



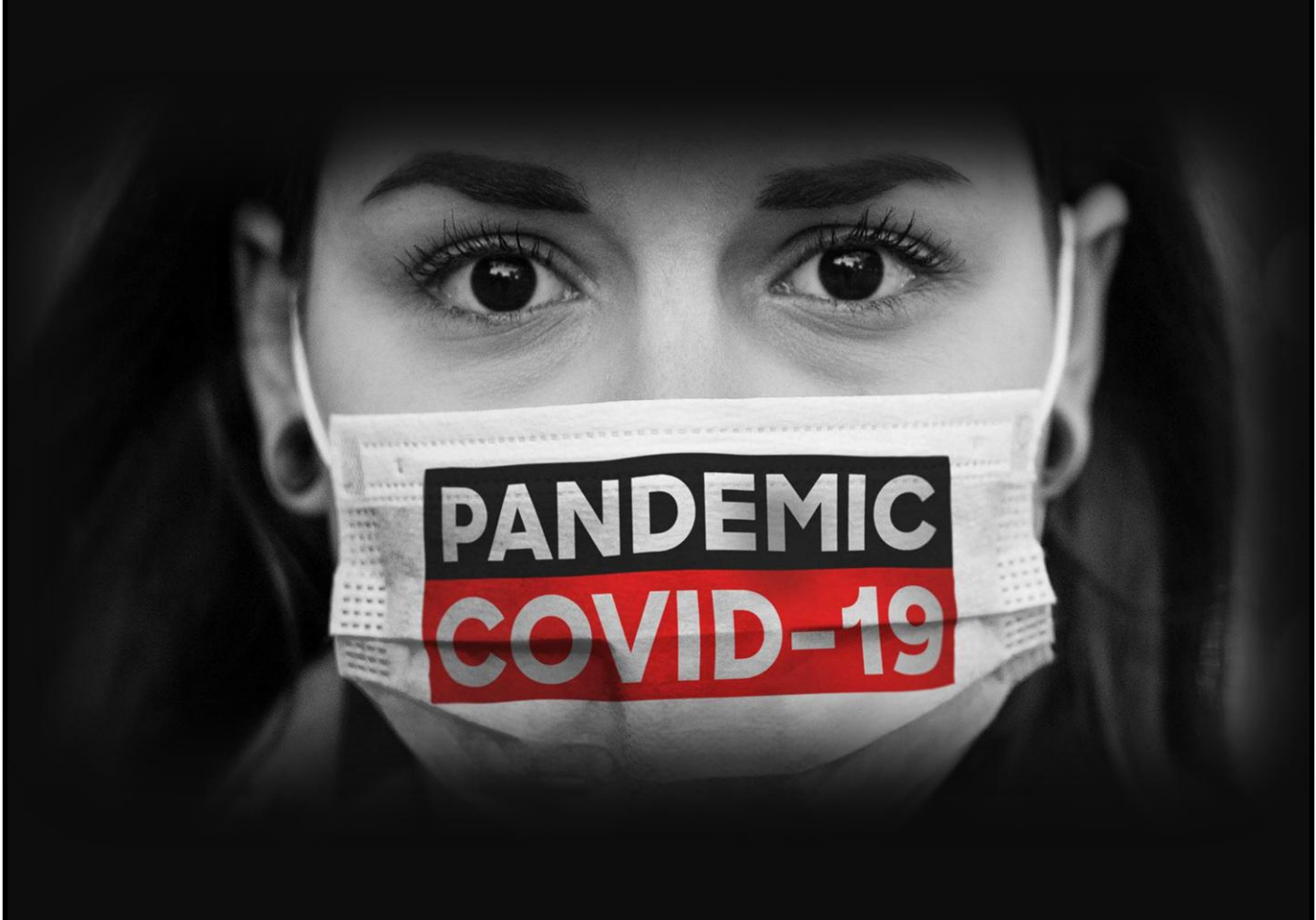
The Corona Prophecy

Done By:-

ABHINAND KRISHNAN H, XII A10

Github Repo Link:

<https://github.com/habhi1234/Corona-Prophecy>



BONAFIDE CERTIFICATE

Certified that this project is a bonafide work
of Master/~~Miss~~ **ABHINAND KRISHNAN H**

Roll No. _____ of Class XII of

Maharishi Vidya Mandir Sr. Sec. School, Chetpet,
Chennai during the year 2020 – 2021.

Date: 04/01/2021

Teacher – in – charge

Submitted for AISSCE Practical Examination
held in the **COMPUTER SCIENCE** Laboratory at
Maharishi Vidya Mandir Sr. Sec. School, Chetpet,
Chennai.

DATE :

INTERNAL EXAMINER

PRINCIPAL

EXTERNAL EXAMINER

SEAL

ACKNOWLEDGEMENT

I express my gratitude to our
Principal, Shri. G. HARIBABU of our institution for his
continuous support and encouragement.

I express my sincere thanks to my
Computer Science Teachers,

Smt. V. SUNDARI

Smt. G. ANITHA

Smt. H. NAZREEN BANU

for helping me to complete this project successfully.

I would also like to thank our Laboratory
Assistant **Smt. A. VIJAYA** for all the help extended
to us.

INDEX:

1.	PROJECT DEFINITION
2.	PROJECT ANALYSIS
3.	HARDWARE AND SOFTWARE REQUIREMENTS
4.	FUTURE ENHANCEMENTS
5.	SOURCE CODE
6.	OUTPUT
7.	BIBLIOGRAPHY

Project Definition

“We’re not just fighting a **pandemic**; we’re fighting an **infodemic**”

-**Tedros Ghebreyesus**
WHO’s director-general

The coronavirus outbreak came to light on December 31, 2019 when China informed the World Health Organization of a cluster of cases of pneumonia of an unknown cause in Wuhan City in Hubei Province. Subsequently the disease spread to more Provinces in China, and to the rest of the world. The WHO had declared it a pandemic. And it spread like wildfire

As COVID-19 surged across the country, data and predictive analytics were being used for “better allocating resources with regards to testing, personal protective equipment, medications and more.” In the fight against coronavirus, insight into preventive actions, population mobility, the spread of the disease, and the resilience of people and systems to cope with the virus, can help public health and humanitarian leaders respond more effectively to the COVID-19 epidemic. **For common man to understand the criticality of the situation, he needs access to right data, which is also easy to comprehend. This is what The Corona Prophecy Application does, provides the right info, which is easily understandable, directly into the hands of the public .**

Features:

This application has the following Features:

- A. It provides an interactive Graphical User Interface for the common man to understand
- B. Enhanced with advanced and exclusive Graphical Visualizations like Maps, Pie Charts, Graphs, etc.
- C. One stop location for all accurate data in an orderly fashion in regards to COVID-19 in India.
- D. Inbuilt Prophecy App, to give the common man an approximate idea of how the country's situation would be in near future, using a simplified SIR Model
- E. Gives data for the whole India and most of the states in the country.

Overall this app in itself is a treasure trove for anybody. With Unpredictable future at hand, the Data provided by the app Becomes invaluable to anybody in any field of work.

Project Analysis

COVID PROPHECY

HOME

STATS

**Primary Data
Visualization**

STATE

NATION

Visualizations

- 1) Graphs**
- 2) Bar Graphs**
- 3) Pie Chart**

- 1) Graphs**
- 2) Bar Graphs**
- 3) Pie Chart**
- 4) Maps**

For

- 1) Active Cases**
- 2) Recovered**
- 3) Death**
- 4) Total Infected**

COVID PROPHECY

About

PROPHECY

Prediction for:

STATE

NATION

Gives data:

- Susceptible
- Exposed
- Infected (asym)
- Infected (sym)
- Admitted
- Recovered
- Dead
- Total

Modules Used

❖ *Pandas Module and Numpy Module:*

- ❖ Pandas is an open source library in Python. It provides ready to use high-performance data structures and data analysis tools.
- ❖ Pandas module runs on top of NumPy and it is popularly used for data science and data analytics.
- ❖ NumPy is a low-level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality.
- ❖ DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure.
- ❖ Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc.

❖ *Geopandas Module:*

- ❖ GeoPandas is a project to add support for geographic data to pandas objects.
- ❖ The goal of GeoPandas is to make working with geospatial data in python easier. It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in python that would otherwise require a spatial database.

❖ *ImageTk Module:*

The ImageTk module contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images. It has been used in this project to import image files into the program.

❖ *Scipy.integrate:*

It takes as input arguments the function $f(x)$ to be integrated (the “integrand”), and the lower and upper limits a and b . It returns two values (in a tuple): the first one is the computed results and the second one is an estimation of the numerical error of that result.

❖ *Datetime:*

In Python, date and time are not a data type of its own, but a module name `datetime` can be imported to work with the date as well as time. `Datetime` module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and `datetime` are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

❖ *Matplotlib -*

- `Matplotlib` is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Create

- Develop **publication quality plots** with just a few lines of code
- Use **interactive figures** that can zoom, pan, update...

Customize

- **Take full control** of line styles, font properties, axes properties...
- **Export and embed** to a number of file formats and interactive environments

❖ *Mysql.connector:*

`MySQL Connector/Python` enables Python programs to access `MySQL` databases, using an API that is compliant with the `Python Database API Specification v2.0 (PEP 249)`. It is written in pure Python and does not have any dependencies except for the `Python Standard Library`.

❖ ***Response Module*** - The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

▪ Features:-

- Make requests using a variety of different HTTP methods such as GET, POST, and PUT
- Customize our requests by modifying headers, authentication, query strings, and message bodies
- Inspect the data we send to the server and the data the server sends back to us
- Work with SSL Certificate verification
- Use requests effectively using max_retries, timeout, Sessions, and Transport Adapters

❖ ***Tkinter Module*** –

- Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. Tkinter provides us with a variety of common GUI elements which we can use to build our interface – such as buttons, menus and various kinds of entry fields and display areas. We call these elements widgets.
- We can construct a tree of widgets for our GUI – each widget will have a parent widget, all the way up to the root window of our application. For example, a button or a text field needs to be inside some kind of containing window.

❖ ***Threading Module***-

- Threading module is used for creating, controlling and managing threads in python. A *threading.Timer* is a way to schedule a function to be called after a certain amount of time has passed. We can create a Timer by passing in a number of seconds to wait and a function to call.

❖ ***CSV Module-***

- ❖ The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel

❖ ***JSON Module –***

- This module contains functions for working with JSON data. You can use it to process JSON that is embedded in other file formats.

❖ ***WebBrowser Module-***

- The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.

❖ ***Warnings Module –***

- The warnings module suppresses repeated warnings from the same source to cut down on the annoyance of seeing the same message over and over. You can control the messages printed on a case-by-case basis using the -W option to the interpreter or by calling functions found in warnings from your code.

