****

**Operating System Project**

**Multi-Level Queue Scheduling Algorithm**

**Introduction to Multi-Level Queue Scheduling:**

A common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements and so may have different scheduling needs. In addition, foreground processes may have priority (externally deﬁned) over background processes. A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm. For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by an FCFS algorithm, while the background queue is scheduled by an RR algorithm.

In addition, there must be scheduling among the queues, which is commonly implemented as ﬁxed-priority preemptive scheduling. For example, the foreground queue may have absolute priority over the background queue.

Let’s look at an example of a multilevel queue scheduling algorithm with ﬁve queues, listed below in order of priority:

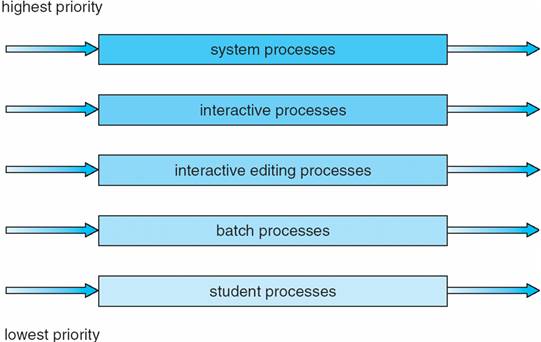
1. System processes

2. Interactive processes

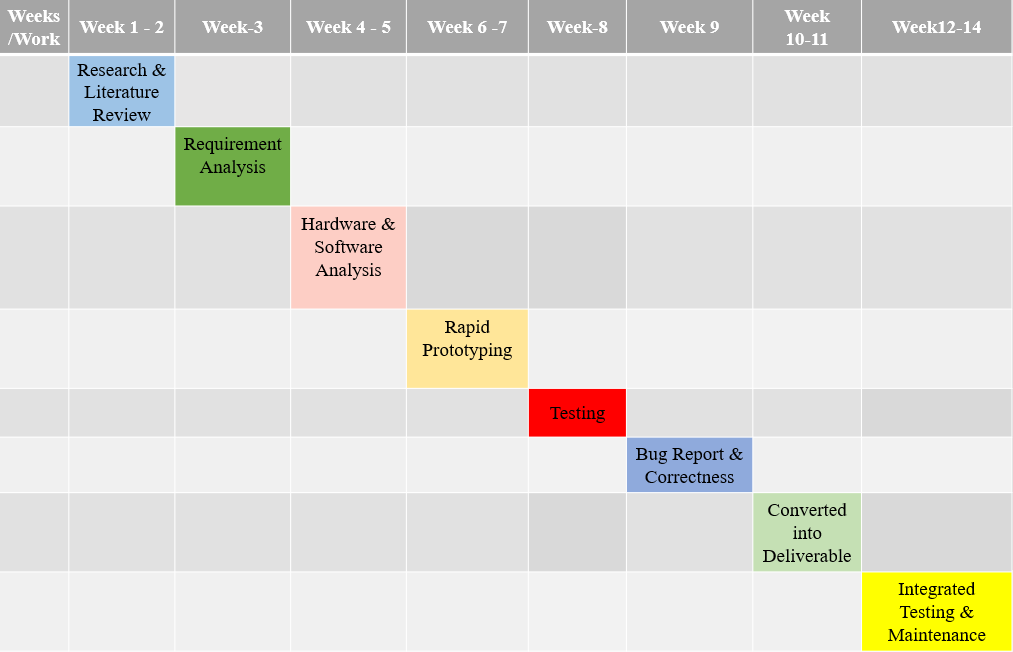
3. Interactive editing processes

4. Batch processes

5. Student processes



**Gantt Chart:**

****

**Literature Survey:**

[***Multi - Level Queue with Priority and Time Sharing for Real Time Scheduling***](http://www.ijmse.org/Volume5/Issue8/paper3.pdf) **– Iqra Sattar, Muhammad Shahid and Nida Yasir**

**Testbed and experimental setup:**

Test bed consists of total number of process and priority is assigned to each process. The burst time of each process is also given as input to the system.

