



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**INTERNSHIP ASSESSMENT/MINI PROJECT REPORT**

**ABES ENGINEERING COLLEGE, GHAZIABAD**

INTERNSHIP ASSESSMENT/MINI PROJECT REPORT SUBMITTED IN PARTIAL  
FULFILLMENT OF THE DEGREE OF BACHELOR OF TECHNOLOGY,  
2020-2024

**From (30 September 2021 To 30 April 2022)**

**Name of the student: Sachin Kumar Singh**  
**University Roll No: 2000321030137**

## **ACKNOWLEDGEMENT**

I would like to acknowledge the contributions of the following people without whose help and guidance this report would not have been completed.

I acknowledge the counsel and support of **Ms. Anshika Agarwal, Assistant Professor, IT Department**, with respect and gratitude, whose expertise, guidance, support, encouragement, and enthusiasm has made this report possible. Their feedback vastly improved the quality of this report and provided an enthralling experience. I am indeed proud and fortunate to be supported by him. I am also thankful to **Prof. (Dr.) Amit Sinha, H.O.D of Information Technology Department, and Dr. Kanika Gupta, A.H.O.D of Information Technology Department** for his constant encouragement, valuable suggestions and moral support and blessings.

Although it is not possible to name individually, I shall ever remain indebted to the faculty members of ABES Engineering College, Ghaziabad for their persistent support and cooperation extended during this work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement.

**Name: Sachin Kumar Singh**

**Roll Number: 2000320130137**

## **CONTENTS**

	<b>Page No.</b>
1. Introduction to Company/Platform	04
2. Internship/Course overviews	05
3. Project Based Learning during the Internship /Course	06-13
4. Data flow diagram of project	14-19
5. Summary and Conclusion	19
6. References	19
7. Annexures:	20-25
<ul style="list-style-type: none"><li>• Codes snippets</li><li>• Project Demo snippets</li><li>• Certificate(s)</li></ul>	

## 1. Introduction to Company/Platform

The platform that I used for learning and opting the course is **Udemy**. Udemy, Inc. is a for-profit massive open online course (MOOC) provider aimed at professional adults and students. It was founded in May 2010 by **Eren Bali**, **Gagan Biyani**, and **Oktay Caglar**. The headquarters of Udemy is located in San Francisco, US, with hubs in Denver, US; Dublin, Ireland; Ankara, Turkey; Sao Paulo, Brazil; and Gurugram, India.

Udemy is a platform that allows instructors to build online courses on their preferred topics. Using Udemy's course development tools, they can upload videos, PowerPoint presentations, PDFs, audio, ZIP files and live classes to create courses. Instructors can also engage and interact with users via online discussion boards.

Courses are offered across a breadth of categories, including business and entrepreneurship, academics, the arts, health and fitness, language, music, and technology.

Most classes are in practical subjects such as Excel software or using an iPhone camera. Udemy also offers Udemy for Business, enabling businesses access to a targeted suite of over 7,000 training courses on topics from digital marketing tactics to office productivity, design, management, programming, and more. With Udemy for Business, organizations can also create custom learning portals for corporate training.

Courses on Udemy can be paid or free, depending on the instructor. In 2015, the top 10 instructors made more than \$17 million in total revenue.

In April 2013, Udemy offered an app for Apple iOS, allowing students to take classes directly from iPhones; The Android version was launched in January 2014. As of January 2014, the iOS app had been downloaded over 1 million times, and 20 percent of Udemy users access their courses via mobile. In July 2016, Udemy expanded their iOS platform to include Apple TV. On January 11, 2020, the Udemy mobile app became the #1 top grossing Android app in India.

## 2. Internship/Course overviews

The course that I studied is “**The Python Mega Course 2022: Build 10 Real-World Programs**” by Ardit Sulce. The Python Mega Course is an online course that uses a hands-on teaching approach which has proved to be very successful with thousands of students who have taken the course, built their own programs, and even found a Python job afterward. With 50,000+ student reviews and an outstanding 4.6 average rating, this learning package guarantees you will become a Python programmer after successful completion. The course focuses on teaching you Python by building real-world Python programs. That way, you learn the language syntax, but most importantly, you will learn the actual skill of designing and building real-life programs.

