Name: Ogundimu Oluwatobi Daniel

Matric No:18/ENG04/058

Department: Electrical/Electronics Engineering

Course: Structural Programming

Title: Algorithm for collecting data on COVID-19 on a web application software.

## Plan

The idea here is to develop software that would help to detect, and display the number of infected persons, rate of infection spread, store the information so that it is available at the request of a client on the designated website. The website would likely be used for epidemiological research. So it may need an idea of geographical location of the populations as well.

Detecting if a person is infected is not practical as it would require the same thing to be done for an entire population. It is more economic to make a mathematical model that can predict the infected people with a significant accuracy (give and take 5%) and determine the rate of spread of the disease by comparing with the numbers of previous observations.

Since the application would store large volumes of data over time, it should use a database to organise the data. The database would get large, such information would be costly if lost, so it must be backed up. The new entries are also stored in the second database. Normally, the requested data is read from the primary database but in the event that the first database is compromised, the data can be read from the second database.

It is unnecessary to create another website for the application, a new webpage within the company's website can be used for making requests and receiving data. Data from the database can then be passed through the existing server, shaving costs of re-topologising.

## Analysis

The web application would require a statistical model, that estimates the number of infected people and the approximate rate of virus spread using data collected from the population over regular intervals. The data calculated may also have additional features as desired in future updates.

There would also need to be a database to organise and store the results calculated from the model. The results are organised according to numbered intervals of calculations, which correspond to date. The results are then accessible to the clients through the server.

The web application would then allow clients to send requests for the information to the server, which has direct access to the database. It would also need security to prevent hacking and 'zombie' attacks.

Thus, the application is going to need statistical and mathematical libraries for building the statistical model. The libraries contain the necessary functions and variables for the statistical operations. The model would also require internet access to gain access to data used in the model for accurate predictions.

The database would be a spreadsheet large in size so we would require mass storage devices. Hard disks are most preferred for this application. We would require multiple hard disks for both the main database and the back-up database.

Though we are using the main server to handle the requests made by clients, we may need to install a more powerful server to ensure that the website does not crash because of too many requests.

There is also the issue of security; the data is delicate and must not be corrupted by malicious attacks from the internet. Therefore, we would install stronger internet firewalls. To minimise zombie attacks, we would add algorithms to observe the behaviour of the computers and identify computers that are likely to be zombies and block them.

## Design

The web application does not need direct access to the model software; It needs access to the database only. Thus the model can be built completely independent of the website. However, the model still needs access to the internet to obtain data for the calculations.

The server would therefore have access to the web application, the model and the database. The bottom up system would look like this:

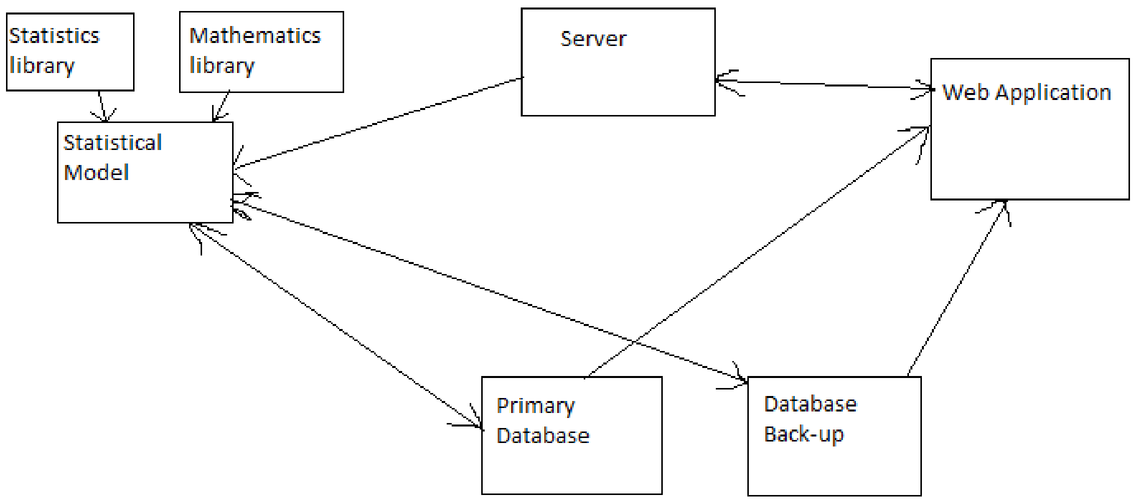

*Illustration 1: Bottom-up design of the web application*

## Implementation

The web application itself would be programmed using Java, which is an efficient program for web development. On the webpage would be a request button, which may be clicked after setting some parameters regarding with data set the client desires. Once clicked, the webpage relays the request to the server, which then returns the request to the web application base. The data is then retrieved from the primary database. If the primary database is corrupted or otherwise unavailable, it retrieves from the back-up database. Once the data is retrieved, it is sent to the server, which sends it back to the client computer to be displayed on the screen as a table. The application does not have access to the model: It is not required, yet it may compromise the security of the model.

