**DISCORD BOT FOR WEATHER FORECASTING**

* **DEPARTMENT:**

INFORMATION AND COMMUNICATION TECHNOLOGY**.**

* **TEAM:**
* SHANNON DIAS 190953017
* TEJAS ARORA 190953074
* ANKIT190953069
* **BATCH NUMBER:** 1

INTRODUCTION

The aim of our project is to create a user friendly weather forecasting Discord bot which takes in specific commands, interprets it and presents various attributes of the weather. We are going to use a number of tools, including the Discord API, Python libraries, and a cloud computing platform called Repl.it.

**Discord** is a free voice, video, and text chat app that's used by tens of millions of people ages 13+ to talk and hang out with their communities and friends. People use Discord daily to talk about many things, ranging from art projects and family trips to homework and mental health support.

**Bots** on Discord, the group messaging platform, are helpful artificial intelligence that can perform several useful tasks on your server automatically. That includes welcoming any new members, banning troublemakers, and moderating the discussion.

**JSON** is a lightweight data interchange format inspired by [JavaScript](https://en.wikipedia.org/wiki/JavaScript) object literal syntax. . It exposes an API familiar to users of the standard library marshal and pickle modules. It is the externally maintained version of the json library contained in Python 2.6, but maintains compatibility with Python 2.4 and Python 2.5 and has significant performance advantages, even without using the optional C extension for speedups.

An**API(Application Programming Interface)**is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange. Each API contains and is implemented by **function calls** – language statements that request software to perform particular actions and services.

**Weather Forecasting** is an important aspect in one’s everyday life and can help many individuals prepare their agenda and also their workflow for the day. Temperature forecasts are used by utility companies to estimate demand over coming days. Numerous individuals are extremely familiar with Discord and the Discord bots and how to use them, that discord bots would be a distinctive and simple way of getting the weather forecast.

In this project, we make use of the programming language Python and its libraries and also utilize certain api to configure a Discord bot to determine the weather and its features like temperature, condition, etc.

LITERATURE SURVEY

We have made use of certain Python libraries which are:

* **Discord.py**

A library for Python to aid in creating applications that utilize the Discord API.

-https://discordpy.readthedocs.io/

To gather and obtain the weather data, an API was used obtained from:

* **-**[**https://www.weatherapi.com/**](https://www.weatherapi.com/)

WeatherAPI.com provides access to weather and geo data via a JSON/XML restful API. It allows developers to create desktop, web and mobile applications using this data very easy.

METHODOLOGY AND IMPLEMENTATION

* **Discord bot creation** –

1. Make sure you’re logged on to the discord website

2. Navigate to the application page

3. Click on the “New Application” button.

4. Give the application a name and click “Create”.

5. Go to the “Bot” tab and then click “Add Bot”. You will have to confirm by clicking "Yes, do it!"

Keep the default settings for **Public Bot** (checked) and **Require OAuth2 Code Grant** (unchecked).

Your bot has been created. The next step is to copy the token.

This token is your bot's password so don't share it with anybody. It could allow someone to log in to your bot and do all sorts of bad things.

* **Integration of bot into discord server -**

Go to the "OAuth2" tab. Then select "bot" under the "scopes" section.

Now choose the permissions you want for the bot. Our bot is going to mainly use text messages so we don't need a lot of the permissions. You may need more depending on what you want your bot to do. Be careful with the "Administrator" permission.

After selecting the appropriate permissions, click the 'copy' button above the permissions. That will copy a URL which can be used to add the bot to a server.

Paste the URL into your browser, choose a server to invite the bot to, and click “Authorize”.

To add the bot, your account needs "Manage Server" permissions.

Now that you've created the bot user, we'll start writing the Python code for the bot.

* **API implementation -**

We use Python language along with real-time weather data using API request( [**http://api.weatherapi.com/v1/current.json?key=4cc2be672eca4a6282a165515210412**](http://api.weatherapi.com/v1/current.json?key=4cc2be672eca4a6282a165515210412&q)**).**

1. A menu is displayed on the discord channel showing the commands that the bot is able to interpret and carry out.
2. User is free to make commands for choices like basic and advanced data on weather forecasting and current weather conditions.
3. We take in the user-input command and based on the command we use Regex to match the command with predefined expressions in order to map the function to the command.
4. Each function creates and sends an http request to the API using requests module to get the required weather data in a JSON Format.
5. Before making the http request, placeholders of the url are fitted with values like the place taken in form user-input. The advanced functions take in the date as input value and add the date as a value to the placeholder(“&dt=”).
6. The response code is tested to see if our http request was successful and if not then “Incorrect arguments: is returned. The response which is in JSON format is converted to a dictionary format using loads attribute of JSON.
7. From the dictionary, we select the required key value pair and sift out only the weather data which is displayed as a string.
8. Based on the function, the Weather data returned could be detailed or rudimentary with an option to choose using the command specified in the menu.

