FOSSEE SCREENING TASK - 2

CSV FILE EDITOR AND PLOTTER
APPLICATION USING PyQt5 - by Jishu Dohare

GitHub Repository Link - **https://github.com/JishuDohare/fsf_2019_screening_task2**

Introduction

Made the desktop Application which was specified in the FOSSEE Screening Task - 2, with all the necessary features such as Loading the .csv file and performing operations like Display Data, Edit Data, Add new data and plotting of graph for different columns of the file.

This software application can be used for performing operations on a file and also for plotting the data of the file, with changes here and there the editor can be also made compatible for other file extensions like .xls using their respective modules which are available as a library in Python.

Libraries Used

PyQt5 - for application development

Matplotlib - to plot the graphs

Scipy, numpy - for Scatter Plot with smooth Lines

Approach

The application consists of two main parts (i.e. two classes):

1. **App class** - with functions like
    a. **initUI()** - using this function the UI features are added with their respective functionalities, e.g. - on click of Save button the saveData() function is called to save the data, etc.

    b. **loadCsv()** - when a user clicks Load button, this function is called and the File Dialog gets opened so the user can easily select the file to open, the user is only allowed to open a .csv file.

    c. **createTable()** - after the file is selected from the loadCsv() function the flow of application heads to this function so that we can initialize a QTableWidget() and show the data into our application using it. We set flags for each data item of the table so that the user cannot edit the data without clicking the edit button.

    d. **editData()** - if the user clicks Edit button under Edit menu, then this function is called and using this function we disable the flags for each data item of the table so that the user can edit the data.

    e. **saveData()** - using this function there are two functionalities provided to the user.
        i. Save - if the user wants to save data in the current file, then the user can click on the Save button or press Ctrl+S

ii. Save As - if the user wants to save data in a new file, then the user can click on the Save As button or press Ctrl+Shift+S

f. **addData()** - using this functionality the user can either add column or row depending on his choice.
   i. **New Column** - if the user clicks this button or presses Ctrl + Shift + C a new column is added at the right end of the data table.
   ii. **New Row** - if the user clicks this button or presses Ctrl + Shift + R a new row is added at the bottom of the data table.

g. **closeEvent()** - this function is useful in case the user has not plotted anything and closes the application then the plot part will also get closed.

h. **plotData()** - if the user wants to plot data then he can click the Plot button on the menu bar or press Ctrl + Shift + P, then a new window gets open for plotting where the user can select any two columns for x and y-axes, and plot data as either Scatter Plot, Scatter Plot with Smooth Lines, or simply plot lines.

2. **Plot_Data class** - with functions like

a. **initui()** - to set the UI features for this window along with their functionalities like initialization of tab space, setting up selection box for selecting columns for x and y-axis, etc.

b. **fig()** - in this function we are handling three requests
   i. **Scatter Plot** - if the user presses plot for scatter plot the application then the function handles three conditions:

1. **Tab not initialized for Scatter Plot** - in this case, we make a new tab for Scatter plot, then plot the graph and set it as current Tab on the QTabWidget()
2. **When the tab is already initialized**:
    a. **Values of the Columns are changed** - in this case, we plot for the new values and set this tab as current Tab on the QTabWidget()
    b. **Values of the Columns are not altered** - in this case, we don't have to plot again just simply set this tab as the current Tab on the QTabWidget()

ii. **Scatter Plot with Smooth Lines** - all the conditions are handled similarly as for Scatter Plot, just in this case for smoothening of plot scipy libraries interp1d function and numpy library are used.

iii. **Line Plot** - all the conditions are handled similarly as for Scatter Plot.

c. **savePNG()** - this function is invoked if the user clicks on the 'Save as png' button under the File menu of the menu-bar. In case if the user has not plotted anything and tries to save, then he will get a Message Box with a warning "First Plot a Graph to save an Image in .png format" otherwise whichever Plots tab is currently active on QTabWidget() the user can save that plot. If the user wants to save other plots as well, then he can select the tab for that plot and then save it using the "Save as png" button.

SHORTCUT KEYS:

For the App Window:

Ctrl+O - to load a file.

