Covid Simulation and Data Analysis

Background

Our motivation in doing this project was to analyze the Corona pandemic that has been raging on for so long with tools that we as computer science students have learned so far and also to expand our knowledge of python libraries and its various functionalities.

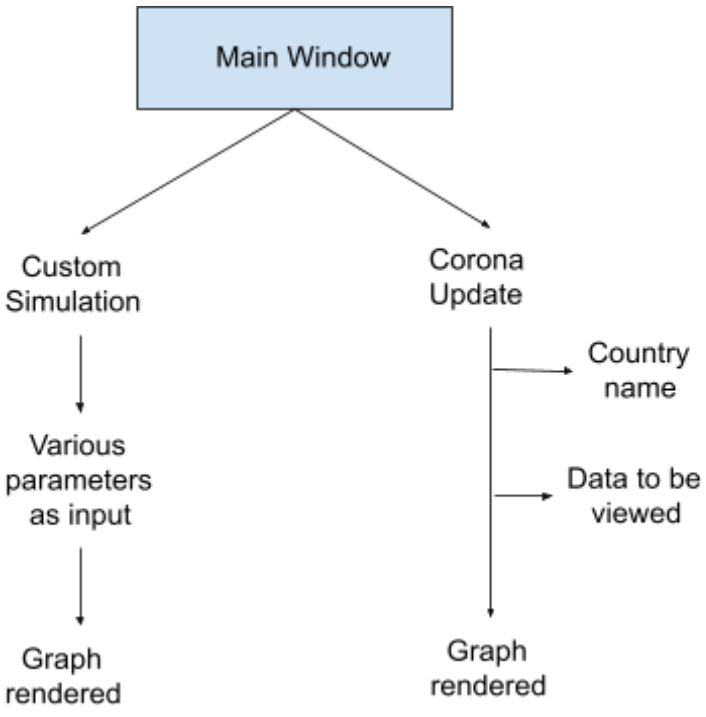
Our project tries to simulate a pandemic by plotting a graph of the number of cases per day till the pandemic is over. The user has 2 options initially. He/she can either choose to witness how a pandemic would look like(ie. the graph of daily cases) subject to certain constraints like masks and lockdowns or can choose to view graphs of any country's total case count or daily case count or deaths count.This is the simulated case data and real time case data respectively.

System Requirements

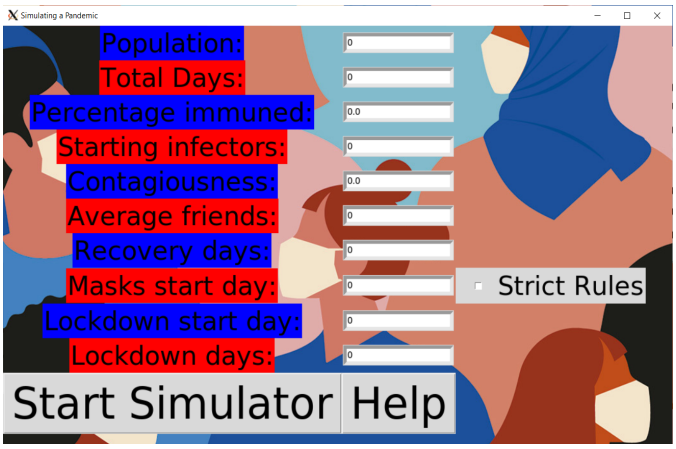
The user must have these libraries installed on his/her system :

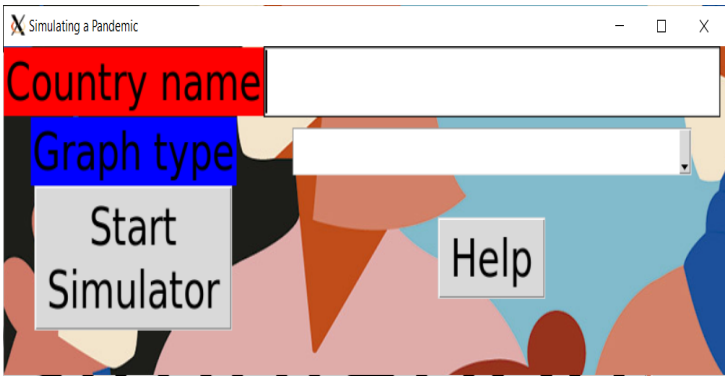
1.Beautiful Soup
2.Tkinter
3.Matplotlib
4.Random
5. Scipy
6. Requests

