

TRAINING REPORT

on

Mocking Weather

Submitted in partial fulfillment of the

Requirements for the award of

Degree of Bachelor of Technology in Computer Science & Engineering



Submitted By

Name: Aakriti Sharma

University ID: 18BCS1628

Training Group: CSG-11

SUBMITTED TO:

Department of Computer Science & Engineering

Chandigarh University

Gharuan, Mohali

1. Introduction To Project

1.1 Introduction

1.2 Existing System

1.3 Disadvantages of Existing System

1.4 Proposed System

1.5 Advantages of Proposed System

2. Tools & Technology Used

2.1 Introduction

2.2 HTML5 and CSS3

2.3 Bootstrap

2.4 JavaScript

2.5 Django

2.6 SQLite

3. Snapshots

3.1 Introduction

3.2 Landing Page

3.3 Multiple Cards

3.4 Sample Page

3.5 Mobile View

3.6 I-Page

4. Results and Discussions

4.1 Result

4.2 Discussion

5. Conclusions and Future Scope

Chapter 1: Introduction

1.1 Introduction

Weather Reporting has been going for ages however in its existing form it has grown pale and boring. The idea behind this website to bring back the fun and exciting quotient to weather reporting websites.

1.2 Existing System

Currently we have several applications and websites which provide us with detailed weather report and data, however they all lack a pleasing aesthetic to them and with so much information users are often lost in piles of ugly looking web pages and as a result unable to find the result they are looking for.

1.3 Disadvantages of Existing System

1. Plenty of information without any proper sorting or one stop place to get relevant information.
2. Need lot of manual effort to get to core and important information, which is mainly lost in heaps of pages.
3. Weather not being broken down for layman to understand with ease.

1.3 Proposed System

The idea behind this web application is to make a website which displays a detailed weather report through a card based UI (User Interface) and also provide users ability to search for weather conditions of any city. Besides this users will also be displayed a positive motivational message to bring a smile on their face and give them something to cheer about. Users will also be given the ability to add their own custom messages to our database, these messages will later be sorted and selected to become part of our production database and finally displayed on the website.

1.4 Advantages of Proposed System

1. A single page application which provides sufficient and relevant data to users in a single glance giving them access to core information with ease.
2. Data being broken down with easy to understand UI(user interface) with icons and image to explain different weather conditions.
3. A positive message as a bonus to cheer up our users, later add the ability to send this message as a notification depending upon user requirements.

Chapter 2: Tools & Technology Used

2.1 Introduction

To deliver a web application which compiles with modern web standards, we made sure all modern technologies were used however it meant our application won't be able to run on older web browsers such as IE 8 or older versions of Chrome and Firefox. To support older browsers certain functionality had to be withdrawn or a replacement had to be provided, however by the end due to our versatile technology stack we were able to target a large section of devices of varying sizes and form factor.

2.2 HTML5 and CSS3

Using the latest standards of HTML and CSS gave us the leverage to get basic design of our website up and running with ease. Using the new HTML5 tags such as header, section and footer made the code readable and maintainable. Using the new CSS3 standards help provide more customisation options such as using gradients and making a responsive website design which scales across devices with ease.

An example of new HTML5 tags and elements.

```
<!-- Header Tag -->
<header>
    <h1> Heading Goes Here </h1>
</header>

<!-- Nav Tag -->
<nav>
    <a href="#"> HTML </a>
    <a href="#"> CSS </a>
    <a href="#"> JS </a>
</nav>
```

An example of new styling features in CSS3.

```
div {  
    box-shadow: 2px 5px 0 0 rgba(72,72,72,1);  
    background: linear-gradient(to right, red , yellow);  
    opacity: 0.5;  
}
```

2.3 Bootstrap

Bootstrap is a free and open-source CSS-framework directed at responsive, mobile first front-end web development with design templates for typography, forms, buttons, navigation and other useful components. Using bootstrap helped develop a mobile responsive website which allowed us to target a large set of devices. With optional Javascript we were also able to add more interactive elements and make forms with client side Javascript check. We used a precompiled version of Bootstrap which is available in the form of one CSS file and three JavaScript files that can be readily added to any project. The raw form of Bootstrap, however, enables developers to implement further customization and size optimizations. This raw form is modular, meaning that the developer can remove unneeded components, apply a theme and modify the uncompiled Sass files.

