MINI PROJECT REPORT: PERFORMANCE MODELING AND EVALUATION

The Open University of Sri Lanka
Faculty of Engineering Technology

System: Hospital Emergency Room (ER)

Patient Waiting Time & Doctor Utilization Analysis

Registration Number: 321428806

Table of Contents

1. Introduction

This study focuses on modeling and analyzing the performance of a **Hospital Emergency Room (ER)** system. The main purpose is to evaluate key performance aspects such as **patient waiting time**, **utilization of doctors**, and **overall throughput**. Emergency rooms operate under unpredictable patient arrival patterns and limited medical resources, which can lead to congestion, long waiting times, and workload pressure on staff.

By using a simulation-based approach, this study aims to identify operational bottlenecks, measure efficiency under different staffing levels, and propose practical improvements. The results are useful for understanding how changes in doctor availability affect patient delays and service capacity.

2. System Description and Performance Goals

2.1 System Description

The emergency room process begins when a patient arrives at the hospital and proceeds through the service flow. In this model, the focus is on the main bottleneck stage: **waiting for doctor consultation**.

System flow (modeled):
Patient Arrival → Wait for Doctor → Consultation → Exit (discharge)

The ER operates with limited doctor resources. When all doctors are busy, arriving patients must wait in a queue. The system performance depends strongly on arrival rate, service times, and the number of doctors available.

2.2 Performance Objectives

The main performance objectives of this study are:

1. **Minimize Patient Waiting Time:** Reduce average time patients wait before consultation.
2. **Maximize Resource Utilization:** Keep doctors efficiently used without extreme overload.
3. **Identify System Bottlenecks:** Determine where delays are mainly created.
4. **Optimize Throughput:** Increase number of patients served per hour.
5. **Check Scalability:** Observe performance changes when doctor capacity increases.

3. Modeling Approach and Assumptions

3.1 Modeling Approach

This study uses a **discrete-event, queue-based simulation** to model patient flow. Patients arrive over time and form a queue if no doctor is free. Doctors serve patients in parallel, and patients are handled using **First-Come, First-Served (FCFS)** logic.

To evaluate how staffing affects performance, the simulation compares **three scenarios**:

- 1 doctor
- 2 doctors
- 3 doctors

The arrival and service-time dataset is kept the same across scenarios to ensure a fair comparison.

3.2 Assumptions

To keep the model clear and manageable, the following assumptions are used:

- **FCFS queue discipline** (no triage priority modeled).
- **Doctor consultation stage only** is modeled (registration/nurse/lab/beds are not included).
- Each patient has an independent service time based on the dataset.
- Patients do not leave the queue once they arrive (no abandonment).
- The number of doctors stays constant during each simulation run.
- The simulation ends when the last patient finishes consultation.

4. Data Description and Methodology

4.1 Dataset Description

A simulated dataset of **50 patients** was used (dataset.csv). Each record contains:

- PatientID (P001–P050)
- ArrivalTime_min (minutes from start)
- ServiceTime_min (consultation duration in minutes)

Inter-arrival gaps were generated between **1–5 minutes**, and service times were generated between **4–10 minutes**, representing realistic variability in ER arrivals and consultation times.

4.2 Methodology

For each scenario, the simulation performs the following:

1. Read patient arrival and service times from dataset.csv.
2. Assign each arriving patient to the **earliest available doctor**.
3. Compute per-patient metrics:
    - StartService = max(arrival time, doctor available time)
    - WaitingTime = StartService − ArrivalTime
    - FinishTime = StartService + ServiceTime
4. Compute overall performance metrics:
    - Average waiting time
    - Average queue length (estimated)
    - Doctor utilization (%)
    - Throughput (patients/hour)
5. Save outputs:
    - results_summary.csv
    - detailed results per scenario
    - graphs as PNG files

5. Detailed Analysis and Findings

5.1 Scenario Summary Results (From results_summary.csv)

Average Waiting Time

- 1 Doctor: 101.32 minutes
- 2 Doctors: 11.94 minutes
- 3 Doctors: 0.36 minutes

Waiting time drops massively when moving from 1 to 2 doctors. With 3 doctors, waiting becomes almost zero, meaning the system can handle the load very smoothly.

