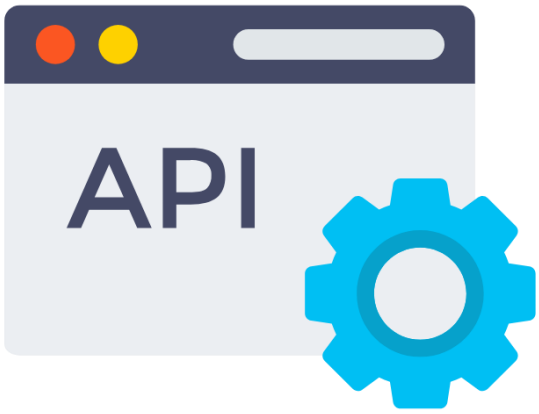
SVKM'S
NMIMS
Deemed to be UNIVERSITY



API WEATHER PROGRAM

2022 - 2023

ABSTRACT

This report is a fragment of the assignments that have to be submitted to Prof. Sachin Bhandari for partial fulfilment of Internal Continuous Assessments (ICA), as per the course policy. Our topic for the report is "API Weather Program", in which we utilised APIs to retrieve weather data of a particular location and save it to a MySQL database and into a CSV file.

Submitted by:

Sanyam Jain (N229)
Daksh Gehlot (N230)
Gautam Kundalia (N243)

INDEX

INTRODUCTION

API

An API, which stands for Application Programming Interface, is a software intermediary which allows two application to communicate with each other. Each time you use an app like Instagram, send an instant message, or check the weather on your phone, you're using an API.

To simplify, an API delivers a user response to a system and sends the system's response back to a user. It enables this process to be completed efficiently and without any huge delays. Modern APIs adhere to standards (like HTTP) that are developer friendly, easily accessible and broadly understood, and are used by major companies like Google, eBay, Amazon and Expedia.

A website, which itself uses an API to complete requests, is displaying what an API is.

PURPOSE AND USE

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

Imagine a waiter at a café. You want to order something from the menu and the kitchen has the cooks that will provide you with your favorite meal. You need a link to communicate your order with the kitchen and deliver your food back to your table. That's where the waiter – or the API – enters the picture.

PROBLEM STATEMENT AND SOLUTION

Problem Statement:
It is only natural that sometimes a person might want to find the weather of a particular location. This area can be anywhere around the globe – not only in the country. Also, they might want the weather data to be saved so that they can refer to it later.

Proposed Solution:
The program that we implemented uses an API to retrieve weather data for any particular location. On running the program, the user is asked if they want the current weather data, or data for yesterday or tomorrow. The respective info will be retrieved when the user selects the option. After that, they will be given a choice to see a weather forecast graph for the next 7 days. The graph can be viewed in Celsius as well as Fahrenheit.

