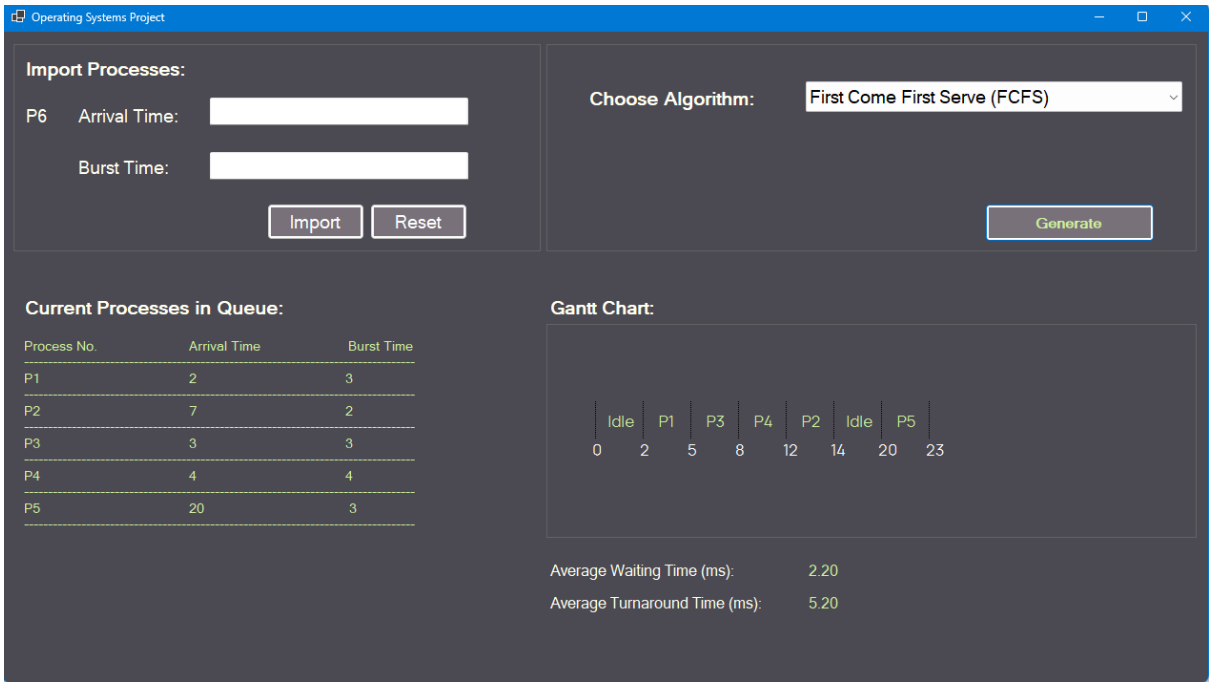
Operating Systems Project

Team Members:

- Sara Abdelraheem Hamed
- Abdelraheem Mohamed Alsadiq
- Ahmed Adel Abo Elwafa
- Ahmed Abdelrahman Shams

Supervised By: Dr. Ibrahim Shawky

1- First Come First Serve (FCFS)



FCFS (First-Come-First-Serve) scheduling algorithm is a non-preemptive scheduling algorithm that executes processes in the order in which they arrive in the ready queue. In this algorithm, the process that arrives first gets executed first.

Using FCFS scheduling, the processes will be executed in the following order:

Process 1 (P1) arrives at time 2 and has a burst time of 3. It will execute from time 2 to time 5.

Process 3 (P3) arrives at time 3 and has a burst time of 3. It will execute from time 5 to time 8.

Process 4 (P4) arrives at time 4 and has a burst time of 4. It will execute from time 8 to time 12.

Process 2 (P2) arrives at time 7 and has a burst time of 2. It will execute from time 12 to time 14.

Process 5 (P5) arrives at time 20 and has a burst time of 3. It will execute from time 14 to time 17.

The turnaround time for each process is the time it takes for the process to complete execution, i.e., the difference between the completion time and the arrival time. The waiting time for each process is the time the process spends waiting in the ready queue, i.e., the difference between the turnaround time and the burst time.