CODE

import discord

import os

import re

import requests

import json

import datetime

# Function to validate the date

def validate(date\_text):

try:

datetime.datetime.strptime(date\_text, '%Y-%m-%d')

except ValueError:

return False

else:

return True

# Creating regular expressions to detect commands

redexc = '\$current.\*'

redexf ='\$forecast.\*'

redexca = '\$advanced current.\*'

redexfa ='\$advanced forecast.\*'

# Functions for displaying current weather conditions

def current(place):

if(place.isalpha()==False):

return "🚫Incorrect arguments!"

place = place.title()

var = "http://api.weatherapi.com/v1/current.json?key=4cc2be672eca4a6282a165515210412&q="

var = var + place

respons = requests.get(var)

if (respons.status\_code >299):

return "🚫Incorrect arguments!"

# In API if our request is correct the status code is less than 300 and if it's wrong then the value of the status code is 300 and more depending upon the error

abc = respons.text

res = json.loads(abc)

# Response.text is the json file we are converting it to dictionary here

# Obtaining the right data and returning as string

curr = res['current']

out = "Last recorded : " + curr['last\_updated'] + "\nTemp: "+ str(curr['temp\_c']) +u'\N{DEGREE SIGN}' +"C\nCondition: " + curr['condition']['text']

return out

# Returns basic weather forecast for any date in the next 10 days

def forecast(place,date):

if(place.isalpha()==False):

return "🚫Incorrect arguments!"

if(validate(date)==False):

return "Wrong date"

var = "http://api.weatherapi.com/v1/forecast.json?key=4cc2be672eca4a6282a165515210412&q="

place = place.title()

var = var + place

var = var +"&dt="

var = var + date

respons = requests.get(var,date)

if (respons.status\_code >299):

return "🚫Incorrect arguments!"

abc = respons.text

res = json.loads(abc)

a = res['forecast']['forecastday'][0]

fore = a['day']

out = "Max Temp: "+ str(fore['maxtemp\_c']) +u'\N{DEGREE SIGN}'+"C\nMin Temp: "+ str(fore['mintemp\_c']) + u'\N{DEGREE SIGN}'+"C\nCondition: " + fore['condition']['text']

return out

# Returns the current weather conditions in a detailed manner

def currentadvance(place):

if(place.isalpha()==False):

return "🚫Incorrect arguments!"

place = place.title()

var = "http://api.weatherapi.com/v1/current.json?key=4cc2be672eca4a6282a165515210412&q="

var = var + place

respons = requests.get(var)

if (respons.status\_code >299):

return "🚫Incorrect arguments!"

abc = respons.text

res = json.loads(abc)

curr = res['current']

out = "Last recorded : " + curr['last\_updated'] + "\nTemp: "+ str(curr['temp\_c']) +u'\N{DEGREE SIGN}' +"C\nFeels like: "+ str(curr['feelslike\_c'])+ u'\N{DEGREE SIGN}'+ "C\nCondition: " + curr['condition']['text'] +"\nWindspeed :" + str(curr['wind\_kph']) +" Km/h\nWind direction:" + curr['wind\_dir'] +"\nPrecipitaion:"+ str(curr['precip\_mm']) + " mm" +"\nHumidity:" + str(curr['humidity']) + "%"

return out

# Returns the forecasted weather conditions in a detailed manner

def forecastadvance(place,date):

if(place.isalpha()==False):

return "🚫Incorrect arguments!"

if(validate(date)==False):

return "Wrong date"

var = "http://api.weatherapi.com/v1/forecast.json?key=4cc2be672eca4a6282a165515210412&q="

place = place.title()

var = var + place

var = var +"&dt="

var = var + date

respons = requests.get(var,date)

if (respons.status\_code >299):

return "🚫Incorrect arguments!"

