

Software Requirement Specification

**FIRST COME FIRST SERVE (FCFS) DEMONSTRATION**

**Developed By:**

* CS-5 SP-22 Batch

**Supervised By:**

* Sir Ajeet Kumar

**Lead By:**

* Muhammad Hussain

**Allocation of Responsibilities:**

|  |  |
| --- | --- |
| Work | Assigned To |
| Overall Ui | Hussain, Ibtesam, Preetum |
| Main Logic | Rushan |
| Extra Logics (Timers etc.) | Khadija |
| Dom Manipulation | Dur e Shawar |
| Testing | Amir and Talha Ansari |
| Poster | Daim and Talha Siddiqui |
| Documentation (SRS) | Kinza and Janita |

**Management Approach:**

We started by breaking down the project into smaller tasks according to a schedule. We began by creating a basic outline, or wireframe, for our web application. While that was being polished and made to look better, we also began working on the part of the website that handles the information behind the scenes, like gathering data from forms. At the same time, we added a simple timer feature to the website to make it more functional

The most challenging part of the project was building the core logic, which took up a significant amount of our time. This part involved implementing the main functionalities, including the First Come First Serve (FCFS) algorithm, which was essential for our project.

Once all the coding work was finished, we focused on making sure that all the different parts of the website worked well together. This involved testing everything thoroughly to identify and fix any issues or bugs that we encountered.

After completing the web application, we shifted our attention to designing a poster to showcase our project. Additionally, we continued testing the application to ensure its reliability and performance, addressing any issues we found along the way.

Throughout the entire process, we made sure to document our progress and decisions made, which helped us keep track of our work and ensure that everyone involved was on the same page.

**Time Allocation:**

The project was completed over a four-day period. Approximately 1.5 days were dedicated to designing the user interface (UI), ensuring it was both visually appealing and user-friendly. Another 1.5 days were allocated to developing the core logic of the operating system, with a focus on implementing the First Come First Serve (FCFS) algorithm effectively. The remaining day was devoted to debugging and testing, ensuring the system's stability and functionality.

**Development Tools Utilized:**

The project, being web-based, utilized HTML and CSS for designing the interface, with the additional support of the Bootstrap CSS framework to ensure responsiveness across various devices. For implementing the logic, JavaScript was employed to develop the core functionalities, including the First Come First Serve (FCFS) algorithm. This combination of technologies allowed for the creation of an interactive and visually appealing web application that effectively demonstrates the FCFS algorithm.

**User Interface Design Approach:**

The user interface (UI) was designed with the intention of visually representing how the CPU processes tasks. To achieve this, we created three queue-like structures within the UI: one for holding incoming processes, another for processes waiting for the processor to become available, and a third for storing data of executed processes.

On the opposite side of the interface, we included a form where users can input details such as the arrival time and burst time of processes. Additionally, a panel was incorporated to display relevant information about the processes, such as average turnaround time (TAT), average waiting time (WT), and a start button to initiate the processing.

This UI layout allows users to interactively input process details, visualize the process flow through the queue-like structures, and monitor key metrics, enhancing understanding of how CPU processes tasks in a First Come First Serve (FCFS) algorithm scenario.

**FCFS Scheduling Algorithm:**

The First Come First Serve (FCFS) algorithm, a simple CPU scheduling algorithm, operates as follows:

**1.Arrival:** When a process arrives, it is added to the end of the ready queue, forming a First-In-First-Out (FIFO) queue.

**2.Execution:** The CPU is allocated to the first process in the queue. This process continues execution until it completes its CPU burst

**3.Completion:** Upon completion of the current process's CPU burst, it is removed from the queue, making way for the next process in line to be executed.

**Key characteristics of FCFS:**

* **Non-preemptive:** Once a process starts executing, it continues until completion.
* **Simple to implement:** FCFS scheduling requires minimal overhead and is easy to implement.

**Implementation of logic:**

The implementation of the FCFS (First Come First Serve) scheduling algorithm involves setting up a queue data structure to manage the order of processes. Here's a basic outline of how the logic can be implemented:

* **Initialization:** Initialize an empty queue to hold processes.
* **Process Arrival:** When a process arrives, it is added to the end of the queue.
* **CPU Allocation:** When the CPU becomes available, the process at the front of the queue (the first one to arrive) is selected for execution.
* **Execution:** The selected process is executed until it completes its CPU burst.
* **Process Completion:** After a process finishes its CPU burst, it is removed from the queue.
* **Repeat:** Steps 3 to 5 are repeated until all processes have completed execution.

**Testing:**

During the testing phase, we encountered several bugs which we addressed systematically. One notable issue was that the form continued to accept input even while the processor was running, causing unexpected behavior. We resolved this issue by implementing a mechanism to disable the form during processing.

Another issue we identified was that the processor timer did not stop after the completion of all processes. This inconsistency was rectified by implementing a check to halt the timer upon completion of all tasks.

Additionally, we encountered minor calculation errors that required attention. These discrepancies were meticulously addressed to ensure accurate computation of relevant metrics.

Overall, through diligent testing and troubleshooting, we successfully identified and resolved various bugs, enhancing the reliability and functionality of the web application.

**Conclusion and learning:**

Developing this project was a remarkable journey that provided us with valuable insights into CPU processes and system development. Beyond the technical aspects, it taught us crucial lessons in teamwork, stress management, and meeting deadlines. Despite the challenges we faced, we were able to overcome them through collaboration and perseverance, ultimately delivering a successful project. Overall, it was an enriching experience that contributed significantly to our growth and development.

**Future goals:**

* Enhance the project's educational value by integrating additional CPU scheduling algorithms and refining the user interface with interactive elements. This will provide users with a more comprehensive understanding of scheduling strategies and improve engagement.
* Optimize the project for educational deployment by creating accompanying resources such as documentation, tutorials, and exercises. These resources will assist educators in effectively incorporating the project into academic settings, maximizing its impact on student learning.
* Introduce a feature to visualize real-time backend processes of the host system where the project is deployed. This addition will offer users valuable insights into system resource utilization and management, enriching their understanding of operating system principles.

**THE END**