Weather App

Mini Project Report

Open Source Technology Lab

Submitted by

Roll No.	Batch Sr. No.	Name
18CE1044	A3-16	Kartik khanna
18CE1034	A3-14	Chetan khairnar
18CE1009	A3-10	Shivam Kendre
18CE2020	A3-17	Raj Sarode

Guided by

Sumithra T. v

Department of Computer Engineering

Ramrao Adik Institute of Technology

Dr .D.Y. Patil Vidyanagr,sector 7, Nerul, Navi Mumbai 400706.

(Affiliated to university of Mumbai)

April-2020

Index

ABSTARCT

Weather forecasting is an important application in meteorology and has been one of the most scientifically and technologically challenging problems around the world.

In this project on weather app is to be implemented to make the task of getting accurate climate conditions better while providing a better user friendly environment and will also help to keep a track its previous city names been searched. Due to its simplicity and openess it can be made easy for the server to provide new implementation in the app. The System provides user appropriate,easy to use easy recovery of errors and providing a overall better experience to the user.

Introduction

NEED Climatology and Weather Forecasting is important since it helps determine future climate expectations. Most of the people in world a depedent on accurate weather condition for there day to day activity to go well.

Hence do overcome the need we introduce the weather app produced by project team in order to overcome the problems that occurred due to age old methods. The newly introduced system will provide an easy access to system and contains user friendly functions with a Attractive interface. Coding is handled through an Object oriented approach which made the project team have a clear idea about problem and make ease to development.

Consisting of many functions such as form to import the city name, delete function to delete the unnecessary data , models and the most important views consisting of all such functions in it and making the body fully functional for the frontend we have used Ninja form which make the web frontpage more attractive and ease to the serever to detect errors quickly And for the accurate information team has used api named **openeweathermap** which will give the most accurate information to user input. Intefaces will be designed user friendly and the functions will dispalyed in a simple manner.

Issues faced in checking climate condition manually

- Manually keeping records is very time consuming.
- Information is not always reliable on news or online due chances of some human errors might occur example display wrong temp of any city.
- Finding Previous records are difficult .
- Any sudden changes can not me made quickly at times leading to false planning
- the major obstacles that weather forecasters confront are ill-suited expectations.

The system we are going to develop will give remedies for the problems that are currently facing by public . Shifting to our sysyem can acquire advantages such as saving of time, accurate data, space wastage. This will increase the efficiency in the overall activities.

Here by increasing the state of efficiency in weather report over time.

Problem Analysis

Functional requirements:

1. Take city name as input from user
2. Verify the city name entered using openweathermap api
3. After verification Pass the city name to the models and add it into the admin panel
4. Display a message after entering the city name
5. Display the city name and city weather information using openweathermap api
6. Delete the city name if required

Other requirements:

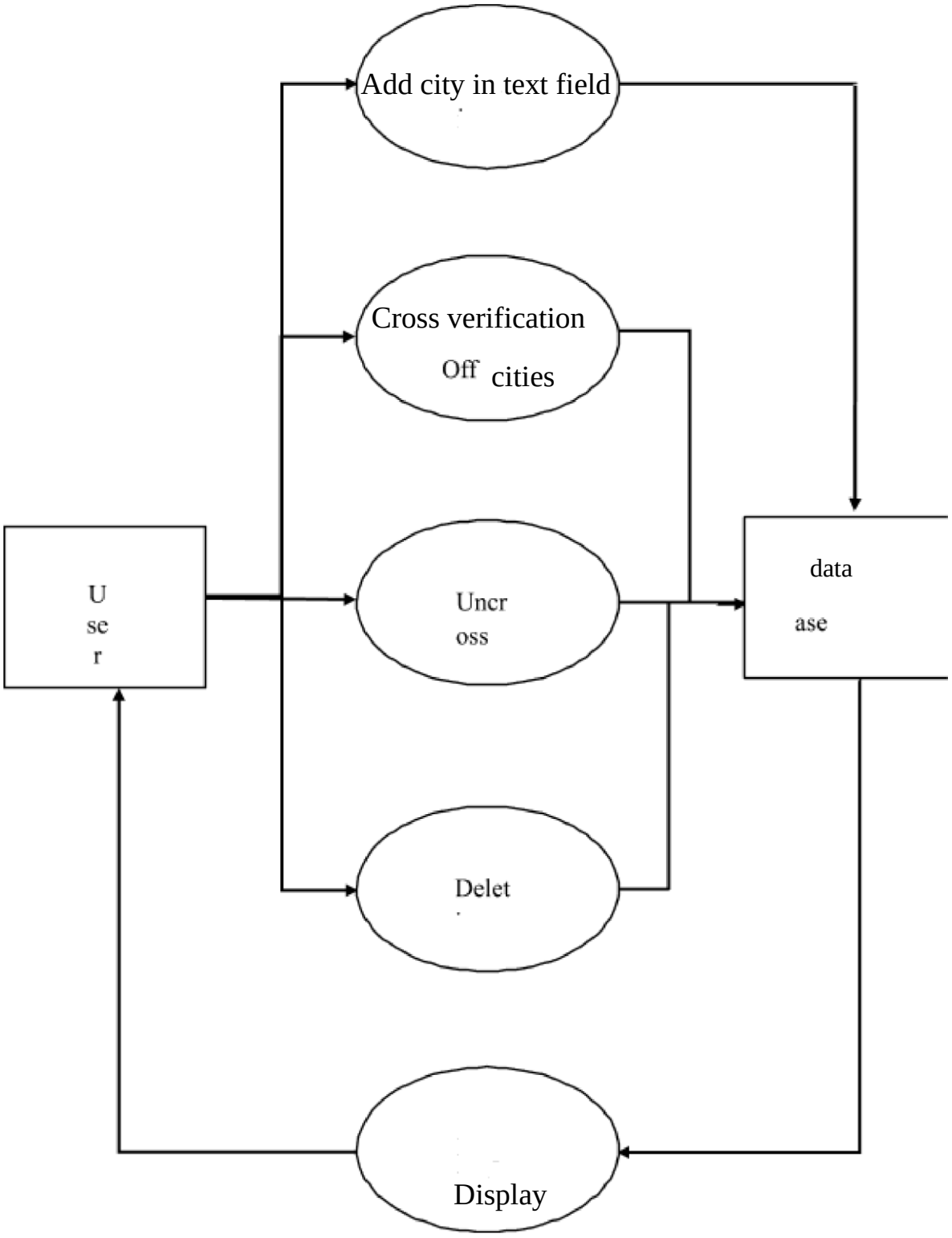
- In this web app we use the admin panel as the database to store the city name
- Develop a suitable frontend using html and css for displaying the city information.
- Create an account in admin panel to create different models.
- Use of jinja extension to accept the backend data.
- Styling the webpage using bootstrap classes.
- Use of django server to run the app.

Design

In this project a user will request a process from the web app, the web-app will deal with or execute the function associated with that particular process. After completion of the process the web-app will make some changes in the database and display the information back to the user.

For instance if the user decides to add a city then the city entered in the input field will be verified through the api and if the city is valid then it is added in the models in the admin module. If the city is not valid or has already been added then a suitable message will be displayed.

Furthermore the user can delete a city from the app if he/she wishes to.


Add city in text field
Cross verification
Off cities
U
se
r
Uncr
oss
data
ase
Delet
Display

Implementation

Python offers django framework to create web applications. To create a web application using the django framework, first we need to install django in our system and then set up a feasible working environment for a django app to work properly.

Steps to install django and set up the working environment:

i. Install pip : open command prompt and enter command-python -m
pip install -u pip

ii. Install virtual environment-
python install virtualenv

iii. Set virtual environment:
Setting up the environmental will allow to edit the dependency which generally system wouldn't allow-
   a. Create a virtual environmental-
   Virtualenv env_site-
   b. Change directory to env_site-
   Cd env_site
   c. Go to script directory and activate the virtual environmental-
   cd Scripts
   Activate

iv. Install Django-pip
Install Django

v. Return to the env_site directory
cd..

vi. Start project-Django-
Admin.py startproject the_weather

vii. Change directory
to the_weather
cd
weather_web_ap

viii. Start the server-
python manage.py runsever

ix. Goto web browser and enter http://127.0.08000 to check whether the Django is installed or not.

In this project we have used Admin panel as database to store the informationof the city name.. For the frontend framework we made use of Bootstrap, CSS, HTML.

