<u>CSE2005 – OPERATING SYSTEMS</u>

<u>Project Report</u>

Project Members:

Anirban Dutta (19BCB0015)

Kumar Sparsh (19BCB0025)

V Prashanth (19BCB0066)

Project Title:

SCHEDULING ALGORITHM

An improved Round Robin CPU Scheduling incorporated with SJF along with Dynamic Time Quantum Algorithm-Analysis and Implementation Technique proposed.

Contents covered in the Report: -

Abstract:

The Central Processing Unit (CPU) scheduling in today's generation plays a deep- seated role during designing of latest processors and units. Not only it is the premium desire of today's processing systems to efficiently switch between processes to maximize throughput, but also to do it in a way so that the idle time is reduced to a bare minimum. By switching the CPU among various processes, these targets are attempted to be achieved.

The performance of any CPU majorly depends on the scheduling algorithm used by the Processor. Primarily, the CPU is one of the most essential computer resources we have today. Not much can be done with the efficiency in executing the processes, since they take almost similar times, but a lot can be done with the scheduling. And, since the round robin scheduling algorithm is considered to be one of the most widely used algorithms, a new proposed variant of this algorithm is attempted to be established. We shall attempt improving upon the Round Robin CPU Scheduling System incorporating Shortest Job First Scheduling, achievement of stability in the Round Robin Scheduling, using the concept of Dynamic Time Quantum.

Keywords:

Operating System, CPU Scheduling, Round Robin algorithm, Shortest Job First, Dynamic Time Quantum, Waiting time, Turnaround time, Context switch.

Introduction:

As processor is the most important resource that essentially determines the working capacity of a machine, CPU scheduling becomes very important in accomplishing the operating system design goals. The target of any Operating System should be **"To allow maximum processes running at all the times in order to make best use of the CPU."** The efficiency of any CPU scheduler primarily depends on the design of the well-structured scheduling algorithms which goes well with the scheduling goals.

In this project, we propose an algorithm which can handle all types of processes with optimum scheduling criteria.

The efficiency of scheduling algorithm is measured by the following performance factors:

1. **Throughput:**
   Defined as the number of processes executed in one-unit time.
2. **Waiting- time:**
   It is the time a process has to wait for CPU in ready-queue to come in main memory.
3. **Turn-around time:**
   Defined as the amount of time required by any process to complete its execution.
4. **Response time:**
   The time between generation of a request and the first response.
5. **Context Switching:**
   The process of switching CPU between processes, also known as pre-emption.

The **CPU Utilization** is a measure of maximum usage of CPU, or how effectively CPU is made busy. The performance and effectiveness of the Round Robin algorithm is largely dependent on the value of **Time Quantum** selected. If the value of time quantum is too small then the number of context switches will be more and algorithm will not be effective. If the value of the time quantum is too large, then the algorithm will work more or less like FCFS algorithm.

Improved Round Robin with Dynamic choosing an optimum Time Quantum can significantly decrease the number of context switches, maintaining the RR nature and also can improve performance. The number of context switches can further be reduced if there is a strategy to execute processes with smallest remaining burst- times.

CPU scheduling is the basis of all operating systems. The technique used for controlling the order of job which is to be performed by a CPU of a computer is called Scheduling.

Most CPU scheduling algorithms concentrate on:

- **<u>MAXIMIZING:</u>**

1. CPU utilization
2. Through
   put And,

- **<u>MINIMIZING:</u>**

1. Turnaround time
2. Response time
3. Waiting time
4. Number of context switching for a set of requests.

Here, are concentrating on a **new Round Robin Scheduling which is designed with the Shortest job first along with using the concept of Dynamic Quantum Time.**

This Scheduling gives better results as compared to:

1. First Come First Serve (FCFS)
2. Round Robin (RR)

3. Improved Round Robin (IRR)
4. Improved Round Robin with shortest job first (IRRSJF)

The Project is about an algorithm which is a variant of RR algorithm. We attempt at briefing the proposed methodology, algorithmic procedure, pseudo code and flow chart, and illustrating it with an example, the working of the proposed algorithm. We further analyze the comparison of the proposed algorithm with other existing RR variants. Finally, we conclude and look into the future enhancements respectively.

Literature Survey:

The References have been listed in the References Section. An in-depth study of the papers and journals has been done, and the following survey has been established. We looked into the reviews, papers and journals presented by the earlier authors, and gathered following points of our own regarding the reviews.

[1]

This was one of the First greatest advancements made in the Round Robin Scheduling Algorithm. Proposed by Sir Manish Kumar Mishra [1] in 2012, the paper describes a new improved version of RR. Improved Round Robin picks the first process from the ready queue and allocate the CPU to it for a time interval of up to one time quantum. Once a process's time quantum is elapsed, it checks the remaining CPU burst times of the processes currently in execution. If the CPU burst time remaining of the current process is less than one TQ, the CPU shall again be allocated to the process currently in execution for the remaining burst time.

[2]

In year 2013, Aashna Bisht [2] performed an analysis, and proposed a work Enhanced Round Robin (ERR) model, in which, the time quantum of only those processes which require a slightly greater time than the allotted time quantum cycle were modified. The remaining process will be executed in the already proposed Round robin manner.

[3]

The Next advancement came in the year 2011, by Saroj Hiranwal [3]. In this, first of all we arrange the processes according to the execution time/burst time in increasing order that is smallest the burst time higher the priority of the running process. The smart time slice, then calculated, is the median process burst time of all CPU burst times in the ready queue.

[4]

Many other advancements came along the years between 2011 and 2014. In year 2011, H.S. Behera proposed an improved process scheduling algorithm by using dynamic quantum time along with its weighted mean.

In year 2011, Rakash Mohanty & Manas Das [4] performed a work in which a new variant of Round Robin scheduling algorithms was created by executing the processes according to the calculated Fit factor and they also used the concept of dynamic quantum time.

[5]

In year 2012, Ishwari Singh Rajput, Deepa Gupta [5] proposed priority-based Round-robin CPU scheduling algorithms is based on the integration of round robin and priority scheduling. It still holds the advantage of round robin in reducing starvation and also integrates the advantage of SJF scheduling.

[6]

In year 2012, P. Surendra Varma [6] performed a work. In this paper the quantum time is computed with the help of median and highest burst time.