This course assumes you have no previous knowledge of programming. Whenever a new programming term emerges in the lectures, we will first explain it academically. Then we use it practically in a real-world code example and reuse it in exercises until you learn everything by heart.

The first 12 sections of the course cover Python basics. The other 27 sections cover intermediate and advanced Python, and you can jump right into those sections if you know the basics.

That is up to how you take the course. Simply watching the videos is not enough. You should try the code on your computer, change it, rerun it, improve it further, fix the bugs, try making a similar app, and ask questions in the Q&A when you get stuck. You will be guided through that entire process, so don't worry about falling off track.

You can publish the apps in your own GitHub private or public account. However, I recommend changing or adding something to the app to make it unique and really yours. Changing or adding something new will also immensely help your learning. If the code in a video does not work because a new version of Python is released or a new version of a Python third-party package breaks the current code, that video is immediately updated with a new one.

It is recommended to watch 30 minutes of video content per day, followed by two hours of independent work and exercise activities provided along with the videos. The course has 33 hours of video, so it should take you three months to complete the course, considering you study five days a week. If you want to speed up the process, I recommend watching up to 1 hour of content, but not more, followed by 4 hours of independent work and exercises.

You can drop a question in the Q&A, and the instructor or the teaching assistant will answer your questions within the same day. You can also use the chat in the course Discord server to ask questions and chat with fellow students about Python.

That will likely not happen. But, if it does, you are covered by the Udemy 30-day money-back guarantee, so you can quickly return the course. No questions asked.

The course used **Python 3** for all the coding purpose. We will use Visual Studio Code in the course. However, many students prefer to use their favorite IDE. PyCharm, Atom, and even IDLE will work fine.

The operating system does not matter. The code covered in the videos will work 100% the same in all operating systems. Once you buy the course, it is yours. You will get all future updates for free as well.

### 3. Project Based Learning during the Internship /Course

The course is designed around mastering Python by building ten applications that vary from simple to complex. The ten apps are carefully served to students through a combination of videos, coding exercises, quizzes, code notebooks, cheat sheets, and unlimited support from the instructor and the teaching assistant in the course forum. You will also receive an invitation to join our Discord server to chat with other fellow students, participate in code challenges, and get help when you need it to make sure you stay tuned and motivated. This course is for those who know Python basics and want to master Python and have no prior knowledge of Python. The main learning of this course are given below:

- Become a Python programmer by learning to build real-world apps in Python 3
- Build desktop database apps, webcam motion detectors, data visualization dashboards, blog websites, web scrapers, and more
- Practice the skills with hundreds of interactive Python exercises and projects
- Build a personal website entirely in Python
- Build a mobile app that improves your mood with positive quotes
- Create a web app that processes Excel and CSV files
- Build a book inventory GUI app with an SQL database backend
- Create a webcam app that records video and detects moving objects
- Create a web scraper that extracts real-estate data
- Create a modern data visualization app
- Build an app that sends automated emails
- Interact with our Python online community and get help when you need it

#### ▪ Python overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

**Python is interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

**Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

- History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

- Features of Python

**Easy-to-learn:** Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

**Easy-to-Read:** Python code is more clearly defined and visible to the eyes.

**Easy -to-Maintain:** Python's source code is fairly easy-to-maintain.

**A broad standard library:** Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable:** Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

**Extendable:** You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

**Databases:** Python provides interfaces to all major commercial databases.

**GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- i. It support functional and structured programming methods as well as OOP.
- ii. It can be used as a scripting language or can be compiled to byte code for building large applications.
- iii. It provides very high level dynamic datatypes and supports dynamic type checking.
- iv. It supports automatic garbage collections.
- v. It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA

#### ▪ Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore ( `_` ) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as `@`, `$`, and `%` within identifiers. Python is a case sensitive programming language.