Testbed capabilities:

Ability to calculate the wait time, Turnaround time and average wait and Turnaround time.

Experimental Setup:

RAM, jdk1.8, jre1.8, HTML, CSS, Bootstrap, Javascript, Web browser are the essential components

**Software requirement and specification analysis:**

Software’s Required:

* NetBeans IDE, and Visual Studio Platform
* Java 1.8, Java Server Pages (JSP)
* Apache POI framework (for reading data from excel sheet)

Problem Statement:

Implementing multilevel queue with First Come First Serve and Round Robin scheduling algorithms using Java, and designing a graphical user Interface for the following.

\*Additional Information is available in the log book.

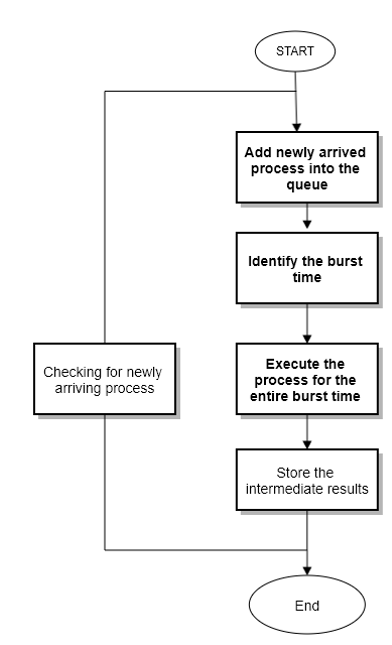
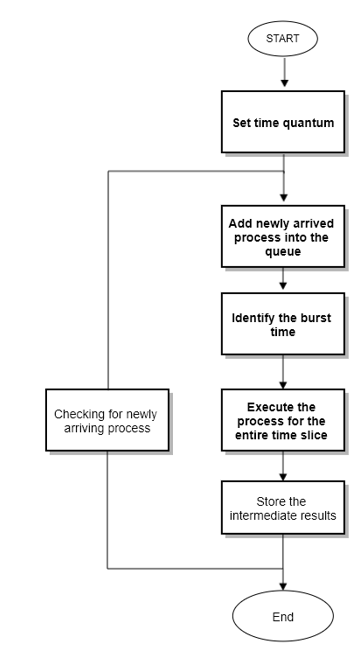
**Hardware and Software Analysis:**

* Least RAM 1GB
* Minimum clock 1.3 GHz
* NetBeans IDE, and Visual Studio Platform
* Java 1.8, Java Server Pages (JSP), Apache Tomcat webserver

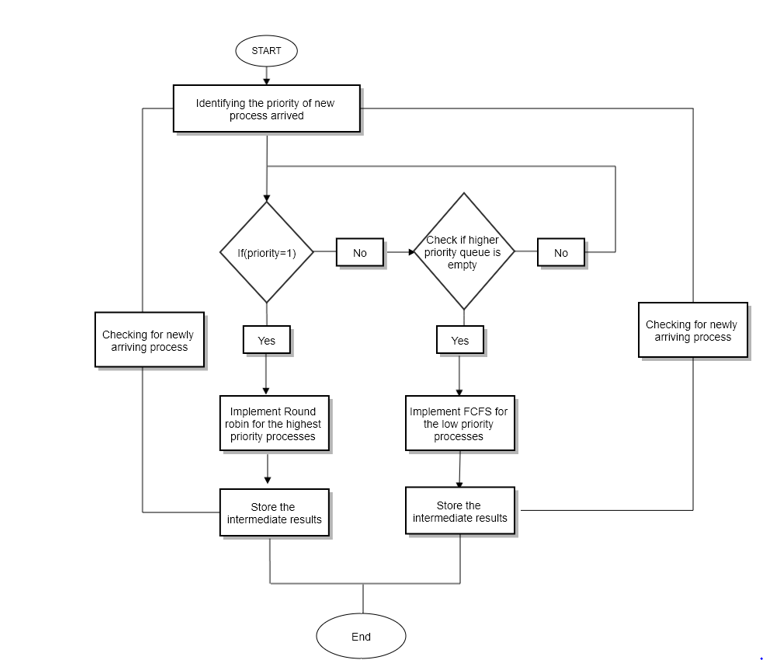
**Rapid Prototyping:**

It is also known as Waterfall model, for our entire project we have followed waterfall model, they are divided into the following categories:

* Requirements
* Design
* Implementation
* Verification
* Maintenance

**Flowcharts:** 

FCFS, Round Robin & Non-preemptive Multilevel Queue Scheduling:

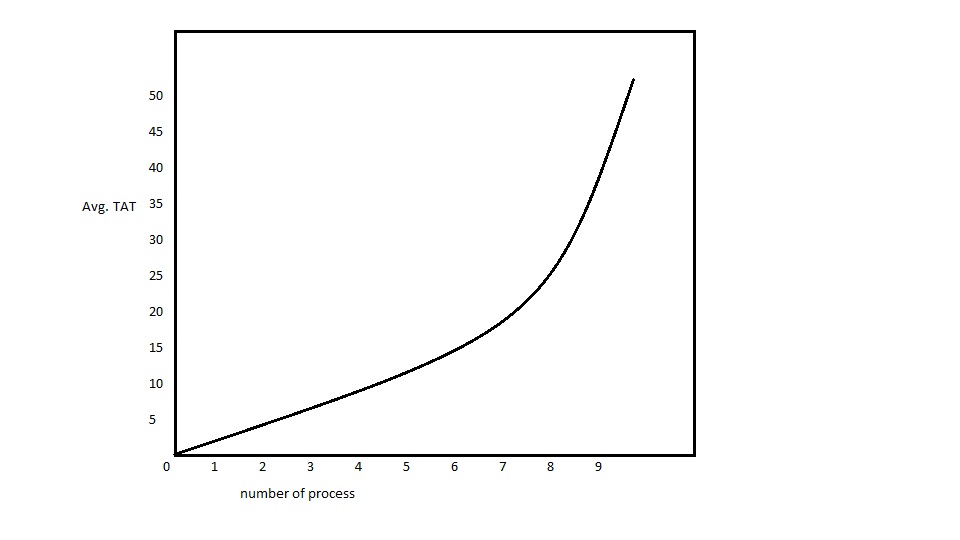
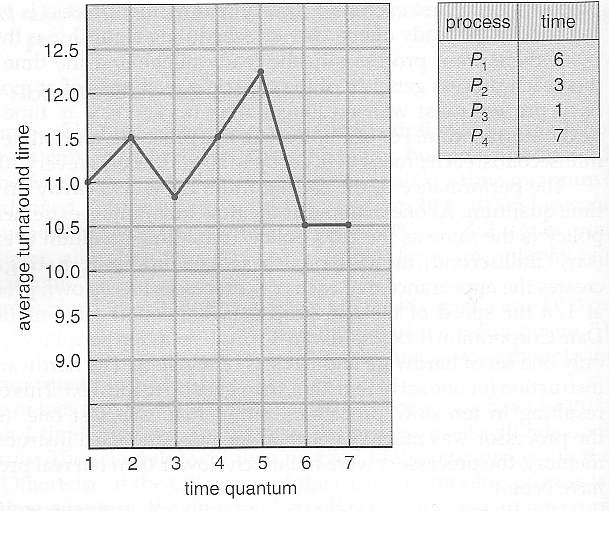


\*Additional information is available in the log book.

Complexity analysis of the code:

* For every for loop the complexity is calculated in terms of n.
* For every nested for loop the total complexity is calculated in terms of n^m.
* For every if statement, it is multiplied by 1.
* The complexity of the entire code is (4n (1)) ^2.
* Time taken for code execution: 313 milliseconds= 0.313 seconds (varies from machine to machine)

The complexity depends on priority of each process. It will be dynamic and varies for high and low priority process.