The turnaround time and waiting time for each process are as follows:

Process P1: Turnaround time = 5 - 2 = 3 Waiting time = 3 – 3 = 0

Process P2 Turnaround time = 14 - 7 = 7 Waiting time = 7 – 2 = 5

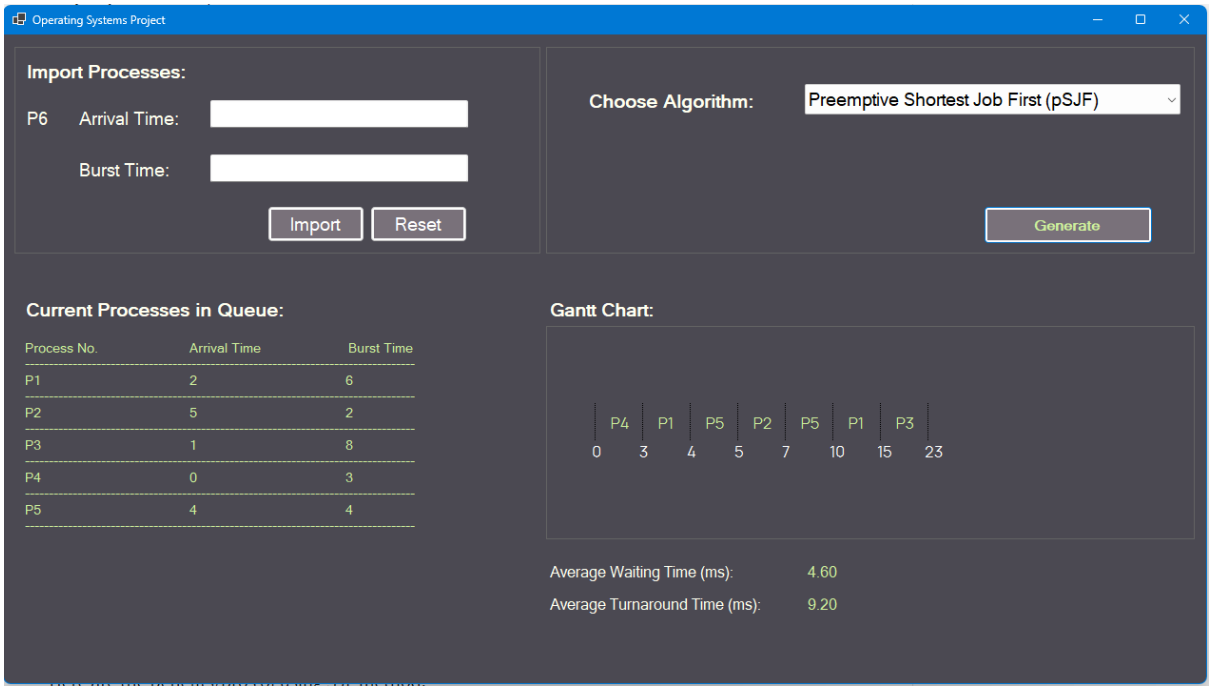
Process P3: Turnaround time = 8 - 3 = 5 Waiting time = 5 – 3 = 2

Process P4: Turnaround time = 12 - 4 = 8 Waiting time = 8 – 4 = 4

Process P5: Turnaround time = 20 – 17 = 3 Waiting time = 3 – 3 = 0

- The average turnaround time is: $(3 + 7 + 5 + 8 + 3)/5 = 5.2$
- The average waiting time is: $(0 + 5 + 2 + 4 + 0)/5 = 2.2$

2- Preemptive Shortest Job First (pSJF):



In Preemptive SJF Scheduling, jobs are put into the ready queue as they come. A process with shortest burst time begins execution. If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle:

- At time=0, P4 arrives and starts execution.
- At time= 1, Process P3 arrives. But, P4 has a shorter burst time. It will continue execution.
- At time = 2, process P1 arrives with burst time = 6. The burst time is more than that of P4. Hence, P4 will continue execution.
- At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is lower.
- At time = 4, process P5 will arrive. The burst time of P3, P5, and P1 is compared. Process P5 is executed because its burst time is lowest. Process P1 is preempted.
- At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.
- At time =6, P2 is executing.
- At time =7, P2 finishes its execution. The burst time of P1, P3, and P5 is compared. Process P5 is executed because its burst time is lesser.
- At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is executed because its burst time is less.
- At time =15, P1 finishes its execution. P3 is the only process left. It will start execution.
- At time =23, P3 finishes its execution.

