Report for Program 1 - Simulation

Course: Simulation(20564)

Keivan Jamali 403209743
Professor Shafahi

March 12, 2025

Contents

1 Introduction **3**
1.1 Background . . . 3
1.2 Objective . . . 3

2 Methodology **4**
2.1 System Design . . . 4
2.2 Simulation Process . . . 4
2.3 Performance Metrics . . . 5

3 Implementation **5**
3.1 Programming Language and Tools . . . 5
3.2 Classes and Methods . . . 5
3.3 Event Handling and Scheduling . . . 6
3.4 Policy Handling . . . 6
3.5 Performance Monitoring . . . 7

4 Simulation Results **8**
4.1 Small example . . . 9
4.1.1 7 Customer get service . . . 9
4.1.2 45 min of simulating . . . 15
4.2 Test Cases . . . 21
4.3 Simulation . . . 21
4.3.1 10,000 customer threshold - M/M/1 and M/D/1 . . . 21
4.3.2 1000 hours threshold - M/M/1 and M/D/1 . . . 23

5 Conclusion **24**

List of Figures

1 Handwritten: solution FCFS|MM1|7 . . . 9
2 Code: Sample Path for FCFS|MM1|7 . . . 10
3 Code: FCFS|MM1|7 . . . 10
4 Handwritten: solution SPT|MM1|7 . . . 11
5 Sample Path for SPT|MM1|7 . . . 12
6 Code: SPT|MM1|7 . . . 12
7 Handwritten: solution FCFS|MD1|7 . . . 13
8 Sample Path for FCFS|MD1|7 . . . 14
9 Code: FCFS|MD1|7 . . . 14
10 Handwritten: solution FCFS|MM1|45 . . . 15
11 Sample Path for FCFS|MM1|45 . . . 16
12 Code: FCFS|MM1|45 . . . 16

13 Handwritten: solution SPT|MM1|45 17
14 Sample Path for SPT|MM1|45 18
15 Code: SPT|MM1|45 18
16 Handwritten: solution FCFS|MD1|45 19
17 Handwritten: solution FCFS|MD1|45 19
18 Sample Path for FCFS|MD1|45 20
19 Code: FCFS|MD1|45 20
20 Test Cases 21
21 Test case results 22

1 Introduction

1.1 Background

Queueing systems are essential in modeling real-world systems where entities (often referred to as customers) wait for service from a server. Such systems are ubiquitous, occurring in fields ranging from telecommunications, computer networks, and manufacturing, to customer service and health care. A queueing system typically consists of entities waiting in line for service, and the server responsible for providing that service. The efficiency of a queueing system can be heavily influenced by how customers are processed, and thus, understanding and optimizing the policy used to manage the queue is crucial.

The design of queueing policies determines the system's performance in terms of factors such as wait times, server utilization, and throughput. Two common queueing policies are **First-Come, First-Served (FCFS)** and **Shortest Processing Time (SPT)**. In this report, we explore these two policies by simulating a queueing system and comparing their performances.

1.2 Objective

The primary objective of this report is to simulate a queueing system using two different policies—FCFS and SPT—and analyze their performance in terms of several key metrics. These include the average waiting time in the queue, server utilization, and the overall number of people served.

2 Methodology

2.1 System Design

The system being modeled in this report simulates a queue with a single server where customers arrive randomly, and their service times are also randomly distributed. The system follows a discrete event simulation model, where the main events in the system are the arrival of a customer and the departure of a customer after their service is completed.

The simulation uses two different policies for handling customer arrivals and departures:

- **First-Come, First-Served (FCFS)**: Customers are served in the order of their arrival. The first customer to arrive is the first to be served.
- **Shortest Processing Time (SPT)**: The customer with the shortest processing time (service time) is prioritized and served next, regardless of their arrival time.

2.2 Simulation Process

The simulation follows the steps outlined below:

1. **Initialization**: At the start of the simulation, the system is initialized with a generator function that will produce a number randomly when we call it. The arrival and service times are generated randomly using an exponential distribution.
2. **Event Handling**: The main events (arrivals and departures) and additional events (departure from queue and the time service will start) are handled based on the current policy (FCFS or SPT). There is some functions to decide on events at any step in the code (There is no event-list, events will recognize through the process steps.)
3. **Queue Management**: When a customer arrives, they are added to the queue. If the server is available, the customer starts service immediately; otherwise, they wait in the queue. All the statistical parameters get updates after each step.
4. **Simulation Termination**: The simulation terminates after a predefined number of customers have been processed, or when the system reaches a predefined time.

2.3 Performance Metrics

The performance of the system is evaluated based on several key metrics:

- **Average Waiting Time (Q-bar)**: The average time that customers spend waiting in the queue before being served.
- **Server Utilization (U)**: The proportion of time the server is actively serving customers.
- **Number of People in Queue (N-Q)**: The average number of customers in the queue over the course of the simulation.
- **Percentage of people being in system longer that 4.5 min**: The percentage of people that had a longer experience of being in the system.

The simulation collects these metrics over time and uses them to evaluate the performance of the queueing policies under various conditions.

3 Implementation

3.1 Programming Language and Tools

The simulation was implemented in Python using the following libraries:

- **Matplotlib**: Used for plotting the sample path of the system, visualizing the number of people in the system over time.

3.2 Classes and Methods

The simulation is implemented using object-oriented programming. The key classes in the simulation are:

- **Queue Class**: Manages the queue of waiting customers. Includes methods for adding customers to the queue and processing departures. It also update the time in queue parameter in each event. This class can store all the historical data too.
- **Server Class**: Represents the server that processes the customers. Includes methods for checking if the server is occupied and for marking the server as unoccupied. It also updates total time for the occupied and unoccupied condition for the server.

- **System Class**: Manages the overall state of the system, including arrivals, departures, and the calculation of performance metrics. It also interacts with the queue and server. This class will update the number of people who stayed in the system more that 4.5 min.
- **Simulation Class**: The main class that runs the simulation, schedules events, and collects statistics.

