



**KENYATTA UNIVERSITY**  
**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTING AND INFORMATION  
TECHNOLOGY**

**SCO400 PROJECT SOLUTION DESIGN**

**PROJECT TITLE: COVID-19 CASELOAD AND MORTALITY  
ANALYSIS AND PREDICTION WITH MACHINE LEARNING**

submitted by

by

Name: **IAN MOSES NJARI**

Reg No: **J17/0803/2017**

on

**April 22<sup>nd</sup>, 2021**

**SUPERVISOR: Mr. Joseph Muriuki, M.Sc.**

This solution design document is submitted in partial fulfillment of the requirements for a  
Bachelor of Science (Computer Science) Degree at Kenyatta University

## Contents

Introduction .....	3
1. Solution Design Architecture .....	3
1.1 Data Preprocessing modules .....	3
1.2 Data Stores.....	4
1.3 Web Application.....	4
2. Process Design .....	6
3. Database Design.....	9
5. Interface Design .....	11

# Introduction

This document outlines the design principles of the final product. It comprises of the design process, data formats, data preprocessing and interface design prototypes.

## 1. Solution Design Architecture

The final system will have the following components;

- a) Data pre-processing modules,
- b) Data-stores,
- c) The web application made up of;
  - Global choropleth/heat map,
  - Interactive Plotly Dashboard,
  - Report Generator,
  - Cases Forecasting Engine.

### 1.1 Data Preprocessing modules

After the preview of the data in a [Jupyter Notebook](#), the code will be transferred to a python script that loads the data, processes it(described in the Process Design section of this document) and processes it and stores it the data stores.

A preview of the notebook is shown below;

The screenshot shows a Jupyter Notebook titled "Data Preview" with a last checkpoint of "04/17/2021 (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar (Trusted, Python 3, Logout). The notebook content is divided into two sections:

### 1. Import Dependencies and data

```
In [73]: import pandas as pd
import os
import plotly.express as px
import plotly.graph_objects as go
```

Fetch the data from the **COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University**

```
In [74]: cases= pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series')
In [75]: deaths= pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series')
```

### 2. Preprocess the Cases Data

```
In [76]: cases.shape
Out[76]: (275, 463)
```


















```
In [77]: cases
Out[77]:
```

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/15/21	4/16/21	4/17/21	4/18/21	4/19/21
----------------	----------------	-----	------	---------	---------	---------	---------	---------	---------	-----	---------	---------	---------	---------	---------

## 1.2 Data Stores

The data stores are CSV/XSLX files containing the raw data, preprocessed data and other necessary files.

An example of the data files is shown below;

 .gitignore	Add Plotly Dash app, update notebook and map data	14 days ago
 Analysis.pdf	Add the analysis document	17 days ago
 Concept Paper.pdf	update concept paper	17 days ago
 Data Preview.ipynb	Test plotly figure and dash callback	6 days ago
 LICENSE	Initial commit	26 days ago
 Project Proposal.pdf	Add the proposal document	17 days ago
 README.md	Modify project description	26 days ago
 app.py	App callback finally works!	6 days ago
 casemapdata.xlsx	Modified data	6 days ago
 cases.xlsx	Modified data	6 days ago
 cases_plot.xlsx	Modified data	6 days ago
 deathmapdata.xlsx	Modified data	6 days ago
 deaths.xlsx	Modified data	6 days ago
 map_code.xlsx	Update notebook and map code excel files, add casemapdata datase	15 days ago
 map_code1.xlsx	Modified data	6 days ago
 map_code_copy.xlsx	Update notebook and map code excel files, add casemapdata datase	15 days ago
 pre-processor.py	Remove test plot	6 days ago

