Report of the Project

### **Introduction:**

This project is to summarize the data by giving it a data visualization for the number of occurrences in any factor based on the given ID.

The assumptions I have made in this project since this project supports GUI. With User input we could have a histogram of any UUID and generate the result for any factor.

### **Requirements checklist:**

The project is coded with Python programming language. Checklist for 1.

The program takes UUID as User input and takes a task from the user to generate the histogram.

There are various options for the user to choose what they want.

Histogram for the number of countries viewed or Histogram for the number of continents viewed, Checklist for 2

Histogram for the number of browsers viewed. From the user it can present the result with either the version of the browser or the name of the browser. Checklist for 3

During the launch of the program, the program asks for the UUID from the User input and the task for what histogram they want.

Get top 10 readers. Checklist for 4

Get top 10 documents. Checklist for 5d.

Command line usage for all the above operations. Checklist for 6

### **Design Consideration:**

After the launch of the program, the user will be asked to write a valid UUID and choose among the options what histogram they need. In case if the user writes an invalid UUID or leaves the box empty and clicks on the option. It will throw a error message "Invaid UUID"

<https://user-images.githubusercontent.com/81755254/205446605-0c61190d-b414-41fe-8e0e-f149fdc6b2b2.png>

### **User Guide:**

During the launch of the program. The program requires an argument for the executable file and the document for json file.

We edit the Makefile to send the file name:

https://user-images.githubusercontent.com/81755254/205446506-1f5c689a-5367-4957-9187-32b8f44f28d0.png

After the launch it finishes the reading and sets up the configuration details for the UUID. It opens the window and asks the user to write a valid UUID and provides multiple options for the user to generate the histogram for which aspect.

https://user-images.githubusercontent.com/81755254/205446563-7e4e1851-a94e-4568-9565-f41dbfd17831.png

If the user writes an invalid UUID means if the textbox is empty or the given UUID does not exist in the json file.

https://user-images.githubusercontent.com/81755254/205446605-0c61190d-b414-41fe-8e0e-f149fdc6b2b2.png

If the user writes a valid UUID and clicks on Get Histogram for views in Countries.

https://user-images.githubusercontent.com/81755254/205446694-3f9d107f-c530-4a7a-933c-b3118c5d06a6.png

