GSFC UNIVERSITY
EDUCATION RE-ENVISIONED

COVID-19 STATISTICS NOTIFICATION AND DASHBOARD SYSTEM

With Python

Developed By:

20BT04006: Avanish Deshpande
20BT04045: Devendra Sharma
20BT04051: Varada Deshpande
(Students, Semester 3, B.Tech. Computer Science and Engineering, GSFC University, Vadodara, Gujarat)

Under the Valuable Guidance of:

Ms. Zalak Kansagra
Ms. Rujul Desai

TABLE OF CONTENTS

THE IDEA:
INTRODUCTION TO THE PROJECT

The "COVID-19 Statistics Notification and Dashboard System" is a GUI-based Python-coded system, which uses some basic concepts of **web-scraping***. The code focuses on extracting the state-wise Covid-19 Data from an existing website (*http://bioinfo.usu.edu/covidTracker/india.php*) and displaying it in a presentable format through tkinter windows.

The state-wise extracted data is displayed in the tkinter window through treeview, while four distinct buttons have been created to enable the user to see the bar graphs for each of the statistics (the total confirmed cases, recovered cases, active cases and deaths), state-wise.

***Web scraping** or web harvesting or web data extraction is a technique of accessing the HTML of the webpage and extract useful information/data from it.

The system also gets the real time location of the user (from the website: *https://ipinfo.io/*) and notifies her/him with the statistics of her/his state as per the location fetched.

LIBRARIES USED: THEIR PURPOSE

Several Python libraries have aided in the development of the "COVID-19 Dashboard and Notification System". These include plyer, requests, BeautifulSoup, tkinter and matplotlib. The functionalities of the libraries that have been used for the project development are discussed in the following few pages.

PLYER

The plyer library is used to access the features of the hardware.

For this project, the "notification" class of the plyer module has been used to create desktop notifications. Calling the notify method of this class with appropriate data would create a desktop notification using the passed parameters. This is what a call to the function would look like:

```
notify(title, message, app_name, app_icon, timeout,
                    ticker, toast)
```

REQUESTS

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).

Python requests module has several built-in methods to make Http requests to specified URI using GET, POST, PUT, PATCH or HEAD requests. A Http request is meant to either retrieve data from a specified URI or to push data to a server.

This project makes use of the built-in GET method 'get()' of requests, which helps us to retrieve information from the given server using a given URI. The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the '?' character.

"`requests.get(url).json`" returns a JSON object of the result (if the result was written in JSON format, if not it raises an error). Python requests are generally used to fetch the content from a particular resource URI. Whenever we make a request to a specified URI through

Python, it returns a response object. Now, this response object would be used to access certain features such as content, headers, etc.

The json object is in the form of a dictionary, which, in this project, has been used to extract/ obtain the current state of residence of the user. The desktop notification for the COVID-19 statistics is generated in accordance with this state, using plyer.

"`requests.get(url).text`" returns the source code of the webpage whose url has been provided as a parameter. This is the first step in the process of data extraction for the project.

BS4/ BEAUTIFUL SOUP 4

Beautiful Soup is a Web Scraping framework of Python. To parse an HTML code, we pass the HTML source code into the BeautifulSoup constructor:

```
soup = BeautifulSoup(SourceCode, 'html.parser')
```

The source code used as an argument in this constructor could come from a local HTML document, or can be fetched real time from some URL, using the "`requests.get(url).text`" method of the "requests" library. For the purpose of this project, the latter method is used.

The BeautifulSoup object represents the document as a whole, in the form of a nested data structure. It transforms a complex HTML document into a complex tree of Python objects.

The nested tree structure that has been obtained using Beautiful Soup constructor contains numerous HTML tags. These tags may contain strings and other tags, also referred to as the tag's children. Beautiful Soup provides a lot of different attributes for navigating and iterating over a tag's children.

In the URL from where the COVID-19 statistics are being extracted, the required data has been represented in a tabular format, with the use of the HTML tags such as table, tr, th and td. In order to extract the statistical data, we first need to extract all the td tags from the HTML document (source code of the url). For this purpose, the "`find_all()`" method provided by Beautiful Soup is used. The data within these td tags is extracted using the "`get_text()`" method, and stored as a Python list. All this is done within a for loop:

```
myDataStr = []
for td in soup.find_all('td'):
    myDataStr.append(td.get_text())
```

The data thus obtained is further used to create objects and for displaying everything in a presentable format with tkinter window(s) and matplotlib graphs.

TKINTER

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. A basic tkinter window can be created using the method:

```
m=tkinter.Tk()
```

There are several widgets which can be put in a tkinter application.

The tkinter windows used for this project consist of the following widgets: -

- *Frame: It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.*
- *Treeview: A treeview widget is helpful in displaying more than one feature of every item listed in the tree to the right side of the tree in the form of columns. The output is similar to a table.*
- *Button: It is used to add buttons to the tkinter window. Action listeners can be added to the buttons to handle "click" events (such as opening a new window on button click).*
- *Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.*
- *Canvas: It is used to draw pictures and other complex layout like graphics, text and widgets (Used to plot graphs in the tkinter window for the purpose of this project)*

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes in tkinter, `pack()`, `grid()` and `place()`.

In this project, the `pack()` method has been used for the placement of widgets. This method organizes the widgets in blocks before placing them in the parent widget.

