CSC3002F Assignment 2 – Process Schedular Sarah the Barman
KSHKAV001

1. Project Overview

The objective of this project is to simulate and evaluate various CPU scheduling algorithms – First-Come-First Serve (FCFS), Shortest Job First (SJF) – non-pre-emptive and Round-Robin (RR). The algorithms are simulated in the context of a barman which acts as the CPU who makes drinks for patrons where each patron is treated as a separate process with a fixed number of jobs(drinks). The performance of the specified algorithms is evaluated based on various metrics including response time, waiting time, turnaround time, CPU utilisation time and throughput.

2. Solution Implementation

The solution was implemented in java by using System.currentTimeMillis() added to the program to get timing values for each process (patrons) which are further summarised to get mean, median and distribution. In the context of this simulation timing values are recorded as described in Table 1. Each drinks has a fixed preparation time which is taken as "CPU bursts" and each patron orders a fixed number of five drinks.

Performance metric	**Meaning in this simulation**
Response Time	Time from when a patron places the first drink order until the first order is received by the patron.
Waiting Time	Time from when a patron places a drink order until the barman starts preparing the order. This is taken as a total for all 5 drinks ordered the patron.
Turnaround Time	Time from when a patron places the first order until the patron is finished drinking the last drink.
CPU Utilisation	Time the CPU (barman) spends on preparing drinks as a fraction of the total simulation time.
Throughput	The number of processes completed (patrons) per fixed unit time (taken as 2000ms for this assignment).

Code description

- In the Patron.java three arrays are implemented to calculate times of order placed, order received and order drink complete. This is useful in calculating the response time, waiting time and turnaround time and log it in the TimingLog.java. All Patrons finish times are recorded in TimingLog.java for throughput calculations.
- In Barman.java the time when a barman starts working on a job till the job is complete is recorded – this is used to calculate the total time the barman spent preparing drinks within the TimingLog.java for CPU Utilisation.
- All processing for the metrics is done in the TimingLog.java. It records the total simulation time, all performance metrics for each patron in expeirments.csv and summary values across each run in experiments_summary.csv. The throughput is recorded by sorting the

collection of patron finish times and recording them over a window of 2000ms giving approximately 10 equal time intervals.

3. **Context Switching Time**

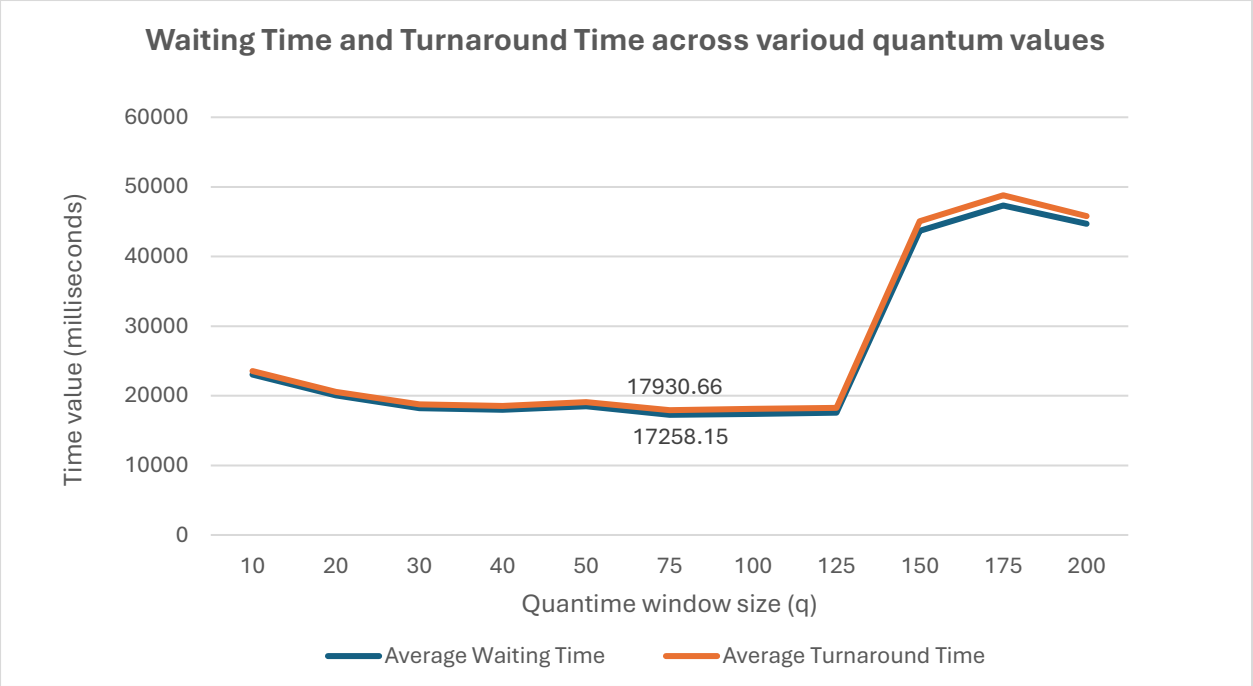
   The context switching time is the time spent between switching the processes (patrons) by the barman. This time is included in the total execution time as it represents the transition overhead of process switching. In a real-world simulation it would take a barman a few seconds to clean up between making drinks. Ideal context switching time is zero within an operating system. In this experiment context switching time, c, is takes as 3 based on the following reasoning:

   - It shows a value that realistically simulates the switching time in an operating system. Taking a time of zero would be too idealistic and taking a real-life context switching time would be meaningless as we are working with process scheduling with a CPU.
   - A very small value such as 1 would make context switching time very insignificant and a very large value would unrealistically penalise algorithms like Round Robin from an operating system perspective.
   - Through initial trial runs, we observed that **c = 3** provided a reasonable balance.

4. **Quantum Values for Round Robin Scheduling Algorithm**

   The round robin scheduling algorithm is highly dependent on the scheduling time quantum q which determines the maximum amount of time a patron can be served before the next patron is given a chance to be processes. To determine optimal values the RR algorithm was run with the same seed across different time quantum values ranging from 10 to 200. This range is based on the CPU burst time (preparation time of drinks) which ranges from 20 to 200. The best quantum value is considered as the one which results in at least 80 percent of the CPU bursts to be shorter than the time quantum.

   The optimal value for time quantum, q = 75 based on Graph 1 shown below. The value is chosen as it results in the lowest average turnaround time and lowest average waiting time per patron. **The number of patrons selected was fixed at 80 patrons.**



As can be seen from the graph and the data experiment the lowest average waiting time and turnaround time occurs at q=75. The lowest average waiting time = 17258.15 ms and lowest average turnaround time = 17930.66 ms. 73% (11 out of 15) drinks have a preparation time shorter than the time quantum and approximately 87% drinks fall within the time quantum (with prep time = 75).

5. Scheduling Algorithm Comparison

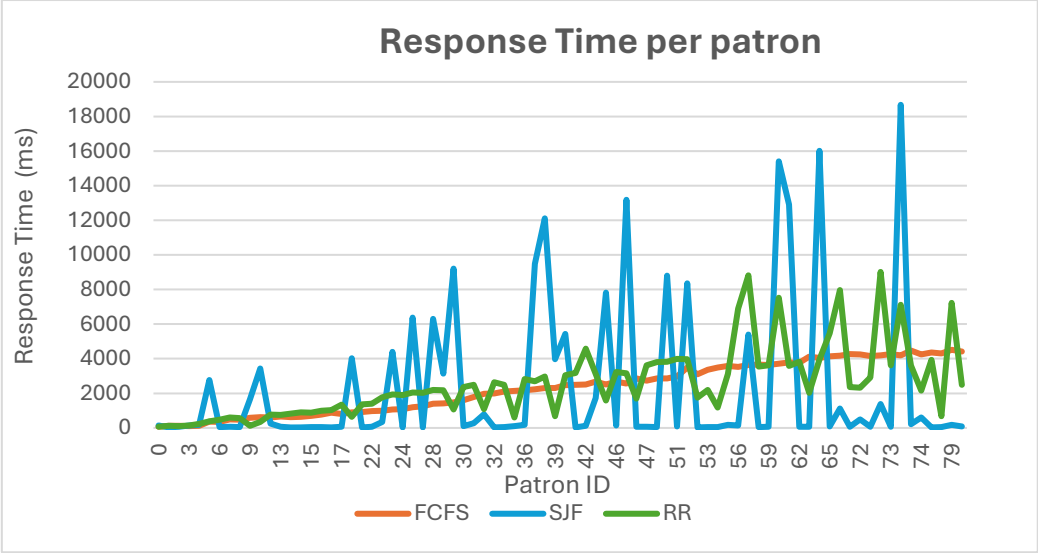
Multiple experiments were conducted to effectively compare the performance of the three scheduling algorithms across various metrics discussed in the section below. As per the sections mentioned above, the following values were fixed as:

a) Number of patrons = 80 patrons which gives thorough results and accounts for thorough experimentation without overloading the system.
b) Context switching time (c) = 3
c) Quantum value (q) = 75