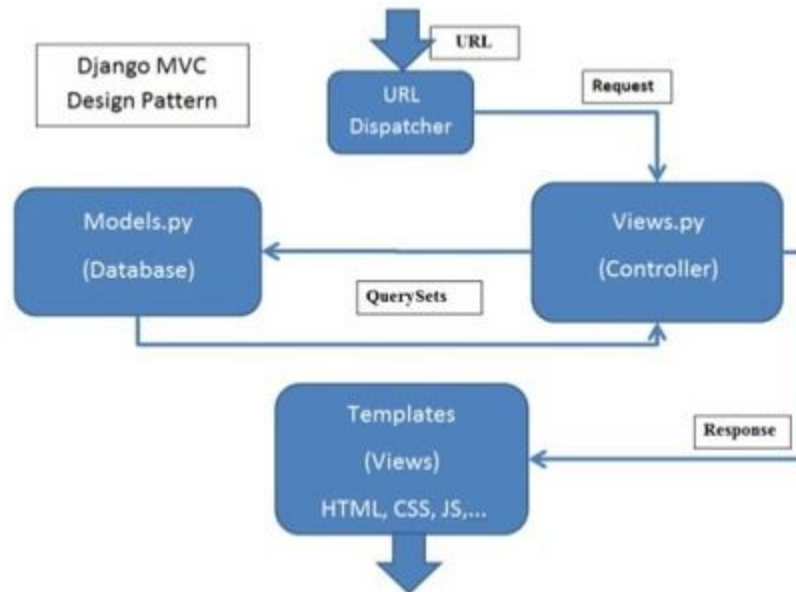
2.4 JavaScript

Although Mocking Weather relies on a Python based framework on server side to make it a dynamic website, JavaScript still remains the language of browser. Using JavaScript gave us the ability to add dynamic elements such as on hover and client side checking for forms and added functionality when used with Bootstrap. Latest standards of JavaScript as per EcmaScript were used to make sure speed and security standards were kept.

2.5 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so one can focus on writing their app without needing to reinvent the wheel. It's free and open source. Besides ensuring rapid development, Django ensures a secure and healthy environment for web development and also makes sure one can have a scalable architecture for building web apps in future. It also acts as a bridge between our database and website.

How things in Django works?



Django searches for the user request in the provided urls present in 'urls.py' module:

```
# urls.py
from .views import index, info_page, feedback, feedbacklist,
add_message

urlpatterns = [
    path('', index, name='home'),
    path('info/', info_page, name='info'),
    path('all-feedbacks/', feedbacklist),
    path('add-message/', add_message, name='message'),
    path('feedback/', feedback, name='feedback'),
]
```

From 'urls.py' the request is passed to 'views.py', which renders the required html page to the user request.

```
# views.py
def feedback(request):
```

```

    form = FeedbackForm(request.POST or None)
    if form.is_valid(): # To check the Validity of the user
input
        obj = form.save(commit=False)
        obj.save() # Inserting a new entry to the database
        form = FeedbackForm()

    context = {'form': form, 'title': 'Feedback'}
    return render(request, 'form.html', context)

```

Views further acts on the user requests and deals with database according to requirement by applying the queries.

In django module 'models.py' is responsible for creating tables and schemas.

```

# models.py

class Feedback(models.Model):
    name = models.CharField(max_length=30) # name VARCHAR(30)
    email = models.EmailField(null=True, blank=True) # email
EMAIL
    feedback = models.TextField(null=False, blank=False) #
feedback TEXT NOT NULL

```

Django also provide with a built in admin page, which allows us to handle all the databases manually with an easy interface. The admins page allow us to update, delete and even add new entries.