3.3 Event Handling and Scheduling

The simulation uses an event-driven approach, where each event (arrival or departure) triggers specific actions. Events are scheduled using a priority queue, with each event associated with a timestamp. The key steps in event handling are:

- **Initializing**: When we start the system, we add a person to the system and queue class. Then immediately depart him from queue and send him to the server (The server become occupied).
- **Arrival Event**: When a customer arrives, the customer waits in the queue. In this condition, we will add him to the queue and system class.
- **Departure Event**: When a customer's service is complete, the server serves the next customer according to the policy (FCFS or SPT).
- **Special case**: If we depart a customer and the server become unoccupied, we will start the program again. (starts by initializing and continue).

3.4 Policy Handling

The simulation supports two different service policies:

- **First-Come, First-Served (FCFS)**: The server serves customers in the order of their arrival. The simulation handles arrivals by simply adding the customer to the queue and serving the next customer when the server is free.
- **Shortest Processing Time (SPT)**: The server serves the customer with the shortest service time next, regardless of their arrival order. The simulation sorts the queue by service time to ensure the correct customer is served.

The choice of policy affects how the system behaves, particularly in terms of customer waiting times and server utilization.

3.5 Performance Monitoring

During the simulation, performance metrics are monitored and updated after each event. Key metrics include the total waiting time in the queue, the server's utilization, and the total number of people in the system with more than 4.5 min. These metrics are used to answer the questions in the assignment.

4 Simulation Results

After running the simulation under both the First-Come, First-Served (FCFS) and Shortest Processing Time (SPT) policies, various performance metrics were collected. These include the average number of people in the queue, the average waiting time in the queue, the server's utilization, and percentage of people who had been in the system more than 4.5 minutes.

4.1 Small example

4.1.1 7 Customer get service

Input data is:

$$\text{Intervals}: [1.175, 0.373, 5.669, 0.848, 11.253, 6.172, 2.515]$$

$$\text{Service times}: [8.648, 3.615, 0.242, 8.244, 0.760, 1.931, 1.263]$$

Solution to FCFS | M/M/1 | 7:

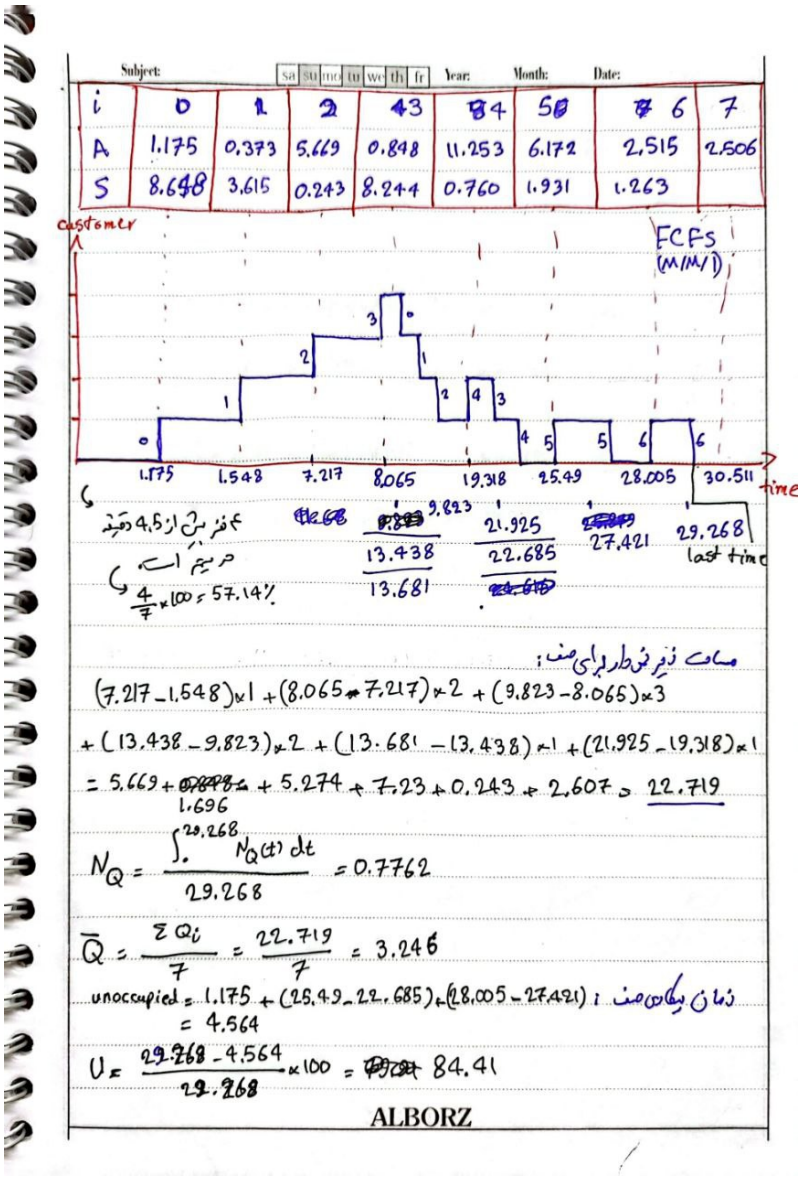


Figure 1: Handwritten: solution FCFS|MM1|7

- FCFS Policy MM1:
  - The average waiting time for people in the queue was 3.25 minutes.
  - The average number of people in the queue was 0.78.
  - 57.14% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 84.41% of the total simulation time.

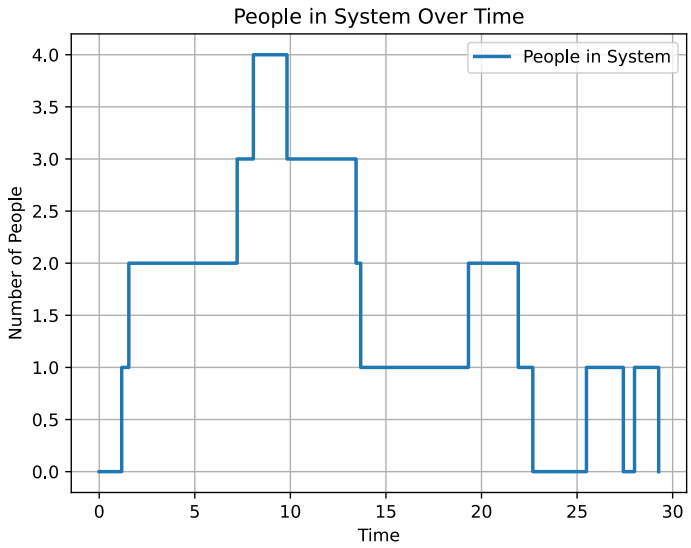


Figure 2: Code: Sample Path for FCFS|MM1|7

```
from modular.main import main

model = main(policy="FCFS",
             stop_type="Customer",
             stop_limit=7,
             queue_type="MMI",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 0.0s