Ctrl+E - to edit a file.

Ctrl+S - to save data in the currently open file.

Ctrl+Shift+S - to save data in a new file or as same file name by overwriting(replace it).

Ctrl+Shift+C - to add new column.

Ctrl+Shift+R - to add new row.

Ctrl+Shift+P - to open plotting window.

For the Plotting Window:

Ctrl+S - to save the image in png format.

ScreenShots of the Application:

1 - Loaded CSV File

	1	2	3	4	5	6	7	8	9
1	2323	22	72	35	0	33.6	0.627	50	1
2									
3	3232	85	66	29	0	26.6	0.351	31	0
4									
5	8	183	64	0	0	23.3	0.672	32	1
6									
7	1	89	66	23	94	28.1	0.167	21	0
8									
9	0	137	40	35	168	43.1	2.288	33	1
10									
11	5	116	74	0	0	25.6	0.201	30	0
12									
13	3	78	50	32	88	31	0.248	26	1
14									
15	10	115	0	0	0	35.3	0.134	29	0
16									
17	2	197	70	45	543	30.5	0.158	53	1
18									
19	8	125	96	0	0	0	0.232	54	1
20									
21	4	110	92	0	0	37.6	0.191	30	0
22									
23	10	168	74	0	0	38	0.537	34	1
24									
25	10	139	80	0	0	27.1	1.441	57	0
26									