Prominent User Defined Functions

- ❖ **load_scr()**: Initiates the load screen . The sub function `mysql_update()` updates the existing database with all the new data from the API
- ❖ **Optionb(root)** : Responsible for the User Interactive Top Menu with Logo and buttons – Home, Stats, Prophecy, About.
- ❖ **homescr()**: When the Home button is clicked in Top Menu, the function is initiated which opens the Home screen which has Primary Data like ActiveCases, Deaths and Recovered.
- ❖ **Statsscr()**– When the Stats button is clicked in Top Menu, the function is initiated which opens the Stats screen which has two further Subdivisions : Nation wide and Statewise:
 - ❖ **Nationwide()**: The interface for choosing the required visualisations for the whole of India
 - ❖ **Statewise()**: The interface where we can choose the required state. Shows Graphs, Pie Charts and all the data concerned.
- ❖ **Prophecyscr()**– When the Prophecy button is clicked in Top Menu, the function is initiated which opens the Prophecy screen which has the input fields for the number of days to predict and State/Nation.
- ❖ **Aboutscr()**– When the About button is clicked in Top Menu, the function is initiated which opens the About screen which has all the information about the sources of the API, and Credits.

HARDWARE AND SOFTWARE REQUIREMENTS:

SPECS AND TYPE OF SYSTEM IT IS COMPATIBLE:

1. It requires python 3.5+
2. It requires all modules to be installed (like matplotlib, scipy, threading, tkinter, datetime, pillow, pandas, geopandas, mysql.connector, ImageTk, etc.)
3. The above modules can be installed by using pip install in the command prompt or in the power shell

FOR WINDOWS OS:

The binaries of AMD64 will also work on processors that implement the Intel 64 architecture (also known as x64 architecture and formerly known as both EM64T and x86-64)

FUTURE ENHANCEMENTS

Although the present version is self-sufficient, it of course can improve with further enhancements. Some of the ones which we think our Transformational are:

1. The same way COVID data for other countries can be obtained and shown. A separate screen for selecting countries can be provided for the same. Animation picture can be made using a series of days' data.
2. Lockdown effect can be interpreted. Showing how effective preventive measures taken by the government can help the common man realize his duties towards fight against COVID.
3. It can be modified further to understand the impact of the pandemic on various sectors and the economy of different countries .It can be further developed into an app , for contact tracing , providing us with the necessary information about the spread in different localities . It can be used effectively in case of any national emergency or natural disaster . Maven , a well known US software platform is specifically designed to support case management , and information integration . Our program can also be developed further which can be an essential tool for public health , providing the accurate analytical tools for collaboration thus controlling the situation (future pandemic) .

Source Code

```
import tkinter as tk
from tkinter import ttk
from PIL import ImageTk, Image
import pandas as pd
import geopandas as gpd
from scipy.integrate import odeint
import numpy as np
import math
import csv
import mysql.connector
import urllib.request as request
import matplotlib.pyplot as plt
import matplotlib.backends.backend_tkagg as tkagg
import matplotlib.ticker as ticker
import matplotlib.dates as mdates
import json
import threading
import datetime
import webbrowser
from datetime import date
import warnings
warnings.filterwarnings("ignore")

Path=r"E:\Corona Prophecy"

today = date.today()
datt=today.strftime("%Y-%m-%d")#"2020-08-03"
yester=today-datetime.timedelta(days=1)
dat = yesterday.strftime("%Y-%m-%d")
highConf=""
highDeath=""
highRec=""
inf=0
rec=0
dea=0
Statelist=["Andhra Pradesh","Arunachal Pradesh","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana","Himachal Pradesh","Jammu and Kashmir","Jharkhand","Karnataka","Kerala","Madhya Pradesh","Maharashtra","Manipur","Meghalaya","Mizoram","Nagaland","Odisha","Punjab","Rajasthan","Sikkim","Tamil Nadu","Telangana","Tripura","Uttar Pradesh","Uttarakhand","West Bengal","Andaman and Nicobar Islands","Chandigarh","Lakshadweep","Delhi","Puducherry"]
def raise_above_all(window):
    window.attributes('-topmost', 1)
    window.attributes('-topmost', 0)

def mysqltopd(TabName,Constraint, db_connection):
    df = pd.read_sql('SELECT * FROM {} where {}'.format(TabName,Constraint),
con=db_connection)
    df.head()
    return df
```



```

def barchart(t):
    t1=t.lower()
    statesE,num,rest=[],[],0

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345",database="coronadata")
mycursor=mydb.cursor()
mycursor.execute("SELECT S{} from covid_statewise where Sdate=\"{}\" ORDER BY S{} DESC".format(t1,dat,t1))
read=mycursor.fetchall()
nu= [int(j[0]) for j in [list(i) for i in read]]
mycursor.execute("SELECT Sname from covid_statewise where Sdate=\"{}\" ORDER BY S{} DESC".format(dat,t1))
read=mycursor.fetchall()
states= [str(j[0]) for j in [list(i) for i in read]]
for i in range (len(states)):
    if states[i]=="Dadra and Nagar Haveli and Daman and Diu":
        del states[i]
        del nu[i]
        break
mydb.close()
for j in range (len(nu)):
    if j<10:
        statesE.append(states[j])
        num.append(nu[j])
    else:
        rest+=nu[j]
statesE.append("Others")
num.append(rest)
x=[]
for i in range(1,len(num)+1):
    x.append(i)
plt.figure(figsize=(15,8))
axes=plt.axes()
plt.bar(x,num,tick_label=statesE,width=0.75,color='red')
locator={'death':1000,'infected':30000,'recovered':20000,'active':30000}
axes.yaxis.set_major_locator(ticker.MultipleLocator(locator[t1]))
plt.ylabel('Number of Cases',size=20)
plt.xticks(rotation='vertical',size=12,color='black')
plt.yticks(size=10,color='black')
str1='COVID 19 IN INDIA : '+ t +'\\n'
plt.title(str1,size=20,color='black')
plt.subplots_adjust(left=0.13,right=0.9,top=0.85,bottom=0.3)
plt.savefig("{}\\States\\Bar Graph\\Bar Graph - {}.png".format(Path,t), dpi=100)
plt.clf()
def piechart_india(t):
    t1=t.lower()

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345",database="coronadata")
mycursor=mydb.cursor()
mycursor.execute("SELECT S{} from covid_statewise where Sdate=\"{}\"".format(t1,dat))
read=mycursor.fetchall()
y= [int(j[0]) for j in [list(i) for i in read]]
mycursor.execute("SELECT Sname from covid_statewise where Sdate=\"{}\"".format(dat))
read=mycursor.fetchall()
x= [str(j[0]) for j in [list(i) for i in read]]
mydb.close()
tot=0

```

```

labels,slices,rest,exp=[],[],0,[]
for i in y:
    tot+=i
for j in range(len(x)):
    if (y[j]/tot)*100>3 :
        templ=x[j]
        labels.append(templ)
        temps=y[j]
        slices.append(temps)
    else :
        rest=rest+y[j]
slices.append(rest)
labels.append('Other States')
for i in range(len(slices)):
    exp.append(0.05)

plt.pie(slices,labels=labels,textprops=dict(size=10,color='black'),radius=1.3,auto
pct='%2.2f%%',explode=exp,shadow=False,startangle=90)
str1='Share of each State in India : '+'\n'+t+'\n\n\n'
plt.title(str1,color='black',size=13)
plt.savefig("{}\States\Pie Charts\Pie chart of India -
{}.png".format(Path,t),bbox_inches="tight", dpi=200,autoscale=True)
plt.clf()

def graphs_state(t,State=None):
    t1=t.lower()

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
,database="coronadata")
mycursor=mydb.cursor()
if State!=None:
    mycursor.execute("SELECT S{} from covid_statewise where
Sname=\"{}\"".format(t1,State))
    read=mycursor.fetchall()
    y= [int(j[0]) for j in [list(i) for i in read]]
    mycursor.execute("SELECT Sdate from covid_statewise where
Sname=\"{}\"".format(State))
    read=mycursor.fetchall()
    x= [str(j[0]) for j in [list(i) for i in read]]
    mydb.close()
else:
    mycursor.execute("SELECT I{} from covid_india".format(t1))
    read=mycursor.fetchall()
    y= [int(j[0]) for j in [list(i) for i in read]]
    mycursor.execute("SELECT Idate from covid_india")
    read=mycursor.fetchall()
    x= [str(j[0]) for j in [list(i) for i in read]]
    mydb.close()
    State="India"
plt.figure(figsize=(25,8))
axes=plt.axes()
axes.grid(linewidth=1,color='dimgray')
axes.set_facecolor('ivory')
if t1=="death":
    colr="darkslategray"
if t1=="recovered":
    colr="green"
if t1=="infected":
    colr="red"
if t1=="active":
    colr="skyblue"

```

```

axes.set_xlabel('\nDate',size=20,color=colr)
stry=t
axes.set_ylabel(stry,size=20,color=colr)

axes.xaxis.set_major_locator(ticker.MultipleLocator(7))
plt.xticks(rotation=30,size=8,color='black')
plt.yticks(size=15,color='black')
plt.tick_params(size=9,color='black')
st='COVID 19 IN '+State+ ':' + t
plt.title(st,size=25,color='black')

axes.plot(x,y,color=colr,marker='.',linewidth=2,markersize=8,markeredgecolor=colr)
plt.subplots_adjust(left=0.13,right=0.97,top=0.9,bottom=0.13)
plt.tight_layout()
plt.savefig("{}\States\Graphs\{} of {}.png".format(Path,t,State), dpi=100)
plt.clf()

def piechart_state(slices,Title):
    labels=['Active','Recovered','Deaths']
    exp=[]
    for i in range(len(slices)):
        exp.append(0.05)

plt.pie(slices,labels=labels,textprops=dict(size=10,color='black'),radius=1.3,auto
pct='%2.2f%%',explode=exp,shadow=False,startangle=90)
str1='Status of the '+' Confirmed cases in '+Title+'\n\n\n'
plt.title(str1,color='black',size=15)
plt.subplots_adjust(left=0.13,right=0.9,top=0.8,bottom=0.3)
plt.savefig("{}\States\Pie Charts\Pie chart of {}.png".format(Path,Title),
bbox_inches="tight",dpi=200,autoscale=True)
plt.clf()

def showmapAI(show_data,ToCompare,CoCo,Title):
    map_data = gpd.read_file("{}\States\Map Files\Indian_States.shp".format(Path))
    map_data.rename(columns = {'st_nm':'Sname'}, inplace = True)
    map_data['Sname'] = map_data['Sname'].str.replace('&','and')
    map_data['Sname'].replace('Arunachal Pradesh',
                             'Arunachal Pradesh', inplace = True)
    map_data['Sname'].replace('Telangana', 'Telengana', inplace = True)
    map_data['Sname'].replace('NCT of Delhi',
                             'Delhi', inplace = True)
    map_data['Sname'].replace('Andaman and Nicobar Island',
                             'Andaman and Nicobar Islands',
                             inplace = True)
    #show_data[ToCompare] = show_data[ToCompare].map(int)
    merged = pd.merge(map_data, show_data, on='Sname')
    merged.head()
    fig, ax = plt.subplots(1, figsize=(13, 12))
    ax.axis('off')

    ax.set_title(Title, fontsize=25)
    merged.plot(column =ToCompare,cmap=CoCo, linewidth=0.8, ax=ax,
edgecolor='0.8', legend = True)
    fig.tight_layout()
    fig.savefig("{}\States\Maps\{}.png".format(Path,Title), dpi=100)
    plt.clf()

