Operating System Project

instructor:

Dr. Ahmed Selim

TA: Eng. Gehad

Project Title:

OS Scheduler Simulation

Team Members

1. Mariam Mohamed - ID: 692300140
2. Maram Mohamed - ID: 692300130
3. Rawan Osama - ID: 692300455
4. Amira Saad - ID: 692300351

introduction:

< In this project, we simulated the role of the CPU scheduler in an operating system by designing and implementing a program that schedules processes using several well-known algorithms. We focused on performance analysis and comparing algorithms under the same conditions.

< The program was developed by using Python language and PyQt5 library

Scheduling Algorithms Implemented:

FCFS First Come First Serve Runs processes in order of arrival.
Round Robin (RR) Each process gets a fixed time slice (quantum).
(Non-Preemptive) Highest Priority First Chooses the process with the highest priority.
SRTF Shortest Remaining Time First Always selects the process with the least remaining time

{SRTF}performed best overall, with the lowest average waiting and turnaround times.
{HPF} came next, as it prioritizes shorter jobs but may cause starvation for low-priority processes
{RR}. showed fair scheduling, but its efficiency depends on the chosen time quantum.
{FCFS} had the highest waiting time, especially when early processes had long burst times

Operating System Project

Summary

This Python script simulates a process generator for an Operating System project.

It utilizes inter-process communication (IPC) with message queues, and signal handling to coordinate the timing and generation of processes.

Key components of the code:

- Reads process information from a file.
- Sends process data to a scheduler at specific time intervals.
- Handles SIGINT to clean up resources.
- Uses a forking method to launch a clock and scheduler program.

It's designed to simulate realistic behavior in a basic OS environment.

Code

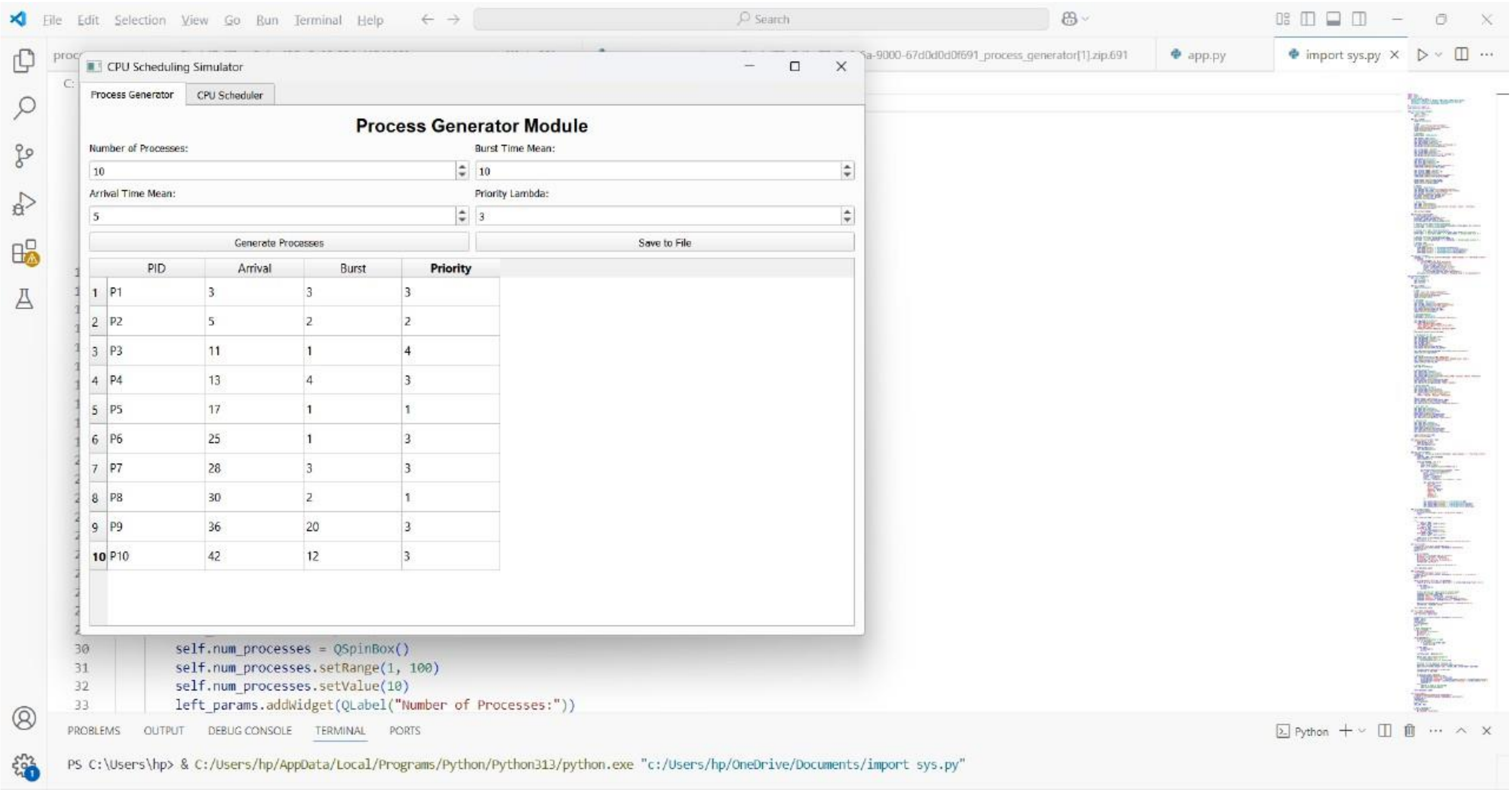
```
import sys
import random
import numpy as np
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel,
    QPushButton, QComboBox, QTableWidget, QTableWidgetItem, QSpinBox,
```

