Name: Hemen “Hemu” Babis
Email: hemu@pdx.edu
GitHub Repo: https://github.com/hemen-babis/rustqueuesim

Rust Queue Simulation

1. Project Name

Rust Queue Simulation - A Queueing System Simulation in Rust

2. Project Topic Area

Instrumented Simulation • Queueing Theory • Applied Mathematics • Computer Science • Stochastic Processes

3. Project Vision

RustQueueSim is a discrete-time simulation of a real queueing system (eg., checkout line, call center, hospital triage, CPU job queue). The simulation models how jobs arrive, wait, and are processed by a server, and it records detailed instrumentation data over time.

Core behaviors:

- Jobs arrive at random intervals using a probability distribution.
- A server processes one job at a time with a configurable service duration.
- A queue (FIFO) holds all waiting jobs.
- The system advances in discrete time steps.

Instrumentation logged each time step:

- Current queue length
- Average wait time (rolling + overall)

- Number of jobs completed
- Server status (busy or idle)
- Server utilization percentage
- Throughput rate
- Peak congestion moments
- Total time spent waiting

Implementation plan:

The project will be implemented as a Rust binary crate with well-organized modules:

- `job.rs` — job structure (arrival time, service time)
- `server.rs` — server state machine (Idle, Busy)
- `queue.rs` — FIFO queue wrapper (VecDeque)
- `sim.rs` — simulation engine controlling time steps
- `metrics.rs` — instrumentation and analysis
- `main.rs` — driver application

The project will include:

- Rustdoc for all public interfaces
- Unit tests for queue logic and service scheduling
- No rustfmt/clippy warnings
- A README describing the design, results, and lessons learned

This project fits the instrumented simulation theme perfectly, demonstrating a dynamic system that evolves over time while being continuously monitored.

4. Issues of Concern

Possible challenges include:

- Ensuring random arrival/service times feel realistic
- Designing a clean, maintainable simulation loop
- Keeping metrics accurate as job count grows
- Avoiding overcomplication while maintaining mathematical correctness

Optional extensions (if time permits):

- Multiple servers (M/M/c model)
- Priority queues (e.g., ER triage)
- Different arrival distributions (Poisson, uniform, custom)
- ASCII visualization

Base version will be simple, correct, and computationally efficient.

5. Repository

The project repository is available at:
https://github.com/hemen-babis/rustqueuesim

The repository includes:

- `proposal.pdf` (this document)
- `README.md` explaining the project and results
- Rust source code organized into modules

- MIT + Apache 2.0 licensing