2.6 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an

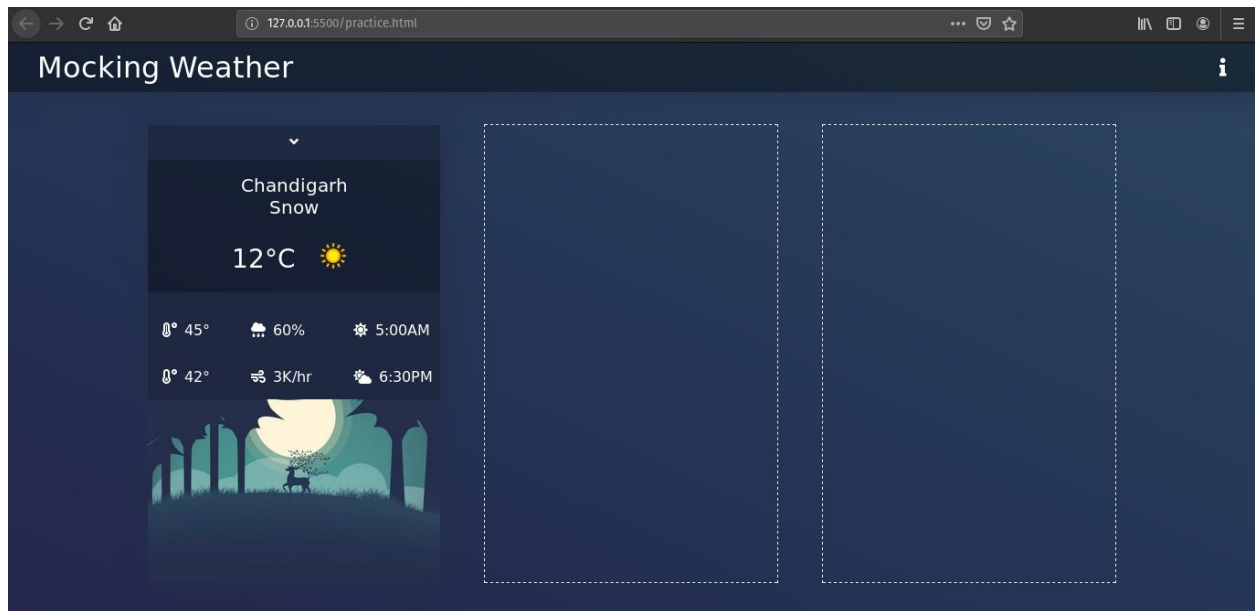
Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen().