#### ▪ Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

**And, exec, not**

**Assert, finally, or**

**Break, for, pass**

**Class, from, print**

**continue, global, raise**

**def, if, return**

**del, import, try**

**elif, in, while**

**else, is, with**

**except, lambda, yield**

#### ▪ Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.



The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True: print "True"

else:

print "False"
```

- Variable Types

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

- Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10 # An integer assignment
```

```
weight=10.60 # A floating point
```

```
name="Ardent" # A string
```

- Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1
```

```
a,b,c = 1,2,"hello"
```

- Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- String
- List
- Tuple
- Dictionary
- Number

- Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	<b>int(x [,base])</b> Converts x to an integer. base specifies the base if x is a string
2	<b>long(x [,base] )</b> Converts x to a long integer. base specifies the base if x is a string.
3	<b>float(x)</b> Converts x to a floating-point number.
4	<b>complex(real [,imag])</b> Creates a complex number.
5	<b>str(x)</b> Converts object x to a string representation.
6	<b>repr(x)</b> Converts object x to an expression string.
7	<b>eval(str)</b> Evaluates a string and returns an object.
8	<b>tuple(s)</b> Converts s to a tuple.
9	<b>list(s)</b> Converts s to a list.

- Functions

Defining a Function:

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

- Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

# Function definition is here

```
def changeme(mylist):
    "This changes a passed list into this
    function" mylist.append([1,2,3,4]);
    print"Values inside the function:
    ",mylist return
```

# Now you can call changeme function

```
mylist=[10,20,
30];
changeme(myli
st);
print"Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]  
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

- Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;          # This is global variable.
```

# Function definition is here

```
def sum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2;    # Here total is local
variable. print"Inside the function local total :
", total
return total;

# Now you can call sum function

sum(10,20);
print"Outside the function global total : ", total
```

When the above code is executed, it produces the following result –

```
Inside the function local total : 30
Outside the function global total : 0
```

## ■ Modules

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named aname normally resides in a file named aname.py. Here's an example of a simple module, support.py

```
def print_func( par ):
    print"Hello : ", par
    return
```

## The import Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The import has the following syntax –

```
import module1[, module2[,... moduleN]
```

- Packages

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file Pots.py available in Phone directory. This file has following line of source code –

```
def Pots():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- Phone/Isdn.py file having function Isdn()
- Phone/G3.py file having function G3()

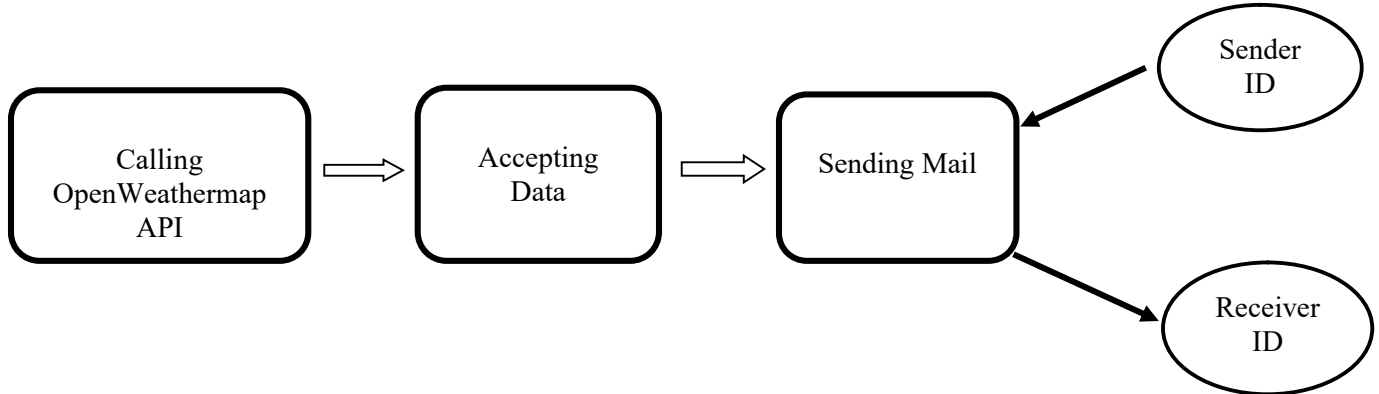
Now, create one more file `__init__.py` in Phone directory –

