**FCP project report**

**Zihe(Ziz) Zhu – colour code**

My assigned parts were to represent new infected cases using colour. There are 5 modules that I used, ***‘xlrd’***, ‘***numpy***’, ‘***os***’, ‘***PLT***-***Image’*** and ‘***tkinter***’ which some are shared being used with other members in the group.

The packs that we used are ***‘xlrd’*** to open and read data from excel documents at the start. It’s used and shared for all our group members and acts as a foundation to the project; *‘****numpy****’* to process data from excel by set into numpy array and dictionary; ‘***PLT***-***Image***’ and *‘****os***’ to open image files and ‘***tkinter***’ to add legends on the side of simulated plane.

For my part, I worked closely with Yue to make sure that the simulated plane can change size and colour according to data. I used four images of green, yellow, orange, and red to colour code and represent a plane’s level of safety to fly due to number of new cases at the day.

To fulfil the colour change functionality, I did not focus specifically on one module but used multiple traditional functions such as ‘select\_image’ and modified Yue’s ‘n\_size’ to make sure the code not only change size but also change colour as well as simplify the code. I noticed that Yue’s code saved images of plane to different sizes in a folder according to dictionary, ‘dict1’ I then created another ‘dict2’ (date and new cases data from excel). As passenger flow and new cases shares the same date, I assigned it to Yue’s code of changing image sizes, which it chose the correct coloured image for each size modification before saved in folder. I also implemented ‘if not’ logic to avoid repeating error by not creating folder as there already was one created from previously run. I added a legend for simulation in the end to specify the colour code with respective to number of new cases.

**Yue Wang – change the image size according the passenger load factor**

My codes can be divided into 2 parts in general. The first part is how to generate different sizes of image and save them into a folder. I used **os**.mkdir to create a folder, name it as ‘new\_plane’. Then created two lists by using **np**.array(), “date1” contains the date, “psrate” contains the passenger load factor, these data come from the excel form we made before we started programming, and remove the first element is the lists, because they are descriptions, then use dict(zip( , )) to form “dict1” which keys are the load factors and values are the corresponding dates. Open an image of the plane and create a function “n\_size” to resize the image. In the for loop below, for example, the passenger load factor is between 50 and 53, use the function n\_size(100,100) to resize the image with width and length 100\*100, do the same when the load factor is different, and finally save and name these new images with corresponding dates, this is achieved by using dict1[each].png which “each” is the passenger load factor.

The second part I made is to create a GUI, the first step is to set a window by using window = **tk.TK()**, set a title(window.title); set the size of the window(window.geometry); set the position of the labels(window.columnconfigure, window.rowconfigure). The second step is to label things on the window, use **tk**.label(window, . . . ) to label the picture of the plane, the text of the date and the notes on it. The third step is to set a listbox by using **tk**.listbox(window, . . .), add the date1 list into the listbox, because the first element and the final element contains “[ ]”, so delete them(listbox.delete()) and add 1.1, 8.3 to these two position(listbox.insert()). Step four is to form a button by using **tk**.button(window, . . .), the last step which is also the most important step, is to create a function, use listbox.get() to assign the date chosen by users to the “slected\_value”, and let the “selected\_value” be the text labelled in the window and also be the filename of the image been labelled. When users choose a date in the listbox and press the button, the function is called, the image and the text will change to the correct one corresponding to the user’s choice, then use. grid(column=?,row=?) to arrange their order.

**Jelly Jinzhe Liu- GUI and web report.**

Inspired by the CMD, I decided to use HTML as our project report. You can compare the pictures, download the data table, and see our division of labor and what we have done.

The language has high flexibility and high fault tolerance. I introduced the explanation of our web-report by <h1>, which can make it bigger and clear. I made a hyperlink for checking the log history. <a href="#ct1">. In <style>part, I create a button by using .drop. and I achieved /\* Target drop-down content \*/, /\* Drop-down content (hidden by default) \*/, /\* Drop-down content (hidden by default) \*/, /\* Link to drop-down menu \*/, /\* Move the mouse up to modify the link color of the drop-down menu \*/, /\* Modify the background color of the drop-down button when the drop-down content is displayed \*/. By using dropbtn for Menu, I made 10 hyperlink for each part, and create each specific HTML for each month or download the excel. For the rest is log history, I made an alert to show everyone’s duty and information. The biggest reason that people can use this software easily is GUI, so I want to combine all programs into one file, which is “open this.py” OS library provides general and basic operating system interaction function. So need to import os. Tkinker library is used to make GUI. And named root is same as Tk(). I defined four values as four buttons, which are project\_chart(), data\_pic(), animated(), all(), html(). I try to keep each function individual. So that users can focus on different demand. os.system can run each py file. I insert the text for introduction by text.insert. The button is also defined for text, anchor, background color, frontground color, button size and border size. And using text.pack() & root.mainloop() to finished the text and program.

**Lau Lok Man Mario – graph plots and animated graph**

My codes are separated into 2 different parts. The first part is to create a simple line graph using “***Matplotlib***”. But before that I used ***‘xlrd’*** to read the data from the data sheet we created by doing research on the actual data for different aspects during the pandemic for the company ”Air China”, such as the amount of income and outcome, the passenger load factor, number of flights and the total cases of COVID-19 in the main cities of China. Then I used ‘***numpy***’ for later process by setting them into numpy array. All the data were process with a simple calculation so that all of them can be put into a single line graph. Then I set up the graph using the module “***Matplotlib***”. Each line is labeled to show which represent which data set from the excel datasheet. This graph is then shown using ***“plt.show()”***.

The second part is to create an animated line graph using the data of total number of cases and compare it with the simulated number of cases. At the start, I used “***pandas***” to read the data from the excel datasheet. Then I coded a function called ***”argument“*** to add more point in between each data value from the data sheets so that the line plotted will be a little bit smoother and have less sharp edges. Another function coded which is “***predictedcases***”is used for simulation the number of cases during the pandemic. In this function, I used the module “***random***” to get a random integer for cases increased each week. After these two functions being coded, data are set into numpy arrays and process through the function “***argument***”, then put into a dataframe which can be used for plotting the graph later in the program. Next step is to set up the graph and the writer for it to be animated and plotted on a line graph. I chose “***ffmpeg***” to be the writer and then set up the graph using the module “***Matplotlib***”. After that, I coded another function called “***animate***” to be called later in the “***matplotlib.animation.FuncAnimation***”. This function will be looped X times depending on how many frames is set in the “***matplotlib.animation.FuncAnimation***” plotting one point on the data each time. The graph is then shown by using ***“plt.show()”***.