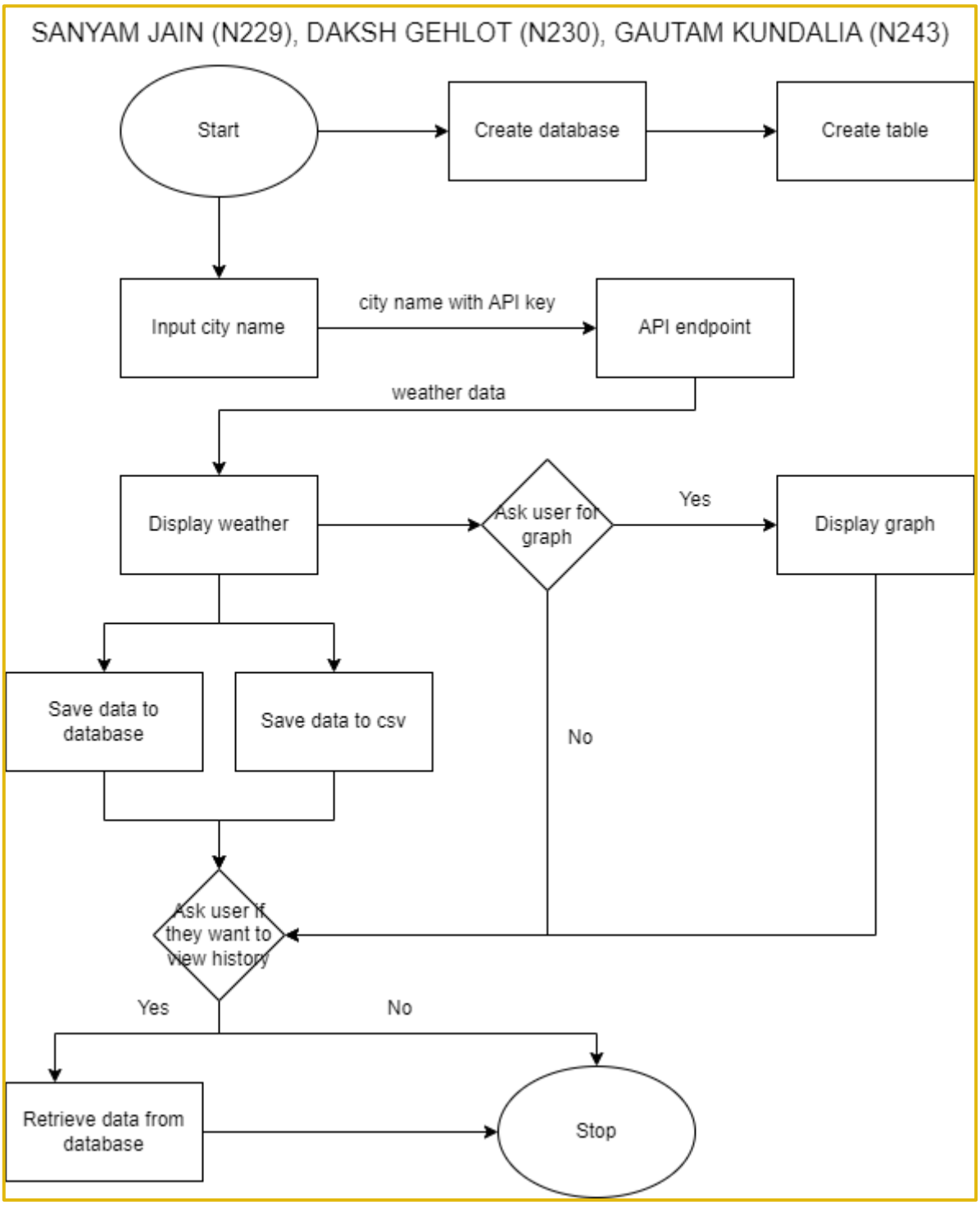
The weather data originally retrieved is saved in a MySQL database as well as in a CSV file. The user is given an option to retrieve this data if they would like to go through the history.

Hardware Requirements:
Intel i3 3rd gen or higher or AMD Ryzen 3 1600 or higher

Software Requirements:
Windows 7, 8, 10, 11 (32 or 64 bit), Visual Studio Code / PyCharm, API Key, MySQL database

SOFTWARE ARCHITECTURE DIAGRAM

Diagram link: [Software-Architecture-Diagram]



MODULES USED

1. **Requests module**: The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

2. **datetime module**: In Python, date and time are not a data type of its own, but a module named datetime can be imported to work with the date as well as time. It also combines date and time information.

3. **time module**: This module provides various time-related functions, which gives us many ways of representing time in code, such as objects, numbers, and strings. It refers to the time independent of the day (hour, minute, second, microsecond).

4. **JSON module:** JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. It's used by lots of APIs and databases, and it's easy for both humans and machines to read. JSON represents objects as name/value pairs, just like a Python dictionary. It parses return objects in human readable form.

5. **matplotlib.pyplot**: Matploitlib is a Python Library used for plotting graphs, this python library provides with objected-oriented APIs for integrating plots into applications. matplotlib.pyplot is a plotting library used for 2D graphics in python.

6. **geocoder module**: It allows the user to access the coordinates of a location using its name, and vice versa. It is helpful in identifying and determining the exact location of a place, with good accuracy.

7. **mysql.connector:** Python needs a MySQL driver to access the MySQL database. This library provides Python with the access to MySQL to execute queries. Queries can be executed to create, read, update and delete data.

8. **csv:** The csv module implements classes to read and write tabular data in CSV format. It allows programmers to write data in the format preferred by Excel or read data from a file which was generated by Excel, without knowing the precise details of the CSV format used by Excel.