When the application is ready to run, the `mainloop()` is called. The `mainloop()` is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

MATPLOTLIB

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Plots helps us to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Matplotlib is a multi-platform data visualization library which comes with a wide variety of plots like line, bar, scatter, histogram etc.

To visualize the state-wise statistics of COVID-19, we can make use of horizontal bar charts, with states on the y-axis, and numbers on the x-axis. These can be plotted using the `barh()` method provided by the matplotlib library.

```
barh(y, x, bar_width, color=color)
```

When Matplotlib is used from the Python shell, the plots are displayed in a default window. These plots can however also be embedded in many graphical user interfaces like wxpython, pygtk, or Tkinter. These various options available as a target for the output plot are referred to as 'backends'. There are various modules available in matplotlib.backend for choosing the backend. One such module is backend_tkagg which is useful for embedding plots in Tkinter.

To embed a graph in a tkinter window, we first need to create a figure object using the `Figure()` class. Then, a Tkinter canvas (containing the figure) is created using `FigureCanvasTkAgg()` class.

Matplotlib charts by default have a toolbar at the bottom. When working with Tkinter, however, this toolbar needs to be embedded in the canvas separately using the `NavigationToolbar2Tk()` class.

THE BUILDING BLOCKS: BASIC PYTHON CONCEPTS

Apart from all the libraries, several basic concepts of Python have aided the development of the "COVID-19 Statistics Dashboard and Notification System". These include list manipulation, working with dictionaries, type casting, program control statements (loops and branch conditions), OOP (Object Oriented Programming) (classes, object, constructors, static methods etc.) and the use of default, positional and keyword arguments.

THE PROJECT:
SOURCE CODE

```
# COVID-19 Statistics Notification and Dashboard System

"""
    A Python Project by:
        20BT04006: Avanish Deshpande
        20BT04045: Devendra Sharma
        20BT04051: Varada Deshpande

    Students, 2nd year (Semester 3),
    B.Tech. Computer Science and Engineering,
    GSFC University, Vadodara.
    (Academic Year: 2021-22)
"""




""" Importing the Required Libraries """

# plyer: for desktop notifications
from plyer import notification
```

```
# For Web Scraping

# requests: to extract the required data from the website(s)
import requests

# bs4.BeautifulSoup: HTML Parsing
from bs4 import BeautifulSoup



# matplotlib: to plot graphs of data, and embed them into the tkinte
r window
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, Na
vigationToolbar2Tk)


# tkinter: to create a dashboard with the relevant data
from tkinter import *
from tkinter import ttk
import tkinter as tk




""" class covidData to create objects having state-
wise covid information """

class covidData:
    def __init__(self, state, total, recovered, death, active):
        self.state = state
        self.total = total
        self.recovered = recovered
        self.death = death
        self.active = active


    @staticmethod
    def createTable(treev, data):
        for i in data.values():
            e = treev.insert("", 'end', text ="L1", values =(i.state
, i.total, i.recovered, i.death, i.active))
```

```
    @staticmethod
    def showGraph(states, stats, chartTitle, color):
      # Create tkinter window
        g = tk.Tk()
        g.title(chartTitle)
        g.resizable(width = 850, height = 1000)

        fig = Figure(figsize = (15, 10), dpi = 100)
        a = fig.add_subplot(111)
        rects1 = a.barh(states, stats, 0.5, color=color)
        fig.suptitle(chartTitle, fontweight ="bold")

        canvas = FigureCanvasTkAgg(fig, master=g)
        canvas.draw()
        canvas.get_tk_widget().pack()

        # creating the Matplotlib toolbar
        toolbar = NavigationToolbar2Tk(canvas, g)
        toolbar.update()
        canvas.get_tk_widget().pack()
        g.mainloop()




def notifyMe(title="Let us beat this virus together!", message="A Py
thon assignment by:\n20BT04006: Avanish Deshpande\n20BT04045: Devend
ra Sharma\n20BT04051: Varada Deshpande", icon = "Logo.ico", Covid19D
ata=None):
    if Covid19Data!=None:
        r = requests.get('https://ipinfo.io/')
        data = r.json()
        print("\n\nUser Data Collected:\n\n", data)
        region = data['region']
        title = "\nState: " + Covid19Data[region].state
        message = "Confirmed Cases: " + str(Covid19Data[region].tota
l) + "\nRecovered Cases: " + str(Covid19Data[region].recovered) + "\
nDeaths: " + str(Covid19Data[region].death) + "\nActive Cases: " + s
tr(Covid19Data[region].active)

    # Creating Desktop Notification using the plyer library (notific
ation module)
```

```
    notification.notify(
        title = title,
        message = message,
        app_icon = icon,
        timeout = 10
    )









""" THE 'main' METHOD """

if __name__ == "__main__":

    """ Extract data from url """
    myHtmlData = requests.get('http://bioinfo.usu.edu/covidTracker/i
ndia.php').text
    soup = BeautifulSoup(myHtmlData, 'html.parser')

    # Extracting the required data from the table in the website
    myDataStr = []
    for td in soup.find_all('td'):
        myDataStr.append(td.get_text())

    print("\nmyDataStr: data extracted from the table:\n\n", myDataS
tr, "\n\n")




    """
        Creating objects of type "covidData" using the extracted dat
a, and storing them in a dictionary with name of the state as key.
        Separate lists are for storing statistical data: to help plo
t graphs.
    """
    data = {} # Dictionary for covidData objects
    states = []
    totalCases = []
    recoveredCases = []
    deaths = []
    activeCases = []
```