Round Robin FCFS

Proof of correctness of algorithm:

Algorithm:

* Burst time and priority of each process is read from the excel sheet and saved into two arrays of size 500.
* Based on the priority all the process is moved into 2 queues.
* The process with high priority (1) is moved to queue 1 and all the process with low priority are moved to queue 2.
* First come First serve (FCFS) algorithm is performed on queue 1. If all the process in queue are completed, then the execution will move to queue 2.
* Round Robin scheduling algorithm is used on queue 2.
* After the execution of both the queues is completed then we calculate waiting time and turnaround time of each process.

Input:

* Process [0,1,2,3,4]
* Burst time [3,4,7,2,4]
* Priority [0,1,1,0,1]
* Arrival time =0
* Time quantum = 3

Example:

* Q1-----Process [1,2,4]
* Q2-----Process [0,3]
* FCFS on Q1
* RR on Q2

Final Gant chart is

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P4 | P0 | P3 |
| 0 4 | 4 11 | 11 15 | 15 18 | 18 20 |

Average Waiting time: 9.6

Average Turnaround time: 13.6

Bug Report and Maintenance criteria:

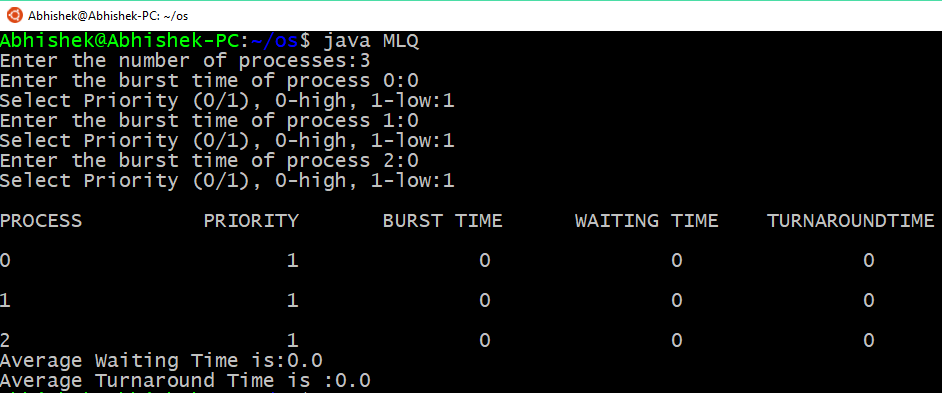
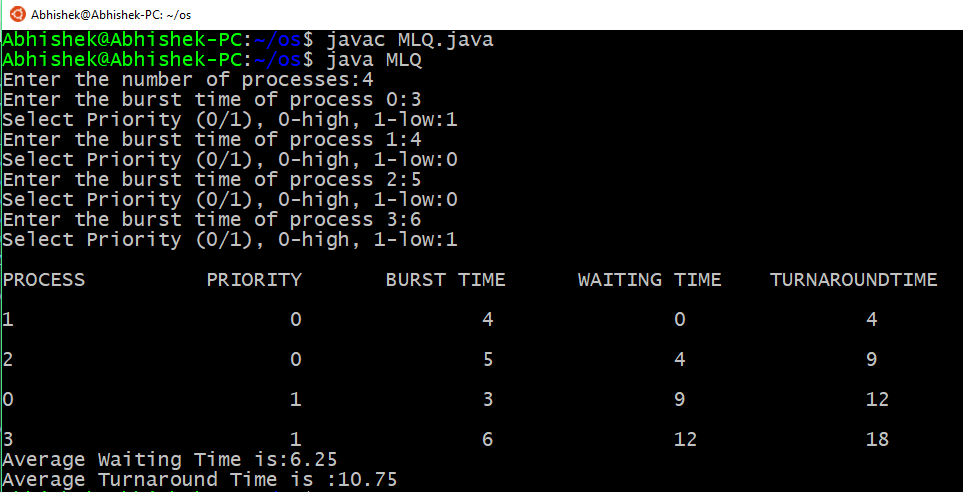
There was a bug while entering the negative numbers. The code used to accept even all the negative numbers, but it is fixed.

