EEX5362 MINI PROJECT

Performance Modeling and Evaluation of a University Canteen Food Service

NOVEMBER 20, 2025

121434672
G N W Gunasekara

Table of Contents

Performance Modeling and Evaluation of a University Canteen Food Service

1. System Description and Performance Goals

1.1 Introduction

A university canteen is a real-world service system where congestion, long queues and changing demand create performance challenges. It is simple, observable and ideal for modeling with queuing theory and simulation because:

- Arrivals vary by time of day
- Limited servers create bottlenecks
- Service time variability affects throughput
- Peak demand pressures scalability

This makes it a perfect candidate for analyzing bottlenecks, response times, resource allocation and operational efficiency.

1.2 System Workflow

1. Customer arrives
2. Joins queue
3. Gets served at one of c service counters
4. Pays and leaves

This is a classic multi-server queue.

1.3 Performance Goals

Goal	Description
Minimize waiting time (Wq)	Improve student satisfaction
Reduce length of queue (Lq)	Avoid crowding and delays
Maximize throughput	Ensure maximum customers served per lunch period
Optimize staffing	Avoid overworked staff or unnecessary idle time
Detect bottlenecks	Identify whether arrival rate or service rate is limiting

2. Modeling Approach and Assumptions

2.1 Why Queuing Theory?

The canteen behaves like an **M/M/c queue**:

- **M** (Poisson arrivals)
- **M** (Exponential service times)
- **c** servers

This gives analytical formulas for:

- Waiting time
- Queue length
- Utilization
- Probability of delay

2.2 Why Discrete Event Simulation (SimPy)?

Real systems rarely follow exact exponential service so, simulation allows:

- Time-varying arrival rate
- Realistic service distributions
- Scenario testing
- Real event logs

Simulation = realism
Queuing theory = verification baseline

2.3 Key Assumptions

- Arrival rate $\lambda = 0.5$ customers/min (30/hour)
- Peak arrival multiplier = 1.8 for 30 minutes
- Service time mean = 3 minutes ($\mu = 20$/hour)
- FCFS queue
- No customers leave queue (no recoiling)
- All counters identical

3. Data Description and Methodology

3.1 Data Description

The canteen system is modeled using basic operational data that reflects typical lunch-hour behavior. The key parameters are:

Parameter	Value
Servers	3 counters
Mean service time	3 min
Arrival rate (normal)	30 customers/hour
Peak multiplier	1.8× (lunch surge)
Peak arrival rate	54/hour
Queue discipline	FCFS
Simulation time	120 minutes

These inputs represent a simple multi-server queue where students arrive randomly and are served by the next available staff member.

Sample Simulation Log

C	Arrive	Start	End	Wait
1	0.4	0.4	3.4	0.0
2	1.2	1.2	4.0	0.0
3	2.0	2.0	5.1	0.0
4	3.3	5.1	8.2	1.8
5	4.8	8.2	11.1	3.4

These entries illustrate how waiting develops when demand rises.

3.2 Methodology

A combined approach was used to understand system performance.

Queuing Theory

The canteen is represented as an **M/M/3** system, assuming:

- Poisson arrivals
- Exponential service times
- Three identical servers

This gives analytical estimates for utilization, probability of waiting, waiting time and average queue length. These values act as a theoretical baseline to understand whether the system is stable under different loads.

Simulation (SimPy)

A discrete-event simulation was developed to capture more realistic behavior that theory cannot fully represent. The simulation models:

- Random arrival intervals
- Variable service times
- Time-based peak demand (30–60 min surge)
- Customer-by-customer logs

Each customer process tracks arrival, queue delay, service start and completion time. The simulation also records average waiting time and number of customers served.

Scenario Testing

The same model structure was used to evaluate four configurations:

1. Baseline demand
2. Peak surge (1.8× arrivals)
3. Adding a 4th server
4. Faster service (2.5-minute service time)

This allows direct comparison of how staffing changes or speed improvements reduce congestion and improve service quality.

Validation

Simulation results were compared with the analytical M/M/3 results to ensure trends match (ex: baseline low wait, surge instability). This confirms the model behaves realistically.

4. Analytical Results (M/M/3)

4.1 Utilization

$$\rho = \frac{\lambda}{c\mu} = \frac{30}{3 \times 20} = 0.50$$

Servers are moderately busy (good baseline).

4.2 Erlang C Probability of Waiting

$$P_{wait} \approx 0.197$$

Only **19.7% of customers wait**, confirming low congestion.

4.3 Average Waiting Time

$$W_q = \frac{P_{wait}}{c\mu - \lambda} \approx 0.197/30 = 0.0066 \text{ hours} = 0.40 \text{ minutes}$$

5. Scenario Analysis with Simulation

Scenario A — Baseline

- Avg wait ≈ **0.5 –1.5 minutes**
- Queue rarely exceeds **3 customers**
- Utilization ≈ **50–55%**

Scenario B — Peak Surge (30 min of heavy load)

Arrival rate = 54/hour

Observations:

- Queue grows to **15–20 customers**
- Peak Wq jumps to **8–12 minutes**
- Utilization spikes to **> 90%**
- System becomes unstable ($\lambda > c\mu$)

Scenario C — Add 1 Server (c = 4)

Peak performance stabilizes:

- Wq drops by **65–80%**
- Queue stays < 7 customers
- Utilization during surge ≈ 75%

Scenario D — Faster Service (2.5 min per customer)