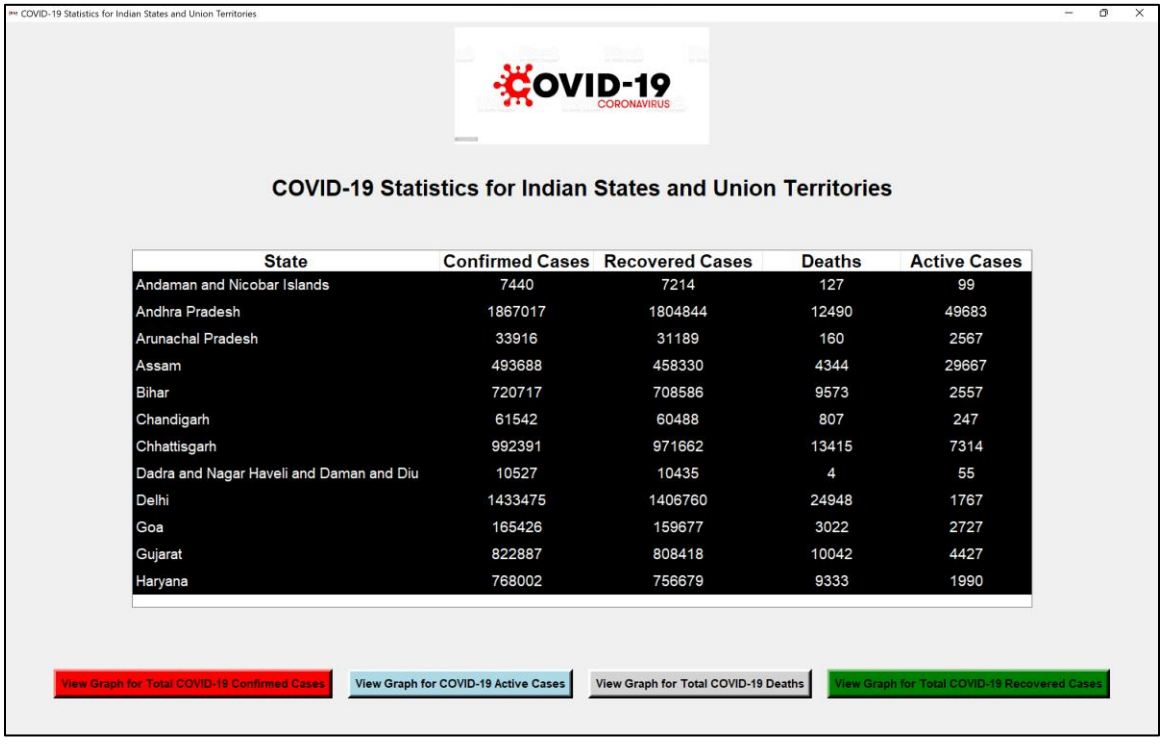
```
    for i in range(0, len(myDataStr), 5):
        state = myDataStr[i]
        if (state!="State Unassigned"):
            total = int(myDataStr[i+1])
            recovered = int(myDataStr[i+2])
            death = int(myDataStr[i+3])
            active = int(myDataStr[i+4])
            data[state] = covidData(state, total, recovered, death,
active)
            states.append(state)
            totalCases.append(total)
            recoveredCases.append(recovered)
            deaths.append(death)
            activeCases.append(active)



    """ Desktop Notification """
    # Notification 1: Python Assignment members
    notifyMe()
    # Notification 2: Covid 19 Data (Confirmed, Recovered, Active ca
ses and deaths): Data displayed according to the current state of re
sidence of user
    notifyMe(Covid19Data=data)



    """ Tkinter Window """
    # Create tkinter window, specify its size
    window = tk.Tk()
    # window.resizable(width = 1500, height = 1000)
    window.geometry("1500x1000+0+0") #window.geometry("window width
x window height + position right + position down")

    # Creating object of photoimage class: Image should be in the sa
me folder in which script is saved
    p = PhotoImage(file = 'logo.png')

    # Setting icon of tkinter window
    window.iconphoto(False, p)

    # Setting window title
    window.title("COVID-
19 Statistics for Indian States and Union Territories")
```

```
    img = Label(window, image=p, width=1500)
    l = Label(window, text="COVID-
19 Statistics for Indian States and Union Territories", font=(None,
22, 'bold'))
    img.pack(pady=5)
    l.pack(pady=30)



    """ Tkinter Frame """
    # Creating frame within tkinter window
    frame = Frame(window)



    """ Treeview """
    # Creating a Treeview within the frame, and setting its style
    treev = ttk.Treeview(frame, selectmode ='browse', height=15)
    ttk.Style().configure("Treeview", background="black", foreground
="white", fieldbackground="black", font=(None, 14), rowheight=35)
    ttk.Style().configure("Treeview.Heading", font=(None, 17, 'bold'
))

    # Calling pack method with respect to treeview
    treev.pack(side="top")

    # Defining the number of columns in treeview
    treev["columns"] = ("1", "2", "3", "4", "5")

    # Defining headings for treeview
    treev['show'] = 'headings'


    # Assigning the width and anchor to the respective columns
    treev.column("1", width = 400, anchor ='w')
    treev.column("2", width = 200, anchor ='c')
    treev.column("3", width = 220, anchor ='c')
    treev.column("4", width = 180, anchor ='c')
    treev.column("5", width = 170, anchor ='c')


    # Assigning the heading names to the respective columns
    treev.heading("1", text ="State")
```