9. **numpy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

THE SOURCE CODE

Below starts the source code of our API program:

```
#A program by Sanyam Jain (N229), Daksh Gehlot (N230) and Gautam Kundalia (N243)
import requests
import json
import time
import datetime
import geocoder
import matplotlib.pyplot as plt
import mysql.connector as mcon
import csv
import numpy as np

print()
print("Request pulled on:")
print(time.asctime())
a = datetime.date.today()

pswm = input("PLease enter your mysql password for connectivity: ")
mcon=mcon.connect(host='localhost', user='root', passwd=pswm)
cursor=mcon.cursor()

x=input("Do you want the database to be created automatically (Y/N)? ")
while True:
    if x in ("Y","y"):
        while True:
            datb = input("Please enter the preferred name of the database in which the
table should exist: ")
            h = input("Are you sure that the above entered info is correct? (Y/N): ")
            if h in ('Y','y'):
                print('Good job, moving on...')
                print()
                break
            elif h in ('N', 'n'):
                print("Give it another try!")
            else:
                print("Invalid choice, try again!")
                print()
        datn=datb.replace(" ", "_")
        cursor.execute('create database {}'.format(datn))
        cursor.execute('use {}'.format(datn))
    elif x in ("N", "n"):
        while True:
            datb=input("Enter the name of an existing database in which you want the
data to be saved: ")
            h = input("Are you sure that the above entered info is correct? (Y/N): ")
            if h in ('Y','y'):
                print('Good job, moving on...')
                print()
                break
            elif h in ('N', 'n'):
                print("Give it another try!")
            else:
                print("Invalid choice,try again!")
                print()
```

```
        datn=datb.replace(" ", "_")
        cursor.execute('use {}'.format(datn))

    else:
        print("Invalid choice, try again!")
    break

lapi = '33b327f0ca624ce089892bd897843bb0'
wapi = '16ff12c7ba9d5ea26a00939999d81ee2'

cursor.execute('show tables;')
st=cursor.fetchall()
while True:
    if st == [('weather_report',)]:
        print("The table already exists, nice!")
        s = 0
        break
    else:
        print("Would you like the MySQL table to be created automatically by the
program?")
        print("Note: If you decline, no values will be written in the database!")
        x=input("Enter 'Y/y' to accept, or 'N/n' to decline: ")
        if x in ("Y","y"):
            cursor.execute('''create table weather_report(
            Date varchar(10) not null not null,
            Current_Temperature_CELSIUS decimal(4,2) not null,
            Feels_Like_CELSIUS decimal(4,2) not null,
            Humidity int(2) not null,
            Time varchar(5) not null,
            City varchar(15) not null)''')
            s = 0
            print()
            break
        elif x=="N" or x=="n":
                print("The table doesn't exist, no values will be written in
database!!")
                s = 1
                break
        else:
            print("Please enter a valid choice")

cursor.reset()
if s == 0:
    if mcon.is_connected():
        print('Database Connection Succesful')
        print()
        h = input("Do you want to know the description of the table (Y/N): ")
        print()
        if h in ('Y', 'y'):
            cursor.execute("Desc weather_report")
            print('Description of Database is')
            print()
            for x in cursor.fetchall():
                print(x)
    else:
        print('Database Connection Error')

def coord(city):
    global result
    result = geocoder.opencage(city, key=lapi)
    print("")
    #print(result)
    global lat
    global lon
    try:
        lat, lon = result.latlng[0], result.latlng[1]
```

```
        return lat, lon
    except TypeError:
        print("Enter a valid city!")

def cel(val):
    cels = round((val - 273.15), 2)
    return cels

def far(val):
    faren = round((val * (9/5) - 459.67), 2)
    return faren

def curr_weather():
    l = 3
    while l < 8:
        global city
        city = str.capitalize(input("Enter City name: "))
        coord(city)
        if result.latlng == None:
            print("Invalid City name, try again...")
        else:
            resp =
requests.get('https://api.openweathermap.org/data/2.5/weather?q={}&appid={}'
                                .format(city, wapi))
            if resp.status_code == 200:
                t = json.loads(resp.text)
                print("The coordinates of entered location are: \n", coord(city))
                print("-------------------------")
                print()
                print("::::::The Current Weather of {} is::::::".format(city))
                global temp
                global hum
                global feel
                hum = t['main']['humidity']
                temp= (cel(t['main']['temp']),far(t['main']['temp']))