```

```

def load_scr():
    global inf,rec,dea,highConf, highDeath, highRec,HConf, HRec, HDeath, Statelist
    loads=tk.Tk()
    raise_above_all.loads()
    loads.configure(bg='white')
    loads.geometry("500x220")
    loads.title("CoronaVirus Prophecy - Loading....")

Percent=tk.Label(master=loads,text="0%",fg="black",bg="white",font=("Baskerville
Old Face",14))
Percent.pack()
Info=tk.Label(master=loads,text="Loading Data from API! Please
Wait...",fg="black",bg="white",font=("Baskerville Old Face",14))
Info.pack()
Opimg = Image.open(r"{}\States\Other\Logo.jpg".format(Path))
Opimg=Opimg.resize((450,200), Image.ANTIALIAS)
Opimg = ImageTk.PhotoImage(Opimg)
Opimg1=tk.Label(image=Opimg)
Opimg1.pack()
def mysql_update():
    global inf, rec, dea, highConf, highDeath, highRec,HConf, HRec, HDeath,
Statelist

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
)
    mycursor=mydb.cursor()
    mycursor.execute("CREATE DATABASE IF NOT EXISTS coronadata")
    mydb.close()

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
,database="coronadata") #user needs to give own credentials
    mycursor=mydb.cursor()
    mycursor.execute("CREATE TABLE IF NOT EXISTS covid_india( Idate
char(10),Iinfected integer,Irecovered integer,Iactive integer,Ideath integer)")
    mycursor.execute("CREATE TABLE IF NOT EXISTS covid_statewise( Sdate
char(10),Sname char(60) NOT NULL,Sinfected integer,Srecovered integer,Sactive
integer,Sdeath integer)")
    data=get_data_datewise()
    mycursor.execute("DELETE FROM covid_india;")
    mydb.commit()
    mycursor.execute("DELETE FROM covid_statewise;")
    mydb.commit()
    mycursor.execute("DELETE FROM covid_ip;")
    mydb.commit()
    for i in range(len(data)) :
        I=[]
        Percent['text']=(str(round((i+1)*90/(len(data)),2))+"%")
        DATE=data[i]['date']
        Iinf=data[i]['total']['confirmed']
        Irecv=data[i]['total']['recovered']
        Iactv=data[i]['total']['active']
        Ideath=data[i]['total']['deaths']
        I.append(DATE)
        I.append(Iinf)
        I.append(Irecv)
        I.append(Iactv)
        I.append(Ideath)
        tI=tuple(I)

```

```

        for j in range(len(data[i]['statewise'])):
            S=[]
            Sname=data[i]['statewise'][j]['state']
            Sinf=data[i]['statewise'][j]['confirmed']
            Srecv=data[i]['statewise'][j]['recovered']
            Sactv=data[i]['statewise'][j]['active']
            Sdeath=data[i]['statewise'][j]['deaths']
            S.append(STATE)
            S.append(Sname)
            S.append(Sinf)
            S.append(Srecv)
            S.append(Sactv)
            S.append(Sdeath)
            tS=tuple(S)
            sqlcomm="INSERT INTO covid_statewise
VALUES(\"{0[0]}\",\"{0[1]}\",{0[2]},{0[3]},{0[4]},{0[5]})".format(tS)
            mycursor.execute(sqlcomm)
            sqlcomm1="INSERT INTO covid_india
VALUES(\"{0[0]}\",{0[1]},{0[2]},{0[3]},{0[4]})".format(tI)
            mycursor.execute(sqlcomm1)
            mydb.commit()
mydb.close()

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
,database="coronadata") #user needs to give own credentials
mycursor=mydb.cursor()
file=mysqltopd("covid_india","Idate=\"{}\"".format(dat),mydb)
for index, rows in file.iterrows():
    my_list =[rows.Iactive, rows.Irecovered, rows.Ideath]
    piechart_state(my_list,"India")
    Percent['text']="100.00%"
    mycursor.execute("SELECT Sname, Sinfected from covid_statewise where
Sinfected=(SELECT MAX(Sinfected) from covid_statewise where Sdate=\"{}\") and
Sdate=\"{}\"".format(dat,dat))
    read=mycursor.fetchall()
    highConf=read[0][0]
    HConf=int(read[0][1])
    mycursor.execute("SELECT Sname, Sdeath from covid_statewise where
Sdeath=(SELECT MAX(Sdeath) from covid_statewise where Sdate=\"{}\") and
Sdate=\"{}\"".format(dat,dat))
    read=mycursor.fetchall()
    highDeath=read[0][0]
    HDeath=int(read[0][1])
    mycursor.execute("SELECT Sname,Srecovered from covid_statewise where
Srecovered=(SELECT MAX(Srecovered) from covid_statewise where Sdate=\"{}\") and
Sdate=\"{}\"".format(dat,dat))
    read=mycursor.fetchall()
    highRec=read[0][0]
    HRec=int(read[0][1])
    mycursor.execute("SELECT Iinfected from covid_india where
Idate=\"{}\"".format(dat))
    read=mycursor.fetchall()
    inf=int(read[0][0])

```

```

mycursor.execute("SELECT Ideath from covid_india where Idate=\"{}\".format(dat))
read=mycursor.fetchall()
dea=int(read[0][0])
mycursor.execute("SELECT Irecovered from covid_india where
Idate=\"{}\".format(dat))
read=mycursor.fetchall()
rec=int(read[0][0])
loads.destroy()
mydb.close()
homescr()
timer1=threading.Timer(2.0,mysql_update)
timer1.start()
loads.mainloop()

def Optionb(root):
    global Titimg, himg, Proph, Stats, Abo
    Titimg = Image.open(r"{}\States\Other\Titlelogo.jpg".format(Path))
    Titimg=Titimg.resize((300,100), Image.ANTIALIAS)
    Titimg = ImageTk.PhotoImage(Titimg)
    himg = Image.open(r"{}\States\Other\Home.jpg".format(Path))
    himg=himg.resize((180,60), Image.ANTIALIAS)
    himg = ImageTk.PhotoImage(himg)
    Proph = Image.open(r"{}\States\Other\Prophecy.jpg".format(Path))
    Proph=Proph.resize((185,60), Image.ANTIALIAS)
    Proph = ImageTk.PhotoImage(Proph)
    Stats = Image.open(r"{}\States\Other\Stats.jpg".format(Path))
    Stats=Stats.resize((185,60), Image.ANTIALIAS)
    Stats = ImageTk.PhotoImage(Stats)
    Abo= Image.open(r"{}\States\Other\About.jpg".format(Path))
    Abo=Abo.resize((180,60), Image.ANTIALIAS)
    Abo= ImageTk.PhotoImage(Abo)
    Optionb=tk.Frame(master=root,relief=tk.RAISED,
width=int(root.winfo_screenwidth())/2,
height=int(root.winfo_screenheight())/10,bg="Royalblue")
    Title=tk.Label(master=Optionb,text="Corona
Prophecy",fg="white",bg="Royalblue1",borderwidth=0,font=("Baskerville Old
Face",32),image=Titimg)
    Title.pack(side=tk.LEFT, fill=tk.X,padx=(0,10))

    BHome=tk.Button(master=Optionb,image=himg,borderwidth=0,command=lambda:[root.destr
oy(),homescr()])
    BHome.pack(side=tk.LEFT,fill=tk.X,pady=10)

    BStats=tk.Button(master=Optionb,image=Stats,borderwidth=0,command=lambda:[root.des
troy(),Statsscr()])
    BStats.pack(side=tk.LEFT,fill=tk.X,padx=50,pady=10)

    BSIR=tk.Button(master=Optionb,image=Proph,borderwidth=0,command=lambda:[root.destr
oy(),Prophecyscr()])
    BSIR.pack(side=tk.LEFT,fill=tk.X,padx=(0,50),pady=10)

    BAbout=tk.Button(master=Optionb,image=Abo,borderwidth=0,command=lambda:[root.destr
oy(),Aboutscr()])
    BAbout.pack(side=tk.LEFT,fill=tk.X,padx=(0,150),pady=10)
    Optionb.grid(row=0, column=0)

```

```

def homescr():
    home=tk.Tk()
    raise_above_all(home)
    home.configure(bg='white')
    home.title("CoronaVirus Prophecy - Home")
    home.columnconfigure(0,weight=1)
    home.rowconfigure([0,1],weight=1)
    home.rowconfigure(2,weight=3)
    Optionb(home)
    Toggleb=tk.Frame(master=home,relief=tk.RAISED, width=home.winfo_screenwidth(),
height=int(home.winfo_screenheight())/10,bg='white')
    Toggleb.columnconfigure([0,1,2,3,4,5],weight=1)
    Toggleb.rowconfigure(0,weight=1)
    Inf = Image.open(r"{}\States\Other\Infected.jpg".format(Path))
    Inf=Inf.resize((260,135), Image.ANTIALIAS)
    Inf = ImageTk.PhotoImage(Inf)
    Infe=tk.Label(master=Toggleb,image=Inf,bg="white")
    Infe.grid(row=0, column=0,sticky="nsew")

    InfeI=tk.Label(master=Toggleb,text=str('{:,'}.format(inf)),fg="black",bg="white",font=("Baskerville Old Face",32))
    InfeI.grid(row=0,column=1,sticky="nsew")
    Death = Image.open(r"{}\States\Other\Deaths.jpg".format(Path))
    Death=Death.resize((270,135), Image.ANTIALIAS)
    Death = ImageTk.PhotoImage(Death)
    Deathe=tk.Label(master=Toggleb,image=Death,bg="white")
    Deathe.grid(row=0,column=2,sticky="nsew")

    DeatheI=tk.Label(master=Toggleb,text=str('{:,'}.format(dea)),fg="black",bg="white",font=("Baskerville Old Face",32))
    DeatheI.grid(row=0,column=3,sticky="nsew")
    Rec = Image.open(r"{}\States\Other\Recovered.jpg".format(Path))
    Rec=Rec.resize((280,135), Image.ANTIALIAS)
    Rec = ImageTk.PhotoImage(Rec)
    Reco=tk.Label(master=Toggleb,image=Rec,bg="white")
    Reco.grid(row=0,column=4,sticky="nsew")

    RecoI=tk.Label(master=Toggleb,text=str('{:,'}.format(rec)),fg="black",bg="white",font=("Baskerville Old Face",32))
    RecoI.grid(row=0,column=5,sticky="nsew")
    Toggleb.grid(row=1,column=0,sticky="nsew")
    PFrame=tk.Frame(master=home,relief=tk.RAISED, width=home.winfo_screenwidth(),
height=int(home.winfo_screenheight())/2,bg='white')
    PFrame.rowconfigure(0,weight=1)
    PFrame.columnconfigure(0,weight=1)
    PFrame.columnconfigure([1,2],weight=2)
    Map = Image.open(r"{}\States\Maps\COVID Hotspots.png".format(Path))
    Map=Map.resize((500,500), Image.ANTIALIAS)
    Map = ImageTk.PhotoImage(Map)
    Mapr=tk.Label(master=PFrame,image=Map,bg="white")
    Mapr.grid(row=0,column=2,sticky="nsew")
    Map1 = Image.open(r"{}\States\Pie Charts\Pie chart of India.png".format(Path))
    Map1=Map1.resize((450,450), Image.ANTIALIAS)
    Map1 = ImageTk.PhotoImage(Map1)
    Mapr1=tk.Label(master=PFrame,image=Map1,bg="white")
    Mapr1.grid(row=0,column=1,sticky="nsew")
    AFrame=tk.Frame(master=PFrame,relief=tk.RAISED,
width=(home.winfo_screenwidth())/7, height=int(home.winfo_screenheight())/2,bg='black')
    AFrame.rowconfigure([0,1,2],weight=1)
    AFrame.columnconfigure(0,weight=1)