```
==================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 3.25
Therefore, Average number of People in Queue (N_Q) will be 0.78
We have 4 people with more that 4.5 min system time. Therefore the percentage is 57.14
Therefore the U=84.41

==================================================
```

Figure 3: Code: FCFS|MM1|7

Solution to SPT | M/M/1 | 7:

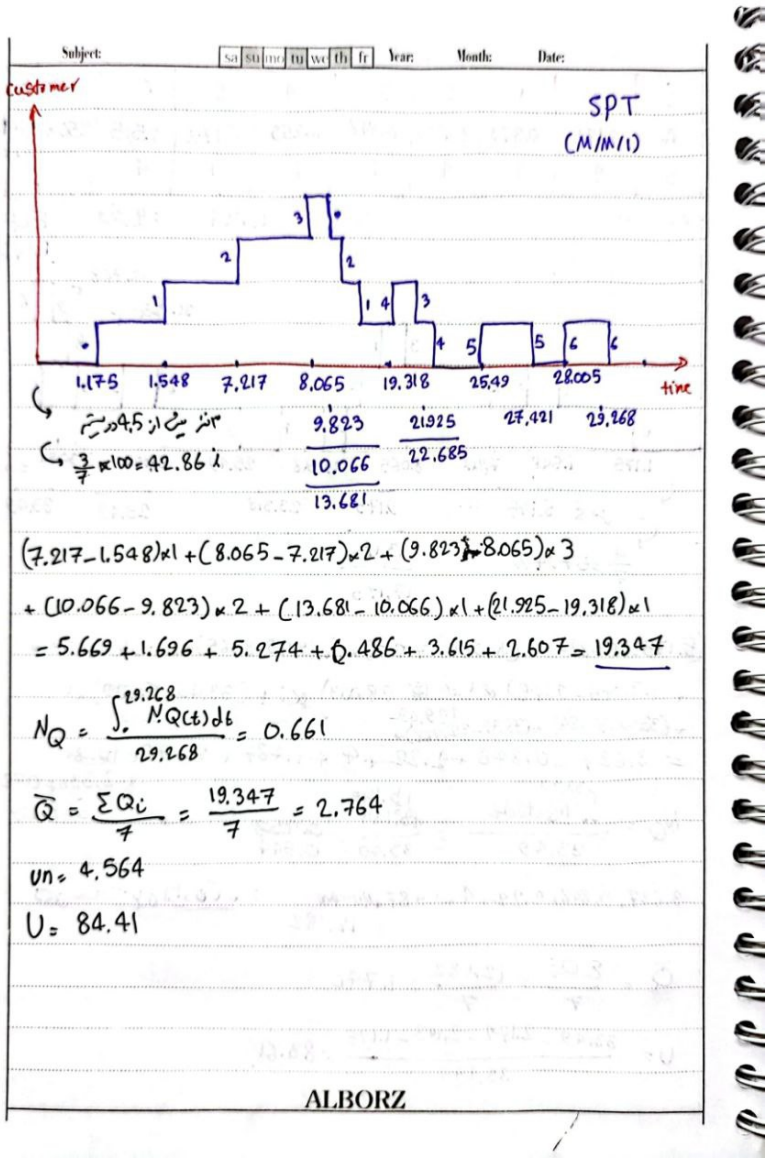


Figure 4: Handwritten: solution SPT|MM1|7

- SPT Policy MM1:
  - The average waiting time for people in the queue was 2.76 minutes.
  - The average number of people in the queue was 0.66.
  - 42.86% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 84.41% of the total simulation time.

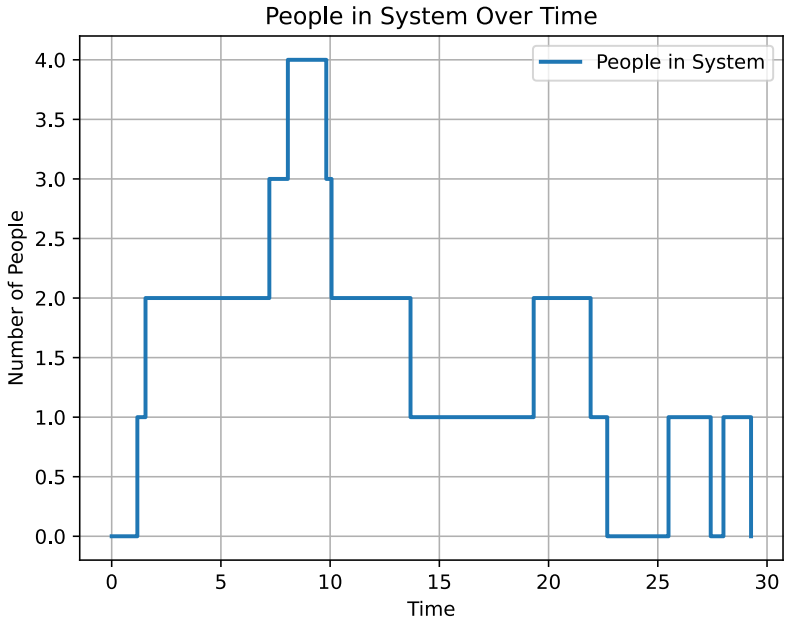


Figure 5: Sample Path for SPT|MM1|7

```
from modular.main import main

model = main(policy="SPT",
             stop_type="Customer",
             stop_limit=7,
             queue_type="MMI",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 0.0s

```
==================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 2.76
Therefore, Average number of People in Queue (N_Q) will be 0.66
We have 3 people with more that 4.5 min system time. Therefore the percentage is 42.86
Therefore the U=84.41

==================================================
```

Figure 6: Code: SPT|MM1|7

Solution to FCFS | M/D/1 | 7:

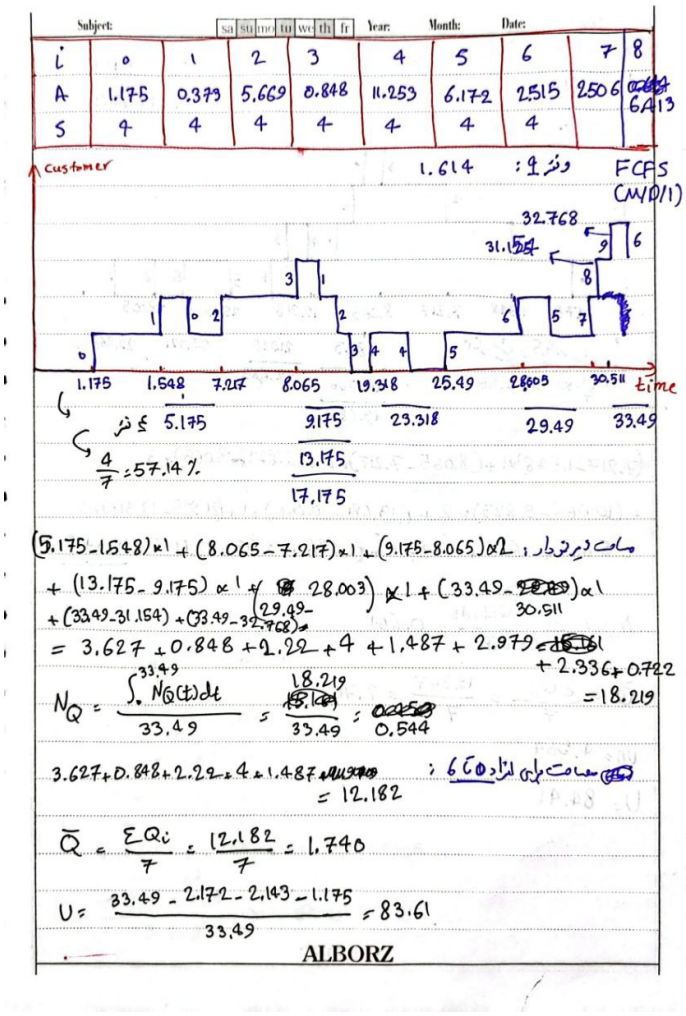


Figure 7: Handwritten: solution FCFS|MD1|7

- FCFS Policy MD1:
  - The average waiting time for people in the queue was 1.74 minutes.
  - The average number of people in the queue was 0.54.
  - 57.14% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 83.61% of the total simulation time.

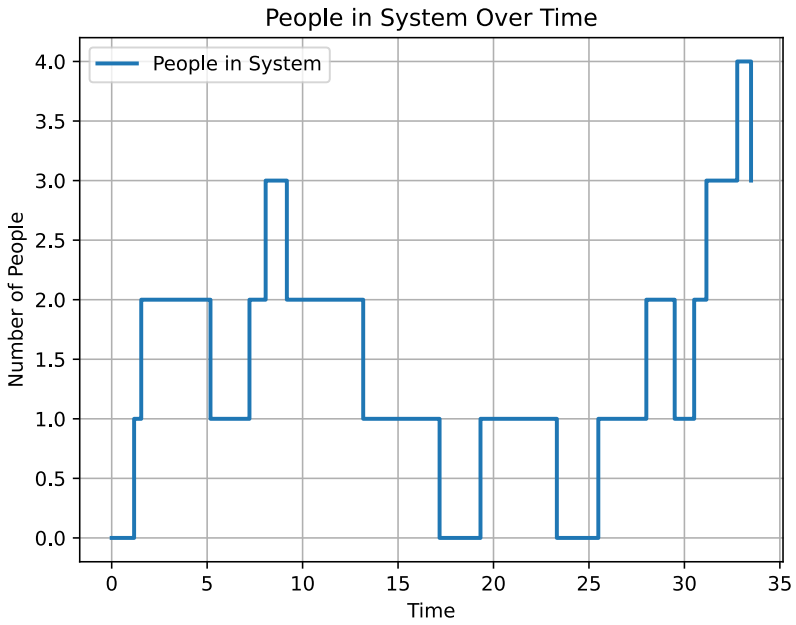


Figure 8: Sample Path for FCFS|MD1|7

```
from modular.main import main

model = main(policy="FCFS",
             stop_type="Customer",
             stop_limit=7,
             queue_type="MDI",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 0.0s

```
======================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 1.74
Therefore, Average number of People in Queue (N_Q) will be 0.54
We have 4 people with more that 4.5 min system time. Therefore the percentage is 57.14
Therefore the U=83.61