The turnaround time and waiting time for each process are as follows:

Process P1: Turnaround time = 15 - 2 = 13 Waiting time = (3-2) + 6 = 7

Process P2: Turnaround time = 7 - 5 = 2 Waiting time = 5-5 = 0

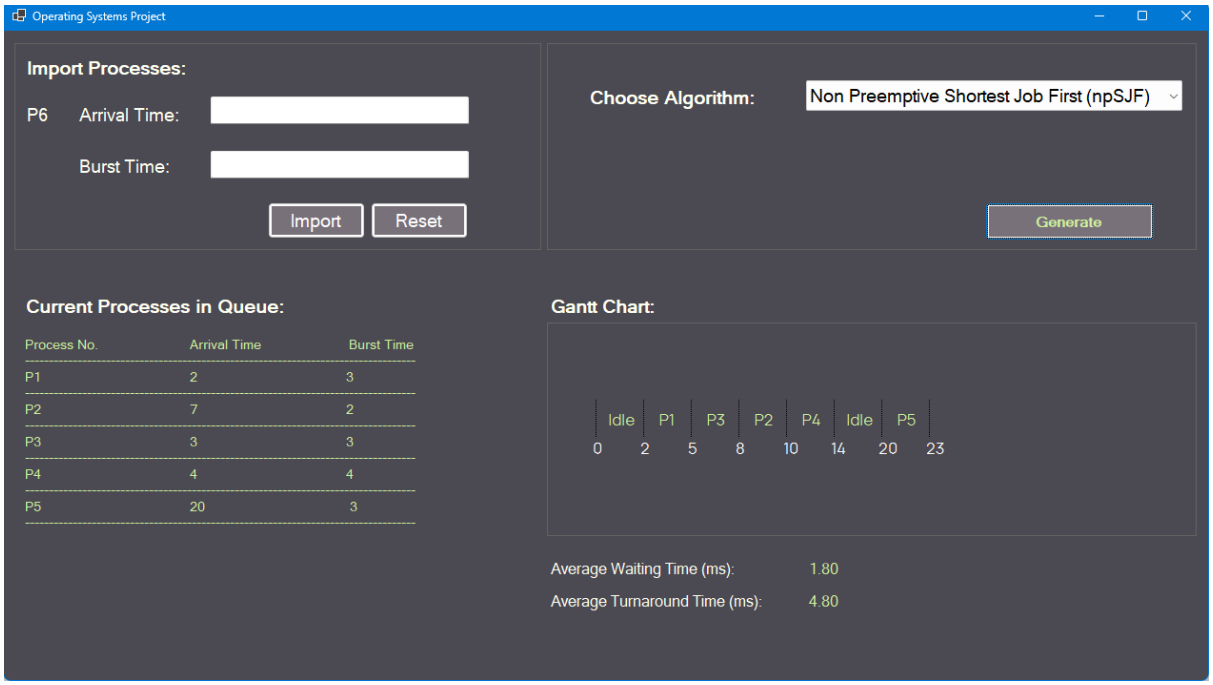
Process P3: Turnaround time = 23 - 1 = 22 Waiting time = 15 – 1 = 14

Process P4: Turnaround time = 3 - 0 = 3 Waiting time = 0 – 0 = 0

Process P5: Turnaround time = 10 - 4 = 6 Waiting time = (4-4) + 2 = 2

- The average waiting time is $(7 + 0 + 14 + 0 + 2)/5 = 4.6$.
- The average turnaround time is $(13 + 2 + 22 + 3 + 6)/5 = 9.2$.

3- Non-Preemptive Shortest Job First (npSJF):



NP-SJF (Non-Preemptive Shortest Job First) scheduling algorithm is a non-preemptive scheduling algorithm that selects the process with the shortest burst time from the ready queue to execute next. If multiple processes have the same burst time, the process that arrived first is executed first.