                feel = (cel(t['main']['feels_like']), far(t['main']['feels_like']))
                print()
                print("■ Today's average temperature:", temp[0], '℃', 'or', temp[1],
'℉')
                print('■ Feels Like:', feel[0], '℃', 'or', feel[1], '℉')
                print('■ Humidity:', hum, '%')
                l += 2
                break
            elif resp.status_code == 400:
                print("Server error \n trying again in", l, "seconds...")
                time.sleep(l)
                l += 1
            if l == 7:
                print("Too many errors, check your internet connection and arguments
given and try again")
                break

def yest_weather():
    l = 3
    while l < 8:
        global city
        city = str.capitalize(input("Enter City name: "))
        dt = int((time.time()) - 86400)
        coord(city)
        if result.latlng == None:
            print("Invalid City name, try again...")
        else:
            resp =
requests.get('https://api.openweathermap.org/data/2.5/onecall/timemachine?lat={}\
                                &lon={}&dt={}&appid={}'.format(lat, lon, dt, wapi))
```

```
            if resp.status_code == 200:
                t = json.loads(resp.text)
                print("The coordinates of entered location are: \n", coord(city))
                print("-------------------------")

                print()
                print(":::::The Yesterday's Weather of {} is:::::".format(city))
                global temp
                global feel
                global hum
                temp = (cel(t['current']['temp']), far(t['current']['temp']))
                feel = (cel(t['current']['feels_like']),
far(t['current']['feels_like']))
                hum = t['current']['humidity']
                print()
                print("■ Yesterday's weather: ", temp[0], '℃', 'or', temp[1], '℉')
                print('■ Felt Like:', feel[0], '℃', 'or', feel[1], '℉')
                print('■ Humidity:', hum, '%')
                l += 2
                break
            elif resp.status_code == 400:
                print("Server error \n trying again in", l, "seconds...")
                time.sleep(l)
                l += 1
            if l == 7:
                print("Too many errors, check your internet connection and arguments
given and try again")
                break

def tomm_weather():
    l = 3
    while l < 8:
        global city
        city = str.capitalize(input("Enter City name: "))
        coord(city)
        if result.latlng == None:
            print("Invalid City name, try again...")
        else:
            resp =
requests.get('https://api.openweathermap.org/data/2.5/onecall?lat={}&lon={}&appid={}'
                                 .format(lat, lon, wapi))
            if resp.status_code == 200:
                t = json.loads(resp.text)
                print("The coordinates of entered location are: \n", coord(city))
                print("-------------------------")
                print()
                print(":::::The Weather Forecast of {} is:::::".format(city))
                print()
                global temp
                global feel
                global hum
                temp = (cel(t['daily'][0]['temp']['day']),
far(t['daily'][1]['temp']['day']))
                feel = (cel(t['daily'][0]['feels_like']['day']),
far(t['daily'][1]['feels_like']['day']))
                hum = t['daily'][0]['humidity']
                print("■ Tomorrow's weather will be: ", temp[0], '℃', 'or', temp[1],
'℉')
                print("■ It will feel like: ", feel[0], '℃', 'or', feel[1], '℉')
                print("■ Humidity will be: ", hum, '%')
                l += 2
                break
            elif resp.status_code == 400:
                print("Server error \n trying again in", l, "seconds...")
                time.sleep(l)
```

```
                l += 1
            if l == 7:
                print("Too many errors, check your internet connection and arguments
given and try again")
                break

def graph():
    while True:
        g = input("Do you want forecast graph for next 7 days? (Y/N): ")
        if g in ('Y', 'y'):
            days = []
            temps = []
            mintemps = []
            maxtemps = []
            info =
requests.get("https://api.openweathermap.org/data/2.5/onecall?lat={}&lon={}&appid={}"
                                  .format(lat, lon, wapi))
            info = info.text
            data = json.loads(info)
            print("Enter the unit you want the graph in \n▶for CELSIUS, enter C/c.
\n▶for FAHRENHEIT,\
 enter F/f.")
            while True:
                unit = input("Enter your choice (C/F): ")
                if unit in ('C', 'c'):
                    plt.ylabel("Temperature (in celsius)")
                    break
                elif unit in ('F', 'f'):
                    plt.ylabel("Temperature (in fahrenheit)")
                    break
                else:
                    print("Invalid choice try again....")
            print()
