**ACADEMIC TASK**

**CSE316**

(OPERATING SYSTEM)

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by:

Name: Prashant Kumar Jha

Registration number: 12314682

Roll No.: 18

Section: K23DX

**Submitted to : Mr. Harsh Sharma**

**………………………………………..**



**LOVELY PROFESSIONAL UNIVERSITY**

Real-Time Multi-threaded Application Simulator

Project Description

* In modern computing systems, multithreading is a fundamental concept that allows multiple threads to run concurrently within a process. Understanding how threads are managed, scheduled, and synchronized is essential for developers, especially in fields related to operating systems, parallel processing, and system programming. This project aims to develop a Real-Time Multi-threaded Application Simulator that visually represents the behaviour of different threading models and synchronization techniques. The simulator offers an interactive, graphical interface to aid in learning and comprehension of multithreading concepts.

Objectives

* To simulate and visualize various multithreading models like Many-to-One, One-to-Many, and Many-to-Many.
* To demonstrate thread lifecycle transitions: Ready -> Running -> Waiting -> Terminated.
* To implement synchronization mechanisms using Semaphores and Monitors. - To provide a real-time graphical interface to show thread interactions and CPU scheduling. - To create an educational tool that helps students and developers understand concurrent programming.

Objectives

Threading Models Simulated

* Many-to-One Multiple user-level threads are mapped to a single kernel thread. Only one thread executes at a time, leading to inefficient CPU usage.
* One-to-Many A single user-level thread is mapped to multiple kernel threads. This model is rare but useful for understanding limitations of thread management.
* Many-to-Many Multiple user-level threads are mapped to a pool of kernel threads. It combines the advantages of the previous two models and is suitable for high-performance systems.

Thread Synchronization Techniques

* Semaphores Used to control access to shared resources. They help avoid race conditions and ensure that critical sections are accessed by only one thread at a time.
* Monitors encapsulate mutual exclusion and condition synchronization, offering a structured way to write thread-safe code.