2 - With Added Column

FOSSEE SCREENING TASK 2

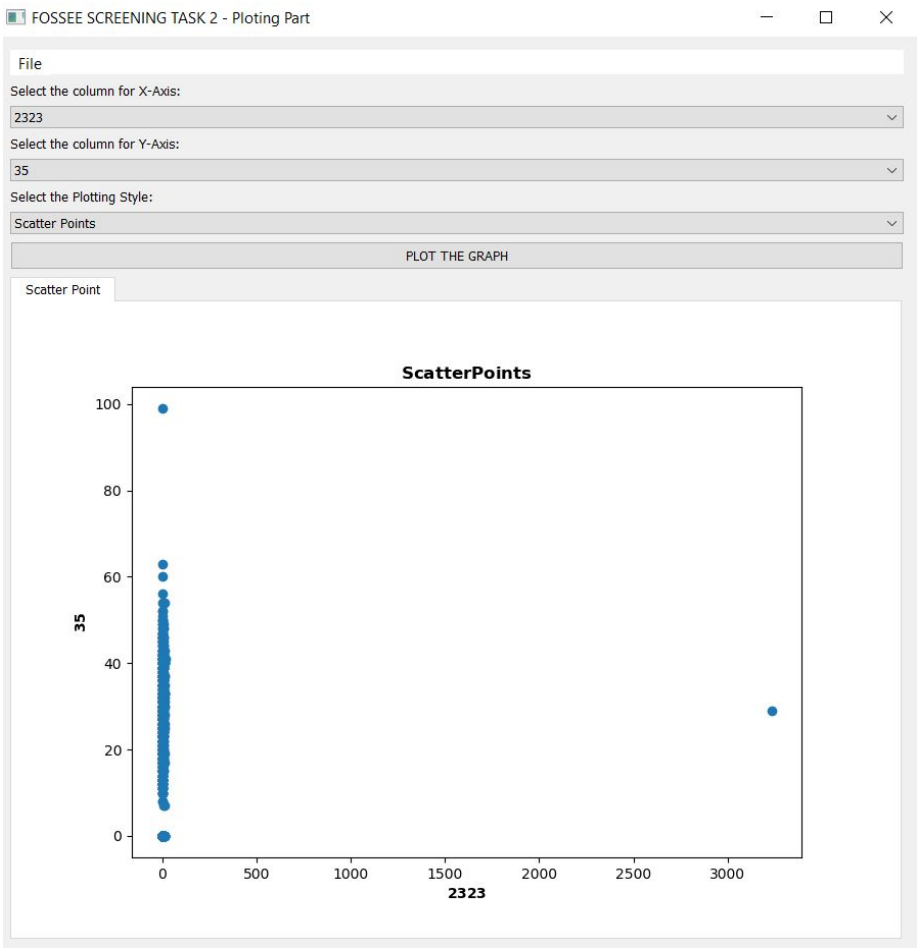
File Edit Plot

	1	2	3	4	5	6	7	8	9	10
1	2323	22	72	35	0	33.6	0.627	50	1	23
2										34
3	3232	85	66	29	0	26.6	0.351	31	0	545
4										45
5	8	183	64	0	0	23.3	0.672	32	1	2
6										4
7	1	89	66	23	94	28.1	0.167	21	0	345
8										44
9	0	137	40	35	168	43.1	2.288	33	1	2
10										24
11	5	116	74	0	0	25.6	0.201	30	0	5
12										5
13	3	78	50	32	88	31	0.248	26	1	6
14										5
15	10	115	0	0	0	35.3	0.134	29	0	5
16										5
17	2	197	70	45	543	30.5	0.158	53	1	2
18										4
19	8	125	96	0	0	0	0.232	54	1	4
20										3
21	4	110	92	0	0	37.6	0.191	30	0	4
22										
23	10	168	74	0	0	38	0.537	34	1	
24										
25	10	139	80	0	0	27.1	1.441	57	0	
26										