======================================================
```

Figure 9: Code: FCFS|MD1|7

4.1.2 45 min of simulating

Input data is:

$$\text{Intervals}: [1.175, 0.373, 5.669, 0.848, 11.253, 6.172, 2.515, 2.506,$$
$$0.641, 1.614, 1.795, 0.575, 3.761, 5.013, 0.645]$$
$$\text{Service times}: [8.648, 3.615, 0.242, 8.244, 0.760, 1.931, 1.263, 0.321$$
$$1.183, 4.001, 2.527, 3.211, 2.304, 1.289]$$

Solution to FCFS | M/M/1 | 45min:

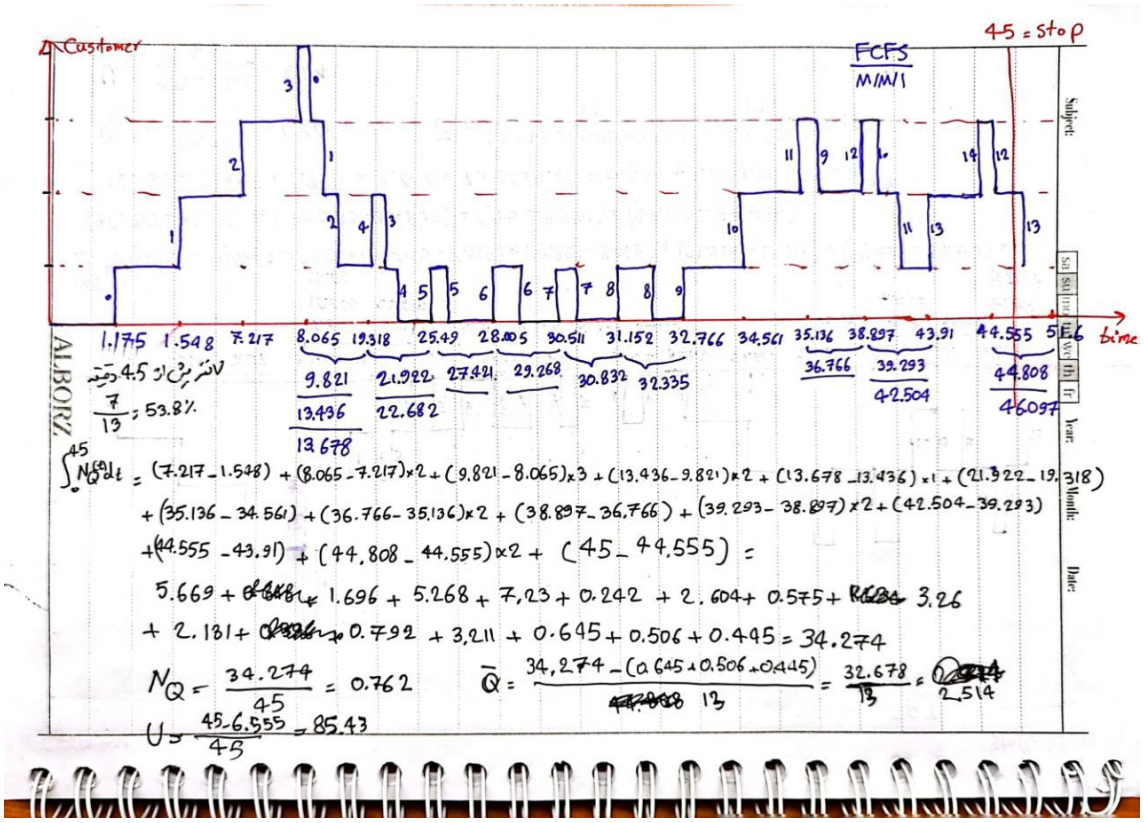


Figure 10: Handwritten: solution FCFS|MM1|45

- FCFS Policy MM1:
  - The average waiting time for people in the queue was 2.51 minutes.
  - The average number of people in the queue was 0.76.
  - 53.85% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 85.43% of the total simulation time.

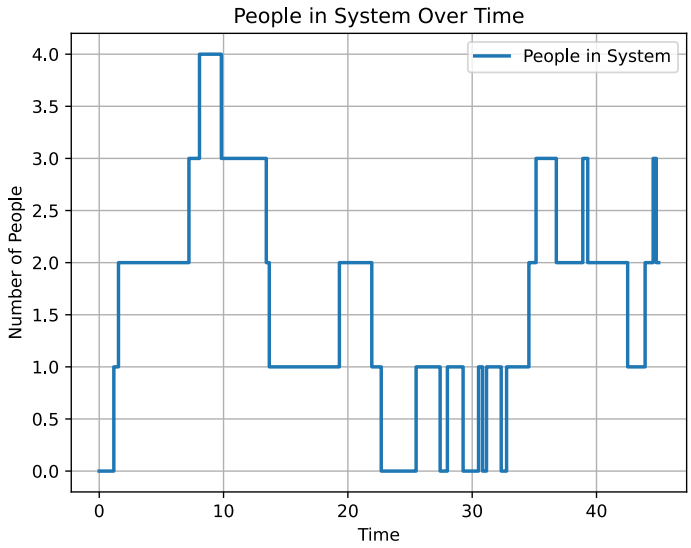


Figure 11: Sample Path for FCFS|MM1|45

```
from modular.main import main

model = main(policy="FCFS",
             stop_type="Time",
             stop_limit=45,
             queue_type="MMI",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 1.9s

```
=========================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 2.51
Therefore, Average number of People in Queue (N_Q) will be 0.76
We have 7 people with more that 4.5 min system time. Therefore the percentage is 53.85
Therefore the U=85.43

=========================================================
```

Figure 12: Code: FCFS|MM1|45

Solution to SPT | M/M/1 | 45min:

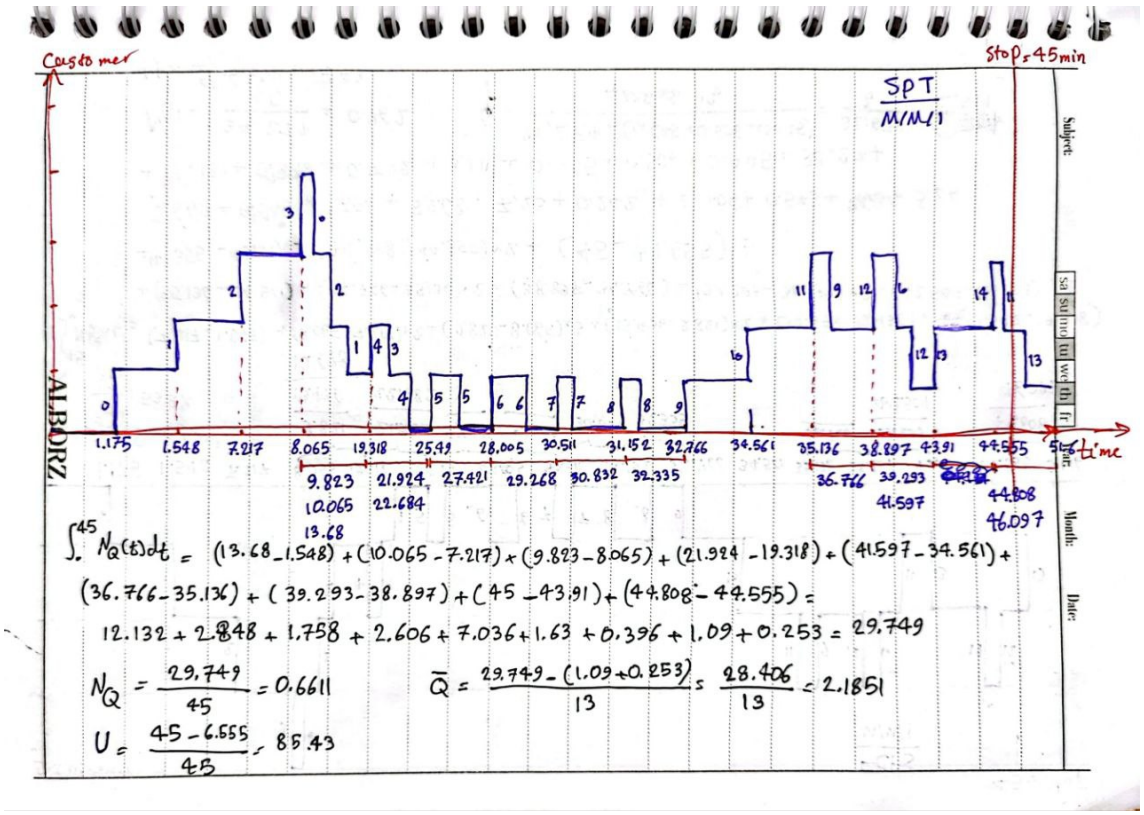


Figure 13: Handwritten: solution SPT|MM1|45

- SPT Policy MM1:
  - The average waiting time for people in the queue was 2.19 minutes.
  - The average number of people in the queue was 0.66.
  - 38.46% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 85.43% of the total simulation time.