The databases are simple comma delimited files (.csv) and do not require much space on their own. The CSV files would contain information on the total number of infected persons, rate of disease spread and other related fields. It is accessible to the application as a read-only file. The model can read from the files and write to them.

The statistical model would be written using python because python has a powerful statistics library and a notable mathematical library, Its operations may be run in parallel to the web application. Over a regular interval, which can be adjusted depending on processing power and needs, the model collects data from the internet through the server. Then the model begins the calculations on the data and returns an information set containing all the necessary fields, the information is then written to both the primary database and the back-up database.

For the web application, the algorithm for the client goes as follows:

```
1. Start
2. Input Parameters
3. Send Request to server
4. Wait For Response from server
5. Server relays request to the web application
6. If Primary Database is not corrupt
       Collect data from primary database
   Else
       Collect data from back-up database
7. Server sends data back to client computer
8. Display Data on client's screen.
9. Stop
```

The Flowchart looks like the following image:

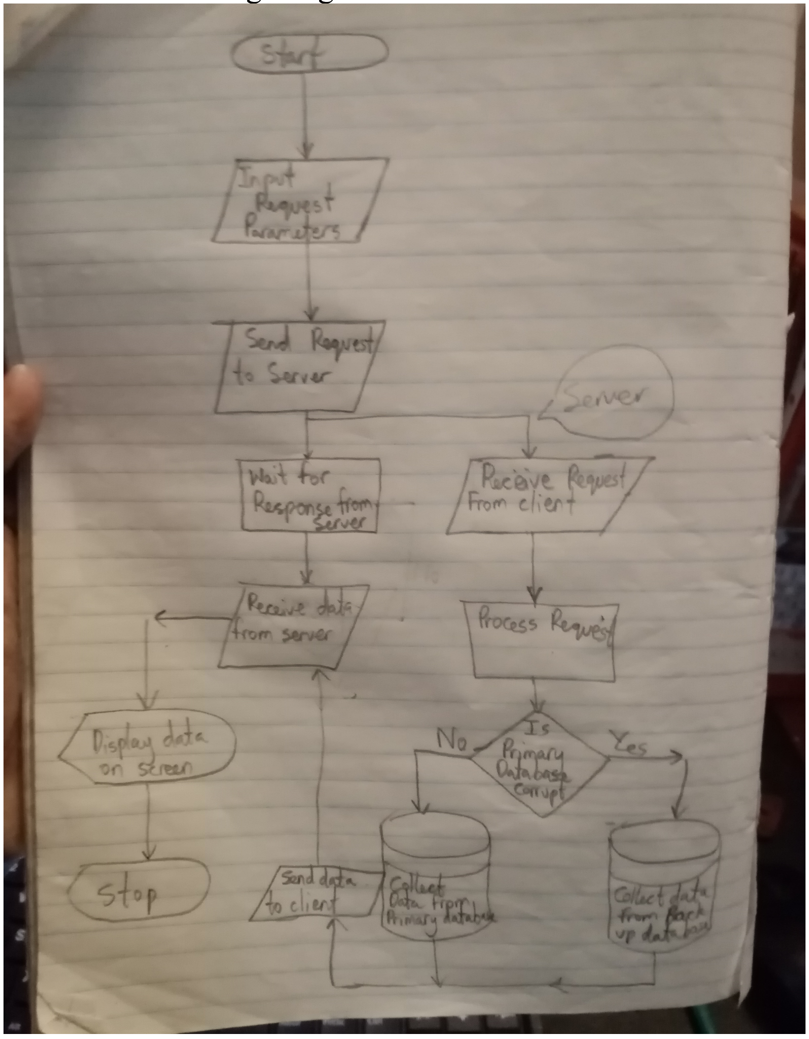
Start
Input Request Parameters
Send Request to Server
Server
Wait for Response from Server
Receive Request From client
Receive data from server
Process Request
Display data on screen
Is Primary Database Corrupt
No
Yes
Stop
Send data to client
Collect Data from Primary database
Collect data from Back up database

*Illustration 2: Flowchart of the web application.*

## Testing

There are various crucial parts of the application that are prone to errors. The testing process would begin with the model: we would have to thoroughly test the model by comparing the results with existing models of the spread of epidemics and then compare it with the observed results in reality. The storage of the results would also be checked to ensure that the data values are stored correctly. The rest would would mostly be error checks to ensure that the application does what it is meant to do, without errors.

## Maintenance

The databases and server would be under stress from frequent use, they would need to frequently be maintained to prevent sudden crashes. The security systems would need to frequently be updated to protect the servers and the model from malware attacks.

Even though the COVID-19 pandemic would soon end, the database will still be important for research into the spread of the disease, perhaps in preparation for another potential future outbreak, to know what measures would be required for limiting its spread. Thus in future tweaks, the model may not be needed and would soon be removed from the system, leaving a web application that sends requests for data from the database.