# Milestone 4: Analysis and Validation of an Emergency Room Simulation

Aashna Suthar  Ash Arul
CS 4632 – Modeling and Simulation
Kennesaw State University
Fall 2025

***Abstract*—This milestone presents the analysis and validation of our discrete–event simulation of a small hospital emergency room. In Milestone 3 we finished the main model and collected output for several runs with different numbers of beds, labs, and random seeds. For Milestone 4 we used those data to study how the model behaves when key resources change and to check if the results are reasonable for the kind of system we tried to represent. We focus on three files: a per–run summary table, a beds versus total time table, and a labs versus total time table. From these we built simple graphs and basic statistics to perform sensitivity analysis, scenario testing, and face validation. Overall the model reacts in a consistent way when resources change and the variation across runs is stable, so we consider the model good enough for the level of detail of this class project, while still noting limits and future improvements.**

## I. Introduction

The goal of our project is to model a simplified emergency room using discrete event simulation in Python and SimPy. Patients arrive according to a Poisson process, wait in triage, then either go straight to a bed or pass through a lab station before discharge. The key resources we control are the number of beds and the number of lab stations.

In Milestone 3 we showed that the code runs correctly and we produced batch runs that saved detailed results to comma separated value (CSV) files. Milestone 4 focuses on analysis and validation. We want to see how sensitive the model is to changes in beds and labs and whether the output looks reasonable for a busy but not extreme emergency room. This report uses the data already stored in our GitHub repository and organizes it into three main parts: sensitivity analysis, scenario testing, and validation.

## II. Data used for Analysis

All of the data for this milestone come from the batch runs we executed in Milestone 3. The CSV files are in the `data/m4_summary` folder of the repository and were generated from the main simulation script.

### A. Per–Run Summary

The file `per_run_summary.csv` contains one row per run. Each row has the file name of the run and the following columns:

- `seed`: The random seed used for the run.
- `beds`: The number of bed resources.
- `labs`: The number of lab resources.
- `n_patients`: The number of patients processed.
- `avg_triage_wait`: The mean waiting time in triage.
- `avg_bed_wait`: The mean waiting time for a bed.
- `avg_lab_time`: The mean time spent in the lab.
- `avg_total_time`: The mean time from arrival to discharge.
- `std_total_time`: The standard deviation of total time.

These values are already aggregated for each run, so we used them directly in our tables and plots. The runs cover a range of settings such as $(\text{beds} = 2, \text{labs} = 1)$, $(\text{beds} = 5, \text{labs} = 1)$, $(\text{beds} = 5, \text{labs} = 2)$, $(\text{beds} = 5, \text{labs} = 5)$, $(\text{beds} = 8, \text{labs} = 2)$, and $(\text{beds} = 10, \text{labs} = 1)$ with several different seeds.

### B. Beds vs. Total Time

The file `beds_vs_total_time.csv` summarizes how the average total time in system changes as the number of beds changes. For each bed setting we have a mean and standard deviation of total time computed over all runs that share that bed count. This dataset feeds into the "beds" sensitivity plot our team created.

### C. Labs vs. Total Time

The file `labs_vs_total_time.csv` plays the same role for labs. For each lab capacity we store the mean and standard deviation of total time over all runs that share that lab count. This supports the "labs" sensitivity plot.

## III. Parameter Analysis

Sensitivity analysis checks how output measures change when we move one parameter while keeping the others close to a baseline. For this project we focused on total time in system because that combines all parts of the patient journey.

### A. Effect of Beds

Figure 1 shows the mean total time as a function of the number of beds using the values from `beds_vs_total_time.csv`. Each point represents the average over multiple seeds and lab settings. In general runs with more beds process more patients and avoid extreme bed waits. When the system only has two beds, bed waits can become large and total time tends to be high. Adding beds from two to five gives the biggest drop in total time. Going

from five to eight and ten beds gives smaller improvements. This pattern suggests that the system is bed constrained at low capacity but becomes limited by other stages, such as triage or labs, once bed capacity is high enough.

