OPERATING SYSTEM

GUI BASED SIMULATOR ON CPU SCHEDULING ALGORITHMS

G1 TEAM 1

Hardik Inani (Team Leader)	20BCP012
Nilay Patel	20BCP005
Rahul Gulati	20BCP024
Dhvanil Bhagat	20BCP027
Shrey Makadiya	20BCP028
Drashti Bhavsar	20BCP040

TABLE OF CONTENTS

Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Website, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

PyQt5

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications.

PyQt5 Components

PyQt5 comprises a number of different components. First of all there are a number of Python extension modules. These are all installed in the PyQt5 Python package and are described in the list of modules.
Each extension module has a corresponding PEP 484 defined stub file containing type hints for the module's API. This can be used by static type checkers such as mypy.
PyQt5 contains plugins that enable Qt Designer and qmlscene to be extended using Python code. See Writing Qt Designer Plugins and Integrating Python and QML respectively for the details.

PyQt5 also contains a number of utility programs.

- pyuic5 corresponds to the Qt uic utility. It converts QtWidgets based GUIs created using Qt Designer to Python code.
- pyrcc5 corresponds to the Qt rcc utility. It embeds arbitrary resources (eg. icons, images, translation files) described by a resource collection file in a Python module.
- pylupdate5 corresponds to the Qt lupdate utility. It extracts all of the translatable strings from Python code and creates or updates .ts translation files. These are then used by Qt Linguist to manage the translation of those strings.

The DBus support module is installed as dbus.mainloop.pyqt5. This module provides support for the Qt event loop in the same way that the dbus.mainloop.glib included with the standard dbus-python bindings package provides support for the GLib event loop. The API is described in DBus Support. It is only available if the dbus-python v0.80 (or later) bindings package is installed. The QtDBus module provides a more Qt-like interface to DBus.
PyQt5 includes a large number of examples. These are ports to Python of many of the C++ examples provided with Qt. They can be found in the examples directory of the sdist.
Finally, PyQt5 contains the specification files that allow bindings for other Qt based class libraries that further extend PyQt5 to be developed and installed.

PyQt Introduction

PyQt5 also contains a number of utility programs.

- pyuic5 corresponds to the Qt uic utility. It converts QtWidgets based GUIs created using Qt Designer to Python code.
- pyrcc5 corresponds to the Qt rcc utility. It embeds arbitrary resources (eg. icons, images, translation files) described by a resource collection file in a Python module.
- pylupdate5 corresponds to the Qt lupdate utility. It extracts all of the translatable strings from Python code and creates or updates .ts translation files. These are then used by Qt Linguist to manage the translation of those strings.

The DBus support module is installed as dbus.mainloop.pyqt5. This module provides support for the Qt event loop in the same way that the dbus.mainloop.glib included with the standard dbus-python bindings package provides support for the GLib event loop. The API is described in DBus Support. It is only available if the dbus-python v0.80 (or later) bindings package is installed. The QtDBus module provides a more Qt-like interface to DBus.

PyQt5 includes a large number of examples. These are ports to Python of many of the C++ examples provided with Qt. They can be found in the examples directory of the sdist.

Finally, PyQt5 contains the specification files that allow bindings for other Qt based class libraries that further extend PyQt5 to be developed and installed.

Major Classes

PyQt API is a large collection of classes and methods. These classes are defined in more than 20 modules. Following are some of the frequently used modules: QtCore
Core non-GUI classes used by other modules

