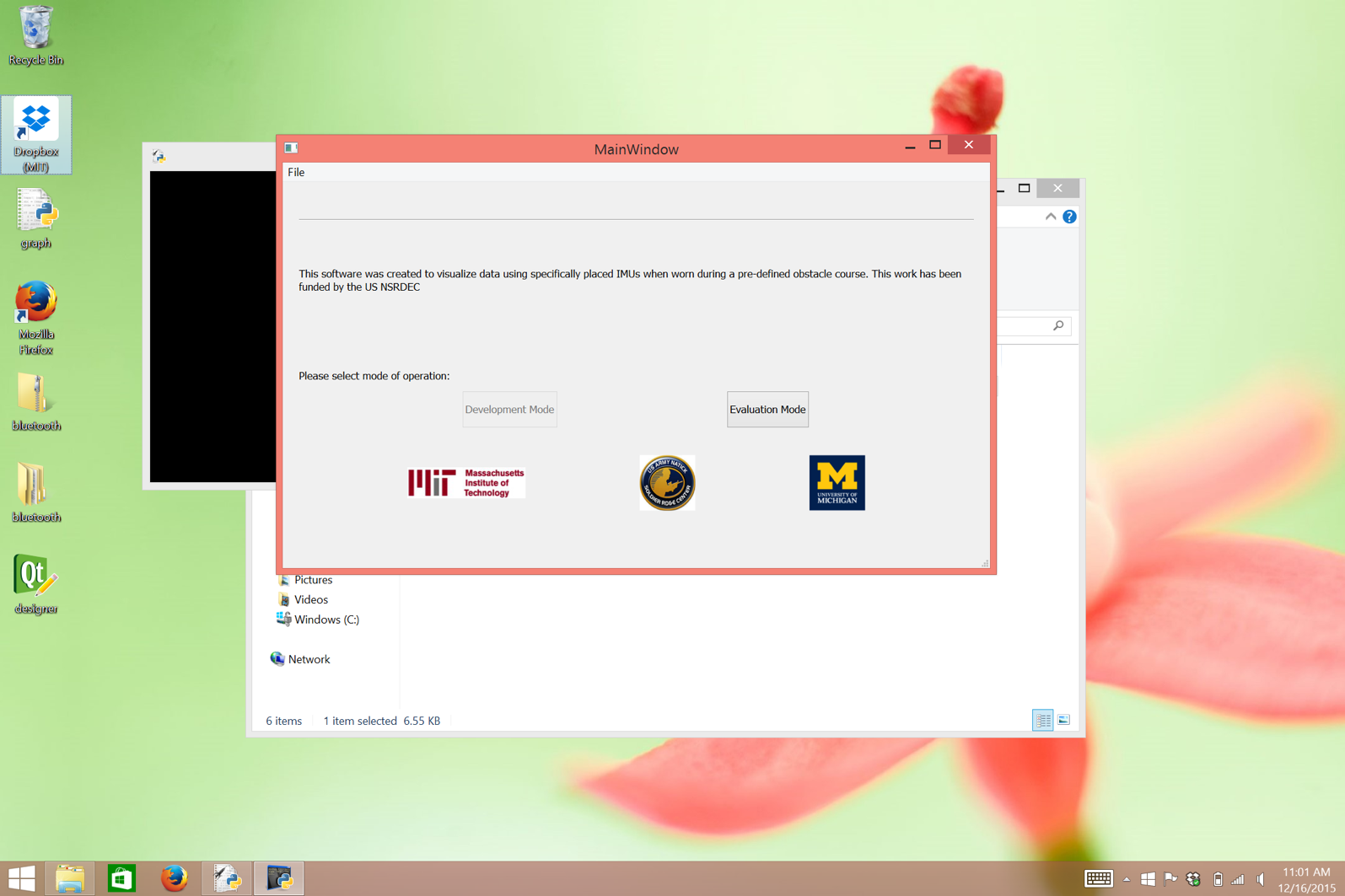
