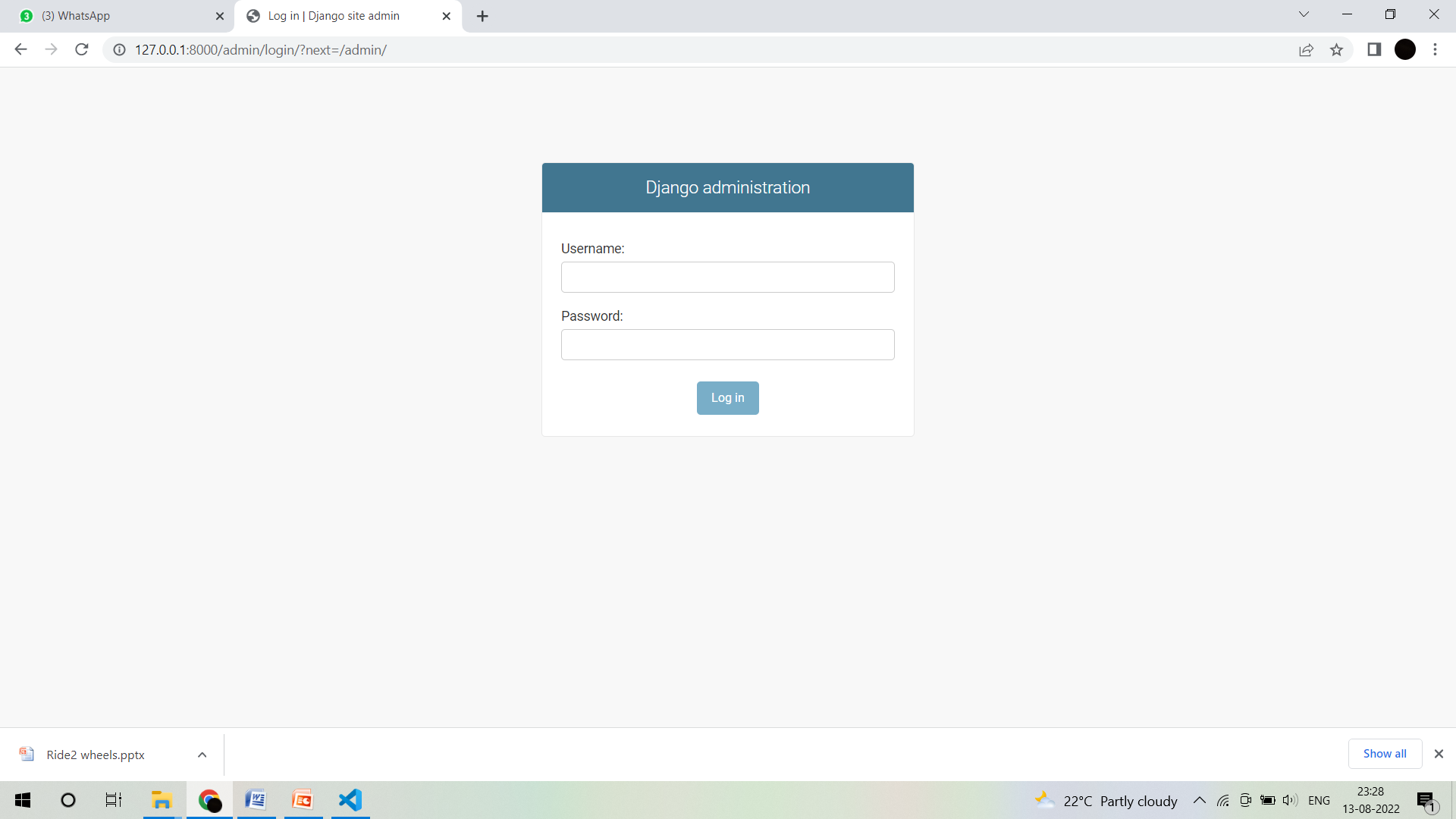
**DISCUSSION OF THE RESULTS**