- Phone/ `__init__.py`

To make all of your functions available when you've imported Phone, you need to put explicit import statements in `__init__.py` as follows –

```
from  
Pots  
import  
Pots  
from  
Isdn  
import  
Isdn  
from  
G3  
import
```

#### 4. Data flow diagram of project



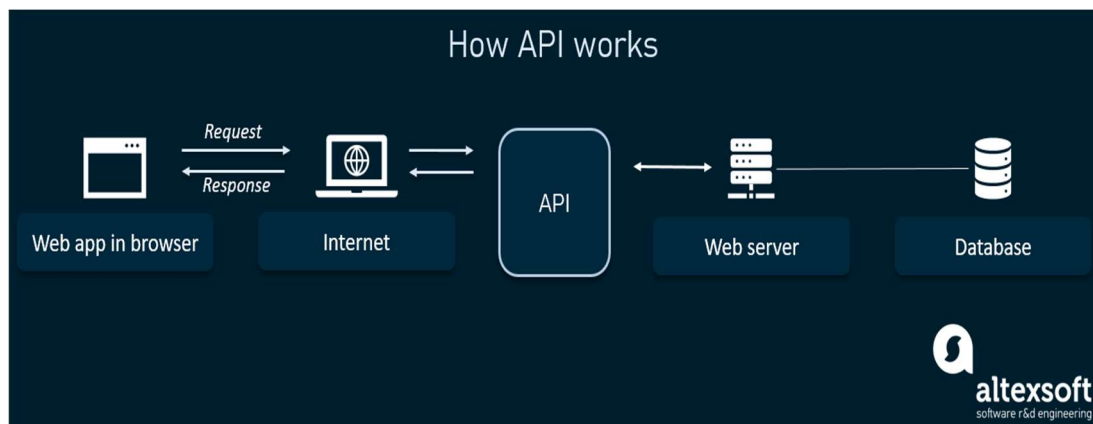
##### ▪ API

An **API** 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

Application programming interfaces consist of two components:

- i. Technical specification describing the data exchange options between solutions with the specification done in the form of a request for processing and data delivery protocols
- ii. Software interface written to the specification that represents it