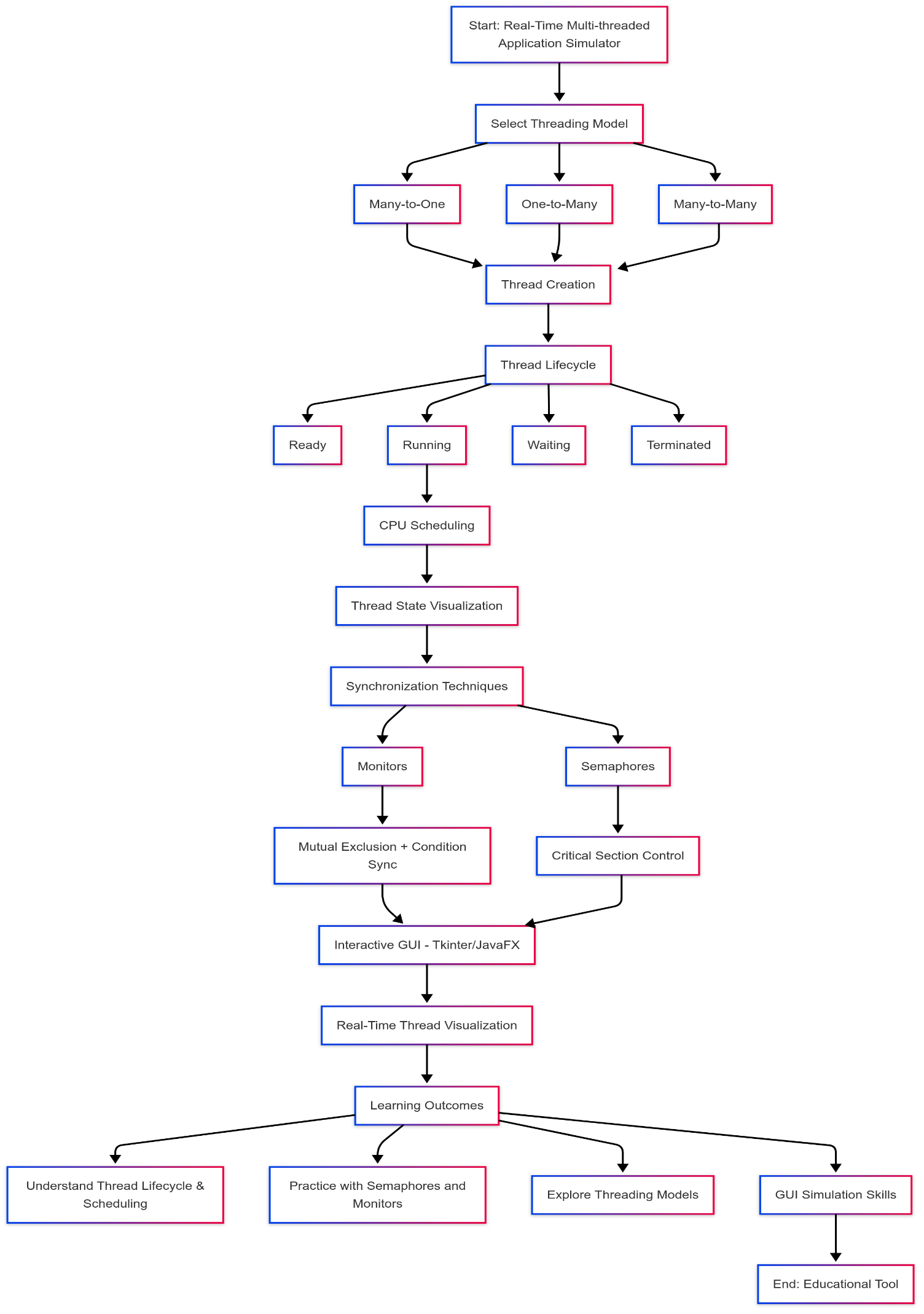
Features of the Simulator

* Real-Time Visualization: Graphically represents thread creation, execution, waiting, and termination.
* Interactive GUI: Built using tools like Tkinter (Python) or JavaFX (Java), enabling user interaction with simulation speed, thread count, and model selection.
* Thread State Indicators: Threads are color-coded or labeled based on their current state (e.g., green for running, yellow for waiting).
* CPU Scheduling Animation: Simulates how threads are picked by the CPU for execution in different models.

