# Atal Bihari Vajpayee Indian Institute Of Information Technology and Management, Gwalior - 474015

विश्वजीवनामृतं ज्ञानम्

**OPERATING SYSTEMS**

**PROJECT REPORT**

# Topic: <u>Interactive CPU Scheduler</u>

**Presented By:**

**Group 1**

1. Aadit Agarwal (2018 IMT-001)
2. Aashish Khatri (2018 IMT-003)
3. Abhishek Jindal (2018 IMT-006)
4. Aitik Gupta (2018 IMT-010)
5. Himanshu Ruhela (2018 IMT-039)
6. Madhavik Agarwal (2018 IMT-048)

# INTRODUCTION

In operating systems, scheduling is an essential function since almost all computer resources are scheduled before use. It involves the allocation of CPU resources and time to processes or tasks in such a way that certain performance requirements are met.

In multiprogramming, a computer has multiple processes, and they are competing for the allocation of CPU at the same time. So, a choice has to be made which process to run next, this is the scheduling algorithm's job. Whenever the CPU becomes idle, it is also the job of the CPU scheduler to make sure that the CPU utilisation is maximum with the CPU being idle for the minimum time possible. A scheduler executes the process with the help of scheduling algorithms. Thus, scheduling algorithms are the heart of the OS design. A CPU scheduler is the part of an operating system and responsible for arbitrating access to the CPU.

Schedulers are often implemented to keep all computer resources busy, ensure load balancing, allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU).

The process scheduler is a part of the operating system that decides which process runs at a certain point in time. It usually can pause a running process, move it to the back of the running queue and start a new process

The key to multiprogramming is scheduling. Scheduling of the CPU is one of the critical factors that affect the efficiency of the CPU utilisation and make multiprogramming more cost-effective in terms of CPU time allocation.

Scheduling of CPU resources has many algorithms by which it can be scheduled like First Come First Serve (FCFS), Round Robin, Shortest Job First (SJF) and Priority Scheduling. Scheduling a CPU which has different types of processes, which are required to run, can be scheduled using multi-level queue and multi-level feedback queue CPU scheduling techniques. Multi-level queue scheduling is of much importance and is widely used for scheduling of jobs in a processor.

Many of the CPU scheduling algorithms have two types of implementation, namely:

1. **Preemptive Scheduling:** In Preemptive Scheduling, the tasks are mostly assigned to their priorities. It might be important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task is held and resumes when the higher priority task finishes its execution.
2. **Non-Preemptive Scheduling:** In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It also doesn't need specialised hardware (for example, a timer) like preemptive scheduling.

Few terms important in the field of CPU scheduling algorithms are:

1. **Burst Time:** It is a time required by the process to complete execution.
2. **Arrival Time:** The time when the process arrives in the ready queue.
3. **Response Time:** The time required by the CPU to respond to the process for the first time.
4. **Waiting Time:** The total time the process spends in the ready queue from its arrival in the queue to complete its execution.
5. **Turn-Around Time:** The total time taken by the process for completion from the time it arrives in the ready queue.
6. **Throughput:** The total number of processes executed in a unit time.

The main criteria of the CPU scheduling algorithms are categorised as:

1. **Maximise**
    a. **CPU Utilisation:** CPU utilisation is the main task in which the operating system needs to make sure that the CPU remains as busy as possible.
    b. **Throughput:** Following its definition, maximising throughput increases the number of processes executed in a unit time, thus improving the complete execution time of the ready queue.
2. **Minimise**
    a. **Waiting Time:** A decrease in the waiting time indicates the CPU is utilised to its maximum potential, and all the processes get executed with minimum time spent idle in the ready queue.
    b. **Response Time:** A decrease in the response time indicates the scheduling algorithm provides fairness to all the processes and CPU allocation is given uniformly to all the processes in the ready queue.
    c. **Turnaround Time:** A decrease in the complete waiting time period as well as in the response time of the process with increase in the throughput directly implies the processes get executed faster and thus indicates a decrease in the turnaround time of the process.

Thus the purpose of the CPU scheduling algorithms is:

1. The CPU uses scheduling to improve its efficiency.
2. It helps you to allocate CPU resources and time among competing processes.
3. The maximum utilisation of CPU can be obtained with multi-programming as well as multitasking.
4. Scheduling provides an ordered approach for the efficient execution of the process in the ready queue.

# SCHEDULING ALGORITHMS

When more than one process is ready, the operating system must then use a CPU scheduling algorithm to decide which one is to run first and for how long. Modern operating systems are moving towards multitasking environments which mainly depends on the CPU scheduling algorithm since the CPU is the most effective and essential part of the computer. There are various scheduling algorithms. We have studied various CPU scheduling algorithms used around the world. We have accumulated a few of the most common and robust algorithms which provide a complete overview of the different types of algorithms CPU scheduling requires and all the different approaches there possibly can be. A basic explanation of scheduling algorithms accumulated and implemented are:

- **First Come First Served (FCFS):** It is the most straightforward scheduling algorithm. It queues processes in the order that they arrive in the ready queue. The process which arrives first in the ready queue is served first. It is a non-preemptive scheduling algorithm, which means the processor cannot release the process before its execution is over. The FCFS scheduling is fair in the formal sense of fairness, but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait. It cannot cater to the execution of processes according to priority or importance.

- **Shortest Job First (SJF):** The algorithm gives priority to the shortest process available in the ready queue. It can be preemptive or non-preemptive. The main problem with SJF is the necessity of previous knowledge about the time required for a process to complete. The process which has high Burst Time (BT) has to suffer most in this algorithm. This type of suffering is called 'ageing'.

- **Round Robin (RR)**: This scheduling algorithm is based on the concept of time-sharing. It is the combination of first come first served (FCFS) scheduling algorithm and preemption among processes. To implement the RR scheduling algorithm, we keep the ready queue as a FIFO queue of processes. As a new process arrives, it will be inserted into the tail of the ready queue. A process is only given access to the CPU for a specified amount of time called 'TIme Quantum' if it fails to execute completely in the time frame, it has to come again as a new process with the updated burst time.

- **Priority Scheduling:** Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler chooses the tasks to work as per the priority, which is different from other types of scheduling, for example, a simple round-robin scheduling algorithm. Priority scheduling algorithms can also be implemented with preemption or non-preemption. Priority scheduling involves priority assignment to every process, and processes with higher priorities are carried out first. In contrast, tasks with equal priorities are carried out on a Shortest Job First (SJF) or round-robin basis.

- **Multi-Level Queue:** In the multi-level queue scheduling algorithm, the ready queue is divided into separate queues. Based on priority of the process; like memory size, process priority, process burst time or process type these processes are permanently assigned to different queues. Each queue has its own scheduling algorithm. For example, some queues are used for the foreground process and some for the background process. The foreground queue can be scheduled by using a round-robin algorithm while the background queue is scheduled by a first come first serve algorithm. There is no switching of queues for process, i.e, the priority assigned are permanent.

- **Multi-Level Feedback Queue:** This algorithm follows the same principles as the Multi-level Queue Algorithm but with a few changes. It allows a process to move between the queues, i.e. the priorities of the process change during runtime based on their burst times. In this algorithm, the processes are separated with different CPU burst time. If a process uses too much CPU time then it will be moved to the lower priority queues. This idea leaves I/O bound and interactive processes in the higher priority queue. Similarly, the process which waits too long in a lower priority queue will be moved to a higher priority queue. This form of scheduling prevents starvation and aging as well. The queue switching or priority change of the process is dependent upon the burst time and the user has the control over it. This priority control is termed as 'Threshold', i.e., if a process executes a threshold amount of times in a particular queue and still needs more CPU time, then its priority is decreased, and it continues execution in the lower queue. This continues until the process is completely executed.

