Vol. 1 - No. 1
November 2021 E-ISSN:

Analysis and Implementation of Efficient Hybrid CPU Scheduling Algorithm

Meghna Manoj Nair, G. Harika Sai, Bheemisetty Neha and Dr. R. Maheswari
School of Computer Science Engineering
Vellore Institute of Technology, Chennai, Tamil Nadu, India

Abstract
CPU Scheduling and enhancing the utilization of the CPU is an extremely important concept in the field of Operating Systems and in order to acquire maximum utilization of the CPU in the system, it's essential to have multiple programs or processes run parallelly. This ensures that the CPU has reduced idle time and is constantly working and scheduling processes in turn leading to a maximization of the throughput and overall work output of the CPU. Scheduling of CPUs is one of the critical factors that affect the efficiency of the CPU and the utilization of the system is maximized when we allocate processors in a precise manner in process scheduling. While curating this idea, the key focus for us is to develop and build over the NOVEL algorithms so as to significantly reduce the waiting time and turnaround time for all scheduled processes when compared to the various conventional algorithms like FCFS, SJF, SRTF, RR, and priority scheduling algorithms. In order to incorporate new and innovative features, we have also created an extensive application which creates a Gantt chart based on the values taken as input from the user/client.

Keywords
CPU Scheduling Algorithms, Hybrid Algorithm, Efficiency, Execution Time, Scheduler, Optimization

I. INTRODUCTION

Operating Systems support wireless communication. As we already know that wireless communication occurs in the number of applications. And conventionally in the wireless network portable devices such as the laptops and mobile phones are associated with a single access point where each device deals with a rigid pool of data speed. In a Single Processor Operating Software System, only one method/process can run at a particular time. All processes which are ready to be executed need to wait for as long as a processor is dispatched to the process. Once the CPU is freed from its current task, the next process at the head of the ready queue is sent in for execution. Usually, the computer enters the idle state each time there's a gap between finishing the execution of a process and waiting for a process to be dispatched into the running state. No work is accomplished during this time. In order to accomplish this CPU Scheduling is introduced. Multiple processes are kept in the main memory at one time. In the multi programming system, multiple processes are stored in the main memory and are maintained in the main memory. Majority of the processes have a dedicated memory location which contains all intricacies and detailing of the respective process including those of the program to be processed, contents of the registers, etc. Each of the processes also takes it in turn between the processor consumption and I/O event occurrences. The highlight of multiprocessing in a system is to ensure that the CPU never sits idle and that it needs to consistently indulge in executing/processing programs and tasks as this is one of the best ways to increase and elevate the CPU utilization and efficiency. Most importantly, all peripherals and resources to be used by the scheduled processes need to be

scheduled and arranged in an organized manner. So, the main focus and duty of action of CPU scheduling is to monitor and control the sequence and order in which processes need to be executed. This makes CPU scheduling a dominant factor in OS as it creates a huge impact on the usage of resources and performance of the machine.

CPU scheduling related activities usually occur during the following condition:

1. When the respective process shifts from running to waiting state
2. When the respective process changes from running to ready state
3. When the respective process changes from waiting to ready state
4. When the process comes to an end or terminates

In cases one and four, a new process must be dispatched for execution. However, in cases two and three, there are options and choices. In cases one and four, the scheduling is non preemptive and in situations two and three, its preemptive. FCFS is a simple and easy scheduling algorithm with regards to implementation and understanding. The process which arrives first and requests for the CPU will be allotted the processor and goes into execution. This is done through the FIFO queue. However, it can lead to extremely large waiting times and takes quite long for completion. One of the other conventional scheduling algorithms is Shortest Job First (SJF) wherein the CPU allocates the processor to those processes which have the shortest amount of time for completion and considers the burst time for scheduling the processes. In case of situations where more than one process has the same amount of time for completion, then FCFS is used to decide which process is scheduled next. Priority scheduling, is the algorithm that executes and completes the processes based on the priority which has been assigned to each of the tasks/processes in the schedule queue. Those processes with a higher priority are executed initially while those with lower priority are processed towards the end. This algorithm can also be of preemptive or non-preemptive type. Round Robin is the algorithm which is capable of ensuring that each of the processes get an equal share of CPU allotment on the basis of the pre-decided time quantum is the RR algorithm. Each of the processes from the scheduling queue will be executed in a cyclic order for a limited amount of time. The major highlight of this algorithm is that it ensures starvation free process execution.