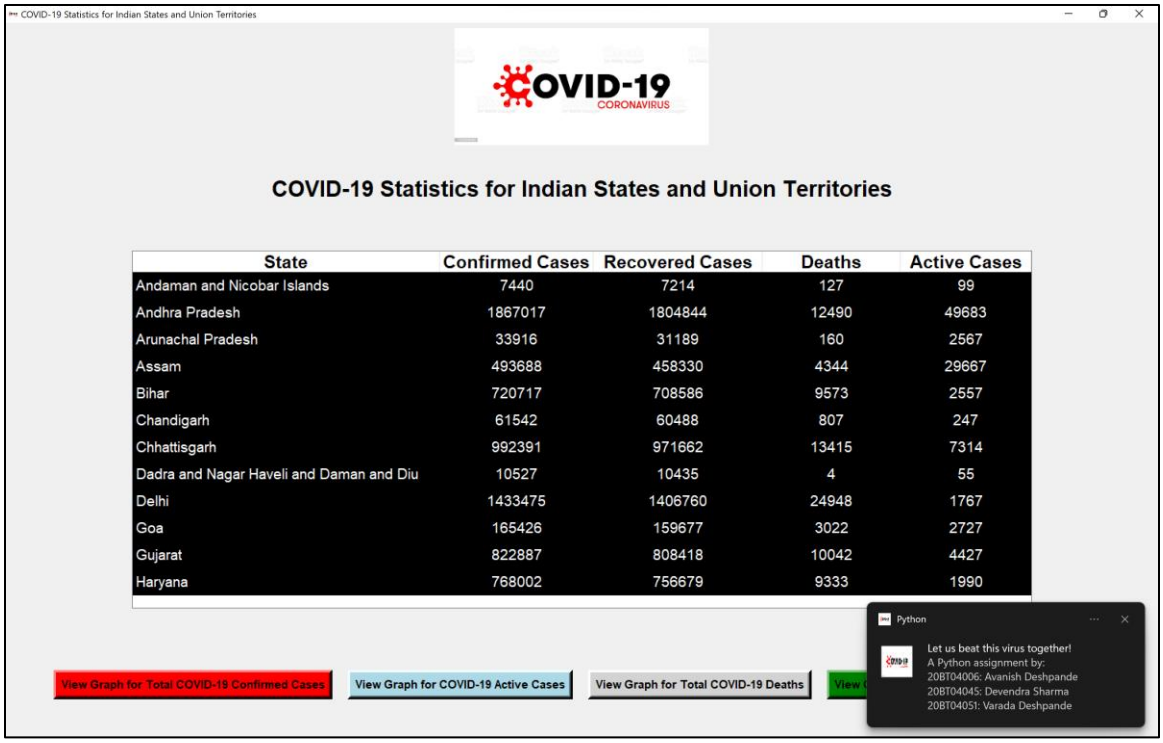
```
    treev.heading("2", text ="Confirmed Cases")
    treev.heading("3", text ="Recovered Cases")
    treev.heading("4", text ="Deaths")
    treev.heading("5", text ="Active Cases")


    # Inserting the items and their features to the columns built
    t = covidData.createTable(treev, data)



    """ Ploting graphs, and displaying them to the tkinter window ""
"


    btnFrame = Frame(window)

    # Create Buttons, add action listeners to the buttons, and "pack
" them to a frame
    # Action Listener: on clicking a button, open a new tkinter wind
ow containing the graph for the requested statistics

    btn1 = Button(btnFrame, text = 'View Graph for Total COVID-
19 Confirmed Cases', bd = '5', bg='red', font='sans 11 bold')
    btn1.bind("<Button>", lambda e: covidData.showGraph(states, tota
lCases, 'State-wise Total COVID-19 Confirmed Cases', 'red'))
    btn1.pack(side='left', padx=10)

    btn2 = Button(btnFrame, text = 'View Graph for Total COVID-
19 Recovered Cases', bd = '5', bg='green', font='sans 11 bold')
    btn2.bind("<Button>", lambda e: covidData.showGraph(states, reco
veredCases, 'State-wise Total COVID-19 Recovered Cases', 'green'))
    btn2.pack(side='right', padx=10)

    btn3 = Button(btnFrame, text = 'View Graph for COVID-
19 Active Cases', bd = '5', bg='lightblue', font='sans 11 bold')
    btn3.bind("<Button>", lambda e: covidData.showGraph(states, acti
veCases, 'State-wise COVID-19 Active Cases', 'blue'))
    btn3.pack(side='left', padx=10)

    btn4 = Button(btnFrame, text = 'View Graph for Total COVID-
19 Deaths', bd = '5', bg='lightgrey', font='sans 11 bold')
    btn4.bind("<Button>", lambda e: covidData.showGraph(states, deat
hs, 'State-wise COVID-19 Deaths', 'grey'))
    btn4.pack(side='right', padx=10)
```

```
    # Pack the buttons-containing frame to the tkinter window
    btnFrame.pack(side='bottom', pady=50)



    """ Calling pack with respect to frame """
    frame.pack(pady=30)



    """ Calling mainloop """
    window.mainloop()



    """ Desktop Notification: Displayed on closing the tkinter windo
w """
    notifyMe("Thank You!", "We are grateful for your support and gui
dance!\nMs. Rujul Desai\nMs. Zalak Kansagra", 'thankYou.ico')
```