- **Default Algorithm:** The scheduling algorithm used in most common operating systems is a specific version of the Multi-Level Feedback Queue algorithm. It has 2 layers of Round Robin scheduling with a time quantum of 2 and 4 units respectively and the last layer of FCFS algorithm along with a threshold of 4. That is what we have as our default algorithm. We have also provided functionality that compares every implemented combination of process queue and scheduling algorithm with the default algorithm. This is not an optimal solution. It is only a reference point as to how various process queues behave concerning different approaches of CPU scheduling. The default algorithm may not always be optimal, but in a large set of process queues, the default algorithm stands out.

# APPLICATION FOR CPU SCHEDULER

## Objective

To create a Tkinter based GUI application for user-friendly CPU scheduling algorithms. The purpose being fulfilled here is making the user more aware of the effectiveness of various CPU scheduling algorithms and also about how various algorithms work in types of process queues.

## Implementation

The user can choose to randomly generate a process queue or create a user-defined queue with process id, burst time and arrival time as attributes. The user is allowed to choose between several CPU scheduling algorithms such as First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin, Multi-Level Feedback and more. If Priority Scheduling or Round Robin is selected, the user is prompted to enter the priorities and the desired time quantum, respectively.

Also, in the case of Multi-Level Feedback as well as Multi-Level Queue algorithms, the choice of the Time Quantum for the initial levels is user-defined. After the user has inputted the desired algorithm, he/she will submit the algorithm for processing, and the output will be displayed. The displayed out will contain the process queue, the result of the output parameters for the selected algorithm and also the results of the output parameters for the default algorithm.

This will not only provide an evaluation of the selected algorithms based on the output parameters for the process queue, but also a comparison of how the default algorithm would perform on the similar queue and the comparison between both algorithms based on the output parameters will give the user a more in-depth understanding of the CPU scheduling algorithms.

## Output :

**Process Queue:** The process queue processed is also displayed with Process ID, Arrival Time, and Burst Time attributes.

**Parameters:** To quantify the scheduling done by the chosen algorithm, the application displays the four primary parameters of CPU scheduling, namely:

- Waiting Time
- Response Time
- Turn-Around Time
- Throughput

These parameters will allow the user to compare and evaluate the algorithms.

## Novel Features:

1. The user can randomly generate a process queue for testing and comparison.
2. Choose from standard non-preemptive and preemptive scheduling algorithms to process your queue.
3. You can choose the time quantum for the Round Robin algorithm within a range using an interactive slider.
4. In Multi-Level and Multi-Level Feedback Queue, you can easily choose the algorithm for each level using a drop-down menu.
5. An additional solution for the given process queue is provided along with the selected algorithm using our Default Algorithm for comparison with the selected algorithm.

# ISSUES FACED

As we went through the development phase, we encountered many issues and resolving these issues made our application even more robust. Some of the issues were:

## Error Handling

- Duplicate process ids.
- Non-integer input
- No input detected.
- Non-existent process id entered.
- Process ID being reused in different levels of Multi-Level Queue algorithm.

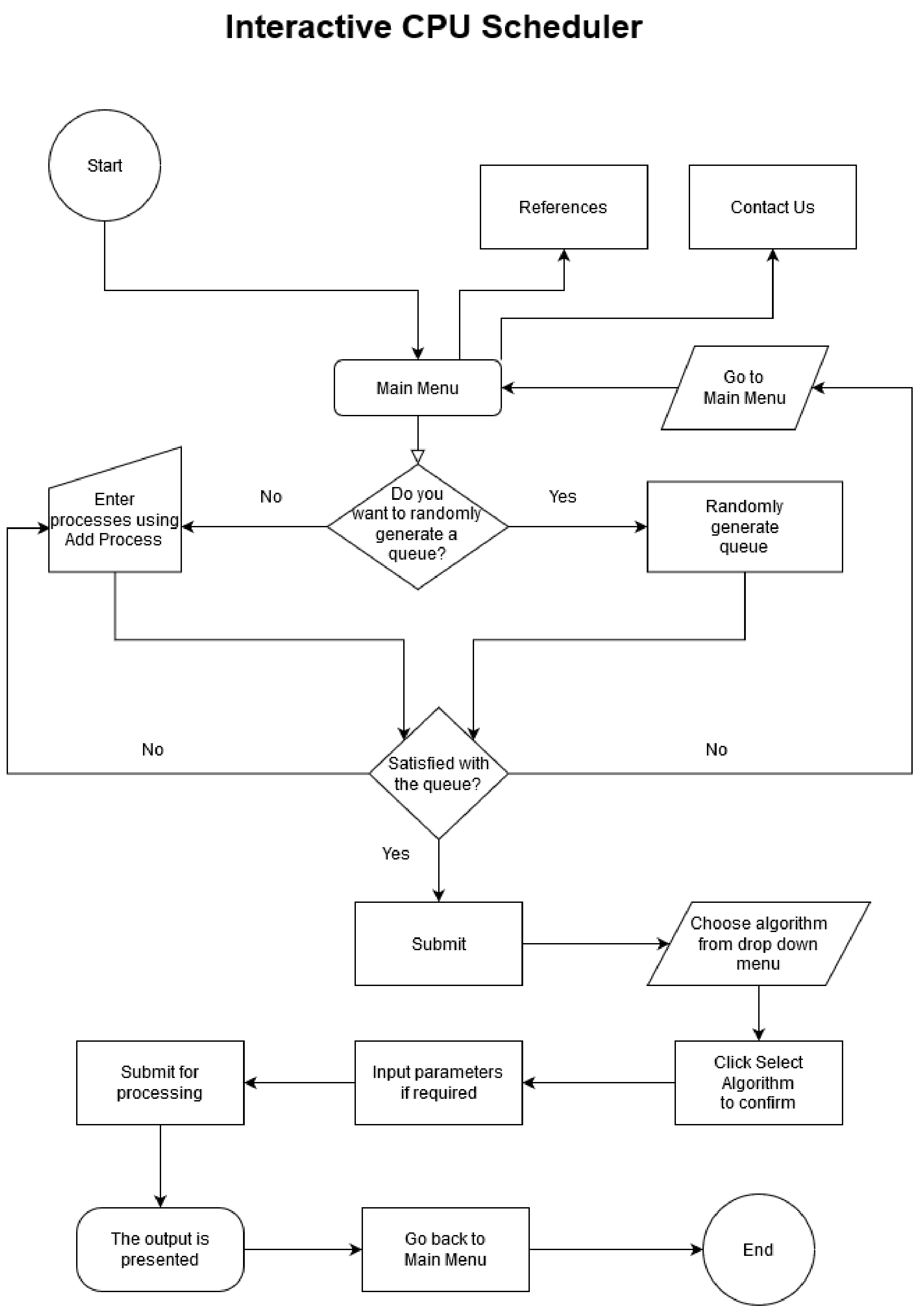
## UX Issues

- GUI sizing and parts of the application being cropped by the window.
- Having a uniform and neat design throughout the application.
- Showing details of input processes during runtime.
- The output being unclear.
- Termination of the application from windows other than the main page.

## Bugs

- Response time and Waiting Time were shown as 0 in Multi-Level Queue.
- If the previous process terminates before the next process arrives, the algorithm goes into an infinite loop.
- If priorities of all processes are the same in the Priority scheduling algorithms.