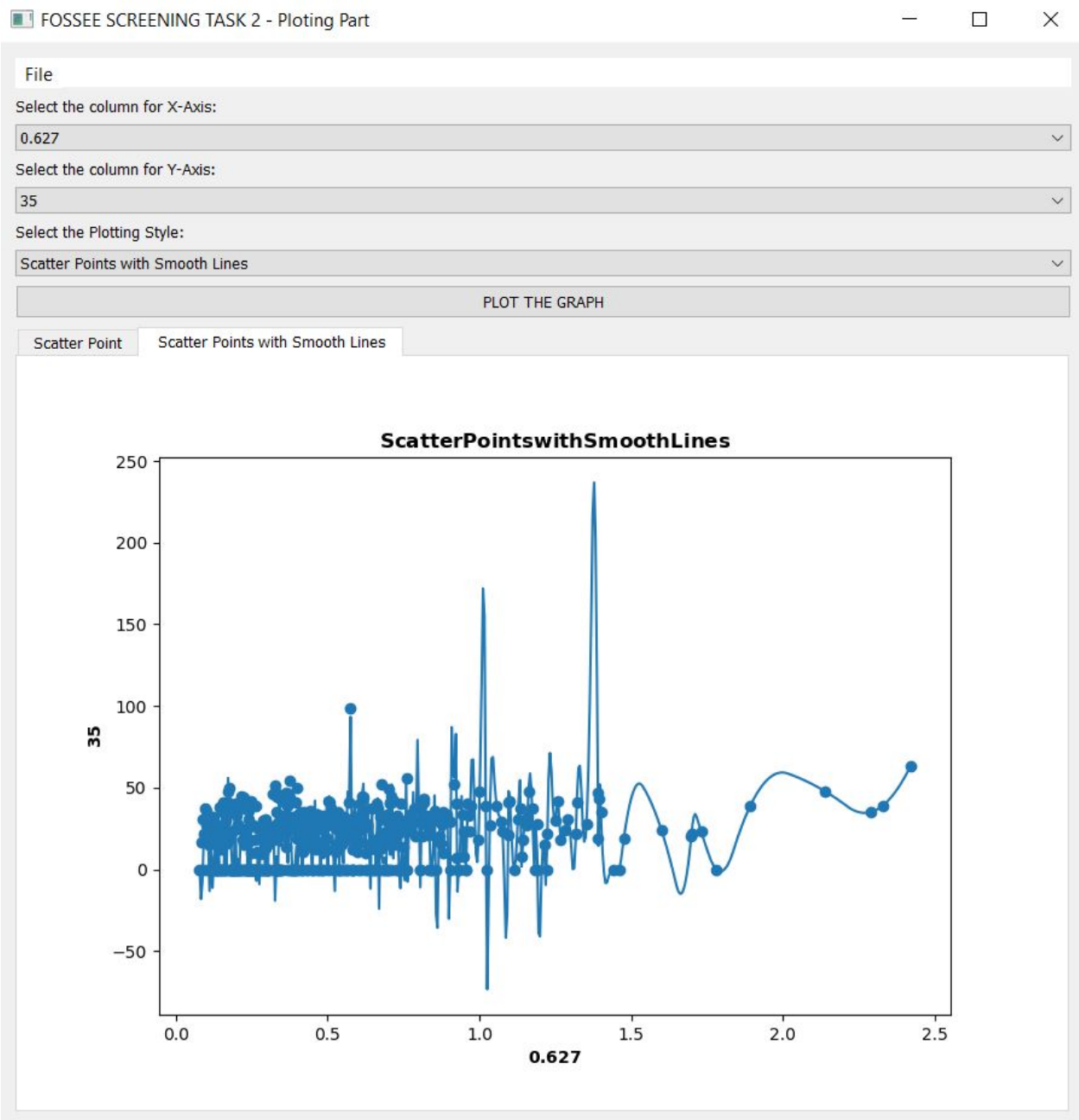
3 - With Added Row’s

1531	5	121	72	23	112	26.2	0.245	30	0
1532									
1533	1	126	60	0	0	30.1	0.349	47	1
1534									
1535	1	93	70	31	0	30.4	0.315	23	0
1536									
1537									
1538									
1539									
1540									
1541									
1542									