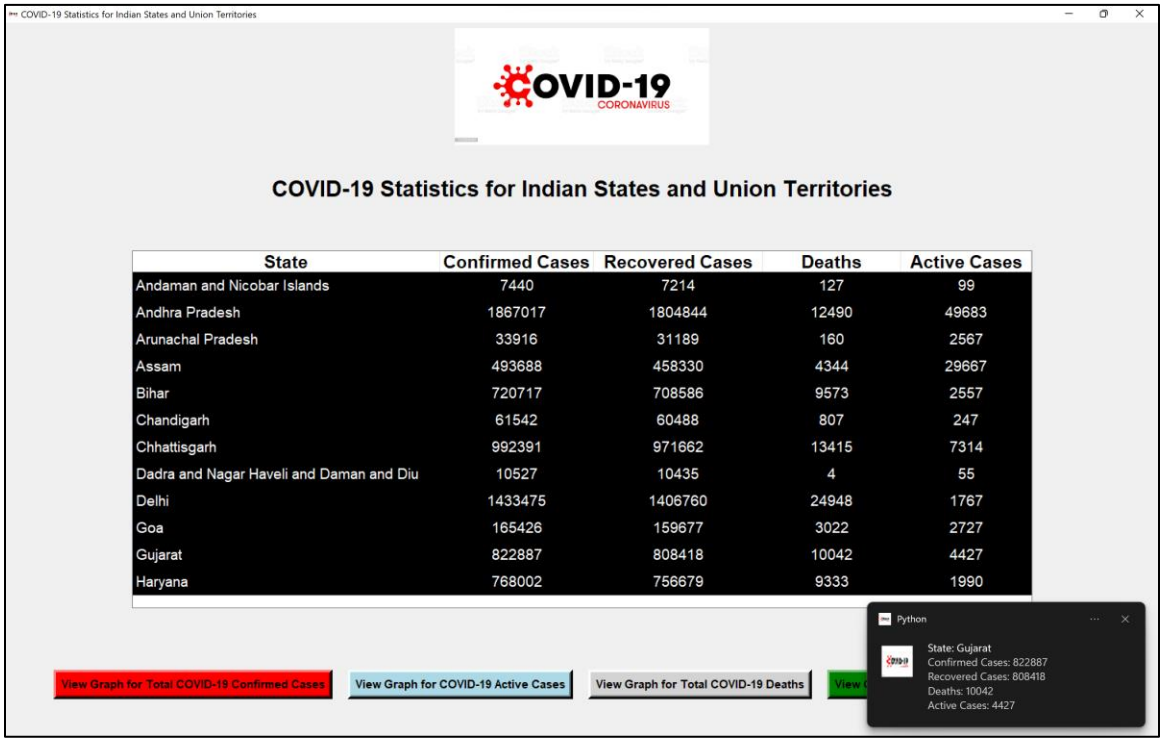
IMPLEMENTATION OF THE PROJECT: SCREENSHOTS