- **QtGui**

Graphical user interface components

- **QtMultimedia**

Classes for low-level multimedia programming

- **QtNetwork**

Classes for network programming

- **QtOpenGL**

OpenGL support classes

- **QtScript**

Classes for evaluating Qt Scripts

- **QtSql**

Classes for database integration using SQL

- **QtSvg**

Classes for displaying the contents of SVG files

- **QtWebKit**

Classes for rendering and editing HTML

- **QtXml**

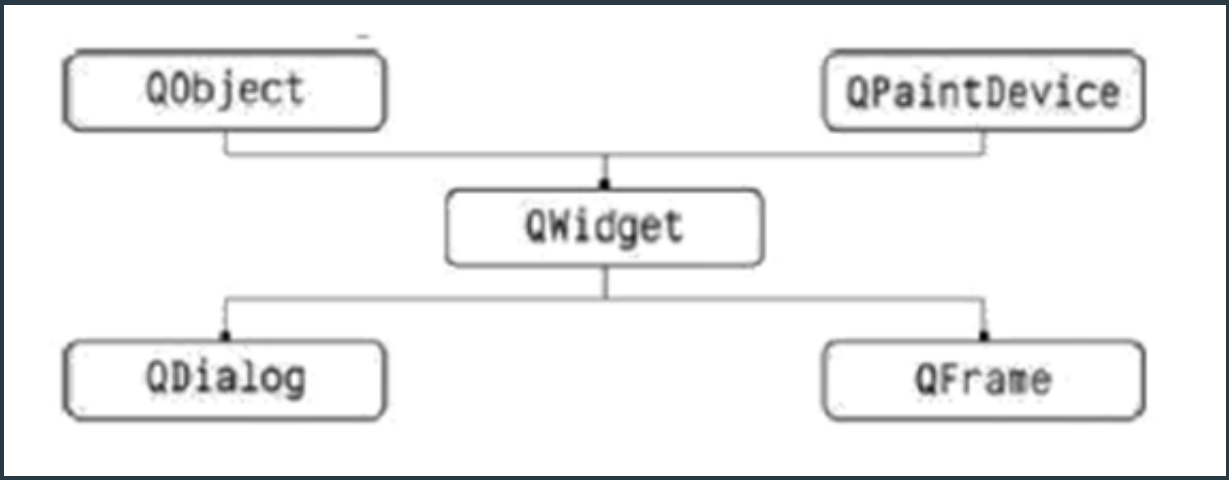
Classes for handling XML

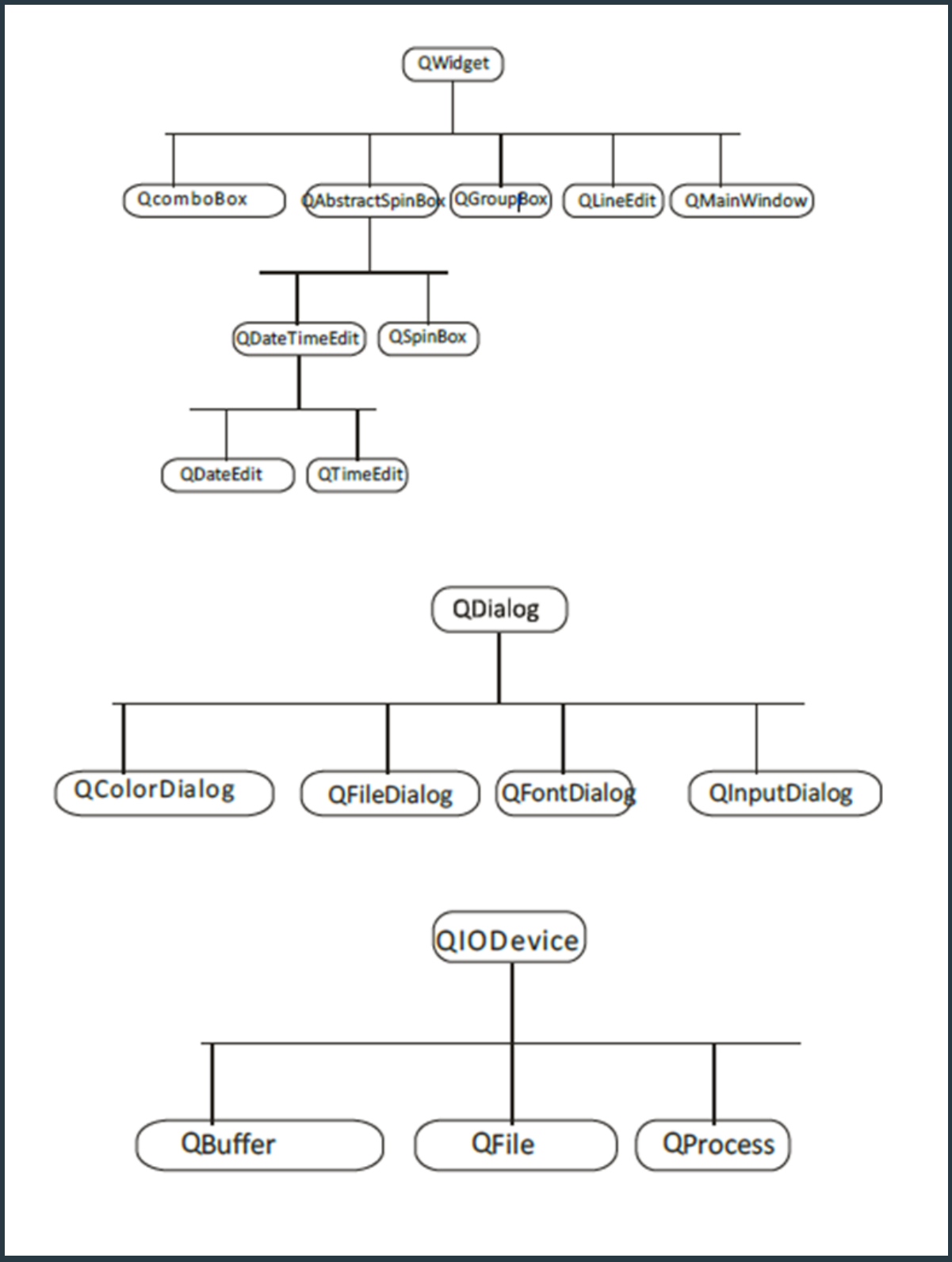
- **QtAssistant**

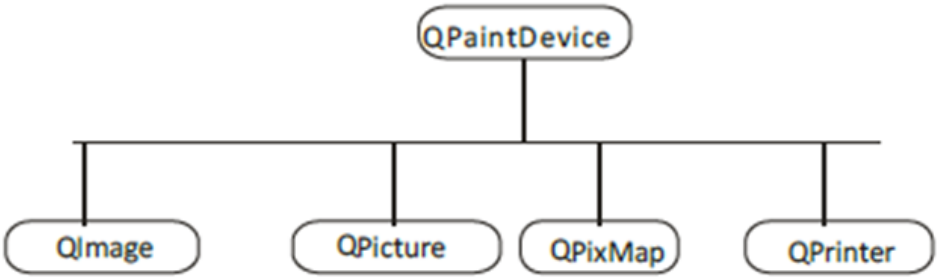
Support for online help

- **QtDesigner**

Classes for extending Qt Designer


QObject
QPaintDevice
QWidget
QDialog
QFrame


QWidget
QcomboBox
QAbstractSpinBox
QGroupBox
QLineEdit
QMainWindow
QDateTimeEdit
QSpinBox
QDateEdit
QTimeEdit
QDialog
QColorDialog
QFileDialog
QFontDialog
QInputDialog
QIODevice
QBuffer
QFile
QProcess



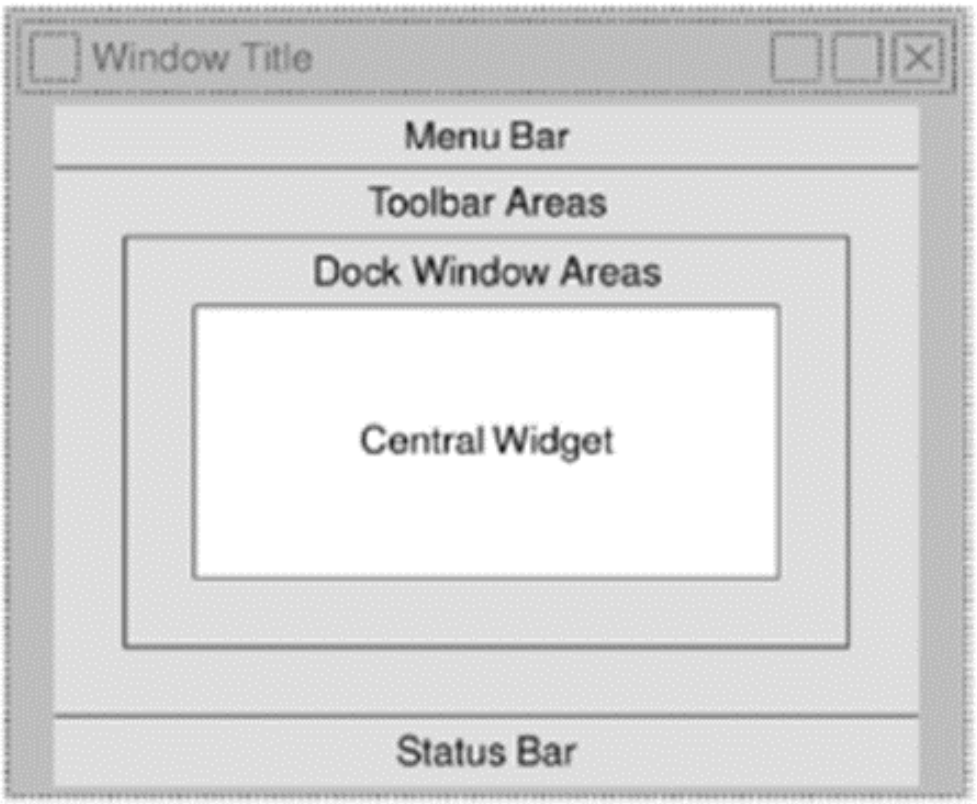
Here is a select list of frequently used widgets:

QLabel	Used to display text or image
QLineEdit	Allows the user to enter one line of text
QTextEdit	Allows the user to enter multi-line text
QPushButton	A command button to invoke action
QRadioButton	Enables to choose one from multiple options
QCheckBox	Enables choice of more than one options
QSpinBox	Enables to increase/decrease an integer value
QScrollBar	Enables to access contents of a widget beyond display aperture
QSlider	Enables to change the bound value linearly.
QComboBox	Provides a dropdown list of items to select from
QMenuBar	Horizontal bar holding QMenu objects
QStatusBar	Usually at bottom of QMainWindow, provides status information.
QToolBar	Usually at top of QMainWindow or floating. Contains action buttons
QListView	Provides a selectable list of items in ListMode or IconMode
QPixmap	Off-screen image representation for display on QLabel or QPushButton object
QDialog	Modal or modeless window which can return information to parent window

A typical GUI based application's top level window is created by **QMainWindow** widget object. Some widgets as listed above take their appointed place in this main window, while others are placed in the central widget area using various layout managers.

The following diagram shows the QMainWindow framework:

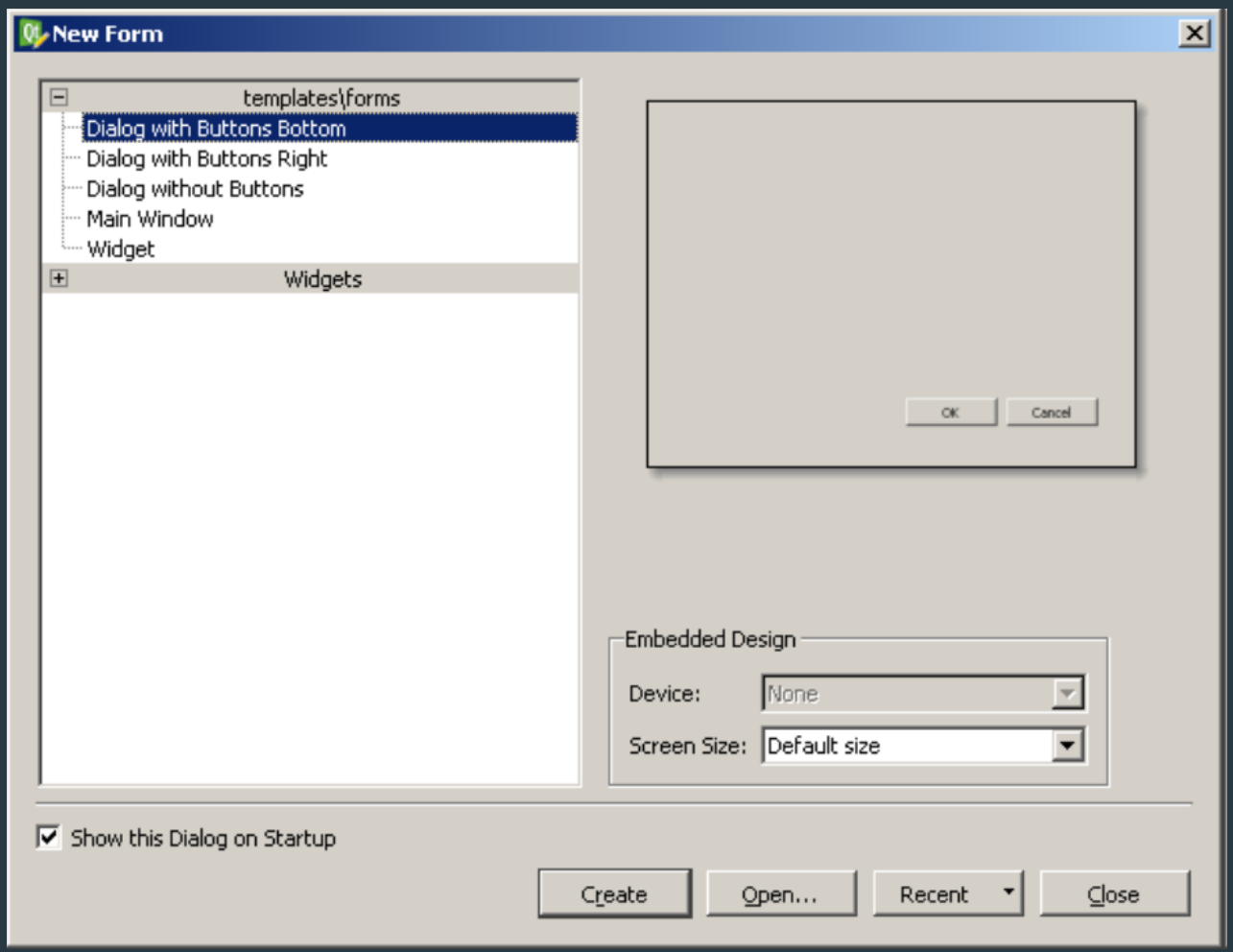
1


Window Title
Menu Bar
Toolbar Areas
Dock Window Areas
Central Widget
Status Bar

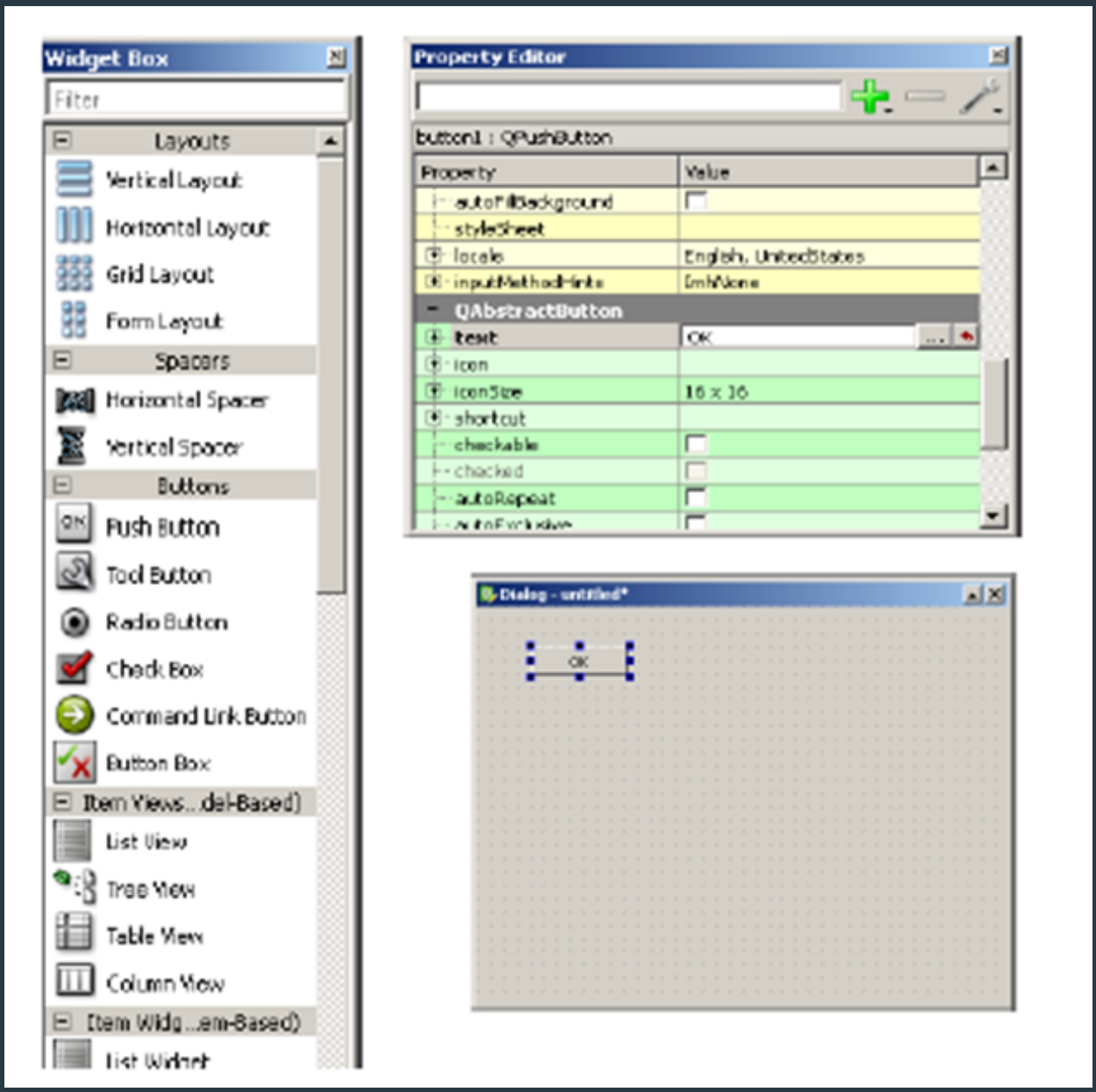
3

Using QtDesigner

The PyQt installer comes with a GUI builder tool called Qt Designer. Using its simple drag and drop interface, a GUI interface can be quickly built without having to write the code. It is however, not an IDE such as Visual Studio. Hence, Qt Designer does not have the facility to debug and build the application.
Creation of a GUI interface using Qt Designer starts with choosing a top level window for the application.



You can then drag and drop required widgets from the widget box on the left pane. You can also assign value to properties of widget laid on the form.


Widget Box
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Button Box
Item Views...del-Based)
List View
Tree View
Table View
Column View
Item Widg...em-Based)
List Widget
Property Editor
button1 : QPushButton
Property
Value
autoFillBackground
styleSheet
locale
English, UnitedStates
inputMethodHints
ImhNone
QAbstractButton
text
OK
icon
iconSize
16 x 16
shortcut
checkable
checked
autoRepeat
Dialog - untitled*
OK

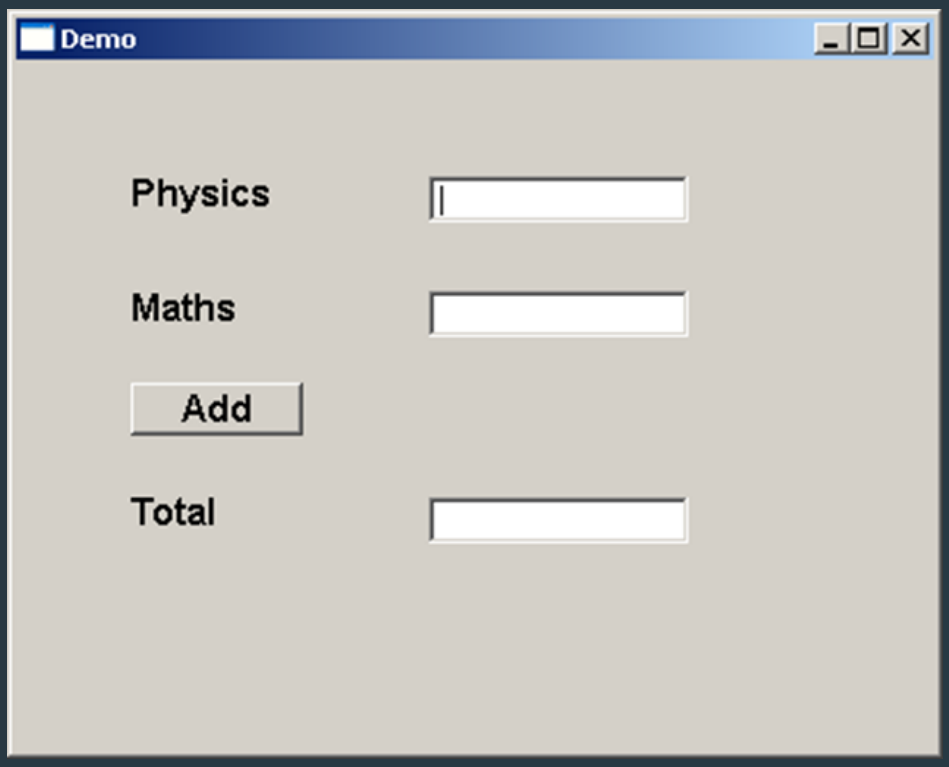
The designed form is saved as demo.ui. This ui file contains XML representation of widgets and their properties in the design. This design is translated into Python equivalent by using pyuic4 command line utility. This utility is a wrapper for uic module. The usage of pyuic4 is as follows:

pyuic4 –x demo.ui –o demo.py

In the above command, -x switch adds a small amount of additional code to the generated XML so that it becomes a self-executable standalone application.