* 1. Screen Shots

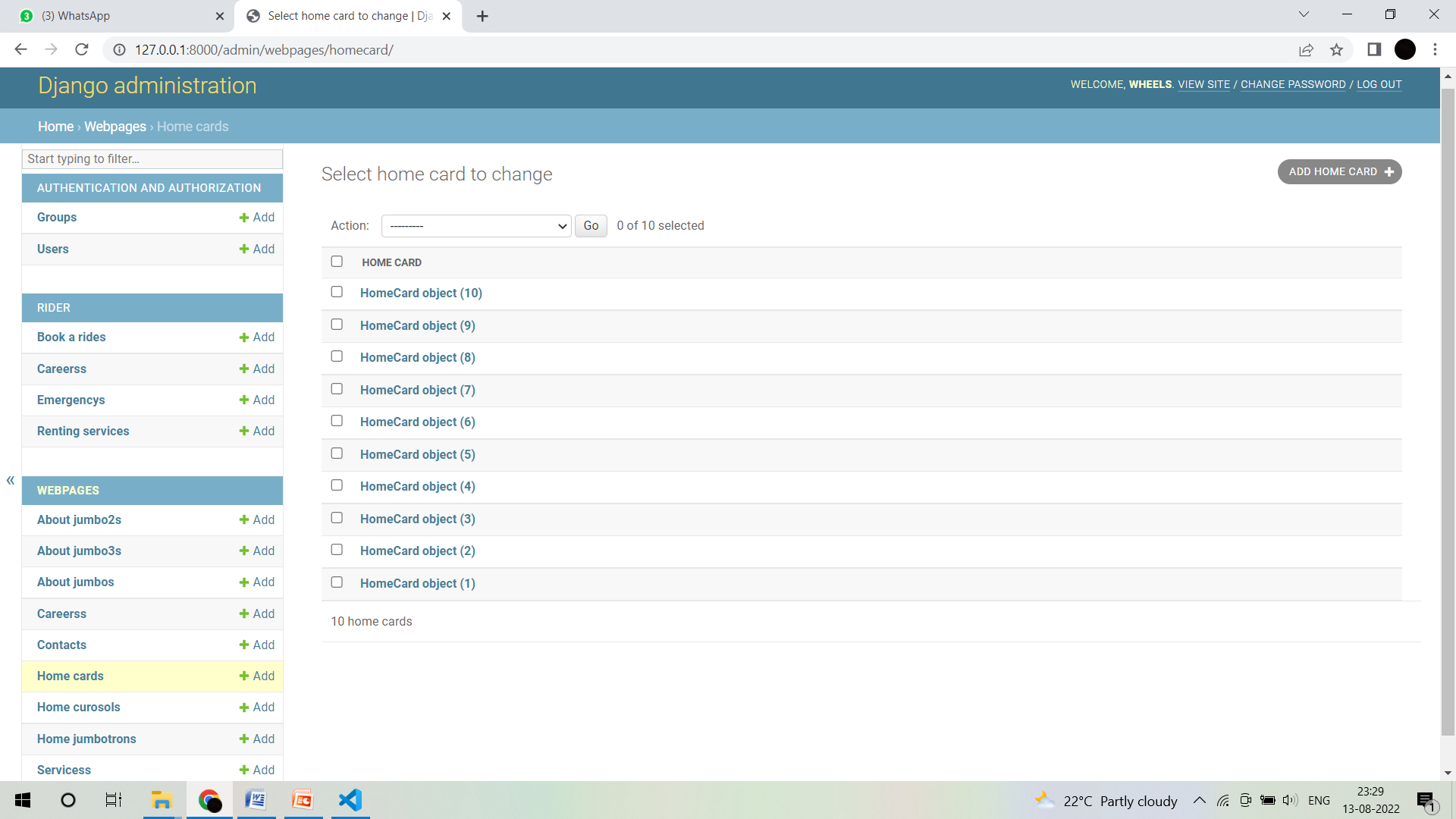
Home Page:



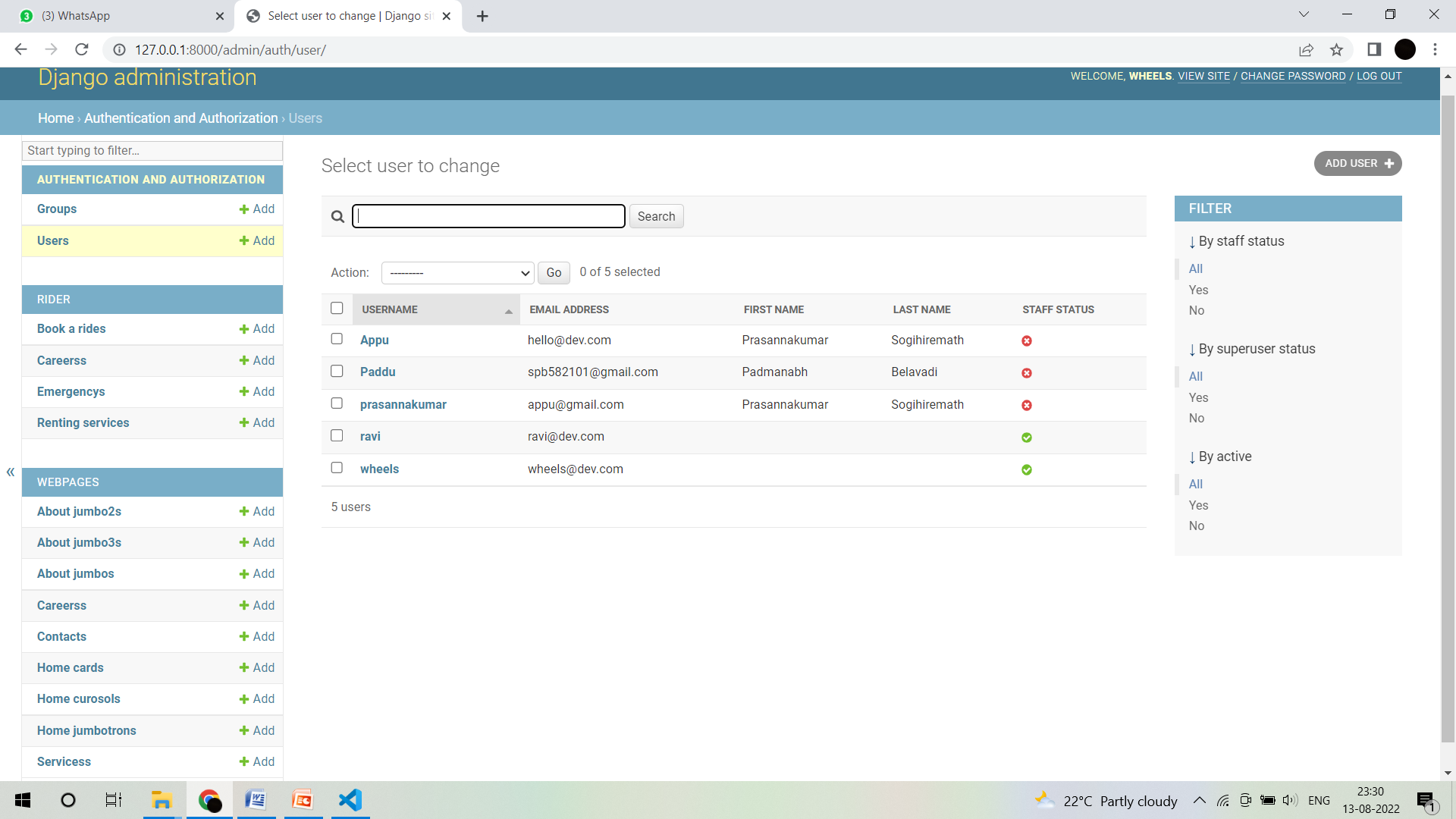
Admin login page:



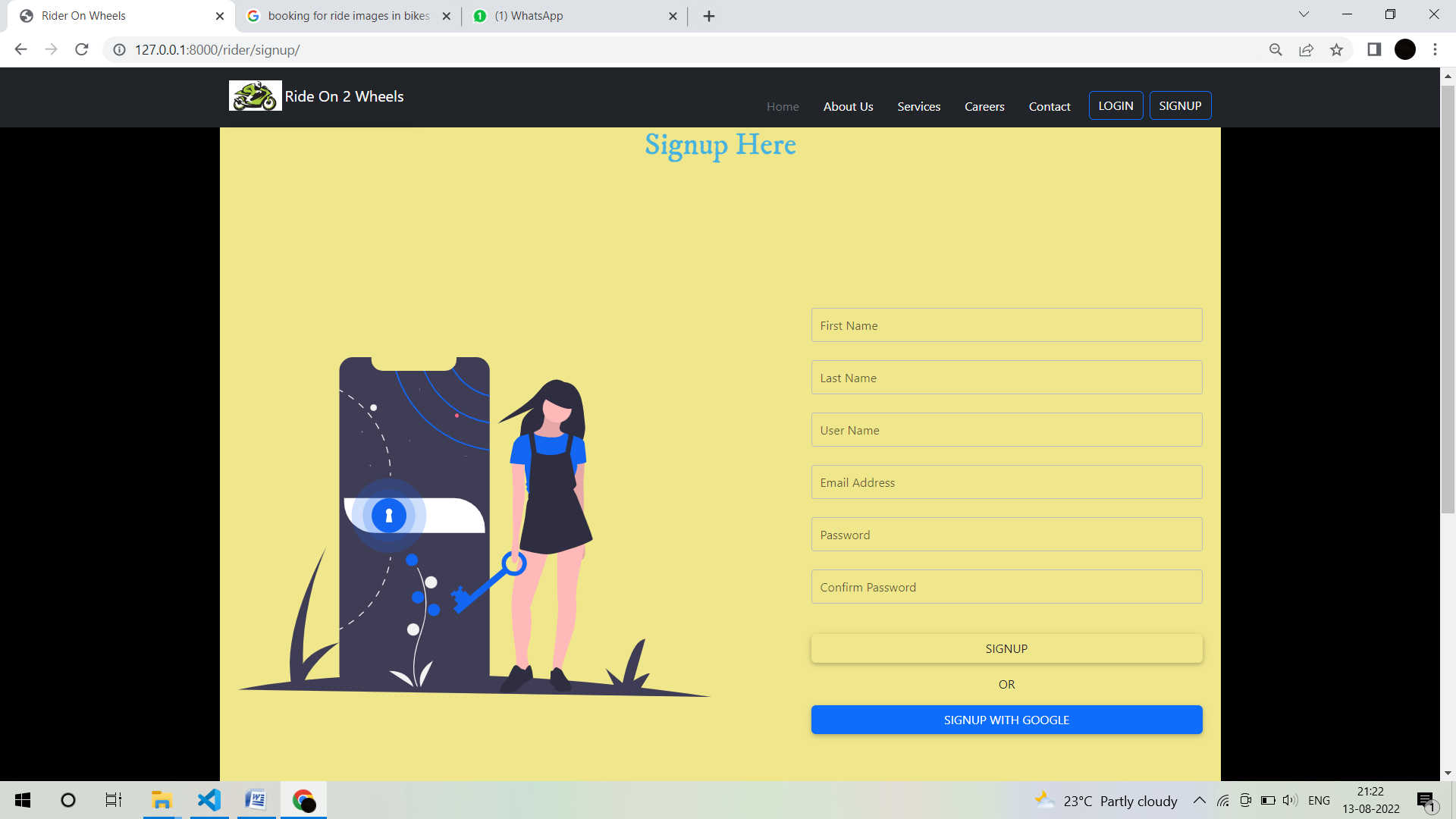
Manage User Page:



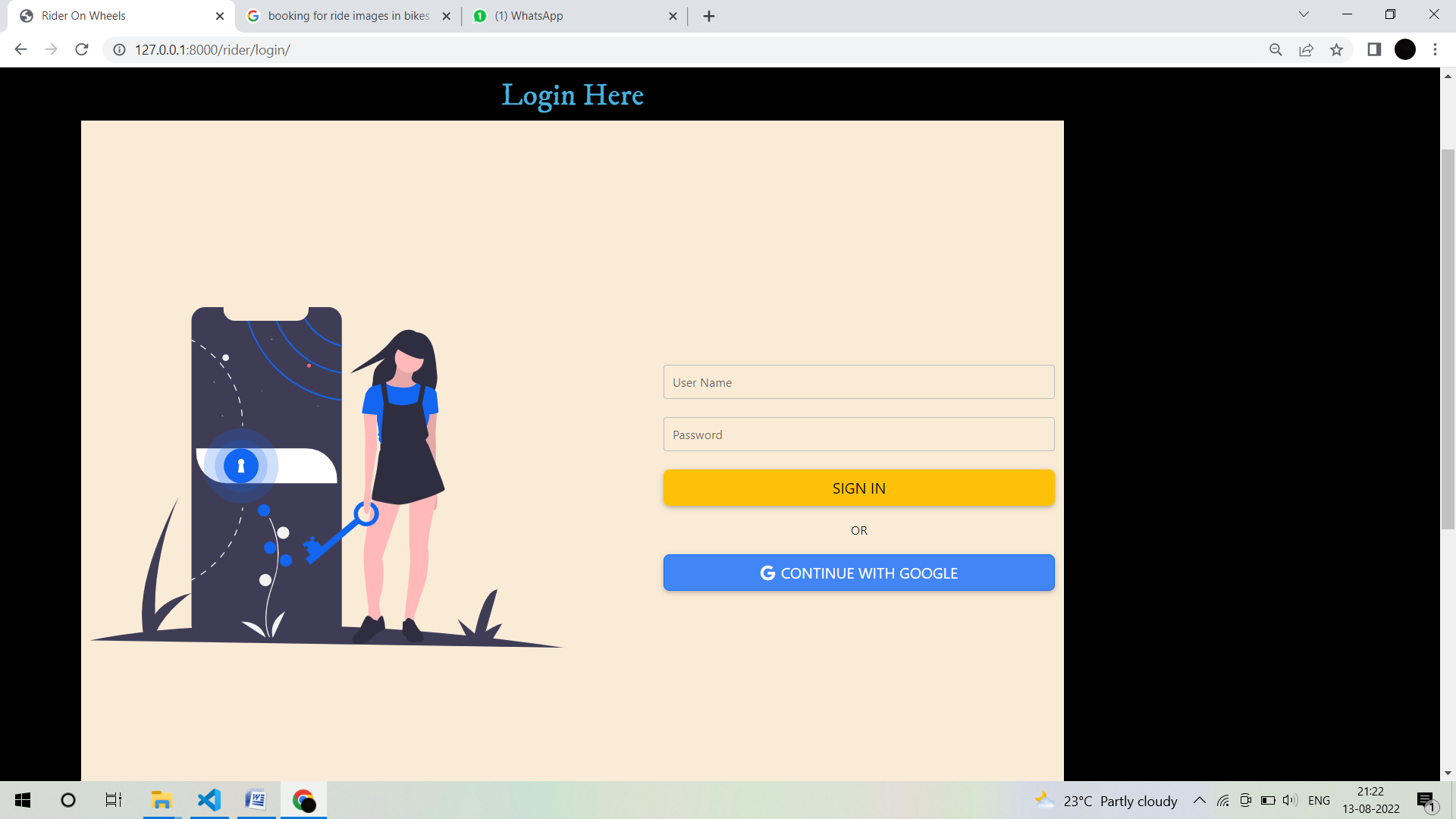
User login manage page:



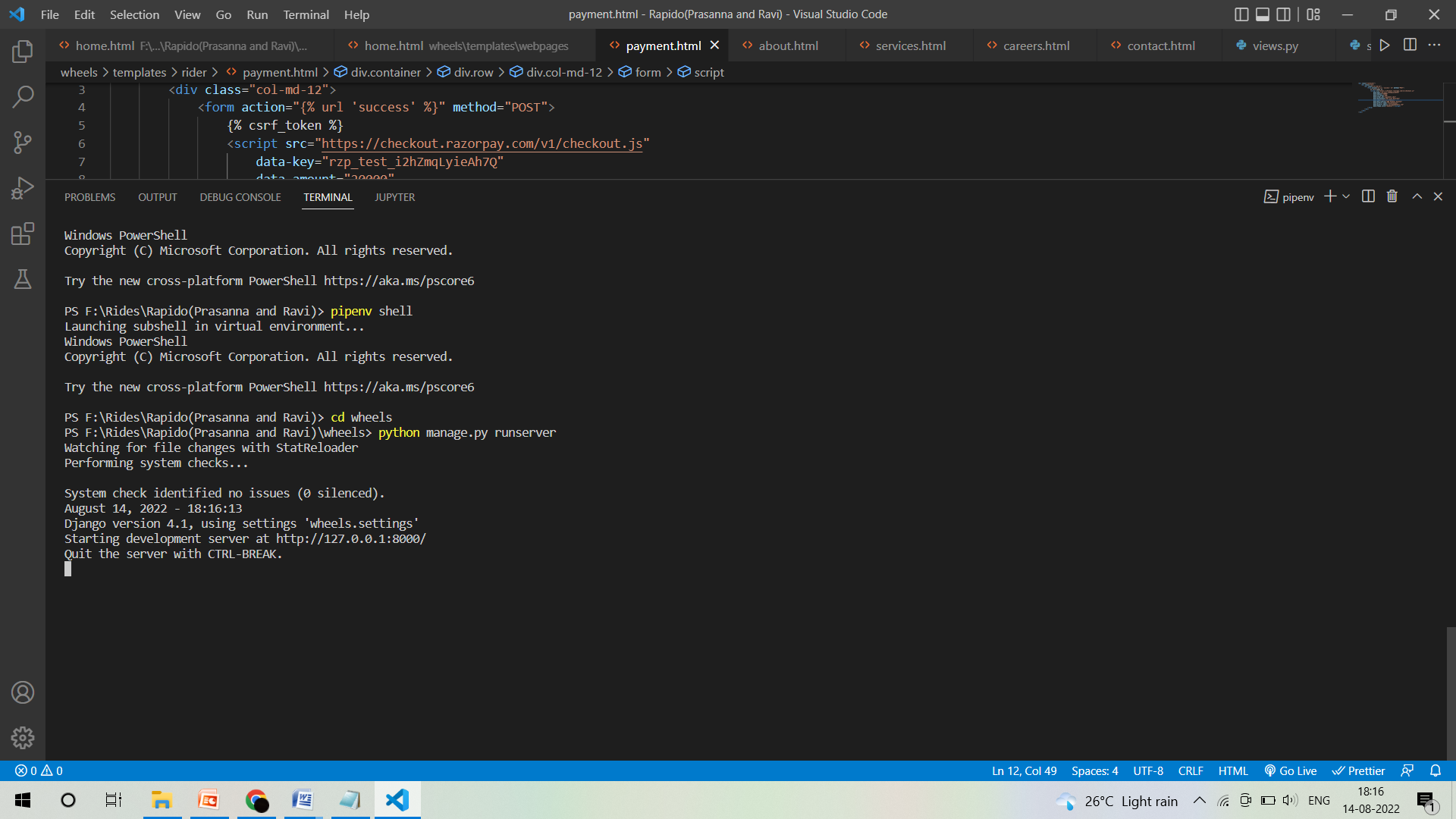
User Registration Page:



User Login:



Running the Server:

Chapter 6

**CONCLUSION**

We can conclude that it is a highly economical service when compared to public transport and sometimes we may need to wait for the drivers depending upon the availability and found that it is not very easy to cross the targets they fix for Riders. However, anything is possible with your efforts.

To conclude the description about the project : The project, developed using HTML, Bootstrap, Python and Django is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.

The expanded functionality of today’s Web Application requires an appropriate approach towards Web Application development. This Ride on 2 Wheels web application is designed for people who want transportation support for their any kind of work. Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

**REFERENCES**

1.https://cdnjs.com/

2.https://mdbootstrap.com/

3.https://ckeditor.com/

4.https://pypi.org/

5.https://pypi.org/project/djangocms-admin-style/

6.https://pypi.org/project/pipenv/

7.https://docs.djangoproject.com/en/4.1/

8.https://kenwheeler.github.io/slick/

9.https://razorpay.com/

10.https://regex101.com/