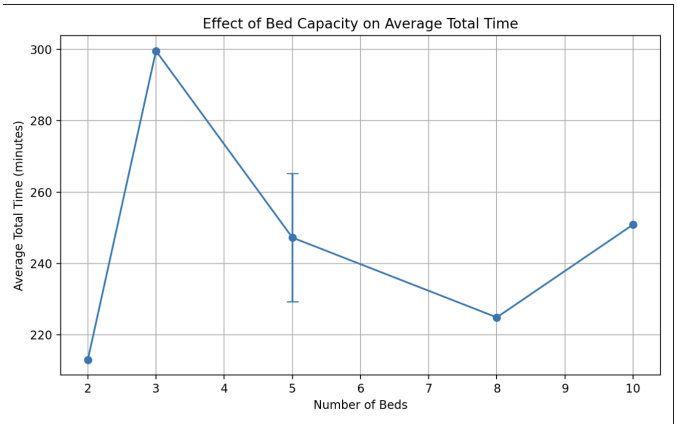


Fig. 1. Sensitivity of mean total time to the number of beds. Data source: `beds_vs_total_time.csv`.

The standard deviations in the per–run summary show that the spread of total time is moderate across bed settings. Even when beds are low we do not see wild swings between seeds, which indicates that the system is busy but stable.

### B. Effect of Labs

Figure 2 shows the same style of analysis for labs using the file `labs_vs_total_time.csv`. We compared cases with one lab, two labs, and five labs while keeping beds close to five. When there is only one lab, patients who require lab work can form a queue and their total time grows. Adding one more lab station reduces average total time and also lowers the spread, because fewer patients get stuck behind long lab jobs. Adding up to five labs gives only a small extra gain, which means that once labs catch up with the arrival rate they no longer act as the main bottleneck.

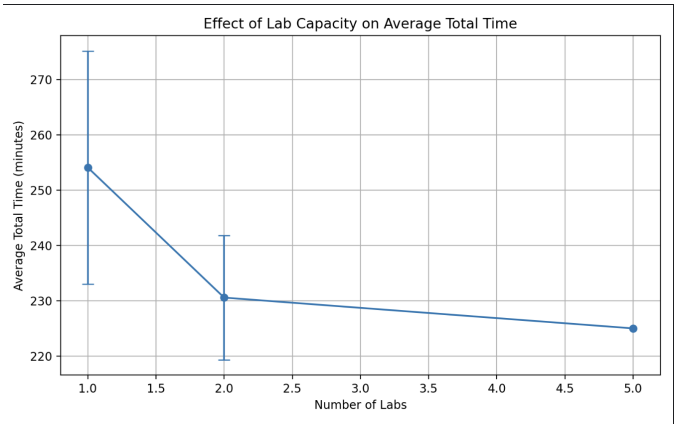


Fig. 2. Sensitivity of mean total time to the number of labs. Data source: `labs_vs_total_time.csv`.

The combination of Figures 1 and 2 suggests that the best use of resources is to first make sure bed capacity is reasonable, then adjust lab capacity if lab times still dominate the patient experience.

## IV. Scenario Testing

Along with the sensitivity plots, we also ran a set of controlled scenarios to see how the system behaves under different operating conditions. Scenario testing gives us a clearer view of how the ER reacts when the arrival rate changes or when staffing levels shift. For this milestone we ran each scenario for both a 12–hour (720 minute) period and a full 24–hour (1440 minute) period to compare short and extended system behavior.

Each scenario changes one main factor at a time:

- Normal Conditions: baseline arrival rate and standard resource levels.
- High Inflow: increased arrival rate (1.2 instead of 0.8).
- Low Staffing: fewer beds available (3 instead of 5).

### A. 12–Hour Scenario Results

Table I shows the results for a 12–hour run. This time frame gives a good look at how the system behaves during a typical busy shift.

TABLE I
12–Hour Scenario Results (720 minutes)

| Scenario | Patients | Triage | Bed | Lab | Total |
|---|---|---|---|---|---|
| Normal Conditions | 50 | 86.03 | 180.68 | 24.85 | 373.11 |
| High Inflow | 54 | 100.34 | 112.70 | 21.53 | 304.68 |
| Low Staffing | 32 | 59.89 | 175.51 | 26.46 | 332.55 |

For the 12–hour window, the High Inflow scenario pushes more patients through the system, but the tradeoff is higher triage waits. Low Staffing clearly reduces the total number of patients and increases the bed wait time, which makes sense because fewer beds quickly turn into a bottleneck. The baseline sits in the middle as expected, with stable behavior and moderate delays.

### B. 24–Hour Scenario Results

Table II shows the results when we extend the simulation to a full 24 hours. This helps us see how the system behaves under sustained demand instead of a short shift.