Using NP-SJF scheduling, the processes will be executed in the following order:

Process 1 (P1) arrives at time 2 and has a burst time of 3. It will execute from time 2 to time 5.

Process 3 (P3) arrives at time 3 and has a burst time of 3. It will execute from time 5 to time 8.

Process 2 (P2) arrives at time 7 and has a burst time of 2. It will execute from time 8 to time 10.

Process 4 (P4) arrives at time 4 and has a burst time of 4. It will execute from time 10 to time 14.

Process 5 (P5) arrives at time 20 and has a burst time of 3. It will execute from time 20 to time 23.

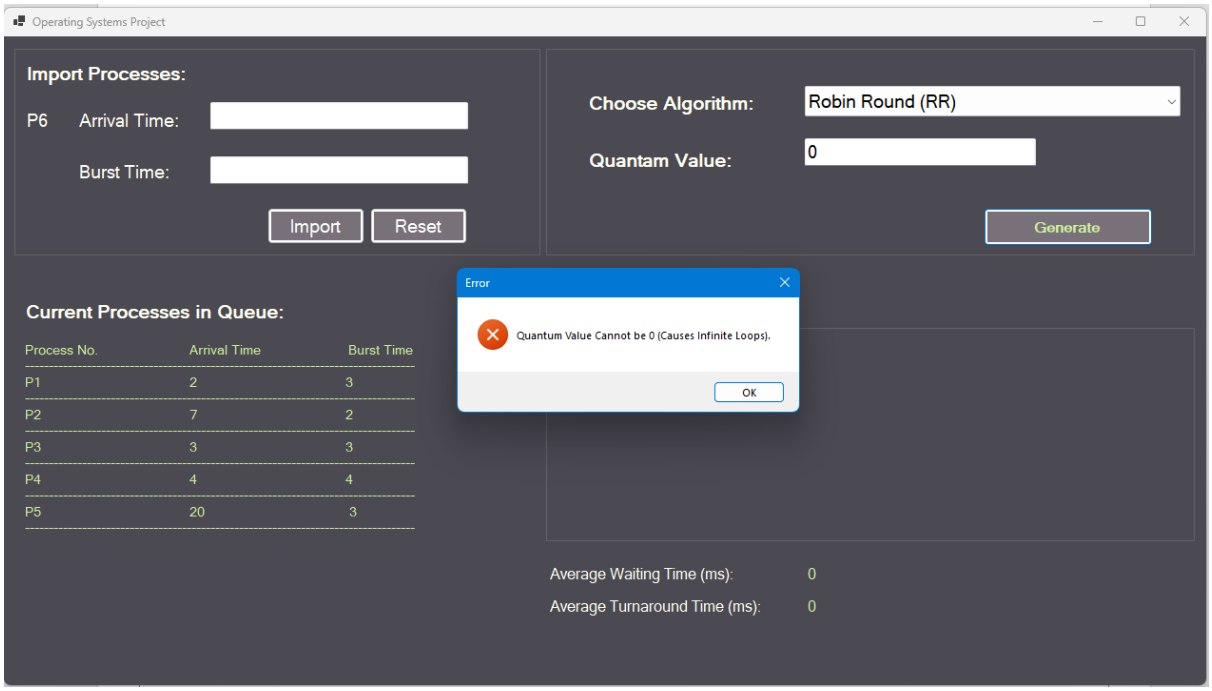
The turnaround time for each process is the time it takes for the process to complete execution, i.e., the difference between the completion time and the arrival time. The waiting time for each process is the time the process spends waiting in the ready queue, i.e., the difference between the turnaround time and the burst time.

The turnaround time and waiting time for each process are as follows:

Process P1:	Turnaround time = 5 - 2 = 3	Waiting time = 3 - 3 = 0
Process P2:	Turnaround time = 10 - 7 = 3	Waiting time = 3 – 2 = 1
Process P3:	Turnaround time = 8 - 3 = 5	Waiting time = 5 – 3 = 2
Process P4:	Turnaround time = 14 - 4 = 10	Waiting time = 10 – 4 = 6
Process P5:	Turnaround time = 20 - 17 = 3	Waiting time = 3 – 3 = 0

- The average waiting time is $(0 + 1 + 2 + 6 + 0)/5 = 1.8$.
- The average turnaround time is $(3 + 3 + 5 + 10 + 3)/5 = 4.8$.