Program Code

models.py:

```
from django.db import models

# Create your models here.
class City(models.Model):
    name=models.CharField(max_length=25)

    def __str__(self):
      return self.name

    class Meta:
        verbose_name_plural='cities'
```

forms.py:

```
from django.forms import ModelForm, TextInput
from .models import City

class CityForm(ModelForm):
    class Meta:
      model = City
      fields = ['name'      widgets= {'name' : TextInput(attrs={'class' : 'input' , 'placeholder' : 'City Name'})}
```

Urls.py:

```
from django.urls import path
from . import views
urlpatterns = [

    path('',views.index,name='home'),
    path('delete/<city_name>',views.delete_city,name='delete_city'),
]
]
```

Views.py:

```
import requests
from django.shortcuts import render, redirect
from .models import City
from .forms import CityForm


# Create your views here.
def index(request):
    url='http://api.openweathermap.org/data/2.5/weather?
q={}&APPID=cf02b9e527d5dfbaed9f2b9e4d509b09'

    err_msg=''
    message=''
    message_class=''


    if request.method== 'POST':
        form=CityForm(request.POST)
        if form.is_valid():

            new_city=form.cleaned_data['name']

            existing_city_count=City.objects.filter(name=new_city).count() ##returns the count of the city name
if it already exists

            if existing_city_count == 0:

                r = requests.get(url.format(new_city)).json()

                if r['cod'] == 200:

                    form.save() ## saves the form ie city name in databsaae ie admin

                else:
                    err_msg='Invalid City Name'
```

```
            else:
                err_msg='City already exists'

        if err_msg:
            message=err_msg
            message_class='is-danger'

        else:
            message='City added successfully'
            message_class='is-success'


    print(err_msg)

    form=CityForm()## returns the input field empty after enter a city name

    weather_data = []

    cities=City.objects.all() ##city are being fetched from admin db

    for city in cities:

      r = requests.get(url.format(city)).json() ## api is being used

      celsius=int(r['main']['temp']-273.15)

      city_weather={
        'city':city.name,
        'description':r['weather'][0]['description'],
        'icon':r['weather'][0]['icon'],
        'temperature':celsius,
      }
      weather_data.append(city_weather)


    context={'weather_data' : weather_data,
    'form' : form,
    'message_class' : message_class,
    'message' : message,
    }

    return render(request,'weather/weather.html',context)

def delete_city(request, city_name):
    City.objects.get(name=city_name).delete() ## deletes the city from the City model which is linked with the
admin database
    return redirect('home')
```

weather.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
  <link href="https://fonts.googleapis.com/css?family=Oxanium&display=swap" rel="stylesheet">
</head>

<body style=" background: linear-gradient(315deg, #f8ceec 0%, #a88beb 74%); ">
    <div class="container-fluid ">
        <h1 class="title" style=" padding-top: 2%;">
            <p class="text-center" style=" font-family: 'Oxanium', ;
            font-weight: 300;font-size: 45px; color:#383738;">Weather Forecast</p>
         </h1>

    <section class="section">
        <div class="container">
            <div class="columns">
                <div class="column is-offset-4 is-4">
                    <form method="POST">
                        {% csrf_token%}
                        <div class="field has-addons">
                            <div class="control is-expanded">
                                {{ form.name }}
                            </div>
                            <div class="control">
                            <button type="submit" style="marginleft: 0.4em; background: rgba(119, 21, 143, 0.75);
                              color: #ffff; border-radius: 10%; padding-top: 0.3em; padding
bottom: 0.3em; padding-left: 0.7em; padding-right: 0.7em; border: 1px solid rgba(182, 62, 212, 0.75);">
                                    Add City
                                </button>
                            </div>
                        </div>
                        {% if message %}
                        <div class="notification {{ message_class }}">
                            {{ message }}
                        </div>
                        {% endif %}
                    </form>
                </div>
            </div>
        </div>
    </section>
    <section class="section">
        <div class="container">
            <div class="columns">
                <div class="column is-offset-4 is-4">
                    {% for city_weather in weather_data %}
```

```
            <div class="box">
               <article class="media">
                  <div class="media-left">
                     <figure class="image is-50x50">
                   <img src="http://openweathermap.org/img/w/{{ city_weather.icon }}.png" alt="Image">
                     </figure>
                  </div>
                  <div class="media-content">
                     <div class="content">
                        <p>
                           <span class="title">{{ city_weather.city }}</span>

                           <br>
                           <span class="subtitle">{{ city_weather.temperature }}° C</span>
                           <br> {{ city_weather.description }}
                        </p>
                     </div>
                  </div>
                  <div class="media-right">
                 <a href="{% url 'delete_city' city_weather.city %}"><button class="delete"></buttn></a>
                  </div>
               </article>
            </div>
            {% endfor %}
         </div>
      </div>
   </div>
   <footer class="footer-2" style="background-color: #383738; padding:0.8em; margin-top: 18em;">
      <div class="copyright text-center" style="color: #ffff;">
         © Copyright <strong><span style="color: #ffff;">WEATHER FORECAST</span></strong>. Al
l Rights Reserved
      </div>
   </footer>
</body>