In general, there are three types of process schedulers in Operating System:

A. Long Term Scheduler / Job Scheduler

Long Term Scheduler is an essential dispatcher as it is responsible for bringing out the newly created processes into the ready state. Apart from this it also monitors and controls the different I/O peripherals and CPU bounds which cater to the process to be executed. Long term scheduler is usually invoked only in necessary cases and hence it’s called infrequently.

B. Medium Term Scheduler

The Medium-Term Scheduler is accountable for resuming the process and also to suspend the process. On the whole this scheduler swaps i.e., moves the process from the main memory to the disk and also from disk to the main memory. This scheduler also reduces the degree of the multiprogramming. This is also useful for maintaining a perfect balance between IO and CPU bounds. Swapping/rolling of processing could be required to improvise

the sequence and to cater to the memory requirements in order to ensure that the there's no over commitment towards the less memory space made available.

C. Short Term Scheduler

The short-term scheduler is one of the highly used schedulers as it's used to dispatch the process which needs to be scheduled next. It carries out the task of choosing the next process from the ready queue based on the algorithm being implemented and transfers it into the running state. Furthermore, this scheduler is also responsible for ensuring that each process gets a chance to get executed and isn't starved, especially the processes which possess a high burst time. Due to these tasks and chores of responsibility, this scheduler is approached frequently.

II. LITERATURE SURVEY

The most important asset of the PC framework is CPU that is the reason in multiprogramming working framework; the fundamental memory has a few procedures and there must be appropriate criteria to serve these procedures, known as CPU Scheduling. There are distinctive CPU booking calculations. Everyone has their own benefits and faults. There are a plethora of prominent and commonly used algorithms like First Come First Serve (FCFS) algorithm, Shortest Job First (SJF) algorithm, Priority Scheduling algorithm, etc. However, the Round Robin scheduling algorithm portrays a major bottleneck of an extremely high waiting time and turnaround time for the processes. Improved Round Robin picks the main procedure from the prepared line and dispenses the CPU to the procedure for a period interim of up to time Quantum.

Once the time slice has been fulfilled for a particular process, the dispatcher looks into the right portion of the queue for getting insights into the next available process which is ready for execution, and this is the Improved Round Robin scheduling algorithms as portrayed by the authors of in 2012. Authors of have proposed a hybrid approach for scheduling where the sum of average standard deviation and time quantum based concept is used for scheduling. In 2013 Increased Round Robin calculation was projected, by neutering the time quantum of a simply those procedures that needs a slightly additional noteworthy time than the appointed time quantum cycle. The rest of the procedures will be executed in the ordinary round robin way. Another variation of Round Robin planning calculation was proposed in 2011, in which the procedures were executed by the new computed match issue that used the concept of dynamic time quantum. In the middle of 2012, a few alternative esteems were added to the time quantum. In 2012,a work was projected during which the procedures were organized in climbing request as indicated by the foremost restricted outstanding burst time and the Time Quanta was figured by duplicating the conventional summation of least and greatest Burst Time.

III. EXISTING SYSTEM

CPU Scheduling is an extremely essential and necessary task for any CPU to ensure that the tasks/processes to be executed are done in an organized, systematic and consolidated manner. Some of the highly common and exposed algorithms include:

First Come First Serve (FCFS): As the name suggests, this scheduling algorithm tries to assign the CPU to whichever task/process/file that reaches the scheduling queue first. It is one of the easiest types of scheduling algorithms and it works hand in hand with that of a FIFO queue. This algorithm replicates real life instances in a number of ways, for example – standing in a queue to buy a ticket, etc. However, this algorithm is non-preemptive in nature (i.e., the process will only release the CPU after it has completed its execution to its entirety). The main drawback in this case is the extremely high average waiting time which is a result of the scheduling logic initiated by FCFS.