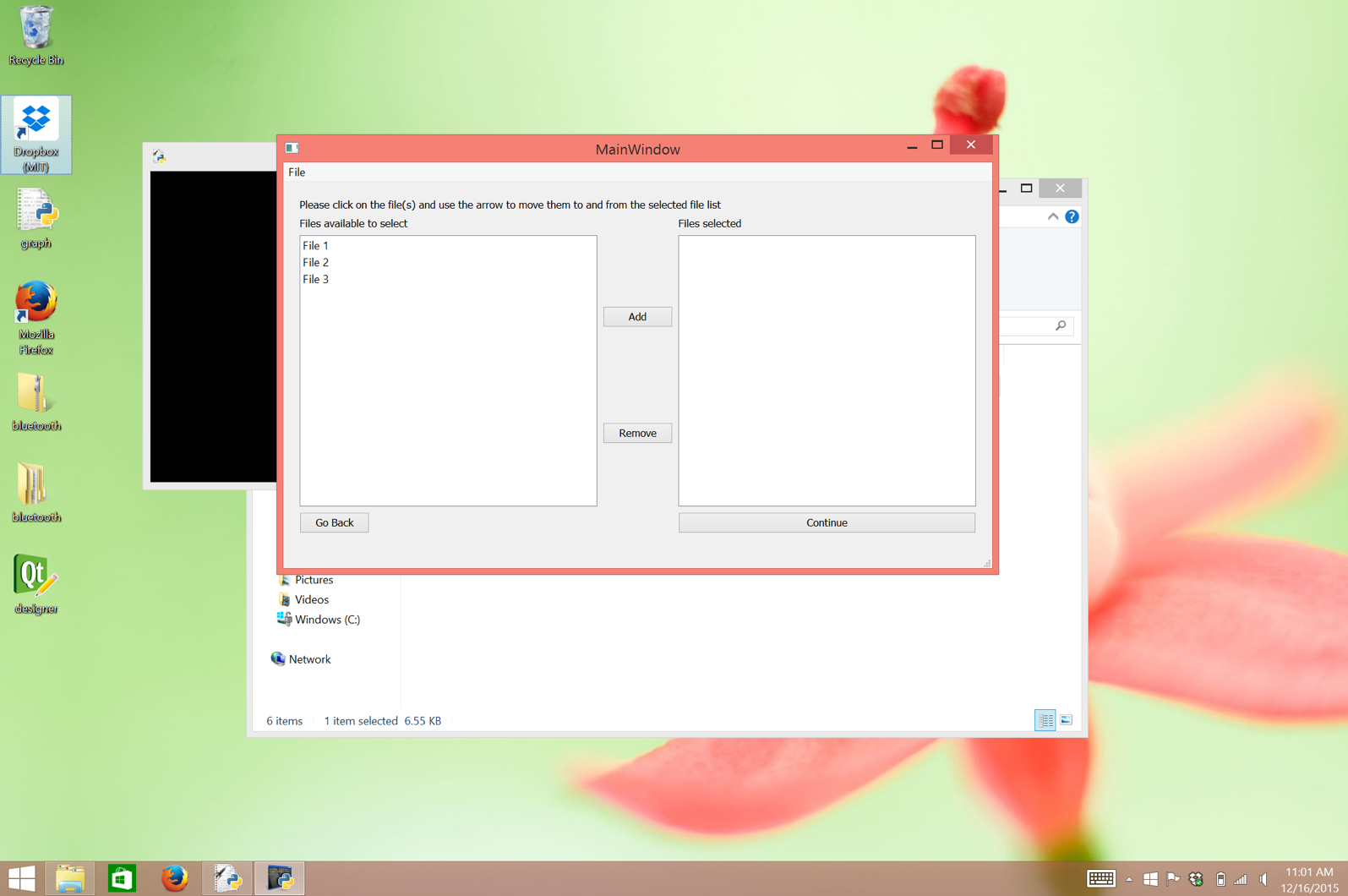