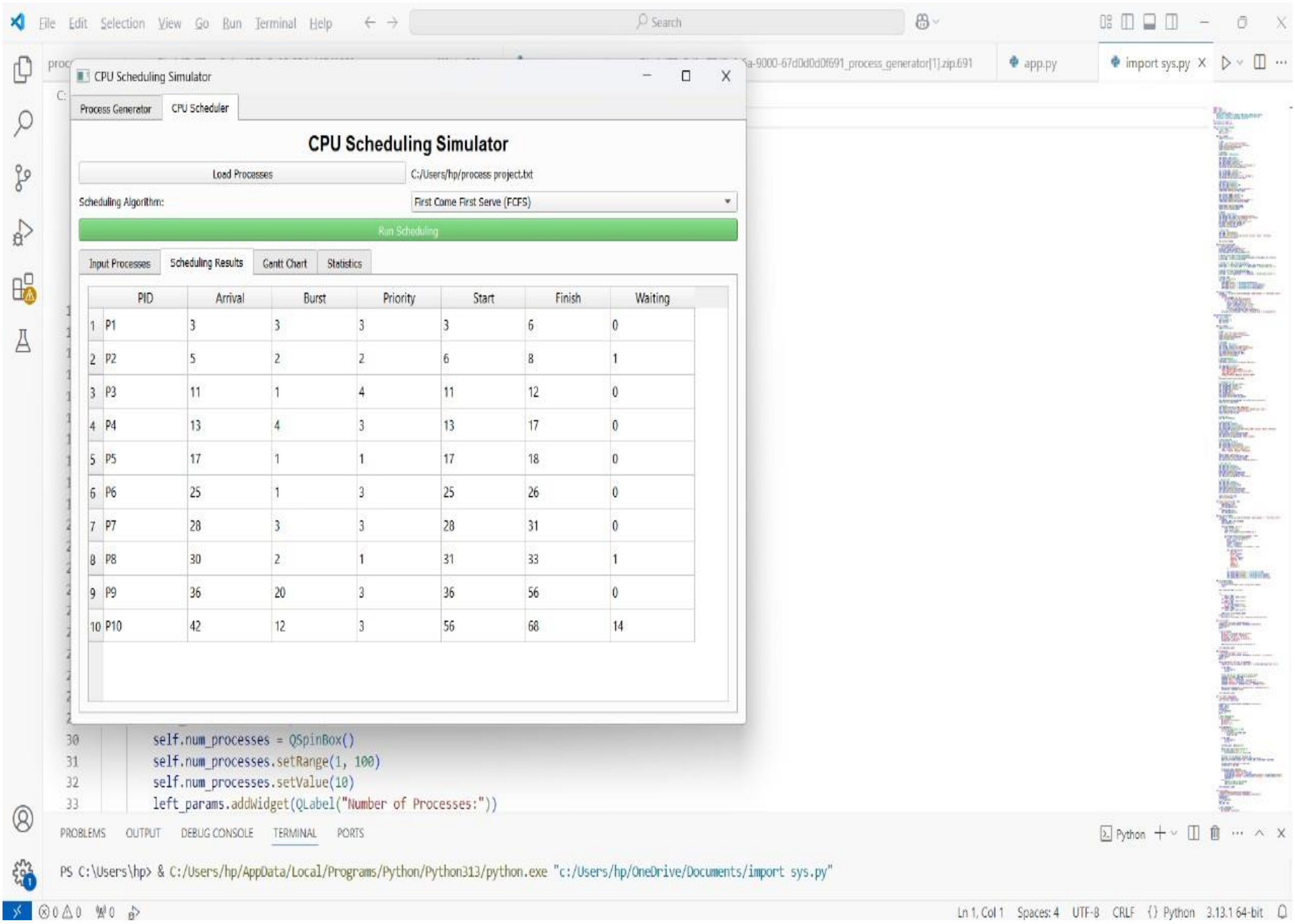
```
    QMessageBox, QFileDialog, QTabWidget, QTextEdit
)
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QFont

class ProcessGenerator(QWidget):
    def _init_(self):
        super()._init_()
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        # Title
        title = QLabel("Process Generator Module")
        title.setFont(QFont('Arial', 14, QFont.Bold))
        title.setAlignment(Qt.AlignCenter)
        layout.addWidget(title)

        # Parameters
        params_layout = QHBoxLayout()

        left_params = QVBoxLayout()
        self.num_processes = QSpinBox()
        self.num_processes.setRange(1, 100)
        self.num_processes.setValue(10)
        left_params.addWidget(QLabel("Number of Processes:"))
        left_params.addWidget(self.num_processes)

        self.arrival_mean = QSpinBox()
        self.arrival_mean.setRange(0, 100)
        self.arrival_mean.setValue(5)
        left_params.addWidget(QLabel("Arrival Time Mean:"))
        left_params.addWidget(self.arrival_mean)

        right_params = QVBoxLayout()
        self.burst_mean = QSpinBox()
        self.burst_mean.setRange(1, 100)
        self.burst_mean.setValue(10)
        right_params.addWidget(QLabel("Burst Time Mean:"))
        right_params.addWidget(self.burst_mean)

        self.priority_lambda = QSpinBox()
        self.priority_lambda.setRange(1, 10)
        self.priority_lambda.setValue(3)
        right_params.addWidget(QLabel("Priority Lambda:"))
        right_params.addWidget(self.priority_lambda)

        params_layout.addLayout(left_params)
```

```
        params_layout.addLayout(right_params)
        layout.addLayout(params_layout)

        # Buttons
        btn_layout = QHBoxLayout()
        self.generate_btn = QPushButton("Generate Processes")
