

Sentiment Analysis
(Twitter Opinion Mining Using Sentiment Analysis)
A

Project Report

Submitted In Partial Fulfillment of the Requirements For The

Award Of the

Bachelor of Computer Application
Under Guidance Of

DEBASISH SAHOO

Project Carried Out At



Ardent Computech Pvt. Ltd.
(An ISO 9001:2015 Certified)

SDF Building, Module No:132, Ground Floor Sector V, GP Block, Kolkata- 700091

Submitted by

RAJDIP RAY



1. Title of the Project: **WEATHER FORECAST WEB APPLICATION**

2. Name of the Guide: DEBASISH SAHOO

3. Software used in the Project

- Django
- HTML
- Java Script
- <https://rapidapi.com>(API)

Signature of the Guide

For Office Use Only

Approved

Not Approved

Signature, Designation Stamp of the

Project Proposal Evaluator

Date:

Self Certificate

This is to certify that the dissertation/project proposal entitled “**WEATHER FORECAST WEB APPLICATION**” is done by **RAJDIP RAY** an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Computer Application** under the guidance of **DEBASISH SAHOO**. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Name of the Student(s):

Signature

1.

: _____



Certificate By Guide

This is to certify that this project entitled “**WEATHER FORECAST WEB APPLICATION**” submitted in partial fulfilment of the degree of **Bachelor of Computer Application (BCA)** by **Institute Of Engineering & Management** done by **RAJDIP RAY**, Is an authentic work carried out under my guidance & best of our knowledge and belief.

Signature of the student

Date:

Signature of the Guide

Date:



Certificate of Approval

This is to certify that this documentation of **Summer Industrial Training Program 2019**, entitled “**WEATHER FORECAST WEB APPLICATION**” is a record of bona-fide work, carried out by **RAJDIP RAY** under my supervision and guidance.

In my opinion, the report in its present form is in fulfilment of all the requirements, as specified by the **Sikkim Manipal Institute of Technology** and as per regulations of the **Ardent Computech Pvt Ltd.** In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for **Bachelor of Computer Application** .

Mr. Rahul Sharma
Ardent Computech Pvt Ltd
 (An ISO 9001:2015 Certified Company)

CONTENT

1. Acknowledgement.....	
..... I	
2. Main report.....	7
2.1 Introduction.....	
..... 8	
2.2 Objective & scope of the project.....	
9	
2.3 Theoretical	
background.....	10
2.4	
ERD.....	
.....	16
2.5 INPUT OUTPUT SCREEN DESIGNING.....	
18	
2.6 Code	
sheet.....	
.....	20
2.7 User	
manual.....	
... 25	
3. Annexure.....	
..... I	
4. A. Bibliography.....	28
B. Website.....	29

MAIN REPORT

2.1 INTRODUCTION

“Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.” That’s a mouthful — or eyeful or pixelful, depending on whether this book is being recited, read on paper or projected to you on a Jumbotron, respectively. Let’s break it down.

Django is a high-level Python Web framework... A high-level Web framework is software that eases the pain of building dynamic Web sites. It abstracts common problems of Web development and provides shortcuts for frequent programming tasks. For clarity, a dynamic Web site is one in which pages aren’t simply HTML documents sitting on a server’s file system somewhere. In a dynamic Web site, rather, each page is generated by a computer program — a so-called “Web application” — that you, the Web developer, create. A Web application may, for instance, retrieve records from a database or take some action based on user input.

A good Web framework addresses these common concerns:

- It provides a method of mapping requested URLs to code that handles requests. In other words, it gives you a way of designating which code should execute for which URL. For instance, you could tell the framework, “For URLs that look like /users/joe/, execute code that displays the profile for the user with that username.”
- It makes it easy to display, validate and redisplay HTML forms. HTML forms are the primary way of getting input data from Web users, so a Web framework had better make it easy to display them and handle the tedious code of form display and redisplay (with errors highlighted).
- It converts user-submitted input into data structures that can be manipulated conveniently. For example, the framework could convert HTML form submissions into native data types of the programming language you’re using.
- It helps separate content from presentation via a template system, so you can change your site’s look-and-feel without affecting your content, and vice-versa.
- It conveniently integrates with storage layers — such as databases — but doesn’t strictly require the use of a database.
- It gets out of your way, neglecting to leave dirty stains on your application such as URLs that contain “.aspx” or “.php”.