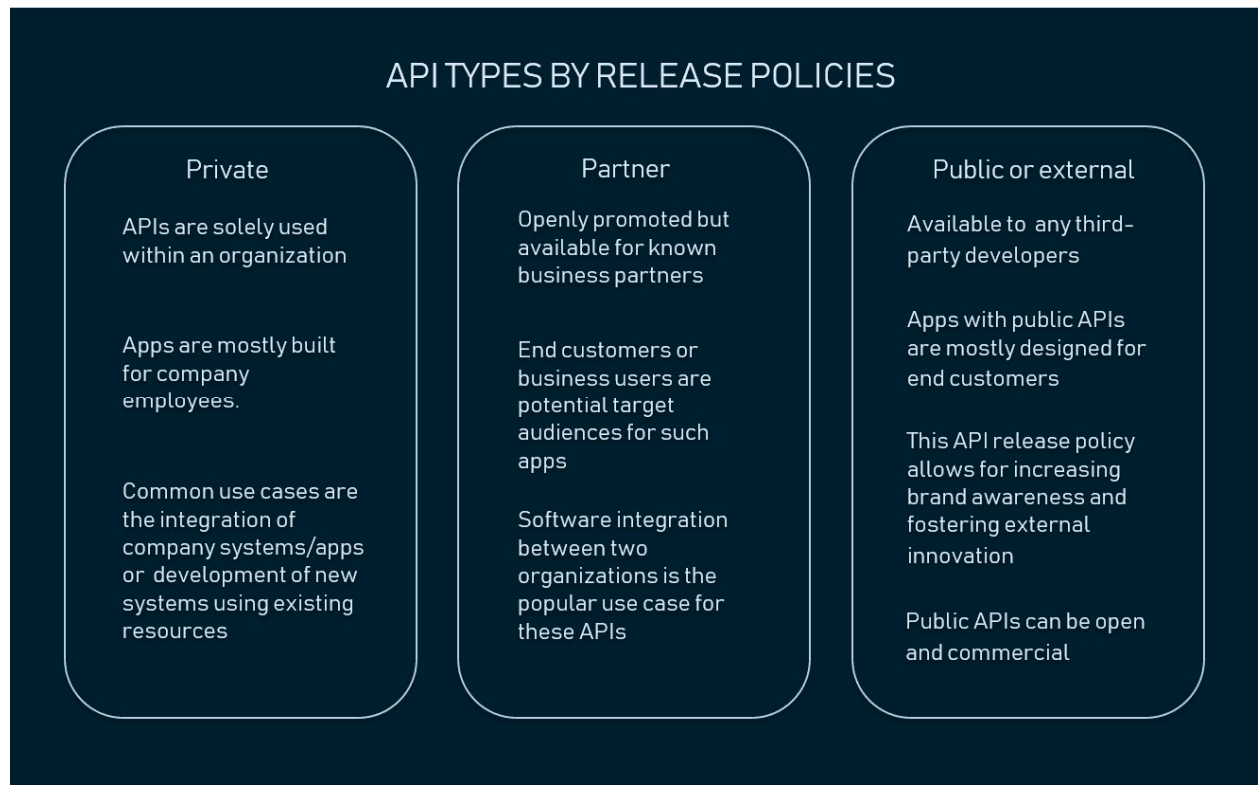
##### ▪ Working of API



- Types of API

APIs by availability aka release policies

In terms of release policies, APIs can be private, partner, and public.



- Private APIs.** These application software interfaces are designed for improving solutions and services within an organization. In-house developers or contractors may use these APIs to integrate a company's IT systems or applications, build new systems or customer-facing apps leveraging existing systems. Even if apps are publicly available, the interface itself remains available only for those working directly with the API publisher. The private strategy allows a company to fully control the API usage.
- Partner APIs.** Partner APIs are openly promoted but shared with business partners who have signed an agreement with the publisher. The common use case for partner APIs is software integration between two parties. A company that grants partners with access to data or capability benefits from extra revenue streams. At the same time, it can monitor how the exposed digital assets are used, ensure whether third-party solutions using their APIs provide decent user experience, and maintain corporate identity in their apps.

- iii. **Public APIs.** Also known as developer-facing or external, these APIs are available for any third-party developers. A public API program allows for increasing brand awareness and receiving an additional source of income when properly executed.

There are two types of public APIs – **open (free of charge)** and **commercial** ones. The Open API Definition suggests that all features of such an API are public and can be used without restrictive terms and conditions. For instance, it's possible to build an application that utilizes the API without explicit approval from the API supplier or mandatory licensing fees. The definition also states that the API description and any related documentation must be openly available, and that the API can be freely used to create and test applications.

Commercial API users pay subscription fees or use APIs on a pay-as-you-go basis. A popular approach among publishers is to offer free trials, so users can evaluate APIs before purchasing subscriptions. Learn more about how businesses benefit from opening their APIs for public use in our detailed article on API economy.

- OpenWeatherMap



---

OpenWeatherMap is an online service, owned by OpenWeather Ltd, that provides global weather data via API, including current weather data, forecasts, nowcasts and historical weather data for any geographical location. The company provides a minute-by-minute hyperlocal precipitation forecast for any location. The convolutional machine learning model is used to utilize meteorological broadcast services and data from airport weather stations, on-ground radar stations, weather satellites, remote sensing satellites, METAR and automated weather stations.

The company has more than 2 million customers, ranging from independent developers to Fortune 500 companies.



The variety of weather APIs provided by OpenWeatherMap have found a significant popularity among the software developers, which resulted in the growing multitude of repositories on GitHub. The APIs support multiple languages, units of measurement and industry standard data formats like JSON and XML.