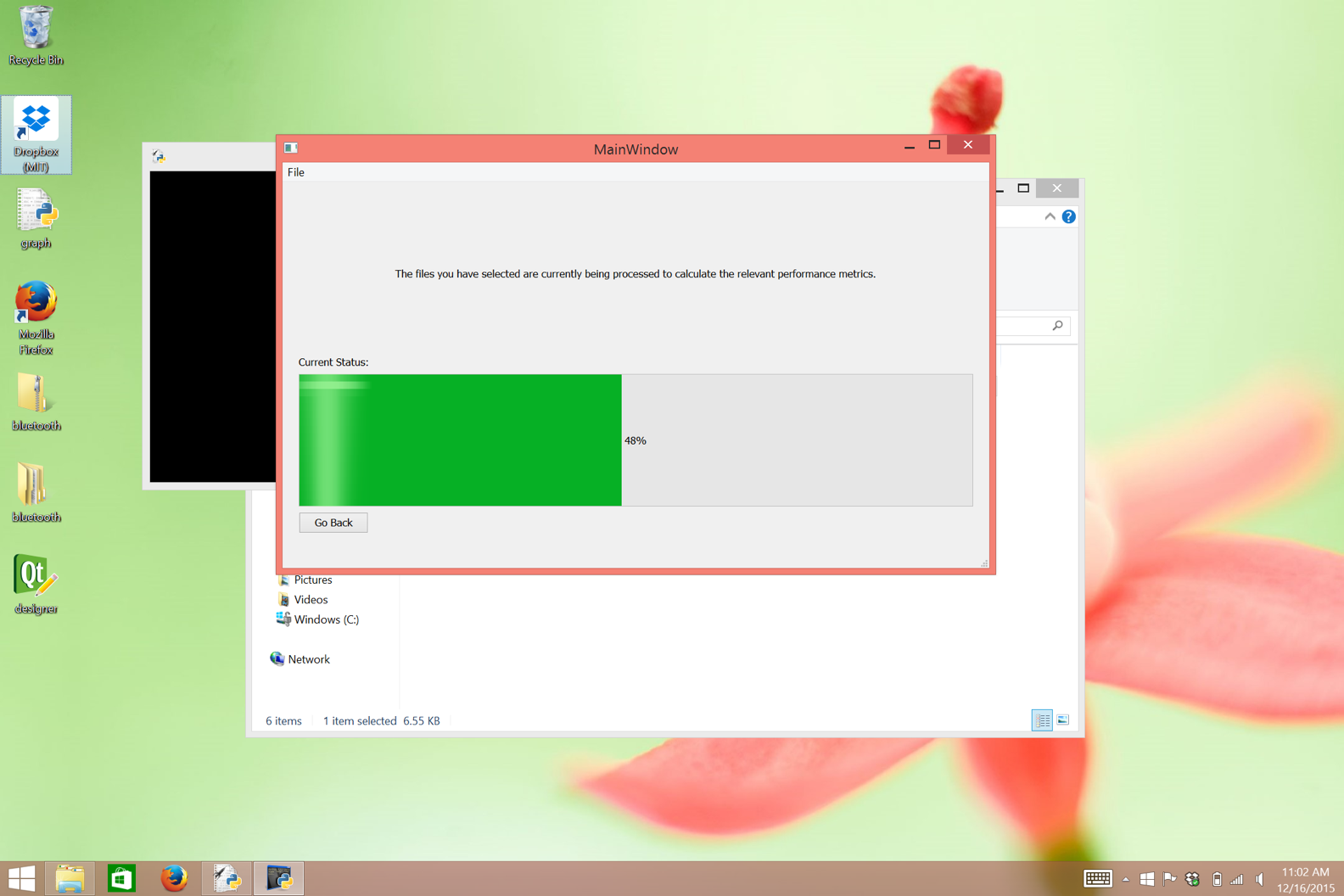