```

```

HighC=tk.Label(master=AFrame,borderwidth=3,text="Highest"+"\\n"+"Confirmed Cases -
"+"\\n"+highConf+"\\n("+str('{:,'}.format(HConf))+")",fg="white",bg="Red",font=("Bas
kerville Old Face",16))
    HighD=tk.Label(master=AFrame,borderwidth=3,text="Highest"+"\\n"+"Deaths -
"+"\\n"+highDeath+"\\n("+str('{:,'}.format(HDeath))+")",fg="white",bg="grey",font=("
Baskerville Old Face",16))
    HighR=tk.Label(master=AFrame,borderwidth=3,text="Highest"+"\\n"+"Recovered
-
"+"\\n"+highRec+"\\n("+str('{:,'}.format(HRec))+")",fg="white",bg="green",font=("Bas
kerville Old Face",16))
    HighC.grid(row=0,column=0,sticky="nsew")
    HighD.grid(row=1,column=0,sticky="nsew")
    HighR.grid(row=2,column=0,sticky="nsew")
    AFrame.grid(row=0,column=0,sticky="nsew")
    PFrame.grid(row=2,column=0,sticky="nsew")
    home.mainloop()

def get_data_datewise():
    response=request.urlopen("https://api.rootnet.in/covid19-
in/unofficial/covid19india.org/statewise/history")
    cont=response.read()
    raw=json.loads(cont)
    data_datewise=[]
    for i in range(len(raw['data']['history'])) :
        d1={}
        d1['date']=raw['data']['history'][i]['day']
        d1['total']=raw['data']['history'][i]['total']
        x=raw['data']['history'][i]['statewise']
        for j in range(0,len(x)-1,1) :
            if x[j]['state']=='State Unassigned':
                x.pop(j)
        d1['statewise']=raw['data']['history'][i]['statewise']
        data_datewise.append(d1)
    return data_datewise

def Statsscr():
    global Statelist
    def NationW():
        global Text1, Text2, GChoose, VChoose, VieB1
        Text1=tk.Label(master=Cho,borderwidth=3,text="Graph
Type:",fg="black",bg="white",font=("Baskerville Old Face",10))
        Text1.grid(row=0,column=0,sticky="nsew")
        GChoose=ttk.Combobox(Cho,
values=("Graph","PieChart","BarGraph","Map"), state="readonly")
        GChoose.grid(row=0,column=1,sticky="nsew")

        Text2=tk.Label(master=Cho,borderwidth=3,text="Visualise:",fg="black",bg="white",fo
nt=("Baskerville Old Face",10))
        Text2.grid(row=0,column=2,sticky="nsew")
        VChoose=ttk.Combobox(Cho,
values=("Active","Recovered","Death","Infected"), state="readonly")
        VChoose.grid(row=0,column=3,sticky="nsew")
        VieB1=tk.Button(master=Cho,text="View",
bg="white",command=lambda:[NationView(GChoose.get(),VChoose.get())])
        VieB1.grid(row=0,column=4,sticky="nsew")

```



```

def NationView(Visual,Type):
    List=View.grid_slaves()
    for i in List:
        i.grid_remove()
    global Grap, Pi, Bar, Mapp
    if Visual=="Graph":
        graphs_state(Type)
        Grap = Image.open(r"{}\States\Graphs\{} of
{}.png".format(Path,Type,"India"))
        Grap=Grap.resize((1200,600), Image.ANTIALIAS)
        Grap = ImageTk.PhotoImage(Grap)
        Graph=tk.Label(master=View,image=Grap,bg="white")
        Graph.grid(row=0,column=1,sticky="nsew")

Dummy1=tk.Label(master=View,bg="white",height=int(int(sta.wininfo_screenheight())/20
))

Dummy2=tk.Label(master=View,bg="white",height=int(int(sta.wininfo_screenheight())/20
))

        Dummy1.grid(row=0,column=0,sticky="nsew")
        Dummy2.grid(row=0,column=2,sticky="nsew")
    if Visual=="PieChart":
        piechart_india(Type)
        Pi = Image.open(r"{}\States\Pie Charts\Pie chart of India -
{}.png".format(Path,Type))
        Pi=Pi.resize((550,500), Image.ANTIALIAS)
        Pi = ImageTk.PhotoImage(Pi)
        Pie=tk.Label(master=View,image=Pi,bg="white")
        Pie.grid(row=0,column=1,sticky="nsew")

Dummy1=tk.Label(master=View,bg="white",height=int(int(sta.wininfo_screenheight())/20
))

Dummy2=tk.Label(master=View,bg="white",height=int(int(sta.wininfo_screenheight())/20
))

        Dummy1.grid(row=0,column=0,sticky="nsew")
        Dummy2.grid(row=0,column=2,sticky="nsew")
    if Visual=="Map":

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
,database="coronadata") #user needs to give own credentials
    mycursor=mydb.cursor()
    if Type=="Infected":
        colr="plasma_r"
    if Type=="Active":
        colr="Oranges"
    if Type=="Recovered":
        colr="Greens"
    if Type=="Death":
        colr="Purples"

showmapAI(show_data=mysqltopd("covid_statewise","Sdate=\"{}\"".format(dat),mydb),T
oCompare="S{}".format(Type.lower()),CoCo=colr,Title="COVID-19_{}".format(Type))
    mydb.close()

```

```

piechart_india(Type)
    Mapp = Image.open(r"{}\States\Maps\COVID-
19_{}.png".format(Path,Type))
    Mapp=Mapp.resize((500,500), Image.ANTIALIAS)
    Mapp = ImageTk.PhotoImage(Mapp)
    Mapp=tk.Label(master=View,image=Mapp,bg="white")
    Mapp.grid(row=0,column=1,sticky="nsew")

Dummy1=tk.Label(master=View,bg="white",height=int(int(sta.winfo_screenheight())/20
))

Dummy2=tk.Label(master=View,bg="white",height=int(int(sta.winfo_screenheight())/20
))

    Dummy1.grid(row=0,column=0,sticky="nsew")
    Dummy2.grid(row=0,column=2,sticky="nsew")
if Visual=="BarGraph":
    barchart(Type)
    Bar = Image.open(r"{}\States\Bar Graph\Bar Graph -
{}.png".format(Path,Type))
    Bar=Bar.resize((1200,600), Image.ANTIALIAS)
    Bar = ImageTk.PhotoImage(Bar)
    BarG=tk.Label(master=View,image=Bar,bg="white")
    BarG.grid(row=0,column=1,sticky="nsew")

Dummy1=tk.Label(master=View,bg="white",height=int(int(sta.winfo_screenheight())/20
))

Dummy2=tk.Label(master=View,bg="white",height=int(int(sta.winfo_screenheight())/20
))

    Dummy1.grid(row=0,column=0,sticky="nsew")
    Dummy2.grid(row=0,column=2,sticky="nsew")

def StateView(State):
    global Map1r,Map2r, Map3r,Piye
    List=View.grid_slaves()
    for i in List:
        i.grid_remove()

mydb=mysql.connector.connect(host="localhost",port=3306,user="root",passwd="12345"
,database="coronadata")
    mycursor=mydb.cursor()
    mycursor.execute("SELECT Sinfected,Sactive,Sdeath,Srecovered from
covid_statewise where Sname=\"{}\" and Sdate=\"{}\"".format(State,dat))
    read=mycursor.fetchall()
    other= [int(j) for j in list(read[0])]
    mydb.close()
    piechart_state([other[1],other[3],other[2]],State)
    for i in ["Infected","Death","Recovered"]:
        graphs_state(i,State=State)
def SF():
    if TextF.winfo_ismapped():
        TextF.grid_remove()
        Pie.grid(row=0,column=1)
    if Pie.winfo_ismapped():
        Pie.grid_remove()
        Map1.grid(row=0,column=1,sticky="nsew")
    if Map1.winfo_ismapped():
        Map1.grid_remove()
        Map2.grid(row=0,column=1,sticky="nsew")
    if Map2.winfo_ismapped():
        Map2.grid_remove()
        Map3.grid(row=0,column=1,sticky="nsew")