In 2021, OpenWeatherMap launched a number of initiatives to support students, researchers and developers across the world.

OpenWeatherMap provides a range of weather-related products in a variable combination of depth and steps of measurement to millions of clients globally. The product range includes current, historical and forecasted weather data with the granularity as high as 1 minute. The length of the nowcast reaches 2 hours, short-term forecast reaches 16 days and long-term forecast can reach up to 1 year length. Historical weather data goes over 40 years deep. OpenWeather also provides a range of weather maps and weather alert services.

In 2015, Google chose OpenWeatherMap as a weather data provider for its bid-by-weather script in Google Ads, that serves the ads based on the local weather conditions, such as temperature, humidity, and cloudiness. Same year, Google published documentation on how to use OpenWeather data to display weather conditions on Google Maps.

In 2020, Samsung included OpenWeatherMap into their Galaxy Watch Studio as a weather data provider for those willing to develop applications for Galaxy Watch. In 2020, OpenWeatherMap has released its weather application for iOS and Android.

OpenWeather provides data for weather risk management on the individual agreement basis to the industries

like energy, agriculture, transportation, construction, municipalities, travel, food processors, retail sales and real estate. OpenWeather also operates under the terms of Creative Commons Attribution-ShareAlike license providing free access to the APIs that include current weather, a minutely forecast for 1 hour, hourly forecast for 48 days, 3-hour forecast for 5 days, daily forecast for 7 days, short-term history, weather maps, alerts, geocoding, air quality weather triggers and weather widgets. The projects with a higher demand of loading, may obtain an extended service on the basis of paid subscription.

- SMTP: (Simple Mail Transfer Protocol )Library

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail –

```
import smtplib

smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

Here is the detail of the parameters –

- a. **host** – This is the host running your SMTP server. You can specify IP address of the host or a domain name like tutorialspoint.com. This is optional argument.
- b. **port** – If you are providing host argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.
- c. **local\_hostname** – If your SMTP server is running on your local machine, then you can specify just localhost as of this option.

An SMTP object has an instance method called sendmail, which is typically used to do the work of mailing a message. It takes three parameters –

- a. **The sender** – A string with the address of the sender.
- b. **The receivers** – A list of strings, one for each recipient.
- c. **The message** – A message as a string formatted as specified in the various RFCs.

### Example:

Here is a simple way to send one e-mail using Python script. Try it once –

```
#!/usr/bin/python

import smtplib

sender = 'from@fromdomain.com'
receivers = ['to@todomain.com']

message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test

This is a test e-mail message.
"""

try:
    smtpObj = smtplib.SMTP('localhost')
    smtpObj.sendmail(sender, receivers, message)
```

```
print "Successfully sent email"
except SMTPException:
    print "Error: unable to send email"
```

Here, you have placed a basic e-mail in message, using a triple quote, taking care to format the headers correctly. An e-mail requires a From, To, and Subject header, separated from the body of the e-mail with a blank line.

To send the mail you use smtpObj to connect to the SMTP server on the local machine and then use the sendmail method along with the message, the from address, and the destination address as parameters (even though the from and to addresses are within the e-mail itself, these aren't always used to route mail).

If you are not running an SMTP server on your local machine, you can use smtplib client to communicate with a remote SMTP server. Unless you are using a webmail service (such as Hotmail or Yahoo! Mail), your e-mail provider must have provided you with outgoing mail server details that you can supply them, as follows –

```
smtplib.SMTP('mail.your-domain.com', 25)
```

## 5. Summary and Conclusion

The weather affects almost everything we do in our daily lives. It affects what we wear and what types of activities that we do. In many professions, it affects either where we work or when we work, and oftentimes, it affects if we work at all that day. Consider roofers, landscapers, and miscellaneous outdoor construction workers- how hot is it outside? Is it raining or snowing? Is there going to be a storm today? All of these conditions will affect what they do.

Through this project, we can send notifications to the user before any severe weather condition so that he can make much needed preparations so as to reduce the loss to be occurred

This project can further be used in various life saving tasks with many fields of improvement.

## 6. References

[How to Create A Weather Chatbot](#)

[Tips and Tricks for Handling Configuration Files in Python](#)

<https://towardsdatascience.com/how-to-create-a-weather-alert-system-in-python-5fab4b42c49a>

## 7. Annexures:

### a) Codes snippets