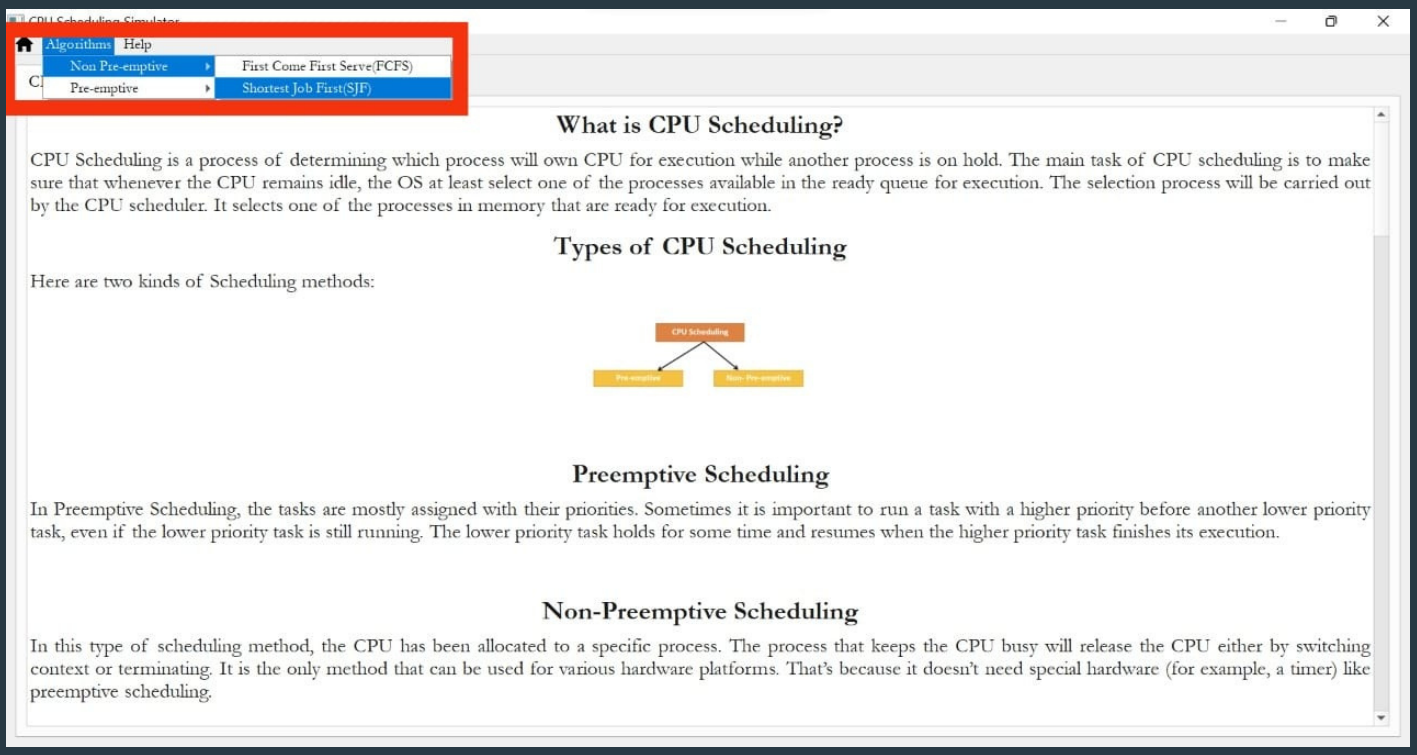
```
if __name__ == "__main__":
import sys
app = QtGui.QApplication(sys.argv)
Dialog = QtGui.QDialog()
ui = Ui_Dialog()
ui.setupUi(Dialog)
Dialog.show()
sys.exit(app.exec_())
```

The resultant python script is executed to show the following dialog box:

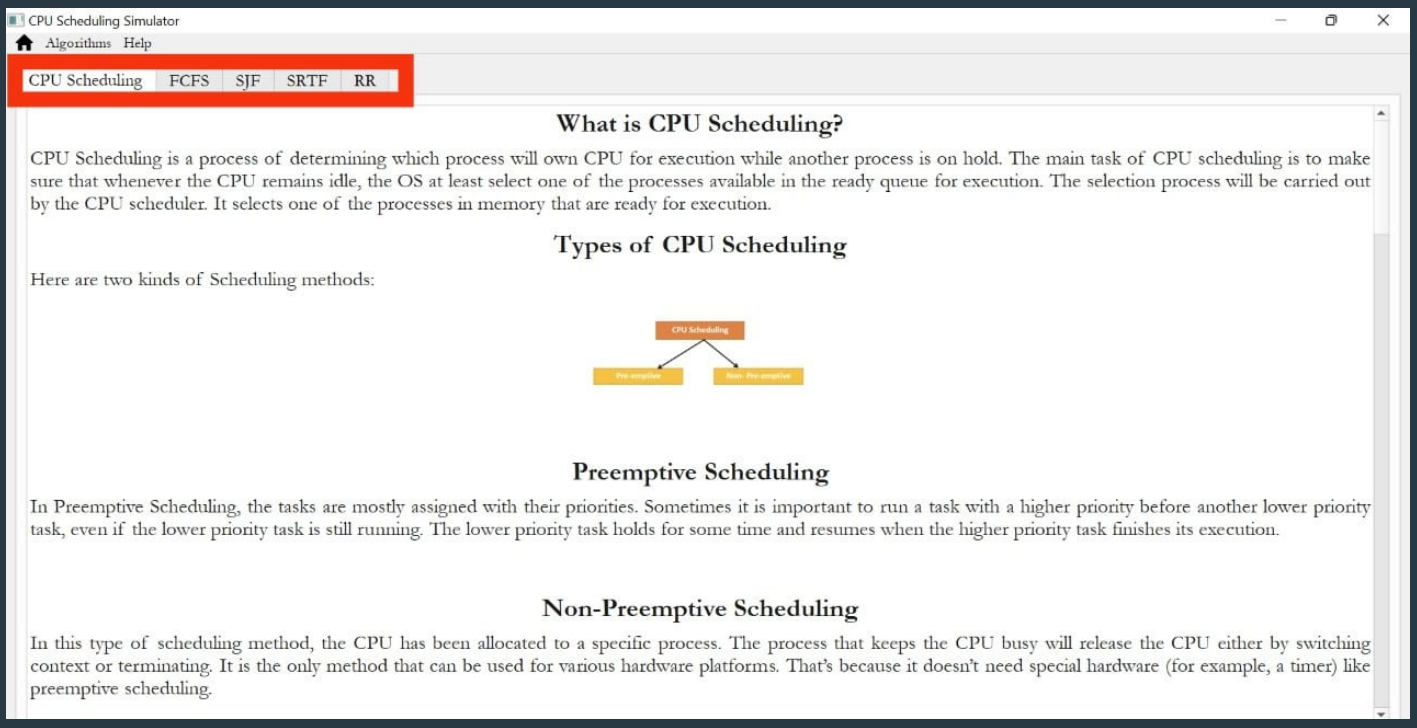


The user can input data in input fields but clicking on Add button will not generate any action as it is not associated with any function. Reacting to user-generated response is called as event handling.