Shortest Job First (SJF): In this type of scheduling algorithm, CPU completes (or prioritizes) and executes those processes which have comparatively lower execution time or burst time. This ensures that all those tasks with lower time of completion are scheduled first while those with higher completion time are pushed towards the end. This algorithm can operate in both preemptive and non-preemptive mode based on which the processes will release the CPU either after completing the task or voluntarily in between respectively. One of the major bottlenecks in this algorithm is that the execution time of each of the processes must be known prior and it can be tedious to predict it prior.

Round Robin (RR): The algorithm which is capable of ensuring that each of the processes get an equal share of CPU allotment on the basis of the pre-decided time quantum is the RR algorithm. Each of the processes from the scheduling queue will be executed in a cyclic order for a limited amount of time. The major highlight of this algorithm is that it ensures starvation free process execution. However, in situations where the time quantum is deemed to be too less, the overhead caused by context switching increases rampantly.

Shortest Remaining Time First (SRTF): This algorithm is often considered to be a modified version of the SJF algorithm, or in other words SRTF is the preemptive form of SJF algorithm. This algorithm executes the processes and allocates the CPU to those tasks and processes which are nearer to completion. However, the flaw of this algorithm is the high context switching overheads which are involved which adds up to the processing time, thus reducing the efficiency of the CPU.

Priority Scheduling: This algorithm executes and completes the processes based on the priority which has been assigned to each of the tasks/processes in the schedule queue. Those processes with a higher priority are executed initially while those with lower priority are processed towards the end. This algorithm can also be of preemptive or non-preemptive type. One of the main concerns which arises as an issue to this algorithm is that in case of any system crashes or data loss, all low priority processes will be lost and there are chances of starvation.

Apart from these conventional algorithms, a number of researchers and scholars have put forward modifications and variation to further enhance and optimize the utilization of the CPU, as mentioned in the literature review. The authors of have proposed a technique to

improvise the round robin scheduling algorithm by incorporating the main features of SJF and ordinary RR algorithms and they have experimentally proven it to be much more efficient than that of conventional RR algorithm. On the contrary, another approach to improvise and enhance the CPU usage is to adopt a dynamic multilevel hybrid scheduling algorithm using median/square root approach. This approach ensures that there's minimum waiting, turnaround and response time involved. Another possible innovation in the field of CPU scheduling algorithms is to incorporate a genetic approach for sequencing and scheduling the tasks as required and this has proven to drastically reduce the average waiting time as a result. Apart from CPU scheduling algorithms, it's essential for students, researchers and scholars related to this field to be able to calculate the different parameters which are linked to that of a scheduling algorithm. Gannt char is one of the useful ways to provide a holistic view with regards to the time consumed for termination of each process and other parameters. There are quite a few CPU scheduling algorithm calculators which allow users to input the arrival/burst time of the processes and it would return the completion time and waiting time as required. A few other calculators also make use of the input data to generate static Gantt charts to provide a visual representation of the sequencing and timing parameters of the processes.

IV. PROPOSED SYSTEM

As mentioned in section 4, each of the conventional approaches do possess a drawback or a flaw which pulls down the efficiency of the algorithm to a great extent. As a team, we have worked on a hybrid algorithm which incorporates modifications to the conventional Round Robin algorithm and are proposing the same in this report. As discussed previously, one of the main drawbacks of the RR algorithm is the high rise in context switching overheads, especially in cases where the time quantum is too less in comparison to the burst time of the processes. The algorithm we have curated would decide on the time quantum on the basis of the burst time of the processes. The proposed algorithm lays its foundation on the NOVEL scheduling algorithm which basically carries differing time quantum to be considered by the processes. This ensures that the scheduler has the control and power to schedule processes in such a way that it reduces and offers a decent response time. The main highlight of this algorithm is that it ensures a reduced waiting time which further enhances the versatile programming environment keeping the higher number of processes to get executed at an earlier time.