[7]

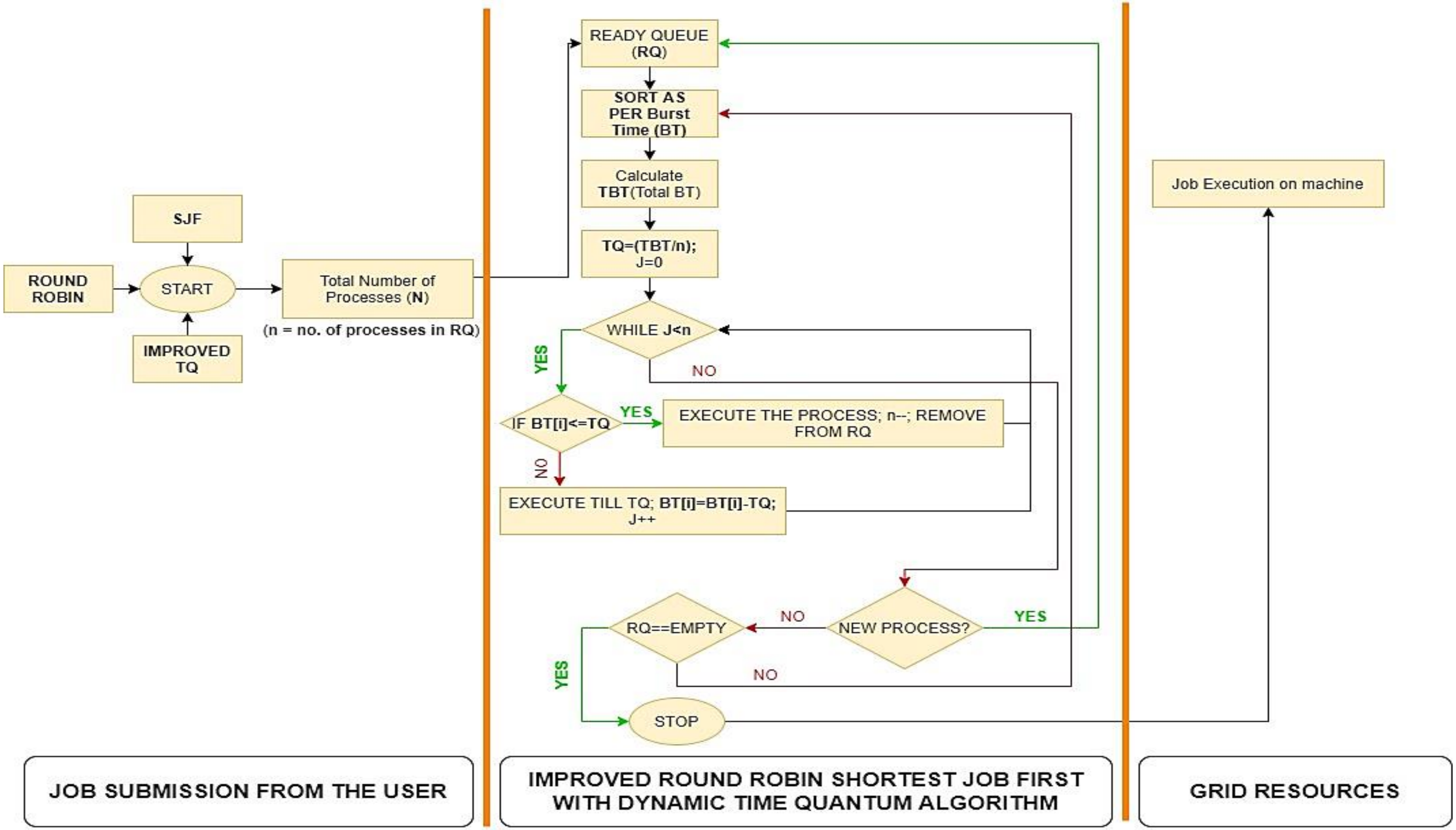
In year 2012, H.S. Behera & Brajendra Kumar Swain [7] performed a work it gives precedence to all processes according to their priority and burst time, then applies the Round Robin algorithm on it. This Proposed algorithm is developed by taking mean of dynamic quantum time in account.

Design of Proposed System:

Architecture:

- Start
- In our proposed new architecture, we are going to take round robin algorithm with improved Time Quantum combined with SJF and use it for scheduling the jobs or processes given by the user.
- This part is the job submission from user
- Now, these jobs are scheduled in the ready queue based on their arrival times.
- These assigned jobs are to sorted in the ascending order of their burst time in the ready queue.
- After sorting the given processes, we are going to calculate the total burst time for the scheduled processes in the ready queue.
- Now we calculate the time quantum using **total burst time of the scheduled processes / the no of the processes scheduled(n)** (i.e.-the number of processes in the Ready Queue). Here we take J=0, considering J as a variable.
- While J<n if (NO) it will put forth to new process.
- If (YES) it checks whether burst time is less than or equal to quantum time (BT[i]<=TQ)
- If the above condition is True (YES) then it will execute the process, do n-- and remove the process from the ready queue
- If its False (NO) then it will execute till the quantum time where the BT[i]=BT[i]-TQ, and increment value of J (J++).
- Then again it will check If J<n is True (YES) and then repeat. If False (NO) then it will assign to a new process.
- If there is no New process then it will check if Ready Queue is Empty (RQ==EMPTY).
- If the above condition is True (YES) the execution will Stop.
- If it is False (NO) then go to the process where the jobs are assigned and where all the processes are sorted in the order to the burst time which applies to the starting of the point of the ready queue.
- If it satisfies the above condition then the job execution is complete for the given processes
- End

Flowchart & Working:



Implementation:

Source Code in C++:

```
#include <bits/stdc++.h>
using namespace std;

struct Process {
	int p; // Process ID
	int bt; // Burst Time
	int at; // Arrival Time
	int rt; // Remaining Burst Time
	int wt; // Waiting Time
	int tat; //Turn Around Time
}p[20], temp, org[20];

struct comp{
	char algo[20];
	float avg_wt;
	float avg_tat;
}algo[20];

void menu();
void input();
void assign();
void comparator();

void fcfs(int);
void rr(int);
void irr(int);
void irrsjf(int);
void irrsjfdtq(int);

int n;

int main()
{
	cout<<"##############################  WELCOME TO OS PROJECT
##############################\n\n";
	menu();
	return 0;
}

void menu()
{
	int ch;
	printf("\n\nTable\n\n1.New   Input\n2.FCFS   Algorithm\n3.RR
Algorithm\n4.IRR   Algorithm\n5.IRR_SJF   Algorithm\n6.IRR_SJF_DTQ
Algorithm\n7.Compare All\n8.Exit");
	printf("\n\nEnter your choice from the above table : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:input();break;
			case 2:fcfs(1);break;
			case 3:rr(1);break;
			case 4:irr(1);break;
			case 5:irrsjf(1);break;
			case 6:irrsjfdtq(1);break;
			case 7:comparator();break;
			case 8:exit(0);break;
			default: printf("\n\nPlease enter choice from 1 to
8 only\n");
					menu();
```

```
		}
	}

	void input()
	{
		cout<<"\nEnter Number of processes : ";
		cin>>n;

		cout<<"\n\nEnter the Arrival Time of the Processes : \n";
		for(int i=0; i<n; i++)
		{
			cout<<"Process "<<i+1<<" : ";
			cin>>org[i].at;
			org[i].p=i+1;
		}

		cout<<"\n\nEnter the Burst Time of the Processes : \n";
		for(int i=0;i<n;i++)
		{
			cout<<"Process "<<i+1<<" : ";
			cin>>org[i].bt;
			org[i].rt=org[i].bt;
		}

		menu();
	}

	void sort_at()
	{
		for(int i=0; i<n-1; i++)
		{
			for(int j=0; j<n-i-1; j++)
			{
				if(p[j].at>p[j+1].at)
				{
					temp=p[j];
					p[j]=p[j+1];
					p[j+1]=temp;
				}
			}
		}
	}

	void assign()
	{
		for(int i=0; i<n; i++)
			p[i]=org[i];
	}

	void fcfs(int menu_flag)
	{
		cout<<"\n\n############		First	Come	First	Serve
	############\n\n";
		assign();
		sort_at();
		int sum=0, tot_tat=0, tot_wt=0;
		for(int i=0; i<n; i++)
		{
			p[i].wt=sum-p[i].at;
			if(p[i].wt<0)
				p[i].wt=0;
			p[i].tat=p[i].bt+p[i].wt;
			sum+=p[i].bt;
			tot_tat+=p[i].tat;
			tot_wt+=p[i].wt;
		}
```

```
     cout<<"Process    Burst Time    Arrival Time    Waiting Time
Turn Around Time\n";
     for(int i=0; i<n; i++)
          cout<<"
"<<p[i].p<<"\t\t"<<p[i].bt<<"\t\t"<<p[i].at<<"\t\t"<<p[i].wt<<"\t\
t"<<p[i].tat<<"\n";
     cout<<"\nAverage Waiting Time = "<<(float)tot_wt/n;
     cout<<"\nAverage Turn Around Time = "<<(float)tot_tat/n<<"\n";
     strcpy(algo[0].algo,"fcfs");
     algo[0].avg_tat=(float)tot_tat/n;
     algo[0].avg_wt=(float)tot_wt/n;
     if(menu_flag)
          menu();
}

void rr(int menu_flag)
{
  cout<<"\n\n############  Round Robin ############\n\n";
  assign();
  int remain=n, time_quantum, tot_wt=0, tot_tat=0, flag=0;
  sort_at();
  cout<<"\nEnter Time Quantum : ";
  cin>>time_quantum;
  for(int i=0,time=0; remain!=0;)
  {
    if(p[i].rt<=time_quantum && p[i].rt>0)
    {
      time+=p[i].rt;
      p[i].rt=0;
      flag=1;
    }
    else if(p[i].rt>0)
    {
      p[i].rt-=time_quantum;
      time+=time_quantum;
    }
    if(p[i].rt==0 && flag==1)
    {
      remain--;
      p[i].wt=time-p[i].at-p[i].bt;
        p[i].tat=time-p[i].at;
      tot_wt+=p[i].wt;
      tot_tat+=p[i].tat;
      flag=0;
    }
    if(i==n-1)
      i=0;
    else if(p[i+1].at<=time)
      i++;
    else
      i=0;
  }
  cout<<"Process     Burst Time     Arrival Time     Waiting Time
Turn Around Time\n";
     for(int i=0; i<n; i++)
          cout<<"
"<<p[i].p<<"\t\t"<<p[i].bt<<"\t\t"<<p[i].at<<"\t\t"<<p[i].wt<<"\t\
t"<<p[i].tat<<"\n";
     cout<<"\nAverage Waiting Time = "<<(float)tot_wt/n;
     cout<<"\nAverage Turn Around Time = "<<(float)tot_tat/n<<"\n";
     strcpy(algo[1].algo,"rr");
     algo[1].avg_tat=(float)tot_tat/n;
     algo[1].avg_wt=(float)tot_wt/n;
     if(menu_flag)
          menu();
}
```

```
void irr(int menu_flag)
{
	cout<<"\n\n############          Improved     Round     Robin
############\n\n";
	assign();
  int remain=n, rt[n], time_quantum, tot_wt=0, tot_tat=0, flag=0;
  sort_at();
  for(int i=0; i<n; i++)
    rt[p[i].p-1]=p[p[i].p-1].bt;
  cout<<"\nEnter Time Quantum : ";
  cin>>time_quantum;
  for(int i=0,time=0; remain!=0;)
  {
    if(p[i].rt<=time_quantum && p[i].rt>0)
    {
      time+=p[i].rt;
      p[i].rt=0;
      flag=1;
    }
    else if(p[i].rt>time_quantum && p[i].rt>0 && time_quantum*2-
p[i].rt>=0)
    {
      time+=p[i].rt;
      p[i].rt=0;
      flag=1;
    }
    else if(p[i].rt>0)
    {
      p[i].rt-=time_quantum;
      time+=time_quantum;
    }
    if(p[i].rt==0 && flag==1)
    {
      remain--;
      p[i].wt=time-p[i].at-p[i].bt;
        p[i].tat=time-p[i].at;
      tot_wt+=p[i].wt;
      tot_tat+=p[i].tat;
      flag=0;
    }
    if(i==n-1)
      i=0;
    else if(p[i+1].at<=time)
      i++;
    else
      i=0;
  }
  cout<<"Process     Burst Time     Arrival Time     Waiting Time
Turn Around Time\n";
	for(int i=0; i<n; i++)
		cout<<"
"<<p[i].p<<"\t\t"<<p[i].bt<<"\t\t"<<p[i].at<<"\t\t"<<p[i].wt<<"\t\
t"<<p[i].tat<<"\n";
	cout<<"\nAverage Waiting Time = "<<(float)tot_wt/n;
	cout<<"\nAverage Turn Around Time = "<<(float)tot_tat/n<<"\n";
	strcpy(algo[2].algo,"irr");
	algo[2].avg_tat=(float)tot_tat/n;
	algo[2].avg_wt=(float)tot_wt/n;
	if(menu_flag)
		menu();
}

void irrsjf(int menu_flag)
{
```

```
    cout<<"\n\n#############   Imporved  Roubd  Robin  Shortest  Job
First  #############\n\n";