Chapter 3: Snapshots

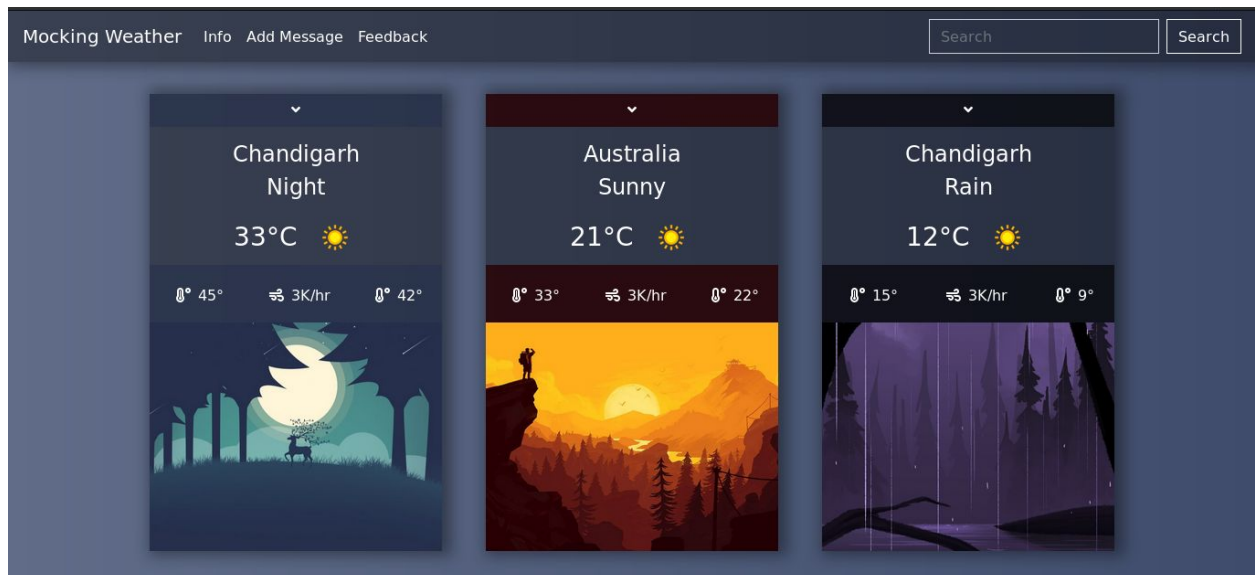
3.1 Introduction

The following section contains a list of screenshots to give a better idea of project.