The hybrid algorithm will determine a time slice which would cater to the needs of all the processes which are scheduled, thus leading to an efficient and optimized utilization of the CPU. On initializing the hybrid algorithm, all the scheduled processes are sorted in an increasing order with regards to the elapsed time. The elapsed time is calculated by computing the sum of the arrival time and the burst time. A loop is initialized until the queue gets empty. Within the loop, all the processes are repeatedly checked at constant intervals of time and the process with the least elapsed time is executed next. This process repeats until the queue becomes empty. Based on the elapsed time of the processes is the time slice/quantum is chosen and the processes are executed in a cyclic fashion. Apart from this,

the proposed hybrid algorithm can be operated in both preemptive and non-preemptive techniques, making it more flexible and dynamic in nature.

In order to add an element of uniqueness and innovation to our project, we have developed an application that helps users to get a holistic idea on conventional CPU scheduling algorithms with a great visual demonstration. On opening the application, the user has the option to choose his/her preferred tab as per their interest from the menu. If the user wishes to gain more insights on the different types of CPU Scheduling Algorithms, the user can click on the scheduling algorithms tab where they can access a crisp and brief explanation of each of the scheduling algorithms.If the user wishes to obtain a Gantt chart along with other details for any number of processes, the user can navigate to the respective tab where he/she can enter values for different time parameters and obtain results. The additional feature here is that the Gantt chart will be displayed in a dynamic fashion with a timer count. Furthermore, if the user has any doubts/clarifications with regards to CPU Scheduling, he/she can chat with our virtual chatbot Lexy to clear the queries.

V. SYSTEM DESIGN AND IMPLEMENTATION

The diagram shown in Fig. 1 illustrates the Hybrid Algorithm proposed as a part of this project. Initially on executing this algorithm some space of the memory is reserved for the ready queue. Following this, the numerous processes which are to be executed are inserted into the read queue. The corresponding arrival time and burst time of each process is also saved and the Total Execution Time (TET) is calculated for each process respectively. After the computation of TET for each individual process, the processes are sorted on the basis of TET. Based on this sorted list, the processes are scheduled one by one and are executed accordingly.

VI. WORKING MODULES

- FCFS - First Come First Serve
- SJF - Shortest Job First
- SRTF - Shortest Remaining Time First
- PRIORITY
- ROUND ROBIN
- HYBRID ALGORITHM
- WEB APPLICATION
    - Home Page
    - Algorithms and Information
    - Visualization and Algorithm Calculator
    - Chatbot

The above-mentioned modules highlight the major algorithms which have been considered for analysis and implementation. The flow of execution and needs are described in the next section.

VII. DESCRIPTION OF EACH MODULE

This section elucidates the different working modules and components involved in the development of the project so as to carry out necessary comparisons and analyses.

A. First Come First Serve (FCFS)

This scheduling algorithm tries to assign the CPU to whichever task/process/file that reaches the scheduling queue first. It is one of the easiest types of scheduling algorithms and it works hand in hand with that of a FIFO queue. Fig. 2 describes the work flow of the FCFS scheduling algorithm. Initially, the processes are input along with their burst time (bt[i]) following which the waiting time (wt[i]) for each process is computed. For every first process which arrives in the queue, the waiting time is set to zero as the scheduler needn't wait for any other process. Consequently, the waiting time for every other process is calculated using the formula in equ(1):

$$wt[i] = bt[i-1] + wt[i-1] \quad (1)$$

Then the turn round time is calculated by the formula in equ (2):

$$tat[i] = wt[i] + bt[i] \quad (2)$$

Further, the average waiting time and average turnaround time for all the scheduled processes are calculated for comparison.