        self.generate_btn.clicked.connect(self.generate_processes)
        self.save_btn = QPushButton("Save to File")
        self.save_btn.clicked.connect(self.save_to_file)
        btn_layout.addWidget(self.generate_btn)
        btn_layout.addWidget(self.save_btn)
        layout.addLayout(btn_layout)

        # Results Table
        self.table = QTableWidget()
        self.table.setColumnCount(4)
        self.table.setHorizontalHeaderLabels(["PID", "Arrival", "Burst", "Priority"])
        layout.addWidget(self.table)

        self.setLayout(layout)

    def generate_processes(self):
        n = self.num_processes.value()
        arrival_mean = self.arrival_mean.value()
        burst_mean = self.burst_mean.value()
        priority_lambda = self.priority_lambda.value()

        # Generate arrival times (normal distribution)
        arrival_times = np.abs(np.random.normal(arrival_mean, arrival_mean/2, n)).astype(int)
        arrival_times = np.cumsum(arrival_times)

        # Generate burst times (normal distribution)
        burst_times = np.abs(np.random.normal(burst_mean, burst_mean/2, n)).astype(int)
        burst_times = np.where(burst_times < 1, 1, burst_times)

        # Generate priorities (Poisson distribution)
        priorities = np.random.poisson(priority_lambda, n)
        priorities = np.where(priorities < 1, 1, priorities)

        # Populate table
        self.table.setRowCount(n)
        for i in range(n):
            self.table.setItem(i, 0, QTableWidgetItem(f"P{i+1}"))
            self.table.setItem(i, 1, QTableWidgetItem(str(arrival_times[i])))
            self.table.setItem(i, 2, QTableWidgetItem(str(burst_times[i])))
            self.table.setItem(i, 3, QTableWidgetItem(str(priorities[i])))

    def save_to_file(self):
             filename, _ = QFileDialog.getSaveFileName(self, "Save Processes", "", "Text Files
```