abc = respons.text

res = json.loads(abc)

a = res['forecast']['forecastday'][0]

fore = a['day']

out = "Max Temp: "+ str(fore['maxtemp\_c']) +u'\N{DEGREE SIGN}'+"C\nMin Temp: "+ str(fore['mintemp\_c']) + u'\N{DEGREE SIGN}'+"C\nCondition: " + fore['condition']['text'] + "\nAverage Windspeed :" + str(fore['avgvis\_km']) + "\nChance of rain: "+ str(fore['daily\_chance\_of\_rain']) + "%\nAverage humidity: " + str(fore['avghumidity']) + "%"

return out

client = discord.Client()

@client.event

async def on\_ready():

print('We have logged in as {0.user}'.format(client))

# Console message log

@client.event

async def on\_message(message):

username = str(message.author).split('#')[0]

user\_message = str(message.content)

channel = str(message.channel.name)

print(f'{username}: {user\_message} ({channel})')

if message.author == client.user:

return

# Reading the user's command and executing the respective functions

if message.channel.name == 'general':

if user\_message.lower() == '$hello':

out = f"Hello {username}! 👋" + "\n I am at your service \n These are my commands " + " :\n1️⃣ $forecast city date(YYYY-MM-DD)\n2️⃣ $current city \n3️⃣ $advanced forecast city date(YYYY-MM-DD)\n4️⃣ $advanced current city\n5️⃣ $hello\n6️⃣ $bye"

await message.channel.send(out)

return

if user\_message.lower() == '$bye':

await message.channel.send(f'See you later {username}! 🤝')

return

if(re.match(redexc,user\_message)):

abc = user\_message.split()

loc = abc[1]

out = current(loc)

await message.channel.send(out)

return

if(re.match(redexca,user\_message)):

abc = user\_message.split()

loc = abc[2]

out = currentadvance(loc)

await message.channel.send(out)

return

if(re.match(redexf,user\_message)):

abc = user\_message.split()

loc = abc[1]

t = abc[2]

out = forecast(loc,t)

await message.channel.send(out)

return

if(re.match(redexfa,user\_message)):

abc = user\_message.split()

loc = abc[2]

t = abc[3]

out = forecastadvance(loc,t)

await message.channel.send(out)

return

if user\_message.lower() == '!anywhere':

await message.channel.send(user\_message)

return

# Secret key(token) has been imported

my\_secret = os.environ['token']

client.run(os.environ['token'])

RESULT

The Discord bot works in the discord channel ‘general’ taking in a ‘$hello’ command and displays a greeting message with the user’s username and with the choice of essential commands i.e. **$forecast city**, **$current city**, **$advanced forecast city (YYYY-MM-DD)**, **$advanced current city (YYYY-MM-DD)** and two greeting commands **$hello** and **$bye**. The $bye command commands the bot to display a farewell message along with the user’s username.

The user can enter the name of the city in the commands in any case since it is case-insensitive.

For the **$forecast city** command, after interpreting the command and sending the request to API, the various attributes i.e. maximum and minimum temperature and current condition regarding the current weather forecast in the city in the last 10 days is displayed by the bot.

The advanced command **$advanced forecast** **city** when chosen, the bot takes in the “place” as the parameter and also the date, displaying complex weather conditions like the current average wind speed, average humidity, chance of rain, etc. of the date that was specified by user.

The bot display basic weather conditions of the city that the user has entered like the current temperature and condition as obtained from the API when the user selects the **$current city** command.

**$advanced current city** command directs the bot to call the current advanced function to take in date and place as parameters, displaying the advanced and detailed weather conditions like the humidity, precipitation, etc. of any date which is specified by the user.

**BOT OUTPUTS –**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

CONCLUSION

According to Discord*,* around 430,000 bots are used every week by the 150 million monthly active Discord users. For the longest time, bots have been something that Discord has supported, but not properly embraced.

Similarly, through this project we incorporated Discord and the Discord bots to shed light on the applications and the latent potential the Discord domain has. A simple discord command from the user can forecast the weather conditions in a jiffy adding to the simple and uncomplicated user interface, hence being versatile and easy to use.

The project has helped us learn extensively how to use and interpret python syntax and its libraries. It has given us a better and clear understanding on the working and use of APIs and their applications. Various python modules like requests, os, JSON helped us to access the data from the API and also configure it according to our necessary needs. We got more familiar with the working and functioning of Discord and the Discord bots, which give us more confidence to drive forward and try newer, sophisticated methods to make bots with more features and characteristics.

We used most of our Python knowledge in the application, including classes, functions, modules, dictionaries, along with additional knowledge mentioned above. The learning experience has thus made us more confident to create and apply real-world applications.