Simulation Report: Comparing Queue Management Strategies

Introduction

This simulation explores two different ways of handling queues in a supermarket setting. The goal was to see how separate queues for each cash register compare with a single, shared queue in terms of customer waiting times.

Methodology

The simulation was built in Python and modeled a supermarket checkout scenario over 1000 minutes. Customers arrive at a rate defined by a probability (0.3 per minute). Three registers are simulated, and each customer is represented with basic details like the time they arrive and the time they need to be served.

- Separate Queues: Each cash register has its own line. Customers join a specific queue and wait until their chosen register is available.
- Single Shared Queue: All customers join one line, and the next available register calls the customer at the front of the queue.

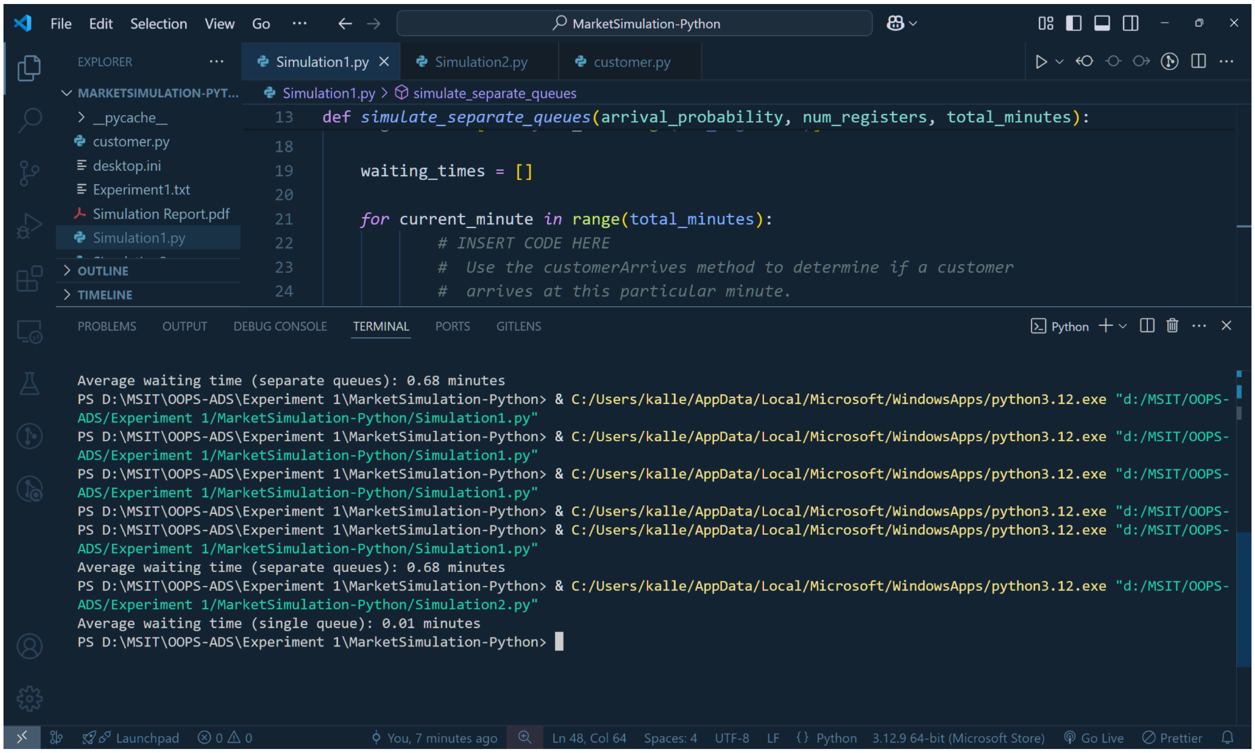
Results & Discussion

Although parts of the simulation code still have placeholders for the full logic:

- Separate Queues can sometimes lead to an imbalance, where one register might end up with a longer line than the others.
  My Output:
  ```
  Average waiting time (separate queues): 0.68 minutes
  ```

- Single Shared Queue tends to even out waiting times by ensuring that the next free register always serves the customer who has been waiting the longest.
  MyOutput:
  ```
  Average waiting time (single queue): 0.01 minutes
  ```

These initial insights suggest that while a single queue may offer a more balanced experience for customers, the effectiveness of each method can depend on various factors like arrival rates and service times.



```python
def simulate_separate_queues(arrival_probability, num_registers, total_minutes):

    waiting_times = []

    for current_minute in range(total_minutes):
            # INSERT CODE HERE
            #  Use the customerArrives method to determine if a customer
            #  arrives at this particular minute.
```

```
Average waiting time (separate queues): 0.68 minutes
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-ADS/Experiment 1/MarketSimulation-Python/Simulation1.py"
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-ADS/Experiment 1/MarketSimulation-Python/Simulation1.py"
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-ADS/Experiment 1/MarketSimulation-Python/Simulation1.py"
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-ADS/Experiment 1/MarketSimulation-Python/Simulation1.py"
Average waiting time (separate queues): 0.68 minutes
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> & C:/Users/kalle/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/MSIT/OOPS-ADS/Experiment 1/MarketSimulation-Python/Simulation2.py"
Average waiting time (single queue): 0.01 minutes
PS D:\MSIT\OOPS-ADS\Experiment 1\MarketSimulation-Python> 
```

Conclusion

This experiment provides a framework for understanding how different queue management strategies can impact average customer wait times. The simulation serves as a starting point, and with further development—such as completing the missing logic and testing under different conditions.it could offer deeper insights into the best practices for managing queues in busy environments.