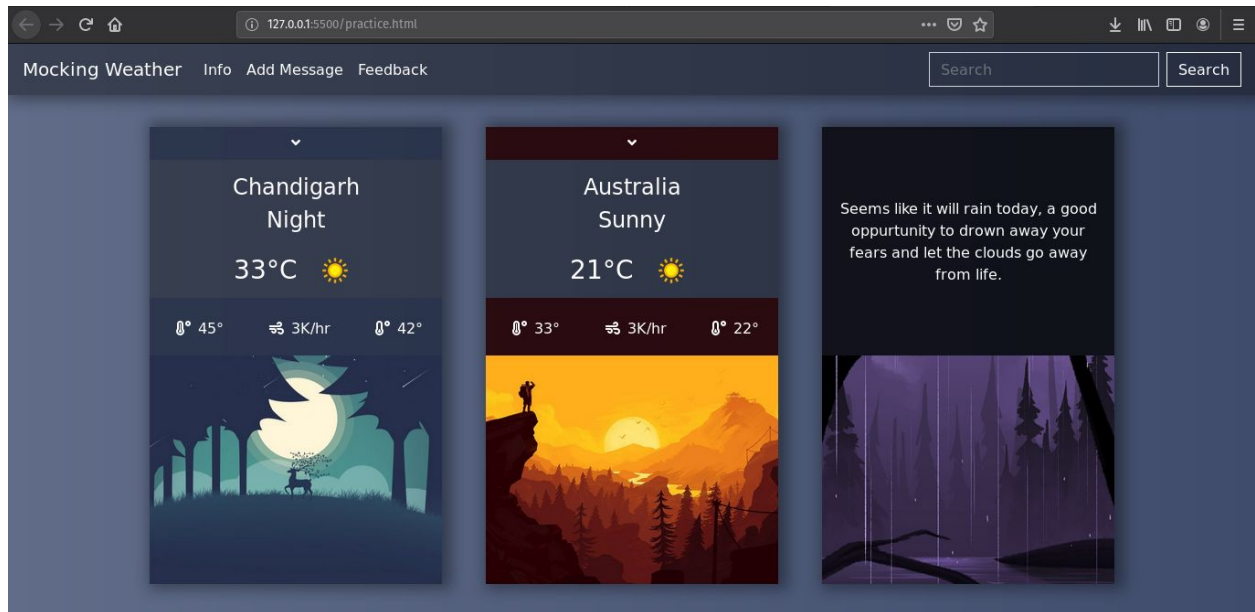
3.2 Front End: Landing Page



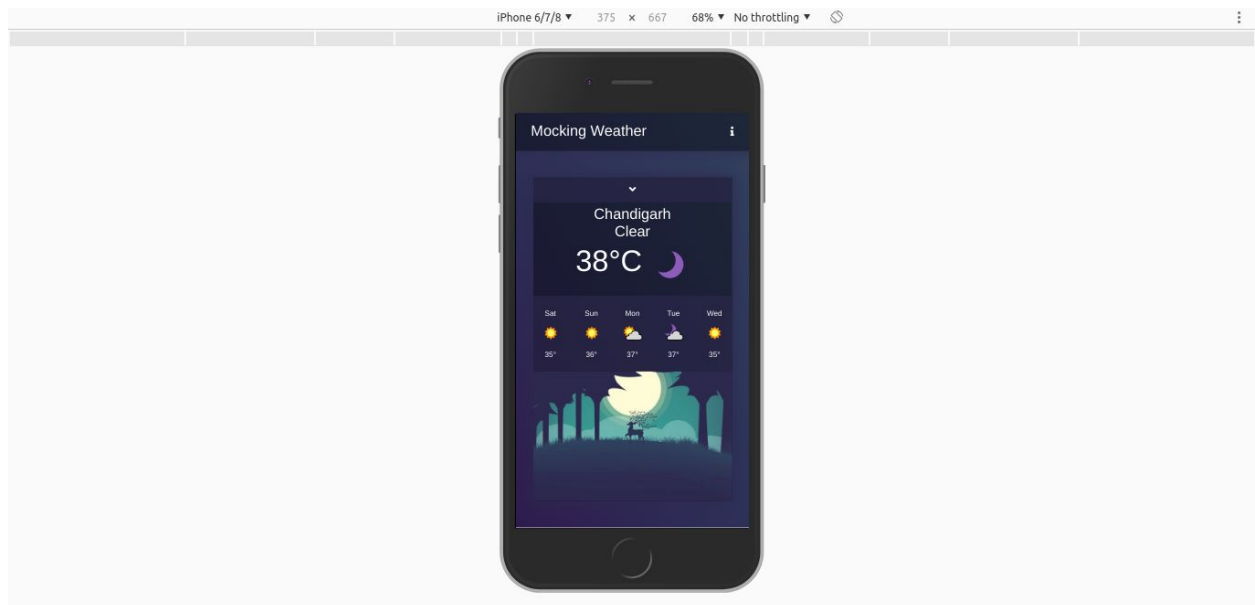
Multiple Cards



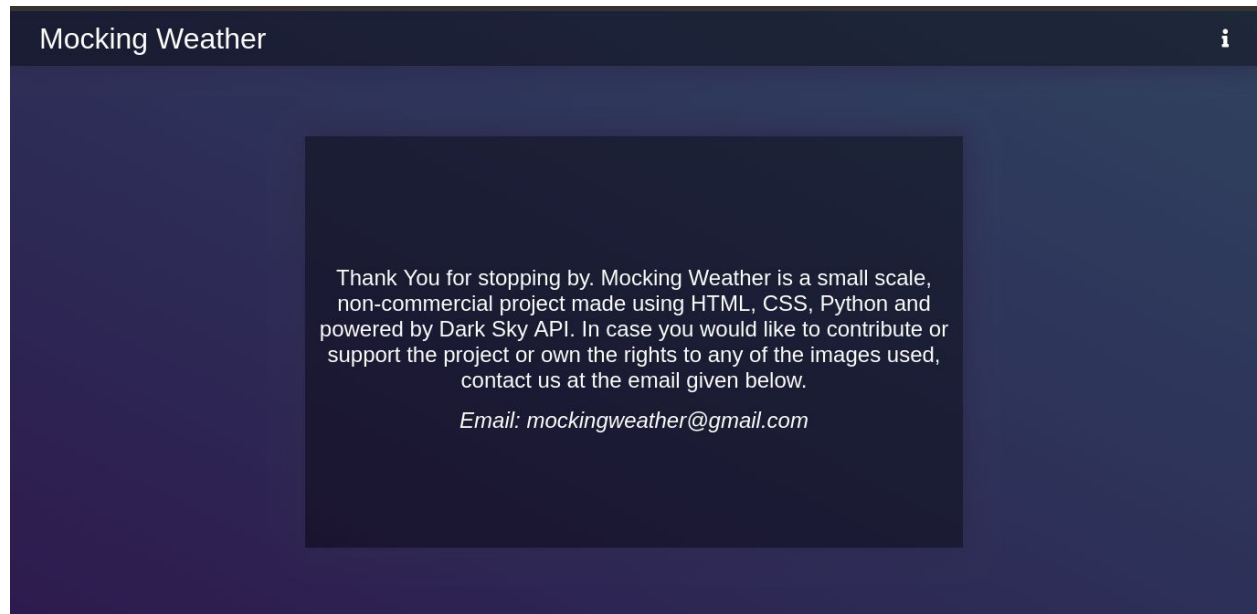
Sample Message



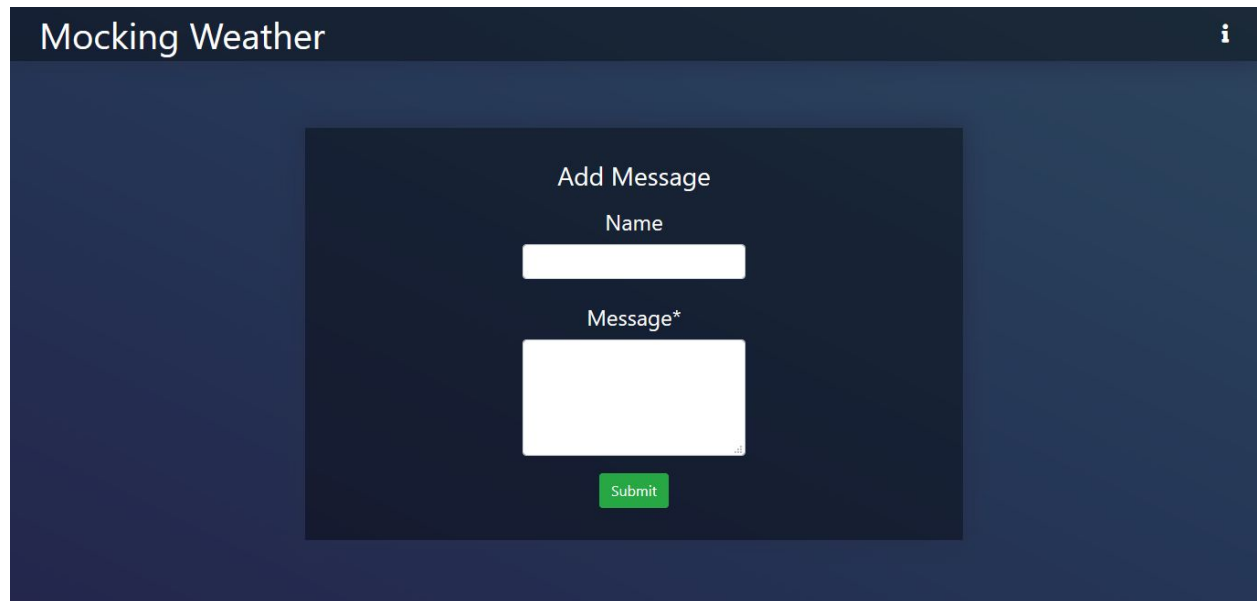
Mobile View



I-Page

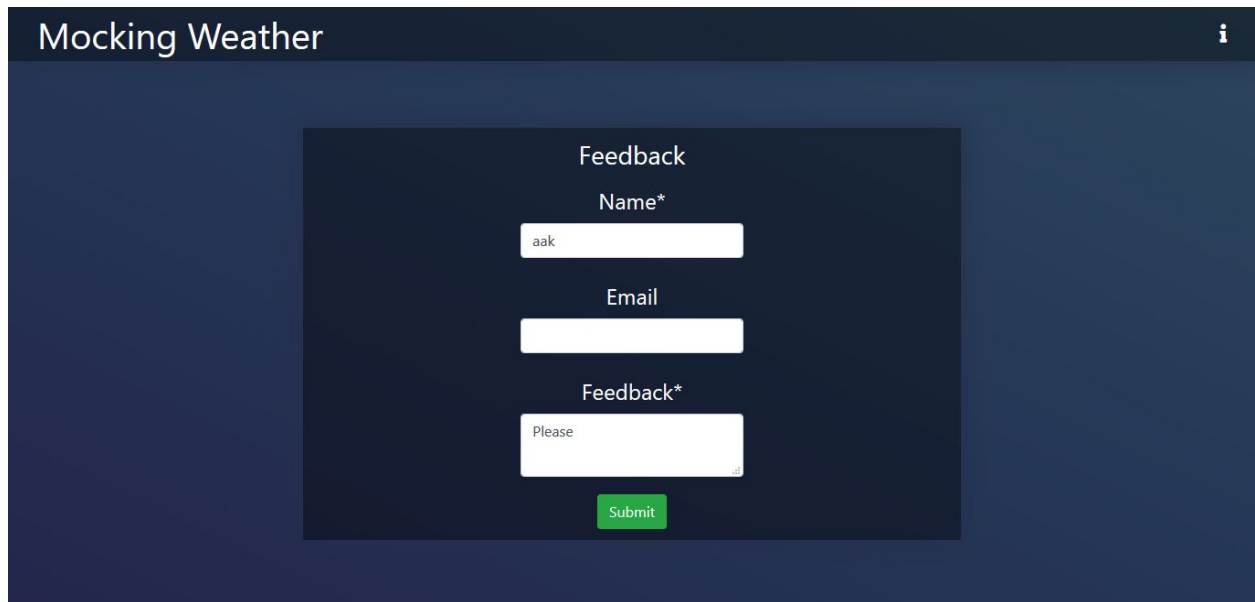


Add Message Form



A screenshot of a web page titled "Mocking Weather" in a dark header bar. The page has a