   int time_quantum,tot_tat=0, tot_wt=0;
   int remain=n;
   int time=0;
    int old_mean=0;

   cout<<"\nEnter Time Quantum : ";
    cin>>time_quantum;

   for(int i=0; i<n; i++)
        p[i].rt=p[i].bt;
    while(remain>0)
    {
        //sorting in ascending order
        for(int i=0;i<n;i++)
            for(int j=0;j<n-i-1;j++)
                if(p[j].rt>p[j+1].rt)
                {
                    temp=p[j+1];
                    p[j+1]=p[j];
                    p[j]=temp;
                }
        for(int i=0;i<n;i++) //loop main
        {
            if(p[i].at<=time && p[i].rt!=0)
        {
            if(p[i].rt<time_quantum && p[i].rt!=0)
            {
                  time+=p[i].rt;
                  p[i].rt=0;
            }
            else   if(p[i].rt>time_quantum   &&   p[i].rt>0   &&
time_quantum*2-p[i].rt>=0)
                 {
                   time+=p[i].rt;
                   p[i].rt=0;
                 }
                  else
                  {
                    time+=time_quantum;
                    p[i].rt-=time_quantum;
            }

                  if(p[i].rt==0)
            {
                    p[i].wt=time-p[i].bt-p[i].at;
                    p[i].tat=time-p[i].at;
                      remain--;
            }
        }
        }
    }
    for(int i=0; i<n; i++)
    {
        tot_wt += p[i].wt;
        tot_tat += p[i].tat;
    }
    cout<<"Process     Burst Time     Arrival Time     Waiting Time
Turn Around Time\n";
    for(int i=0; i<n; i++)
        cout<<"
"<<p[i].p<<"\t\t"<<p[i].bt<<"\t\t"<<p[i].at<<"\t\t"<<p[i].wt<<"\t\
t"<<p[i].tat<<"\n";
    cout<<"\nAverage Waiting Time = "<<(float)tot_wt/n;
    cout<<"\nAverage Turn Around Time = "<<(float)tot_tat/n<<"\n";
```

```
        strcpy(algo[3].algo,"irr_sjf");
        algo[3].avg_tat=(float)tot_tat/n;
        algo[3].avg_wt=(float)tot_wt/n;
        if(menu_flag)
              menu();
}


void irrsjfdtq(int menu_flag)
{
        cout<<"\n\n#############   Improved  Round  Robin  Shortest  Job
First with Dynamic Quantum Time  #############\n\n";
      int time_quantum,tot_tat=0, tot_wt=0;
      int remain=n;
      int time=0;
      int max_bt;
      int sum_bt,prq;  // prq = process in ready queue
        int old_mean=0;

      for(int i=0; i<n; i++)
      {
              p[i].rt=p[i].bt;
              old_mean+=p[i].bt;
        }
        old_mean/=n;
        while(remain>0)
        {
              //sorting in ascending order
              for(int i=0;i<n;i++)
                    for(int j=0;j<n-i-1;j++)
                          if(p[j].rt>p[j+1].rt)
                          {
                                temp=p[j+1];
                                p[j+1]=p[j];
                                p[j]=temp;
                          }
              for(int i=0;i<n;i++) //loop main
              {
                    prq=0;
              max_bt=0;
              sum_bt=0;
              for(int j=0;j<n;j++)
              {
                          if(p[j].at<=time && p[j].rt!=0)
                          {
                          prq++;
                          if(p[j].rt>max_bt)
                                max_bt=p[j].rt;
                          sum_bt+=p[j].rt;
                    }
              }

                    if(prq==0 ||  (sum_bt/prq) == 0)
                    time_quantum = old_mean;
              else
              {
                          time_quantum=sum_bt/prq;
                          old_mean = time_quantum;
                    }
                    if(p[i].at<=time && p[i].rt!=0)
              {
                    if(p[i].rt<time_quantum && p[i].rt!=0)
                    {
                            time+=p[i].rt;
                            p[i].rt=0;
                    }
```

```
                else  if(p[i].rt>time_quantum  &&  p[i].rt>0  &&
time_quantum*2-p[i].rt>=0)
                    {
                     time+=p[i].rt;
                     p[i].rt=0;
                    }
                     else
                     {
                        time+=time_quantum;
                        p[i].rt-=time_quantum;
                }

                     if(p[i].rt==0)
                {
                        p[i].wt=time-p[i].bt-p[i].at;
                        p[i].tat=time-p[i].at;
                           remain--;
                }
           }
           }
     }
     for(int i=0; i<n; i++)
     {
          tot_wt += p[i].wt;
          tot_tat += p[i].tat;
     }
     cout<<"Process    Burst Time    Arrival Time    Waiting Time
Turn Around Time\n";
     for(int i=0; i<n; i++)
          cout<<"
"<<p[i].p<<"\t\t"<<p[i].bt<<"\t\t"<<p[i].at<<"\t\t"<<p[i].wt<<"\t\
t"<<p[i].tat<<"\n";
     cout<<"\nAverage Waiting Time = "<<(float)tot_wt/n;
     cout<<"\nAverage Turn Around Time = "<<(float)tot_tat/n<<"\n";
     strcpy(algo[4].algo,"irr_sjf_dtq");
     algo[4].avg_tat=(float)tot_tat/n;
     algo[4].avg_wt=(float)tot_wt/n;
     if(menu_flag)
          menu();
}

void comparator()
{
     fcfs(0);
     rr(0);
     irr(0);
     irrsjf(0);
     irrsjfdtq(0);
     cout<<"\n\n############  Comparing All  ############\n\n";
     cout<<"\n\n Algorithm\t\tAVG_WT\t\tAVG_TAT\n\n";
     for(int i=0; i<5; i++)
          if(strlen(algo[i].algo)>4)
               cout<<"
"<<algo[i].algo<<"\t\t"<<algo[i].avg_wt<<"\t\t"<<algo[i].avg_tat<<
"\n";
          else
               cout<<"
"<<algo[i].algo<<"\t\t\t"<<algo[i].avg_wt<<"\t\t"<<algo[i].avg_tat
<<"\n";
     menu();
}
```