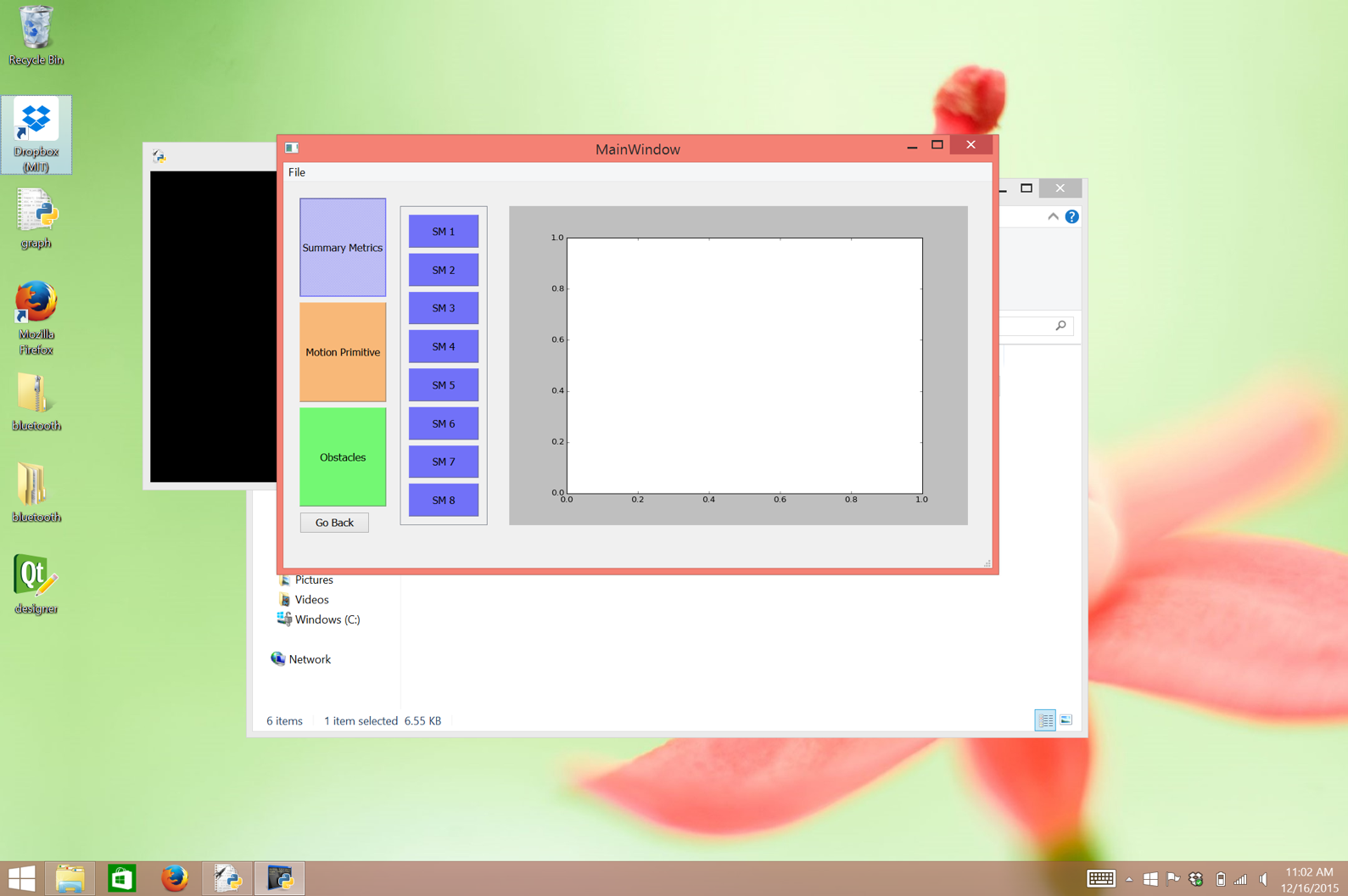
GUI