dark blue background. In the center, there is a dark rectangular box containing a form titled "Add Message". The form has two input fields: "Name" and "Message*", both with white borders. Below the "Message*" field is a green "Submit" button with white text.

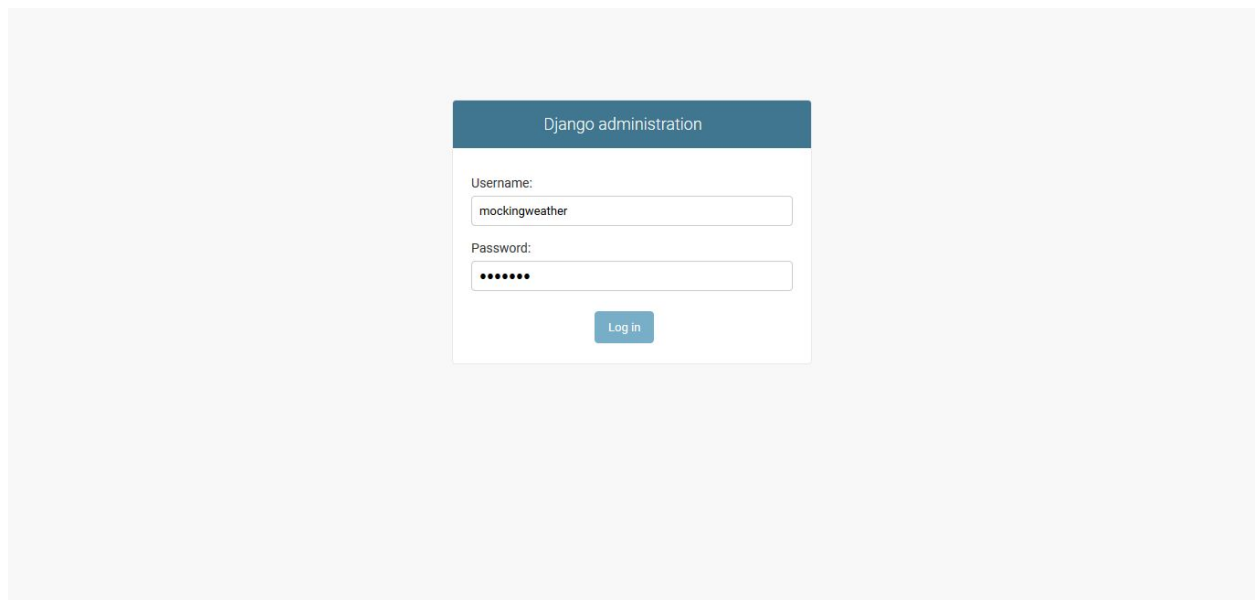
Feedback Form



The image shows a web application titled "Mocking Weather" with a dark blue header. In the top right corner of the header is a small white information icon. The main content area is a lighter blue. Centered in this area is a dark blue rectangular box containing a "Feedback" form. The form has the title "Feedback" at the top. Below it are three input fields: "Name*" with the text "aak", "Email", and "Feedback*" with the text "Please". At the bottom of the form is a green "Submit" button.