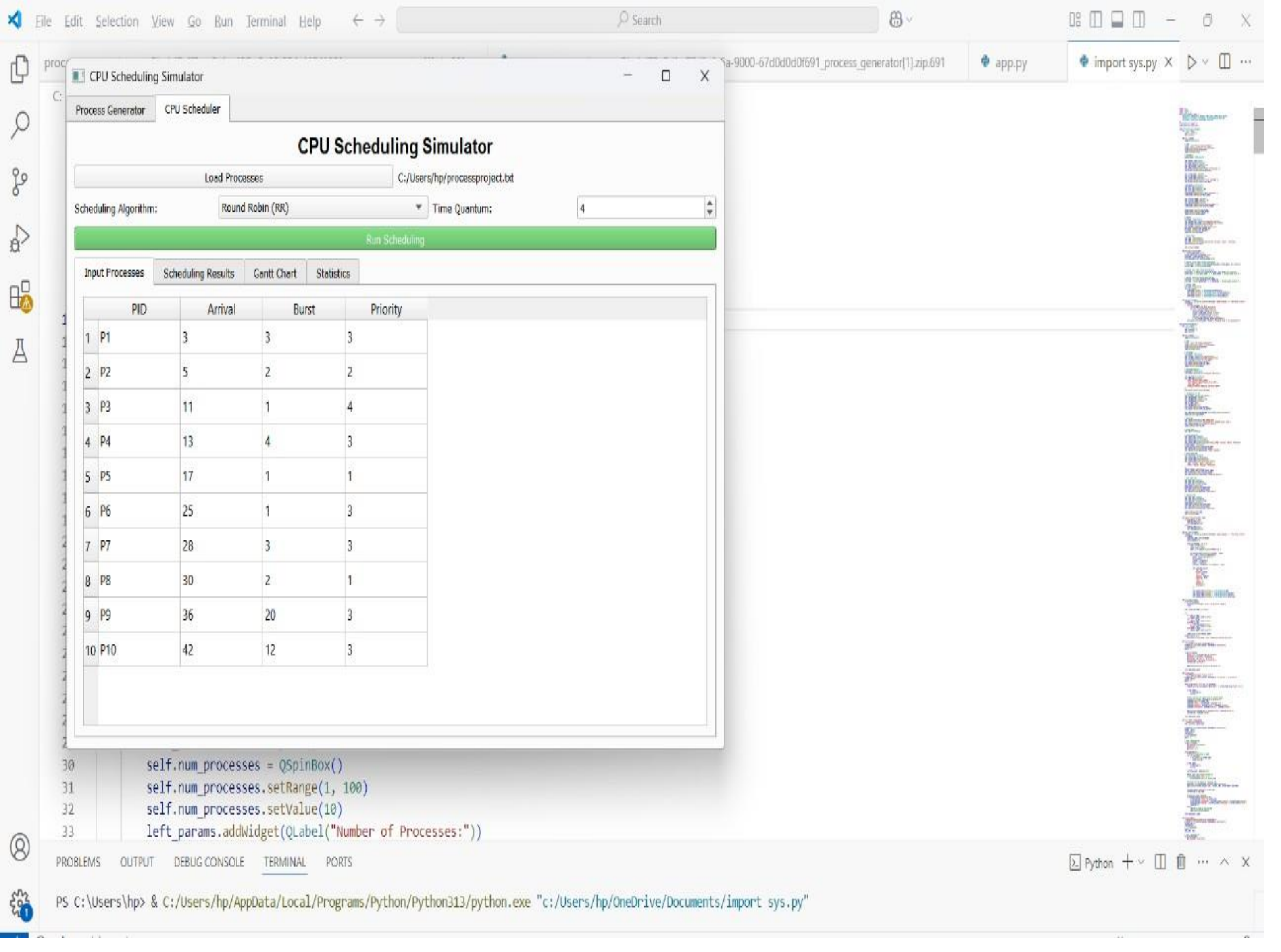
```
(*.txt)")
        if filename:
            with open(filename, 'w') as f:
                f.write("PID,Arrival,Burst,Priority\n")
                for row in range(self.table.rowCount()):
                    pid = self.table.item(row, 0).text()
                    arrival = self.table.item(row, 1).text()
                    burst = self.table.item(row, 2).text()
                    priority = self.table.item(row, 3).text()
                    f.write(f"{pid},{arrival},{burst},{priority}\n")
            QMessageBox.information(self, "Success", "Processes saved to file successfully!")
```