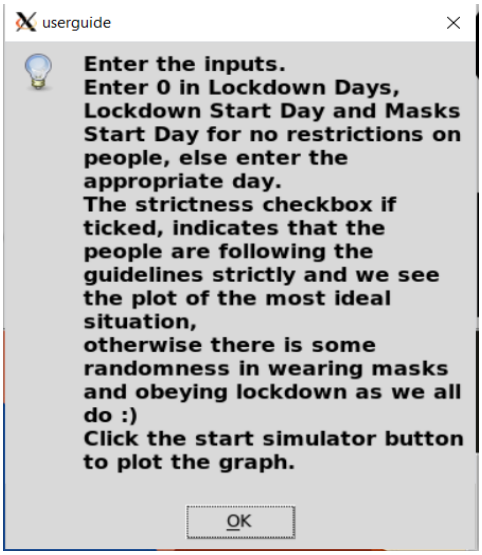
Flowchart

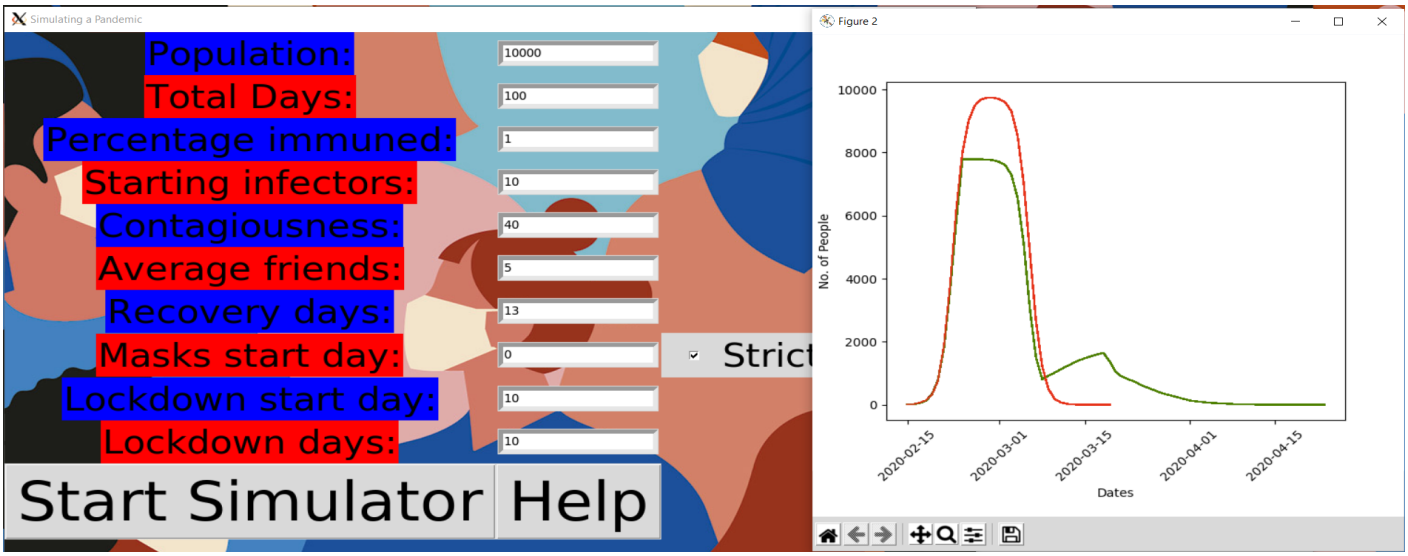


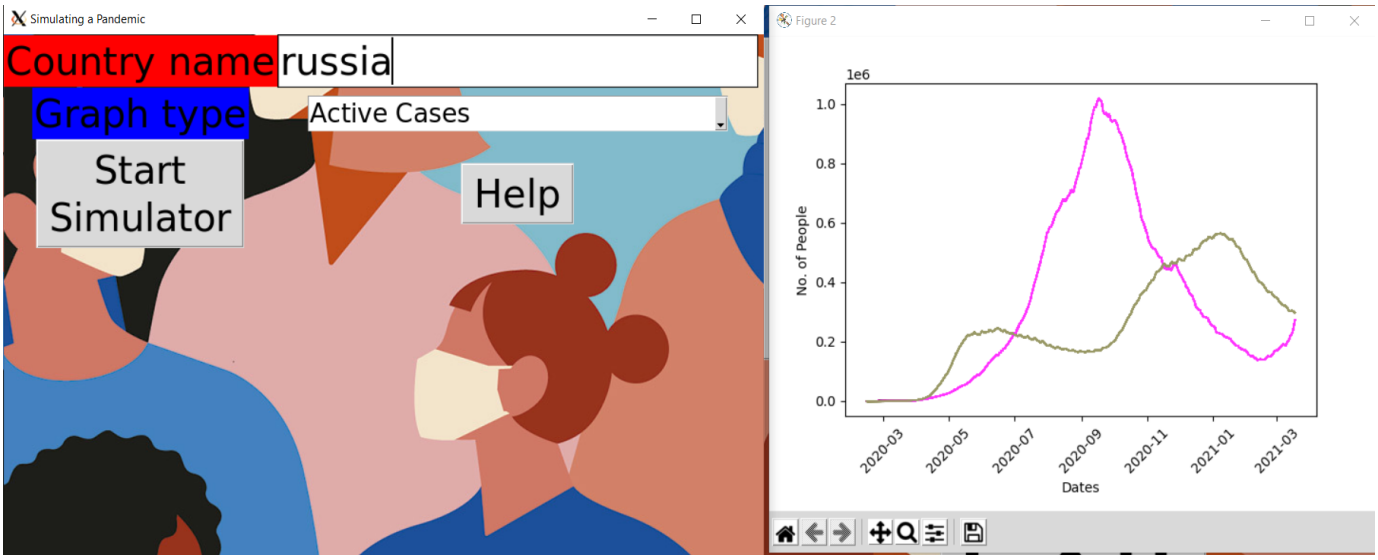
Screenshots











Implementation details

This program has 2 features,

1. Custom simulation- This provides us with a sandbox, in which the user can manipulate various data affecting the spread of corona and see the results in the form of graphs.
2. Corona update- This helps us to display the real data of covid spread of any country.