It was using FCFS for both low and high priority but now for high priority it is using FCFS and for low priority it is using RR Scheduling.

Pre-initialized conditions (for maintenance criteria):

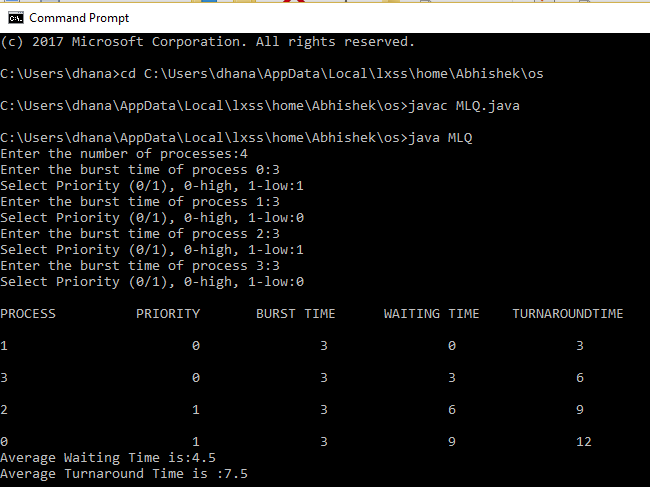
* Time quantum is directly initialized to 4 in the code. It can be changed if any other time quantum requirement popup.
* All the arrays like completion time, waiting time, burst time are initialized with 500.
* Arrival time is taken as 0 for all the process.
* Priority contains 0 and 1.1 Indicates high priority and 0 indicates low priority.

**Alpha testing:**

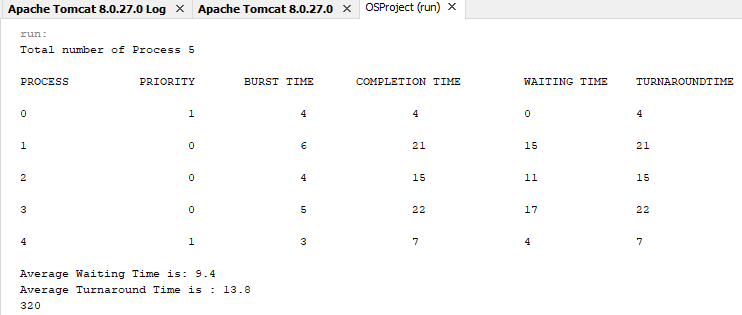
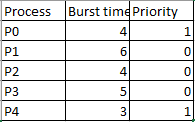