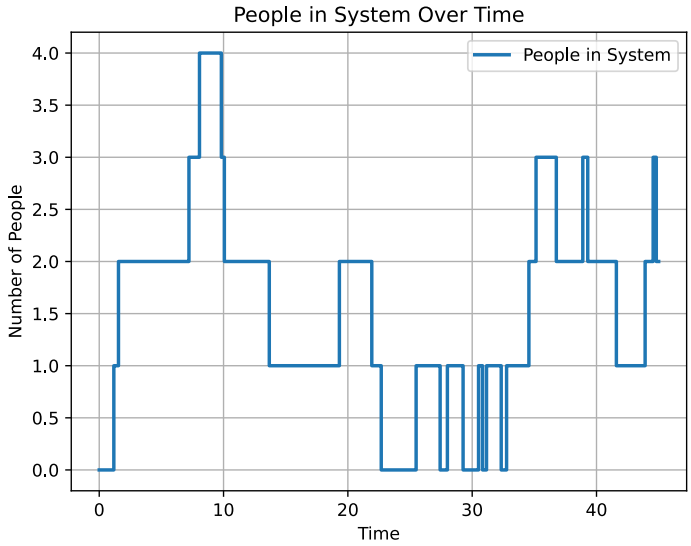


Figure 14: Sample Path for SPT|MM1|45

```
from modular.main import main

model = main(policy="SPT",
             stop_type="Time",
             stop_limit=45,
             queue_type="MM1",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 0.0s

```
==================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 2.19
Therefore, Average number of People in Queue (N_Q) will be 0.66
We have 5 people with more that 4.5 min system time. Therefore the percentage is 38.46
Therefore the U=85.43

==================================================
```

Figure 15: Code: SPT|MM1|45

Solution to FCFS | M/D/1 | 45min:

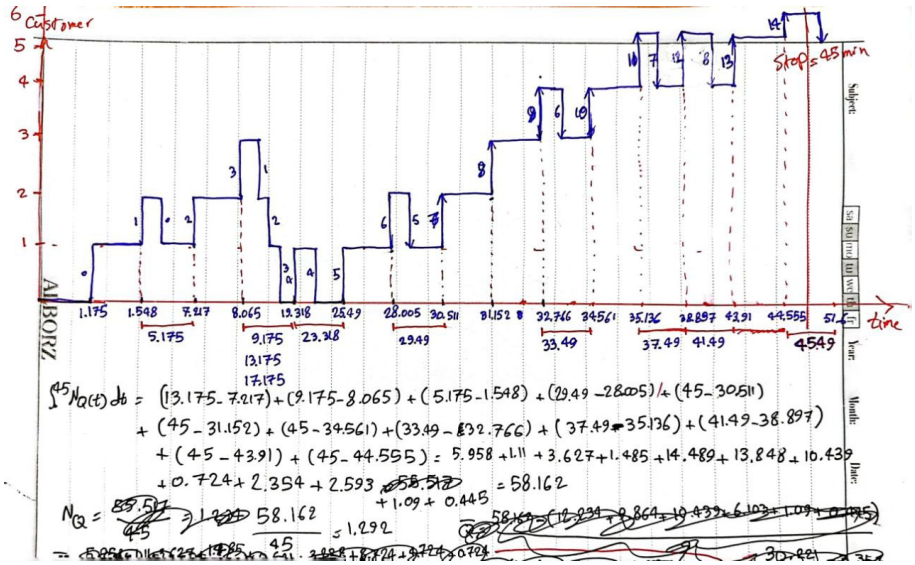


Figure 16: Handwritten: solution FCFS|MD1|45

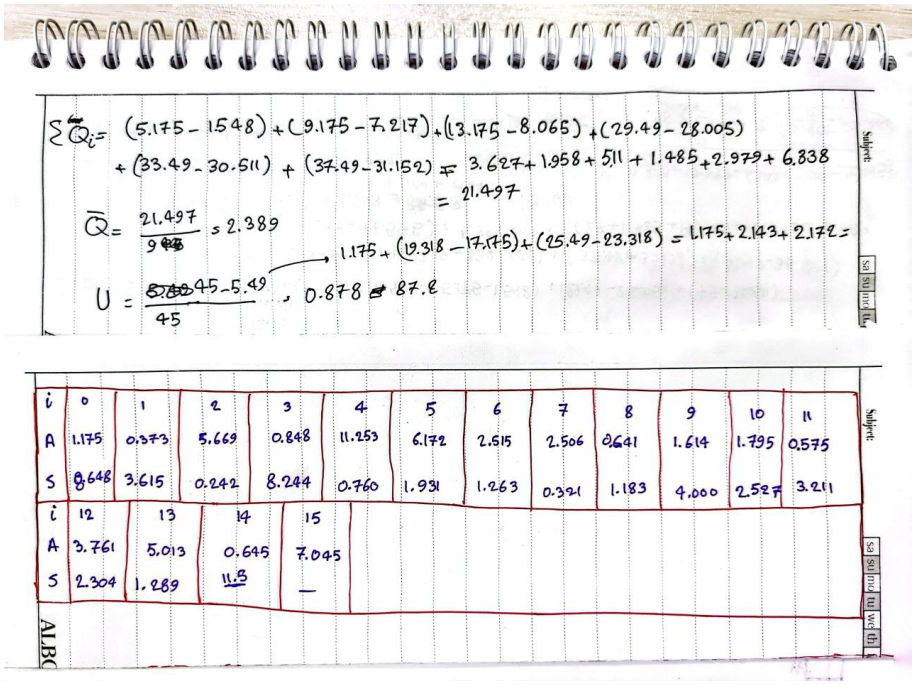


Figure 17: Handwritten: solution FCFS|MD1|45

- FCFS Policy MD1:
  - The average waiting time for people in the queue was 2.39 minutes.
  - The average number of people in the queue was 1.29.
  - 66.67% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 87.80% of the total simulation time.

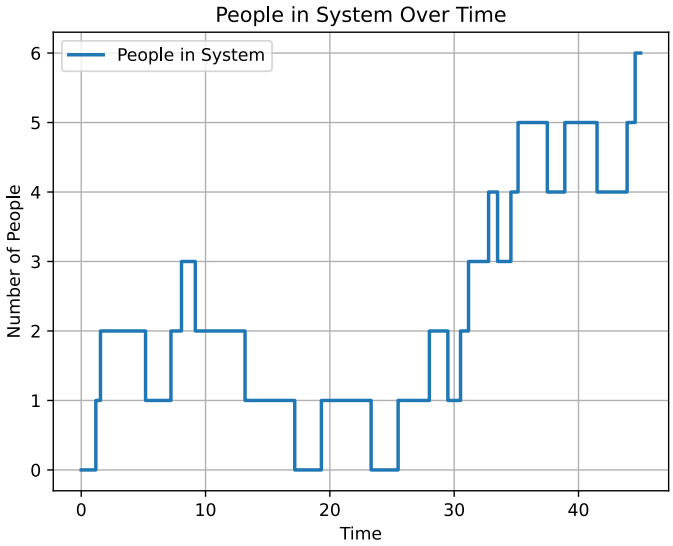


Figure 18: Sample Path for FCFS|MD1|45

