THE OPEN UNIVERSITY OF SRI LANKA

THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

FACULTY OF ENGINEERING TECHNOLOGY

EEX5362 Performance Modelling

Mini Project

University Course Registration System

Name: K.G.P.T. Kaluarachchi

Student Number: s22010111

Registration Number: 222510667

Due Date: 14/12/2025

Table of Contents

Table of Figures

1. System Description and Performance Goals

I selected the University Course Registration Process as the system for performance modelling. On registration day many students come to the university to register for their courses. They need to go to a registration counter and wait until a staff member is free. Only a limited number of counters are available. Because of this can happen long queues of students.

Most students arrive in the morning and around midday. This time the counters become very busy. Some students finish quickly but some students need more time due to course issues or document checking. This is reason for delays to others.

In this system is suitable for performance modelling because it has clear inputs and outputs. We can measure how long students wait; how many students can serve per hour and how many busy counters are. The main performance goals of this study are to reduce student waiting time, increase the number of students served within working hours, find slow periods during the day, and use counters and staff in a better way.

2. Modeling Approach and Assumptions

I used a discrete event simulation approach to study this system. I made the simulation using SimPy. It is a Python library used to model queue-based systems. In this model students are treated as processes. Registration counters are treated as shared resources.

Each student arrives at a certain time. (8.00AM – 4.00PM) If a counter is free, the student can get service immediately. If not, the student waits in a queue. When service starts, the counter is busy until the service finishes.

I made some assumptions to keep the model simple.

- All counters work at the same speed.
- All students join a single queue. (FIFO)
- Staff do not take breaks during the registration time.
- The registration day starts at 8:00 AM and ends at 4:00 PM.
- Service time is assumed to be between 5 and 10 minutes for every student.

These assumptions help me to focus on performance behavior instead of real-world complexity.

3. Data Description and Methodology

I used a dataset of 198 students for the simulation. This dataset is the same as the one created in Deliverable 01 and represents one full registration day. Each student has an arrival time, service time, and exit time. Arrival times change during the day. More students arrive during peak hours.

Service time is between 5 and 10 minutes. This matches a real registration process. The dataset is loaded into the simulation. If the dataset is missing, the code can generate similar data automatically.

The simulation was run under three scenarios. There are 1 counter, 2 counters, and 4 counters. For each scenario, the simulation calculates average waiting time, maximum waiting time, throughput per hour, and counter utilization.

4. Detailed Analysis and Findings

- The results clearly show the number of counters has a strong effect on performance.
- The throughput graph shows that throughput increases when more counters are added. With only 1 or 2 counters, fewer students are served per hour. With 4 counters, more students complete registration within the same time.
- The average waiting time graph shows a significant drop as counters increase. With 1 or 2 counters, students wait for a long time. With 4 counters, the average waiting time becomes much smaller. Adding more than 4 counters would give only a small improvement.
- The maximum waiting time graph shows the worst case waiting. This value is very high when counters are few. This means some students wait too long during peak hours.
- The utilization graph is very important.
- With fewer counters, utilization is very high. This means staff are overloaded. With many counters, utilization drops. This means some counters are idle.
- From this graph, 4 counters give a good balance between waiting time and staff usage.

