BVRIT HYDERABAD College of Engineering for Women
(AUTONOMOUS)
(Approved by AICTE | Affiliated to JNTUH | Accredited by NAAC with Grade 'A' &NBA for CSE, ECE, EEE, & IT)
Bachupally, Hyderabad-500090
Department of CSE(Artificial Intelligence and Machine Learning)

OS PROJECT
Process Scheduling Simulator

22wh1a6604-Sneha
22wh1a6614-Aimen
22wh1a6624-Bhavya
22wh1a6634-Lasya
22wh1a6644-Aishwarya
22wh1a6654-Geetika
22wh1a6664-Pavani

Problem Statement:

Design and implement a process scheduling simulator that allows users to explore and analyse the behaviour of different process scheduling algorithms in an operating system. The simulator should simulate the execution of processes with varying burst times, priorities, and arrival times, providing insights into the efficiency and performance of different scheduling strategies.

Abstract:

This project introduces a dynamic process scheduling simulator designed to explore and analyze the intricacies of various process scheduling algorithms within an operating system environment. The simulator encompasses a range of features, allowing users to gain insights into the efficiency and performance of different scheduling strategies.

The key components of the simulator include a robust process generation module, capable of creating diverse sets of random processes with attributes such as burst time, priority, and arrival time. Users can select from a variety of scheduling algorithms, including First-Come, First-Serve, Shortest Job Next, Round Robin, and Priority Scheduling, to observe their impact on system behavior.

The simulator provides a visually intuitive representation of the timeline of process execution, illustrating when each process obtains CPU time, experiences waiting periods, and ultimately completes its execution. Performance metrics, such as turnaround time, waiting time, and response time, are calculated and presented for each process, offering a comprehensive overview of system performance.

To enhance user interaction, the project incorporates a user-friendly interface, allowing users to input parameters interactively, choose scheduling algorithms, and visualise simulation results. The project further facilitates a comparative analysis of different scheduling algorithms, aiding users in making informed decisions based on generated metrics.

Code:

```c
#include <stdio.h>

struct Process {
    int pid;
    int burst_time;
    int arrival_time;
    int waiting_time;
    int turnaround_time;
};

void calculateWaitingTime(struct Process processes[], int n) {
    int total_waiting_time = 0;
    int total_turnaround_time = 0;

    processes[0].waiting_time = 0;

    for (int i = 1; i < n; i++) {
        processes[i].waiting_time = processes[i-1].burst_time + processes[i-1].waiting_time - processes[i].arrival_time;
        if (processes[i].waiting_time < 0) {
            processes[i].waiting_time = 0;
        }
    }

    for (int i = 0; i < n; i++) {
        processes[i].turnaround_time = processes[i].burst_time + processes[i].waiting_time;

        total_waiting_time += processes[i].waiting_time;
        total_turnaround_time += processes[i].turnaround_time;
    }

    float average_waiting_time = (float) total_waiting_time / n;
```

```c
    float average_turnaround_time = (float) total_turnaround_time / n;

    printf("Process\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst_time, processes[i].arrival_time,
               processes[i].waiting_time, processes[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f\n", average_waiting_time);
    printf("Average Turnaround Time: %.2f\n", average_turnaround_time);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        printf("Enter details for Process %d:\n", i+1);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        printf("Arrival Time: ");
        scanf("%d", &processes[i].arrival_time);
        processes[i].pid = i+1;
    }

    calculateWaitingTime(processes, n);
```

```c
    printf("Process\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst_time, processes[i].arrival_time,
               processes[i].waiting_time, processes[i].turnaround_time);
    }

    printf("\nAverage Waiting Time: %.2f\n", average_waiting_time);
    printf("Average Turnaround Time: %.2f\n", average_turnaround_time);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        printf("Enter details for Process %d:\n", i+1);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        printf("Arrival Time: ");
        scanf("%d", &processes[i].arrival_time);
        processes[i].pid = i+1;
    }

    calculateWaitingTime(processes, n);

    return 0;
}
```

Output:

Sample Output:

```
Enter the number of processes: 3
Enter details for Process 1:
Burst Time: 5
Arrival Time: 0
Enter details for Process 2:
Burst Time: 3
Arrival Time: 2
Enter details for Process 3:
Burst Time: 8
Arrival Time: 4
Process     Burst Time
Arrival Time    Waiting Time
Turnaround Time
1               5                   0
0                   5
2               3                   2
0                   3
3               8                   4
3
```