```
from modular.main import main

model = main(policy="FCFS",
             stop_type="Time",
             stop_limit=45,
             queue_type="MDI",
             print_=False,
             detailed=False,
             plot_it=False)
```

✓ 0.9s

```
==================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 2.39
Therefore, Average number of People in Queue (N_Q) will be 1.29
We have 6 people with more that 4.5 min system time. Therefore the percentage is 66.67
Therefore the U=87.80

==================================================
```

Figure 19: Code: FCFS|MD1|45

4.2 Test Cases

According to test case file:

	Test case			Result			
	Process Policy	Termination	Intervals Dist.	Average time in Queue (mins)	Average Customers in Queue	customers with more than 4.5 mins in System (%)	Utilization (%)
1	FCFS	1000 Customers	MM1	19.93	4.18	85.20	87.44
2	SPT	1000 mins	MM1	10.23	2.14	69.05	90.56
3	FCFS	1000 Customers	MD1	6.84	1.43	78.00	83.56
4	SPT	1000 mins	MD1	6.96	1.46	79.05	84.00
5	FCFS	1000 mins	MM1	26.40	5.54	87.62	90.56
6	SPT	1000 Customers	MM1	8.98	1.89	68.60	87.41

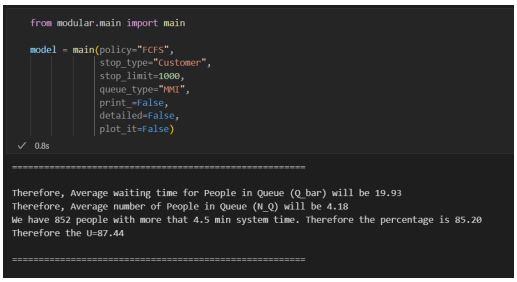
Figure 20: Test Cases

Now we run all the test cases and report the result at figure 21.

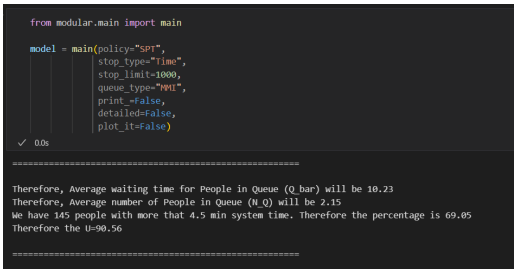
4.3 Simulation

4.3.1 10,000 customer threshold - M/M/1 and M/D/1

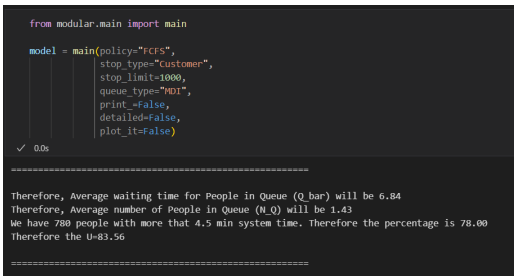
- FCFS Policy M/M/1:
  - The average waiting time for people in the queue was 16.57 minutes.
  - The average number of people in the queue was 3.29.
  - 80.37% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 80.27% of the total simulation time.
  - Run time: 3.3s
- SPT Policy M/M/1:
  - The average waiting time for people in the queue was 7.69 minutes.
  - The average number of people in the queue was 1.53.
  - 65.21% percent of people had more than 4.5 min time in system.



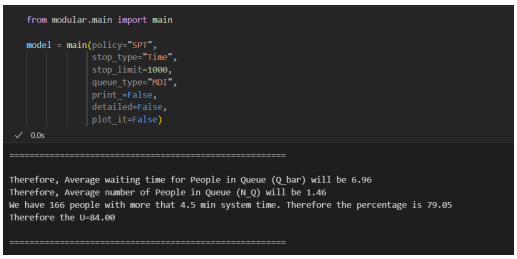
(a) Testcase 1



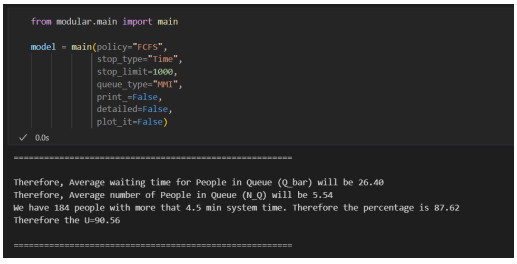
(b) Testcase 2



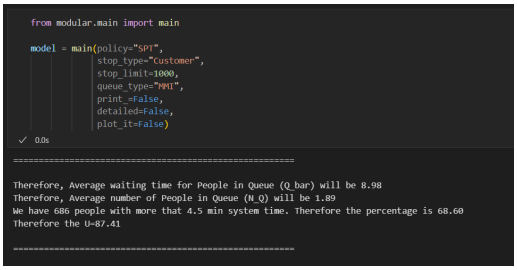
(c) Testcase 3



(d) Testcase 4



(e) Testcase 5



(f) Testcase 6

Figure 21: Test case results

  - The server was utilized for approximately 80.27% of the total simulation time.
  - Run time: 3.2s

- FCFS Policy M/D/1:
  - The average waiting time for people in the queue was 7.44 minutes.
  - The average number of people in the queue was 1.48.
  - 76.38% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 79.37% of the total simulation time.
  - Run time: 3.0s

4.3.2 1000 hours threshold - M/M/1 and M/D/1

- FCFS Policy M/M/1:
  - The average waiting time for people in the queue was 16.89 minutes.
  - The average number of people in the queue was 3.34.
  - 80.56% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 80.45% of the total simulation time.
  - Run time: 5.0s

- SPT Policy M/M/1:
  - The average waiting time for people in the queue was 7.86 minutes.
  - The average number of people in the queue was 1.56.
  - 65.34% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 80.45% of the total simulation time.
  - Run time: 4.8s

- FCFS Policy M/D/1:
  - The average waiting time for people in the queue was 7.28 minutes.
  - The average number of people in the queue was 1.44.
  - 76.20% percent of people had more than 4.5 min time in system.
  - The server was utilized for approximately 79.21% of the total simulation time.
  - Run time: 4.6s

5 Conclusion

In this report, we have analyzed the performance of two common queuing policies: First-Come-First-Serve (FCFS) and Shortest Processing Time (SPT), through simulation. We explored how these policies impact key metrics such as waiting times, system efficiency, and server utilization.

The simulation results clearly demonstrate that the SPT policy outperforms FCFS in terms of average waiting time and system throughput. By prioritizing customers with shorter service times, the SPT policy ensures that tasks are completed more efficiently, reducing the time each customer spends in the system. However, this efficiency comes at the cost of fairness, as customers with longer service times may experience longer delays due to the prioritization of shorter tasks.

On the other hand, the FCFS policy, while simpler and more equitable, leads to higher waiting times for some customers, especially when there is significant variation in service times. This policy ensures fairness by treating all customers equally, but it may not be ideal for systems where minimizing waiting time is critical.

Overall, the choice of queuing policy depends on the specific needs and priorities of the system. If minimizing waiting times and improving system throughput is the primary goal, then the SPT policy may be the better option, provided that fairness is not a major concern. However, in scenarios where fairness is more important than efficiency, such as in customer service systems, FCFS may be more appropriate.

Future work could expand on this simulation by considering more complex systems, such as multi-server models, time-varying service rates, and customer behaviors such as abandonment or re-entry into the queue. Additionally, comparing these policies with other scheduling strategies could provide further insights into their strengths and weaknesses in different contexts.

In conclusion, this report provides valuable insights into the dynamics of queuing systems and the trade-offs between different scheduling policies. By understanding these trade-offs, system designers can make more informed decisions about which policy to implement, depending on the specific goals of the system.

References

- Prof.Shafashi, *Simulation slides*, Winter, 2025.
- ChatGPT-o1, *OpenAi Model of o1(free account)*, Winter, 2025.

Appendix

A. Code for the Simulation

The following is the Python code that was used to run the simulation for the queuing system based on the policy chosen (either FCFS or SPT).