## DESIGN AND LAYOUT PRESENTATION


Interactive CPU Scheduler
Start
References
Contact Us
Main Menu
Go to Main Menu
Enter processes using Add Process
No
Do you want to randomly generate a queue?
Yes
Randomly generate queue
No
Satisfied with the queue?
No
Yes
Submit
Choose algorithm from drop down menu
Submit for processing
Input parameters if required
Click Select Algorithm to confirm
The output is presented
Go back to Main Menu
End

# FINAL OUTPUT

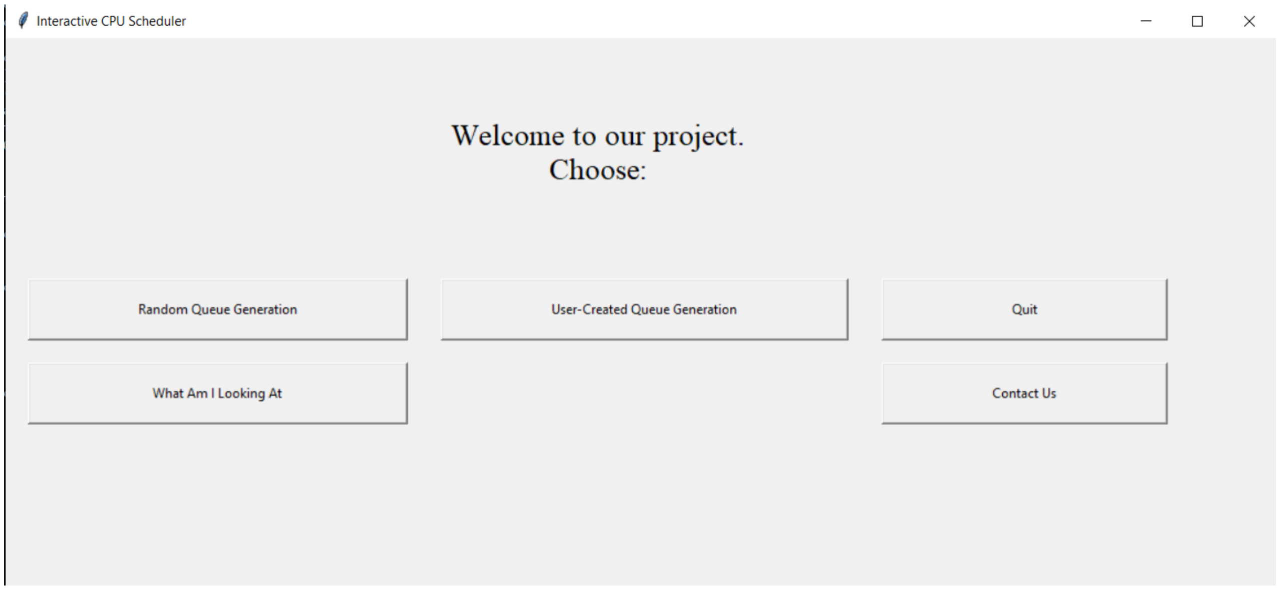
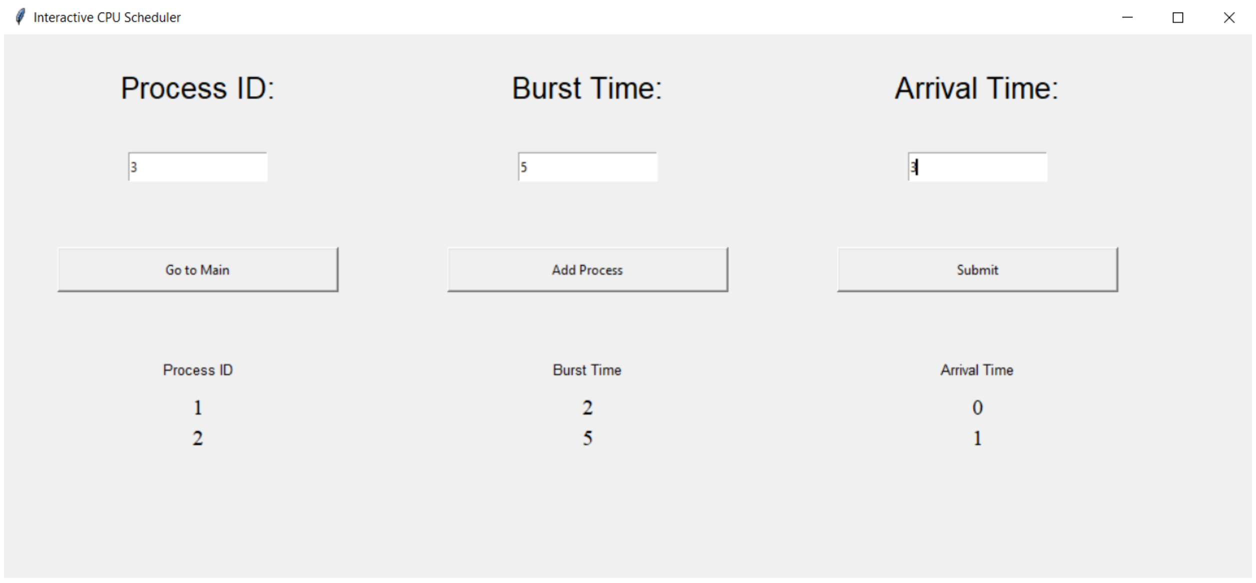


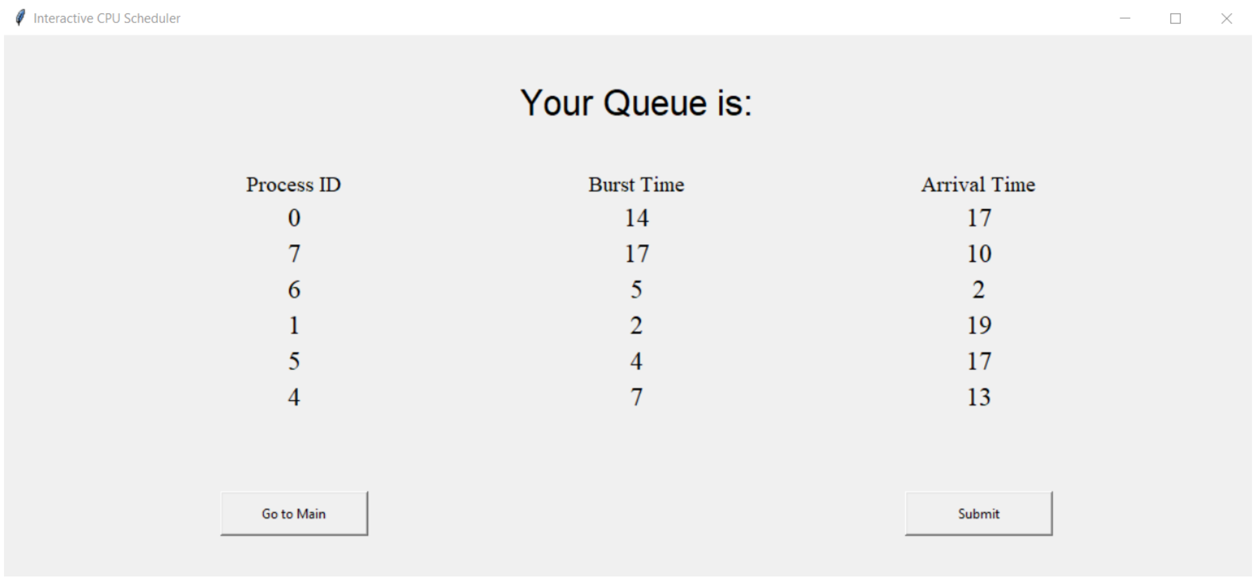
Fig1. Main Page



Process ID: 3
Burst Time: 5
Arrival Time: 3

| Process ID | Burst Time | Arrival Time |
|---|---|---|
| 1 | 2 | 0 |
| 2 | 5 | 1 |