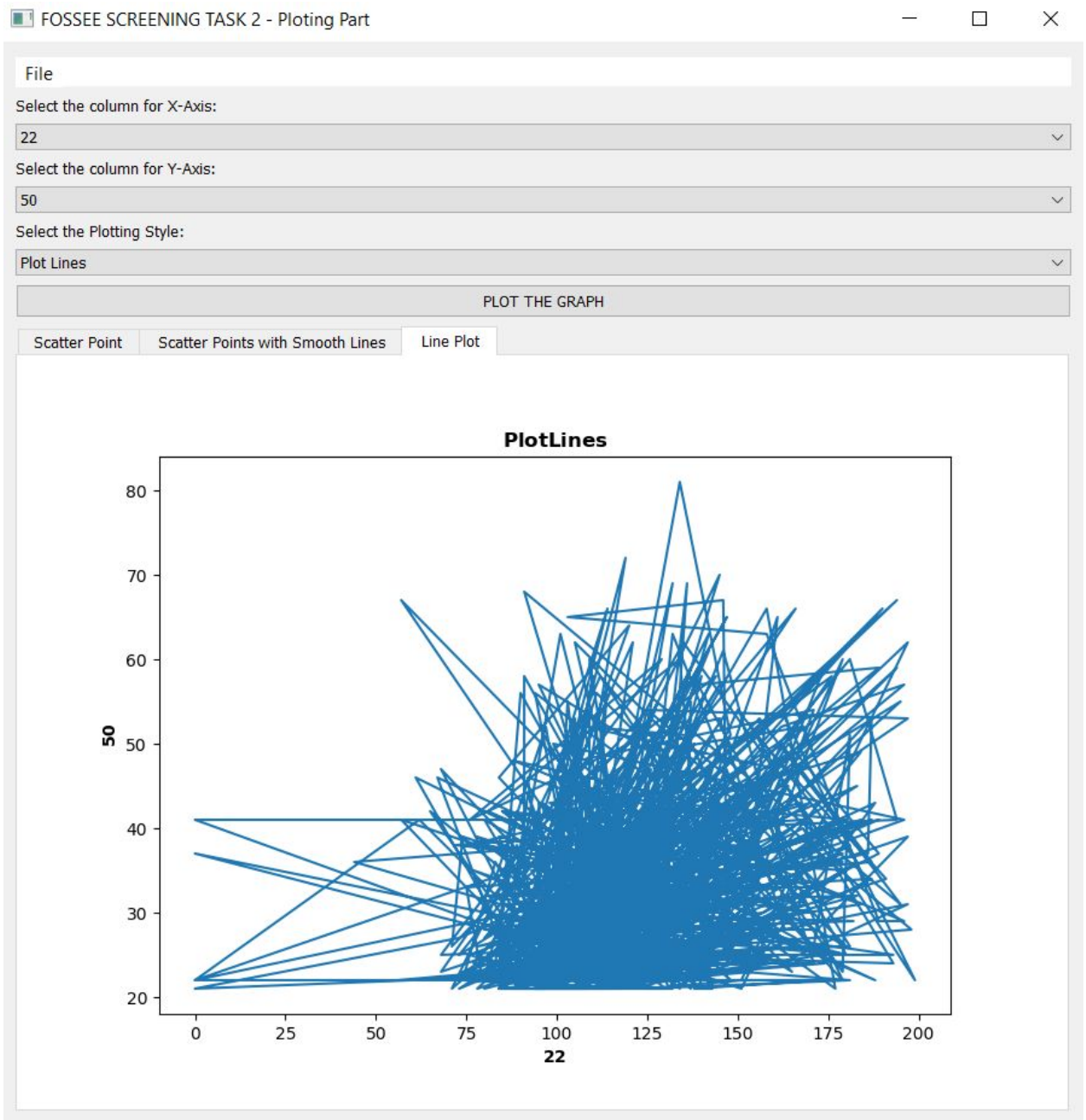
4 - Scatter Plot



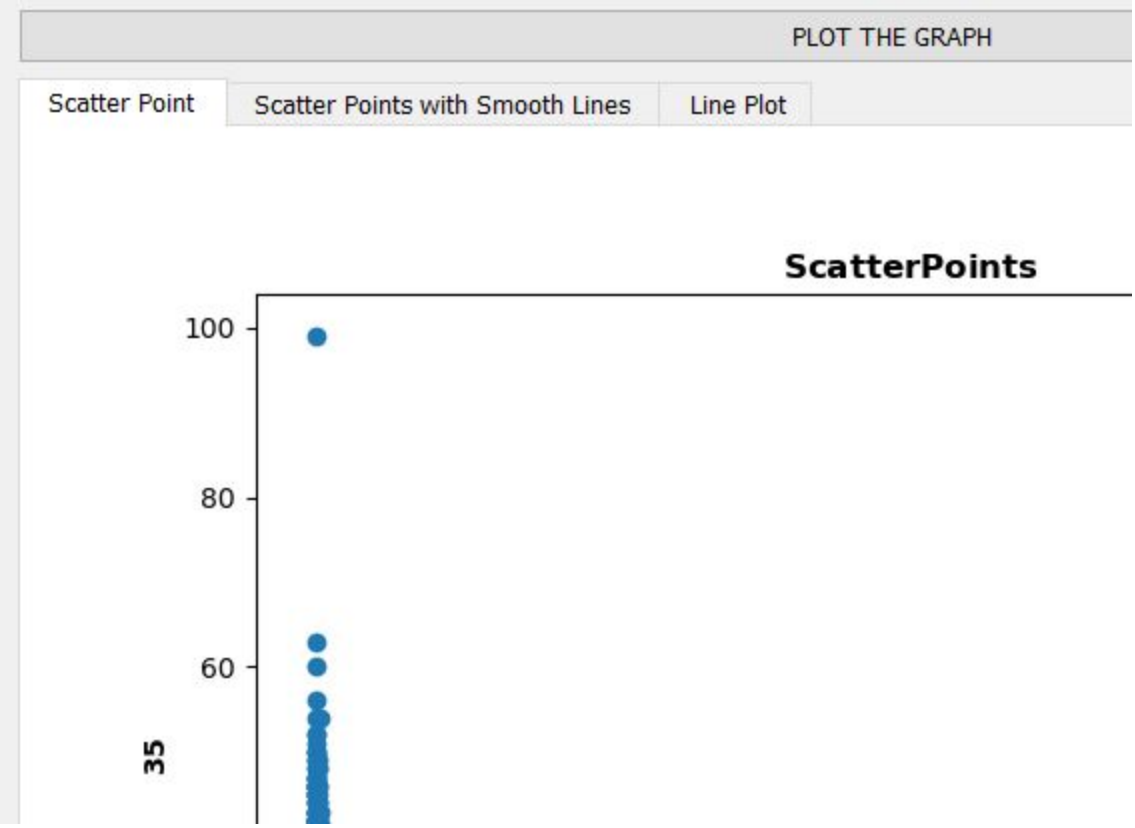
5 - Scatter Plot with Smooth Lines



6 - Line Plot


FOSSEE SCREENING TASK 2 - Ploting Part
File
Select the column for X-Axis:
22
Select the column for Y-Axis:
50
Select the Plotting Style:
Plot Lines
PLOT THE GRAPH
Scatter Point
Scatter Points with Smooth Lines
Line Plot
PlotLines
50
22
80
70
60
50
40
30
20
0
25
50
75
100
125
150
175
200