            for i in range(7):
                day =
datetime.datetime.fromtimestamp(data["daily"][i]['dt']).strftime("%A")
                if unit in ('C', 'c'):
                    temp = cel(data['daily'][i]['temp']['day'])
                    mintemp = cel(data['daily'][i]['temp']['min'])
                    maxtemp = cel(data['daily'][i]['temp']['max'])
                elif unit in ('F', 'f'):
                    temp = far(data['daily'][i]['temp']['day'])
                    mintemp = far(data['daily'][i]['temp']['min'])
                    maxtemp = far(data['daily'][i]['temp']['max'])
                days.append(day)
                temps.append(temp)
                maxtemps.append(maxtemp)
                mintemps.append(mintemp)
            print("▶    Here is the graph:")
            plt.xlabel("Days")
            plt.plot(days, temps, 'c', label='Average temperature', marker='o')
            plt.plot(days, maxtemps, 'r', label='Maximum temperature', marker='o')
            plt.plot(days, mintemps, 'b', label='Minimum temperature', marker='o')
            plt.title("The weather forecast of {} for next 7 days".format(city))
            plt.legend()
            plt.show()
            print()
            break
        elif g in ("n", "N"):
            print()
            print("Okay")
            print()
            break
        else:
            print("Invalid Choice, Try again...")
```

```
print()
#Display Data From Database
def data_from_database():
    cursor.execute('Select * from weather_report order by Time desc')
    n=int(input('Enter the number of past records you want to see: '))
    print()
    data=cursor.fetchmany(n)
    count=cursor.rowcount
    print(data)
    print()
    print('No of Records: ',count)
    cursor.reset()
    print()
print()
# Display Data From Database by city
def data_from_city():
    a=input('Which city\'s record do you want to see? ')
    cursor.execute('Select * from weather_report where city="{}"'.format(a))
    data=cursor.fetchall()
    count=cursor.rowcount
    print()
    print(data)
    print()
    print('No of Records: ',count)
    print()
    cursor.reset()
print()
# Dislplay Highest Temperature in record
def highest_Temp():
    cursor.execute('Select * from weather_report order by Current_Temperature_CELSIUS
DESC')
    data=cursor.fetchmany(1)
    count=cursor.rowcount
    print(data)
    print()
    print('No of Records: ',count)
    cursor.reset()
    print()
print()
# No of Data according to cities
def no_of_cities_data():
    cursor.execute('Select city,count(*) from weather_report group by City')
    data=cursor.fetchall()
    count=cursor.rowcount
    print(data)
    print()
    print('No of Records: ',count)
    cursor.reset()
    print()

y=time.strftime('%H:%M')

def insert():
    cursor.execute("INSERT INTO weather_report(Date, Current_Temperature_CELSIUS,
Feels_Like_CELSIUS,\
                   Humidity, Time, City) VALUES('{}',{},{},{},'{}','{}')"
                   .format(tp1, temp[0], feel[0], hum, y,city.upper()))
    mcon.commit()
    print('(Data successfully saved in the database)')
    cursor.reset()

while True:
    print("What information do you want? \n● 1.Current weather \n● 2.Yesterday's
weather\
          \n● 3.Tomorrow's weather")
    print()
```

```
    op = input("Enter your choice (1, 2 or 3): ")
    if op == '1':
        print("You chose for today's weather...")
        #global tp1
        tp1 = a.strftime("%d-%m-%y")
        print("◖ Today's date is", tp1, " ◗")
        curr_weather()
        print()
        if s == 0:
            insert()
        graph()
        break
    elif op == '2':
        print("You chose for yesterday's weather...")
        #global tp1
        tp2 = datetime.date.today() - datetime.timedelta(days=1)
        tp1 = tp2.strftime("%d-%m-%Y")
        print("◖ Yesterday's date was:", tp1, ' ◗')
        yest_weather()
        print()
        if s == 0:
            insert()
        graph()
        break
    elif op == '3':
        #global tp1
        tp4=datetime.date.today() + datetime.timedelta(days=1)
        tp1 = tp4.strftime("%d-%m-%Y")
        print("You chose for tomorrow's weather...")
        print("◖ The date tomorrow is:", tp1, ' ◗')
        tomm_weather()
        print()
        if s == 0:
            insert()
        graph()
        break
    else:
        print("INVALID CHOICE, Try Again...")
if s == 0:
    while True:
        print('''If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5''')
        x=int(input('Enter Your Choice: '))
        if x==1:
            print()
            print("The order of the columns is")
            print('Date,', 'Current_Temperature_CELSIUS,',
'Feels_Like_CELSIUS,','Humidity,',\
                  'Time,', 'City')
            data_from_database()
        elif x==2:
            data_from_city()
        elif x==3:
            no_of_cities_data()
        elif x==4:
            highest_Temp()
        elif x==5:
            print('Thank You')
            break
        else:
            print('Enter Valid Choice')
    mcon.close()
```