Example Explanation:

Let the Burst Time and the Arrival Times of the Processes be:

PROCESS NUMBER	ARRIVAL TIME	BURST TIME
P1	**20**	**52**
P2	**0**	**22**
P3	**10**	**35**
P4	**15**	**80**

Total Number of Processes (N) = 4

While scheduling the processes we maintain a ready queue and a Gantt Chart.
Initially at time T=0.
We have only one process P2 in the **Ready Queue** having arrival time 0.

P2(BT=22)				

T=0

So, number of processes in ready queue (n) = 1.
Here since we have only one process no need to sort according to burst time (BT).
Total Burst Time (TBT) = 22.
Time Quantum (TQ) = TBT/n = 22/1 = 22. Initialize J=0.
Check IF J<n (0<1,True)
    Check IF BT<=TQ (22<=22,True)
        Execute process P2 and remove from ready queue, n-- .

Gantt Chart:

P2				

T=0 T=22

Now n=0.
Again we check J<n (0<0,False)
    Check if New Processes have arrived by T=22 (True).

Ready Queue:

P3(BT=35)	P4(BT=80)	P1(BT=52)		

T=10 T=15 T=20

So, Number of Processes in Ready Queue (n) = 3.
Now we sort the processes according to burst time (BT).

P3(BT=35)
P1(BT=52)
P4(BT=80)

So, The Processes will now execute in this order (Shortest Job First).
Total Burst Time (TBT) = 35+52+80 = 167
Time Quantum (TQ) = TBT/n = 167/3 = 55.67. Initialise J=0.

Check IF J<n (0<3,True)
Check IF BT[i]<=TQ (35<=55.67,True)
Execute process P3 and remove from ready queue, n-- .

Gantt Chart:

P2	P3			

T=0 T=22 T=57

Now n=2.
Again we Check J<n (0<2,True)
Check IF BT[i]<=TQ (52<=55.67,True)
Execute process P1 and remove from ready queue, n-- .

Gantt Chart:

P2	P3	P1		

T=0 T=22 T=57 T=109

Now n=1.
Again we Check J<n (0<1,True)
Check IF BT[i]<=TQ (80<=55.67,False)
Execute P4 till BT[i]=BT[i]-TQ (BT=80 - 55.67=24.33), J++ (increment value of J)

Gantt Chart:

P2	P3	P1	P4	

T=0 T=22 T=57 T=109 T=164.67

Again we check J<n (1<1,False)
Check if New Processes have arrived by T=164.67 (False).
Check if Ready Queue is Empty (False, some part of P4 is left).

Ready Queue:

P4(RT=24.33)				

So, number of processes in ready queue (n) = 1.
Here since we have only one process no need to sort according to burst time(BT).
Total Burst Time (TBT) = 24.33.
Time Quantum (TQ) = TBT/n = 24.33/1 = 24.33. Initialise J=0.
Check IF J<n (0<1,True)
Check IF BT[i]<=TQ (24.33<=24.33,True)
Execute process P4 and remove from ready queue, n-- .

Final Gantt Chart:

P2	P3	P1	P4	P4

T=0 T=22 T=57 T=109 T=164.67 T=189

Now n=0.
Again we check J<n (0<0,False)
Check if New Processes have arrived by T=189 (False).
Check if Ready Queue is Empty (True).
Hence, we stop the CPU scheduling process as all the processes have been executed.

Final synopsis of our scheduling algorithm:

Process Number	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	20	52	109	89	37
P2	0	22	22	22	0
P3	10	35	57	47	12
P4	15	80	189	174	94

Average Turn Around Time = (89+22+47+174)/4 = 83.
Average Waiting Time = (37+0+12+94)/4 = 35.75.

Results & Discussion:

Case Study:

To test how effective is our newly proposed model, we shall take up a scenario with 4 Processes to be executed by the CPU, by different scheduling algorithms, and we shall study the following:

1. Average Waiting Time.

2. Average Turnaround Time.

Lesser the Average Waiting Time and Turnaround time, better is the Scheduling Algorithm.

Case 1:

Let the Burst Time and the Arrival Times of the Processes be:

PROCESS NUMBER	ARRIVAL TIME	BURST TIME
P1	**20**	**52**
P2	**0**	**22**
P3	**10**	**35**
P4	**15**	**80**

We shall now compare the **Average Waiting and Turnaround time** for the Newly Proposed Algorithm and compare it with the previous existing Scheduling Algorithms.

Gantt Chart for The Processes:

1) **First Come First Serve (FCFS SCHEDULING ALGORITHM)**

P2	P3	P4	P1	
T=0	T=22	T=57	T=137	T=189

Average Waiting Time = **42.75**, Average Turnaround Time = **90.**

2) **Round Robin (RR SCHEDULING ALGORITHM with Time Quantum (TQ) =20))**

P2	P3	P4	P1	P2	P3	P4	P1	P4	P1	P4	
T=0	20	40	60	80	82	97	117	137	157	169	189

Average Waiting Time = **75.75**, Average Turnaround Time = **123.**

3) Improved Round Robin (IRR SCHEDULING ALGORITHM with Time Quantum (TQ) = 20)

P2	P2	P3	P3	P4	P1	P4	P1	P1	P4	P4	
T=0	20	22	42	57	77	97	117	137	149	169	189

Average Waiting Time = **45.75**, Average Turnaround Time = **93.**

4) Improved Round Robin with SJF (IRR-SJF SCHEDULING ALGORITHM with Time Quantum (TQ) = 20)