4- Round Robin (RR) with 0 Quantum Value:



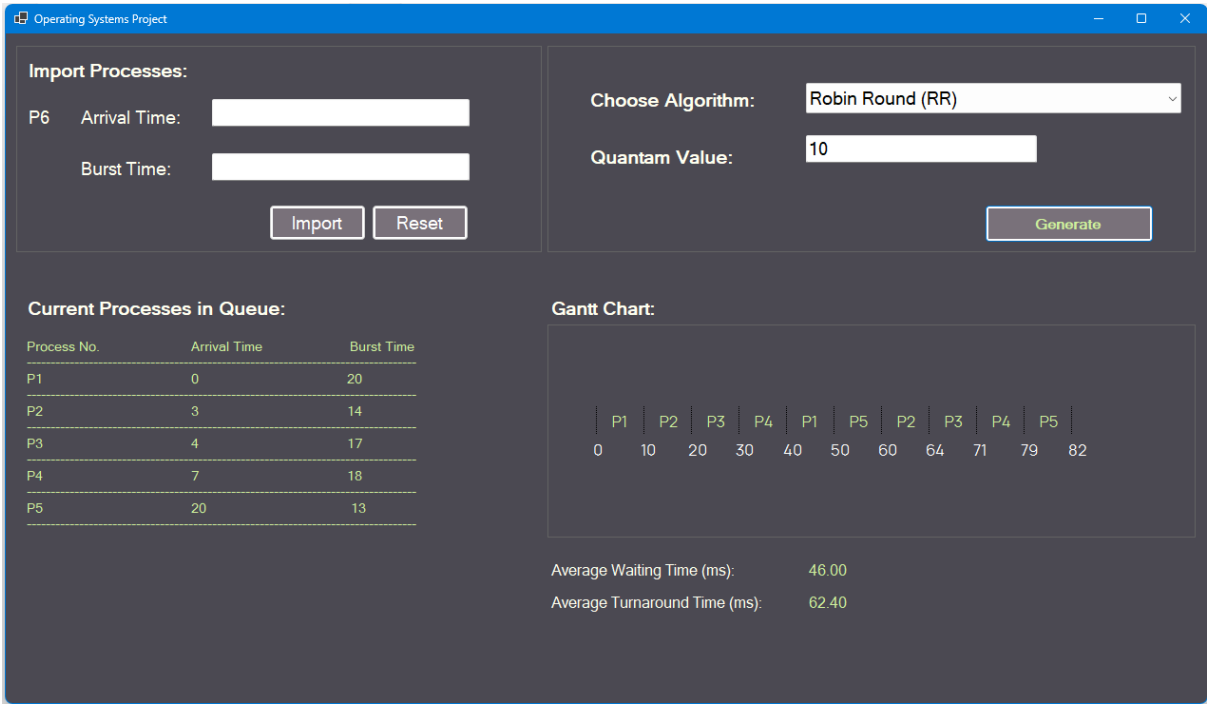
It is not possible to solve Round Robin scheduling with a quantum of 0.

Round Robin scheduling is a preemptive CPU scheduling algorithm in which each process is given a fixed time slice or quantum to execute. Once a process has used up its quantum, it is preempted and the CPU is allocated to another process in the ready queue.

If the quantum is set to 0, it means that each process will be given no time to execute before being preempted. In other words, the scheduler will constantly switch between processes, which will result in a lot of overhead and very little useful work being done.

Therefore, a quantum of 0 is not a valid setting for Round Robin scheduling. The quantum value should be greater than 0 and ideally be chosen based on the specific characteristics of the system being scheduled and the workload it is expected to handle.

5- Round Robin (RR) with 10 Quantum Value:



In Round Robin scheduling algorithm, the CPU executes each process for a fixed time interval called a time quantum. The time quantum determines how long a process can run before it is preempted and added back to the ready queue to wait for its next turn. The scheduler then selects the next process from the ready queue to run and repeats the process until all the processes have completed their execution.