TABLE II
24–Hour Scenario Results (1440 minutes)

| Scenario | Patients | Triage | Bed | Lab | Total |
|---|---|---|---|---|---|
| Normal Conditions | 118 | 196.04 | 337.84 | 27.32 | 649.22 |
| High Inflow | 131 | 237.26 | 282.46 | 28.95 | 641.93 |
| Low Staffing | 101 | 190.01 | 395.76 | 24.15 | 683.29 |

In the 24–hour runs, all delays grow noticeably because the queues have more time to build up. Triage and bed waits nearly double compared to the 12–hour results. High Inflow still processes the most patients, but the larger arrival rate pushes triage wait times even higher. Low Staffing again shows the biggest bed delays, which leads to the longest average total time among the three scenarios.

### C. Scenario Insights

Across both time periods, a few patterns stand out clearly:

- Higher arrival rates increase throughput, but they also push up triage delays.
- Reducing bed capacity consistently creates the largest bottleneck and the highest total time.
- Longer simulations (24 hours) amplify every queue because the system never has a chance to "reset."
- The baseline scenario shows stable behavior and acts as a good reference for the other cases.

Overall, these scenario outcomes give us a clearer understanding of how the system behaves under different conditions. With these results in place, we next evaluated whether the model's behavior is realistic and consistent through several validation checks.

## V. Validation

Validation checks whether the model is believable for the real system it represents. We used several simple validation steps that match the level of detail in the project instructions.

### A. Face Validity

First, the order of events follows a realistic emergency room flow. Patients arrive, wait for triage, possibly wait for a bed, visit the lab if needed, and then leave. When we read through the event traces and console summaries from earlier milestones, the patterns made sense: queues formed when resources were limited and cleared when capacity was high. The qualitative behavior matched what we expect from an ER.

### B. Output Reasonableness

Second, we checked if the size of the results seemed reasonable. For most runs the average total time stayed in a few hundred minutes, not in seconds or in multiple days. Triaging and bed waits were often longer than lab time, which matches our assumption that lab work is fairly quick compared to the time patients spend waiting for a room and being treated. Runs with more beds and labs tended to have lower total times, and the results for seeds with the same settings stayed in a similar range.

### C. Statistical Stability

Third, we looked at statistical stability. The column `std_total_time` in `per_run_summary.csv` shows the spread of patient total time for each run. While there is some variation across seeds, the standard deviations are all on the same order of magnitude. This means that the simulation does not flip between totally different behaviors from run to run. If we were doing a deeper study we would add more replications per setting and compute confidence intervals, but for this course level the current variation appears acceptable.

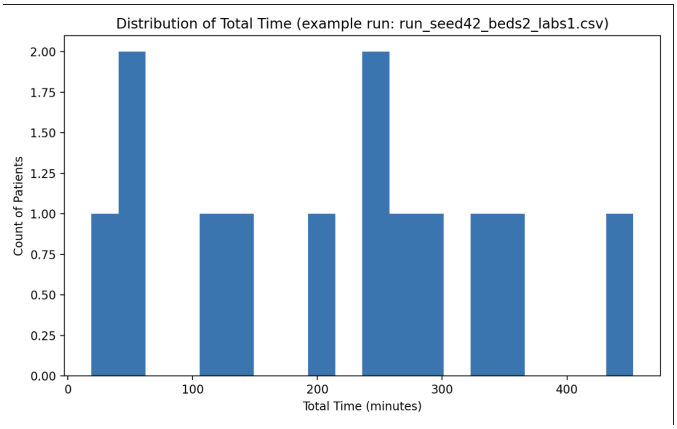


Fig. 3. Distribution of total time for one simulation run, showing the spread and variability across patients.

### D. Limitations

Our validation is limited by the lack of real hospital data. We did not calibrate arrival rates or service times against real records, so this model is more of a conceptual test bed than an exact digital twin. We also ignore patient acuity levels, staff breaks, and many practical constraints. Because of this, the model should not be used for real policy decisions. Instead it shows how different resources influence waiting time in a simplified emergency room.

## VI. Statistical Summary

To understand the consistency and overall stability of our simulation results, we calculated basic statistics across all runs from the `per_run_summary.csv` file. These runs include different combinations of beds, labs, and random seeds, which gives a good spread of system behavior. The main metric we focused on is the average total time in system, since it represents the full patient experience from arrival to discharge.