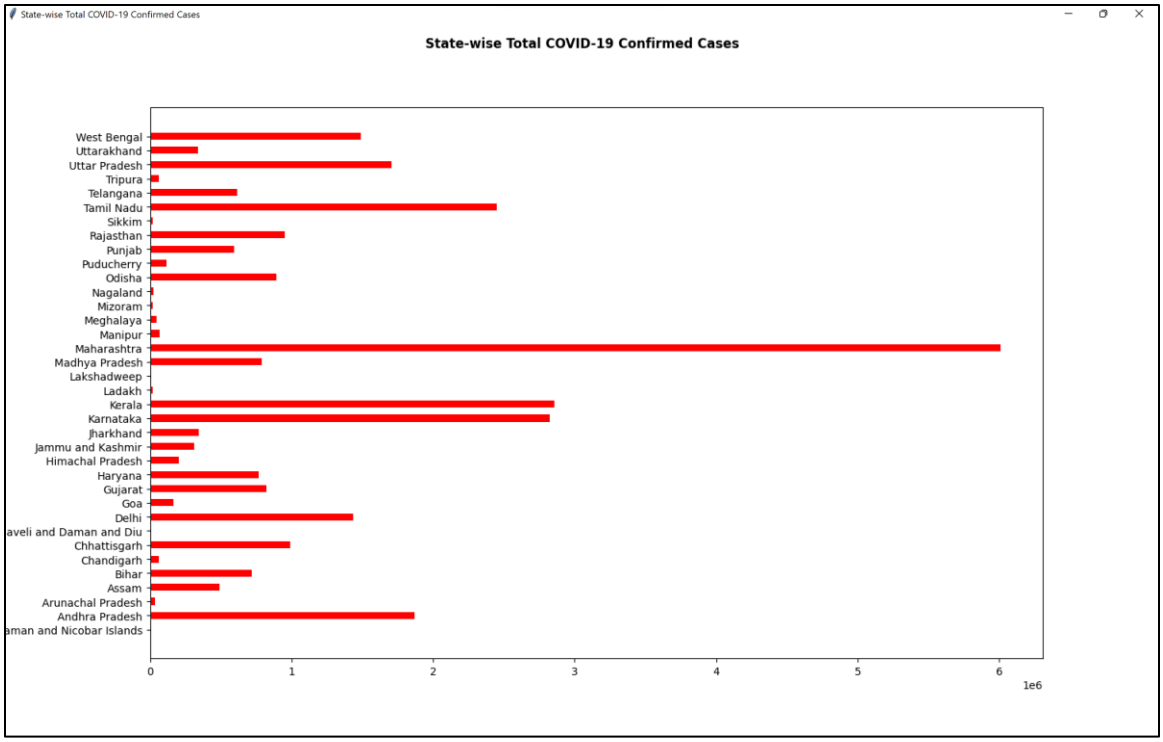
1: The COVID-19 Statistics Dashboard



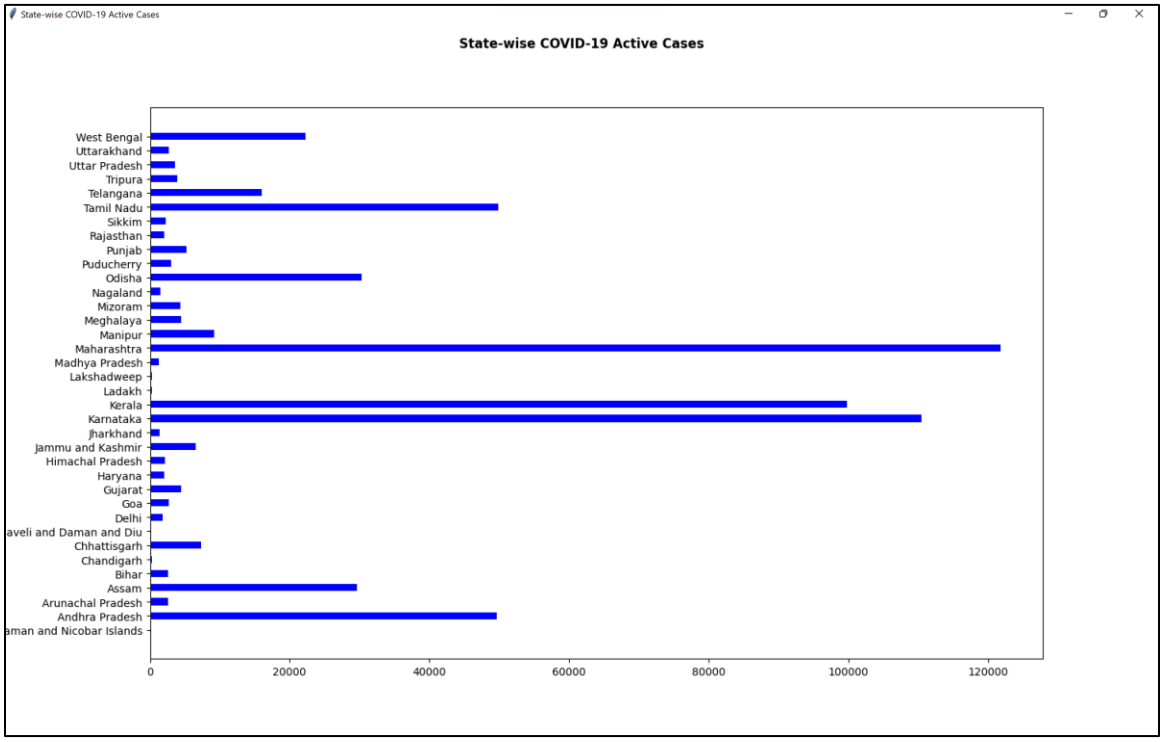
2: Desktop Notification on Launching the Application



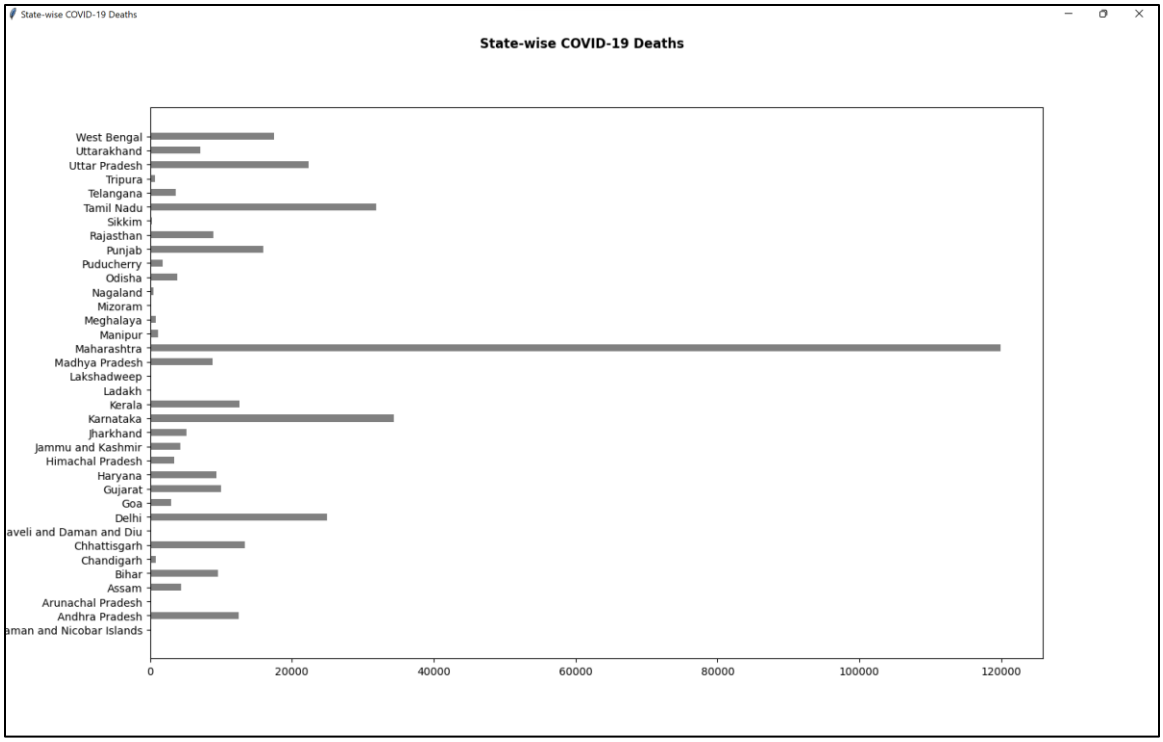
3: Desktop Notification with the Statistics as per user's State of Residence