5. Visualizations

5.1 Throughput by Number of Counters

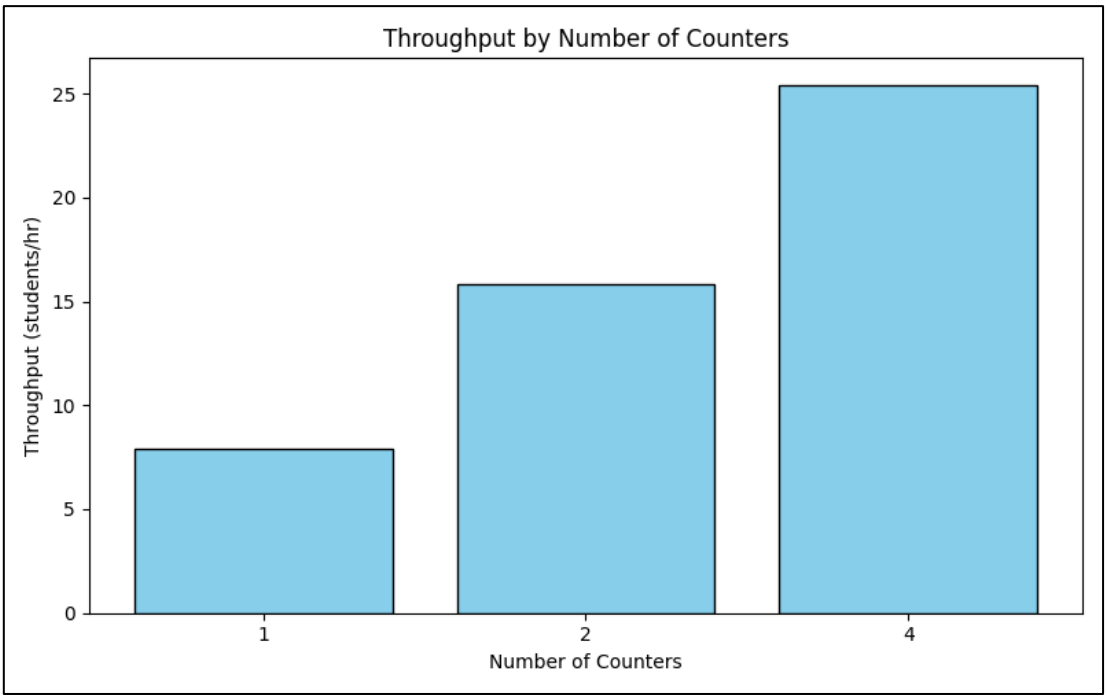

Figure 5.1: Throughput

5.2 Average Waiting Time by Number of Counters

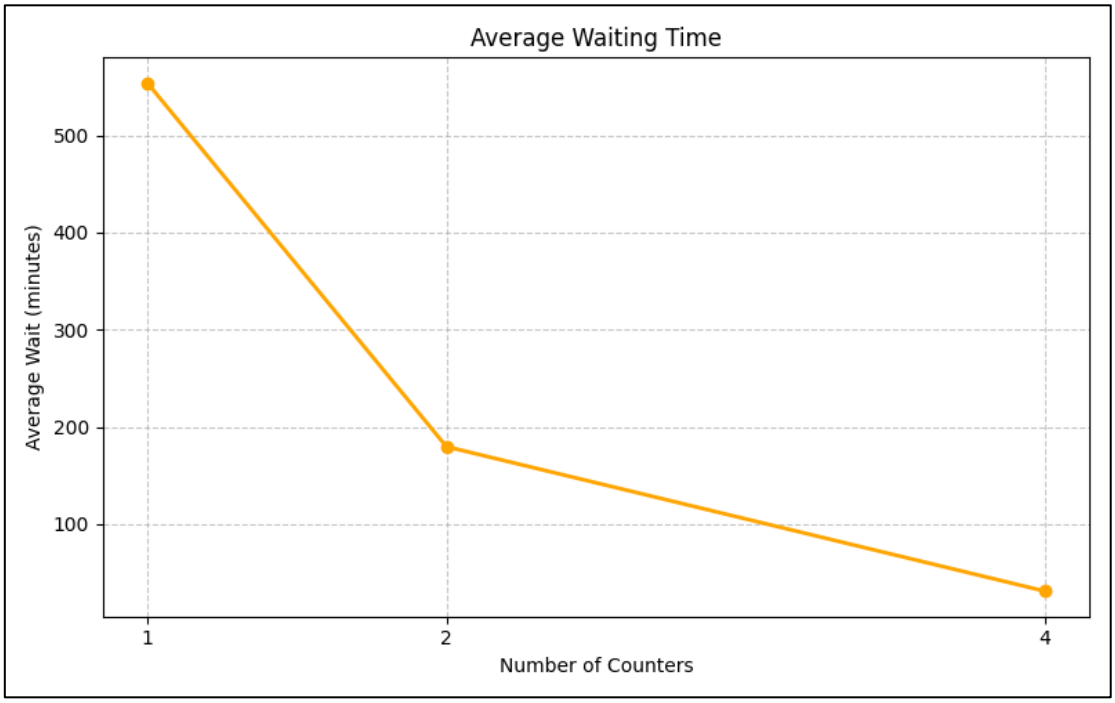

Figure 5.2: Average Waiting Time

5.3 Maximum Waiting Time by Number of Counters

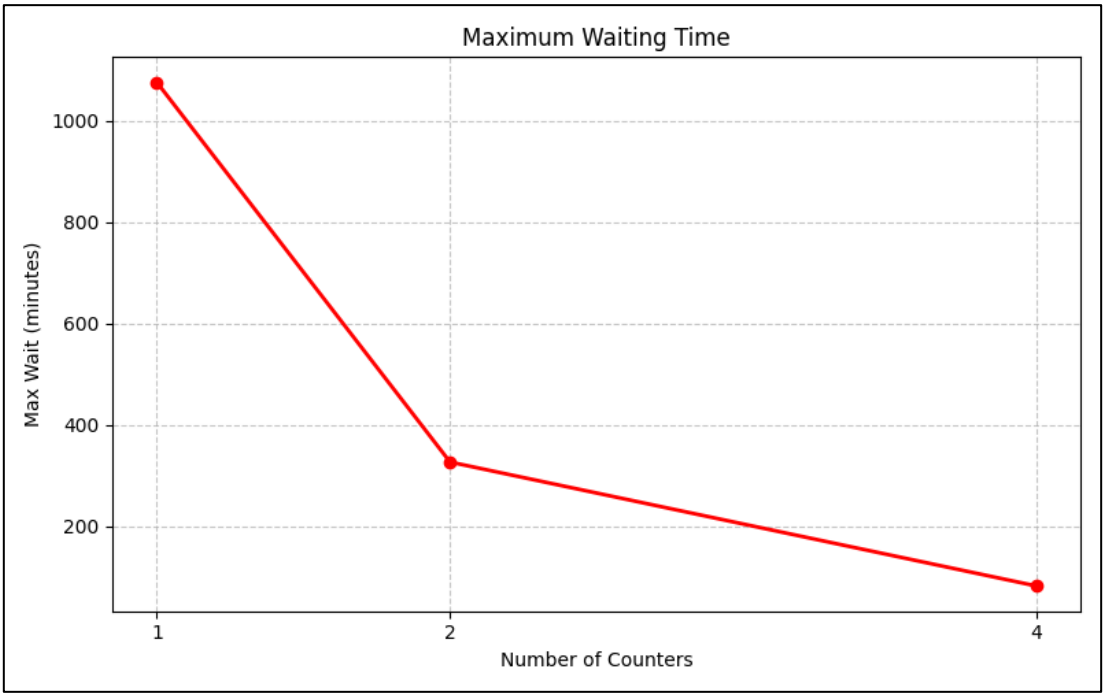

Figure 5.3: Maximum Waiting Time

5.4 Counter Utilization Percentage

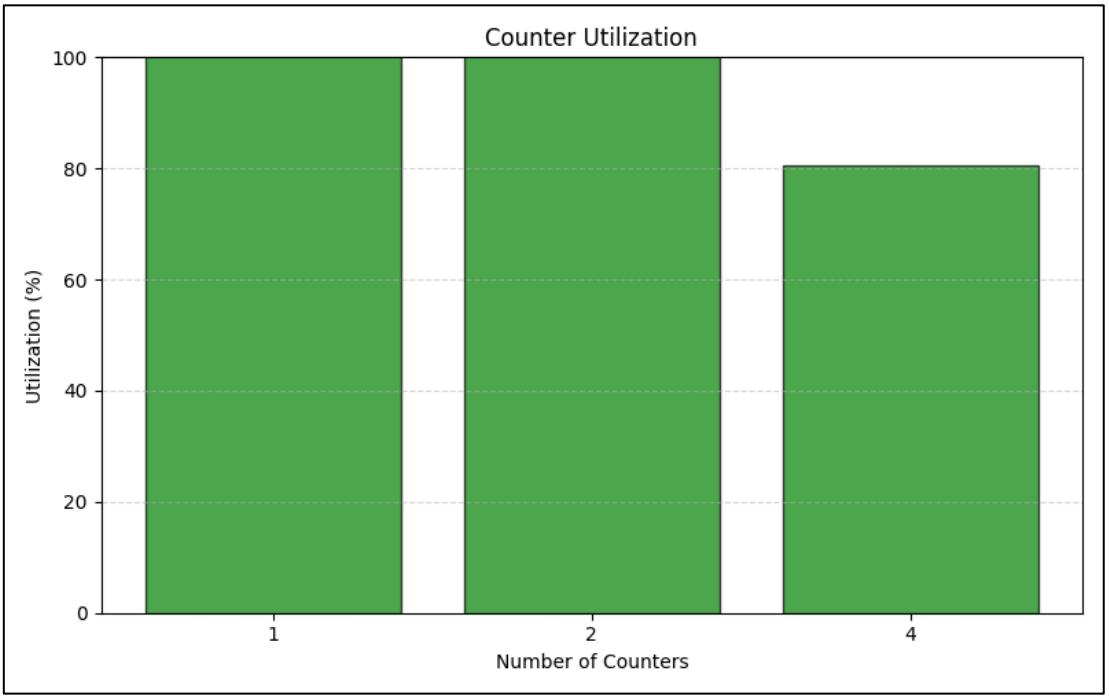

Figure 5.4: Utilization

5.5 Console Output

```
Loading dataset from queue_data.csv

Running scenario with 1 counters...
Running scenario with 2 counters...
Running scenario with 4 counters...

Simulation Completed. Summary of Results:

 counters  throughput_per_hr  avg_wait_min  max_wait_min  utilization_pct  num_served
        1           7.925019    553.985269   1076.050000       100.540342         198
        2          15.854445    179.721717    326.316667       100.568296         198
        4          25.425361     31.113131     81.650000        80.639379         198

