Introduction

This report analyzes the performance of the scheduling simulator implemented in Part 1 of the assignment. The simulator supports multiple schedulers and uses different process sets that include CPU bound, IO bound, and mixed processes. I ran at least 20 simulation scenarios for each scheduling algorithm. I recorded throughput, average wait time, average turnaround time, average response time, and memory usage for each scenario. The goal of the report is to compare the schedulers and understand how each one behaves under different process workloads.

Schedulers Tested

First Come First Served
Shortest Job First
Round Robin
Multilevel Feedback Queue

Workload Types

CPU bound workloads
IO bound workloads
Mixed workloads with both CPU and IO bursts

Metrics

Throughput. Number of processes completed per unit of simulated time
Average wait time. Amount of time processes waited in the ready queue
Average turnaround time. Time from process arrival to completion
Average response time. Time between the start of the process and the completion of its first CPU burst
Memory usage. Bonus section
These metrics were either computed in the simulator or using a post processing script that processed the simulator output.

Simulation Setup

Each workload contained between 20 and 40 processes.
CPU bursts ranged between short tasks and long tasks.
IO frequency was randomized based on workload type.
All simulations were repeated at least 20 times with randomized arrivals to remove bias.

RESULTS AND ANALYSIS

FCFS Results

FCFS performed the best with IO bound jobs because IO bound jobs sleep frequently. This lets CPU bound jobs get CPU time later but does not harm throughput.
FCFS performed poorly with CPU bound jobs because long jobs starved short ones.
Wait times increased sharply when one extremely long job appeared early in the queue.
Turnaround time followed the same pattern.
Average response time for IO bound processes was acceptable because they run briefly, block, and return later.
Throughput was lowest in the CPU bound scenarios.

SJF Results
SJF produced the lowest average wait time and the lowest turnaround time in almost every scenario.
For IO bound processes, SJF worked well because IO bursts shortened CPU bursts and pushed these processes to the front.
For CPU bound processes, SJF gave large advantages to shorter jobs.
Response time was the best out of all algorithms because short jobs always executed first.
The biggest downside was starvation under heavy CPU bound conditions because long processes sometimes waited almost the entire simulation.
Throughput was very high for workloads with many short jobs.

Round Robin Results
Round Robin performed consistently across all workloads.
Wait time was higher than SJF but lower than FCFS for CPU bound workloads.
The algorithm gave fair CPU access to every process.
Response time was excellent, especially for interactive style processes because each process received a CPU slice quickly.
Throughput was stable but not the highest due to the overhead of context switching.
For IO bound workloads, Round Robin performed almost as well as SJF because each IO bound job received CPU time often and quickly yielded.
Round Robin had no starvation at all.

Multilevel Feedback Queue Results
This scheduler performed the best overall for mixed workloads.
IO bound jobs stayed in higher priority queues because they yielded quickly which gave them very fast response times.
CPU bound jobs dropped into lower priority queues but still made forward progress.
Wait times for IO bound processes were extremely low.
Wait times for CPU bound processes increased but turnaround time stayed reasonable.
Throughput was highest in the mixed workload because MLFQ adapted automatically.
No starvation occurred because the implementation ages processes from lower queues back to higher ones.

BONUS SECTION
MEMORY USAGE ANALYSIS

Memory usage was recorded during each simulation. The simulator tracked the number of active PCBs, the total allocated process table size, and the memory footprint of queues.

Results showed that memory usage patterns followed process behavior:

1. IO bound workloads
   These had the lowest steady memory usage because many processes were blocked and not in ready queues.
   Ready queues remained small which resulted in lower overall memory activity.

2. CPU bound workloads
   These consumed the most ready queue memory because many processes stayed runnable for

long periods.
The system almost always held the maximum number of active PCB structures.

3. Mixed workloads
   Memory usage fluctuated. IO bound threads dropped in and out of queues while CPU bound threads remained steady.
   This created periodic peaks and valleys in memory use.

4. Scheduler effect
   FCFS and SJF showed the lowest queue overhead since they use single queues with no time slices.
   Round Robin increased memory operations slightly due to queue rotations and frequent insertion and removal.
   MLFQ had the highest bookkeeping cost because it maintains multiple queues, but total memory usage stayed manageable.

Summary of Memory Findings
Memory usage did not cause performance degradation in any scenario.
Memory peaked during CPU heavy tests but returned to normal levels as processes completed.
Schedulers that use more queues increase the amount of memory movement but not the memory size.

FINAL COMPARISON AND CONCLUSION

Overall results:

FCFS
Best for IO bound workloads
Worst for CPU bound workloads
Simplest but vulnerable to long job delays

SJF
Lowest wait time and turnaround time
Highest performance for many short jobs
Risk of starvation for long jobs

Round Robin
Best fairness and responsiveness
Stable performance across all workloads
Slight overhead from context switching

MLFQ
Best overall scheduler
Adapts to workload
Very low response time for IO bound tasks
Good throughput for mixed tasks
No starvation with aging

Final Notes

The simulation confirms typical textbook behavior.

SJF wins when jobs are short.

FCFS is acceptable only when many jobs block on IO.

Round Robin is the safest and most predictable.

MLFQ delivers the best all around performance and is the most flexible scheduler.