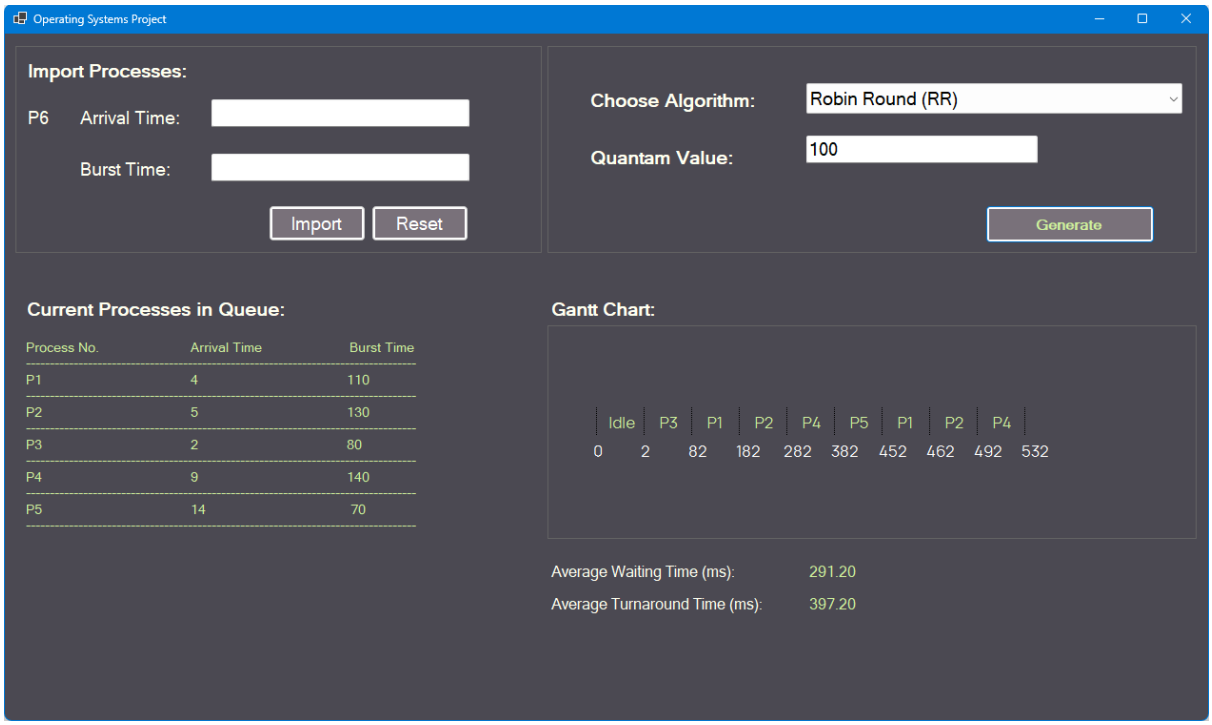
To calculate the average waiting time and turnaround time for the given set of processes using Round Robin scheduling with a quantum of 10, the following steps can be taken:

- Sort the processes based on their arrival time in ascending order: P1, P2, P3, P4, P5
- Create a ready queue and add the processes to the queue in the sorted order.
- Initialize the waiting time and turnaround time for each process to 0.
- Execute the processes in the ready queue for a fixed time quantum of 10 until all the processes have completed their execution. Keep track of the time each process spends waiting in the ready queue and the time it takes to complete its execution.

- P1: Completion time = 50, Turnaround time = 50, Waiting time = 30
- P2: Completion time = 64, Turnaround time = 61, Waiting time = 47
- P3: Completion time = 71, Turnaround time = 67, Waiting time = 50
- P4: Completion time = 79, Turnaround time = 72, Waiting time = 54
- P5: Completion time = 82, Turnaround time = 62, Waiting time = 49

- Average waiting time = (30+ 47+ 50+ 54+ 49) / 5 = 46
- Average turnaround time = (50+ 61+ 67+ 72+ 62) / 5 = 62.4

6- Round Robin with Quantum 100:



In Round Robin scheduling algorithm, the CPU executes each process for a fixed time interval called a time quantum. The time quantum determines how long a process can run before it is preempted and added back to the ready queue to wait for its next turn. The scheduler then selects the next process from the ready queue to run and repeats the process until all the processes have completed their execution.