While maintaining these fixed values the following experiments were conducted:
Experiment 1: Run the experiment with a constant seed (101 in this case) for all three scheduling algorithms. This helps in obtaining data across the different patrons in each case. CPU utilisation and Throughput is calculated for each algorithm.
Experiment 2: This experiment runs by conducting experiment 1 across 10 different seeds. In each run the average values of the waiting time, response time, turnaround time, CPU utilisation and throughput are calculated.

The results obtained for both experiments are analysed per metric as discussed below.

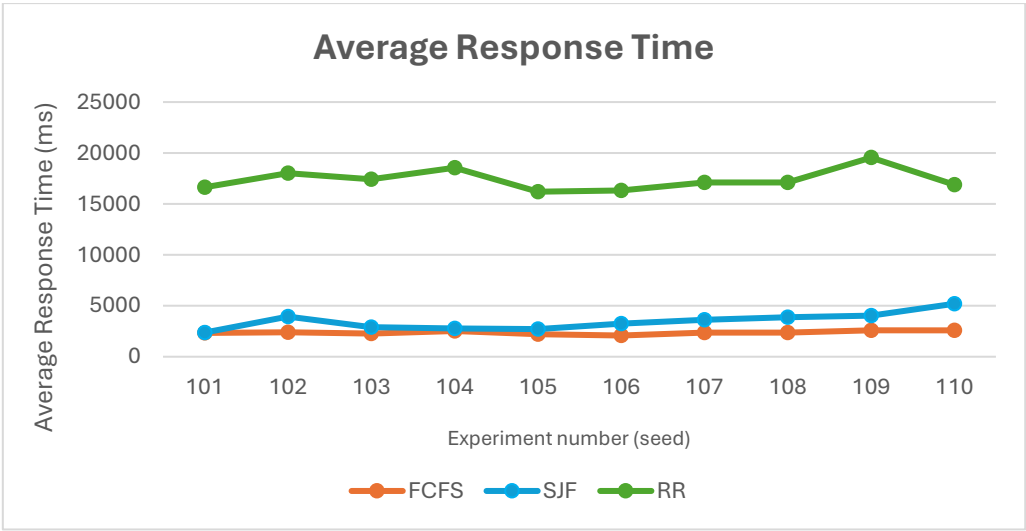
5.1 Response Time



Scheduling Algorithm	Average Time (ms)	Median Time (ms)	Std Deviation (ms)
FCFS	2320	79	1429
SJF	2371	59	4321
RR	2572	79	2055

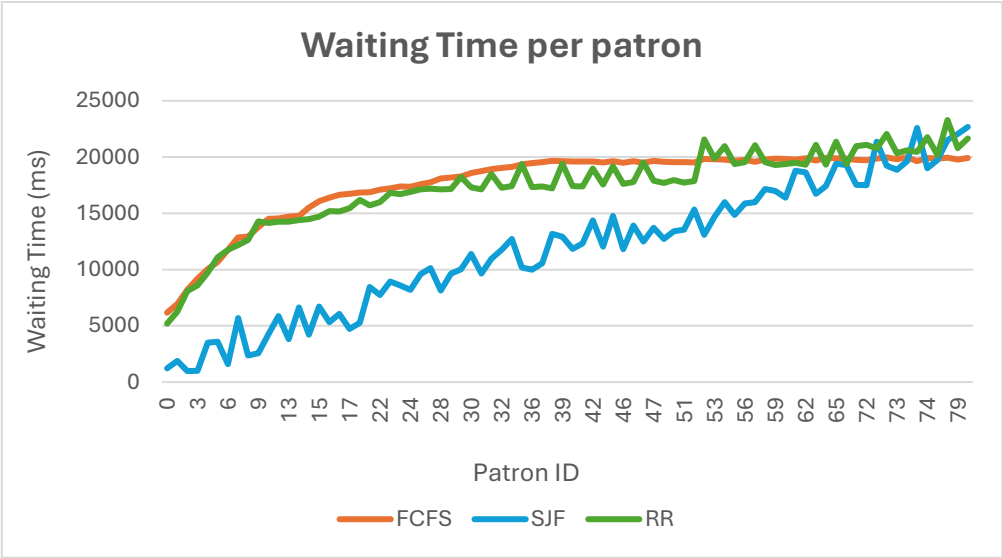
*Rounded off to whole numbers, ms = milliseconds