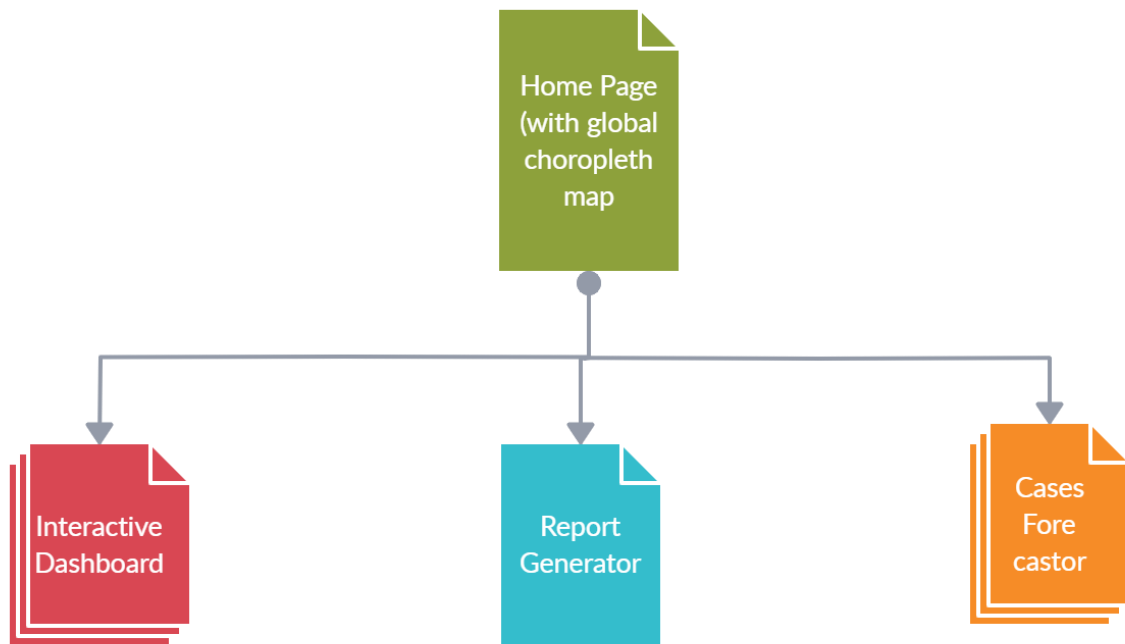
The data will then be fetched during plotting, analysis and Forecasting.

## 1.3 Web Application

The web application comprises the following modules;

- Global choropleth/heat map,
- Interactive Plotly Dashboard,
- Report Generator,
- Cases Forecasting Engine.

Below is a site map of the Web Application:



## 2. Process Design

The process begins through activation of an automatic script that fetches the data from the JHU CSSE repository. The code is as follows;

For cases data,

```
cases= pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv')
```

For deaths data,

```
deaths= pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv')
```

The cases data is in this format,

	Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	...	4/15/21	4/16/21	4/17/21	4/18/21	4/19/21	4/20/21	4/21/21
0	Afghanistan	0	0	0	0	0	0	0	0	0	...	57534	57612	57721	57793	57898	58037	58214
1	Albania	0	0	0	0	0	0	0	0	0	...	129128	129307	129456	129594	129694	129842	129980
2	Algeria	0	0	0	0	0	0	0	0	0	...	119142	119323	119486	119642	119805	119992	120174
3	Andorra	0	0	0	0	0	0	0	0	0	...	12641	12712	12771	12805	12805	12874	12917
4	Angola	0	0	0	0	0	0	0	0	0	...	23951	24122	24300	24389	24518	24661	24883
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
187	Vietnam	0	2	2	2	2	2	2	2	2	...	2758	2772	2781	2785	2791	2800	2812
188	West Bank and Gaza	0	0	0	0	0	0	0	0	0	...	276407	278135	279753	280741	282270	284280	286028
189	Yemen	0	0	0	0	0	0	0	0	0	...	5657	5715	5770	5812	5858	5918	5960
190	Zambia	0	0	0	0	0	0	0	0	0	...	90532	90750	90844	90918	90942	91042	91119
191	Zimbabwe	0	0	0	0	0	0	0	0	0	...	37422	37534	37699	37751	37859	37875	37980