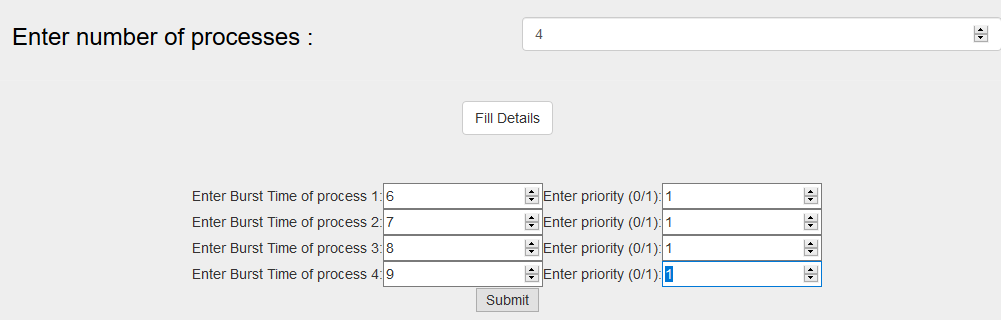
**Beta testing:**

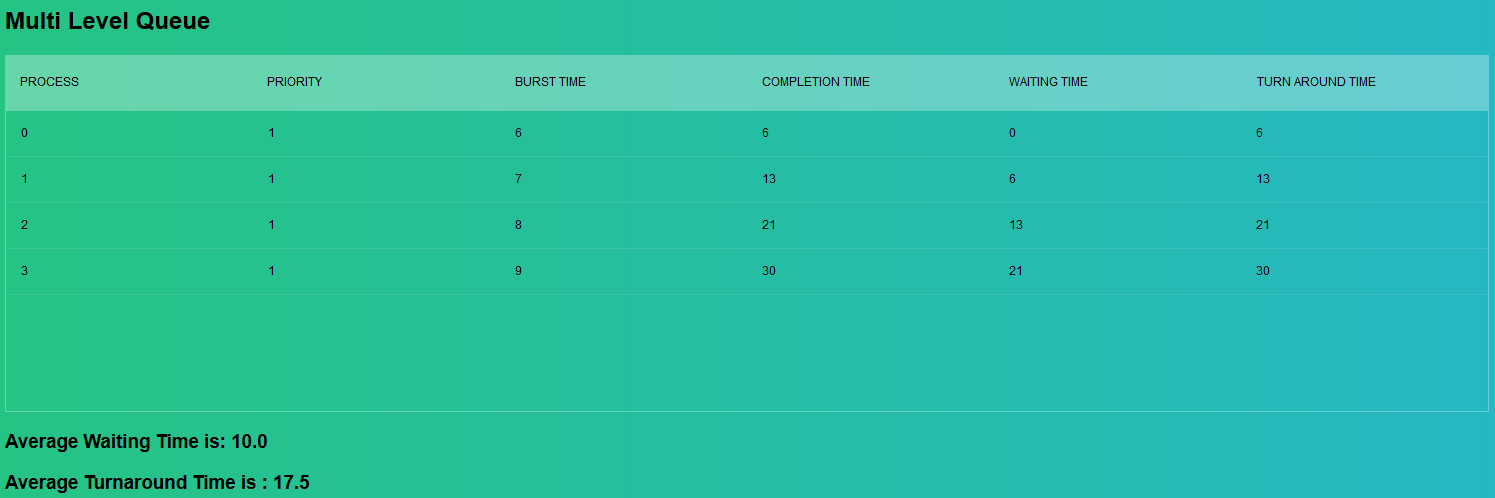
By Poojitha:



**Testing & Final Results:**

****

****

****

**Test Suite:**

[*https://mailuc-my.sharepoint.com/personal/dhanasak\_mail\_uc\_edu/\_layouts/15/guestaccess.aspx?docid=13487393d94214d3ea4e5da8f65c85a13&authkey=Aa8MUzbwZV9vyaAAf2xXF4A&e=d6e603bdc7494b89a04292d4b32cfb08*](https://mailuc-my.sharepoint.com/personal/dhanasak_mail_uc_edu/_layouts/15/guestaccess.aspx?docid=13487393d94214d3ea4e5da8f65c85a13&authkey=Aa8MUzbwZV9vyaAAf2xXF4A&e=d6e603bdc7494b89a04292d4b32cfb08)

**Log Report:**

* Changed FCFS scheduling code to Round Robin and FCFS for high and low priority.
* Changed the code such that it does not accept negative burst time and alphabets.
* Reduced the complexity by using two separate queues for both low priority and high priority process.