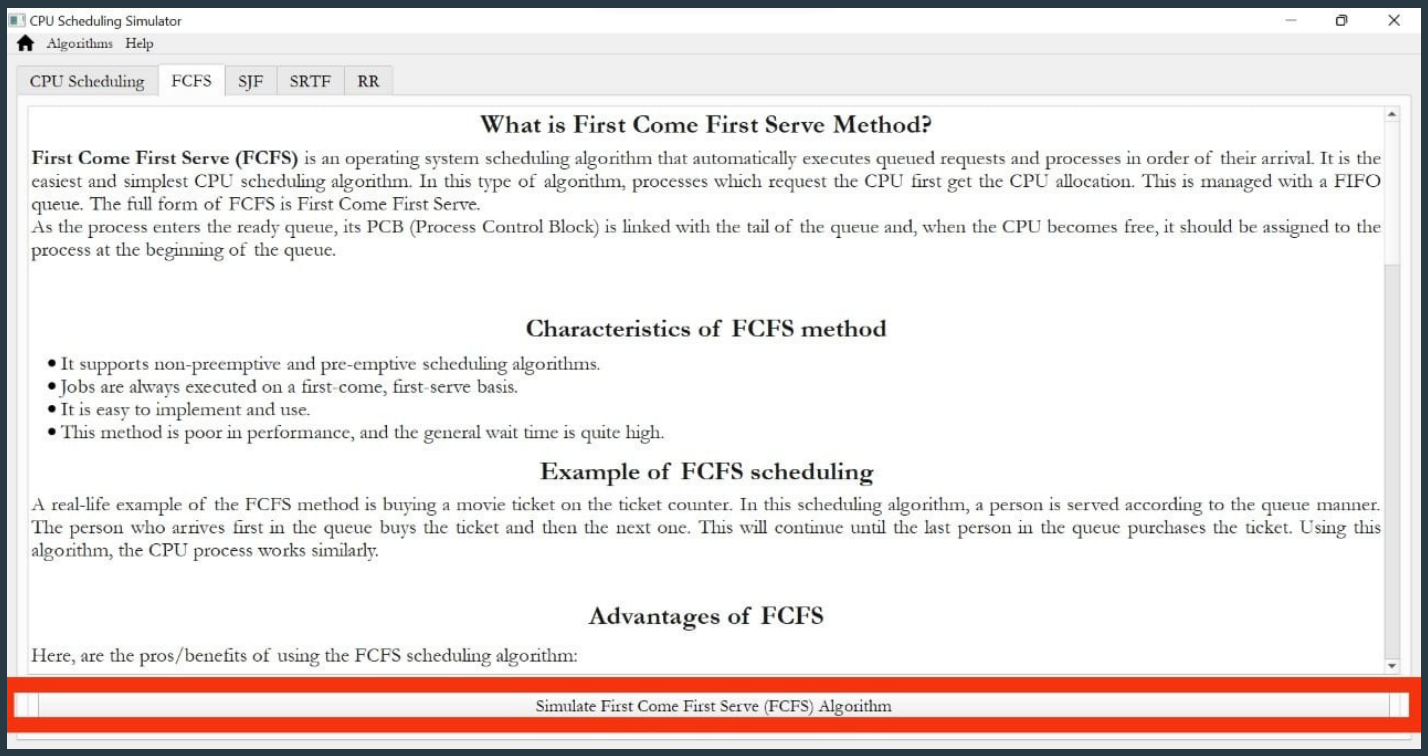
Using the GUI Simulator



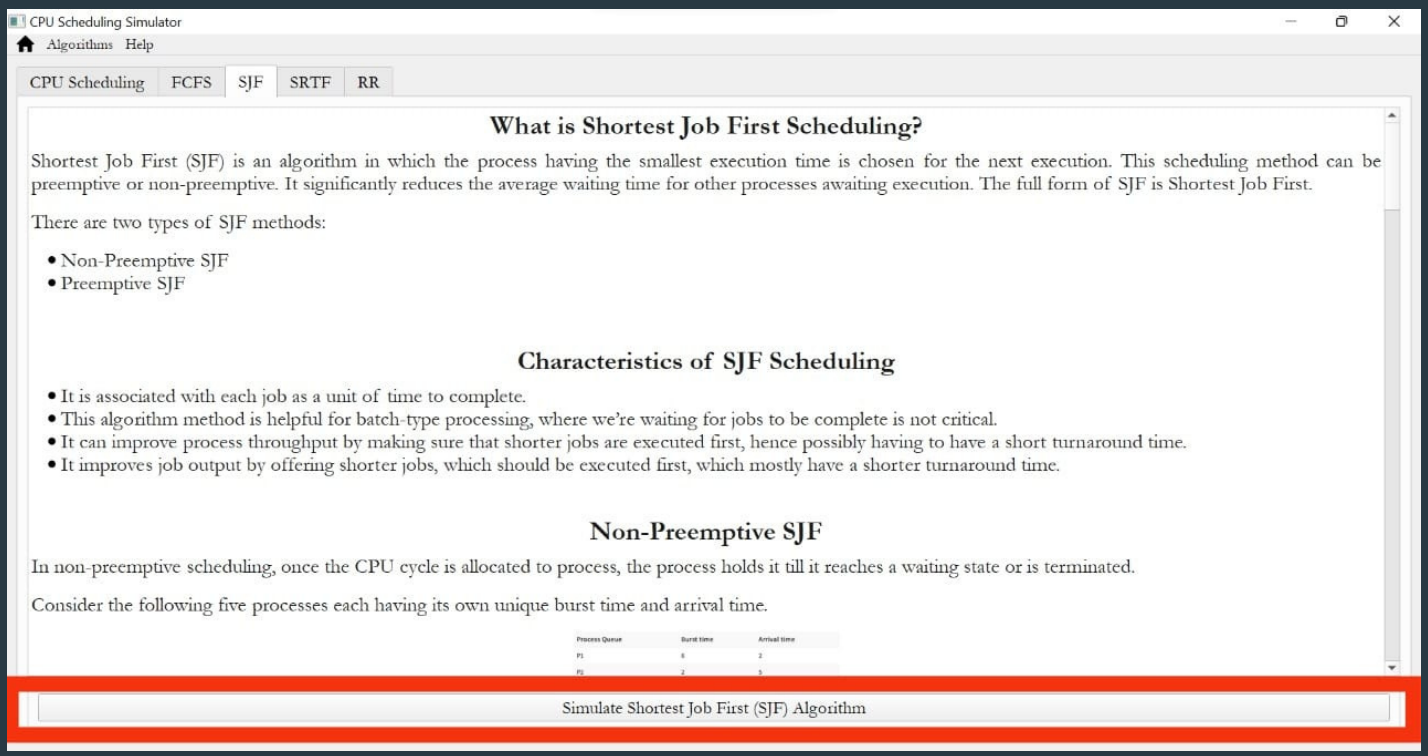
ALGORITHMS SIMULATOR



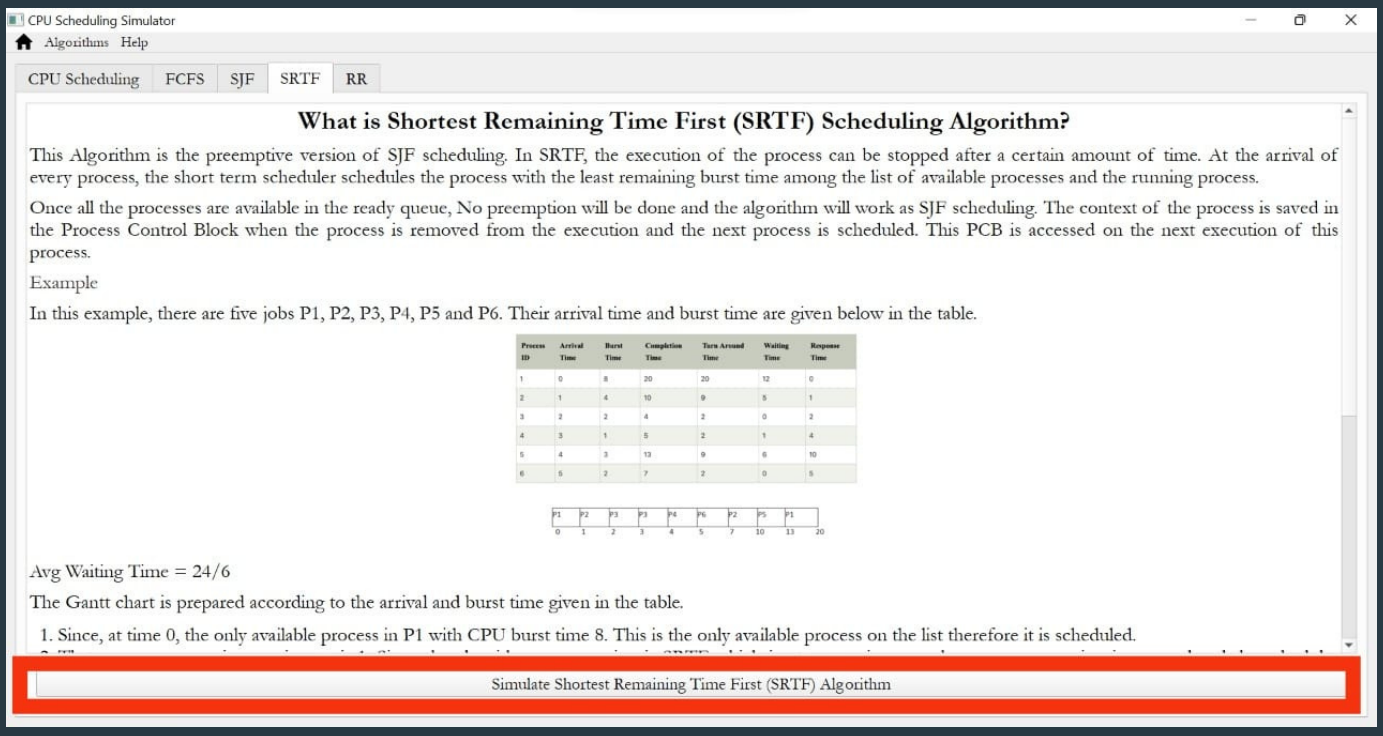
TABS TO GET SOME INFORMATION ABOUT TYPES OF CPU SCHEDULING ALGORITHMS