- As expected SJF has the lowest median response time and one of the lowest average response times along with FCFS. However, it is important to know that the response time varies quite a bit (highest std deviation) – as patrons who request drinks with very high execution time can be served much later as the drinks with the shortest execution time are prioritised.
- FCFS and RR also have relatively similar response time with FCFS being the lowest. RR however has higher time variance than FCFS due to time slicing.
- FCFS has a much shorter response times for patrons that arrive earlier, and response time significantly increases for patrons that arrive later as expected from its prioritises
- Despite SJFS providing the lower response time for certain patrons for overall lowest response time FCFS (average of 2320ms) can be considered the best algorithm.



- The averages taken across the different seed confirm the results discussed above. Overall the algorithms FCFS and SJF can be seen to be much more stable with lower variance across the response times compared to RR which varies between 15000 to 20000 ms.

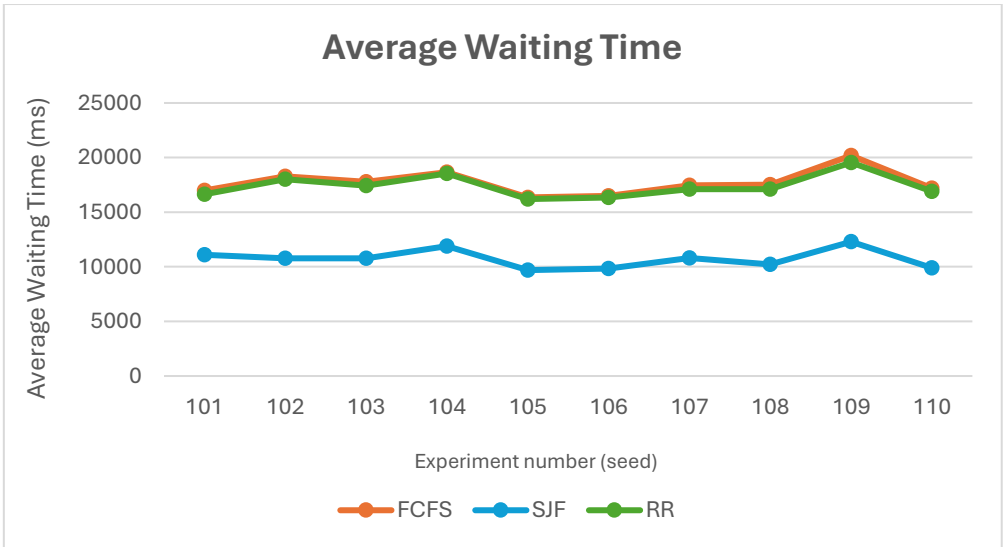
5.2 Waiting Time



Scheduling Algorithm	Average Time (ms)	Median Time (ms)	Std Deviation (ms)
FCFS	16980	4904	3270
SJF	11087	4845	5797
RR	16638	7634	3592