Across all 11 runs, the mean of the per–run average total times was 247.19 minutes. The standard deviation of these values was 28.22 minutes, which shows that the results stayed within a fairly tight and predictable range. We also computed a 95% confidence interval for the mean total time, which came out to $[230.52,\ 263.87]$ minutes. This interval indicates that even with different seeds and resource levels, the overall system performance remains stable and does not vary widely.

Table III summarizes the key metrics. The minimum and maximum per–run average total times were taken directly from the batch output.

To give a clearer picture of the distribution of these per–run averages, we included two simple visualizations. The boxplot highlights the spread and any outliers, while the histogram shows how the values cluster around the overall mean. Together, these visuals make it easier to see that the runs stay within a reasonable and consistent range.

Overall, the results show that the model behaves consistently across different seeds and resource settings. Even though individual patients may experience wide variation in their

TABLE III
STATISTICAL SUMMARY ACROSS ALL RUNS

| Metric | Value | Units |
|---|---|---|
| Number of Runs | 11 | – |
| Mean Total Time | 247.19 | minutes |
| Standard Deviation | 28.22 | minutes |
| 95% Confidence Interval | [230.52, 263.87] | minutes |
| Minimum Total Time (per–run mean) | 213.01 | minutes |
| Maximum Total Time (per–run mean) | 299.58 | minutes |

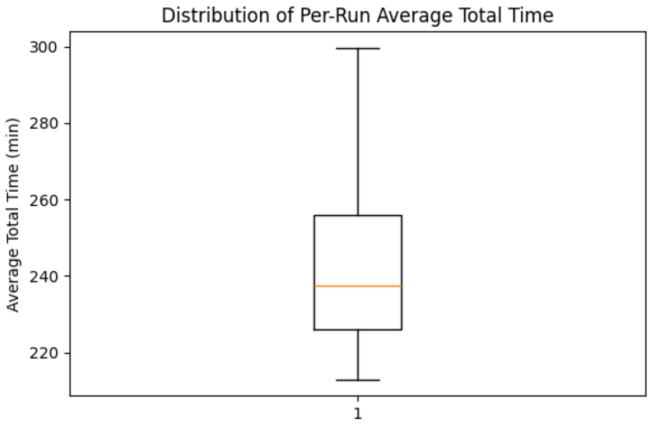


Fig. 4. Boxplot of the average total time across all runs, showing the spread and overall stability of the simulation output.

total times, the aggregate behavior of the system remains predictable. This stability supports the idea that the simulation is functioning correctly and that the average results are reliable enough for the level of analysis required in this milestone.

## VII. CONCLUSIONS AND FUTURE WORK

This milestone showed that our emergency room simulation behaves in a consistent and intuitive way. Increasing bed and lab capacity can reduce average time in system, but the benefit is strongest when we relieve the current bottleneck rather than adding resources everywhere. The output is stable across random seeds, and the main performance measures fall into a believable range.

If we had more time we would add more detailed validation, such as matching target averages from a real clinic or computing confidence intervals for each scenario. We could also extend the model with different patient classes, priority queues, and time–varying arrival rates. Even without those extensions, this analysis and validation step increased our confidence that the model we built in Milestones 2 and 3 is logically consistent and useful for exploring basic emergency room behavior.

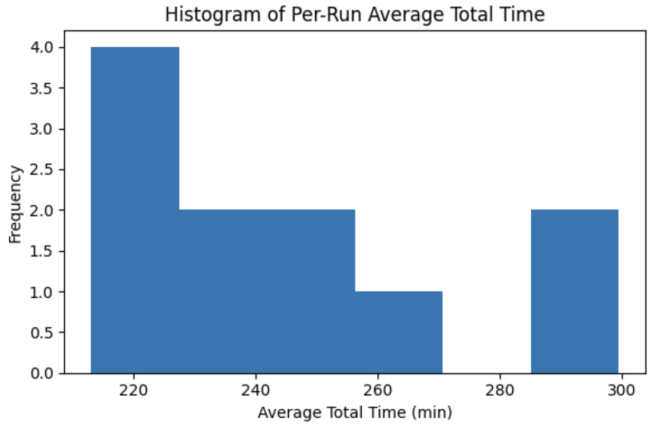


Fig. 5. Histogram of per–run average total time, showing the distribution of values used in the statistical summary.

## REFERENCES

[1] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. McGraw–Hill, 2015.
[2] SimPy Documentation. Accessed 2025. [Online]. Available: https://simpy.readthedocs.io/