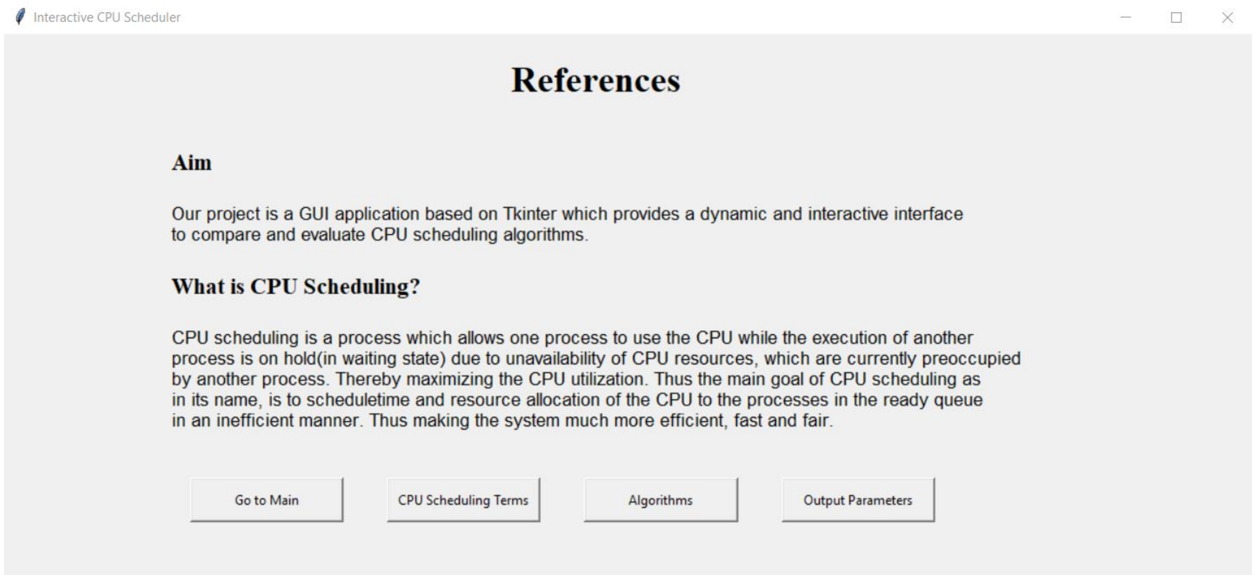
Fig2. Create Own Process Queue



Your Queue is:

| Process ID | Burst Time | Arrival Time |
|---|---|---|
| 0 | 14 | 17 |
| 7 | 17 | 10 |
| 6 | 5 | 2 |
| 1 | 2 | 19 |
| 5 | 4 | 17 |
| 4 | 7 | 13 |

Fig3. Random Queue Generator



References

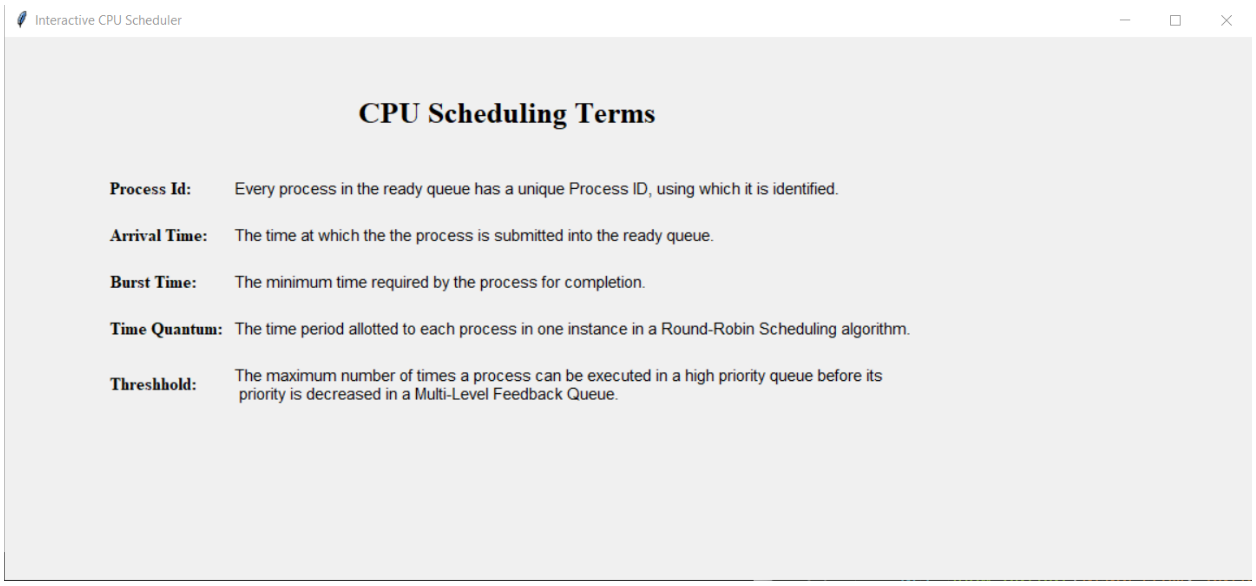
**Aim**

Our project is a GUI application based on Tkinter which provides a dynamic and interactive interface to compare and evaluate CPU scheduling algorithms.

**What is CPU Scheduling?**

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of CPU resources, which are currently preoccupied by another process. Thereby maximizing the CPU utilization. Thus the main goal of CPU scheduling as in its name, is to scheduletime and resource allocation of the CPU to the processes in the ready queue in an inefficient manner. Thus making the system much more efficient, fast and fair.

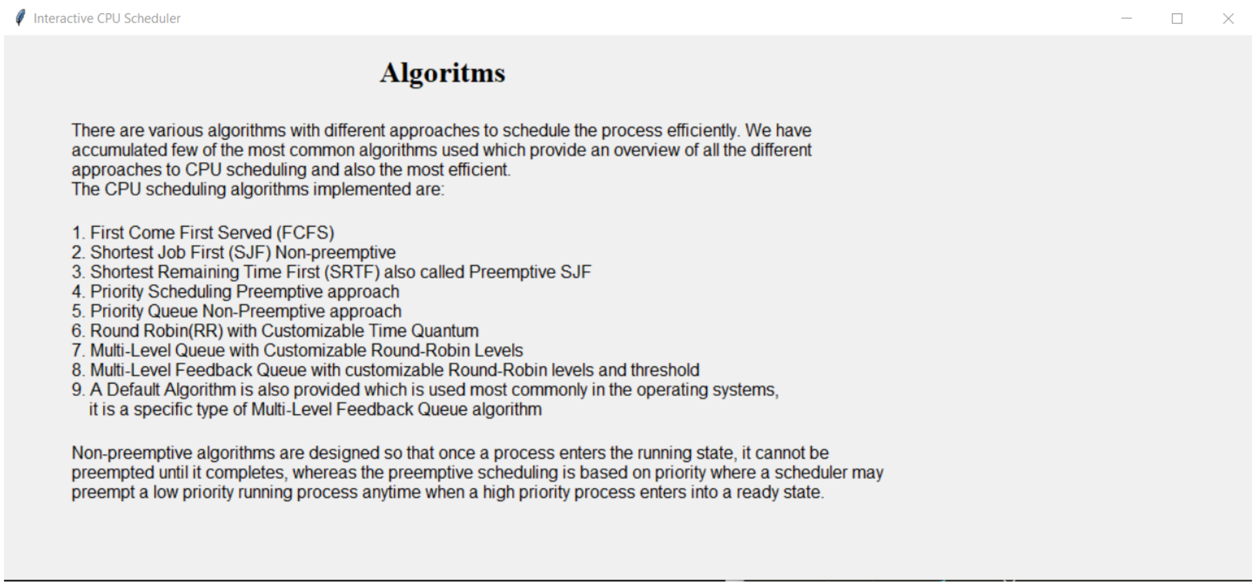
Fig4. References Main Page



CPU Scheduling Terms

| | |
|---|---|
| **Process Id:** | Every process in the ready queue has a unique Process ID, using which it is identified. |
| **Arrival Time:** | The time at which the the process is submitted into the ready queue. |
| **Burst Time:** | The minimum time required by the process for completion. |
| **Time Quantum:** | The time period allotted to each process in one instance in a Round-Robin Scheduling algorithm. |
| **Threshhold:** | The maximum number of times a process can be executed in a high priority queue before its priority is decreased in a Multi-Level Feedback Queue. |