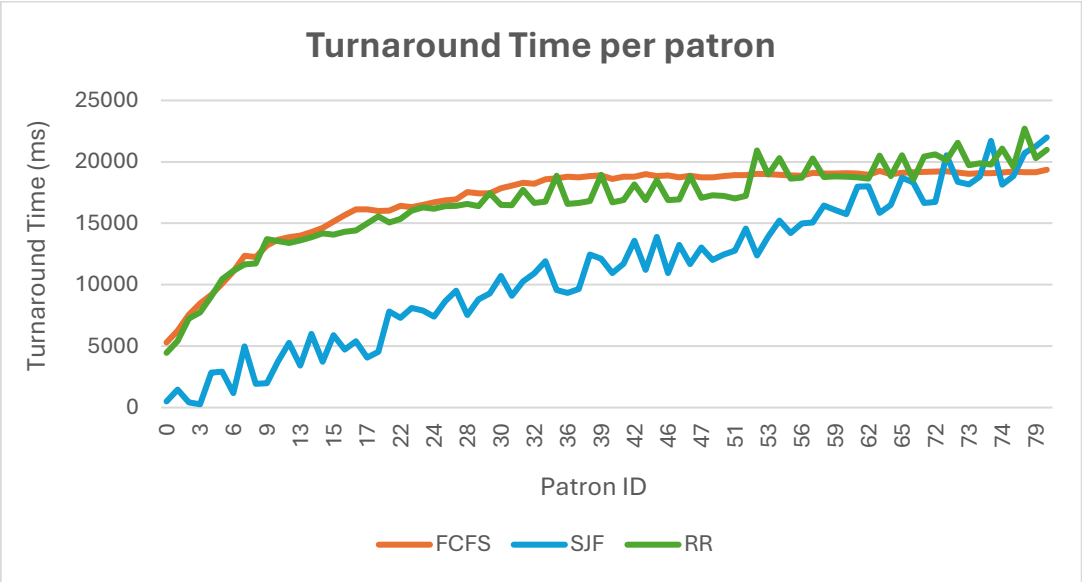
*Rounded off to whole numbers, ms = milliseconds

- SJF (non-pre-emptive) is expected to have the lowest average waiting times which is confirmed by the graph above. This is because it prioritises the drinks with the shortest execution times thus leading to overall faster completion and lower waiting times.
- FCFS and RR have significantly higher waiting times, however towards the end the last few patrons from ID number 62 have similar high waiting times across all algorithms.
- SJF has significantly higher waiting time towards the end contributing to its high variance (as seen in standard deviation).



- As seen in experiment 1 the values keep changing across the different experiments however there are not drastic changes thus confirming the stability of the experiments. Overall, it can be SJF remains the best algorithm for the shortest waiting times.

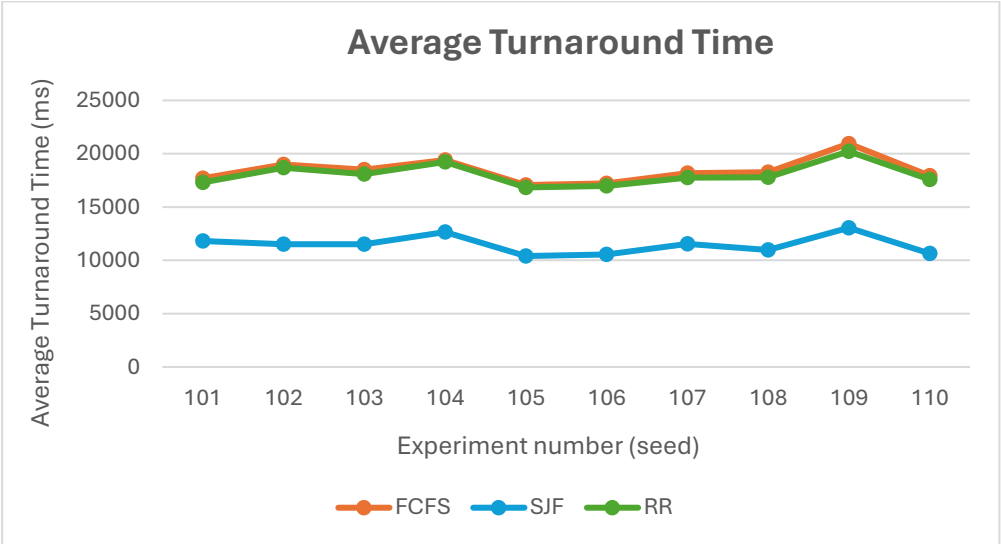
5.3 Turnaround Time



Scheduling Algorithm	Average Time (ms)	Median Time (ms)	Std Deviation (ms)
FCFS	17711	18862	3277
SJF	11818	11814	5878
RR	17307	17366	3569