3.3 Django Databases

Admin Page Login



The image shows the Django administration login page. It has a light gray background. In the center is a white rectangular box with a blue header that says "Django administration". Below the header are two input fields: "Username:" with the text "mockingweather" and "Password:" with masked characters "••••••". At the bottom of the box is a blue "Log in" button.

Admin Page

Django administration

WELCOME, AAKRITI. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

Users

+ Add

Change

WEATHER

Cities

+ Add

Change

Feedbacks

+ Add

Change

Messages

+ Add

Change

Recent actions

My actions

City object (15)

City

Message object (2)

Message

City object (12)

City

City object (10)

City

City object (13)

City

City object (12)

City

City object (10)

City

City object (11)

City

City object (3)

City

City object (3)

City

Databases

Django administration

WELCOME, AAKRITI. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Weather > Cities

Select city to change

ADD CITY +

Action: Go 0 of 15 selected

☐

CITY

☐

City object (15)

☐

City object (14)

☐

City object (13)

☐

City object (12)

☐

City object (11)

☐

City object (10)

☐

City object (9)

☐

City object (8)

☐

City object (7)

☐

City object (6)

☐

City object (5)

☐

City object (4)

☐

City object (3)

☐

City object (2)

☐

City object (1)

15 cities

Django administration
WELCOME, AAKRITI. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home / Weather / Messages

Select message to change
ADD MESSAGE +

Action: Go 0 of 3 selected

<input type="checkbox"/>	MESSAGE
<input type="checkbox"/>	Message object (3)
<input type="checkbox"/>	Message object (2)
<input type="checkbox"/>	Message object (1)

3 messages

Django Database Management

Django administration
WELCOME, AAKRITI. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home / Weather / Feedbacks / Feedback object (1)

Change feedback
HISTORY

Name: aakriti

Email: aakriti01as@gmail.com

Feedback:

Please improve the User Interface

Delete
Save and add another
Save and continue editing
SAVE

Chapter 4: Results and Discussions

4.1 Result

By the end of this 4 week training program, we were able to come up with a proper working web application which was able to handle user requests and have a working flow for the application. While the basic idea of our application is quite basic yet it deals with all the components of a web application. From handling user requests, dynamically generating a web page and dealing with API calls, our application dealt with all. At the

we had a fully functioning product which is now live and can be accessed on mockingweather.pythonanywhere.com.

4.2 Discussion

The project dealt with all components of a web application which enables us to expand our skills in future. Further since this project incorporates several class leading frameworks such as Bootstrap and Django which are popular among the leading corporations in the industry, giving us an important and key achievement which can be added to one's resume.

Having developed a website which deals with most of the components of a web application gives us the ability to expand our skills to large scale with ease and build upon core foundation. Using a framework like Django makes it fairly easy to expand the scale of an application as the basics and base of applications remain the same. Bootstrap is one of the most popular piece of framework and using its components makes one prepared to build scalable websites from scratch.

API (Application Programming Interface) form a key role in web application, they enable different applications to interact with each other and share information. Here we used Open Sky Weather API to get weather data and display data on the website.

Chapter 5: Conclusion and Future Scope

To conclude it we had a weather app up and running by the end of 4th week of training. For future we plan to expand the web application to scale and support more data but still maintain our principle of simplifying the information and providing that information with ease. We also plant to add a Machine Learning algorithm to our message generating app which will Natural Processing of Language to generate more messages with ease and help build a database to help spread positivity among more people.

We also plan to add a login page and help deliver customizable information and user interface depending upon user's interests. This can further be linked with a business model by providing paid functionality and more design options to users.

References

- [FreeCodeCamp](#)
- [Try Django 2.2 - Udemy](#)
- [Django Documentation](#)
- [Bootstrap Documentation](#)
- [SQLite Documentation](#)
- [Open Weather API](#)
- [Mozilla Web Documentation](#)
- [Wikipedia](#)
- [Codecademy](#)
- [SoloLearn](#)