To calculate the average waiting time and turnaround time for the given set of processes using Round Robin scheduling with a quantum of 100, the following steps can be taken:

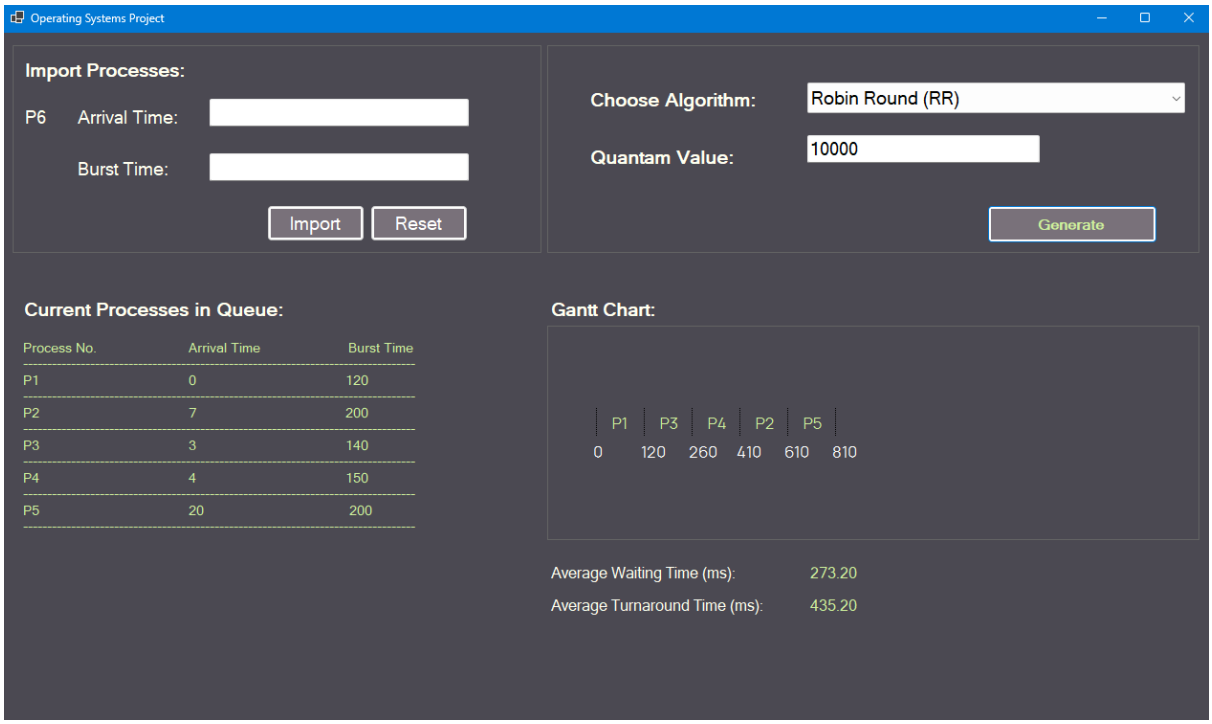
- Sort the processes based on their arrival time in ascending order: P3, P1, P2, P4, P5
- Create a ready queue and add the processes to the queue in the sorted order.
- Initialize the waiting time and turnaround time for each process to 0.
- Execute the processes in the ready queue for a fixed time quantum of 100 until all the processes have completed their execution. Keep track of the time each process spends waiting in the ready queue and the time it takes to complete its execution.