Fig5. References Sub-Page 1



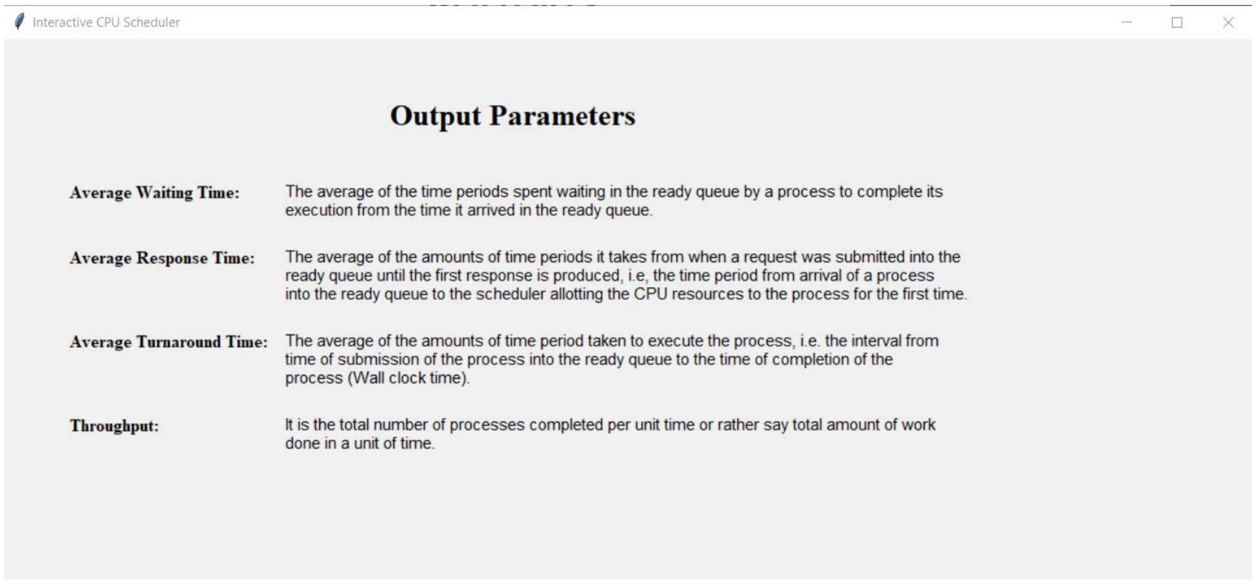
Algoritms

There are various algorithms with different approaches to schedule the process efficiently. We have accumulated few of the most common algorithms used which provide an overview of all the different approaches to CPU scheduling and also the most efficient.
The CPU scheduling algorithms implemented are:

1. First Come First Served (FCFS)
2. Shortest Job First (SJF) Non-preemptive
3. Shortest Remaining Time First (SRTF) also called Preemptive SJF
4. Priority Scheduling Preemptive approach
5. Priority Queue Non-Preemptive approach
6. Round Robin(RR) with Customizable Time Quantum
7. Multi-Level Queue with Customizable Round-Robin Levels
8. Multi-Level Feedback Queue with customizable Round-Robin levels and threshold
9. A Default Algorithm is also provided which is used most commonly in the operating systems, it is a specific type of Multi-Level Feedback Queue algorithm

Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

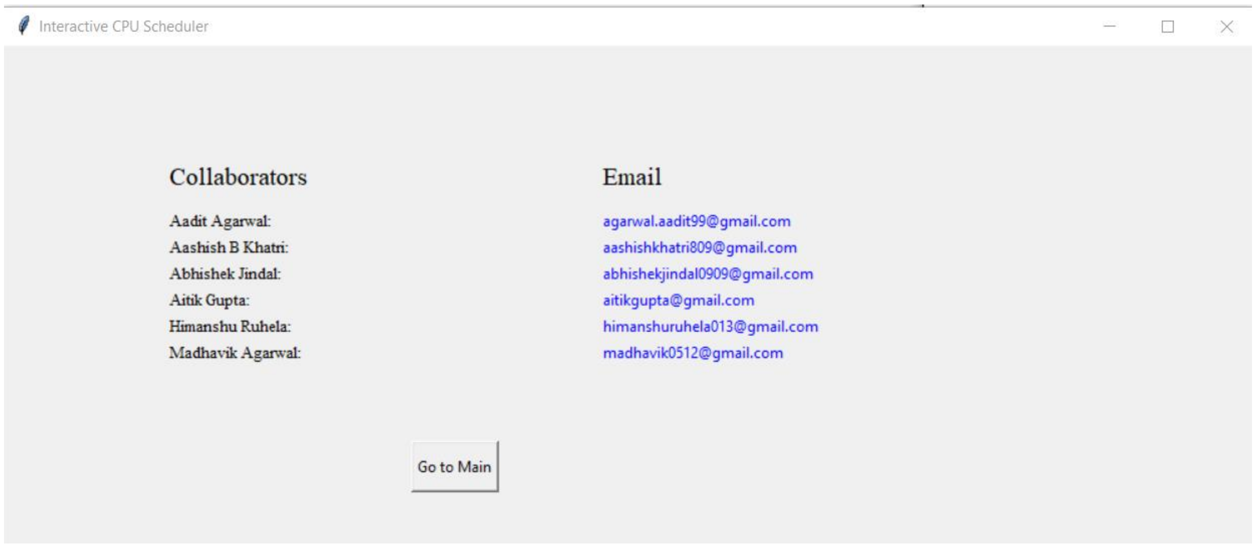
Fig6. References Sub-Page 2


Interactive CPU Scheduler

Output Parameters

| | |
|---|---|
| **Average Waiting Time:** | The average of the time periods spent waiting in the ready queue by a process to complete its execution from the time it arrived in the ready queue. |
| **Average Response Time:** | The average of the amounts of time periods it takes from when a request was submitted into the ready queue until the first response is produced, i.e, the time period from arrival of a process into the ready queue to the scheduler allotting the CPU resources to the process for the first time. |
| **Average Turnaround Time:** | The average of the amounts of time period taken to execute the process, i.e. the interval from time of submission of the process into the ready queue to the time of completion of the process (Wall clock time). |
| **Throughput:** | It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. |

Fig7. References Sub-Page 3


Interactive CPU Scheduler

| Collaborators | Email |
|---|---|
| Aadit Agarwal: | agarwal.aadit99@gmail.com |
| Aashish B Khatri: | aashishkhatri809@gmail.com |
| Abhishek Jindal: | abhishekjindal0909@gmail.com |
| Aitik Gupta: | aitikgupta@gmail.com |
| Himanshu Ruhela: | himanshuruhela013@gmail.com |
| Madhavik Agarwal: | madhavik0512@gmail.com |