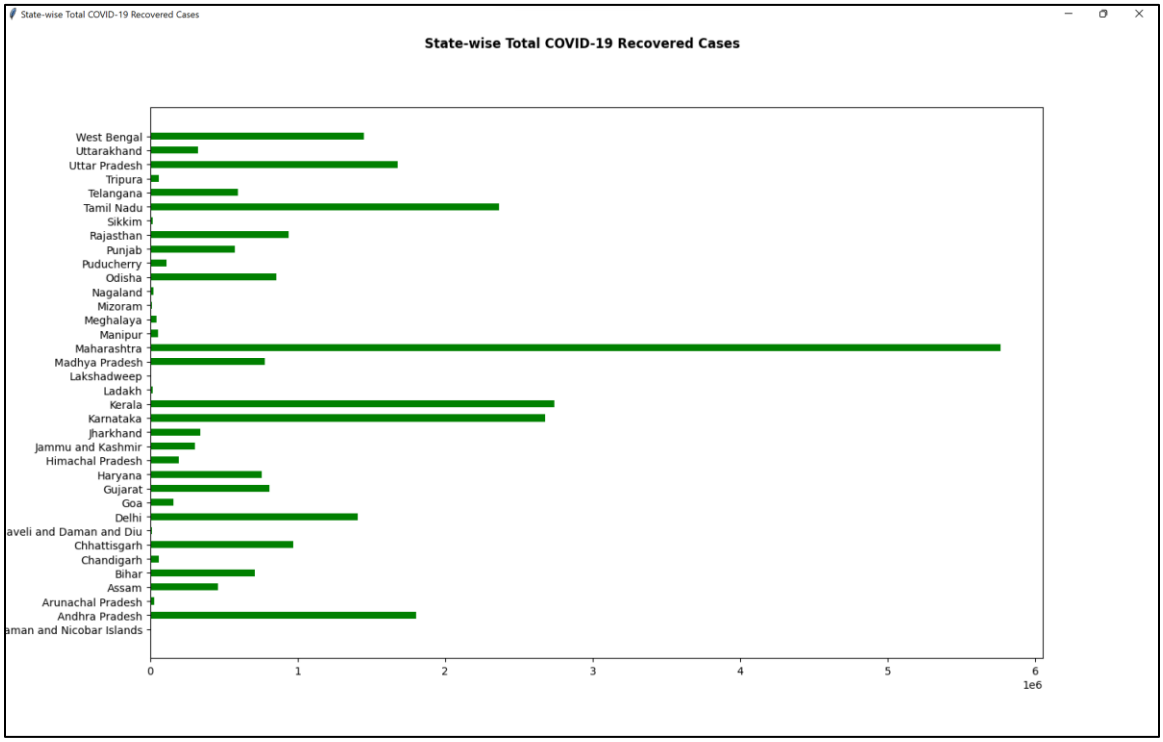
4: tkinter Window with Chart for Total COVID-19 Confirmed Cases