The purpose of the project was to create a python based GUI in order to display and analyse data related to soldier performance. The GUI has two main files: GUI.ui and GUI.py.

The .ui file was created using the QT Designer tool that is automatically installed with the Qt package; it builds the actual GUI interface. With it a 5 screen interface was created. Each screen includes different widgets and layouts depending on the screen’s function. Below are some screenshots of what the user would see in each different section.







The usage of the QT Designer tool, made it significantly easier to manipulate the layout and appearance of the widgets therefore giving the GUI a more intuitive feel and a more pleasing look. In addition, the QT Designer has the ability to rename and edit the properties of each widget used in the GUI. Therefore, reducing the number of lines of code used in the .py file.

The .ui file, once it has been completed, is loaded in the .py file. If any edits are made on the .ui file they will be affect the .py file as well.

The .py files was written based on the design on the .ui file. It is made up of a main class *MyWindowClass* which inherits from *QtGui.QMainWindow*, a class that provides a framework for building an interface. Inside this class, the .ui file is loaded and, functions and variables used in the GUI are defined. The lines 36-48, shown below, connect buttons on the .ui file to functions defined later in the code.

*### Connect buttons to functions*

*self.EvaluationModeButton.clicked.connect(self.EvaluationModeButton\_clicked)*

*self.BackButton1.clicked.connect(self.BackButton1\_clicked)*

*self.ContinueButton.clicked.connect(self.ContinueButton\_clicked)*

*self.ContinueButton2.clicked.connect(self.ContinueButton2\_clicked)*

*self.AddButton.clicked.connect(self.AddButton\_clicked)*

*self.RemoveButton.clicked.connect(self.RemoveButton\_clicked)*

*self.BackButton2.clicked.connect(self.BackButton2\_clicked)*

*self.BackButton3.clicked.connect(self.BackButton3\_clicked)*

*self.BackButton4.clicked.connect(self.BackButton4\_clicked)*

*self.SummaryMetricsButton.clicked.connect(self.SummaryMetricsButton\_clicked)*

*self.ObstaclesButton.clicked.connect(self.ObstaclesButton\_clicked)*

*self.MotionPrimitiveButton.clicked.connect(self.MotionPrimitiveButton\_clicked)*

Immediately, after the above lines, the button functions are defined. They are arranged by what page they appear into, as denoted by the comments. Some functions to note are *AddButton\_clicked* and *RemoveButton\_clicked* functions which add and remove files from their respective lists, as well as the *AddGraph, RemoveGraph* and *Plot* functions. The *AddGraph* function creates a FigureCanvas with the specified figure (graph), adds it onto the layout and finally draws the graph while the *RemoveGraph* does the opposite. Finally, the *Plot* function calls the *AddGraph* function mentioned above and gives it a figure input to plot.

Outside the main class, following the functions, are a couple of example plots and files used to test the plotting and graph functions. There are 4 examples: *fig0, fig1, fig2* and *fig3*. *fig0* corresponds to plotting only axes and is used to show the graph’s place before there is any data to plot. *fig1*, is a figure created by reading a file, plot.txt, use inly to demonstrate the GUI’s ability to open and read data files. Finally, *fig2* and *fig3* are figures that either plot random numbers or a random color mesh. They are used to illustrate the GUI’s ability to graph a different graph depending on what the user has selected.

The last four lines of the code, shown below, execute the main class and all it’s functions. Without those lines, the GUI will not run.

*app = QtGui.QApplication(sys.argv)*

*myWindow = MyWindowClass(None)*

*myWindow.show()*

*app.exec\_()*

Lastly, the first 9 lines of the .py file, are imports necessary to run the GUI. The first two, are importing the GUI libraries as well as system parameters. While the rest, import plotting and math libraries.

In future, there are various things to implement for the GUI to function completely as envisioned in the beginning. Below is list of the things that remain to be done and a brief description of each.

1. Implement the ability to save files.
   1. Currently, the GUI has the ability to open a .txt file that contains a specific data format that is then plotted. However, it currently does not have the ability to save data nor a graph image (.jpeg file or similar)
2. Implement a working download bar.
   1. Currently, the download bar only simulates a real download as it is, in no way, connected to opening files.
3. Connect last page’s buttons to functions
   1. The last page’s buttons used to arrange and select graphs, currently have no functions connected to them.
4. Implement a “Development Mode”.
   1. Currently, the GUI only supports a “Evaluation Mode” however, the button indicating the ability to include “Development Mode” in the future is there.