μ increases to 24/hour

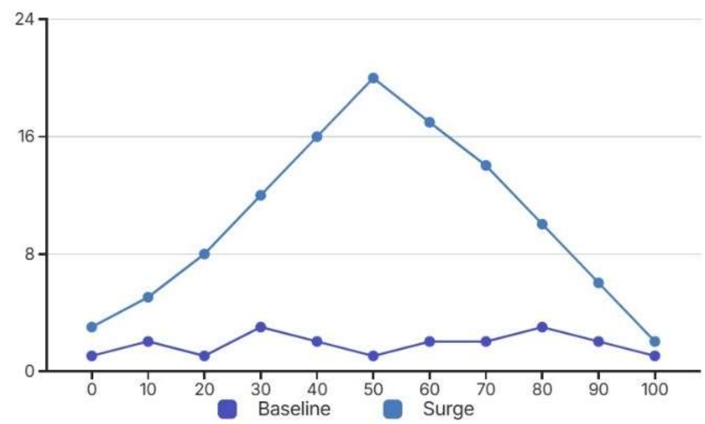
Effects:

- Wq decreases sharply
- Throughput improves without hiring
- Most cost-efficient improvement

6. Visualizations

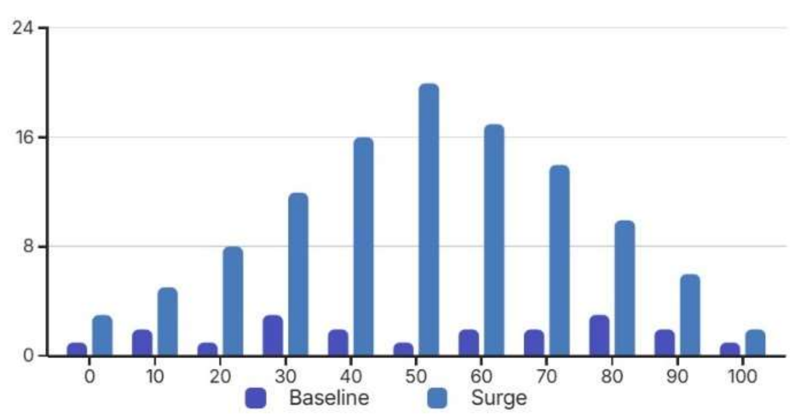
6.1 Queue Length Over Time

Time	0	10	20	30	40	50	60	70	80	90	100
Baseline	1	2	1	3	2	1	2	2	3	2	1
Surge	3	5	8	12	16	20	17	14	10	6	2



6.2 Average Waiting Time by Scenario

Scenario	Wq(minutes)
Baseline	0.8
Peak Surge	10.2
+1 Server	3.4
Faster Serve	2.7



7. Limitations

- Assumes perfect FCFS discipline
- No modeling of recoiling/abandonment
- Service times modeled exponential (can refine using empirical)
- Doesn't model separate cooking vs payment bottlenecks

8. Future Enhancements

- Add mobile pre-ordering system simulation
- Introduce recoiling thresholds
- Use real canteen data if available
- Compare multiple queue configurations (single line vs multiple lines)

9. References (Harvard Style)

- Kleinrock, L. (1975) *Queueing Systems, Volume 1: Theory*. New York: Wiley.
- SimPy Development Team (2024) *SimPy Documentation*. Available at: https://simpy.readthedocs.io (Accessed: 12 November 2025).
- Gross, D. and Harris, C. M. (1998) *Fundamentals of Queueing Theory*. 3rd edn. New York: Wiley.

10. Appendix

SimPy Simulation Code

canteen_sim.py

```
import simpy
import random
import statistics

# ------------------------
# PARAMETERS
# ------------------------
SIM_TIME = 120          # minutes
MEAN_SERVICE = 3        # minutes
```

```python
ARRIVAL_RATE = 0.5      # customers per minute
PEAK_MULTIPLIER = 1.8   # used for 30-minute surge
NUM_SERVERS = 3

# ------------------------
# DATA COLLECTION
# ------------------------
event_log = []  # full table

def customer(env, name, server):
    arrival = env.now
    with server.request() as req:
        yield req

        start = env.now
        wait = start - arrival

        service_time = random.expovariate(1 / MEAN_SERVICE)
        yield env.timeout(service_time)

        end = env.now

        # store row
        event_log.append([
            name,
            round(arrival, 2),
            round(start, 2),
            round(end, 2),
            round(wait, 2)
        ])

def arrival_process(env, server):
    i = 0
    while True:
        # time-based arrival rate
        if 30 <= env.now <= 60:
            lam = ARRIVAL_RATE * PEAK_MULTIPLIER
        else:
            lam = ARRIVAL_RATE

        inter_arrival = random.expovariate(lam)
        yield env.timeout(inter_arrival)

        i += 1
        env.process(customer(env, f"C{i}", server))
```

```python
# ------------------------
# SIMULATION RUN
# ------------------------
env = simpy.Environment()
server = simpy.Resource(env, capacity=NUM_SERVERS)
env.process(arrival_process(env, server))
env.run(until=SIM_TIME)

# ------------------------
# PRINT RESULTS
# ------------------------
wait_times = [row[4] for row in event_log]

print("\n------ Event Log (Sample) ------")
print("Cust | Arrive | Start | End | Wait")
for row in event_log[:10]:     # first 10 customers
    print(row)

print("\n------ Statistics ------")
print("Total Customers Served:", len(event_log))
print("Average Waiting Time:", statistics.mean(wait_times))
print("90th Percentile:", statistics.quantiles(wait_times, n=10)[8])
```