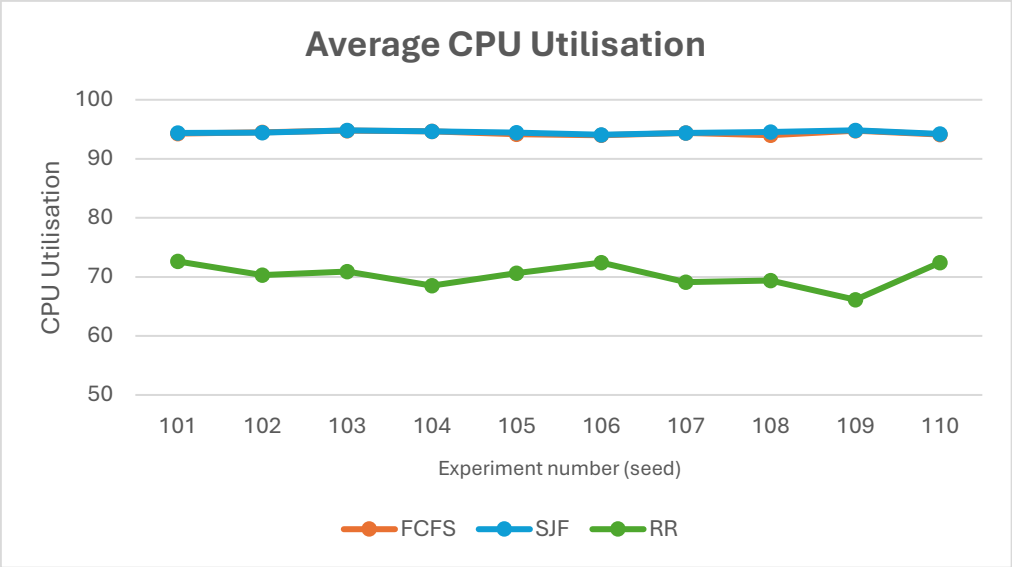
*rounded off to whole numbers, ms = milliseconds

- As expected, SJF gives the shortest turnaround time per patron as it minimises waiting time and prioritises jobs with shorter execution time thus providing an overall lower turnaround time as can also be seen in the graph.
- FCFS and RR similar to the waiting time analysis, have significantly higher turnaround times for the patrons that arrive much later. SJF follows a similar pattern as its significantly increases turnaround times with values similar to RR and FCFS for patrons with ID high than approximately 62 – resulting in its high standard deviation.



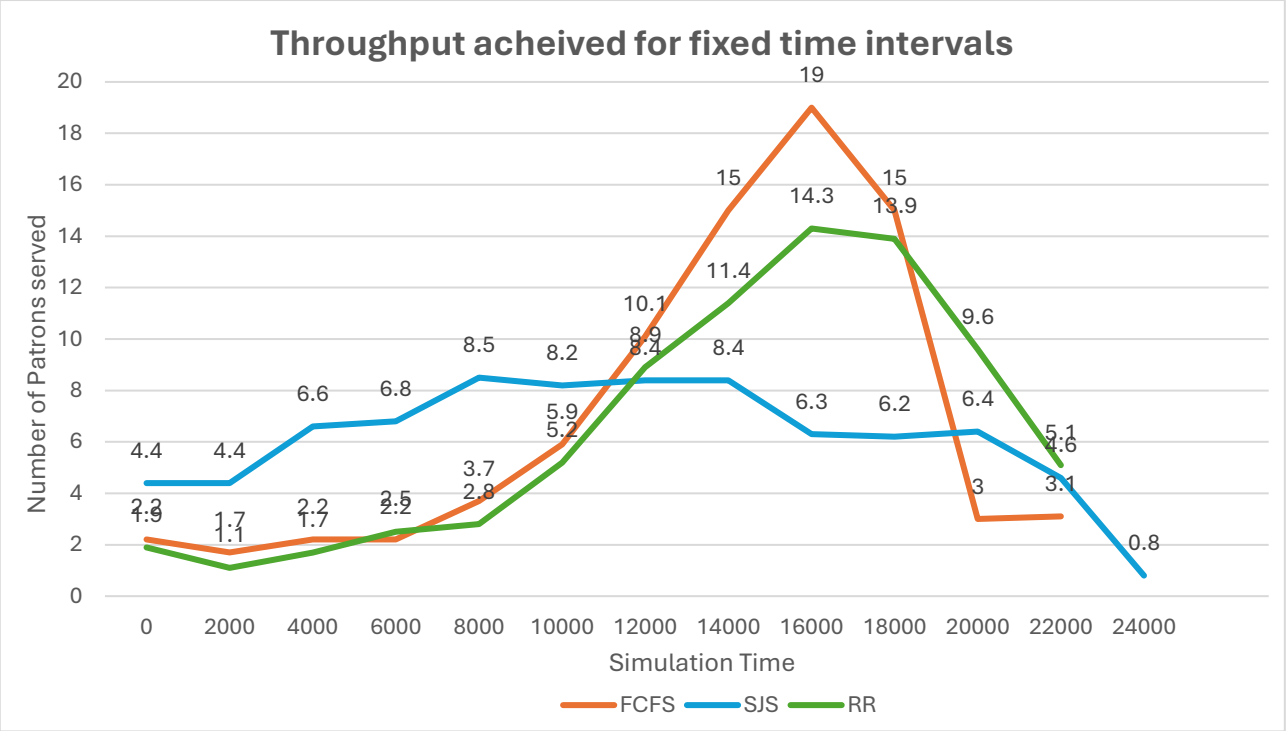
- Overall, across the experiments SJF has the lowest average turnaround time thus making it the best for this metric. The values fluctuate similar to the waiting times however remain relatively in the same range.