```

```

def SB():
    if Map1.winfo_ismapped():
        Map1.grid_remove()
        Pie.grid(row=0,column=1,sticky="nsew")
    if Pie.winfo_ismapped():
        Pie.grid_remove()
        TextF.grid(row=0,column=1,sticky="nsew")
    if Map3.winfo_ismapped():
        Map3.grid_remove()
        Map2.grid(row=0,column=1,sticky="nsew")
    if Map2.winfo_ismapped():
        Map2.grid_remove()
        Map1.grid(row=0,column=1,sticky="nsew")
    SlideB=tk.Button(master=View,text="<",
bg="white",command=SB,height=int(int(sta.winfo_screenheight())/20))
    SlideF=tk.Button(master=View,text=">",
bg="white",command=SF,height=int(int(sta.winfo_screenheight())/20))
    TextF=tk.Frame(master=View,width=(sta.winfo_screenwidth()),
height=3*int(sta.winfo_screenheight())/5,bg="white")
    TextF.columnconfigure([0,1,2],weight=1)
    TextF.rowconfigure(0,weight=1)
    TextF.rowconfigure([1,2],weight=4)

    StateN=tk.Label(master=TextF,borderwidth=3,text=State,fg="black",bg="white",font=(
"Baskerville Old Face",28))

    StateInf=tk.Label(master=TextF,text="Infected:"+"\\n"+str('{:,}'.format(other[0])),
fg="white",bg="orange red",font=("Baskerville Old Face",20))

    StateAct=tk.Label(master=TextF,text="Active:"+"\\n"+str('{:,}'.format(other[1])),fg
="black",bg="OliveDrab1",font=("Baskerville Old Face",20))

    StateDea=tk.Label(master=TextF,text="Deaths:"+"\\n"+str('{:,}'.format(other[2])),fg
="white",bg="cadet blue",font=("Baskerville Old Face",20))

    StateRec=tk.Label(master=TextF,text="Recovered:"+"\\n"+str('{:,}'.format(other[3]))
,fg="black",bg="SpringGreen2",font=("Baskerville Old Face",20))
    if other[0]!=0:
        StateFat=tk.Label(master=TextF,text="Fatality
Rate:"+"\\n"+str(round((other[2])/(other[0])*100,2))+"%",fg="white",bg="purple2",fo
nt=("Baskerville Old Face",20))
        StateRe=tk.Label(master=TextF,text="Recovery
Rate:"+"\\n"+str(round((other[3])/(other[0])*100,2))+"%",fg="black",bg="light
goldenrod",font=("Baskerville Old Face",20))
    else:
        StateFat=tk.Label(master=TextF,text="Fatality
Rate:"+"\\n"+"NA"+"%",fg="white",bg="purple2",font=("Baskerville Old Face",20))
        StateRe=tk.Label(master=TextF,text="Recovery
Rate:"+"\\n"+"NA"+"%",fg="black",bg="light goldenrod",font=("Baskerville Old
Face",20))
    StateN.grid(row=0,column=0,columnspan=3,sticky="nsew")
    StateInf.grid(row=1,column=0,sticky="nsew",padx=5,pady=5)
    StateAct.grid(row=1,column=1,sticky="nsew",padx=5,pady=5)
    StateDea.grid(row=1,column=2,sticky="nsew",padx=5,pady=5)
    StateRec.grid(row=2,column=0,sticky="nsew",padx=5,pady=5)
    StateFat.grid(row=2,column=1,sticky="nsew",padx=5,pady=5)

```

```

StateRe.grid(row=2,column=2,sticky="nsew",padx=5,pady=5)
    TextF.grid(row=0,column=1,sticky="nsew")
    Piye = Image.open(r"{}\States\Pie Charts\Pie chart of
{}.png".format(Path,State))
    Piye = Piye.resize((500,550), Image.ANTIALIAS)
    Piye = ImageTk.PhotoImage(Piye)
    Map1r = Image.open(r"{}\States\Graphs\{} of
{}.png".format(Path,"Infected",State))
    Map1r=Map1r.resize((1200,600), Image.ANTIALIAS)
    Map1r = ImageTk.PhotoImage(Map1r)
    Map2r = Image.open(r"{}\States\Graphs\{} of {}.png".format(Path,"Death",State))
    Map2r=Map2r.resize((1200,600), Image.ANTIALIAS)
    Map2r = ImageTk.PhotoImage(Map2r)
    Map3r = Image.open(r"{}\States\Graphs\{} of
{}.png".format(Path,"Recovered",State))
    Map3r=Map3r.resize((1200,600), Image.ANTIALIAS)
    Map3r = ImageTk.PhotoImage(Map3r)
    Pie=tk.Label(master=View,image=Piye,bg="white")
    Map1=tk.Label(master=View,image=Map1r,bg="white")
    Map2=tk.Label(master=View,image=Map2r,bg="white")
    Map3=tk.Label(master=View,image=Map3r,bg="white")
    SlideF.grid(row=0,column=2,sticky="nsew")
    SlideB.grid(row=0,column=0,sticky="nsew")
def inact(button):
    button['state']=tk.DISABLED
def act(button):
    button['state']=tk.NORMAL
def StateW():
    global Text1, Text2, GChoose, VChoose, ViewB1
    Text1=tk.Label(master=Cho,borderwidth=3,text="Choose
State:",fg="black",bg="white",font=("Baskerville Old Face",10))
    Text1.grid(row=0,column=0,sticky="nsew")
    GChoose=ttk.Combobox(Cho, values=tuple(Statelist),state="readonly")
    GChoose.grid(row=0,column=1,sticky="nsew")

Text2=tk.Label(master=Cho,borderwidth=3,text="Visualise:",fg="grey",bg="white",font=("Baskerville Old Face",10))
    Text2.grid(row=0,column=2,sticky="nsew")
    VChoose=ttk.Combobox(Cho, values=("Active","Recovered","Death","Infected"),
state="disabled")
    VChoose.grid(row=0,column=3,sticky="nsew")
    VieB1=tk.Button(master=Cho,text="View",
bg="white",command=lambda:[StateView(GChoose.get())])
    VieB1.grid(row=0,column=4,sticky="nsew")
sta=tk.Tk()
    raise_above_all(sta)
    sta.configure(bg='white')
    sta.title("CoronaVirus Prophecy - Stats")
    sta.state("zoomed")
    sta.columnconfigure(0,weight=1)
    sta.rowconfigure(0,weight=1)
    sta.rowconfigure([1,2],weight=1)
    sta.rowconfigure(3,weight=6)
    Optionb(sta)
    Cho=tk.Frame(master=sta,relief=tk.RAISED, width=(sta.winfo_screenwidth()),
height=int(sta.winfo_screenheight())/15,bg='white')
    Cho.rowconfigure(0,weight=1)
    Cho.columnconfigure([0,1,2,3,4],weight=1)
    Choose=tk.Frame(master=sta,relief=tk.RAISED, width=(sta.winfo_screenwidth()),
height=int(sta.winfo_screenheight())/15,bg='white')
    Choose.columnconfigure([0,1],weight=1)
    Choose.rowconfigure(0,weight=1)

```

```

Statew=tk.Button(master=Choose,text="Statewise",command=lambda:[StateW(),inact(Statew),a
ct(Nationw)])

Nationw=tk.Button(master=Choose,text="NationWide",command=lambda:[NationW(),inact(Nation
w),act(Statew)])
    Statew.grid(row=0,column=1,sticky="nsew")
    Nationw.grid(row=0,column=0,sticky="nsew")
    Choose.grid(row=1,column=0,sticky="nsew")
    Cho.grid(row=2,column=0)
    View=tk.Frame(master=sta, width=(sta.winfo_screenwidth()),
height=3*int(sta.winfo_screenheight())/5,bg="white")
    View.rowconfigure(0,weight=1)
    View.columnconfigure([0,2],weight=1)
    View.columnconfigure(1,weight=30)
    View.pack_propagate()
    View.grid(row=3,column=0,sticky="nsew")

def Prophecyscr():
    Prop=tk.Tk()
    raise_above_all(Prop)
    Prop.configure(bg='white')
    Prop.title("CoronaVirus Prophecy - The Prophecy")
    Prop.columnconfigure(0,weight=1)
    Prop.rowconfigure(0,weight=1)
    Prop.rowconfigure(1,weight=6)
    Optionb(Prop)
    View=tk.Frame(master=Prop, width=(Prop.winfo_screenwidth()),
height=3*int(Prop.winfo_screenheight())/5,bg="white")
    View.rowconfigure([0,2],weight=1)
    View.rowconfigure(1,weight=6)
    View.rowconfigure(3,weight=2)
    View.columnconfigure([0,1,2,3],weight=1)
    View.pack_propagate()
    View.grid(row=1,column=0,sticky="nsew")
    Content1='''The COVID-19 pandemic, also known as the coronavirus pandemic, is an
ongoing global pandemic of coronavirus disease 2019 (COVID-19), caused
by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).
The pandemic has caused global social and economic disruption.

The SIR model is a simple model of an epidemic of an infectious
disease in a large population. It is an epidemiological model
that computes the theoretical number of people infected with a contagious
illness in a closed population over time. An extended version of the basic SIR
model has been built and is used for predicting purpose'''
    Tit=tk.Label(master=View,borderwidth=3,text="The
Prophecy",fg="black",bg="white",font=("Baskerville Old Face",28))

    Con=tk.Label(master=View,borderwidth=3,text=Content1,fg="black",bg="white",font=("Basker
ville Old Face",18))
    Butt=tk.Button(master=View,text="Predict
Now!",fg="white",bg="black",command=lambda:[ProphView(int(Predicnum.get()),Sta.get())],
height=2)
    Lab2=tk.Label(master=View,borderwidth=3,text="Region
:",fg="black",bg="white",font=("Baskerville Old Face",15))
    Lab2.grid(row=2,column=2,sticky="nsew")
    Sta=ttk.Combobox(View, values=["India"]+Statelist,state="readonly")
    Sta.grid(row=2,column=3,sticky="nsew")
    Lab1=tk.Label(master=View,borderwidth=3,text="No. of Days to
Predict:",fg="black",bg="white",font=("Baskerville Old Face",15))
    Lab1.grid(row=2,column=0,sticky="nsew")
    Predicnum=ttk.Combobox(View, values=list(range(100,301,10)),state="readonly")
    Predicnum.grid(row=2,column=1,sticky="nsew")