Operating System Project

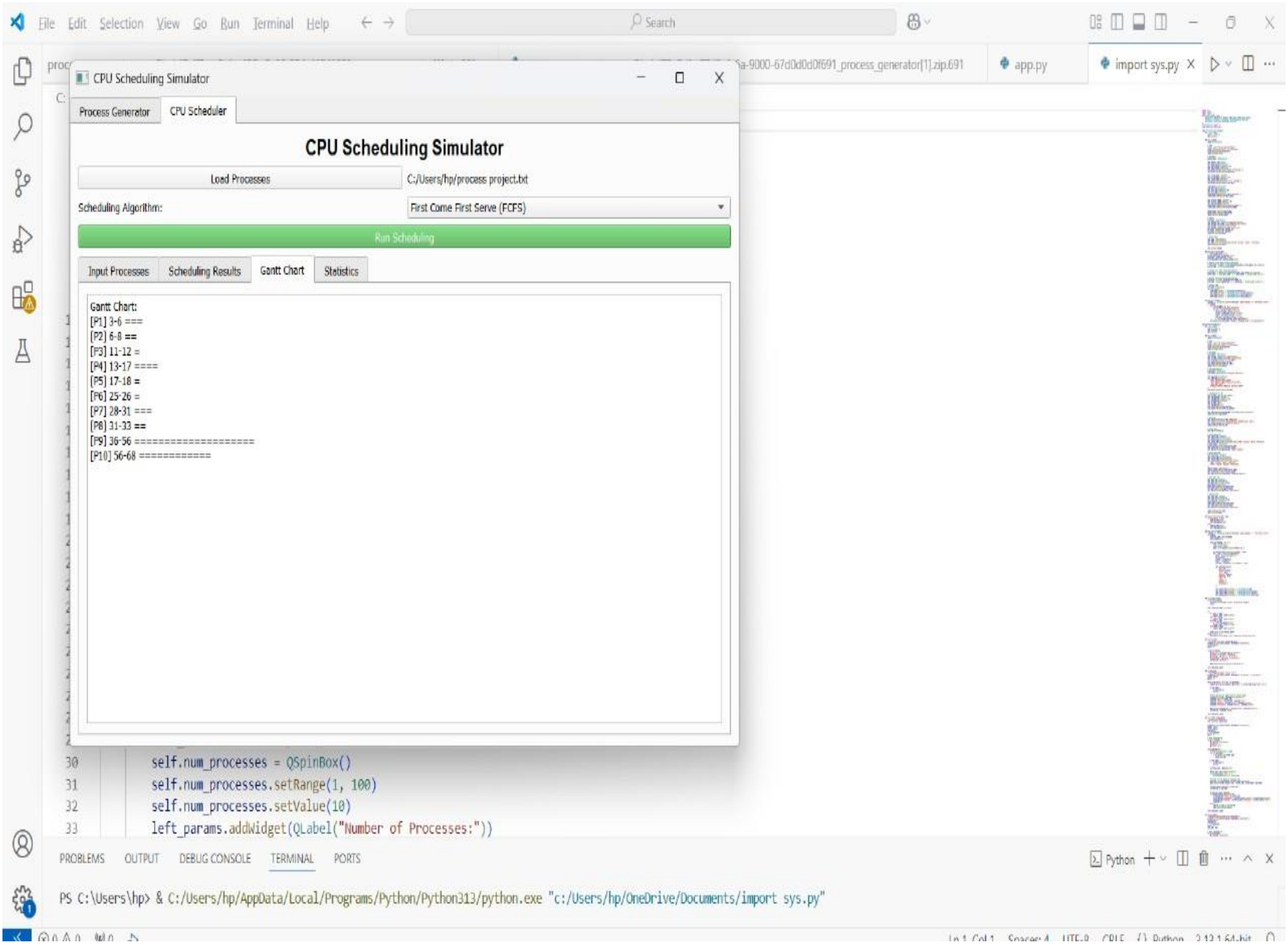




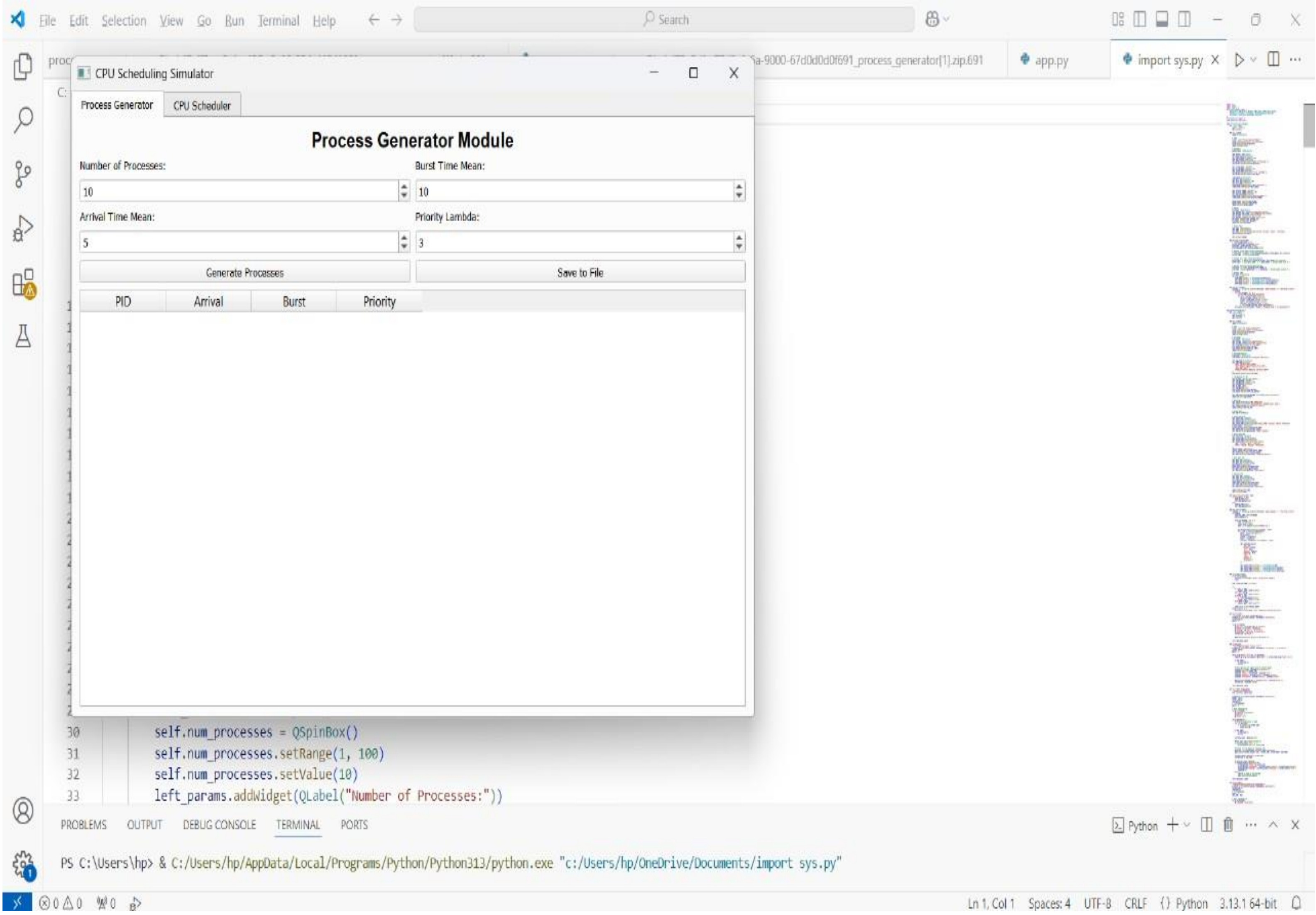
Operating System Project



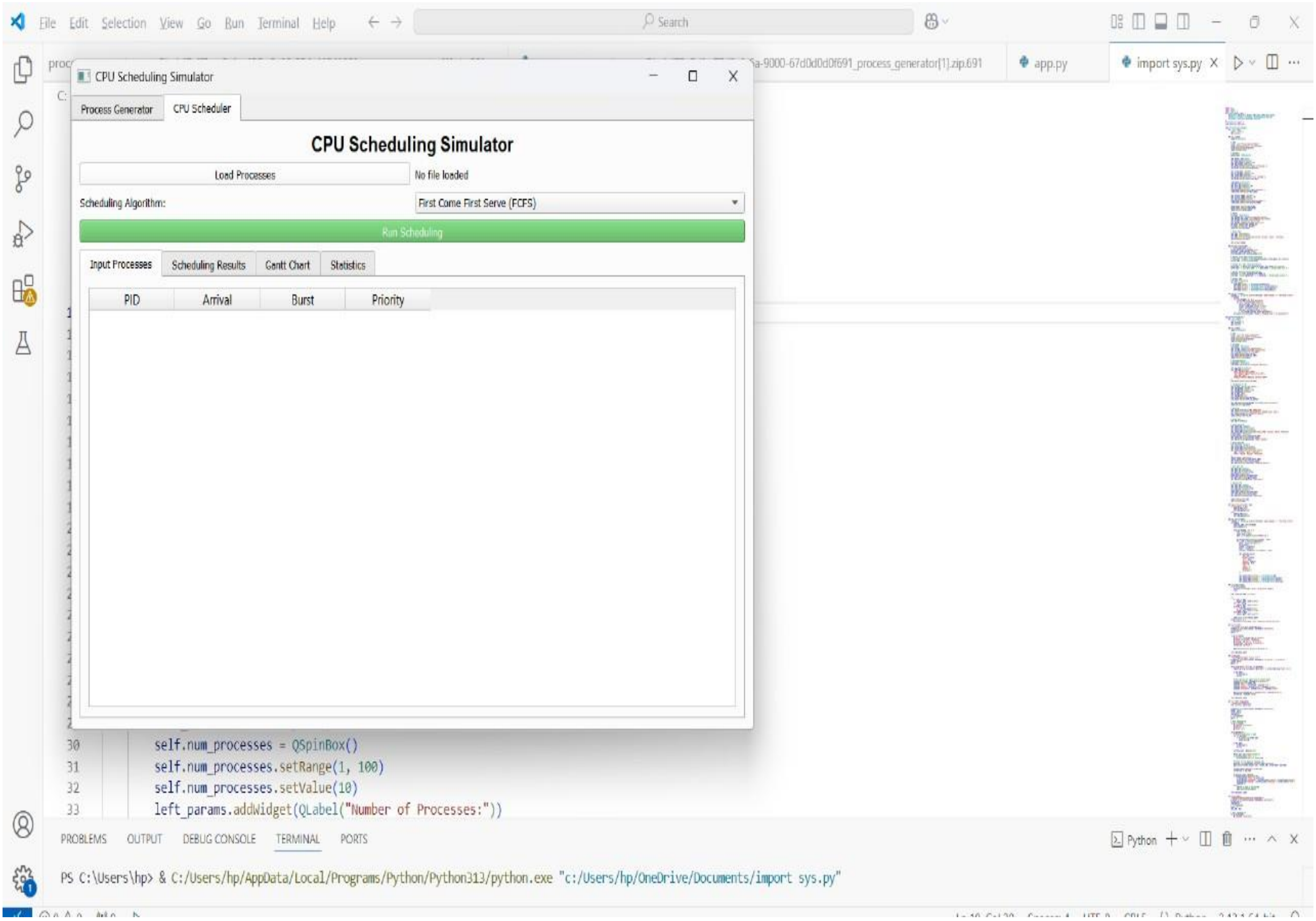