Django does all of these things well — and introduces a number of features that raise the bar for what a Web framework should do. The framework is written in Python, a beautiful, concise, powerful, high-level programming language. To develop a site using Django, we write Python code that uses the Django libraries. Although this book doesn’t include a full Python tutorial, it highlights Python features and functionality where appropriate, particularly when code doesn’t immediately make sense.

2.2 OBJECTIVE & SCOPE OF THE PROJECT

Objective:

The objective is to design and develop a Community website for Social Workers. People on the internet always search for their own taste. People making friends in Social Networking websites, making his/her profile, Viewing articles of their own interest, sharing thoughts, Searching answers of their Queries, Posting Comments, Giving Suggestions, Searching news related to topics of his/her own interests on the internet. Undoubtedly there is no such better place than the web to be connected with others, may it be a Group of People or an Organization or just your social friend. Keeping these in mind this System namely Community website for Social Workers has the objective of bringing all social workers together to communicate, share, suggests, ask, support, follow, update etc. to enjoy and empower Social Working.

Scope:

In adherence with the rules framed by the _____ relating to the restrictions imposed on the volume and time invested in a project for _____, the scope of the project has been limited to general activities of a community website where Social work activities is the main concern.

In future as and when the need is felt, further modules may be developed and integrated with this system. This system would comprise –

- || Warning mail generation and Dynamic User Blocking
- || Chat Facility with Expert
- || Extensive Searching (User, Group)

2.3 THEORETICAL BACKGROUND

To complete this project, the User-Centered Design (UCD) methodology is going to be used to customize the features and adapt the interface to the users' needs UCD methodology outlines the phases throughout a design and development life-cycle all while focusing on gaining a deep understanding of

who will be using the product. Known the philosophy of UCD, this process will be applied in the project, thus dividing it in three phases (analysis, development and evaluation).

However, it's important to remark that UCD is an iterative process, meaning that this is not a closed process or unidirectional. It is dynamic process that evaluates and re-evaluates itself constantly in order to offer the best product tailored to users' needs. In the first phase: the analysis and research phase, there will be an ongoing dialogue with the potential users, our target, is to be able to define properly the scope of the project, its requirements, their real needs, etc. Different techniques will be used to obtain qualitative and quantitative data. Once processed, the data and all the basic input for the development of the application will have been detailed. This stage of the project will be used to define the structure and scope of the project, that will be implemented during the development phase. On this stage there will also be a benchmarking with similar applications and later on all application architecture will be developed including task analysis, use cases, contents tree and flow diagrams. The last phases: development and evaluation will be conducted simultaneously to provide a tested and tailored application to the users' real needs. Once all the basics of the application are defined, such as requirements, scope, etc. low-quality mock-ups will be designed (sketches and wireframes), lastly a high-quality interactive prototype will be created, to verify once again the interaction and user experience. At the same moment a constant evaluation will be carried to assess and verify the usability and accessibility of the prototypes created, using techniques such as: heuristic 32 cognitive walkthroughs, tests with users, etc. To improve or correct problems that have been overlook.

Requirement analysis:

This section plans to define and establish all the requirements needed to successfully complete the design and implement the weather application. This stage of analysis during the phase of design is essential, as the requirements observed in this phase are going to be used during the whole project design. All the requirements detailed in the next sections, haven been obtained by an online survey in which 117 people took part.

Survey link: Data collection for creation of weather application. <http://goo.gl/forms/kbA67kXVDw> In this survey the participants where asked a set of questions in order to find out their needs and what would they expect from a weather application. The questions were:

- Gender
- How old are you?
- How often do you check for the weather forecast?
- Would you be interested in cloud maps? (Radar imaging)
- Would you be interested in videos related to weather?
- Would you be interested in flood warnings?
- Would you be interested in UV values?
- Would you be interested in Wind & Pressure data?

would you be interested in a notification with current weather data, providing access to a detailed view?

- What forecast data would you be interested in?

What kind of application would you be interested in?

- If you had to pay to use the application, how much would you be willing to pay?

User requirements:

All the functions available to the users:

- Users can access weather forecast data and receive notifications with weather updates.
- The weather data has to be for: current, hourly, daily, and up to 10 days' forecasts.
- Users can access radar imaging data.
- Users can receive severe weather conditions notification; such as flood warnings.
- Users can access UV data values.
- Users can access Wind and Pressure data values.
- Users can access Pollen data values.
- Users can access Sunrise and Sunset data values.
- The application can be used by a large number of users simultaneously. This means the API has to allowed a large number of calls per minute.
- The performance of the application cannot decrease while adding new functionalities.

