**Project 2 to do list**

|  |  |  |
| --- | --- | --- |
| **1. Build & verify a *single* IPP/M/1 simulator** | Implement one-server queue fed by a two-state Interrupted-Poisson-Process and check that the program runs without errors. |  |
| **2. Check two analytic corner-cases** | *a)* Let ω₁ ≫ ω₂ so the source is almost always **ON**; performance should match the textbook M/M/1 formulas. *b)* Collect raw inter-arrival times from a lone IPP and confirm they follow a hyper-exponential (high-variance) law. |  |
| **3. Extend to a superposition of *K* identical IPPs** | Replace the single arrival generator by the minimum-of-K next-arrival times so we can study traffic aggregation effects. |  |
| **4. Measure core performance metrics** | Record and later report the empirical **waiting-time (Wₛ, W\_q)** and **queue-length (Lₛ, L\_q)** distributions, which are the “primary quantities of interest” in the brief. |  |
| **5. Add at least two insightful secondary metrics** | Look at **busy-period statistics** and **burstiness (variance-to-mean) index.** e.g. number of busy periods and their mean and variance. Burstiness refers to the spread/”clumpiness” of arrivals. Do arrivals come in chunks or are thye more spread out? |  |
| **6. Run a sensitivity (ρ-) analysis** | Vary the offered load ρ = λ\_eff / μ across several levels “to study different load scenarios” and plot/compare how W\_q and L\_q respond. |  |
| **7. Document validation & verification steps** | Briefly explain how each check in steps 2 & 4 increases confidence that the written code is correct (“verification”) and that the model answers the intended question (“validation”), read: actually models the desired process. |  |
| **8. (Optional extensions)** | If time permits, explore finite-capacity queues and report blocking probabilities, as suggested in “Possible extensions” |  |