P2	P2	P3	P3	P1	P4	P1	P1	P4	P4	P4	
0	20	22	42	57	77	97	117	129	149	169	189

Average Waiting Time = **40.75**, Average Turnaround Time = **88.**

5) Improved Round Robin with SJF with Dynamic Time Quantum (IRR-SJF-DQ SCHEDULING ALGORITHM with Time Quantum (TQ) = 20) [PROPOSED ALGORITHM]

P2	P3	P1	P4	P4	
T=0	T=22	T=57	T=109	T=163.67	T=189

Average Waiting Time = **35.75**, Average Turnaround Time = **83.**

Comparison of all the Scheduling Algorithms:

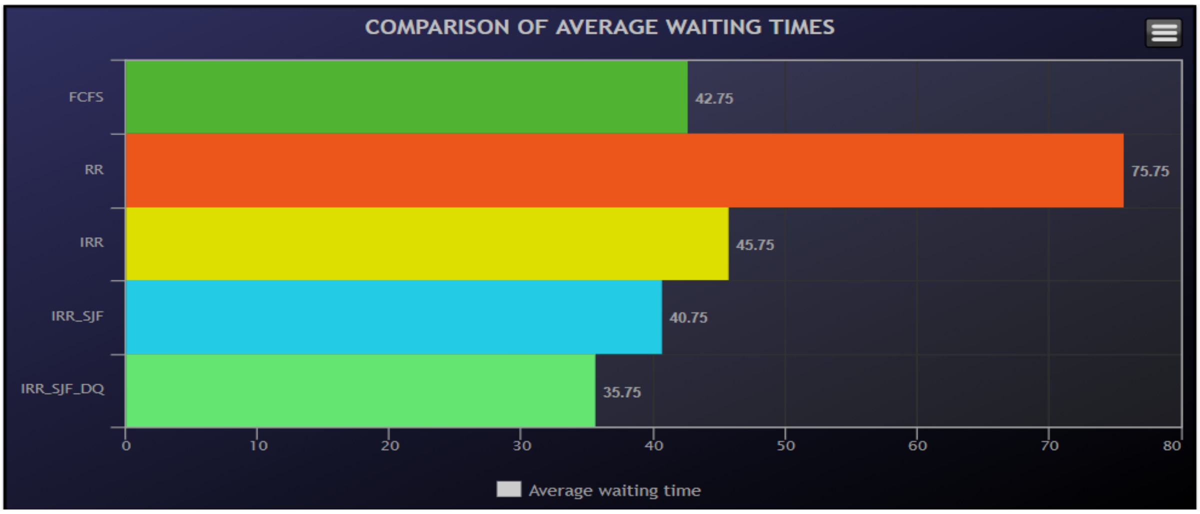


Figure 1 : Graphical comparison of Average Waiting Time for different algorithms

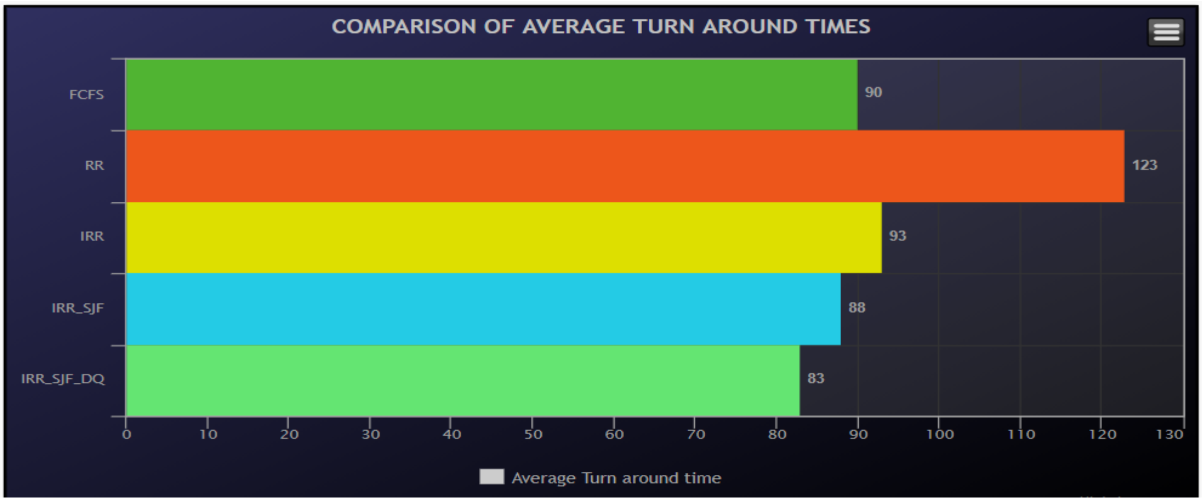
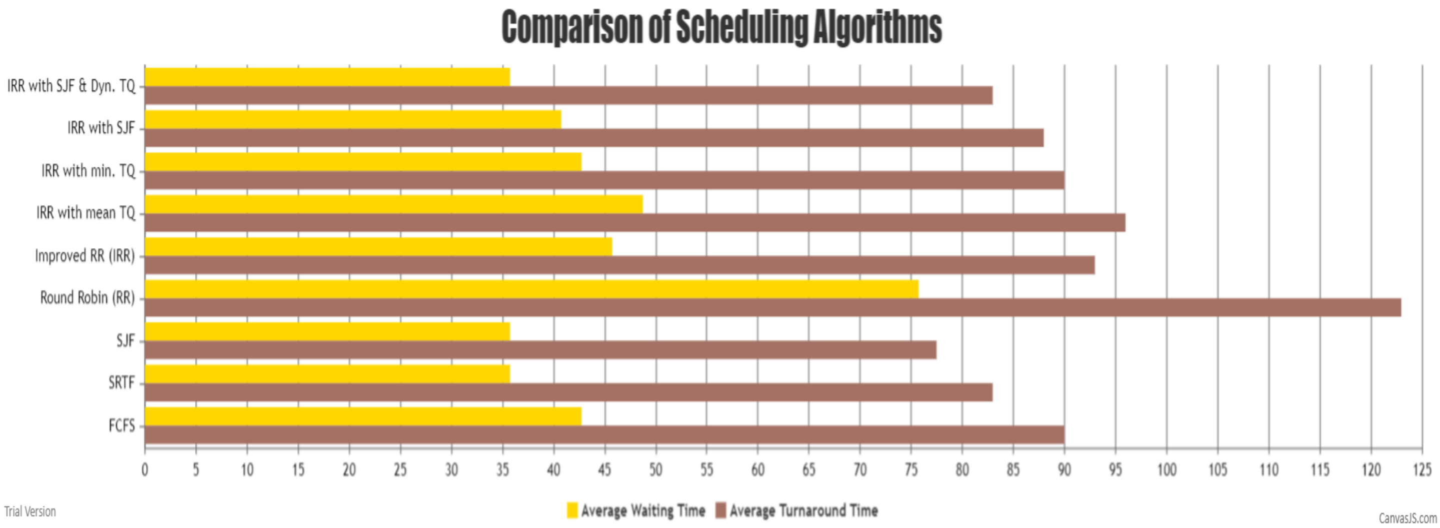