Functional requirements:

These are the requirements that define the internal behavior of the application. They are based on the Use Cases and guarantee a fully functional application.

- Users can add and remove locations to receive their weather data.
- Locations can be added by user coordinates (Geo Position) or location search.
- Application shows notifications on weather update on selected locations

- Application offers data on the location saved by the user, in ranges of:

- o Current
 - o Hourly
 - o Up to 10 days

Notifications allow to access a detailed information.

- On severe weather conditions users gets notified.

- Users can access different types of data:

- o Radar imaging
 - o UV data
 - o Wind Pressure
 - o Pollen
 - o Sunrise and Sunset

- Data shown is obtained and aggregated from different sources.

The application must support different languages, through the use of Locales in Android, so the language selection is transparent to the user, as it will load the text in the language defined in their device.

Interface requirements:

- Interface design must be flexible and scalable to be used in different devices with different screen resolutions and sizes.
- Interface should be easy to use, rather than intimidating and requiring user learning.
- Interface should be intuitive and provide easy to understand items such as: buttons, headings, messages or errors.
- The screen layout and colors should be attractive.
- Interface must be consistent: elements should be organized similarly across the interface, allowing users to predict what will happen when an action is taken.
- Content should be displayed in a logical and natural order.
- Interface should display all the necessary information at all times, so users do not need to remember information.
- Users have to know at any time, in which part of the application are located.
- Users can navigate to the home page from any part of the application.
- Interface response has to be fast, to avoid waiting time for users or making them think that the application does not respond, the ideal response time should be less than 0.2 seconds.
- Progress bar or a timer should be shown in case of slower response times.

Usability requirements:

Users do not need to have any specific skills for normal use of the application.

images and icons used must be representative.

Text must be large enough to be readable without difficulty, using scale-independent pixels (SP), in case the standard defined size does not offer enough readability users can change the text size.

- Efficiency of use: goals (tasks) must be easy to perform and should not lead to confusion.
- Application should be easy to learn.
- All possible actions and elements of the interface must react accordingly and consistently.
- In the event of error messages, they should explain how to fix the error.
- Undo option should be offered in the majority of the actions, especially the permanent ones.
- Offer feedback to the users after completing an action.

Help must be sensitive to the context; it should explain how to perform tasks.

- When a new functionality is added, provide relevant information and guided steps on how to use it.

Technological requirements:

This list includes all the requirements for the execution of the project and for the usage of the application

Mobile Device: smartphones.

Requirements for a recommended experience of the application.

- Operating System Android 4.0.3, API 15, in this way we can target 97.3% of the active users on Google Play Store.
- Low usage of CPU, to prevent high use of battery.
- Low use of data connection on mobile data, as some data plans are capped.
- Low usage of RAM, in order to not slow down the device.

Software used for the development of the project

- Microsoft Word, Excel, Project, PowerPoint used to write all documents related to the project.
- Extension used to create the personas.
- Balsamiq used to create the sketches and Wireframes
- Justin mind, Axure RP used to create the Interactive prototype.
- Adobe Fireworks, Photoshop, Illustrator CC used to capture, design and process images.

Adobe Premiere, Audacity used for capturing audio and creating audio tracks. Acrobat Professional used for the final editing of the Dissertation

- Google Maps used for user positioning, weather maps...
- Android Studio to code, compile and pack the application.

Maintenance requirements:

After following all the stages of UCD methodology detailed in this project, a well-designed and tested application will have been coded, however this will only be the first release of the application, in which all future modifications and upgrades will be built. Once the launch and distribution of the application is completed through Google Play Store, the first phase of lifecycle of the application will be finished and the maintenance and upgrade period will begin. In this period different tasks will be required, to be able to continue and guarantee the optimum performance and up to date functions, such as:

- Functional and usability upgrades: update the application with new functions obtained through user feedback and new usability guidelines.
- Interface and graphic upgrades: to keep the UI up-to-date and follow any modifications on the design guidelines set by Google.
- Technical upgrades: to upgrade the code to new coding standards and to make sure the application is compatible with new releases of the O.S. or new devices.