Fig8. About Us Page

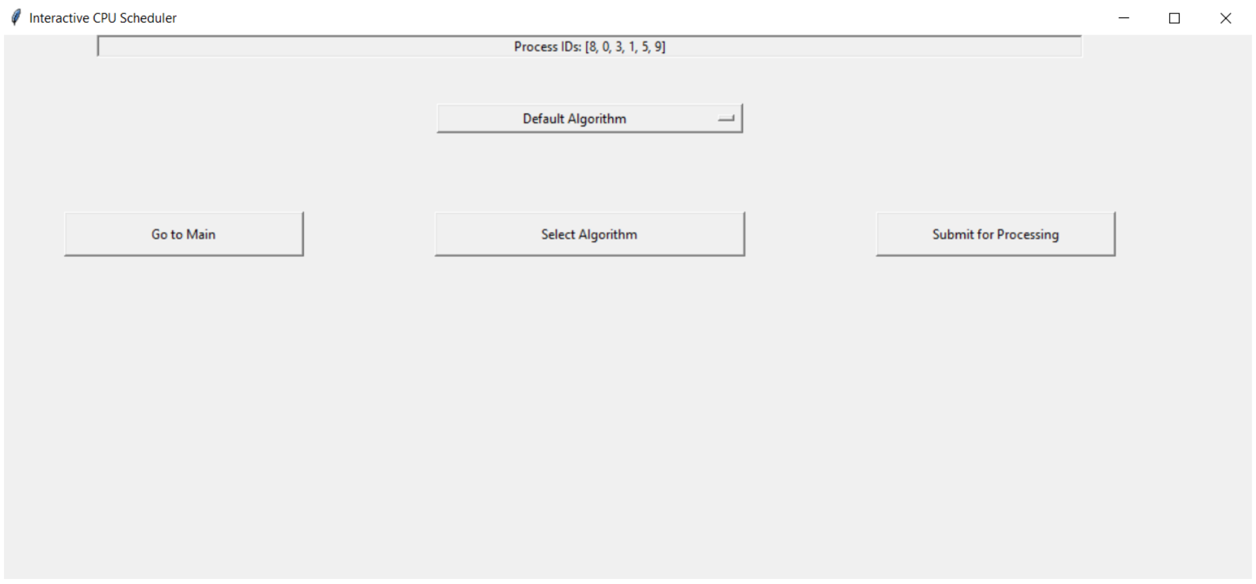


Fig9. Main Algorithm Selection Page

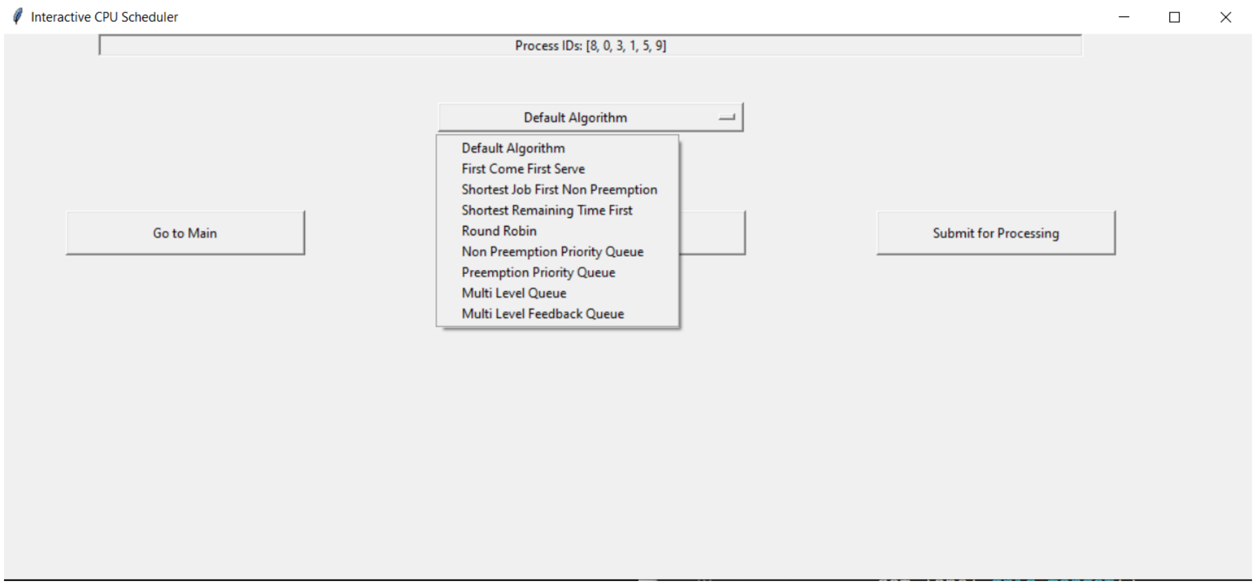
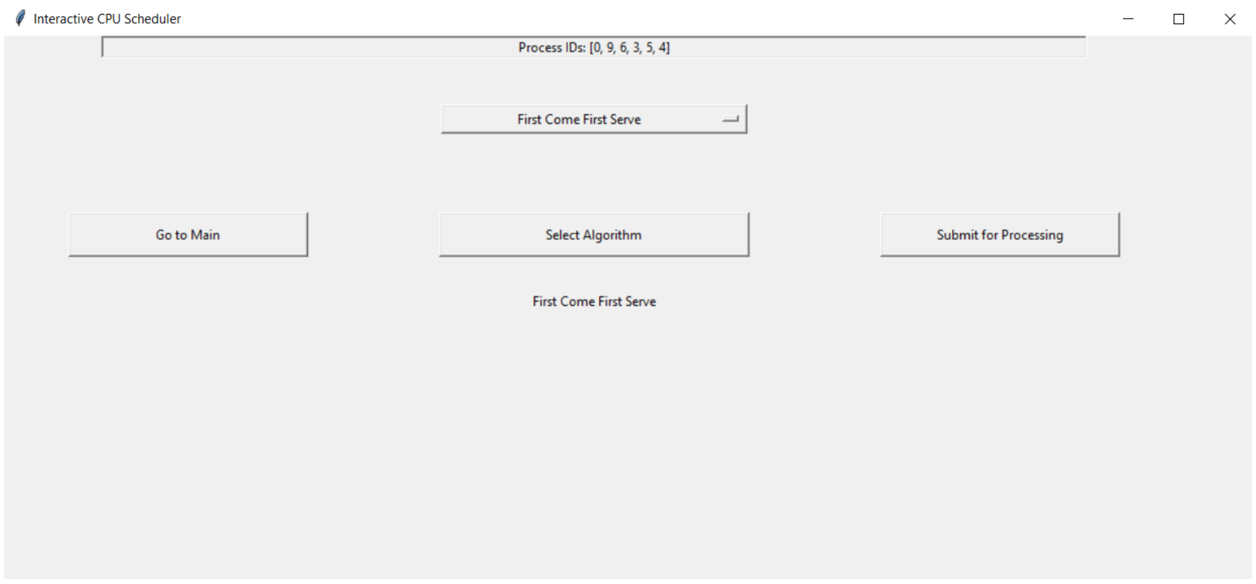

Interactive CPU Scheduler
Process IDs: [8, 0, 3, 1, 5, 9]
Default Algorithm
Default Algorithm
First Come First Serve
Shortest Job First Non Preemption
Shortest Remaining Time First
Round Robin
Non Preemption Priority Queue
Preemption Priority Queue
Multi Level Queue
Multi Level Feedback Queue
Go to Main
Submit for Processing