B. Shortest Job First (SJF)

In this type of scheduling algorithm, CPU completes (or prioritizes) and executes those processes which have comparatively lower execution time or burst time. This ensures that all those tasks with lower time of completion are scheduled first while those with higher completion time are pushed towards the end. Fig. 3 elucidates the flow of algorithm for SJF scheduling. Initially all processes are analyzed along with their arrival times and are sorted on the respective grounds. The process with the least arrival time and minimum burst time is chosen for scheduling. After completing the scheduled set of processes, a process pool is curated in order to repeat the same until a particular process has been completed.

C. Shortest Remaining Time First (SRTF)

This algorithm is often considered to be a modified version of the SJF algorithm, or in other words SRTF is the preemptive form of SJF algorithm. This algorithm executes the processes and allocates the CPU to those tasks and processes which are nearer to completion. Fig. 4 focuses on the flow of execution for SRTF algorithm. The processes arriving are scheduled in the queue and each time a process with less burst time arrives, the currently executing process is preempted and the newly scheduled one will be executed. Similarly, all processes are executed until the queue is empty.

D. Round Robin

Each of the processes from the scheduling queue will be executed in a cyclic order for a limited amount of time. The major highlight of this algorithm is that it ensures starvation free process execution. However, in situations where the time quantum is deemed to be too less, the overhead caused by context switching increases rampantly. Fig. 5 can be referred for describing the execution process of RR algorithm. The processes arriving are placed into the ready queue once it reaches the ready state. The scheduler then chooses a process and checks if the burst time is less than that of the chosen time quantum and if so, that particular process

is executed. The process is preempted and the following processes are executed in a similar manner until all processes have finished execution.

E. Priority Scheduling

This algorithm executes and completes the processes based on the priority which has been assigned to each of the tasks/processes in the schedule queue. Those processes with a higher priority are executed initially while those with lower priority are processed towards the end. This algorithm can also be of preemptive or non-preemptive type. Fig. 6 describes the Priority scheduling algorithm. To begin with the processes along with their burst time and priority are input. Then they are sorted based on their priority and is processed with FCFS to further schedule the processes.

E. Hybrid Algorithm

This CPU scheduling algorithm considers the elapsed time (summation of arrival time and burst time) for organizing and sequencing the processes. Based on the elapsed time, the corresponding parameters of time for each of the processes are sorted and stored. A loop is initialized until the queue gets empty. Within the loop, all the processes are repeatedly checked at constant intervals of time and the process with the least elapsed time is executed next.

F. Web Application

In order to add an element of uniqueness and innovation to our project, we have developed an application that helps users to get a holistic idea on conventional CPU scheduling algorithms with a great visual demonstration. On opening the application, the user has the option to choose his/her preferred tab as per their interest from the menu. If the user wishes to gain more insights on the different types of CPU Scheduling Algorithms, the user can click on the scheduling algorithms tab where they can access a crisp and brief explanation of each of the scheduling algorithms. If the user wishes to obtain a Gantt chart along with other details for any number of processes, the user can navigate to the respective tab where he/she can enter values for different time parameters and obtain results. The additional feature here is that the Gantt chart will be displayed in a dynamic fashion with a timer count. Furthermore, if the user has any doubts/clarifications with regards to CPU Scheduling, he/she can chat with our virtual chatbot Lexy to clear the queries.