Figure 2 : Graphical comparison of Average Turnaround Time for different algorithms

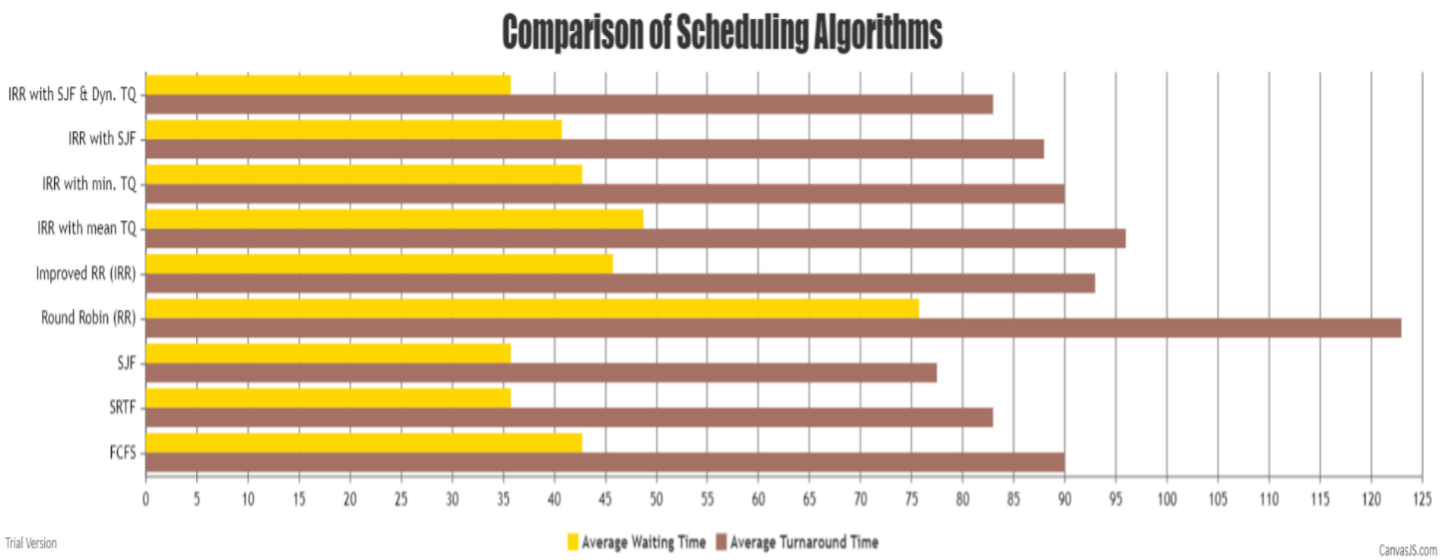
Frontend Implementation:



Case 2:

PROCESS NUMBER	ARRIVAL TIME	BURST TIME
P1	3	32
P2	4	43
P3	0	21
P4	5	54

Frontend Implementation:



Terminal Output (Case - 1):

```
###############################  WELCOME TO OS PROJECT  ###############################

Table
1.New Input
2.FCFS Algorithm
3.RR Algorithm
4.IRR Algorithm
5.IRR_MEAN_TQ  Algorithm
6.IRR_MIN_TQ  Algorithm
7.IRR_SJF  Algorithm
8.IRR_SJF_DTQ  Algorithm
9.Compare All
10.Exit
Enter your choice from the above table : 1

Enter Number of processes : 4

Enter the Burst Time of the Processes :
Process 1 : 52
Process 2 : 22
Process 3 : 35
Process 4 : 80

Enter the Arrival Time of the Processes :
Process 1 : 20
Process 2 : 0
Process 3 : 10
Process 4 : 15

Table
1.New Input
2.FCFS Algorithm
3.RR Algorithm
4.IRR Algorithm
5.IRR_MEAN_TQ  Algorithm
6.IRR_MIN_TQ  Algorithm
7.IRR_SJF  Algorithm
8.IRR_SJF_DTQ  Algorithm
9.Compare All
10.Exit
Enter your choice from the above table : 9


#############  First Come First Serve  #############

Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2           22             0               0               22
   3           35             10              12              47
   4           80             15              42              122
   1           52             20              117             169

Average Waiting Time = 42.75
Average Turn Around Time = 90


#############  Shortest Job First  #############

Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2           22             0               0               0
   3           35             10              12              47
   1           52             20              37              89
   4           80             15              94              174

Average Waiting Time = 35.75
```

```
Average Turn Around Time = 77.5


#############  Round Robin  #############


Enter Time Quantum : 20
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               60              82
   3          35             10              52              87
   4          80             15              94              174
   1          52             20              97              149

Average Waiting Time = 75.75
Average Turn Around Time = 123

#############  Improved Round Robin  #############

Enter Time Quantum : 20
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               0               22
   3          35             10              12              47
   4          80             15              94              174
   1          52             20              77              129

Average Waiting Time = 45.75
Average Turn Around Time = 93

#############  Improved Round Robin with Mean Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               0               22
   3          35             10              12              47
   4          80             15              94              174
   1          52             20              89              141

Average Waiting Time = 48.75
Average Turn Around Time = 96

#############  Improved Round Robin with Min Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               0               22
   3          35             10              12              47
   4          80             15              42              122
   1          52             20              117             169

Average Waiting Time = 42.75
Average Turn Around Time = 90

#############  Improved Round Robin Shortest Job First  #############

Enter Time Quantum : 20
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               0               22
   3          35             10              12              47
   1          52             20              57              109
   4          80             15              94              174

Average Waiting Time = 40.75
Average Turn Around Time = 88

#############  Improved Round Robin Shortest Job First with Dynamic Time Quantum  #####
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   2          22             0               0               22
   3          35             10              12              47
```

```
   1            52              20              37              89
   4            80              15              94              174

Average Waiting Time = 35.75
Average Turn Around Time = 83


#############  Shortest Remaining Time First  #############

Process     Burst Time     Arrival Time     Waiting Time     Turn Around Time
   1            52              20              37              89
   2            22              0               0               22
   3            35              10              12              47
   4            80              15              94              174

Average Waiting Time = 35.75
Average Turn Around Time = 83

#############  Comparing All  #############

 Algorithm                AVG_WT          AVG_TAT
   fcfs                   42.75           90
   rr                     75.75           123
   irr                    45.75           93
   irr_mean_tq            48.75           96
   irr_min_tq             42.75           90
   irr_sjf                40.75           88
   irr_sjf_dtq            35.75           83
   sjf                    35.75           77.5
   srtf                   35.75           83
```

