**OPERATING SYSTEMS LAB**

**Project Report**



**Submitted By:**

Mahnoor Aftikhar 22-CS-194

Minahil Hijab 22-CS-194

Section: A

**Submitted to:**

Lab Instructor, Mr. M. Hassnain

**Department of Computer Science,**

**HITEC University, Taxila**

Contents

[Process Scheduler 1](#_Toc187482390)

[Motivation 1](#_Toc187482391)

[Introduction 1](#_Toc187482392)

[Objectives 2](#_Toc187482393)

[Methodology 2](#_Toc187482394)

[Language and Tools 2](#_Toc187482395)

[Code Structure 2](#_Toc187482396)

[Input Sample 2](#_Toc187482397)

[Sample Ouput 3](#_Toc187482398)

[GUI Implementation (optional) 5](#_Toc187482399)

[Sample Ouput 6](#_Toc187482400)

[Conclusion: 6](#_Toc187482401)

# Process Scheduler

## Motivation

Implement a process scheduler with different scheduling algorithms like FCFS, Round Robin,

and Priority Scheduling. Implement all scheduling algorithms in a single program with if else.

Use of GUI in above project is optional but project will be considered better using GUI.

## Introduction

This project demonstrates the implementation of various CPU scheduling algorithms, specifically:

1. **First-Come, First-Served (FCFS)**
2. **Shortest Job Next (SJN)**
3. **Round Robin (RR)**

The goal is to calculate and compare the waiting times for processes under these scheduling strategies. The project highlights how different algorithms handle process scheduling and their impact on performance metrics like average waiting time.

## Objectives

1. Implement CPU scheduling algorithms in C++.
2. Demonstrate how arrival time and burst time influence process waiting times.
3. Compare average waiting times for FCFS, SJN, and RR algorithms.

## Methodology

### Language and Tools

* **Language:** C++
* **Compiler:** Dev C++/Gedit Compiler

### Code Structure

The project contains the following functions:

1. **CalculateWaitingTimeFCFS**
   * Calculates waiting times for processes based on the FCFS scheduling strategy.
   * Displays waiting times and computes the average waiting time.
2. **CalculateWaitingTimeSJN**
   * Implements SJN scheduling by sorting processes based on burst time.
   * Calculates waiting times and average waiting time.
3. **CalculateWaitingTimeRoundRobin**
   * Implements Round Robin scheduling using a time quantum.
   * Tracks remaining burst times and calculates waiting times accordingly.
4. **Main Function**
   * Provides a menu-driven interface for users to select a scheduling algorithm.
   * Displays the results for each selected algorithm.

### Input Sample

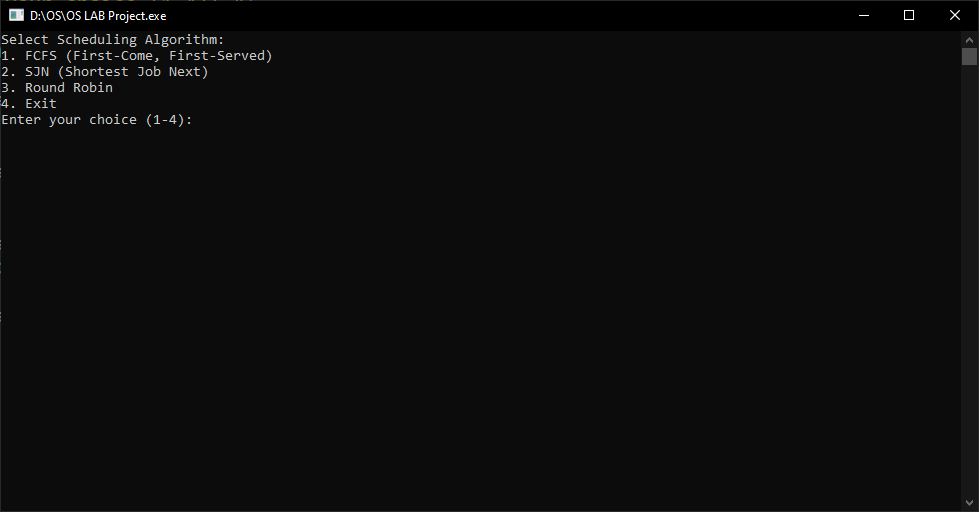
The sample data used in the program includes:

* **Number of Processes (N):** 5
* **Arrival Times (AT):** {0, 1, 2, 3, 4}
* **Burst Times (BT):** {4, 3, 1, 2, 5}