```

```

Tit.grid(row=0,column=0,columnspan=4,sticky="nsew",pady=50)
Con.grid(row=1,column=0,columnspan=4,sticky="nsew",pady=[10,120])
Butt.grid(row=3,column=0,columnspan=4,sticky="nsew",pady=[20,0])
def ProphView(Pred, State):
    List=View.grid_slaves()
    for i in List:
        i.grid_remove()
    def R(t):
        return (R0start-R0end)/(1+(math.e)**(-1*(t-x0))) + R0end
    def b(t):
        return R(t)*0.25
    def r1(t):
        return 0.4
    def d1(t):
        return 0.09
    def cp(y,t,N,b,c,p,a1,a2,r,r1,d1,pr,pd,par,pad):
        S,E,Ia,Is,A,R,D,n=y
        N=S+E+Ia+Is+A+R+D
        dndt=t/300
        dSdt=-b(t)*Is*S/N
        dEdt=b(t)*Is*S/N-c*E
        dIadt=c*p*E-a1*n*Ia-r1(t)*pr*Ia-d1(t)*pd*Ia
        dIsdt=c*(1-p)*E-a2*n*Is-r1(t)*pr*Is-d1(t)*pd*Is
        dAdt=a1*n*Ia+a2*n*Is-r*par*A-(1-r)*pad*A
        dRdt=r*par*A+r1(t)*pr*Ia+r1(t)*pr*Is
        dDdt=(1-r)*pad*A+d1(t)*pd*Is+d1(t)*pd*Ia
        return dSdt,dEdt,dIadt,dIsdt,dAdt,dRdt,dDdt,dndt
    with open(r"{ }\CSVFiles\InfoIndia.csv".format(Path),"r") as file:
        dreader=csv.reader(file)
        for row in dreader:
            if row[0]==State:
                Pop=row[2]
                break
    if State=="India":
        Pop=1381159795
    N=int(Pop)
    R0start=0.1
    R0end=8
    x0=55
    p=0.5
    c=0.5
    a1,a2=0.1,0.06
    r=0.8
    pr,pd,par,pad=0.4,0.04,0.5,0.04
    t=np.linspace(0,Pred,Pred+1)
    S0,E0,Ia0,Is0,A0,R0,D0,n0=N-1,1,0,0,0,0,0,0.6
    y0=S0,E0,Ia0,Is0,A0,R0,D0,n0
    info=odeint(cp,y0,t,args=(N,b,c,p,a1,a2,r,r1,d1,pr,pd,par,pad))
    S,E,Ia,Is,A,R,D,n=info.T
    def plotcurve(t, S, E, Ia, Is, A, R, D,Scr):
        f, ax = plt.subplots(1,1,figsize=(10,4))
        ax.plot(t, S, 'b', alpha=0.7, linewidth=2, label='Susceptible')
        ax.plot(t, E, 'y', alpha=0.7, linewidth=2, label='Exposed')
        ax.plot(t, Ia, 'r', alpha=0.7, linewidth=2, label='Infected (asym)')
        ax.plot(t, Is, 'c', alpha=0.7, linewidth=2, label='Infected (sym)')
        ax.plot(t, A, 'm', alpha=0.7, linewidth=2, label='Admitted')
        ax.plot(t, R, 'g', alpha=0.7, linewidth=2, label='Recovered')
        ax.plot(t, D, 'k', alpha=0.7, linewidth=2, label='Dead')
        ax.plot(t, S+E+Ia+Is+A+R+D, 'c--', alpha=0.7, linewidth=2, label='Total')
        ax.set_title("\nCOVID-19 Prophecy - {}".format(State),fontsize=20,color="b")
        ax.set_xlabel('Time (days)')

```

```

ax.yaxis.set_tick_params(length=0)
    ax.xaxis.set_tick_params(length=0)
    ax.grid(b=True, which='major', c='w', lw=2, ls='-')
    legend = ax.legend()
    legend.get_frame().set_alpha(0.5)
    for spine in ('top', 'right', 'bottom', 'left'):
        ax.spines[spine].set_visible(False)
    Fra=tk.Frame(master=Scr, width=(Prop.wininfo_screenwidth()),
height=3*int(Prop.wininfo_screenheight())/5,bg="white")
    canvas = tkagg.FigureCanvasTkAgg(f, Fra)
    canvas.get_tk_widget().pack(fill=tk.BOTH)
    tkagg.NavigationToolbar2Tk(canvas, Fra)
    Fra.grid(row=0,rowspan=2,column=0, columnspan=4, sticky="nsew")
    plotcurve(t,S,E,Ia,Is,A,R,D,View)
    Titl=tk.Label(master=View,borderwidth=3,text="Stay Home, Stay
Safe!",fg="black",bg="white",font=("Baskerville Old Face",28))
    Titl.grid(row=2,column=0,columnspan=4,sticky="nsew",pady=10)

Butto=tk.Button(master=View,text="Back",fg="white",bg="black",command=lambda:[Prop.destr
oy(),Prophecyscr()])
    Butto.grid(row=3,column=0,columnspan=4,sticky="nsew",pady=[20,0])

def Aboutscr():
    Ab=tk.Tk()
    raise_above_all(Ab)
    Ab.configure(bg='white')
    Ab.title("CoronaVirus Prophecy - About")
    Ab.columnconfigure([0,1],weight=1)
    Ab.rowconfigure([0,2,3],weight=2)
    Ab.rowconfigure(1,weight=5)
    def callback1():
        webbrowser.open_new(r"https://www.mygov.in/covid-19")
    def callback2():
        webbrowser.open_new(r"https://www.mohfw.gov.in/")
    Optionb(Ab)
    AboutI="""
The Coronavirus Prophecy

Last updated: {}

This Coronavirus dashboard provides an overview of the 2019 Novel Coronavirus COVID-19
(2019-nCoV) epidemic.
The Dashboard has been created in Python Language

The raw data for this dashboard is pulled from the
API COVID - INDIA. The data and dashboard is refreshed on a daily basis.

Contribution
    The Basic DB structure and Matplotlib interface - A Niveath, XII A10, MVMSSS
    The Mapping of Data using Geopandas - Eashvar Srinivasan, XII A10, MVMSSS
    The Prophecy and SIR Modelling - H Abhinand Krishnan, XII A10, MVMSSS
    Graphics, User Interface and Compilation - K P Annirudh, XII A10,
MVMSSS""".format(datt)
    AbI=tk.Label(master=Ab,text=AboutI,fg="black",bg="white",font=("Baskerville Old
Face",14))
    AbI.grid(row=1,column=0,columnspan=2,pady=[120,120])
    View=tk.Frame(master=Ab, width=(Ab.wininfo_screenwidth()),
height=1*int(Ab.wininfo_screenheight())/5,bg="white")
    View.rowconfigure(0,weight=1)

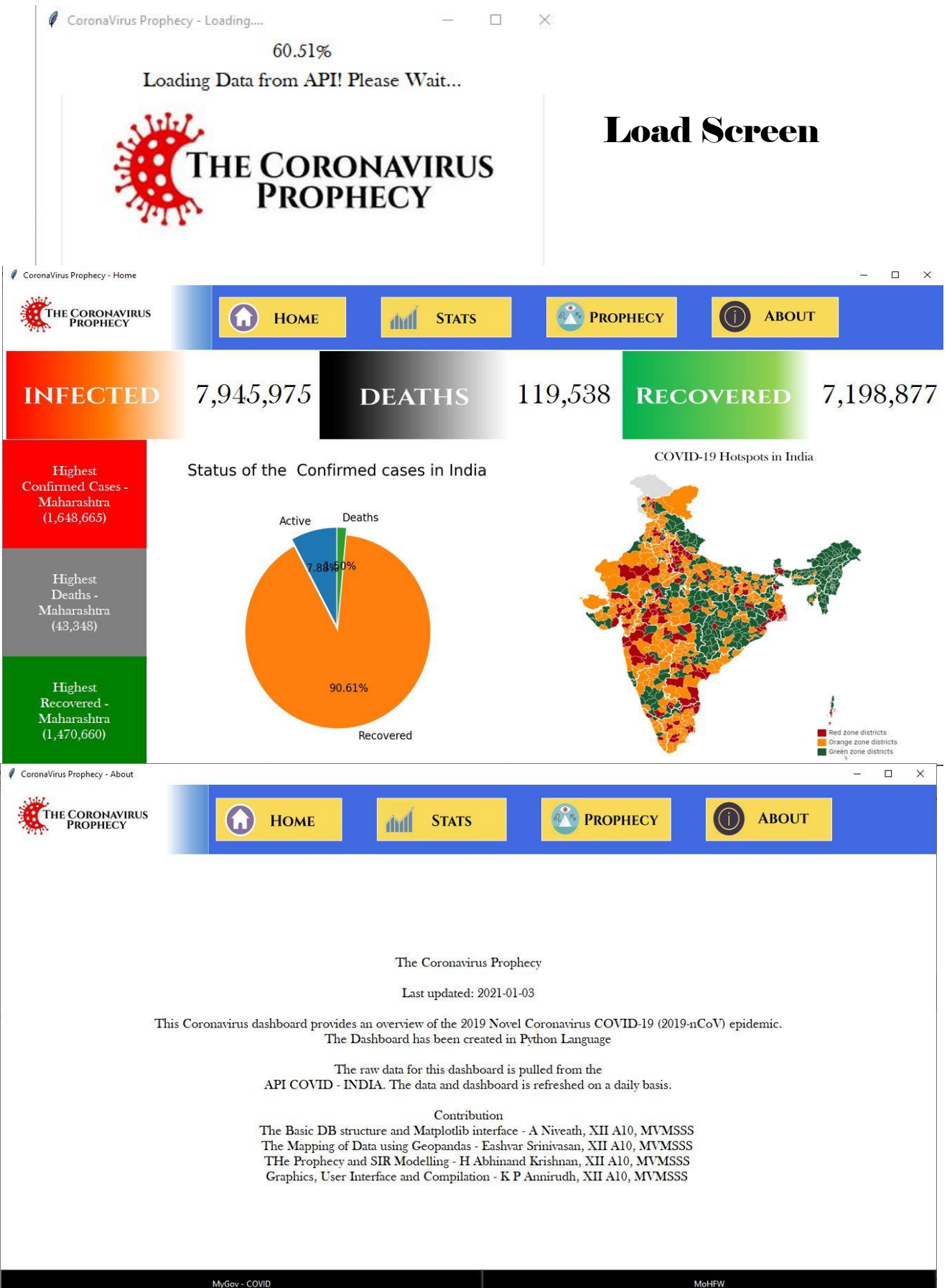
```

```
View.columnconfigure([0,1],weight=1)
    View.grid(row=2, column=0, columnspan=2, sticky="nsew")
    Butto=tk.Button(master=View,text="MyGov -
COVID",fg="white",bg="black",command=callback1,height=2)
    Butto.grid(row=2,rowspan=2,column=0,sticky="nsew")

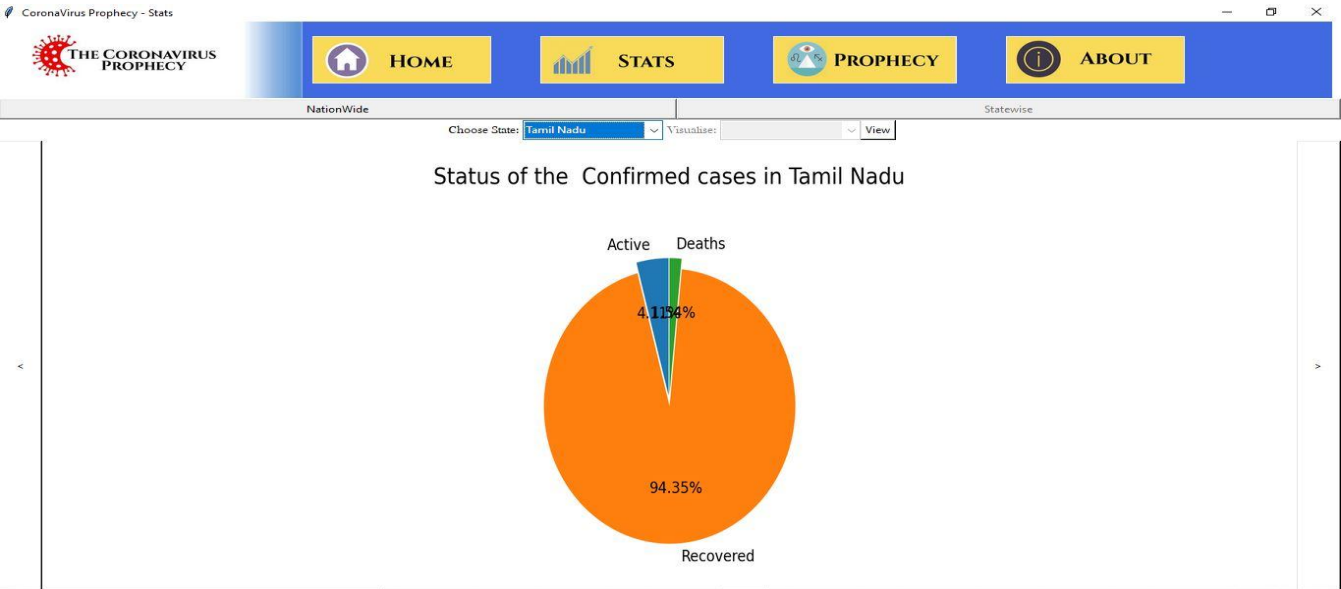
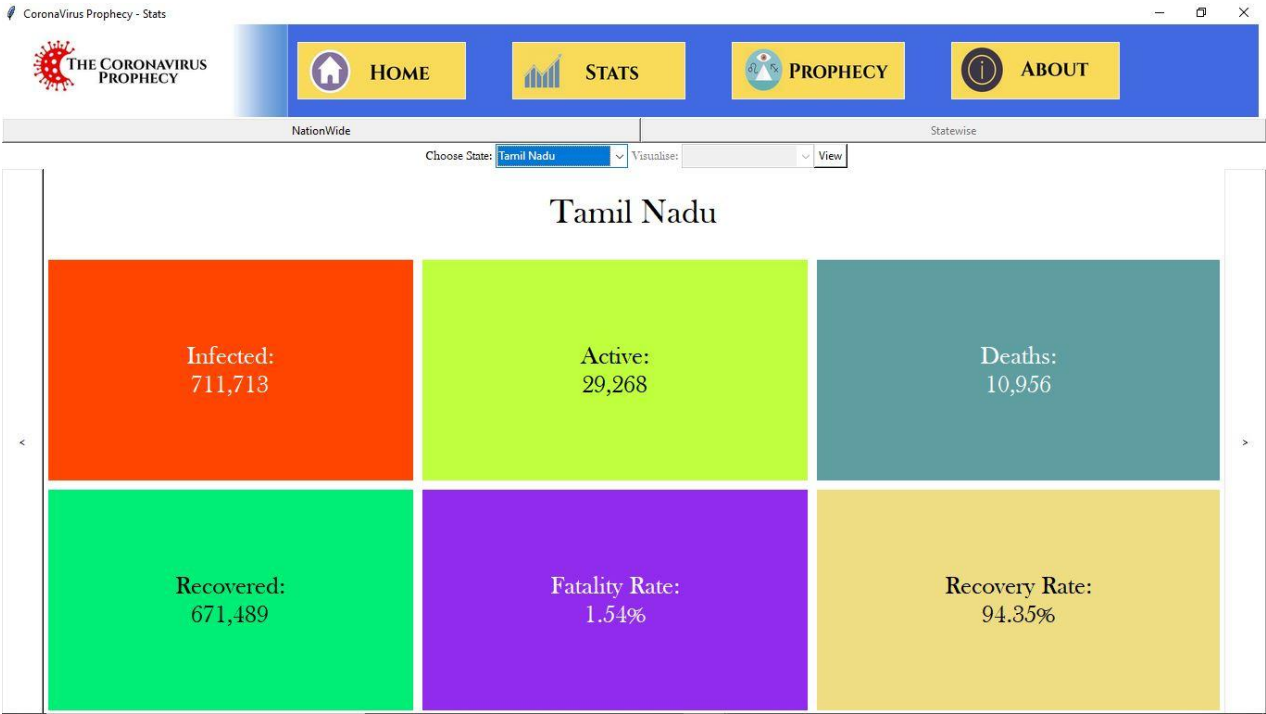
Buttoo=tk.Button(master=View,text="MoHFW",fg="white",bg="black",command=callback2,
height=2)
    Buttoo.grid(row=2,rowspan=2,column=1,sticky="nsew")

load_scr()
```