Average Queue Length (Estimated)

- 1 Doctor: 13.77 patients
- 2 Doctors: 3.13 patients
- 3 Doctors: 0.11 patients

Queue length follows the same pattern as waiting time. One doctor causes major congestion. Additional doctors reduce queue buildup significantly.

Doctor Utilization

- 1 Doctor: 100%
- 2 Doctors: 96.34%
- 3 Doctors: 75.26%

With 1 doctor, the doctor is always busy, which causes high waiting time and staff overload risk. With 2 doctors, utilization remains high but the waiting drops to a much more acceptable level. With 3 doctors, utilization reduces to ~75%, which suggests extra capacity exists.

Throughput (Patients per Hour)

- 1 Doctor: 8.15 patients/hour
- 2 Doctors: 15.71 patients/hour
- 3 Doctors: 18.40 patients/hour

Throughput increases as doctor count increases. However, the improvement from 2 to 3 doctors is smaller than from 1 to 2 doctors, indicating **diminishing returns** after a certain point.

5.2 Bottleneck Identification

The simulation clearly identifies **doctor availability** as the main bottleneck. When the number of doctors is low, waiting time and queue length become very high.

5.3 Recommendations (Based on Results)

- For the given arrival pattern, **2 doctors** provides a strong balance: waiting is greatly reduced while utilization stays high.
- Adding a **3rd doctor** gives excellent waiting time performance but reduces utilization, so it may be best only during peak hours.
- A practical strategy is **dynamic staffing**: add one extra doctor during busy periods rather than permanently.