- P1: Completion time = 462, Turnaround time = 458, Waiting time = 348
- P2: Completion time = 492, Turnaround time = 487, Waiting time = 357
- P3: Completion time = 82, Turnaround time = 80, Waiting time = 0
- P4: Completion time = 532, Turnaround time = 523, Waiting time = 383
- P5: Completion time = 452, Turnaround time = 438, Waiting time = 368

The average waiting time and average turnaround time for the given set of processes using Round Robin scheduling with a quantum of 10 can be calculated as follows:

- Average waiting time = (348+ 357+ 0+ 383+ 368) / 5 = 291.2
- Average turnaround time = (458+ 487+ 80+ 523+ 438) / 5 = 397.2

7- Round Robin with Quantum 10000:



In Round Robin scheduling algorithm, the CPU executes each process for a fixed time interval called a time quantum. The time quantum determines how long a process can run before it is preempted and added back to the ready queue to wait for its next turn. The scheduler then selects the next process from the ready queue to run and repeats the process until all the processes have completed their execution.

To calculate the average waiting time and turnaround time for the given set of processes using Round Robin scheduling with a quantum of 10000, the following steps can be taken:

- Sort the processes based on their arrival time in ascending order: P1, P3, P4, P2, P5
- Create a ready queue and add the processes to the queue in the sorted order.
- Initialize the waiting time and turnaround time for each process to 0.
- Execute the processes in the ready queue for a fixed time quantum of 10000 until all the processes have completed their execution. Keep track of the time each process spends waiting in the ready queue and the time it takes to complete its execution.

- P1: Completion time = 120, Turnaround time = 120, Waiting time = 0
- P2: Completion time = 610, Turnaround time = 603, Waiting time = 403
- P3: Completion time = 260, Turnaround time = 257, Waiting time = 117
- P4: Completion time = 410, Turnaround time = 406, Waiting time = 256
- P5: Completion time = 810, Turnaround time = 790, Waiting time = 590

The average waiting time and average turnaround time for the given set of processes using Round Robin scheduling with a quantum of 10 can be calculated as follows:

- Average waiting time = (0+ 403+ 117+ 256+ 590) / 5 = 273.2
- Average turnaround time = (120+ 603+ 257+ 406+ 790) / 5 = 435.2

FCFS (First-Come, First-Served):

- Advantages:
  - Simplicity: FCFS is easy to understand and implement. It doesn't require complex scheduling algorithms or calculations.
  - Fairness: It provides a fair approach where processes are served in the order they arrive, ensuring no process is given preferential treatment.
- Disadvantages:
  - Poor average waiting time: FCFS can lead to a high average waiting time, especially if long processes arrive before short ones. This can result in decreased system efficiency.
  - No consideration for process priority: FCFS does not consider the priority of processes, which could be crucial in certain scenarios.

nPSJF (Non-Preemptive Shortest Job First):

- Advantages:
  - Minimized waiting time: nPSJF aims to minimize the waiting time by executing the shortest jobs first, resulting in better system performance.
  - Efficient resource utilization: By executing short processes first, nPSJF optimizes CPU utilization, allowing more processes to complete in a given time.
- Disadvantages:
  - Possibility of starvation: nPSJF can result in starvation for longer processes if a continuous stream of shorter processes keeps arriving.
  - Prediction difficulty: It requires accurate estimations of the execution time for each process, which can be challenging in practice.

pSJF (Preemptive Shortest Job First):

- Advantages:
  - Reduced waiting time: pSJF aims to minimize waiting time by preempting longer processes when a shorter process arrives.
  - Efficient resource utilization: By executing shorter processes first and allowing preemption, pSJF optimizes CPU utilization and ensures faster turnaround times.
- Disadvantages:
  - Higher complexity: pSJF requires more complex bookkeeping and overhead due to preemption, which can lead to increased scheduling overhead.
  - Increased context switching: Frequent preemptions and context switches can introduce additional overhead and may impact overall system performance.

Round Robin:

- Advantages:
  - Fairness: Round Robin provides fairness by allocating a fixed time slice to each process, ensuring equal opportunity for execution.
  - Responsive: It provides better response time for interactive systems since each process gets a turn in a timely manner.
- Disadvantages:
  - Higher waiting time for long processes: Round Robin can lead to increased waiting time for longer processes, as they have to wait for their next time slice to execute.
  - Inefficient for short processes: The fixed time slice may result in unnecessary context switches and scheduling overhead for short processes.
  -

It's worth noting that the advantages and disadvantages of scheduling algorithms can vary depending on the specific system requirements and workload characteristics.