*Drop all columns from the first date to the second last.*

Since we need only the total cases on the last day for our map.

```
cols=df1[df1.columns[1:-1]]  
df2=df1.drop(cols,axis=1)
```

The data frame is now in this format;

	Country/Region	4/24/21
0	Afghanistan	58730
1	Albania	130409
2	Algeria	120736
3	Andorra	13024
4	Angola	25492
...	...	...
187	Vietnam	2833
188	West Bank and Gaza	290259
189	Yemen	6105
190	Zambia	91317
191	Zimbabwe	38064

This data frame is unioned with country code data from plotly documentation to help with plotting of the choropleth map and output into a data store;

#### Perform inner join on the map code and cases(df2)

This step eliminates countries (Holy See, North Macedonia and Micronesia) and cruise ships (Diamond Princess and MS Zaandam) in the cases dataset.

```
df5=pd.merge(df2,df4,how='inner',left_on=['Country/Region'],right_on=['COUNTRY'])
df5=df5.drop('COUNTRY',axis=1)
df5
```

	Country/Region	4/24/21	CODE
0	Afghanistan	58730	AFG
1	Albania	130409	ALB
2	Algeria	120736	DZA
3	Andorra	13024	AND
4	Angola	25492	AGO
...	...	...	...
182	Vietnam	2833	VNM
183	West Bank and Gaza	290259	WBG
184	Yemen	6105	YEM
185	Zambia	91317	ZMB
186	Zimbabwe	38064	ZWE

187 rows x 3 columns

The process is repeated for the deaths data.

To obtain data for cases line plots, the initial data frame is transposed, new headers allocated, and data types changed to achieve the following data;

```
plot_df=df2.rename_axis(None, axis=1)
plot_df
```

	Date	Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	...	United Kingdom	Uruguay	Uzbekistan	Vanuatu	Venezuela
0	2020-01-22	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	2020-01-23	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	2020-01-24	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	2020-01-25	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	2020-01-26	0	0	0	0	0	0	0	0	4	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
454	2021-04-20	58037	129842	119992	12874	24661	1217	2743620	209485	29576	...	4408644	169327	87225	3	185736
455	2021-04-21	58214	129980	120174	12917	24883	1217	2769552	210518	29594	...	4411068	172601	87551	4	186745
456	2021-04-22	58312	130114	120363	12942	25051	1217	2796768	211399	29638	...	4413834	175891	87935	4	188063
457	2021-04-23	58542	130270	120562	13007	25279	1222	2824652	212114	29653	...	4416588	179537	88280	4	189381

The data is used to plot as well as stored in a data store as well.

The data will be used to predict the number of cases with a TimeSeries Forecasting model later.



### 3. Database Design

The project does not implement any traditional databases. Instead, excel and coma separated values (CSV) files are used to implement the data storage mechanism.

The data schema in the data files are as follows;

#### **Cases**

```
Cases (  
  country/region char not null,  
  22/1/2020 int,  
  .....  
  01/5/2021 int  
)
```

#### **CaseMapdata**

```
Casemapdata (  
Country/region char not null,  
Total cases int not null,  
CODE char not null  
)
```

#### **Deaths**

```
Cases (  
  country/region char not null,  
  22/1/2020 int,  
  .....  
  01/5/2021 int  
)
```

#### **DeathsMapdata**

```
Casemapdata (  
Country/region char not null,  
Total cases int not null,
```

```
CODE char not null
```

```
)
```

### **Cases\_plot**

```
Cases_plot(
```

```
Date date-time not null,
```

```
Afghanistan int not null,
```

```
Albania int not null,
```

```
.....
```

```
Zambia int not null,
```

```
Zimbabwe int not null
```

```
)
```

### **Daeths\_plot**

```
Deaths_plot(
```

```
Date date-time not null,
```

```
Afghanistan int not null,
```

```
Albania int not null,
```

```
.....
```

```
Zambia int not null,
```

```
Zimbabwe int not null
```

```
)
```

## 5. Interface Design