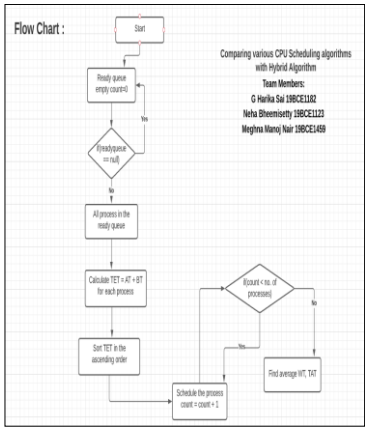

Figure 1: Flowchart of Hybrid Algorithm

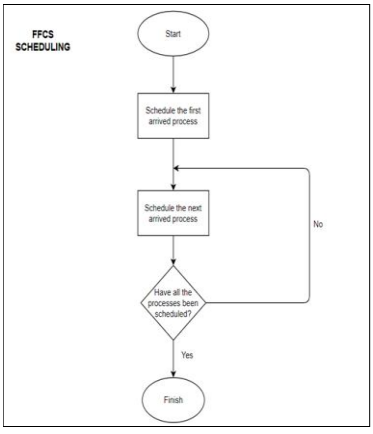

Figure 2: Flowchart of FCFS

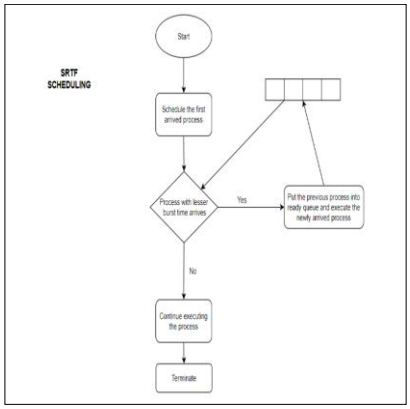

Figure 3: Flowchart of SJF

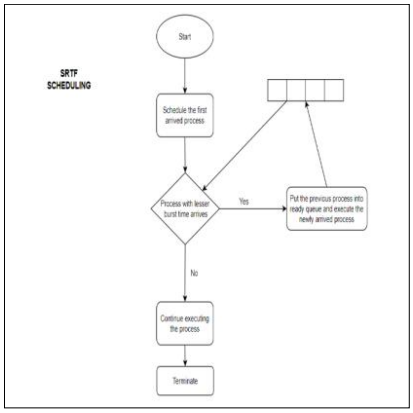

Figure 4: Flowchart of SRTF

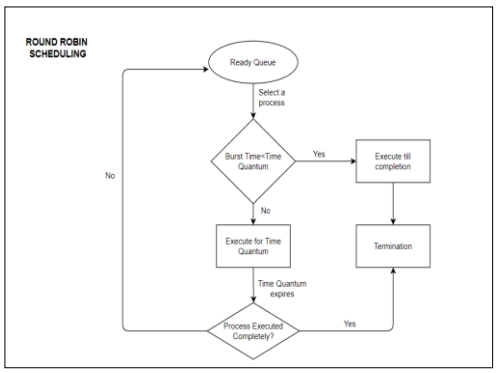

Figure 5: Flowchart of RR

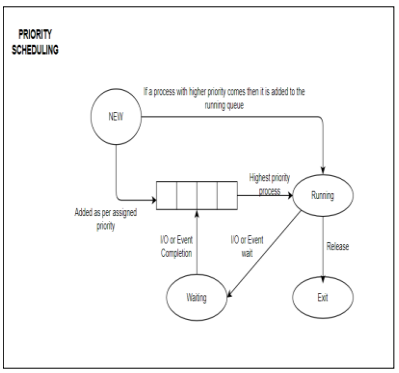

Figure 6: Flowchart of Priority

VIII. RESULTS AND DISCUSSION

This section deals with the results and implementation of the different modules involved in the curation of this project. Fig. 7 deals with all the inputs required for processing and scheduling the different tasks. It caters to the basic details of the process including that of process id, arrival time and burst time. Fig. 8 corresponds to the results obtained after scheduling the process according to Hybrid algorithm. Fig. 9 portrays the results obtained along with the turn-around time, waiting time and completion time while applying FCFS algorithm. On the other hand, Fig. 10 represents the results extracted on applying the SJF algorithm. While Fig. 10 and Fig. 11 represent the solutions obtained after the implementation of Priority and Round Robin scheduling algorithms. Fig. 12 showcases the output of Round Robin algorithm and Fig. 13 that of SRTF Algorithm. Fig. 14 represents the home page of the web application where some basic information of scheduling algorithms is provided in a carousel. Fig. 15 represents the visualization page where the user can select a scheduling algorithm of his/her preferred choice, give inputs on the processes and visualize the Gantt chart for the specified inputs and scheduling algorithm. Fig. 16 represents the results and analysis that have been observed from the comparison of hybrid and conventional CPU scheduling algorithms. Fig. 17 represents the chatbot that has been incorporated in order to clarify any basic doubts on the CPU scheduling algorithms.

```
OS J COMPONENT
19BCE1123-Bheemisetty Neha
19BCE1182-Gadiparthy Harika Sai
19BCE1459-Meghna Manoj Nair
COMPARING DIFFERENT SCHEDULING ALGORITHMS
Enter the number of processes:4
Enter the Process id, Arrival time and Burst time:-
1 0 5
Enter the Process id, Arrival time and Burst time:-
2 1 4
Enter the Process id, Arrival time and Burst time:-
3 2 2
Enter the Process id, Arrival time and Burst time:-
4 4 1
```