Task analysis

The application has to allow the users to complete different sets of tasks in an efficient and simple manner. On the first release of the application the users will be able to access the following tasks, such as:

Location:

- Add location: users can add a new location in order to get updated weather data, this task can be completed by adding the location by a search or by using the Geo-position.
- Remove location: users can remove a location from which they do not want to receive more weather updates.
- Sort locations: users can sort the locations that they have previously saved, in order to see the data in the order they want. Settings:

Customize units for temperature: users can choose between Fahrenheit or Celsius.

Customize units for wind speed: users can choose between Miles per hour or Kilometers per hour.

- Notifications: users can enable or disable weather updates notifications.
 - o General notifications: providing up to date forecast data.
 - o Severe weather: notifies the users from important weather updates.

Weather Data:

-
- Forecast data: users can access weather and other data for their saved locations:

Time interval: the data than can be accessed is separated into sections:

- o Current: provide the user the actual forecast data for a location.
- o Hourly: provide the user hourly forecast data up to 24hours.
- o Up to 10 days: provide the user daily forecast data up to 10 days.
- o Other data: Radar imaging: provide the user cloud forecast data.
- o UV data: provide the user data relative to the Sun intensity values.
- o Wind Pressure: provide the user data related the wind strength and direction.
- o Pollen: provide the user data related to pollen, to prevent allergies.
- ||
- o Sunrise and Sunset: provide the user data about the sunrise.

2.4 Detailed Life Cycle of the project – ERD


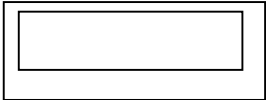
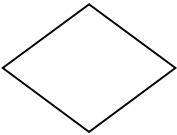
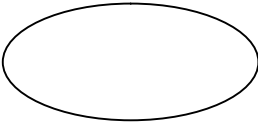

ERD (ENTITY-RELATIONSHIP DIAGRAM) : -

An entity is a thing that exists either physically or logically. An entity may be a physical object such as a house or a car (they exist physically), an event such as a house sale or a car service, or a concept such as a customer transaction or order (they exist logically—as a concept).

- Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entitytype. There are usually many instances of an entity-type.

Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

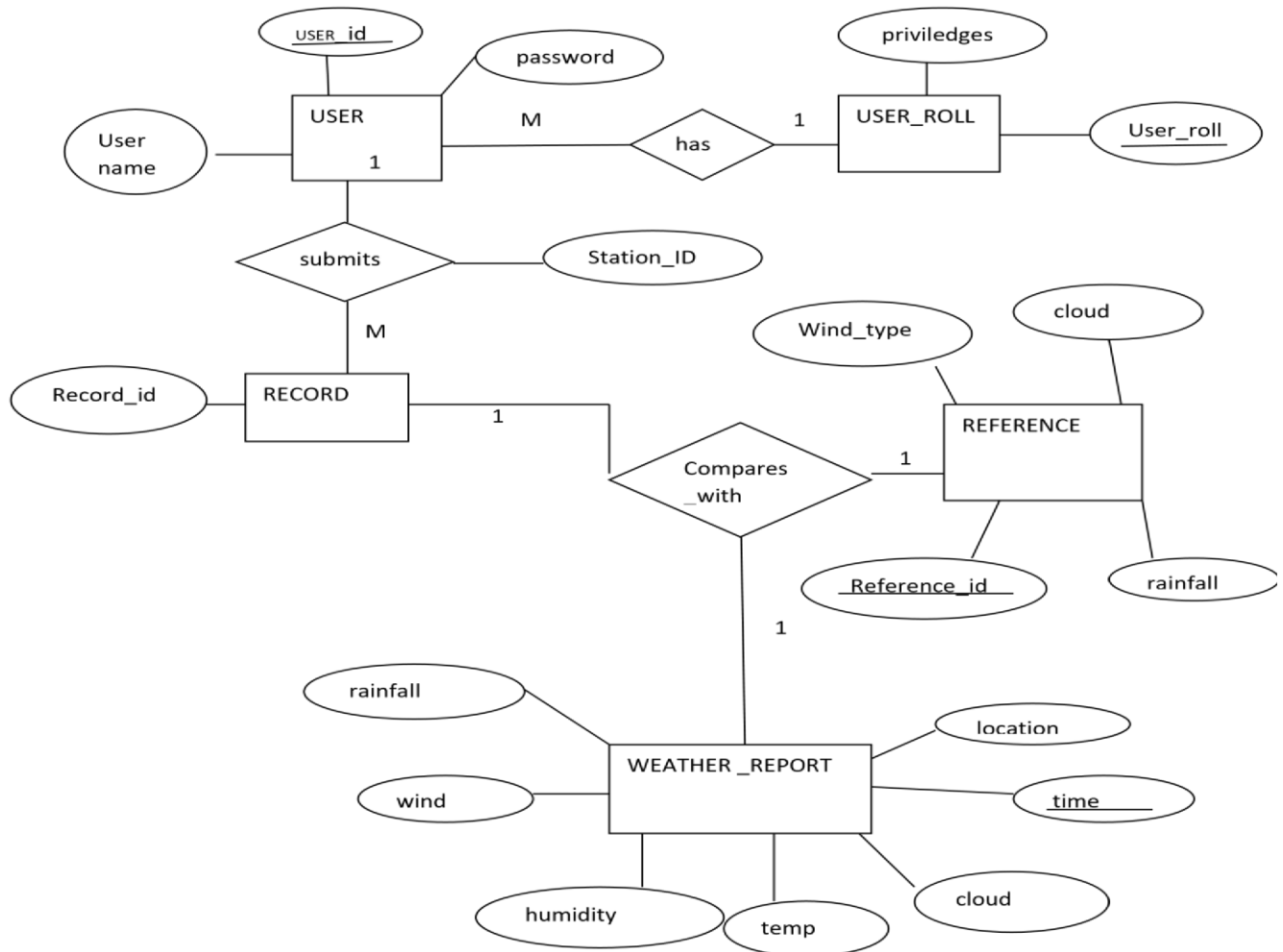
THE BASIC SYMBOLS USED FOR AN ERD ARE:

1. ENTITY 
2. WEAK ENTITY 
3. RELATION 
4. ATTRIBUTE 
5. KEY ATTRIBUTE 

└ Cardinality Marks

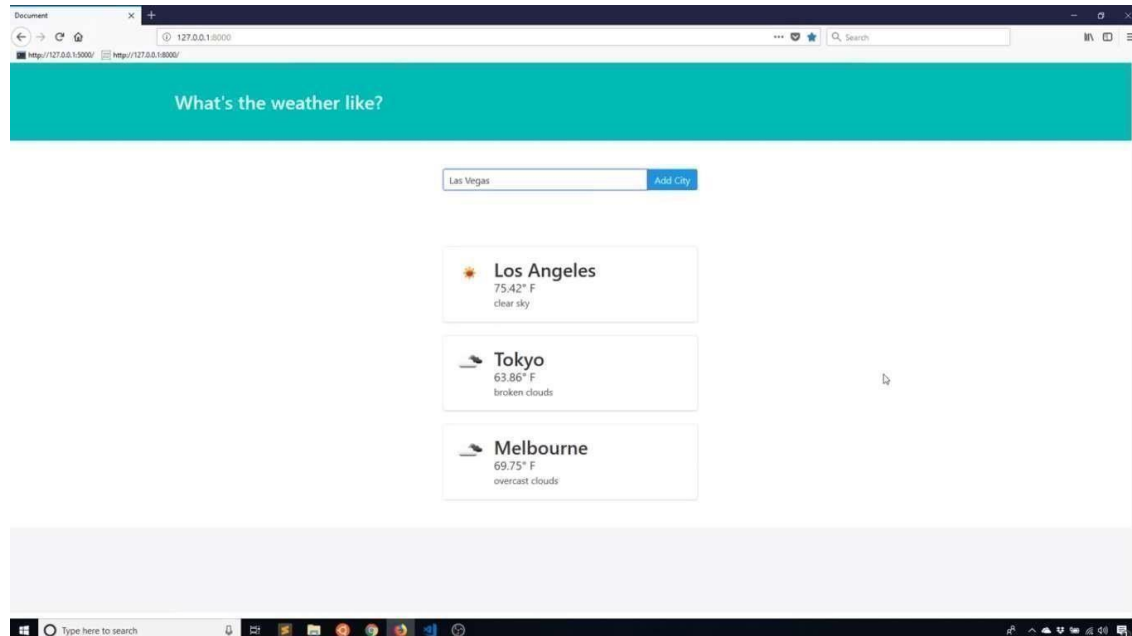
- | | |
|---|---------------------------------|
| 1 | No more than one related Entity |
| M | Many related Entities |