</html>
```

Admin.py:

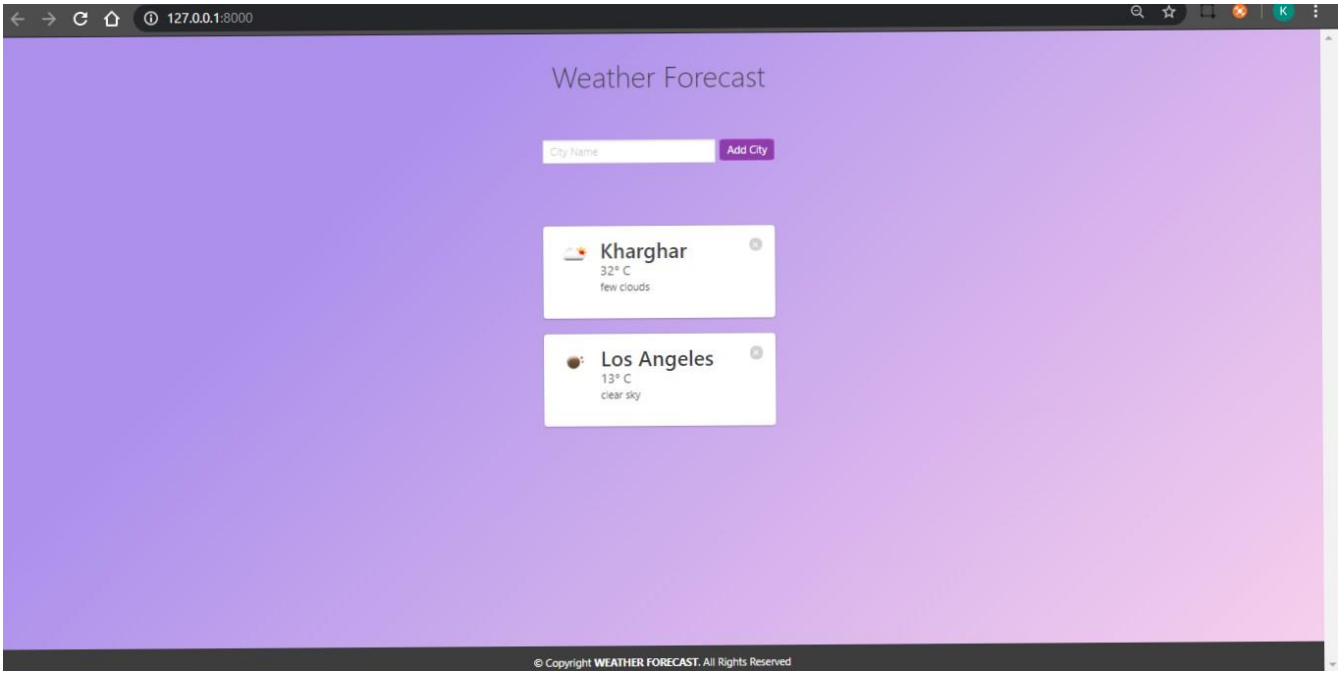
```
from django.contrib import admin