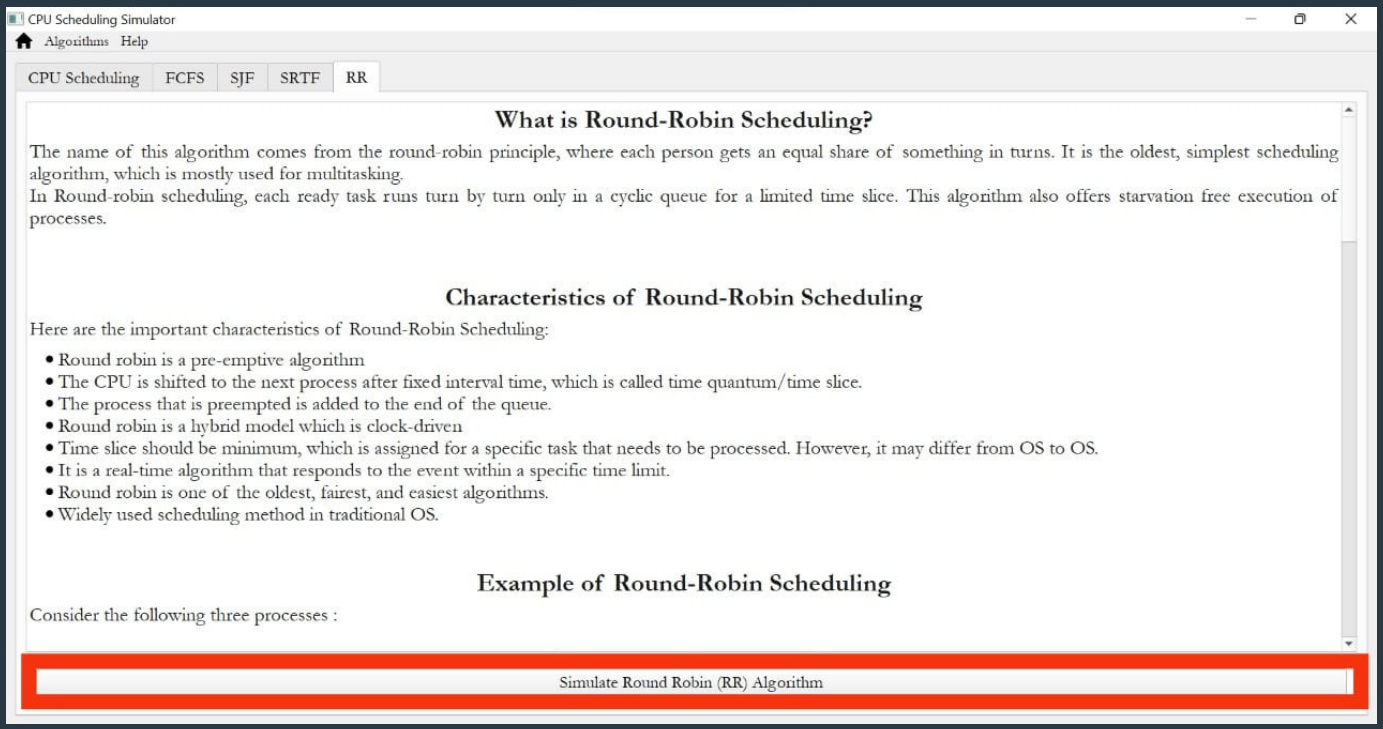
YOU CAN SIMULATE FCFS FROM HERE



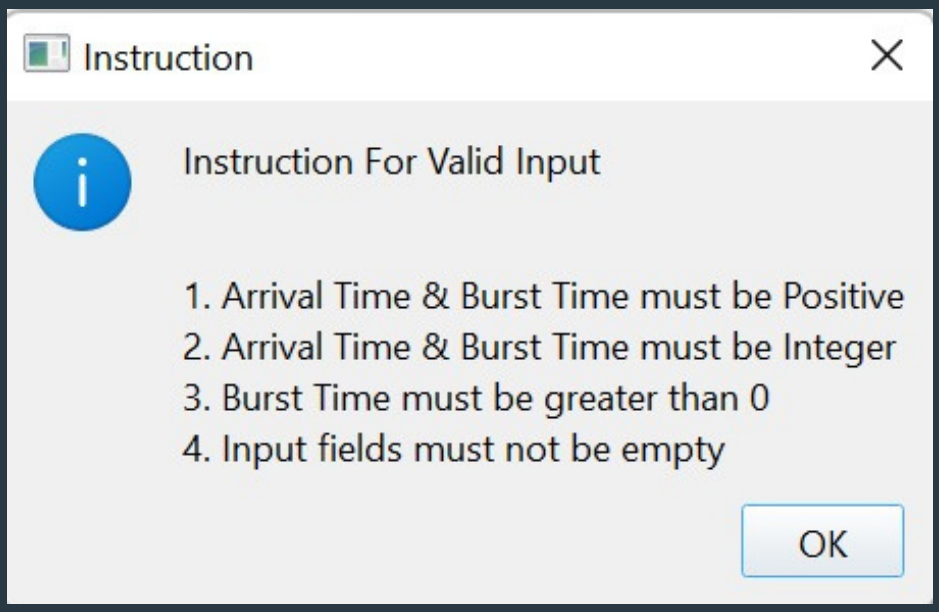
YOU CAN SIMULATE SJF FROM HERE



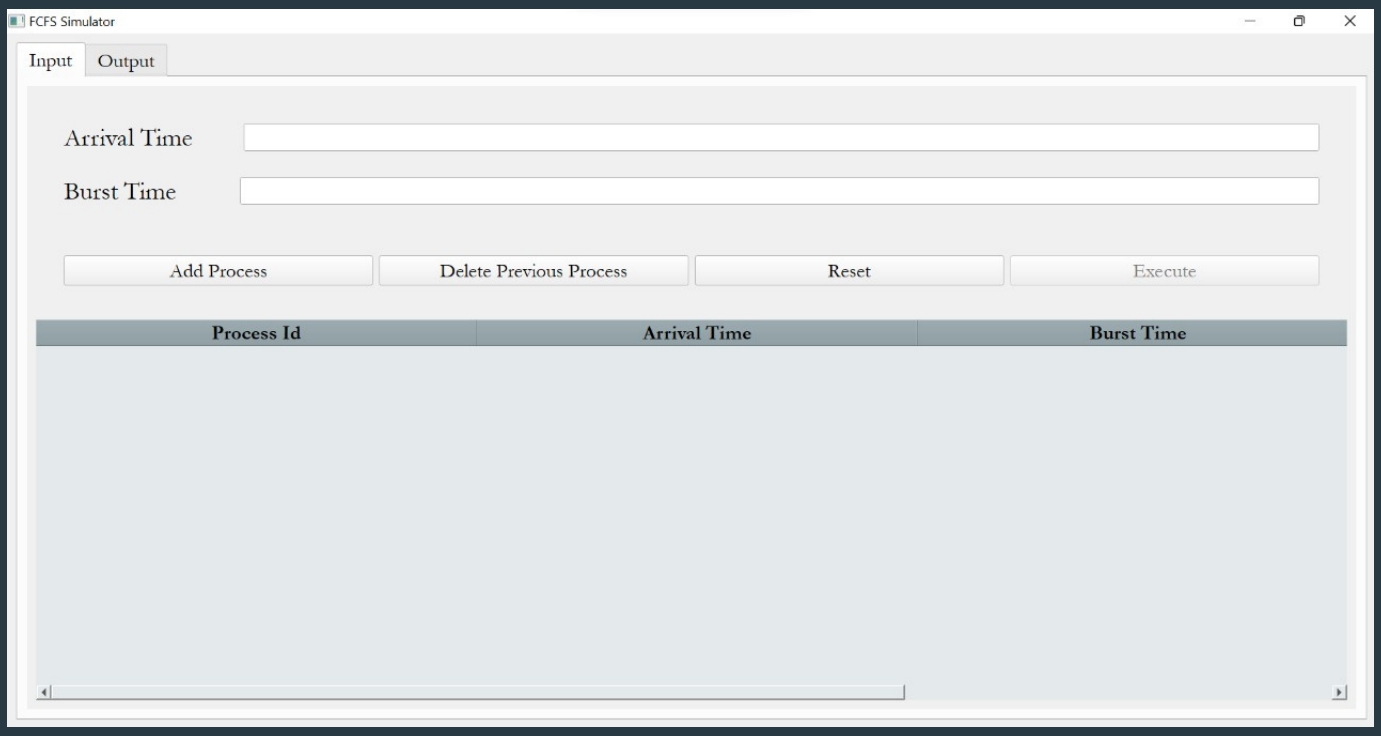