ENTITY RELATIONSHIP DIAGRAM

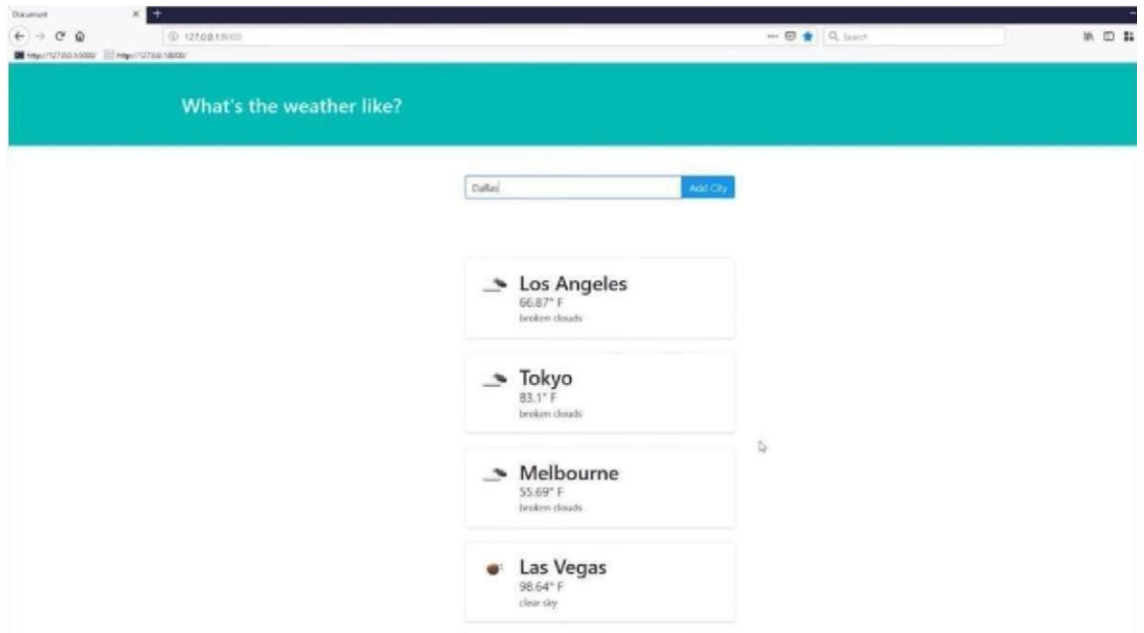


2.5 INPUT OUTPUT SCREEN DESIGNING

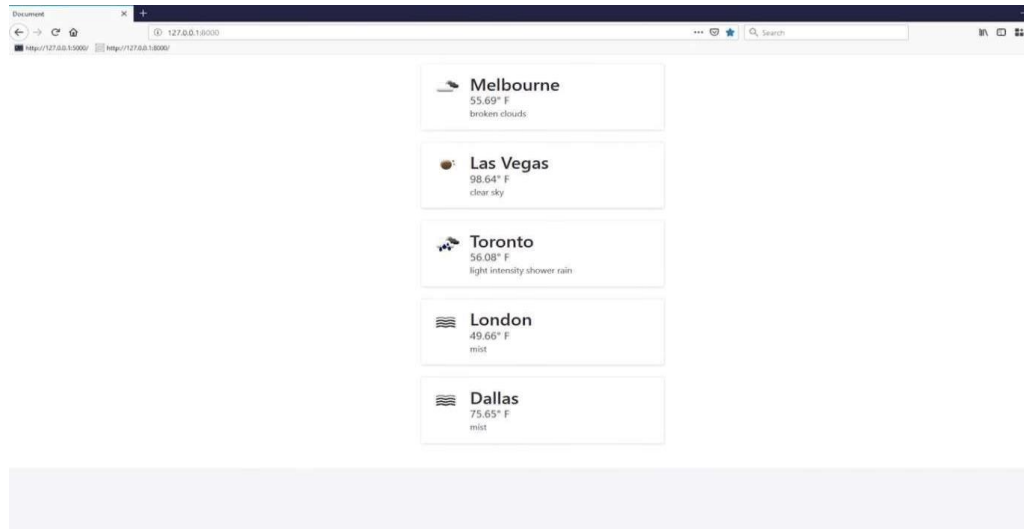
View1:



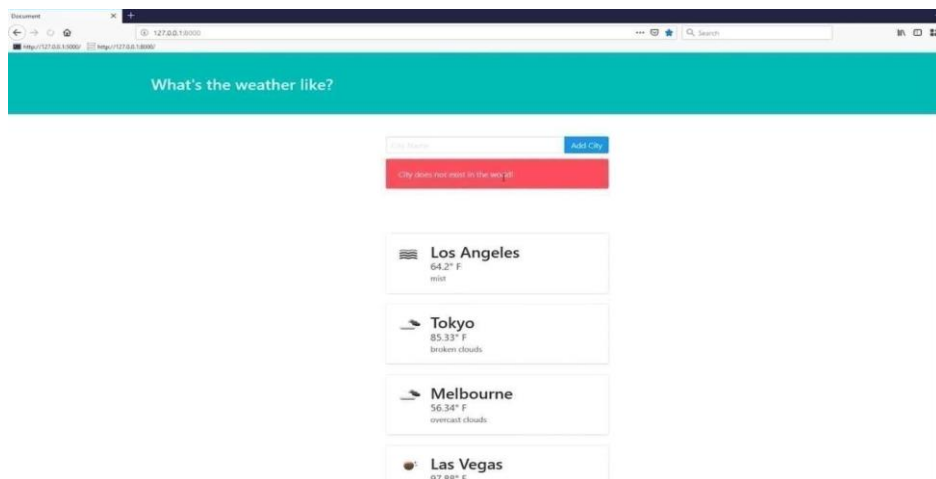
View2:



View3:



View4:



2.6 CODE SHEET

□ Weater.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://cdnjs.cloudflare.com/ajax/libs/bulma/0.6.2/css/bulma.css" />

</head>

<body>

  <section class="hero is-primary">

    <div class="hero-body">

      <div class="container">

        <h1 class="title">

          What's the weather like?

        </h1>

      </div>

    </div>

  </section>

  <section class="section">

    <div class="container">

      <div class="columns">

        <div class="column is-offset-4 is-4">

          <form method="POST">
```

```

<div class="field has-addons">
  <div class="control is-expanded">
    <input class="input" type="text" placeholder="City Name">
  </div>
  <div class="control">
    <button class="button is-info">
      Add City
    </button>
  </div>
</div>
</form>
</div>
</div>
</div>
</section>
<section class="section">
  <div class="container">
    <div class="columns">
      <div class="column is-offset-4 is-4">
        <div class="box">
          <article class="media">
            <div class="media-left">
              <figure class="image is-50x50">
                
              </figure>
            </div>
            <div class="media-content">
              <div class="content">
                <p>
                  <span class="title">Las Vegas</span>

```



```

        'temperature' : r['main']['temp'],
        'description' : r['weather'][0]['description'],
        'icon' : r['weather'][0]['icon'],
    }

    weather_data.append(city_weather)    context = {'weather_data' :
weather_data, 'form' : form}    return render(request,
'weather/weather.html', context)

```

- Urls.py: from django.urls import path from . import views urlpatterns = [path("", views.index),
]

- Models.py: from django.db import models 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={'cl ass' :
'input',
 'placeholder' : 'City Name'})}}

- Apps.py:
from django.apps import AppConfig class


```
WeatherConfig(AppConfig):    name =
'weather'
```

- Admin.py:

```
from django.contrib import admin from
```

```
.models import City admin.site.register(City) •
```

Initial.py: from django.db import