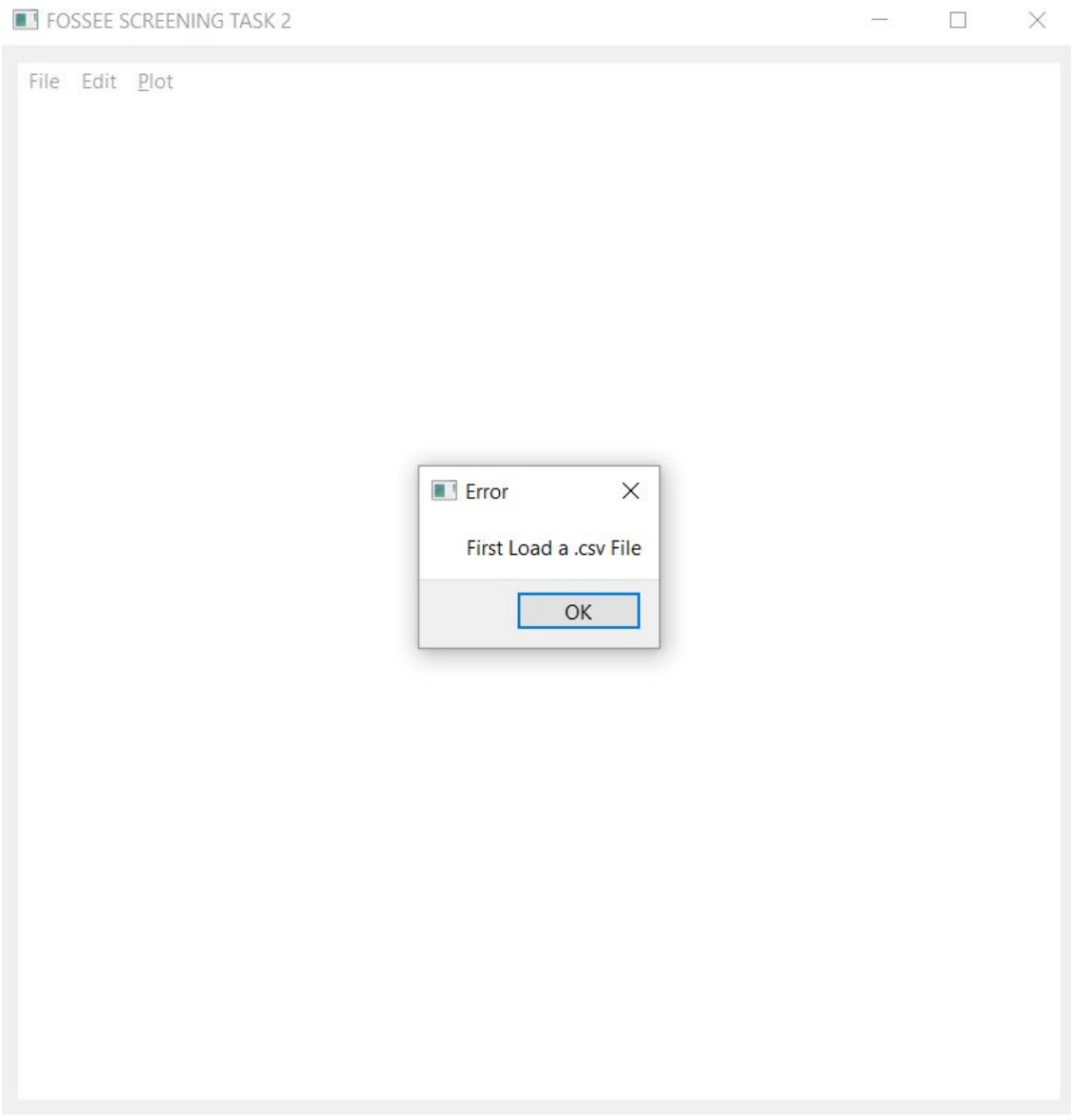
7 - Different Tab for Different Plot



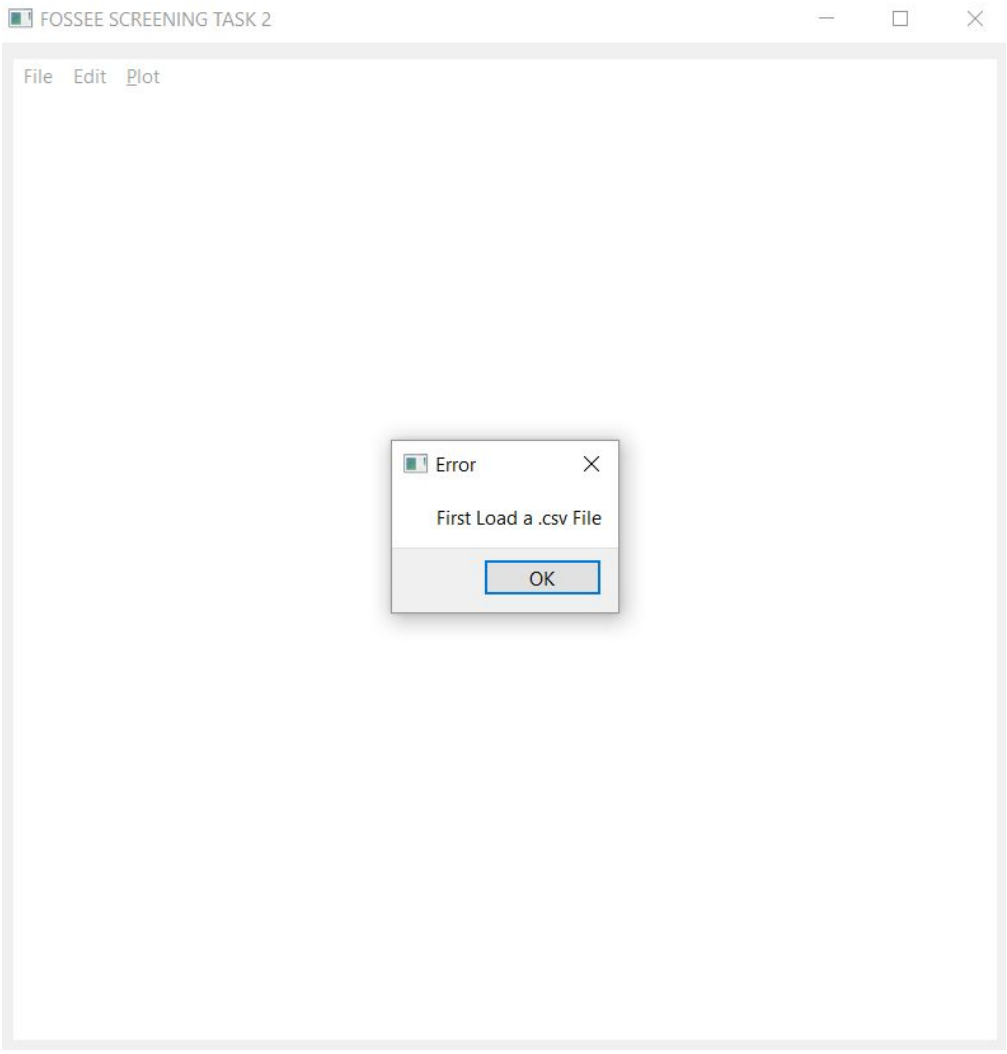
CORNER CASES:

All the corner cases as handled with a MessageBox:

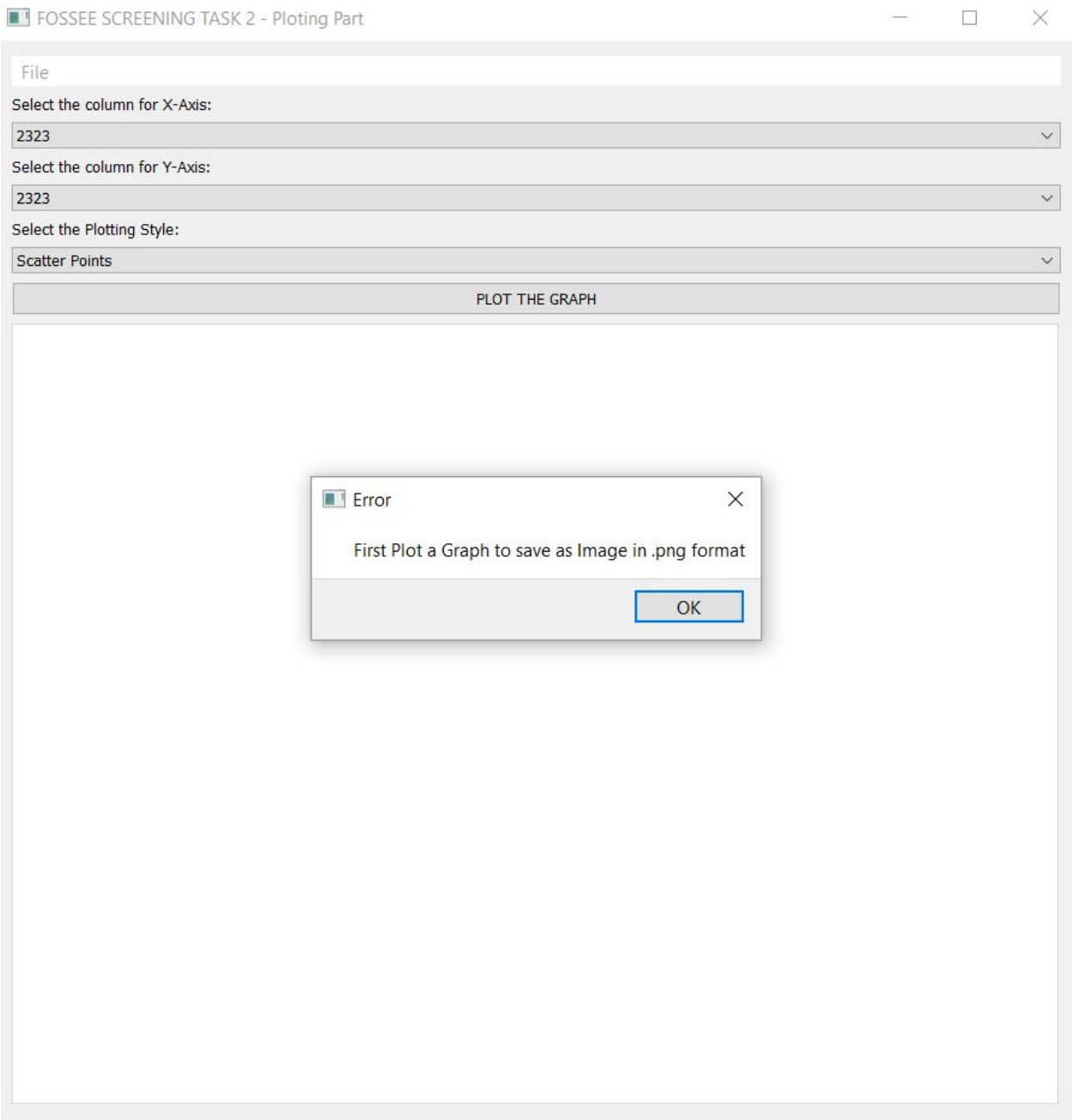
1 - If the user presses the Edit button without opening a file



2 - If the user tries to open the plotting window without loading a file first:



3 - If the user tries to press “Save as png” button without plotting



And other cases like these are also handled with a MessageBox.