YOU CAN SIMULATE SRTF FROM HERE



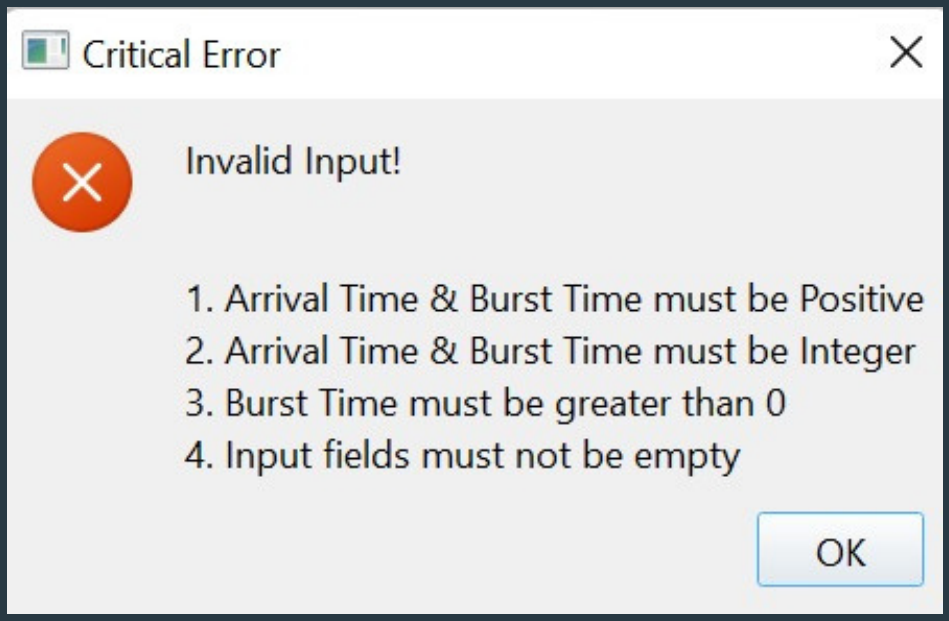
YOU CAN SIMULATE RR FROM HERE



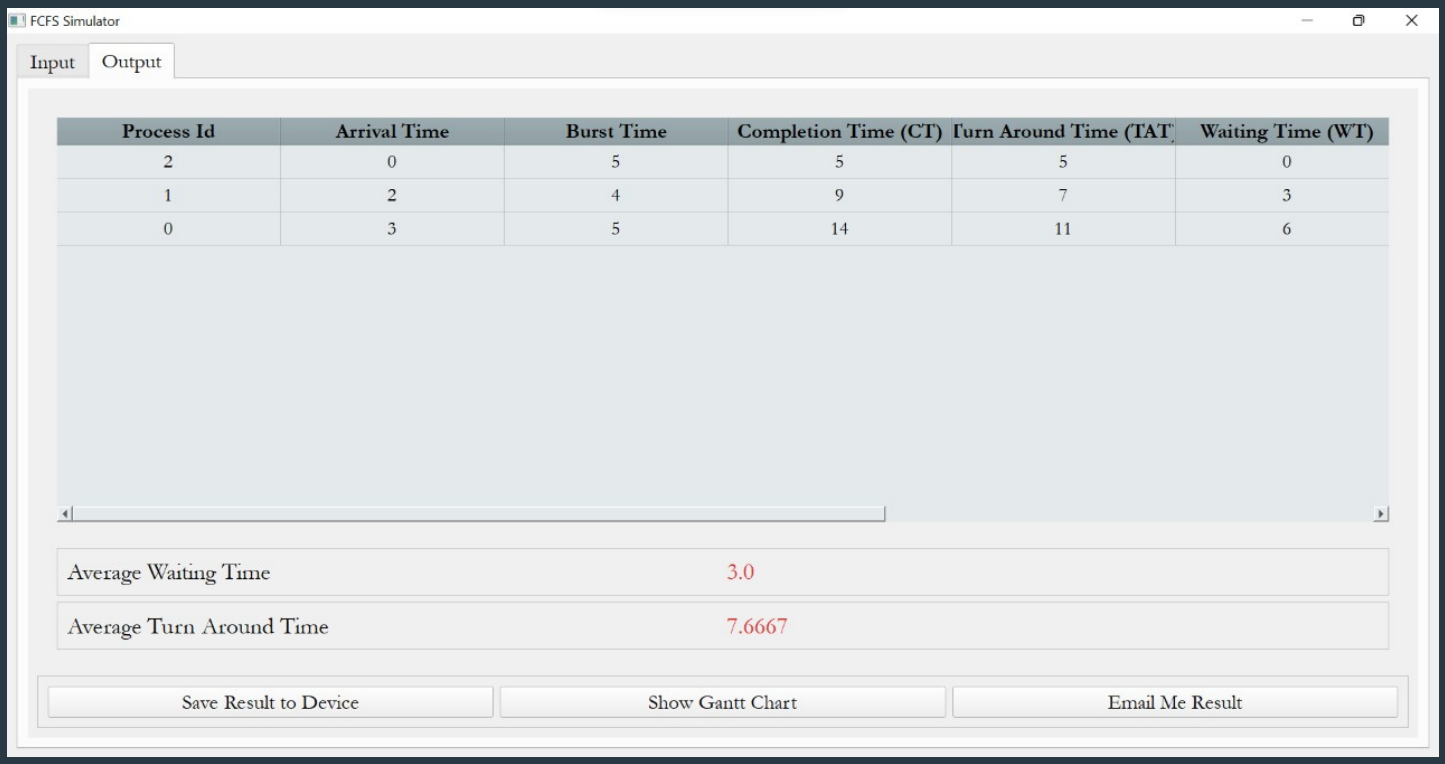
WHEN WE GO FOR SIMULATE SOME INSTRUCTIONS ABOUT VALID INPUT