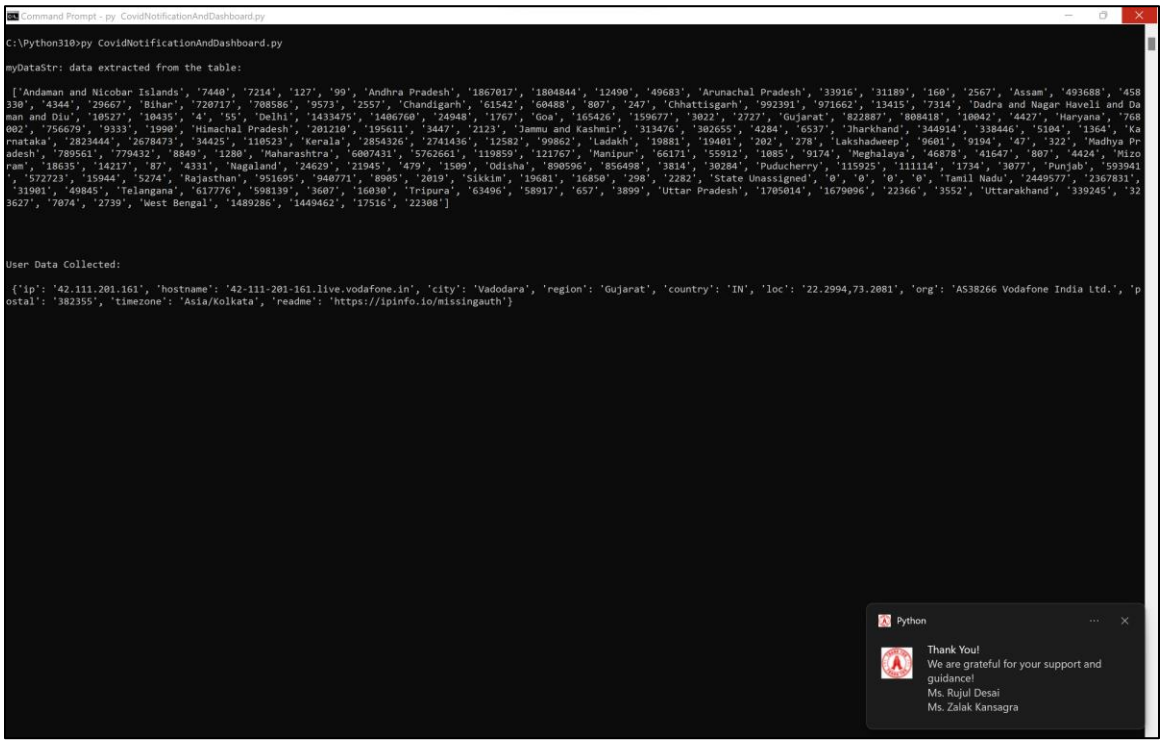
5: tkinter Window with Chart for COVID-19 Active Cases



6: tkinter Window with Chart for Total COVID-19 Deaths



7: tkinter Window with Chart for Total COVID-19 Recovered Cases



8: Desktop Notification on Closing the tkinter Windows, and the Data Extracted from the Websites

LEARNING OUTCOMES: KEY TAKE-AWAYS

During the development of the "COVID-19 Statistics Dashboard and Notification System", we learnt how to apply basic Python concepts – such as list manipulation, working with dictionaries, type casting, program control statements (loops and branch conditions), OOP (Object Oriented Programming) (classes, object, constructors, static methods etc.) and the use of default, positional and keyword arguments – to solve real-time problems.

This project also drove us to explore the use and functionalities of various Python libraries such as plyer, requests, bs4, matplotlib and tkinter. More about the methods used can be seen in the "Libraries Used: Their Purpose" section of this document.

We were exposed to the concept of “Web Scraping” and how it can be done in Python using the “requests” and “bs4” libraries.

We learnt how desktop notifications can be created using the “plyer” library, and how to work with (or create) GUIs (Graphics User Interface) using the “tkinter” library.

Our aim of including statistical charts in our application drove us to explore how to handle “button click events” in Python/ tkinter, and how graphs created using the “matplotlib” library can be embedded in a tkinter window.

TEAM EFFORTS: CONTRIBUTION OF MEMBERS

This project would have been incomplete without the inputs from all the group members in the form of ideas, references and code snippets. We sat together to develop and finalize this project, surfing the internet to troubleshoot wherever we were stuck. Our teachers and classmates have also been good supporters and guides throughout the developmental and troubleshooting processes.

REFERENCES:
THE HELP SITES

- *http://bioinfo.usu.edu/covidTracker/india.php*
- *https://ipinfo.io/*
- *https://colab.research.google.com/drive/1l3d4rcZjlZsBFHkK9Ftahy7N_SREobq1?authuser=1*
- *https://colab.research.google.com/drive/1Xl8v9O3iml7_beDI4CpFfldqOjwdPc5T?authuser=1*
- *https://colab.research.google.com/drive/1g90_xQmwt4yxlUjR0SGPzGBs_OhR6Guk?authuser=1*
- *https://colab.research.google.com/drive/1D-er6zbmjedDtpOFADCKWTZ2MQ2FBR15?authuser=1*
- *https://www.geeksforgeeks.org/class-method-vs-static-method-python/*
- *https://www.geeksforgeeks.org/python-desktop-notifier-using-plyer-module/*
- *https://www.geeksforgeeks.org/python-requests-tutorial/*
- *https://stackoverflow.com/questions/37168052/python-what-is-returned-when-i-use-requests-geturl-and-print-r-text*
- *https://www.geeksforgeeks.org/response-json-python-requests/*

- *https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/*
- *http://omz-software.com/pythonista/docs/ios/beautifulsoup_guide.html*
- *https://www.geeksforgeeks.org/python-gui-tkinter/*
- *https://python-forum.io/thread-26854.html*
- *https://riptutorial.com/tkinter/example/31885/customize-a-treeview*
- *https://www.tutorialspoint.com/how-can-i-set-the-row-height-in-tkinter-treeview*
- *https://yagisanatode.com/2018/02/23/how-do-i-change-the-size-and-position-of-the-main-window-in-tkinter-and-python-3/*
- *https://www.geeksforgeeks.org/python-tkinter-treeview-scrollbar/*
- *https://www.geeksforgeeks.org/python-introduction-matplotlib/*
- *https://www.geeksforgeeks.org/how-to-embed-matplotlib-charts-in-tkinter-gui/*