### Sample Ouput

#### Example Menu Interaction:

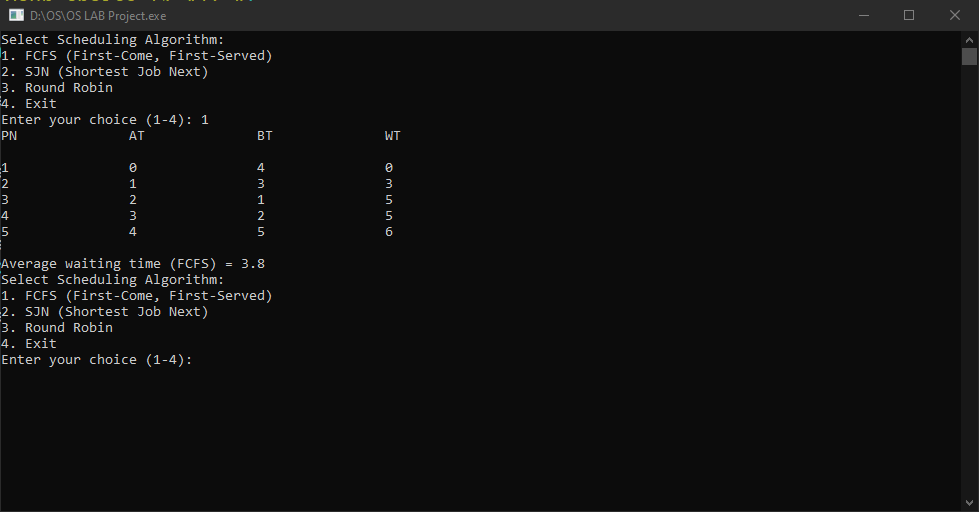
The menu was designed using a switch with case 1 for FCFS, case 2 for SJN and case 3 for RR while case 4 deals with exiting the program.



#### Output:

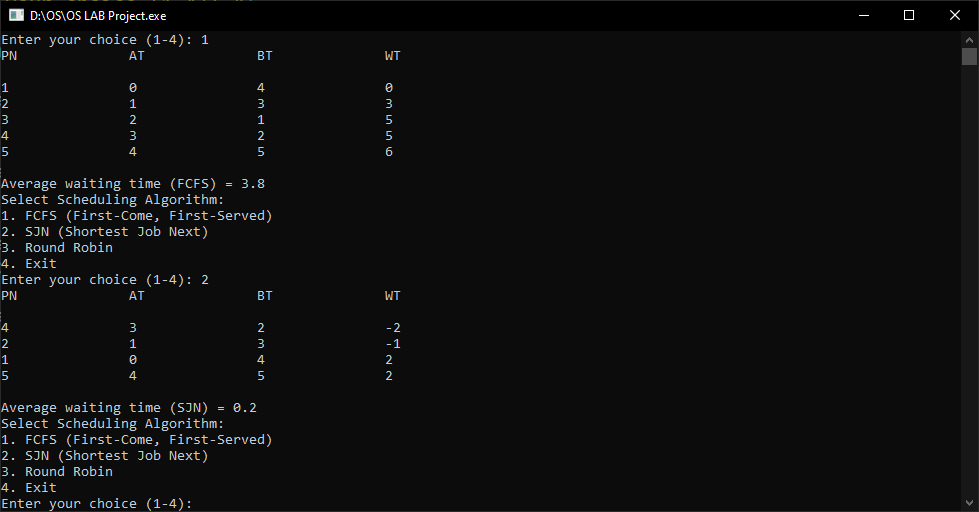
##### FCFS:

Demonstration of FCFS is as follows:



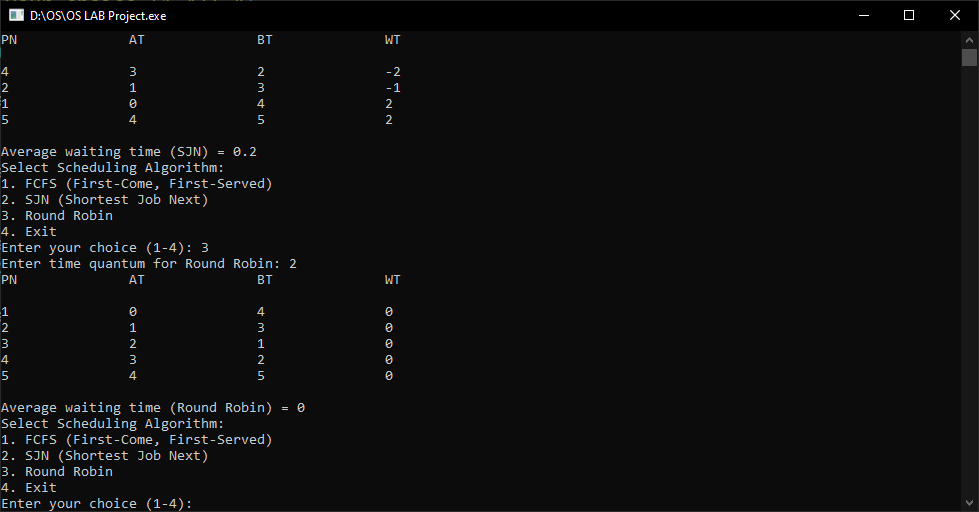
##### SJN:

Demonstration of SJN is as follows:



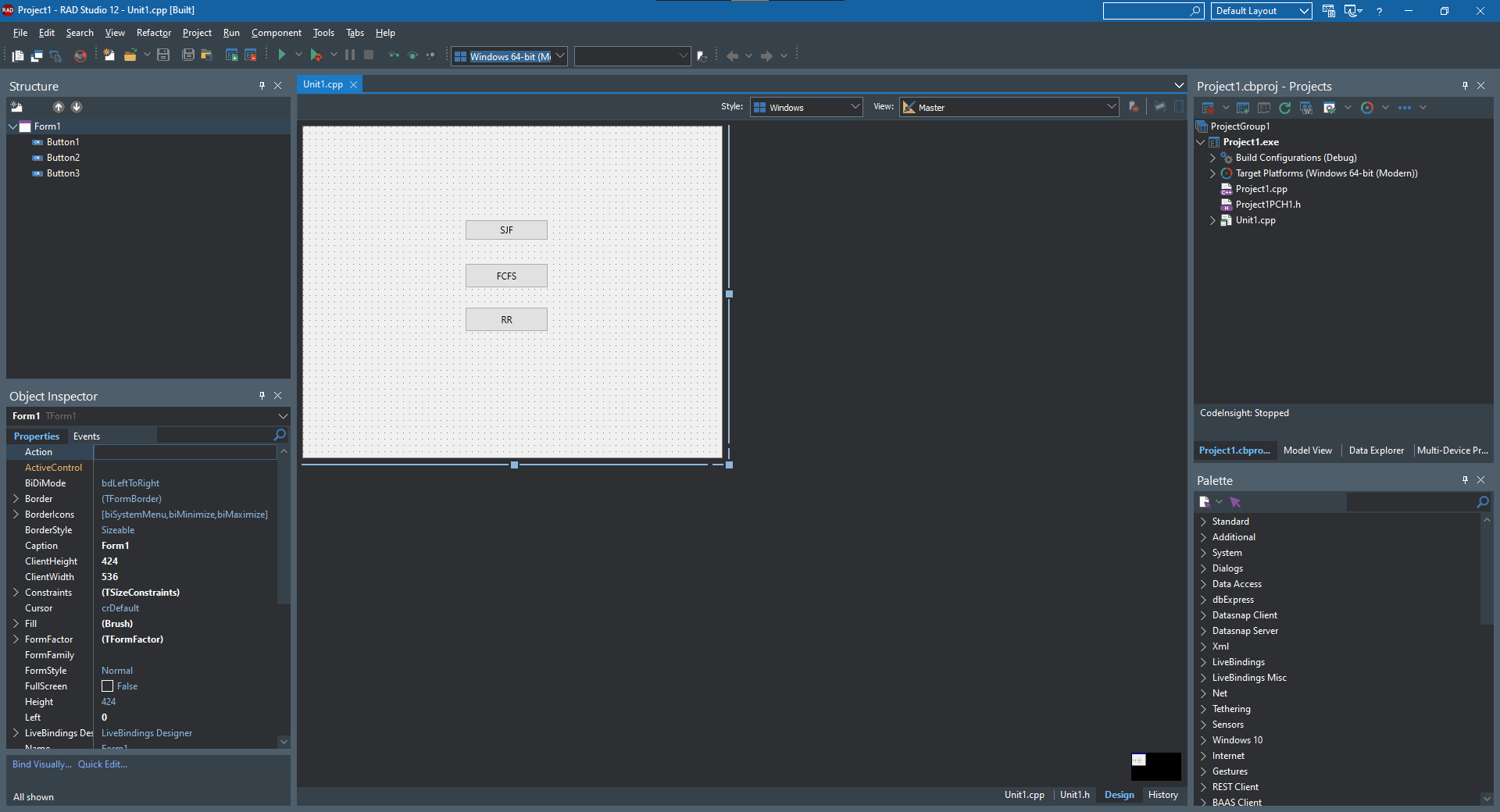
##### Round Robin:

Demonstration of RR is as follows:

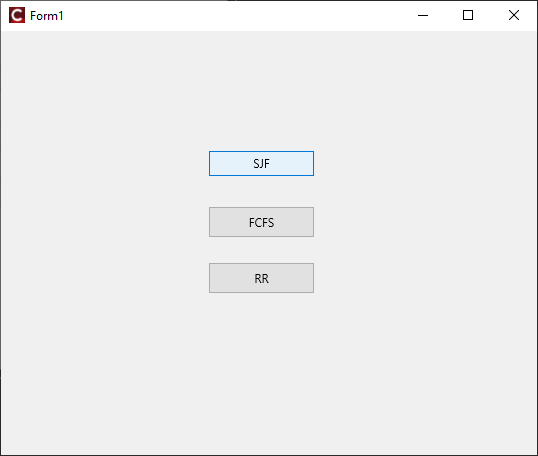


## GUI Implementation (optional)

An effort was made by our team to implement the optional GUI but it wasn’t successful due to our little to no knowledge about the GUI IDE’s. Here are the Screen snips of the incomplete GUI Implementation with RAD Studio.



### Sample Ouput



## Conclusion:

This project successfully demonstrates the implementation and analysis of CPU scheduling algorithms. Each algorithm provides unique advantages:

* **FCFS:** Simple and intuitive but may lead to the convoy effect.
* **SJN:** Optimal for minimizing waiting time but requires accurate job-length prediction.
* **Round Robin:** Ensures fairness but may increase average waiting time for small time quantums.

By understanding the strengths and limitations of each algorithm, we can make informed decisions about their applicability in real-world systems.