Fig10. Drop-Down of Algorithms

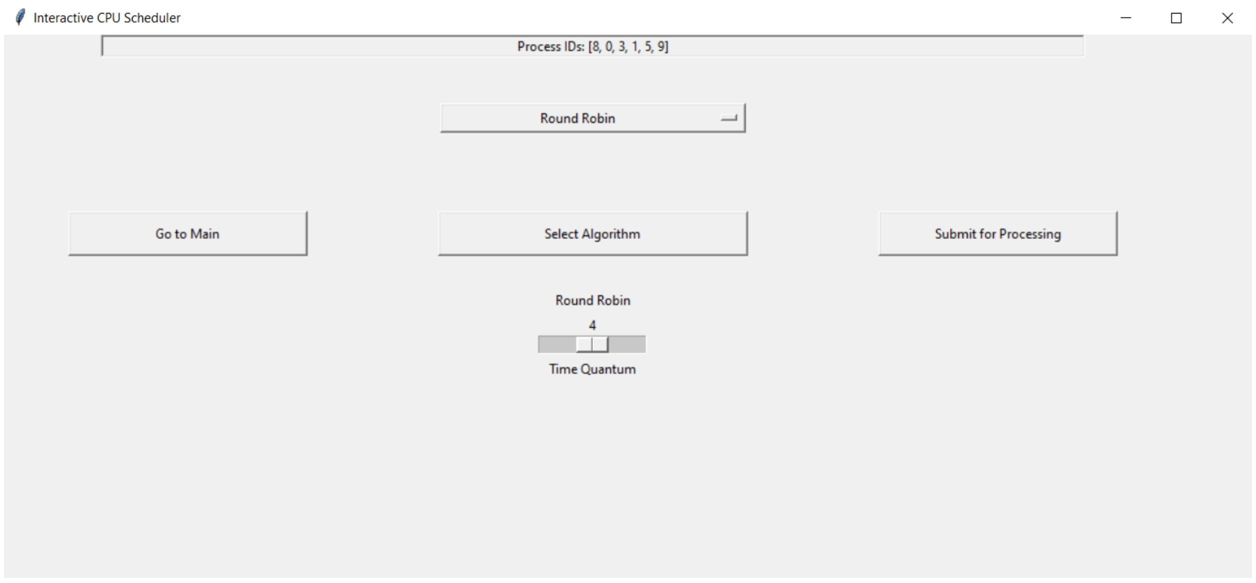


Process IDs: [0, 9, 6, 3, 5, 4]

First Come First Serve

First Come First Serve

Fig11. Example of FCFS algorithm selection



Process IDs: [8, 0, 3, 1, 5, 9]

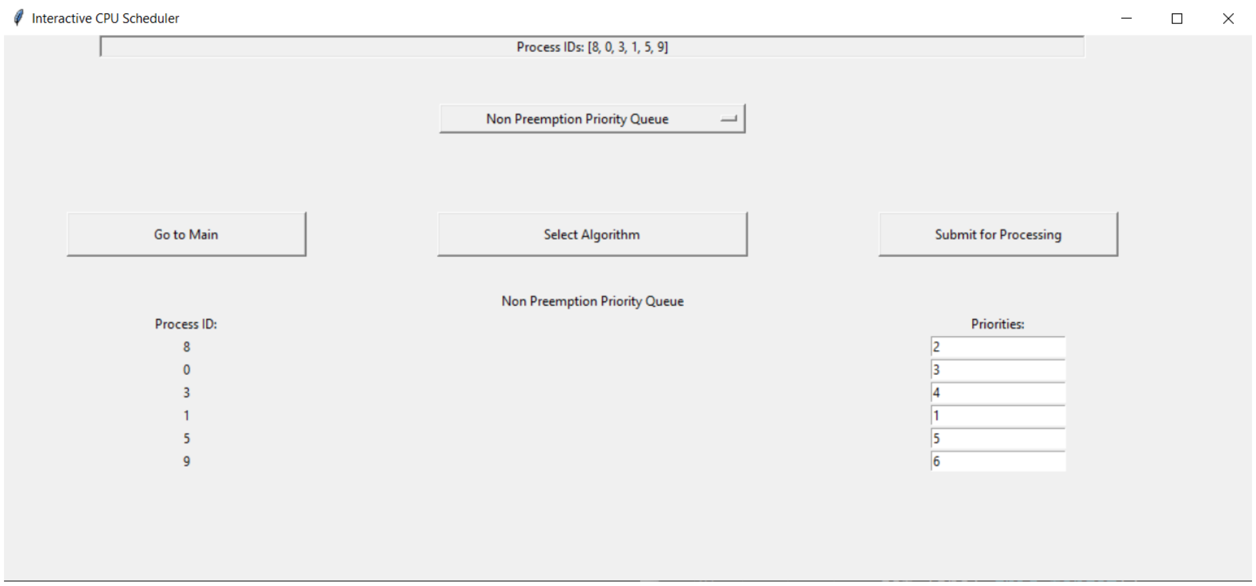
Round Robin

Round Robin

4

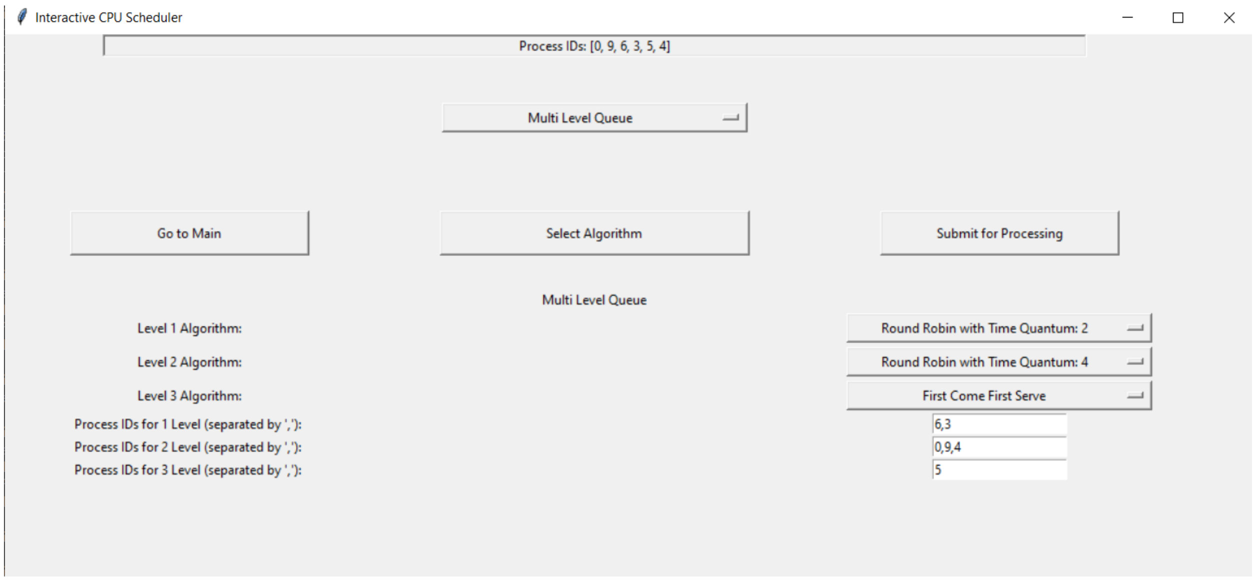
Time Quantum

Fig12. Example of Round Robin Algo Selection


Interactive CPU Scheduler
Process IDs: [8, 0, 3, 1, 5, 9]
Non Preemption Priority Queue
Go to Main
Select Algorithm
Submit for Processing
Non Preemption Priority Queue