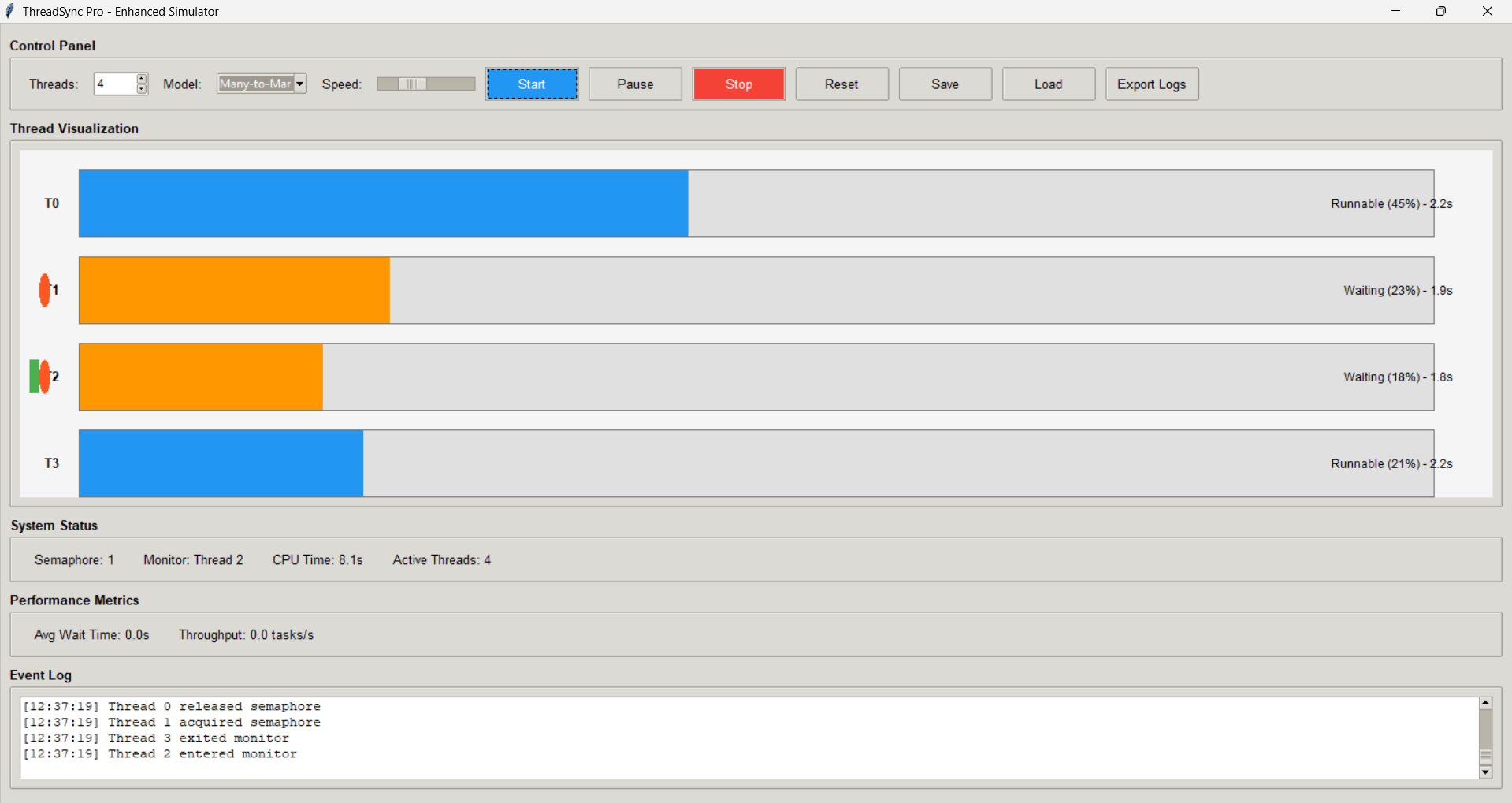
Technology Stack

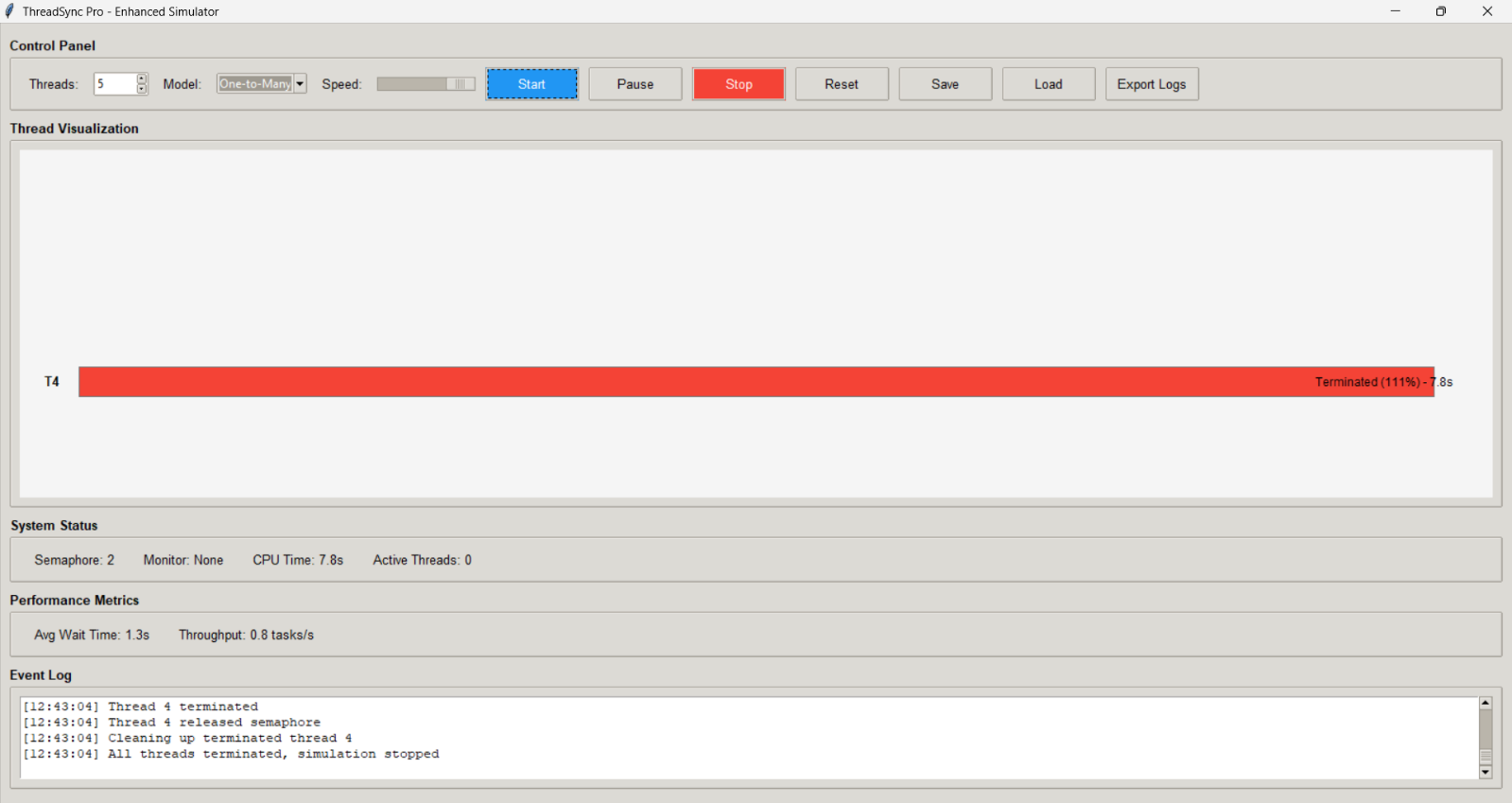
* Frontend / GUI: Python Tkinter or Java Swing/JavaFX
* Backend Logic: Python multithreading module or Java concurrency package
* Visualization: Custom canvas rendering of threads, queues, and CPU state
* Platform: Desktop Application