5.4 CPU Utilisation



- CPU utilisation is expected to be the highest when the CPU spends idle time or context switching time, therefore RR is expected to have the lowest CPU utilisation as is also evident from the results of our experimentation. SJF has the highest CPU utilisation while FCFS also has a very similar high CPU utilisation.
- As can be seen from the graph the CPU utilisation is between 65% to 74% for RR while it is much higher around 95% for FCFS and SJF.
- The CPU utilisation remains relatively stable for FCFS and SJF while it fluctuates quite a bit for RR.
- RR has the lowest CPU Utilisation due to its pre-emptive nature and context switch time.

5.5 Throughput



- The throughput remains the most constant and predictable for SJS where between 4-8 patrons are served approximately every 2000ms.
- Towards the middle of the simulation the throughput is the highest across all algorithms and comparatively FCFS has the highest throughput at 19 patrons served, followed by RR which has 14.3 patrons served while SJF is lowest at 6.3 patrons served.
- Throughput is highest in the middle because...

5.6 Fairness of algorithms

Fairness is the concept of equal CPU access to all processes (patrons). RR is generally known for being the fairest as it gives each patron a turn every quantum cycle. This can be seen in the response time and waiting time which is which the graph is a bit jagged. FCFS can also be considered fair as it prioritises the patrons that arrive earlier however is very unfair to patrons that arrive later resulting in high response times for patrons with high patron IDs.

5.7 Predictability of algorithms

Predictability refers to consistent behaviour across different runs which can be analysed using the variance in the results of the average runs. The table below shows variance values of the different metrics for each of the algorithms. The higher the variance the lower the predictability. The data is obtained from the results of experiment 2. The maximum(orange) and minimum(green) values are highlighted in the table.

Scheduler	Turnaround Time	Waiting Time	Response Time	CPU Utilisation
FCFS	1312863.98	1279890	26540.13	0.088366
RR	1113823.42	1101115	71475.95	4.16704
SJF	776929.884	751835.2	703219.7	0.058654

- Turnaround and waiting time is the most predictable for SJF and least predictable for FCFS. SJF makes it the best algorithm to predict how long a job would take.

- Response time is the most predictable for FCFS and least predictable for SJF. This makes is the best for patrons expecting a response as they can all expect similar behaviour when using FCFS which is good specifically in the barman context.
- CPU utilisation is the most predictable for SJF and least predictable for FCFS. SJF shows very stable performance while RR varies a lot due to time slicing and context switching.
- Overall RR is the least predictable making due to its operation.
- SJF can be overall ranked as the most predictable and stable algorithm except for its highly varying response times.

5.8 Starvation considerations

RR is optimised to prevent starvation while SJF is prone to it as it keeps processes with longer execution times waiting for very long. As can be seen it ha significantly high turnaround, waiting and response times towards patrons arriving much later – which can possibly result in starvation of these processes. RR is the safest against starvation as it ensures regular time slices while FCFS avoids time slices unless it become overloaded with too many processes arriving.

6. Conclusion

6.1 Best Algorithm Recommendation

6.2 Limitations to the process scheduling process