Figure 7: Input details of the processes from the user

```
USING HYBRID SCHEDULING:
Process AT      BT      ET      CT      TAT     WT
3       2       2       4       7       5       3
2       1       4       5       11      10      6
1       0       5       5       5       5       0
4       4       1       5       12      8       7
Average turn-around time:- 7.00
Average waiting time:- 4.00
```

Figure 8: Output for hybrid scheduling

```
USING FCFS SCHEDULING:
Process AT      BT      CT      TAT     WT
1       0       5       5       5       0
2       1       4       9       8       4
3       2       2       11      9       7
4       4       1       12      8       7
Average waiting time:- 4.50
Average turn-around time:- 7.50
```

Figure 9: Output for FCFS scheduling

```
USING SJF SCHEDULING:
Process AT      BT      CT      TAT     WT
1       0       5       5       5       0
4       4       1       6       2       1
3       2       2       8       6       4
2       1       4       12      11      7

Average waiting time =3.000000
Average Turnaround time =6.000000
```

Figure 10: Output for SJF scheduling

```
USING PRIORITY:
Enter the priority of processes:10 20 30 40

Process Priority AT     BT      CT      TAT     WT
1       10      0       5       5       5       0
2       20      1       4       9       8       4
3       30      2       2       11      9       7
4       40      4       1       12      8       7
Average waiting time is:4.500000
Average turnaroundtime is:7.500000
```

Figure 11: Output for Priority scheduling

```
USING ROUND ROBIN SCHEDULING:
Enter time quantum for round-robin scheduling:  2

process AT      BT      CT      TAT     WT
3       2       2       6       4       2
4       4       1       7       3       2
2       1       4       11      10      6
1       0       5       12      12      7
Average_waiting_time=4.250000
Average_turn_around_time=7.250000
```

Figure 12: Output for Round Robin scheduling

```
USING SRTF SCHEDULING:

Process AT      BT      CT      TAT     WT
p1      0       5       8       8       3
p2      1       4       12      11      7
p3      2       2       4       2       0
p4      4       1       5       1       0
Average waiting time =2.500000
Average Turnaround time =5.500000
```