6. Visualizations

Figure 1: Average Waiting Time vs Number of Doctors

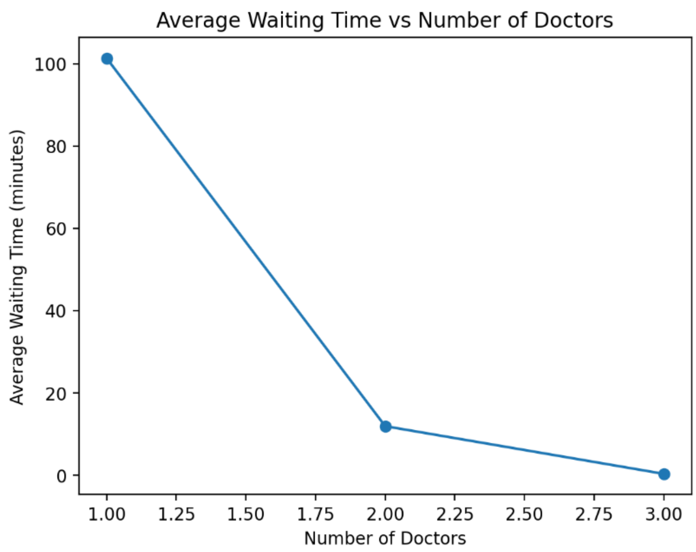


Figure 2: Average Queue Length vs Number of Doctors

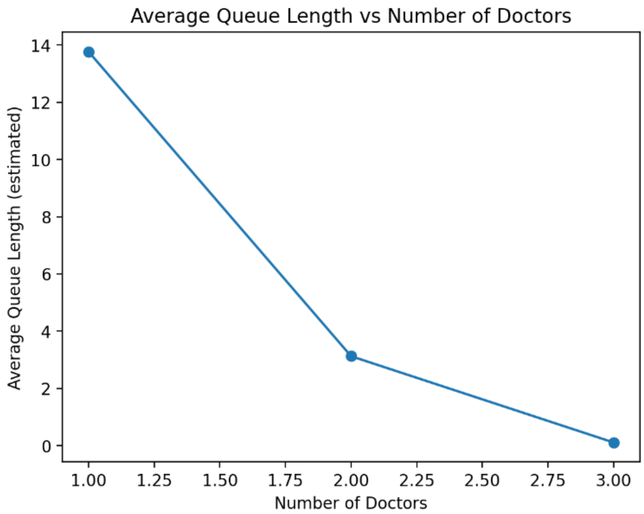


Figure 3: Doctor Utilization vs Number of Doctors

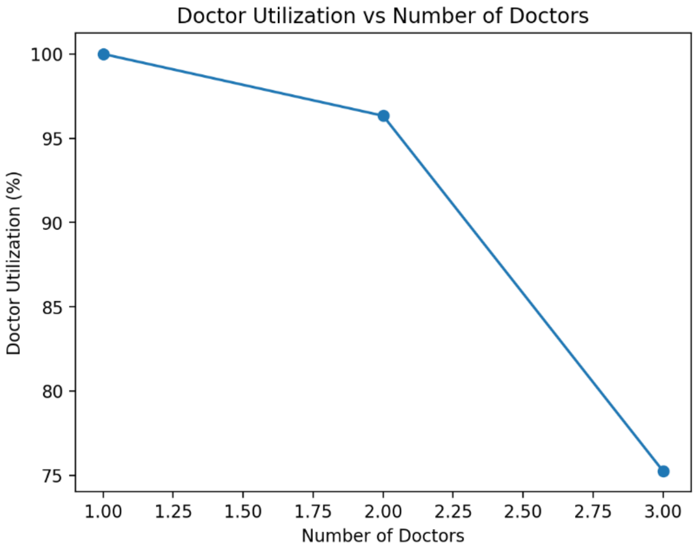
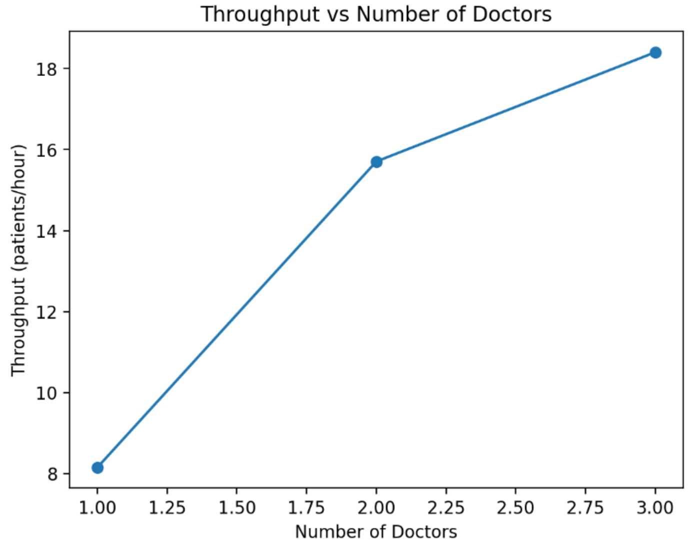


Figure 4: Throughput vs Number of Doctors



7. Limitations and Future Extensions

7.1 Limitations

- The dataset is simulated, not collected from a real hospital ER.
- The model focuses on the doctor consultation stage only.
- No triage priority (urgent vs non-urgent) is included.
- No registration delays, nurse stages, lab tests, or bed availability is modeled.

7.2 Future Extensions

- Add triage priority levels and priority queuing.
- Include registration clerk and nurse resources.
- Add beds/observation capacity and admission decisions.
- Use real ER data (if available) to calibrate arrival and service time distributions.
- Test time-based staffing policies (peak hour staffing).

8. Conclusion

This project modeled a hospital ER using a discrete-event queue simulation to evaluate patient waiting time, queue length, throughput, and doctor utilization. The results show that doctor capacity is a critical factor affecting ER performance. With only one doctor, the system becomes highly congested, producing extremely long waiting times (101.32 minutes average). Increasing capacity to two doctors dramatically improves performance (11.94 minutes average waiting) while still maintaining high utilization (96.34%). With three doctors, waiting time becomes almost zero (0.36 minutes), but utilization drops to 75.26%, suggesting possible overcapacity for the given workload.