```
import requests,time,threading,smtplib
from win10toast import ToastNotifier
ipinfo=requests.get("https://ipapi.co/json").json()

coordinates={"lat": ipinfo["latitude"],"lon":
ipinfo["longitude"]}

key="337e0427f1732f6e68bcb357ab57d189"

weatherAPI="https://api.openweathermap.org/data/2.5/weather"
weather=requests.get(weatherAPI,params={**coordinates,"appid":key
}).json()
condition=weather["weather"][0]["description"]
temp1=weather["main"]["temp"]-273.15
temp2=round(temp1,2)
def show_weather():

weather=requests.get(weatherAPI,params={**coordinates,"appid":key
}).json()
    condition=weather["weather"][0]["description"]
    temp1=weather["main"]["temp"]-273.15
    temp2=round(temp1,2)
    # print(weather)
    # print(condition)
    # print(temp2)
    print(f'Now its {condition}, currently {temp2}°C')
    notif=ToastNotifier()
    notif.show_toast("The current weather at Ghaziabad is",f'Now
its {condition}, currently {temp2}°C')
    # threading.Timer(25,show_weather).start()

show_weather()

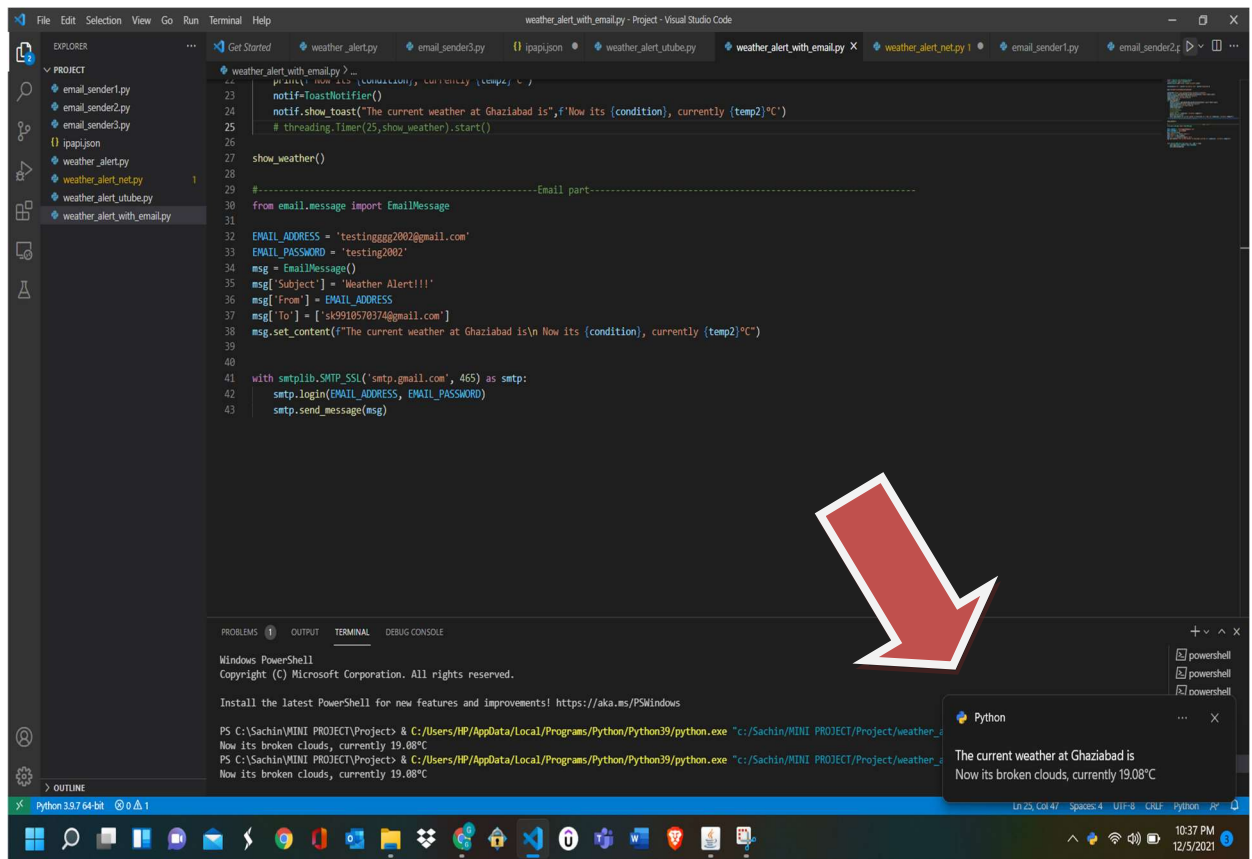
#-----Email
part-----
--
from email.message import EmailMessage