**Developer Guide**:

1. We first need to parse the arguments for the launch of the program since we need to send the document. We import argparse and call for the function argpase.ArgumentParser(). Then we can add an argument for the file reading.
2. First it needs libraries to support our program. It imports string, pandas, argparse, json, tkinter, matplotlib, yplot, numpy, pycountry\_convert libraries.
3. The function read\_data() takes the filename. Opens the file takes the data as a string and closes the file descriptor to avoid file descriptor leaks.
4. the function add\_my\_dict() takes the data and separates every row with a new line and returns it as a list. Since the data is represented as a dictionary.
5. The function get\_visitor\_uuids() takes the li as an argument and inserts all the occurences of the uuid.
6. The function get\_visitor\_countries() takes the li as an argument and inserts all the occurences of the countriy\_codes.
7. The function get\_visitor\_brows() takes the li as an argument and inserts all the occurences of the browsers\_versions.
8. The function filt\_brows() takes the browsers as an argument and filters the browsers , eg "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.117 Safari/537.36" will become "Mozilla/5.0".
9. The function sp\_browsers() takes the filt\_browsers as an argument and trims it to get the common browsers , eg "Mozilla/5.0" will become "Mozilla".
10. The function get\_readers() takes the li as an argument and inserts all the occurences of the readers in env\_type.
11. The function get\_uniue\_countries() takes the li as an argument, checks with the countries and inserts all the unique countries.
12. The program starts from the main and creates the tkinter window and waits for the user to write a valid UUID.
13. Button\_1 will hold the value for the function get\_hist\_countries().
14. get\_hist\_countries will have a global variable uuid, e variable is for the tkinter window, e.get() is the program asking from the user to write a valid UUID. If the user writes or leaves the text box empty and clicks on Get the Histogram for countries. First it checks if the uuid belongs or contains in the uuid\_list (list variable). If not, the function throws an error Message.
15. If it contains, uuid\_dict will have the dictionary type data where keys will be the unique countries and values will be the occurences for each country.
16. The elements of the dictionary will be like this: "uuid: uuid\_value", "ES: 2", "EN: 5", .... So we need to remove the first element. So for that we use the pop function to get rid of the first element.
17. For the x and y values, we split the dictionary with keys and values. So we have list\_dict\_identifiers which holds all the keys and list\_dict\_vals which holds the values for keys.
18. Since we need to show the countries in histogram we use the funtion pc.country\_alpha2\_to\_country\_name() to get the name of the country. Since "ZZ" is not a valid country code. We give an unknown country name.
19. To plot the histogram for the countries, plt\_hist() will take x, y, title, x\_lbael, y\_label. In order to show the x labels perfectly we use plt.xticks() and give a rotation angle of 55 degrees. And to set the y limit we gave the max number in y + 1.
20. Button\_2 will hold the value for the function get\_hist\_continents()
21. get\_hist\_continents will have a global variable uuid, e variable is for the tkinter window, e.get() is the program asking from the user to write a valid UUID. If the user writes or leaves the text box empty and clicks on Get the Histogram for countries. First it checks if the uuid belongs or contains in the uuid\_list (list variable). If not, the function throws an error Message.
22. If it contains, uuid\_dict will have the dictionary type data where keys will be the unique countries and values will be the occurences for each country.
23. In create\_dict\_uuid\_conts() it takes uuid, countries, and li, it converts the country code to the continent\_code by pc.country\_alpha2\_to\_continent\_code(country) and then it converts the continent code to continent name by pc.convert\_continent\_code\_to\_continent\_name.
24. Now we have the dictionary with keys x and values as y. and we plot the histogram for it. We call plt\_hist()
25. Button\_3 will hold the value for the function get\_hist\_browser()
26. Since in the start of the program we had the browsers variable which takes every occurences of the browsers.
27. We call for the function create\_dict\_uuid\_brows() which will take uuid, browsers and li.
28. This will return the dictionary for all the browsers along with their frequencies.
29. To plot histogram we call plt\_hist()
30. Button\_4 will hold the value for the function get\_hist\_sbrowser()
31. We call for the function create\_dict\_uuid\_sbrows() which will take uuid, browsers and li.
32. This will return the dictionary for all the filtered browsers along with their frequencies.
33. To plot histogram we call plt\_hist()
34. Button\_5 will hold the value for the function top\_readers()
35. We don't need UUID for this function. However, we need to count the number of readers in the document. And represent the histogram for the top 10 readers in the document with the frequencies.
36. get\_readers() takes li as a parameter. First the program will take all the reader's uuid and store them in readers\_list.
37. Then we get the unique UUID from the readers\_list.
38. freq\_occureneces will store the value for the number of occurences with every UUID from the readers\_list.
39. Now we have unique\_readers\_list which has the unique UUID of every reader and freq\_occurences which has the values for the number of occurences based on UUID. We will create a dictionary where the key is the UUID and the value is the number of frequencies.
40. Now in order to sort the dictionary based on the values of the dictionary, dict(sorted(read\_dict.items(), key=lambda item: item[1], reverse=True)) will return the dictionary in descending order and return it
41. In the function top\_readers(), readers is the dictionary for the unique UUID. We initiate two lists where reader\_keys will hold the keys of the readers and reader\_vals will hold the values of the readers. raw\_keys will hold the value of top 10 reader\_keys and raw\_vals will hold the value of top 10 reader\_vals
42. plt\_hist() will take raw\_keys and raw\_vals as an argument to draw a histogram.
43. get\_readers takes li as an argument. Fetches the number of reader’s data with visitor UUID. and create a dictionary where key is visitor\_uuid and values are doc\_uuid.
44. get\_unique\_docs() takes li and document list as an argument and returns all the unique documents from the document list.
45. top\_10\_doc() returns the list of top 10 docs sorted with number of readers corresponding to that doc.
46. also\_like() asks the user for visitor\_uuid and doc\_uuid and creates a list of those visitor\_uuid who have already read the document of that input document\_uuid.
47. With every visitor\_uuid we fetch every document which visitor\_uuid have already read.
48. Then we create a new file graph.dot. And write a syntax for every reader pointing to the documents which they have read.
49. The graph should be of type digraph {}. Inside the digraph we label every visitor\_uuid pointing to the doc\_uuid.
50. For Command line usage: we recreate all the functions and pass the necessary parameters from the arguments.
51. For 2a, 2b, we send doc\_uuid.
52. For 3a, 3b, we send visitor\_uuid.
53. For 4 we don’t need to send any input.
54. For 5d and 6 we send both doc\_uuid and visitor\_uuid.
55. For 7. We are running the main loop for the GUI.

**Testing**:

1. Program runs slower if the file is too big.

2. Doesn’t parse uuid from the executable file.

**Reflections on programming language and implementation:**

The project which includes the Data Analytics is quite opposite with the manwork and the codework. In Process, we have a keen eye for every information in the data. However, we take coding, create an algorithm to brief the summary and gave a quick result. This would help in every circumstances. In time consumption, processing data, giving a data visualization. In Python its more easy. Since in this project variables are container types. We can add or remove any element. Design/ Organize them in a way we want. Filter/trim the data in an effective and efficient manner.

**What did I learn from CW1?**

In the field of Data Analytics. We create a statistical model in order to observe, process, design the data to lead it to the final result. For example.  
Netflix has a large number of viewers. For that we calculate the average number of viewers, average number of hours for views. More entertainment tv serials or movies or shows into the recommended list for the viewers. To arrange them in a statistical model. We need to record these data transactions and implement them for our statistical model. However, in coding, There are a bunch of operations which can record data in a second. If we have a website which has the recording for the number of viewers in Netflix. We can call a function which can extract the data. Translate them in any way we want.

Coding makes life easier in any fields:  
In Game learning, stock market analysis, building an algorithm for organizing the data.

The sky's the limit in coding.

**Conclusion:**

With this project we can track the number of views on any aspect for any activities. This project gave me the skills for GUI in Python Programming. Moreover, this gave me the ability to organize the ata in such a manner that we can plot the histogram with it.