from .models import City

# Register your models here.

admin.site.register(City)
```

Apps.py:

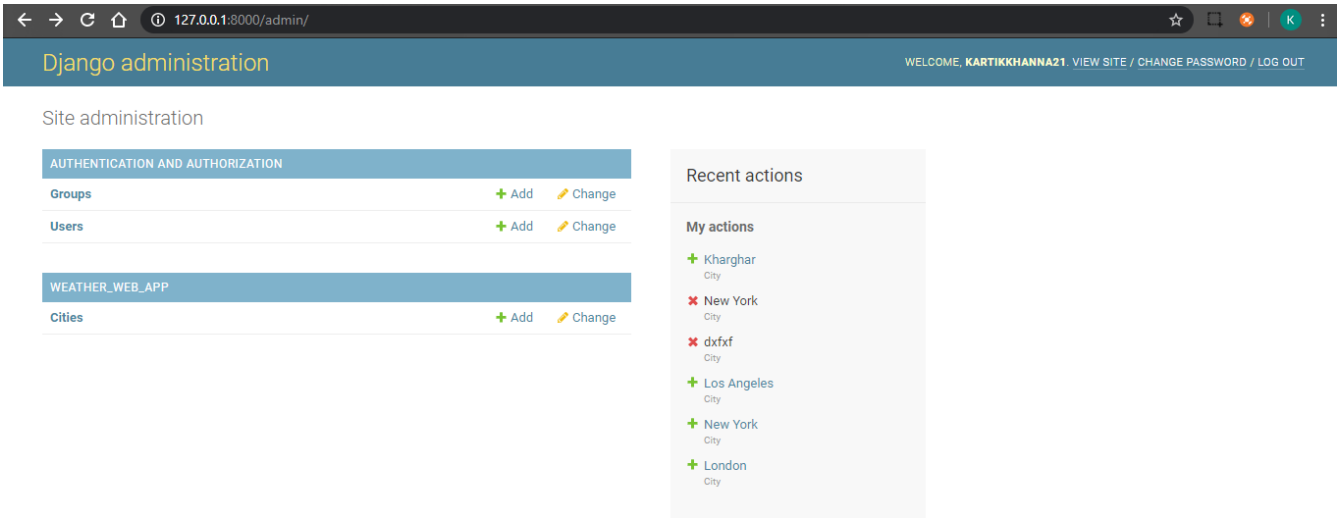
```
from django.apps import AppConfig