Terminal Output (Case - 2):

```
root@kali:~# ./schedule1
###############################  WELCOME TO OS PROJECT  ###############################

Table
1.New Input
2.FCFS Algorithm
3.RR Algorithm
4.IRR Algorithm
5.IRR_MEAN_TQ  Algorithm
6.IRR_MIN_TQ  Algorithm
7.IRR_SJF  Algorithm
8.IRR_SJF_DTQ  Algorithm
9.Compare All
10.Exit
Enter your choice from the above table : 1

Enter Number of processes : 4

Enter the Burst Time of the Processes :
Process 1 : 32
Process 2 : 43
Process 3 : 21
Process 4 : 54

Enter the Arrival Time of the Processes :
Process 1 : 3
Process 2 : 4
Process 3 : 0
Process 4 : 5

Table
1.New Input
2.FCFS Algorithm
3.RR Algorithm
4.IRR Algorithm
5.IRR_MEAN_TQ  Algorithm
6.IRR_MIN_TQ  Algorithm
7.IRR_SJF  Algorithm
8.IRR_SJF_DTQ  Algorithm
9.Compare All
10.Exit
Enter your choice from the above table : 9
```

```
#############  Improved Round Robin with Mean Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Improved Round Robin with Min Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Improved Round Robin Shortest Job First  #############

Enter Time Quantum : 2
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               48              69
   1          32             3               74              106
   2          43             4               89              132
   4          54             5               91              145

Average Waiting Time = 75.5
Average Turn Around Time = 113

#############  Improved Round Robin Shortest Job First with Dynamic Time Quantum  #####
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  First Come First Serve  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Shortest Job First  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               0
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 71.75

#############  Round Robin  #############

Enter Time Quantum : 2
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               54              75
   1          32             3               78              110
   2          43             4               91              134
   4          54             5               91              145

Average Waiting Time = 78.5
Average Turn Around Time = 116

#############  Improved Round Robin  #############

Enter Time Quantum : 2
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               48              69
   1          32             3               74              106
   2          43             4               89              132
   4          54             5               91              145

Average Waiting Time = 75.5
Average Turn Around Time = 113

#############  Improved Round Robin with Mean Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
```

```
#############  Improved Round Robin with Mean Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Improved Round Robin with Min Time Quantum  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Improved Round Robin Shortest Job First  #############

Enter Time Quantum : 2
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               48              69
   1          32             3               74              106
   2          43             4               89              132
   4          54             5               91              145

Average Waiting Time = 75.5
Average Turn Around Time = 113

#############  Improved Round Robin Shortest Job First with Dynamic Time Quantum
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   3          21             0               0               21
   1          32             3               18              50
   2          43             4               49              92
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77
```

```
#############  Shortest Remaining Time First  #############
Process    Burst Time    Arrival Time    Waiting Time    Turn Around Time
   1          32             3               18              50
   2          43             4               49              92
   3          21             0               0               21
   4          54             5               91              145

Average Waiting Time = 39.5
Average Turn Around Time = 77

#############  Comparing All  #############

 Algorithm             AVG_WT          AVG_TAT
   fcfs                39.5            77
   rr                  78.5            116
   irr                 75.5            113
   irr_mean_tq         39.5            77
   irr_min_tq          39.5            77
   irr_sjf             75.5            113
   irr_sjf_dtq         39.5            77
   sjf                 39.5            71.75
   srtf                39.5            77
```

Conclusion:

From the above Experiments, IRR-SJF with Dynamic TQ Algorithm shows higher results than all the variants of the Round Robin Algorithm, in enhancing the Central Processing Unit Performance, and its Potency. By using our Algorithm, We end up getting the least Average Waiting Time and Average Turnaround Time. In future, we can implement this algorithmic program in real time OS.

References:

[1] Manish Kumar Mishra, "Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in computer science, ISSN - 2229-371X, vol. 3, No. 6, June 2012

[2] Aashna Bisht, "Enhanced Round Robin Algorithm for process scheduling using varying quantum precision", IRAJ International Conference-proceedings of ICRIEST-AICEEMCS,29th Dec 2013, Pune, India.

[3] Saroj Hiranwal, "Adaptive Round Robin Scheduling using shortest Burst Approach Based on smart time slice", International Journal of Data Engineering (IJDE), volume2, Issue 3,2 011

[4] Lalit Kishor& Dinesh Goyal, "Time Quantum Based Improved Scheduling Algorithms", International Journal of Advanced Research in Computer science and Software Engineering, ISSN: 2277-128X, Volume 3, Issue 4, April 2013.

[5] Ishwari Singh Rajput, Deepa Gupta, "A Priority based round robin CPU Scheduling Algorithms for real time systems", International journal of Innovations in Engineering and Technology, ISSN:2319-1058, Vol. 1 ISSUE 3rd Oct 2012.

[6] P. Surendra Varma, "A Best possible Time quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) algorithms", International Journal of advanced Research in computer science and software Engineering, ISSN: 2277 128X, Vol. 2, ISSUE 11, November 2012.

[7] H.S. Behera & Brajendra Kumar Swain, "A New proposed precedence based Round Robin with dynamic time quantum Scheduling algorithm for soft real time systems", International Journal of advanced Research in Computer science and software Engineering, ISSN:2277- 128X, Vol. 2, ISSUE 6, June 2012.