Operating System Project



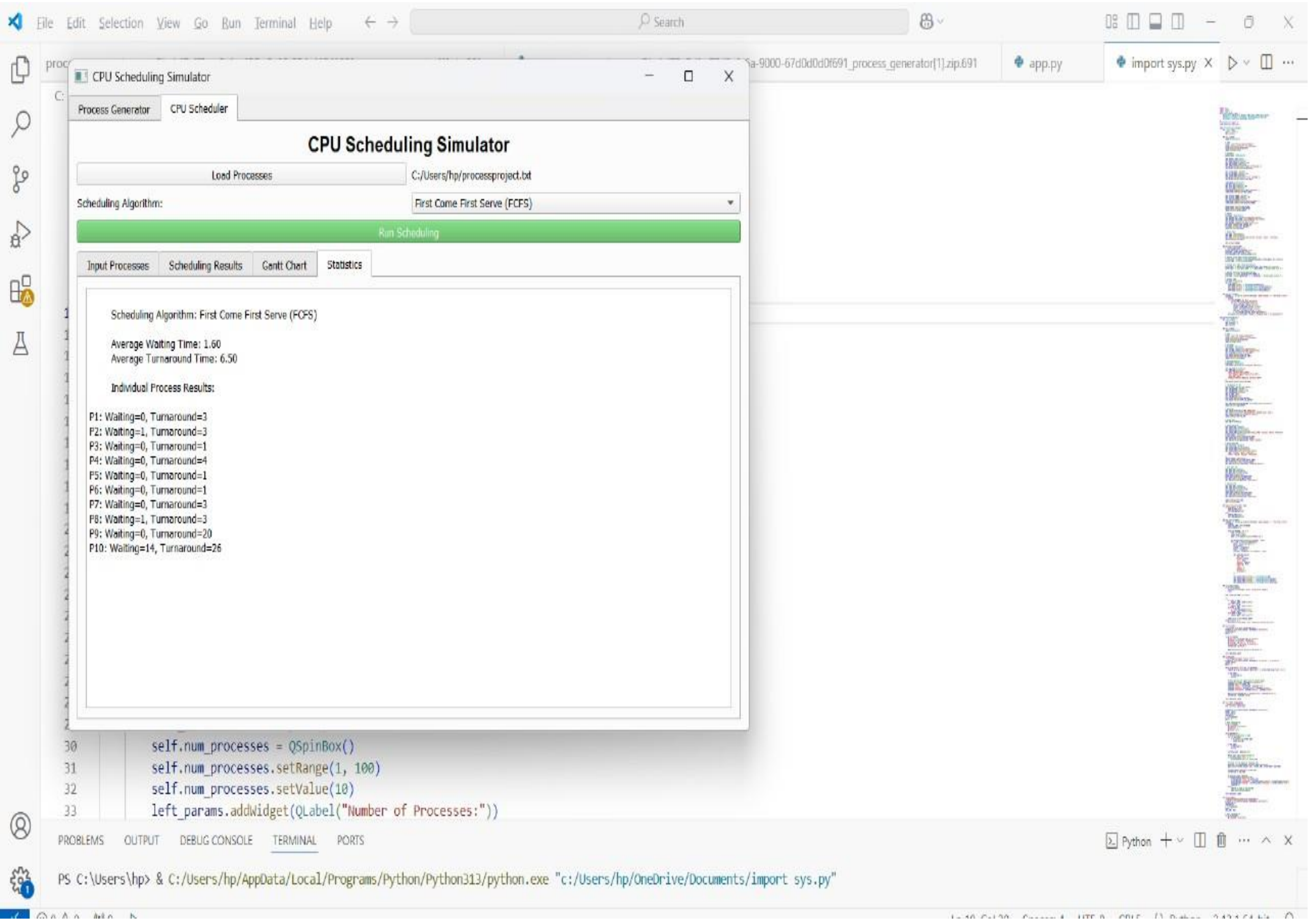
Operating System Project



Operating System Project



Operating System Project



Conclusions

We successfully implemented and compared four scheduling algorithms. Each algorithm has strengths and weaknesses depending on the scenario. We learned how scheduling affects performance and user experience

References:

1. Course Lectures and Notes, Operating Systems, Spring 2025.
2. Project Specifications PDF, OS Scheduler.

Thanks