USER MANUAL

(FOR AUTOMATIC CREATION OF THE DATABASE AND THE TABLE)

The API Weather Code that we have built, includes a function to automatically create a database and the table in it in MySQL to store the data retrieved by the program, provided that the user gives the correct password to their MySQL databases. However, the users have the option to decline the requests and create a database and table on their own (please see page number 17).
Below is the guide for automatic creation of the database and table:

1) On running the code, users will be able to see the current data and time, and a request asking them to enter their MySQL password. Be sure to enter the correct one. After that, they will be asked whether they want the database to be created automatically or not. Press "Y/y" for yes and "N/n" for no, and then confirm the entered info:

```
Request pulled on:
Thu Oct 13 01:36:54 2022
Please enter your mysql password for connectivity: rick@astley
Do you want the database to be created automatically (Y/N)? y
Please enter the preffered name of the database in which the table should exist: python_project
Are you sure that the above entered info is correct? (Y/N): y
Good job, moving on...
```

2) Now the user has the choice to either allow the program to create a table automatically or decline it. If a table is not present, then the data won't be saved. On choosing "Yes", the table would be created, and a message will be displayed confirming the

connection to the database. Users also have the capacity to view the description of the table if they wish to do so:

```
Would you like the MySQL table to be created automatically by the program?
Note: If you decline, no values will be written in the database!
Enter 'Y/y' to accept, or 'N/n' to decline: y

Database Connection Succesful

Do you want to know the description of the table (Y/N): y

Description of Database is

('Date', b'varchar(10)', 'NO', '', None, '')
('Current_Temperature_CELSIUS', b'decimal(4,2)', 'NO', '', None, '')
('Feels_Like_CELSIUS', b'decimal(4,2)', 'NO', '', None, '')
('Humidity', b'int', 'NO', '', None, '')
('Time', b'varchar(5)', 'NO', '', None, '')
('City', b'varchar(15)', 'NO', '', None, '')
```

In case they decide to create the database and the table manually, that is, they choose “No” in either or both of the steps 1 and 2, the instructions given on the next page may be followed.

NOTE: You also need a stable internet connection for this code to work.

USER MANUAL

(FOR MANUALLY CREATING THE DATABASE AND THE TABLE)

There are certain steps that we must know before going further. First of them is to create a MySQL database to store the records.

1) Open MySQL.

2) Create a database by entering the code "create database python_project;" (or any name of your choice).

3) Now change the database to this newly created one by typing "python_project;".

4) Create the table "weather_report" by entering the below queries:

```
create table weather_report(
Date varchar(10) not null not null,
Current_Temperature_CELSIUS decimal(4,2) not null,
Feels_Like_CELSIUS decimal(4,2) not null,
Humidity int(2) not null,
Time varchar(5) not null,
City varchar(15) not null);
```

5) Your MySQL database is now ready.

NOTE: You also need a stable internet connection for this code to work.

OUTPUTS AND RESULTS

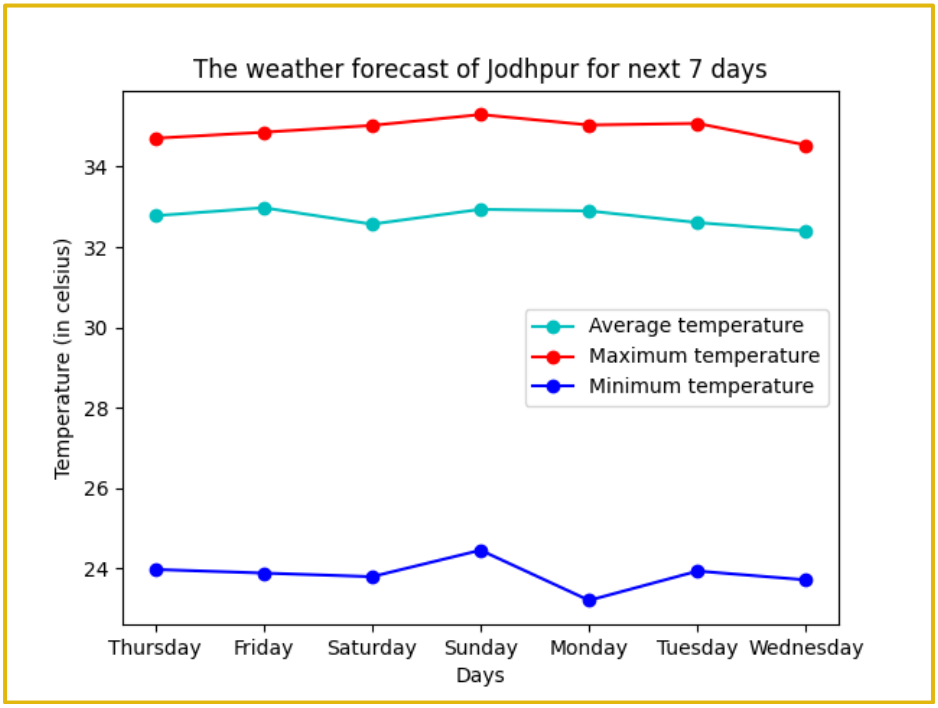
The source code was quite long, wasn't it? Now let's see the outputs.