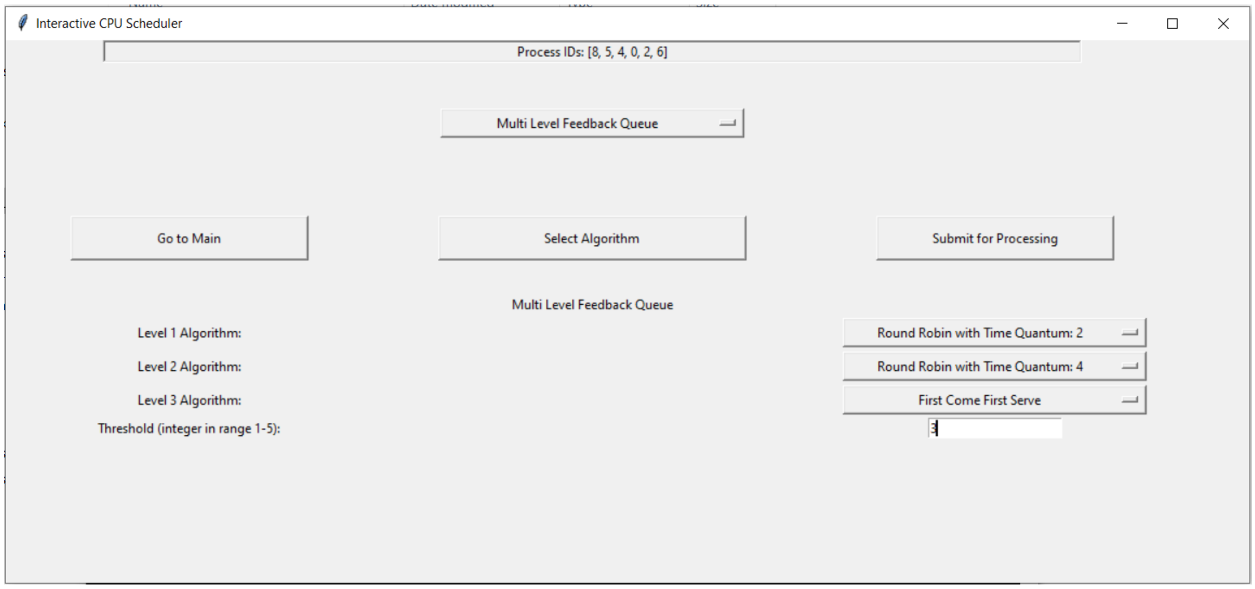
| Process ID: | Priorities: |
|---|---|
| 8 | 2 |
| 0 | 3 |
| 3 | 4 |
| 1 | 1 |
| 5 | 5 |
| 9 | 6 |

Fig13. Example of Priority Algorithm



| | |
|---|---|
| Level 1 Algorithm: | Round Robin with Time Quantum: 2 |
| Level 2 Algorithm: | Round Robin with Time Quantum: 4 |
| Level 3 Algorithm: | First Come First Serve |
| Process IDs for 1 Level (separated by ','): | 6,3 |
| Process IDs for 2 Level (separated by ','): | 0,9,4 |
| Process IDs for 3 Level (separated by ','): | 5 |

Fig14. Example of a Multi Level Queue Algorithm


Interactive CPU Scheduler

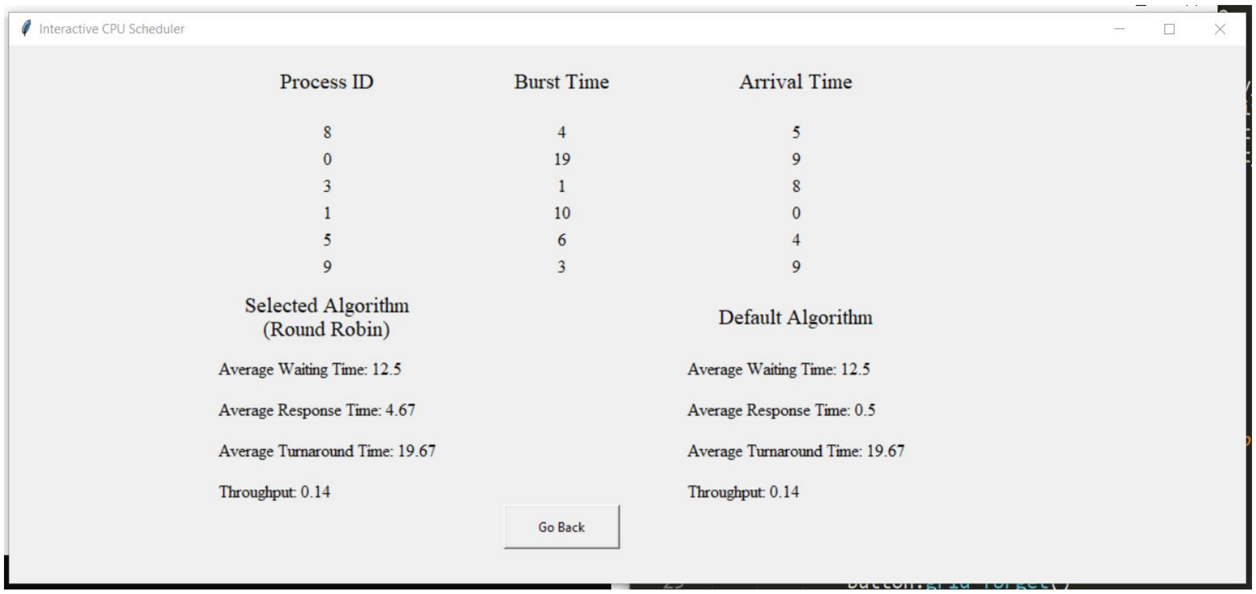
Process IDs: [8, 5, 4, 0, 2, 6]

Multi Level Feedback Queue

Multi Level Feedback Queue

| | |
|---|---|
| Level 1 Algorithm: | Round Robin with Time Quantum: 2 |
| Level 2 Algorithm: | Round Robin with Time Quantum: 4 |
| Level 3 Algorithm: | First Come First Serve |
| Threshold (integer in range 1-5): | 3 |

Fig15. Example of Multi-Level Feedback Queue algorithm


Interactive CPU Scheduler

| Process ID | Burst Time | Arrival Time |
|---|---|---|
| 8 | 4 | 5 |
| 0 | 19 | 9 |
| 3 | 1 | 8 |
| 1 | 10 | 0 |
| 5 | 6 | 4 |
| 9 | 3 | 9 |

Selected Algorithm
(Round Robin)

Average Waiting Time: 12.5

Average Response Time: 4.67

Average Turnaround Time: 19.67

Throughput: 0.14

Default Algorithm

Average Waiting Time: 12.5

Average Response Time: 0.5

Average Turnaround Time: 19.67

Throughput: 0.14

Fig16. Example of Output

## RESULT

The user has a basic understanding of the CPU schedulers in various situations of service request queues based on the type of requests (a few long processes, many small processes, a mix of processes, etc.). The best feature of the project is its entirely GUI based interface, allowing even a user with minimal knowledge of CPU processes and scheduling to gain an understanding of the scheduling algorithms implemented in the CPU to maximise the CPU resources utilisation. This understanding will be depicted by giving statistics of the four parameters of CPU scheduling algorithms to provide even in-depth knowledge of the same.

## CONCLUSION

The application is a Tkinter based GUI application to evaluate various CPU-scheduling algorithms with a complete user-defined environment to project the results of the effectiveness of their algorithm as well compare it to the algorithm that the CPU might use, i.e. the default algorithm and compare the results.

Achieving our goal to make the user more aware of various CPU scheduling algorithms and the science behind the fast effectiveness of CPU to execute such lengthy process in such a short period of time along with performing other small processes as well and not creating a lag in execution so that the user has their output in the shortest amount of time along with the functioning of the complete Operating System.

# REFERENCES

1} Github repository of the project:

https://github.com/aitikgupta/interactive_cpu_scheduler

2} Tkinter reference:

https://www.youtube.com/watch?v=YXPyB4XeYLA

3} Basic Definition of CPU scheduling:

https://en.wikipedia.org/wiki/Scheduling_(computing)

4} Understanding the CPU scheduling algorithms:

https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/

5} Inspiration for the CPU scheduling algorithms:

https://www.guru99.com/cpu-scheduling-algorithms.html