EMAIL_ADDRESS = 'testingggg2002@gmail.com'
```

```
EMAIL_PASSWORD = 'testing2002'
msg = EmailMessage()
msg['Subject'] = 'Weather Alert!!!'
msg['From'] = EMAIL_ADDRESS
msg['To'] = ['sk9910570374@gmail.com']
msg.set_content(f"The current weather at Ghaziabad is\n Now its
{condition}, currently {temp2}°C")

with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
    smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
    smtp.send_message(msg)
```

## b) Project Demo Snippets-1



The screenshot displays the Visual Studio Code interface with a project named 'weather\_alert\_with\_email.py'. The Explorer sidebar on the left shows the project structure, including files like 'email\_sender1.py', 'email\_sender2.py', 'email\_sender3.py', 'ipapi.json', 'weather\_alert.py', 'weather\_alert\_net.py', 'weather\_alert\_utube.py', and 'weather\_alert\_with\_email.py'. The main editor window shows the code for 'weather\_alert\_with\_email.py', which includes a function to show a toast notification and an email-sending function using the 'smtplib' library. The code is as follows:

```
23 import sys, os, json, urllib2, urllib, urllib3, smtplib, threading, time
24 notif=ToastNotifier()
25 notif.show_toast("The current weather at Ghaziabad is",f'Now its {condition}, currently {temp2}°C')
26 # threading.Timer(25,show_weather).start()
27
28 show_weather()
29
30 #-----Email part-----
31 from email.message import EmailMessage
32
33 EMAIL_ADDRESS = 'testingggg2002@gmail.com'
34 EMAIL_PASSWORD = 'testing2002'
35 msg = EmailMessage()
36 msg['Subject'] = 'Weather Alert!!!'
37 msg['From'] = EMAIL_ADDRESS
38 msg['To'] = ['sk9910570374@gmail.com']
39 msg.set_content(f"The current weather at Ghaziabad is\n Now its {condition}, currently {temp2}°C")
40
41 with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
42     smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
43     smtp.send_message(msg)
```

Below the code editor, the TERMINAL panel shows the execution of the script using PowerShell. The output indicates that the weather is 'broken clouds' with a temperature of '19.88°C'. A large red arrow points from the code to the terminal output. In the bottom right corner, a system notification from Python displays the same weather information: 'The current weather at Ghaziabad is Now its broken clouds, currently 19.08°C'.

## System Notification

## Project Demo Snippets-2

# Weather Alert!!!



**testingggg2002@gmail.com**

to sk9910570374 ▼

The current weather at Ghaziabad is  
Now its haze, currently 23.07°C

↩ Reply

➡ Forward

Sender ID

### Project Demo Snippets-3

---

Weather Alert!!! Inbox x



testingggg2002@gmail.com

to me ▾

The current weather at Ghaziabad is  
Now its haze, currently 23.07°C



Reply



Forward

Receiver ID



**c) Certificate(s)**