**Future Work:**

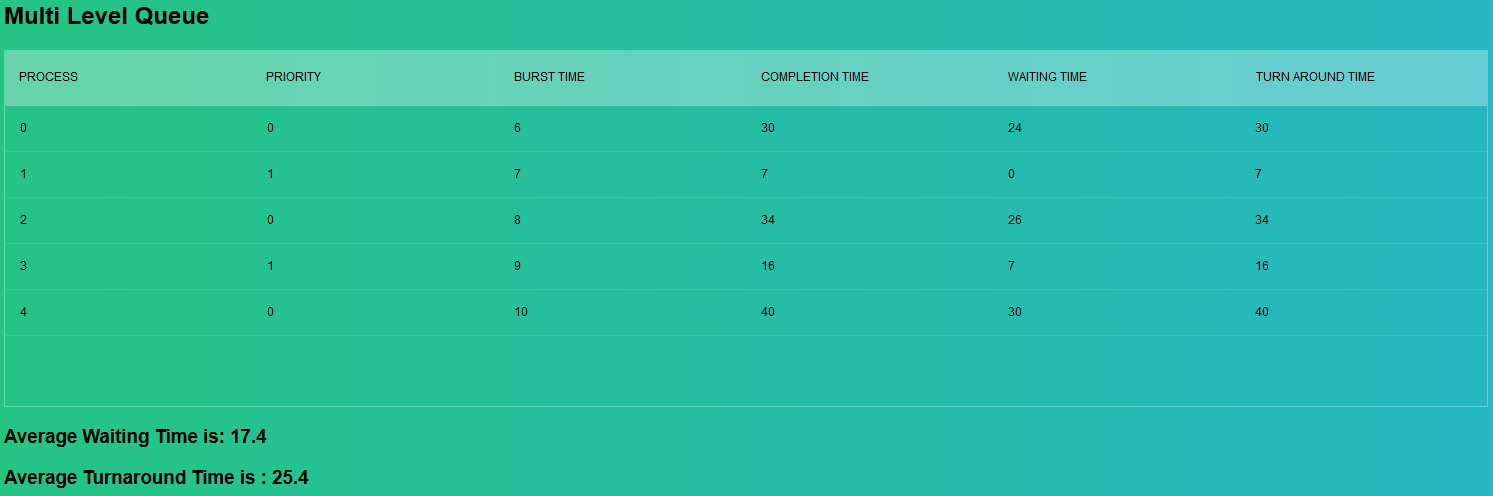
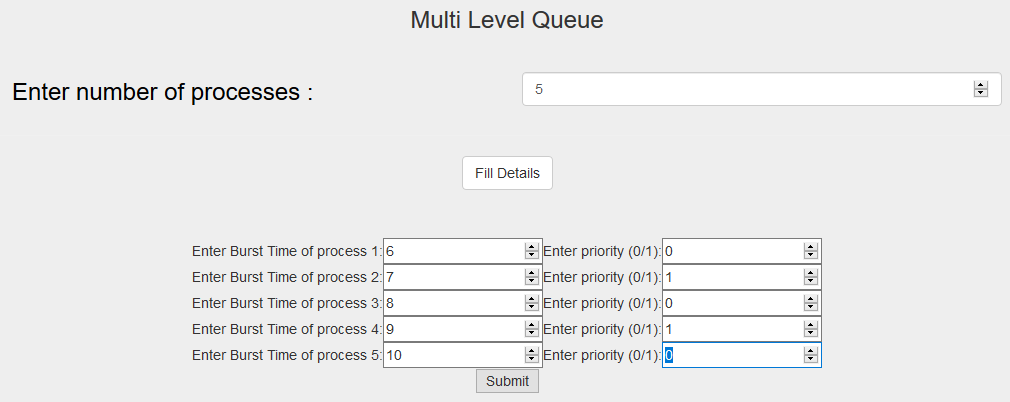
This project can be further improved by including feedback queue. This give rise to Multi level feedback queue.

**Walk through and tested by other teams:**

**Team 1: (Anvesh and team)**

The project is based on multilevel queue scheduling algorithm using FCFS for high priority process and RR for low priority process. We checked the project different test cases i.e., negative numbers, alphanumeric etc. The project is working fine for multiple test cases and give average turn around time and average waiting time.

**Test Suite:**



**Team 2: (Srikanth and team)**

We have tested this project on multilevel queue by providing the inputs in GU interface. Then inputs were then moved to 2 different queues and based on the priority Round Robin and FCFS algorithms are used for execution of these process. Finally, average turnaround time and average waiting time were calculated. We have checked the outputs and they are right.

**Test Suite:**