Flow of Program

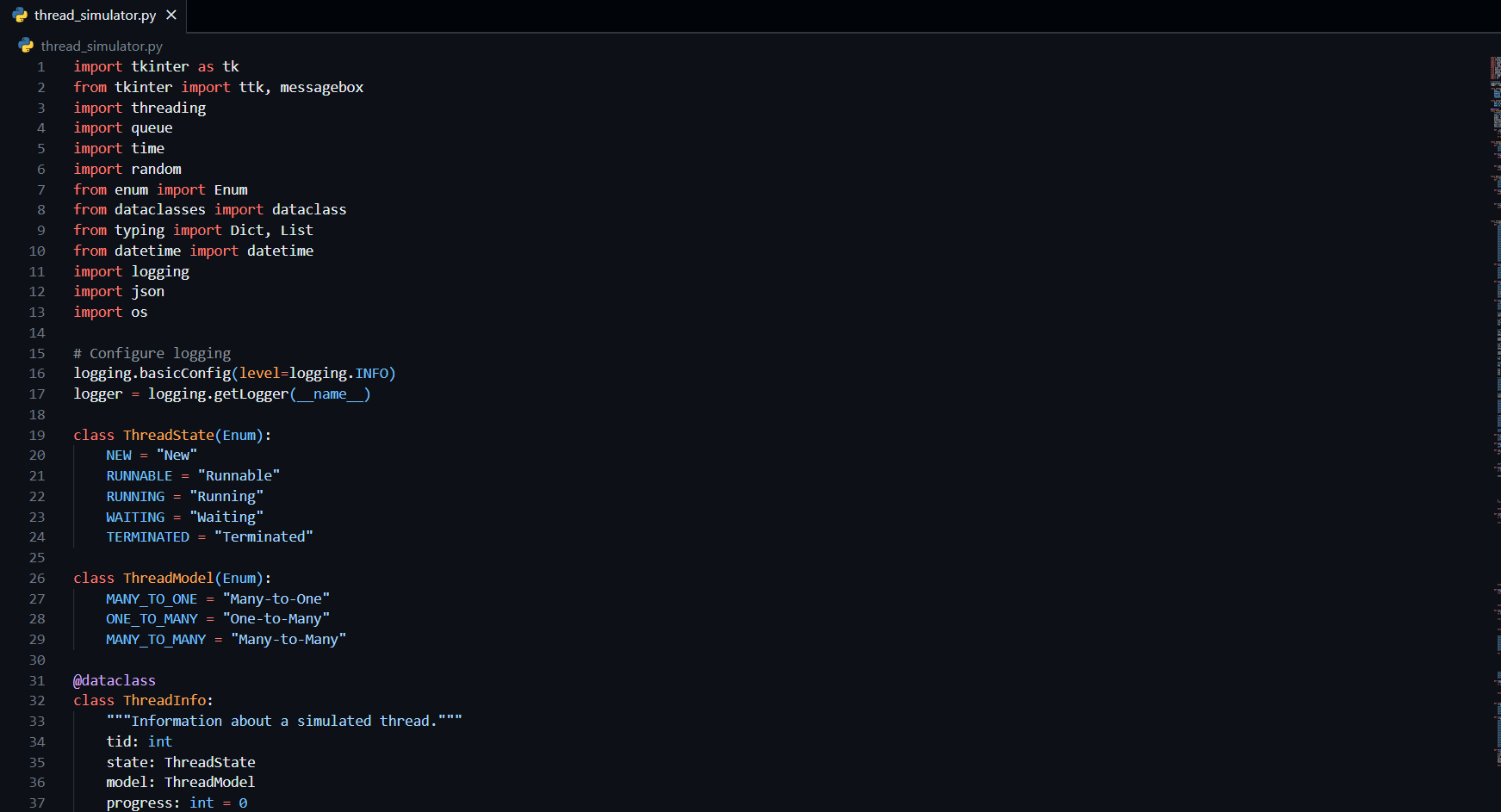


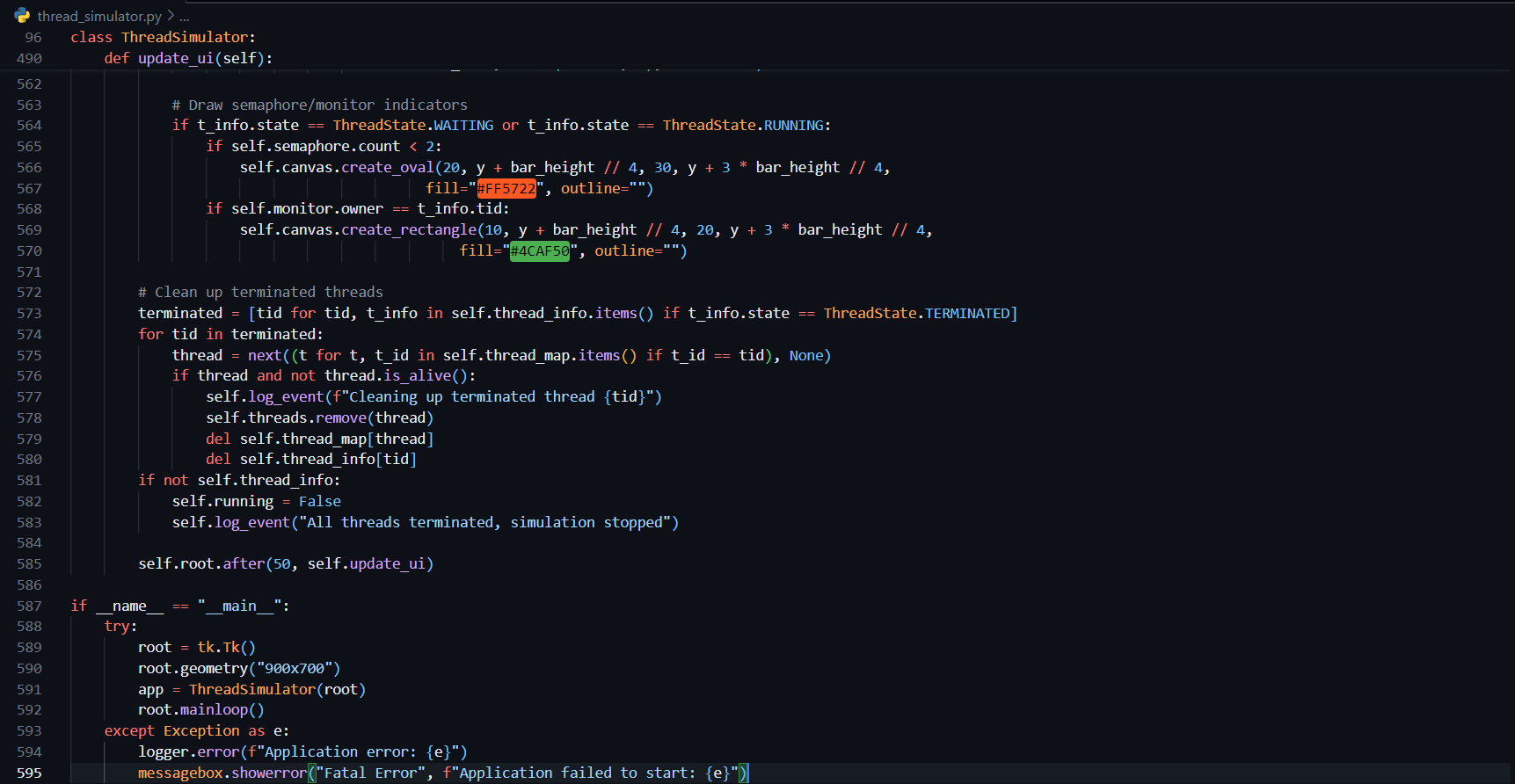
Output Screenshots





Code





Learning Outcomes

* Understanding thread lifecycle and CPU scheduling
* Gaining practical experience with synchronization primitives - Differentiating between threading models and their use cases
* Enhancing skills in GUI development for simulations

Conclusion

* The Real-Time Multi-threaded Application Simulator is a comprehensive learning tool that provides a visual and interactive way to understand the complexities of multithreading and synchronization in modern operating systems. By simulating real-world scenarios in a controlled environment, it bridges the gap between theoretical concepts and practical understanding.

Future Enhancements

* Integrate with **web-based GUI (React + Flask)** for broader accessibility
* Add **custom scheduling algorithms** (e.g., SJF, Priority Scheduling)
* Support **I/O-bound and CPU-bound simulation**
* Save simulation logs as reports (PDF, CSV)