Our project team consisted of 4 people. Sooraj Sathish dealt with the web scraping of coronavirus data from the internet and the basic organization of this data into relevant lists. Rachit wrote the program which can actually simulate a pandemic when given certain parameters like contagiousness,population,presence or absence of lockdown,etc. Manish has written functions to plot and animate the data that Rachit and Sooraj provided. Vatsal has finally combined all of this and has created a frontend/Graphical User Interface for our project.

Webscraping:

The main library used here was Beautiful Soup. Requests library was also used to open connection and also retrieve the html data/text from the required website (https://www.worldometers.info/coronavirus/). Beautiful Soup was then used to parse this html and obtain the required data by accessing the exact divs or spans. The program also performs some neat string manipulation in order to overcome the fact that Beautiful Soup cannot parse javascript as such(javascript is considered as a bunch of strings). The Covid Data class has many functions which can all be called externally but takes only the country name as a parameter. They can return the last time the website was updated,total cases , total deaths , total recovered , a list of total cases from the day covid started in that country, list of daily new

cases , list of active cases and list of deaths everyday. If the given country name does not exist on their website then an appropriate error message is generated using try except block.

Simulation:

The libraries used for this part are random and scipy. A class named Person has been made which contains the details of a person ‘spawned’ in our world such as immunity,friends,contagiousness,etc. After taking some inputs, according to the population, each person is spawned with a random number of friends. To generate this random number gaussian distribution has been used. Gaussian distribution signifies that a random number of friends will be there but the number will be close to an average value rather than being extremely far away from it. According to the number of starting infectors, randomly some people have been made infected with the disease. Then, for each day of the simulation each person is being iterated over and if he is contagious then according to his contagiousness his friends are being infected. The number of people contagious each day are added to a list . This list contains the number of cases each day.
The interesting part is when lockdown is implemented and/or people start wearing masks. A strictness factor has been introduced. If it is 1, then it implies that the people are following the guidelines strictly and thus no interactions are happening during the lockdown period among friends and everybody is wearing masks. Whereas, if the strictness factor is 0, then people are still interacting with a few of their friends during lockdown and there is a 75% chance each day indicating the usage of masks by a person. The list of cases is then plotted to compare different results.

Plotting:

The Libraries used are Matplotlib and Numpy, though Numpy is not actually used directly but by the Matplotlib library for plotting.The data which we simulate or collect from the web would be too much for any single person to go through as it contains a lot of data and numbers, so we plot the data as it makes it more easy to grasp the situation at hand. Here the data used for plotting are the lists given by web scraping and simulating. This way of data representation makes it easy to understand how the virus is spreading, deaths , etc. To make the plotting look a little bit more appealing to the user, its animated to grow from day-1 to the end of data, this is done but iterating through the data given, but instead of plotting all the points at once we give it a small time interval between each point to be plotted(in this case about 10 milliseconds). Datetime library is also use to calculate the Dates which are plotted on X-axis and No. of People is plotted on Y-axis.

GUI:

The library used here is “Tkinter” which is preinstalled with python, along with that “ttk” and “messagebox” is also imported from tkinter. When the user runs the program, a window opens up in full screen which displays 4 buttons. This window is the “root” window and all the

buttons and window sizes are scaled according to the standard 1080p display.A consistent background image is added to all the windows to increase the appeal of the user interference.

- The button “Custom simulation” opens up a new child window in which using various input boxes are made for the user. Once the user has entered input in all the fields, he can click the “Start simulation” button. The corresponding graph is plotted.
- The button “Corona update” opens up a new child window in which an input box is made for entering the name of the country and a combobox is made for selecting the type of graph that the user wants to see. Once the user has entered input in all the fields, he can click the “Start simulation” button. This button first checks if the country name is valid and displays a popup error message if invalid country name entered. If the country name is valid, the corresponding graph is plotted.
- The credits button when clicked shows the names of contributors for the code in a separate popup window.
- The exit button when clicked terminates the program.

Anytime if the user feels he is stuck, he can click on the help button in the window and access the user guide.

References

https://www.worldometers.info/coronavirus/
https://www.youtube.com/watch?v=XQg KtPS sUIt=1234s
https://www.kaggle.com/tanuprabhu/population-by-country-2020/activity
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html
https://cs.gmu.edu/~dfleck/classes/cs112/spring08/slides/tkinter.pdf

Our github repo:https://github.com/vatsal-dhama/Python_projekt
Advait bhaiya is also a member of this github repo.