Figure 13: Output for SRTF scheduling

Figure 14: Home page of the web application

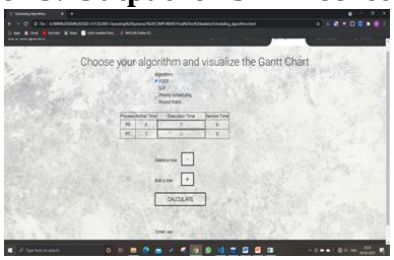

Figure 15: Visualization of Gantt chart for different scheduling algorithms

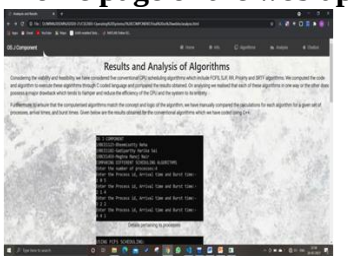

Figure 16: Results and Analysis page

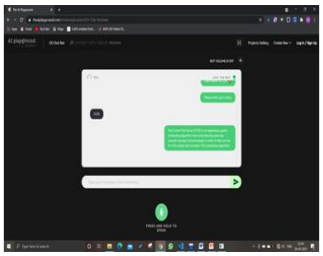

Figure 17: Chatbot page

IX. CONCLUSION

After the successful implementation of programs for each of the CPU scheduling algorithms, and on analyzing and comprehending the comparisons between them, we can clearly conclude that the proposed hybrid algorithm has better efficiency when contrasted with most of the conventional algorithms, mainly because it induces a reduced turnaround time and waiting time. This is achieved by the hybrid algorithm as it has rooted its concept from a combination of SJF and priority algorithm. If the arrival time of all the processes is the same, then the hybrid algorithm adopts the SJF approach. However, when more emphasis is given on the total elapsed time, then the priority algorithm is adopted. Therefore, we can conclude that the proposed hybrid algorithm arises to be efficient when compared to NOVEL algorithm, Priority scheduling algorithm, FCFS scheduling algorithm, RR scheduling algorithm, and SJF scheduling algorithm. Furthermore, the web application we have curated stands to be a great way for users to visualize the CPU scheduling algorithms along with the time parameters. Moreover, the virtual chatbot – Lexy in the application will be able to interact with users and clear their queries.

X. FUTURE ENHANCEMENTS

CPU Scheduling algorithm is a topic of discussion and has room for plethora of innovations and modifications to accommodate better efficiency and optimization and reduce the overheads of the CPU. We can bring about further modifications in the code which would be capable of accommodating a few more conventional algorithms in the proposed algorithm to adhere to various conditions faced by the CPU during scheduling of processes. Furthermore, the website application can be deployed with additional features, i.e. a discussion forum, where like-minded users/researchers can discuss and grow together, bringing about stronger engagement and exploration in the field of CPU scheduling algorithms.

XI. REFERENCES

[1] Bandarupalli, Sukumar Babu, et al. “A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive.” *International Journal of Modern Engineering Research*, vol. 2, no. 6, 2012, pp. 4484-4490.

[2] Abielmona, Rami. *Scheduling Algorithmic Research*, https://www.site.uottawa.ca/~rabielmo/scheduling/elg6171_FinalReport.pdf

[3] Mishra, Manish Kumar, and Faizur Rashid. “An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum.” *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 4, 2014.

[4] Shah, Syed Nasir Mehmood, et al. “Dynamic Multilevel Hybrid Scheduling Algorithms for Grid Computing.” *Procedia Computer Science*, vol. 4, 2011.

[5] Patel, Jyotirmay, and A.K. Solanki. “Performance Enhancement of CPU Scheduling by Hybrid Algorithms Using Genetic Approach.” *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 1, no. 4, 2012.

[6] Dinkar, Sahil Kumar, and Kusum Deep. “A Novel CPU Scheduling Algorithm Based

on Ant Lion Optimizer." *Soft Comp uting for Problem Solving*, 2018.

[7] Dahal, Keshav, et al. "Scheduling in Multiprocessor System Using Genetic Algorithms." 2008.

[8] Singh, Ajit, et al. "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling." *International Journal on Computer Science and Engineering*, vol. 2, no. 7, 2010.

[9] Silberschatz, Abraham, et al. *Operating System Concepts*. Wiley, 2013.

[10] Utama Siahaan, Andysah Putera. "Comparison Analysis of CPU Scheduling: FCFS, SJF and Round Robin." *International Journal of Engineering Development and Research*, vol. 4, no. 3, 2016.

[11] Alworafi, Mokhtar A., et al. "An Improved SJF Scheduling Algorithm in Cloud Computing Environment." *International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques*, 2016.

[12] Akhtar, Muhammad, et al. "An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling." *Journal of Applied Environmental Biological Sciences*, vol. 5, no. 12, 2015, pp. 42-46.

[13] Sinha, Preeti, et al. "Efficient Process Scheduling Algorithm using RR and SRTF." *International Conference on Emerging Trends in Information Technology and Engineering*, 2020.

[14] Andersson, B., et al. "Static-priority Scheduling on Multiprocessors." *IEEE Real-Time Systems Symposium*, 2001.