```
from modular.number_generator import Exponential_Generator
from modular.Tools import Queue, Server, System
import matplotlib.pyplot as plt
import time


class Queue_Simulator:
    """Simulating a one server one queue system using two seprate algorithm for FirstComeFirstServe(FCFS) and ShortestProcessingTimeFirst(SPT).
    """
    def __init__(self, interval_generator:Exponential_Generator,
                 service_time_generator:Exponential_Generator,
                 print_:bool=False,
                 detailed:bool=False) -> None:
        """Important parameters:
           1- intervals, service_times, arrivals, departures: Storing lists for events.
           2- Queue, Server, System: Classes we definded for this class.
           3- last_id, time, last_event: Main step parameters to observe the condition in system.

        Args:
            interval_generator (Exponential_Generator): A generator to make intervals for adding to system.
            service_time_generator (Exponential_Generator): A generator to make service times of the people in the system.
            print_ (bool, optional): If you want to have the informatino in each step to be printed. Defaults to False.
            detailed (bool, optional): If you want to have some more deep informatino about the system. Defaults to False.
        """
        self.print_ = print_
        self.detailed = detailed
        self.interval_generator = interval_generator
        self.service_time_generator = service_time_generator
        self.intervals = []
        self.service_times = []
        self.service_times_copy = []
        self.arrivals = []
        self.departures = []
        self.interval_counter = -1
        self.service_time_counter = -1
        self.queue = Queue()
        self.server = Server()
        self.system = System()
        self.time = 0
        self.last_event = None
        self.last_id = 0
        self.number_of_ids = 0
        self.left_over_queue = []
        self.left_over_system = []
        self.policy = None

    def fit(self, policy:str, last_id:float=float("inf"), last_time:float=float("inf")) -> None:
        """Run the algorithm for the simulation.

        Args:
            policy (str): 'FCFS' or 'SPT'.
            last_id (float, optional): If you know the last person id in the system to stop the simulation. Defaults to float("inf").
            last_time (float, optional): If you knwo the last time that the simulation should continue the run to. Defaults to float("inf").

        Raises:
            ValueError: The minimum value for last_id is 3.
            ValueError: You must set policy to 'FCFS' or 'SPT'. Nothing else.
        """
        self.policy = policy
        if last_id < 3:
            raise ValueError("[ERROR] You must set at least 3 id arrive in system.")
        self.stop_id, self.stop_time = last_id, last_time

        if policy=="FCFS":
            self._FCFS()
        elif policy=="SPT":
            self._SPT()
        else:
            raise ValueError("[ERROR] You must chose the policy correctly.")
```

B. Sample Output

Below is an example output generated by the simulation.

```
=====================================================

Therefore, Average waiting time for People in Queue (Q_bar) will be 7.28
Therefore, Average number of People in Queue (N_Q) will be 1.44
We have 9053 people with more that 4.5 min system time. Therefore the percentage is 76.20
Therefore the U=79.21

=====================================================
```