1) After successfully connecting the python code to MySQL database, the program will give the user 3 choices, asking them if they want the weather for today, yesterday, or tomorrow and will save the data retrieved from the API to MySQL database as well as in a CSV file.

```
What information do you want?
● 1.Current weather
● 2.Yesterday's weather
● 3.Tomorrow's weather

Enter your choice (1, 2 or 3): 1
You chose for today's weather...
◖Today's date is 13-10-22 ◗
Enter City name: Jodhpur


The coordinates of entered location are:
 (26.2967719, 73.0351433)
------------------------

:::::The Current Weather of Jodhpur is:::::

■ Today's average temperature: 26.52 °Cor 79.74 °F
■ Feels Like: 26.52 °Cor 79.74 °F
■ Humidity: 29 %

(Data successfully saved in the database)
(CSV file appended)
```

2) Now the user will be asked if they want to see the forecast graph of the next 7 days (for the given city) and shall receive the said information by entering "Y" or "y". For the graph, we used matplotlib and passed numpy arrays into the plot function of the package:

```
Do you want forecast graph for next 7 days? (Y/N): y
Enter the unit you want the graph in
▶for CELSIUS, enter C/c.
▶for FAHRENHEIT, enter F/f.
Enter your choice (C/F): c

▶    Here is the graph:
```



3) The users may want to check the past records stored in MySQL databases. For that, they will be presented with 5 different choices, i.e., 5 different ways to access the records, depending upon the user. If they don't want to do so, they can end the program immediately by pressing number 5. (Please see the next page.)

Choice 1 will let the user access desired number of latest past records.
Choice 2 will enable the user to see the data of a specific city.
Choice 3 allows the user to see the number of records for all cities that have been entered by all different users.
Choice 4 will display the record containing the highest temperature out of all the cities.

Choice 5 will end the program and the user will receive a gentle "Thank You".

Note that the choices will get displayed again and again after selecting each choice, except choice 5, for which the program will end immediately.

The screenshots for the outputs have been attached on the next page.

```
If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5

Enter Your Choice: 1

The order of the columns is
Date, Current_Temperature_CELSIUS, Feels_Like_CELSIUS, Humidity, Time, City,

Please enter the number of past records you want to see: 7

[('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:27', 'JODHPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:25', 'JODHPUR'),
('10-03-21', Decimal('24.00'), Decimal('22.72'), 46, '04:24', 'JAIPUR'),
('10-03-21', Decimal('21.00'), Decimal('19.08'), 56, '04:24', 'AHMEDABAD'),
('10-03-21', Decimal('16.92'), Decimal('7.87'), 17, '04:23', 'LAS VEGAS'),
('10-03-21', Decimal('24.00'), Decimal('22.72'), 46, '04:23', 'JAIPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:22', 'JODHPUR')]

No of Records:  7

If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5

Enter Your Choice: 2

Which city's record do you want to see? Jodhpur

[('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:21', 'JODHPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:21', 'JODHPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:22', 'JODHPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:25', 'JODHPUR'),
('10-03-21', Decimal('20.41'), Decimal('16.53'), 33, '04:27', 'JODHPUR')]

No of Records:  5
If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5

Enter Your Choice: 3
[('JODHPUR', 5), ('NEW DELHI', 2), ('MUMBAI', 1), ('LAS VEGAS', 1), ('JAIPUR', 2),
('AHMEDABAD', 1)]

No of Records:  6

If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5

Enter Your Choice: 4
[('10-03-21', Decimal('27.00'), Decimal('29.65'), 69, '04:22', 'MUMBAI')]

No of Records:  1

If you want to see data from database, press 1
If you want to see data by city, press 2
If you want to see the number of records for each city, press 3
If you want to see highest temperature in the records, press 4
To end program, press 5

Enter Your Choice: 5
Thank You
```

