Emergency Room Discrete-Event Simulation: Milestone 3 – Complete Implementation and Testing

Aashna Suthar
Kennesaw State University
asuthar@students.kennesaw.edu

Ash Arul
Kennesaw State University
aarul@students.kennesaw.edu

***Abstract*—We built a small discrete-event simulation of an emergency room using Python and SimPy. The model includes triage, treatment beds, and an optional lab stage. We ran multiple experiments by changing the number of beds, the number of lab stations, and other inputs. For each run we saved per-patient records to CSV and then merged all runs into a single file for analysis. Results show that adding beds reduces bed wait time and that lab capacity matters more as the arrival rate increases.**

***Index Terms*—Discrete-event simulation, healthcare, SimPy, queueing, ER flow, Python**

I. Introduction

The goal for Milestone 3 is a working ER prototype with comprehensive testing and a clean data pipeline. Our model keeps the flow simple: patients arrive, wait for a bed, get triage and treatment, may use lab, and then exit. This is enough to study how capacity affects delays.

II. Implementation

We used SimPy resources for beds and labs. Service times are basic random durations that keep the code readable for the class. The code is split into small files:

- `src/main.py`: parses arguments, runs the simulation, and exports CSV
- `src/models/triage.py`, `service_stations.py`, `arrivals.py`
- `src/sim_config.py`: default parameters (seed, minutes, arrival rate)
- `src/tests/test_sim.py`: quick smoke and a directional test

We run with the project venv and simple CLI flags. Example:

```
.\.venv\Scripts\python -m pip install -r requirements.txt
.\.venv\Scripts\python src\main.py --seed 42 --minutes 480 --lam 0.8 \
    --triage 1 --beds 5 --labs 1
```

Each run produces a CSV in `data_runs/` named like `run_seed42_beds5_labs1.csv`. We then merge runs into `data_runs/all_runs_master.csv`.

III. Experimental Setup

We tested across a small grid:

- Beds: {2, 5, 8, 10}
- Labs: {1, 2, 5}
- Minutes: 480 (8 hours) and 720 (12 hours) for smoother averages
- Seeds: 42–51 for repeatability
- Arrival rate λ: baseline 0.8; also 1.0 and 1.2 for heavier load

Each patient row stores: arrival time (row order), triage wait, bed wait, lab time, and total time. We also attach run-level fields: beds, labs, minutes, and seed.

TABLE I
Sample per-patient rows from the combined CSV.

able-format=3.2] S[table-format=3.2] S[table-format=3.2]
S[table-format=3.2] c c c c , table head=

File arrival triage_wait bed_wait lab_time total_time beds labs mins seed
, late after line= , late after last line=

]data$_r uns/all_r uns_m aster.csv Source_N ame =, arrival =, triage_w ait =, bed_w ait =,$

IV. Results

Table I shows a few sampled rows read directly from the merged CSV (we include the file in Overleaf at `data_runs/all_runs_master.csv`[1]).

We also computed simple averages by configuration (beds, labs, minutes). In general: more beds lowered bed wait; more labs helped once we raised λ. Longer runs (12 hours) gave smoother averages than 4 hours. These trends match what we expected.

V. Testing

We kept two quick tests in `src/tests/test_sim.py`.

- **Smoke test:** the model runs and returns records
- **Directional test:** bed wait with 8 beds is lower than with 2 beds

We ran: `.\.venv\Scripts\python -m pytest -q` and both passed.

[1]Upload your final combined CSV to `data_runs/` with the exact header: `Source_Name,arrival,triage_wait,bed_wait,lab_time,total_time,be`

VI. Notes on Submission

We had a short account access issue for one team member. We emailed Prof. Regan the same day and followed the instructions to turn in what we had and then finalize the missing files. The repository is public for review: https://github.com/asharulcoding/cs4632-er-sim.

VII. Conclusion

We finished a working ER simulation with data collection and testing. Capacity changes made the expected impact on delay, which suggests the model is behaving reasonably. Next we plan to clean up charts, try more realistic service-time distributions, and add clearer plots for each sweep.

References

[1] A. Matloff, “Introduction to Discrete-Event Simulation with SimPy,” online tutorial.

[2] IEEE, “IEEEtran LaTeX Class,” https://www.ieee.org/, accessed 2025.