```
migrations, models class
```

```
Migration(migrations.Migration):
```

```
initial = True
```

```
dependencies = [
```

```
]
```

```
operations = [
```

```
migrations.CreateModel(    name='City',    fields=[
```

```
    ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
```

```
    ('name', models.CharField(max_length=25)),
```

```
],    options={
```

```
    'verbose_name_plural': 'cities',
```

```
},
```

```
),
```

```
]
```

2.7 USER MANUAL

The user manual gives guidelines on how to operate the software for all level of Users of the System.

❖Guest:

OPERATIONAL DETAIL:

All the Users are treated as Guest unless they have a valid User name and Password and enter in their respective profiles. In the home page or index page for the Guest there are 4 other links –

The vertical menu contains 4 other links, namely-

- || Smiley Blog – Blog view for Guest.
- || About Smiley – About the System.
- || Smiley Updates – View Updated news.
- || Contact Us – View Contact Information.

On clicking any of these links takes Guest to that page. On top-right position of the same page there are two Buttons, namely –

- || Sign In
- || Join

On clicking any of these two buttons a block area is displayed with relevant controls –

Button “Sign In”:

On clicking this button Users will get the options for Signing in or Logging in entering their valid User name and Password. There are also options for ‘remember me’ and link for Recovering Password (forget Password).

Button “Join”: clicking this button Users will get the options for Joining in this system and start registration process.

❖Administrator:

OPERATIONAL DETAIL:

After successful Log in, Administrator's Profile Home page appears. This page contains Profile Home Page Link (named, Home) and following links –

The vertical menu contains 7 other links, namely-

- || Screen Queries – List of Queries to be screened.
- || Screen Postings – List of Blog postings to be screened.
- || Screen Expert – List of Expert Applications to be screened.
- || Screen NGO – List of NGO Applications to be screened.
- || Screen Group – List of NGO Applications to be screened.
- || Create Group – To insert information for the group to be created. Update Smiley Activity
- || – To insert update information.

On clicking any of these links takes Administrator to that page. On top-right position of the same page there are 7 Image-buttons.

- || Home Image-button – Displays small profile picture and onclick redirects to Profile Home page from any other page.
- ||
- || Action Image-button – Allows the User to Log-out from the system.

Screening status image-buttons – There are 5 such buttons for 5 distinct screen functions of the Administrator and onclick they redirects the Administrator to corresponding Screening pages.

❖Member:

OPERATIONAL DETAIL:

After successful Log in, Member's Profile Home page appears. This page contains Profile Home Page Link (named, Home) and following links –

The vertical menu contains 4 other links, and on clicking these links it redirects to related pages with sublinks –

- || View Forum – List of Queries with Replies from Experts.
 - Home – Redirects to Member profile home page
 - Private Forum – List of Private Queries with Replies(default view of View Forumlink)
 - Public Forum – List of Public Queries with Replies.
- || Each of Private forum page and Public forum page has Query posting option for corresponding Forum type.
- || View Blog – Blog postings, comment view.
 - This page also allows Members to post blogs and comment on a post through Post a Blog and Comment buttons respectively.
 - Back to Home button is there to redirect Member to Profile Home page.
 - Reply against a comment can also be posted on clicking the Comments for this reply link with every comment.
- || View Album – Album view with first picture of the album displayed in a large view area below the displayed image list.
 - on clicking any of the image among the image list the selected image is displayed in the large view area.
 - Next and Previous image-buttons are there to view the next and previous image of a currently displayed image.
 - Upload Image link provided in the small album display view area of the Member profile home page facilitates the image uploading activity for Member Album.

Profile Settings

- Edit Profile – Allows the Member to edit or alter profile information.
- Change Profile Picture – Allows the Member to upload new profile picture file.
- Change Password – Allows the Member to change current Password on submitting valid current password.

On top-right position of the same page there are 4 Image-buttons.

- || Home Image-button – Displays small profile picture and onclick it redirects to Member Profile Home page from any other page.
- || Action Image-button – Allows the Member to Log-out from the system
- || View Profile image-button – On clicking this image-button it displays small block area

where profile details of the Member are listed.

- || View Peer ship Message button – This button can only be seen in Member profile home page. This button shows the no. of Friendship request has been received and on clicking this button a block area is displayed where Peer ship message details is displayed with Accept and Reject buttons.

BIBLIOGRAPHY

<u>BOOK</u>	<u>TOPIC</u>	<u>NAME</u>	<u>AUTHOR</u>
1	DJANGO	PRACTICAL DJANGO PROJECTS	JAMES BENNETT
2	DJANGO	TWO SCOOPS OF DJANGO	D.R.GREEN FIELD
3	PYTHON	PYTHON WEB DEVELOPMENT WITH DJANGO	J.FORCIER

4	DJANGO	DJANGO 2 BY EXAMPLE	A.MELE
---	--------	------------------------	--------

WEBSITES

<http://cooltext.com>

<http://www.php.net> <http://www.apache.org> <http://w3schools.com> <http://www.stackoverflow.com>

<http://www.wikipedia.org> <http://www.w3.org> <http://www.wiser.org>

<http://www.care2.org>