MySQL DATABASE

The description of the table created is presented in this screenshot:

```
mysql> desc weather_report;
+-----------------------------+-------------+------+-----+---------+-------+
| Field                       | Type        | Null | Key | Default | Extra |
+-----------------------------+-------------+------+-----+---------+-------+
| Date                        | varchar(10) | NO   |     | NULL    |       |
| Current_Temperature_CELSIUS | decimal(4,2)| NO   |     | NULL    |       |
| Feels_Like_CELSIUS          | decimal(4,2)| NO   |     | NULL    |       |
| Humidity                    | int         | NO   |     | NULL    |       |
| Time                        | varchar(5)  | NO   |     | NULL    |       |
| City                        | varchar(15) | NO   |     | NULL    |       |
+-----------------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

One can see that the data retrieved by the Python-API program is stored in the table created in MySQL:

```
mysql> select * from weather_report;
+----------+-----------------------------+-------------------+----------+-------+-----------+
| Date     | Current_Temperature_CELSIUS | Feels_Like_CELSIUS | Humidity | Time  | City      |
+----------+-----------------------------+-------------------+----------+-------+-----------+
| 10-03-21 |                       20.41 |             16.53 |       33 | 04:21 | JODHPUR   |
| 10-03-21 |                       20.41 |             16.53 |       33 | 04:21 | JODHPUR   |
| 10-03-21 |                       20.41 |             16.53 |       33 | 04:22 | JODHPUR   |
| 10-03-21 |                       20.00 |             19.92 |       68 | 04:22 | NEW DELHI |
| 10-03-21 |                       20.00 |             19.92 |       68 | 04:22 | NEW DELHI |
| 10-03-21 |                       27.00 |             29.65 |       69 | 04:22 | MUMBAI    |
| 10-03-21 |                       16.92 |              7.87 |       17 | 04:23 | LAS VEGAS |
| 10-03-21 |                       24.00 |             22.72 |       46 | 04:23 | JAIPUR    |
| 10-03-21 |                       24.00 |             22.72 |       46 | 04:24 | JAIPUR    |
| 10-03-21 |                       21.00 |             19.08 |       56 | 04:24 | AHMEDABAD |
| 10-03-21 |                       20.41 |             16.53 |       33 | 04:25 | JODHPUR   |
| 10-03-21 |                       20.41 |             16.53 |       33 | 04:27 | JODHPUR   |
+----------+-----------------------------+-------------------+----------+-------+-----------+
12 rows in set (0.00 sec)
```

DATA OUTPUT IN CSV

The data returned by the API will also be appended in a CSV file. Here is the screenshot of the output obtained in the csv:

	A	B	C	D	E	F
1	Date	Current_T	Feels_Like	Humidity	Time	City
2	12-10-2022	34.68	32.59	19	14:25	Jodhpur
3	12-10-2022	32.77	34.3	44	14:26	Shirpur
4	12-10-2022	26.99	31.13	94	14:35	Mumbai
5	12-10-2022	32.1	32.99	43	14:45	Indore
6	12-10-2022	29.5	29.45	43	14:46	Ujjain
7	12-10-2022	29.05	30.81	58	14:47	Delhi
8	12-10-2022	31.5	29.9	26	14:48	Udaipur
9	12-10-2022	31.67	31.65	39	14:49	Agra
10	12-10-2022	30.23	32.86	58	14:50	Hyderabad
11	13-10-2022	26.52	26.52	29	01:33	Jodhpur
12	13-10-2022	26.52	26.52	29	01:35	Jodhpur
13	13-10-2022	26.52	26.52	29	01:39	Jodhpur
14						

WEB REFERENCES

We used API from the following services:

- Weather API via:

  https://openweathermap.org/api/

- API used for coordinates via:

  https://opencagedata.com/

➢ Other Useful free websites that helped us overcome the errors

1. https://www.mulesoft.com/
2. https://www.infoworld.com/
3. https://pypi.org/
4. https://www.w3schools.com/
5. https://www.geeksforgeeks.org/
6. https://realpython.com/
7. https://stackoverflow.com/
8. Github repo link:
   https://github.com/dakshgehlot/Python-Project-SEM-3

THANK YOU!

A PROGRAM BY SANYAM JAIN (N229), DAKSH GEHLOT (N230) AND GAUTAM KUNDALIA (N243).