Output(Home and About)



Output (Stats Screen - State)



NationWide

Choose State: **Tamil Nadu**

Visualise:

View

Statewise

COVID 19 IN Tamil Nadu:Infected



CoronaVirus Prophecy - Stats

NationWide

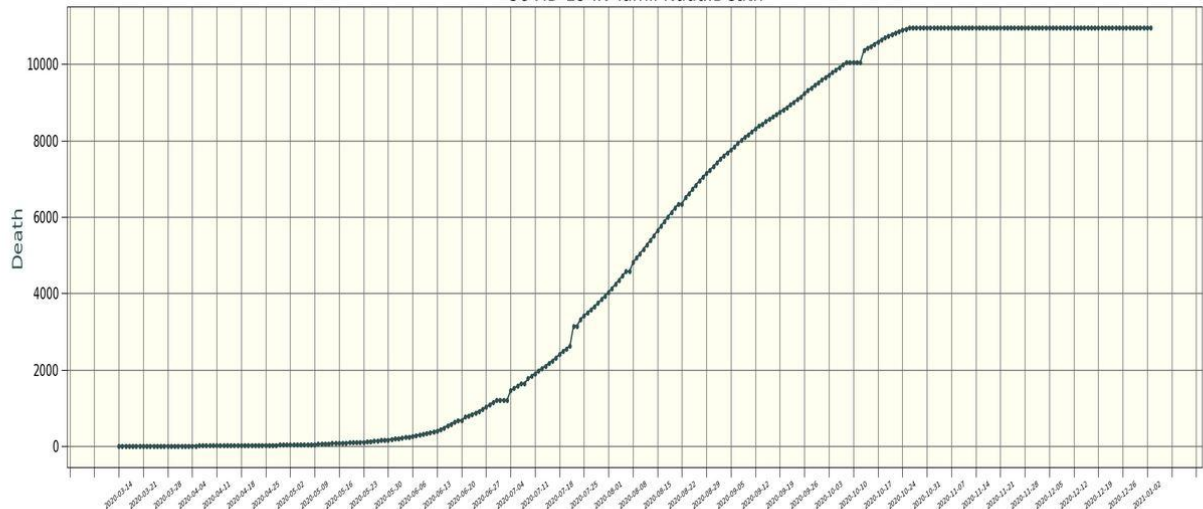
Choose State: **Tamil Nadu**

Visualise:

View

Statewise

COVID 19 IN Tamil Nadu:Death



CoronaVirus Prophecy - Stats

NationWide

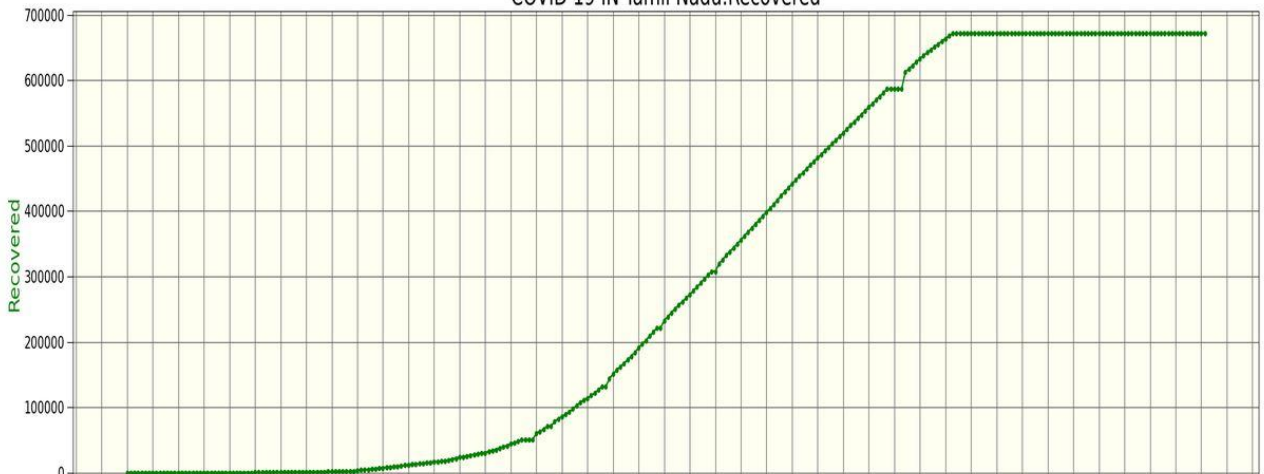
Choose State: **Tamil Nadu**

Visualise:

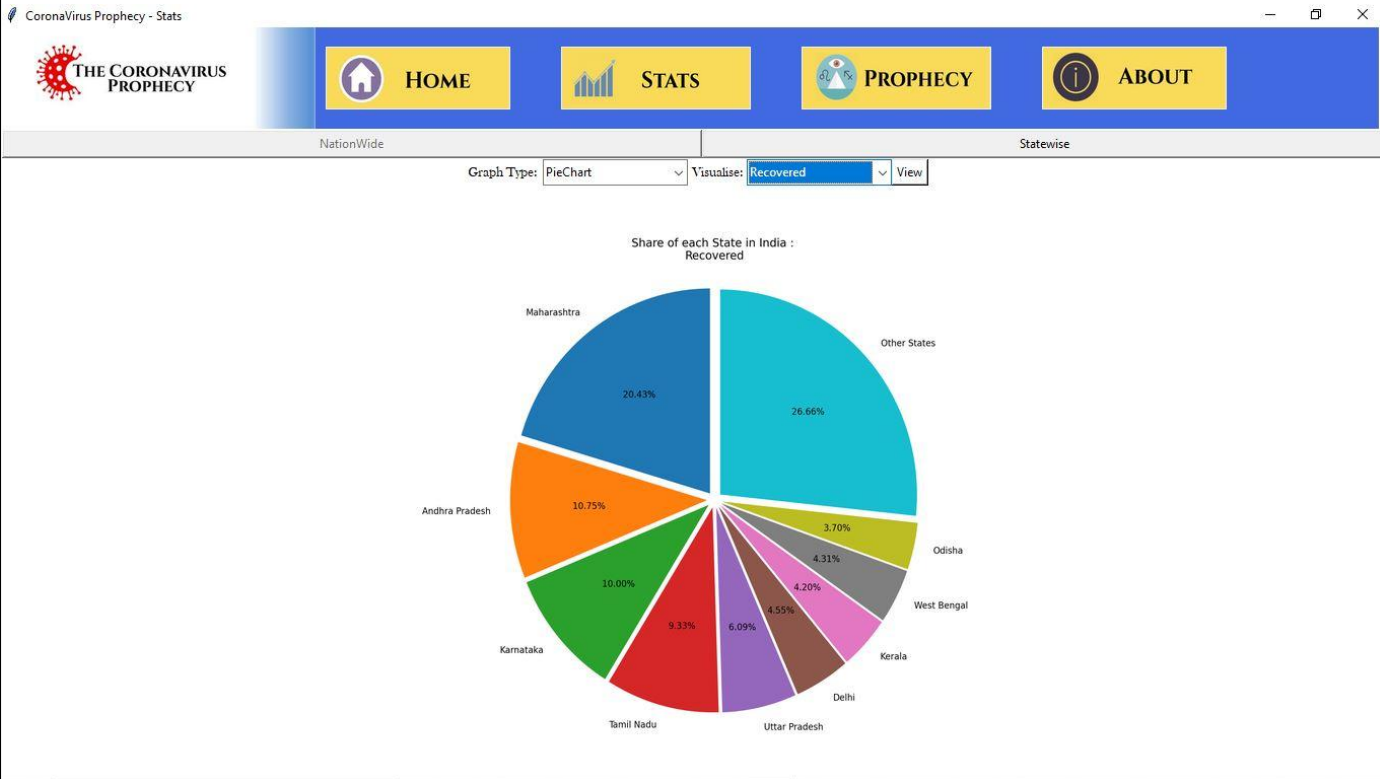
View

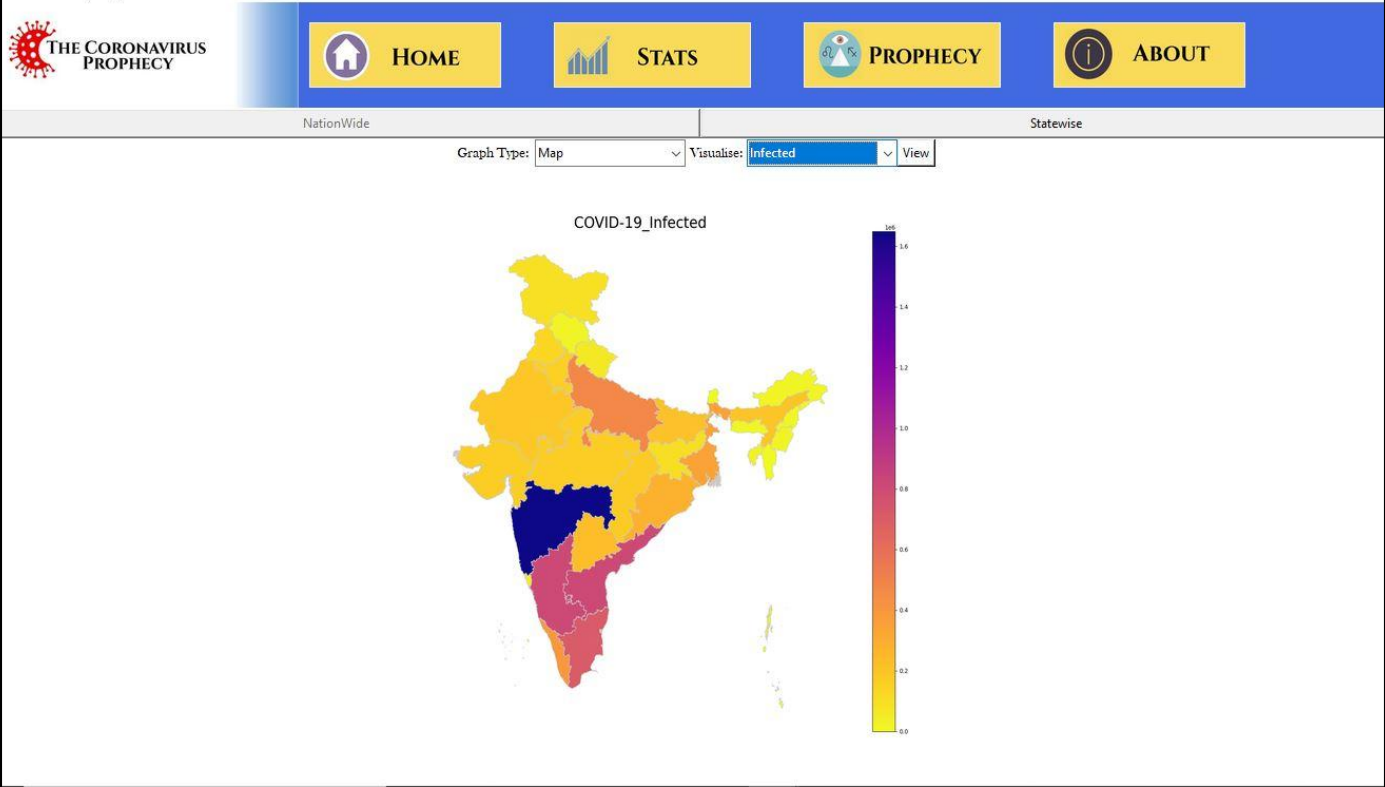
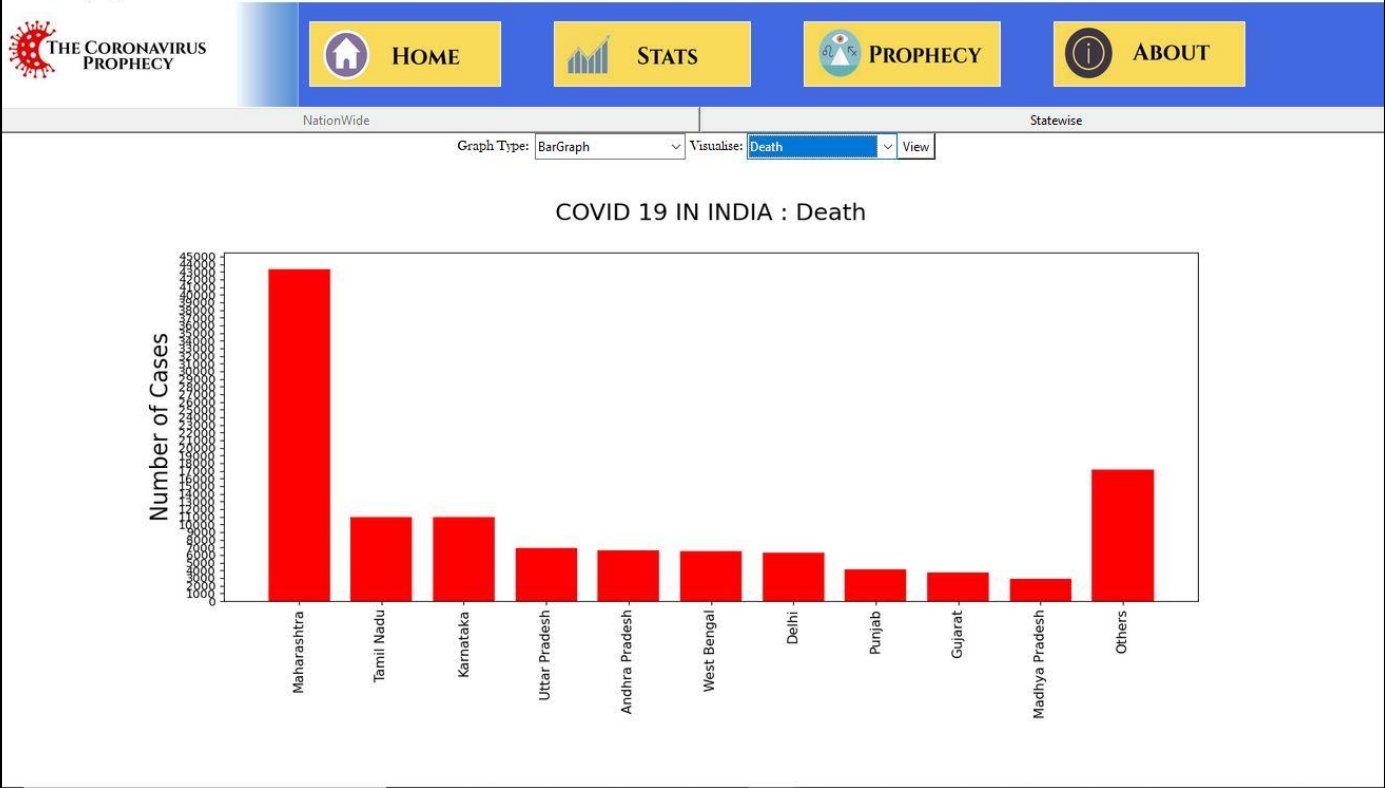
Statewise

COVID 19 IN Tamil Nadu:Recovered




Output (Stats Screen – Nation)








Output (Prophecy)


CoronaVirus Prophecy - The Prophecy

 THE CORONAVIRUS PROPHECY

 HOME

 STATS

 PROPHECY

 ABOUT

The Prophecy

The COVID-19 pandemic, also known as the coronavirus pandemic, is an ongoing global pandemic of coronavirus disease 2019 (COVID-19), caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The pandemic has caused global social and economic disruption.

The SIR model is a simple model of an epidemic of an infectious disease in a large population. It is an epidemiological model that computes the theoretical number of people infected with a contagious illness in a closed population over time. An extended version of the basic SIR model has been built and is used for predicting purpose

No. of Days to Predict: Region :

Predict Now!



Bibliography

1. Exponential growth and epidemics,
3blue1brown,
<https://www.youtube.com/watch?v=Kas0tIxDvrg&feature=youtu.be>
<https://youtu.be/gxAaO2rsdIs>
2. <https://realpython.com/python-gui-tkinter/>
3. <https://realpython.com/python-matplotlib-guide/>
4. <https://apmonitor.com/pdc/index.php/Main/SolveDifferentialEquations>
5. <https://www.sharpsightlabs.com/blog/numpy-linspace/>
6. <https://matplotlib.org/>
7. <https://www.igismap.com/>
8. JSON Module Working:
<https://realpython.com/courses/working-json-data-python/>
9. Data Visualisation:
<https://www.geeksforgeeks.org/data-visualization-different-charts-python/>
10. <https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed>
11. <https://www.kaggle.com/benhamner/python-data-visualizations>
12. COVID 19 Data:
<https://api.rootnet.in/covid19-in/unofficial/covid19india.org/statewise/history>