Overall, the simulation indicates that **two doctors** provides an efficient balance between responsiveness and resource use under the modeled conditions, while a third doctor is best considered as a peak-hour strategy.

9. References (Harvard)

- GeeksforGeeks (n.d.) *Operating Systems and related performance concepts*. Available at: https://www.geeksforgeeks.org/
- StudyTonight (n.d.) *Operating System Tutorial*. Available at: https://www.studytonight.com/operating-system/
- Banks, J., Carson, J.S., Nelson, B.L. and Nicol, D.M. (2010) *Discrete-Event System Simulation*. 5th edn. Pearson.

10. Appendix

Appendix A: Files Produced

- dataset.csv (input dataset)

PatientID	ArrivalTime_min	ServiceTime_min
P001	0	7
P002	5	10
P003	8	7
P004	13	10
P005	18	7
P006	20	8
P007	23	10
P008	26	6
P009	29	9
P010	34	4
P011	38	7
P012	41	5
P013	46	7
P014	48	5
P015	52	9
P016	54	9
P017	58	9
P018	63	5
P019	64	7
P020	68	9
P021	70	8
P022	75	10
P023	79	5
P024	80	5
P025	81	7
P026	84	5

- simulation.py

```python


from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

BASE_DIR = Path(__file__).resolve().parent

DATASET_PATH = BASE_DIR / "dataset.csv"

SCENARIOS = [1, 2, 3]  # number of doctors to compare


def simulate(df: pd.DataFrame, num_doctors: int):
    """Simulate FCFS queue with num_doctors servers."""
    avail = [0.0] * num_doctors      # next available time per docto
    busy = [0.0] * num_doctors       # total busy time per doctor

    rows = []
    for _, r in df.iterrows():
        arrival = float(r["ArrivalTime_min"])
        service = float(r["ServiceTime_min"])

        # assign the patient to the earliest available doctor
        d_idx = int(np.argmin(avail))
        start = max(arrival, avail[d_idx])
        wait = start - arrival
        finish = start + service

        avail[d_idx] = finish
        busy[d_idx] += service

```

```python
def simulate(df: pd.DataFrame, num_doctors: int):
            rows.append({
                "PatientID": r["PatientID"],
                "ArrivalTime_min": arrival,
                "ServiceTime_min": service,
                "Doctor": f"D{d_idx + 1}",
                "StartService_min": start,
                "WaitingTime_min": wait,
                "FinishTime_min": finish
            })

        detail = pd.DataFrame(rows)
        sim_end = float(detail["FinishTime_min"].max())

        avg_wait = float(detail["WaitingTime_min"].mean())
        throughput = len(detail) / (sim_end / 60.0) if sim_end > 0 else float("nan")

        total_busy = float(sum(busy))
        utilization = total_busy / (num_doctors * sim_end) if sim_end > 0 else float("nan")

        # Simple estimated average queue length:
        # (Total waiting time area) / (simulation time)
        avg_queue_len = float(detail["WaitingTime_min"].sum() / sim_end) if sim_end > 0 else float("nan")

        metrics = {
            "Doctors": num_doctors,
            "AvgWaitingTime_min": avg_wait,
            "AvgQueueLength_est": avg_queue_len,
            "Throughput_patients_per_hr": throughput,
            "DoctorUtilization_fraction": utilization,
            "DoctorUtilization_percent": utilization * 100,
            "SimulationEnd_min": sim_end,
```

PS C:\Users\ACER\Downloads\EEX5362-MP-321428806>

- results_summary.csv (scenario results)

Doctors	AvgWaitingTime_min	AvgQueueLength_est	Throughput_patients_per_	DoctorUtilization_fract	DoctorUtilization_percent	SimulationEnd_min
1	101.32	13.76630435	8.152173913	1	100	368
2	11.94	3.12565445	15.70680628	0.963350785	96.33507853	191
3	0.36	0.110429448	18.40490798	0.752556237	75.25562372	163