class WeatherWebAppConfig(AppConfig):
    name = 'weather_web_app'
```

Output/Results

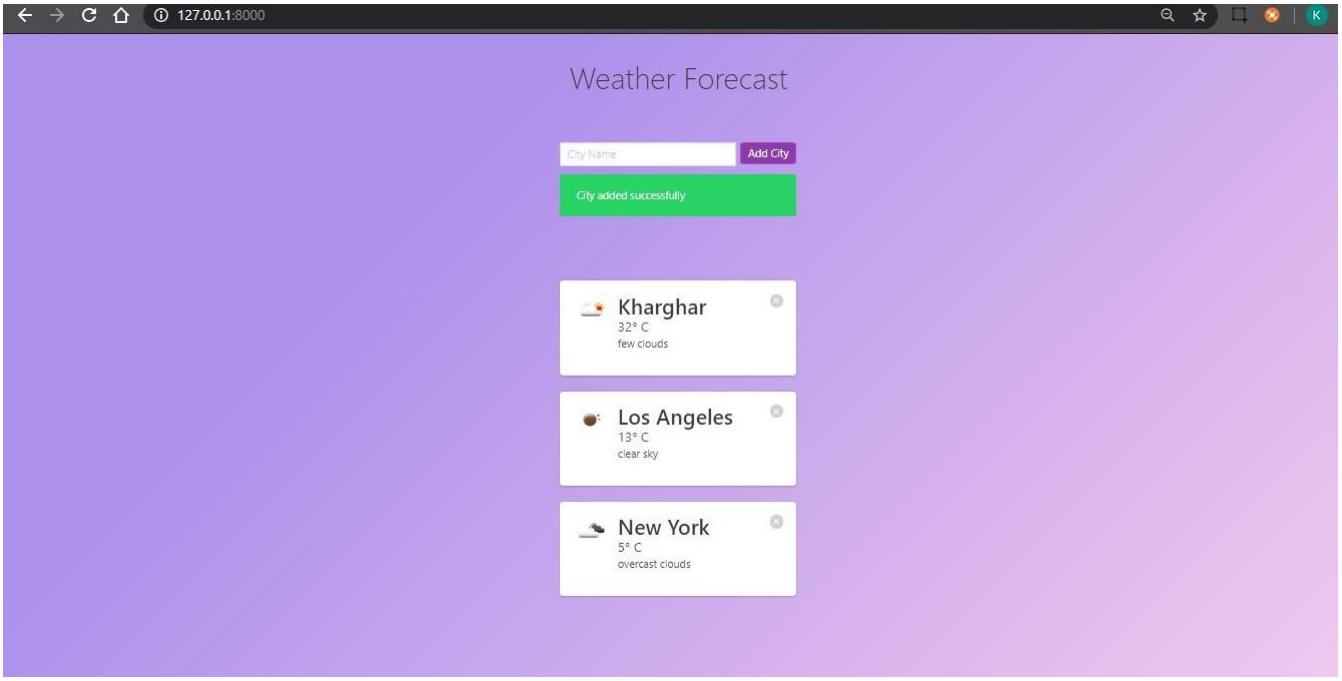
1. Graphical user Interface:



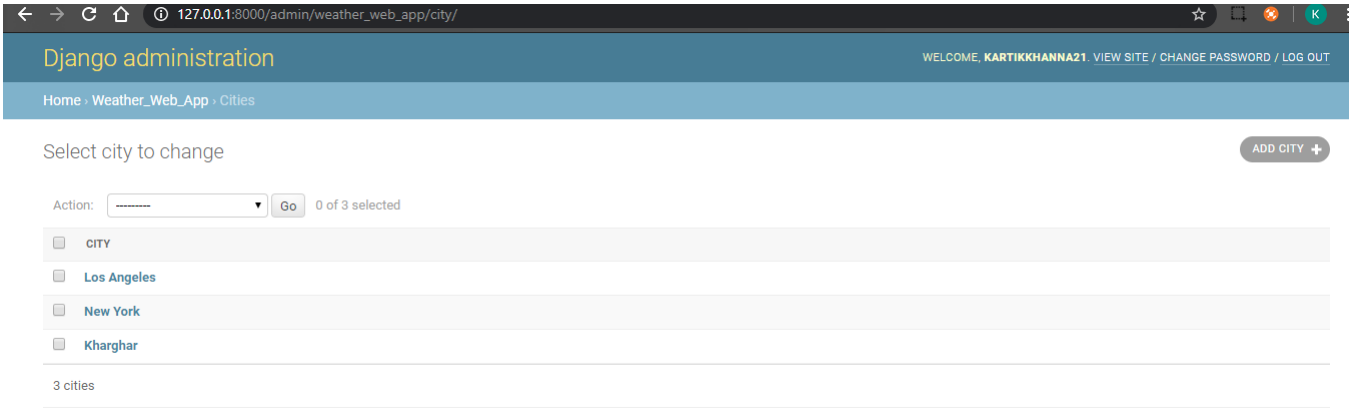
2. Admins panel:



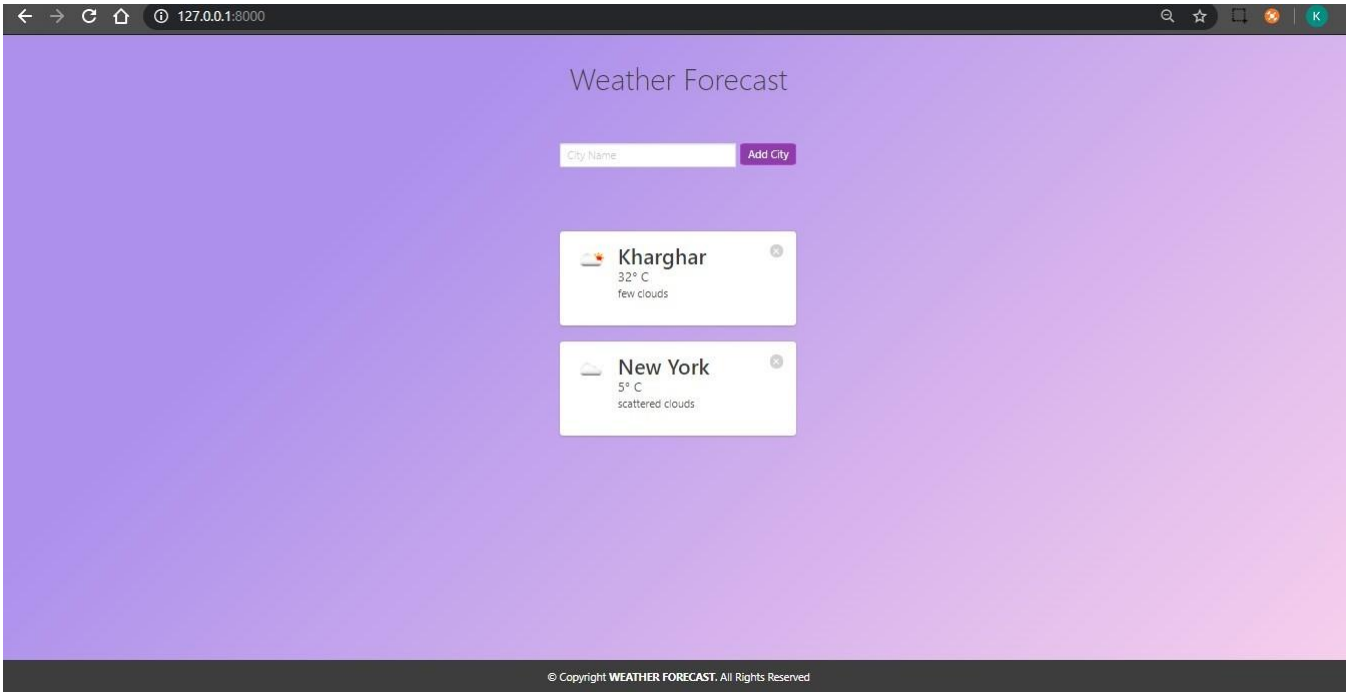
3. After adding a city:



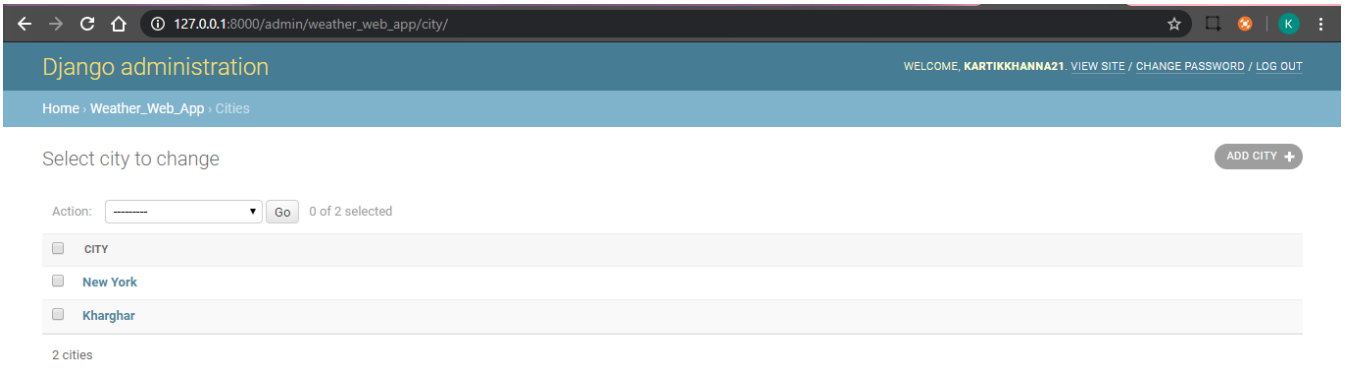
4. Admins panel changes:



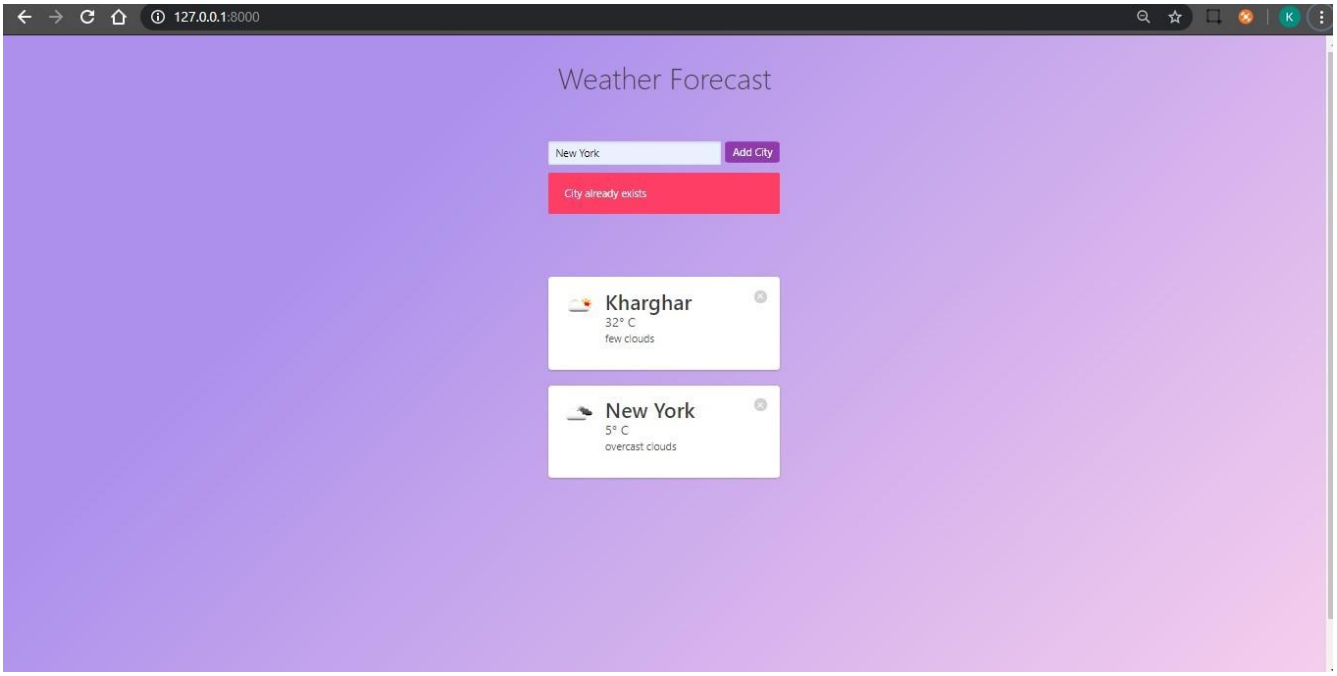
5. After deleting a city:



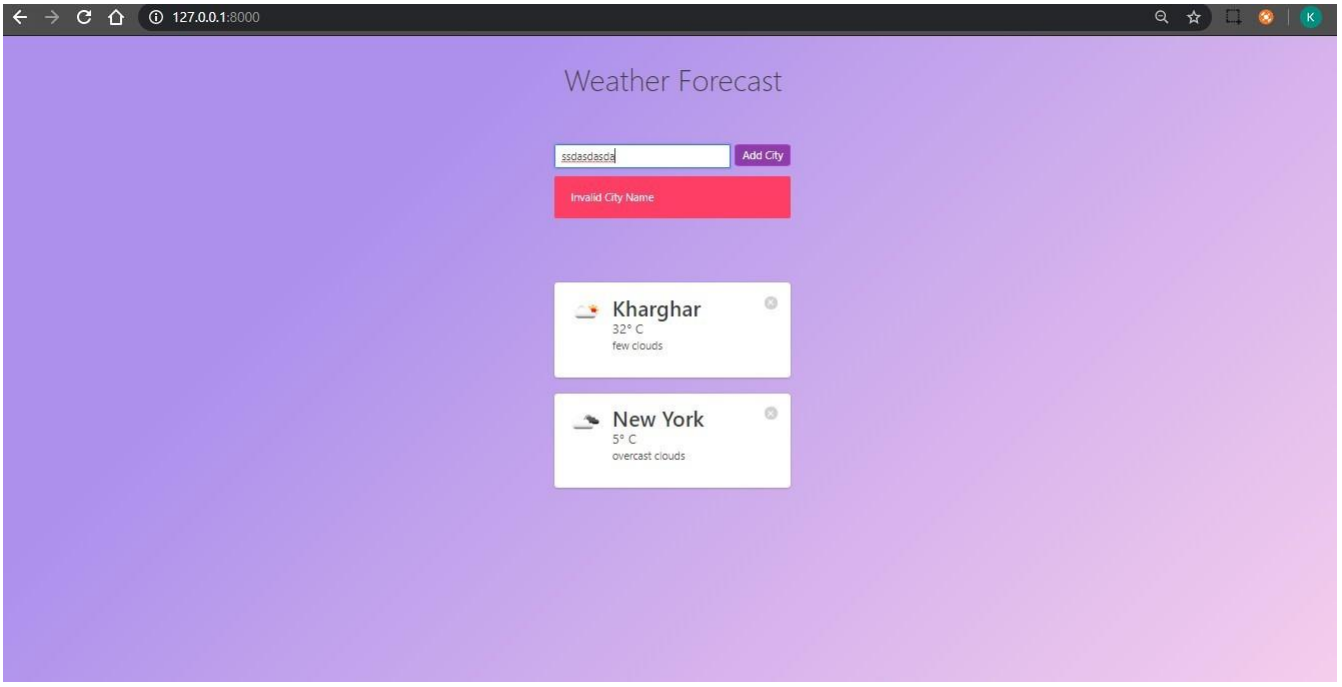
6. Admin panel changes to:



7. After adding the city again:



8. After adding the city with invalid name:



Conclusion:

The project ‘weather app’ has been designed using django framework and openweathermap api to let the user get to know the information about a city weather by simply entering a city name in input field. In turn the city’s name with weather details is displayed to the user with which he/she can stay updated with the current weather situation. In case the user wants to delete the city details it can be done as well to provide the functionality of a complete weather forecast system