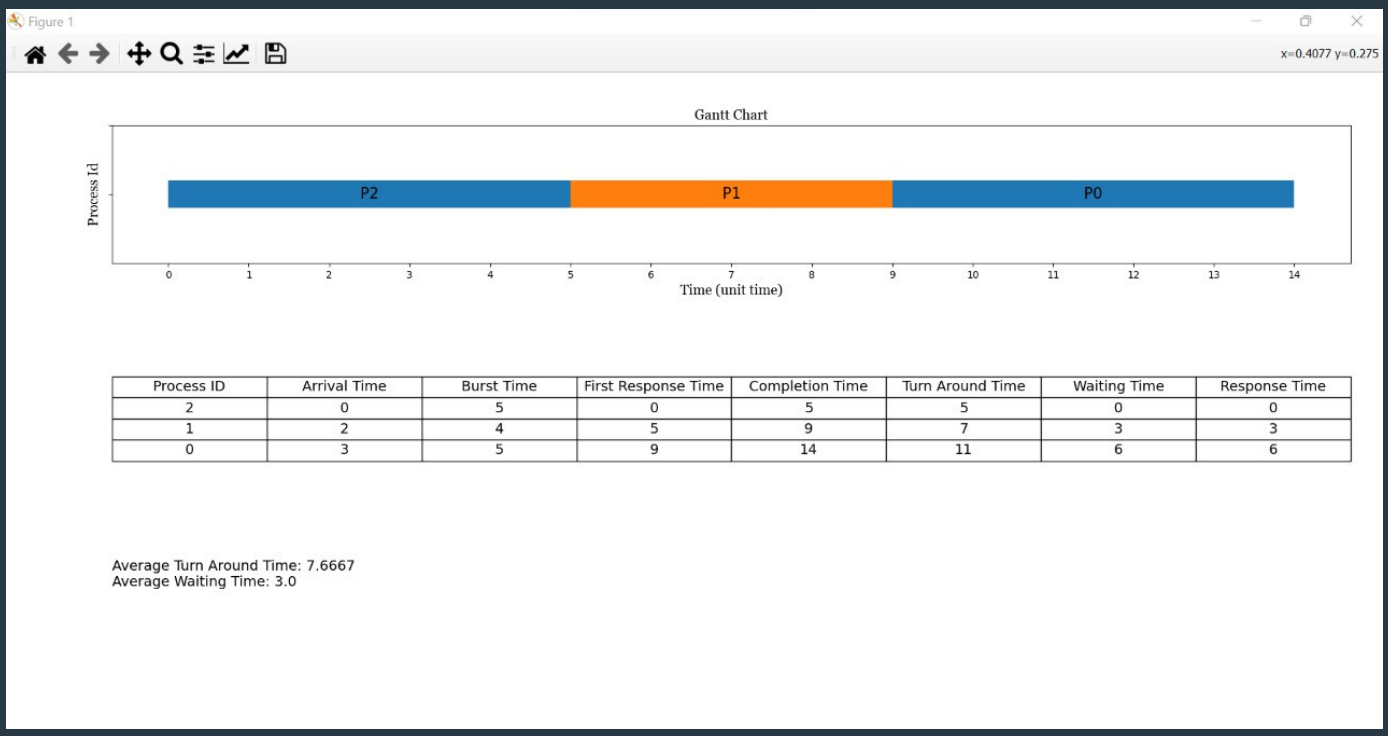
SIMULATOR LIKE THIS



IF YOU PUT INVALID INPUT YOU WILL GET THIS TYPE ERROR



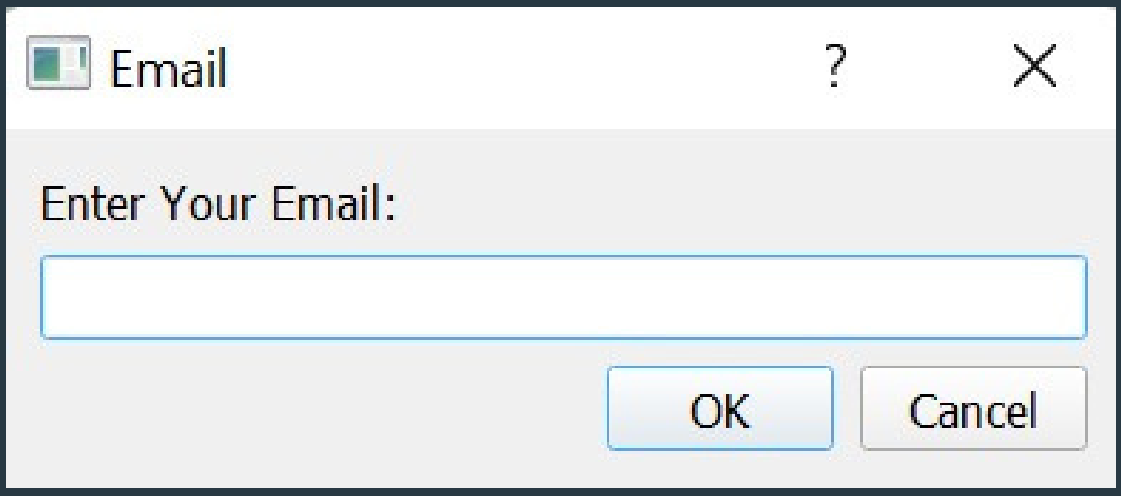
OUTPUT



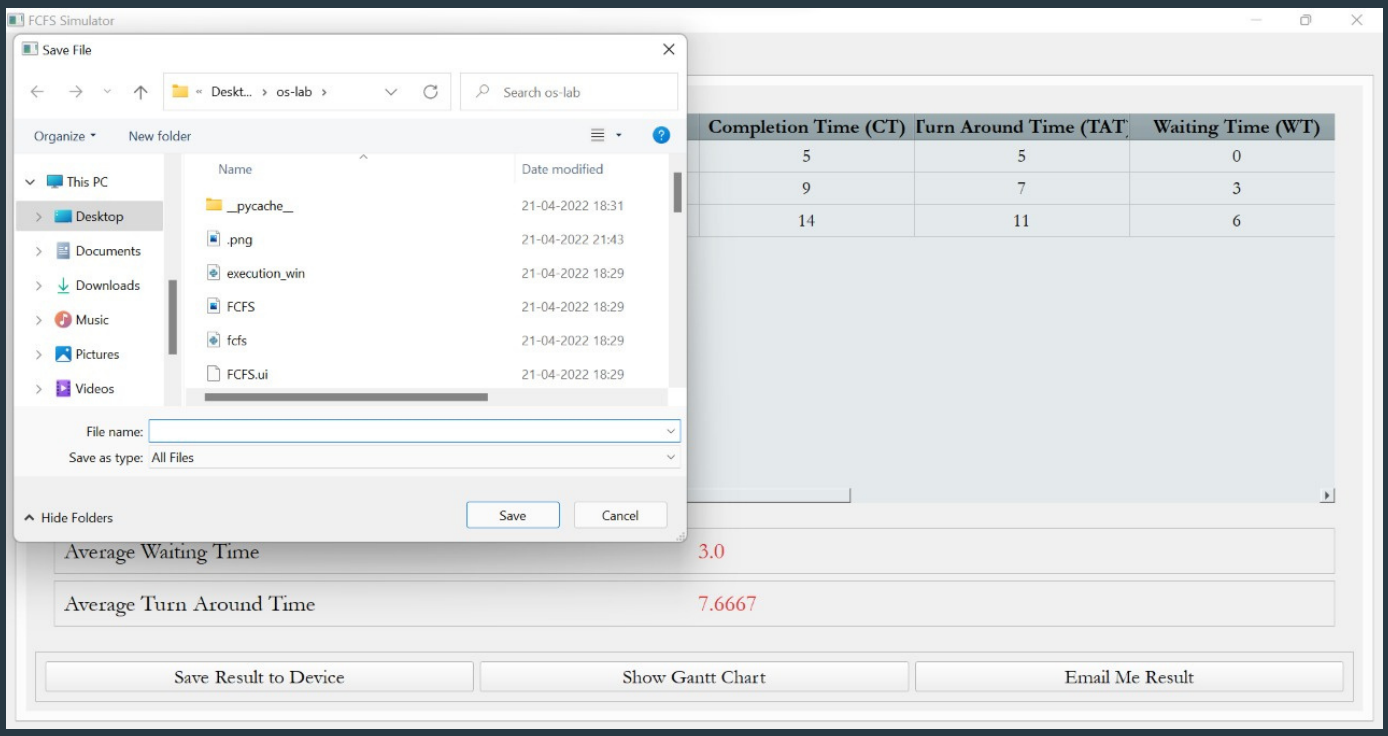
Process ID	Arrival Time	Burst Time	First Response Time	Completion Time	Turn Around Time	Waiting Time	Response Time
2	0	5	0	5	5	0	0
1	2	4	5	9	7	3	3
0	3	5	9	14	11	6	6

Average Turn Around Time: 7.6667
Average Waiting Time: 3.0

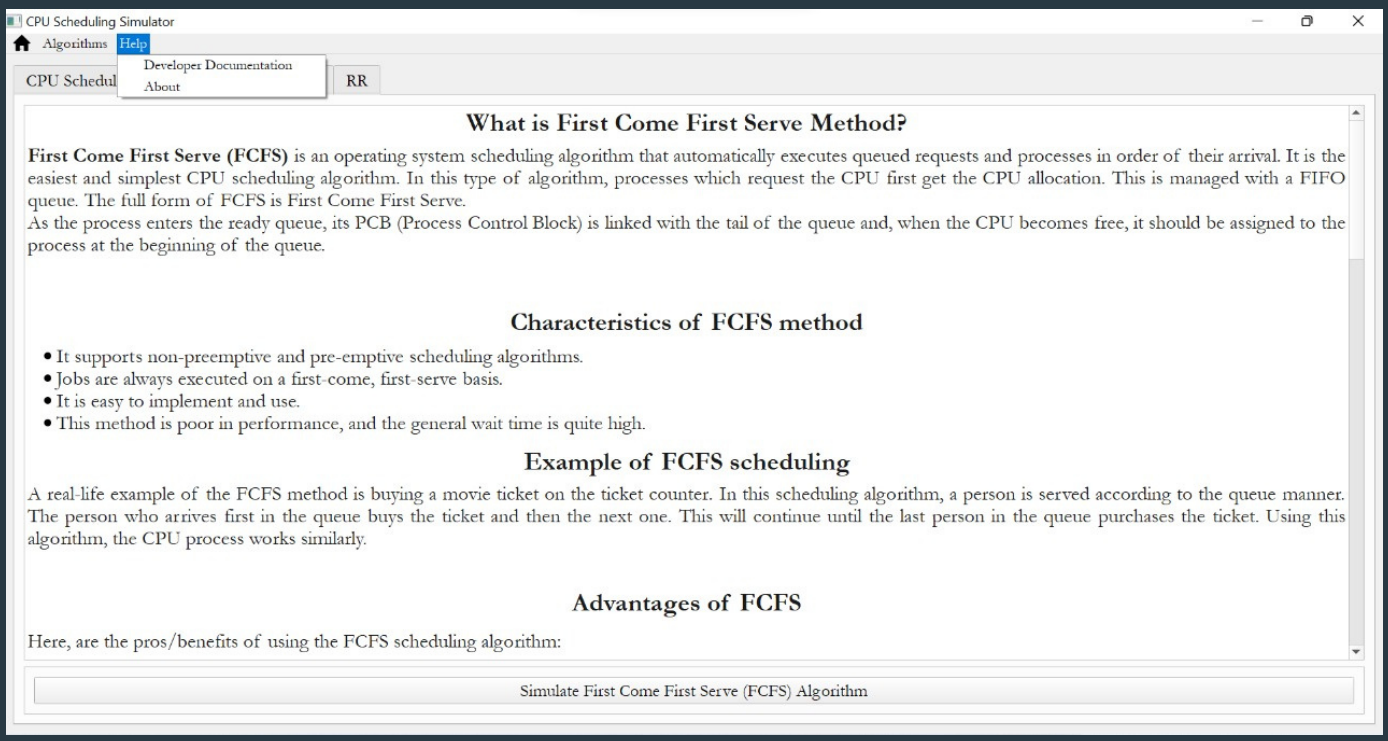
GANTT CHART OF ALGORITHM



IF YOU CLICK ON EMAIL ME RESULT



IF YOU CLICK ON SAVE RESULT TO DEVICE



IF YOU NEED SOME HELP FROM DOCUMENTS OR CONTACT WITH DEVELOPERS

Thank You

Contact Us

Hardik Inani	hardik.ice20@sot.pdpu.ac.in
Nilay Patel	nilay.pce20@sot.pdpu.ac.in
Rahul Gulati	rahul.gce20@sot.pdpu.ac.in
Dhvanil Bhagat	dhvanil.bce20@sot.pdpu.ac.in
Shrey Makadiya	shrey.mce20@sot.pdpu.ac.in
Drashti Bhavsar	drashti.bce20@sot.pdpu.ac.in