All outputs saved in folder:
D:\OUSL\Academic 3rd Year\Semester 1-2\EEX5362 Performance Modelling\MP\Final\Finalized Code\sim_output

PS D:\OUSL\Academic 3rd Year\Semester 1-2\EEX5362 Performance Modelling\MP\Final\Finalized Code>
```

Figure 5.5: Console Output

These graphs help to clearly see performance changes and compare scenarios.

6. Limitations and Future Extensions

This model has some limitations. It does not include staff breaks and assumes that all students join a single queue. It also does not consider online registration.

In the future, the model can be improved by,

- Adding staff shifts and breaks.
- Introducing priority students (e.g., those with special needs or early registration).
- Combining online and physical queues into a single model.

7. Conclusion

This project helped me understand how performance modelling works in real systems. The simulation clearly shows how adding more counters affects student waiting times and overall efficiency. From the results, using around 4 counters provides good performance while avoiding unnecessary resource usage.

This study can help universities plan registration days more effectively, balance staff workload, and reduce student waiting times.

8. Appendix

The complete source code and dataset for the University Registration Queue Simulation using SimPy are available at the following GitHub repository:

8.1 GitHub Repository:

https://github.com/prabhathkaluarachchi/EEX5362_Performance_Modelling

This repository contains:

- Deliver_01_EEX5362_MP_222510667.docx – Deliverable 1 Report
- Mini_Project_EEX5362_MP_222510667.pdf – Final Report
- RegistrationSimulation_222510667.py – Final Simulation Code
- queue_data.csv – Used Dataset from Kaggle

9. References

[1] SimPy Documentation. Available: https://simpy.readthedocs.io

[2] R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, 1991.

[3] S. Tiwary, "Queue Waiting Time Prediction Dataset," Kaggle, 2023. Available: https://www.kaggle.com/datasets/sanjeebtiwary/queue-waiting-time-prediction/data/code

*** END ***