High Level Design & Low Level Design

The purpose of this document is to provide with a template for documenting both HLD & LLD.

**Document Control :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 28/11/2022 | 0.1 | Group-1 | Started understanding SRS | | | |  | |
| 2/12/2022 | 0.5 | Group-1 | Design phase | | | |  | |
| 7/12/2022 | 1 | Group-1 | Implemented coding | | | |  | |

1. INTRODUCTION……………………………………………………………………………………5

1.1. INTENDED AUDIENCE**…………………………………………………………………………5**

1.2. ACRONYMS/ABBREVIATIONS**………………………………………………………………..5**

1.3. PROJECT PURPOSE**.........................................................................................................................5**

1.4. KEY PROJECT OBJECTIVES**……………………………………………………………………5**

1.5. PROJECT SCOPE AND LIMITATION**…………………………………………………………..5**

1.5.1. In Scope**……………………………………………………………………………………...6**

1.5.2. Out of Scope**…………………………………………………………………………………6**

1.6. FUNCTIONAL OVERVIEW**………………………………………………………………………6**

1.7. ASSUMPTIONS, DEPENDENCIES AND CONSTRAINTS**……………………………………...6**

1.8. RISKS**……………………………………………………………………………………………….6**

**2. DESIGN OVERVIEW………………………………………………………………………………...7**

2.1. DESIGN OBJECTIVES**......................................................................................................................7**

2.1.1. Recommended Architecture**………………………………………………………………..7**

2.2. ARCHITECTURAL STRATEGIES**………………………………………………………………7**

2.2.1. Design Alternative**....................................................................................................................7**

2.2.2. Reuse of Existing Common Services/Utilities**………………………………………………7**

2.2.3. Creation of New Common Services/Utilities **…………………………………………......8**

2.2.4. User Interface Paradigms**…………………………………………………………………….8**

2.2.5. System Interface Paradigms**………………………………………………………………….8**

2.2.6. Error Detection/Exceptional Handling**……………………………………………………….8**

2.2.7. Memory Management**………………………………………………………………………...8**

2.2.8. Performance**…………………………………………………………………………………..8**

2.2.9. Security**……………………………………………………………………………………….8**

2.2.10. Concurrency and Synchronization**…………………………………………………………..8**

2.2.11. House Keeping and Maintenance**............................................................................................9**

**3. SYSTEM ARCHITECTURE…………………………………………………………………………..9**

3.1. SYSTEM ARCHITECTURE DIAGRAM.(NOT NECESSARY)**…………………………………..9**

3.2. SYSTEM USE-CASE **……………………………………………………………………………….9**

3.3. SUBSYSTEM ARCHITECTURE**………………………………………………………………….10**

3.4. SYSTEM INTERFACES**…………………………………………………………………………...12**

3.4.1. Internal Interfaces**…………………………………………………………………………….12**

3.4.2. External Interfaces**……………………………………………………………………………13**

**4. DETAILED SYSTEM DESIGN………………………………………………………………………14**

4.1. KEY ENTITIES**…………………………………………………………………………………….14**

4.2. DETAILED-LEVEL DATABASE DESIGN**………………………………………………………14**

4.2.1. Data Mapping Information**…………………………………………………………………...14**

4.2.2. Data Conversion**……………………………………………………………………………...14**

4.3. ARCHIVAL AND RETENTION REQUIREMENTS**……………………………………………...14**

4.4. DISASTER AND FAILURE RECOVERY**…………………………………………………………14**

4.5. BUSINESS PROCESS WORKFLOW**………………………………………………………………14**

4.6. BUSINESS PROCESS MODELLING AND MANAGEMENT (AS APPLICABLE)**......................15**

4.7. BUSINESS LOGIC**…………………………………………………………………………………15**

4.8. VARIABLES**………………………………………………………………………………………..16**

4.9. ACTIVITY/CLASS DIAGRAMS (AS APPLICABLE)**……………………………………………16**

4.10. DATA MIGRATION**………………………………………………………………………………16**

4.10.1. Architectural Representation**……………………………………………………………….17**

4.10.2. Architectural Goals and Constraints**………………………………………………………..17**

4.10.3. Architecturally Significant Design Packages**………………………………………………17**

4.10.4. Data Model **……………………………………………………………………………….17**

4.10.5. Deployment View**…………………………………………………………………………17**

**5. ENVIRONMENT DESCRIPTION………………………………………………………………….17**

5.1.TIME ZONE SUPPORT**…………………………………………………………………………17**

5.2. LANGUAGE SUPPORT**………………………………………………………………………...17**

5.3. USER DESKTOP REQUIREMENTS**…………………………………………………………...17**

5.4. SERVER-SIDE REQUIREMENTS**……………………………………………………………...17**

5.4.1 Deployment Considerations**………………………………………………………………...18**

5.4.2. Application Server Disk Space **…………………………………………………………….18**

5.4.3. Database Server Disk Space**………………………………………………………………..18**

5.4.4. Integration Requirements**…………………………………………………………………...18**

5.4.5. Jobs**………………………………………………………………………………………….18**

5.4.6. Network**……………………………………………………………………………………...18**

5.4.7. Others**........................................................................................................................................18**

5.5. CONFIGURATION**………………………………………………………………………………...18**

5.5.1 Operating System**……………………………………………………………………………..18**

5.5.2. Database**………………………………………………………………………………………18**

5.5.3. Network**……………………………………………………………………………………….18**

5.5.4. Desktop**………………………………………………………………………………………..19**

**6. REFERENCES………………………………………………………………………………………….19**

**7. APPENDIX………………………………………………………………………………………………19**

**1.Introduction**

This programming project is to simulate few CPU scheduling algorithms discussed in the class. we will write a C++ to implement a simulator with different scheduling algorithms. The selected scheduling algorithms to implement in this project are First Come First Serve (FCFS), Round Robin (RR) with time quantum, Shortest job First (SJF), Largest job first (LJF). The scheduler selects a task to run from a ready queue based on the scheduling algorithm. Since the project intends to simulate a CPU scheduler, it does not require any actual process creation or execution. When a task is scheduled, the scheduler will simply print out what task is selected to run at a time.

**1.1 Intended Audience**

This document explains our team architecture, our team’s initial understanding of the user needs. It will assist our team in understanding the system specifications and analyze the critical aspects of our project. This document will briefly discuss the stakeholders involved in the development, documents will show how our team was divided to handle the multiple stakeholders, the sources of the requirements, provide an informal preliminary requirements description, and address any issues encountered while transforming the requirements.

**1.2 Acronyms/Abbreviations**

|  |  |
| --- | --- |
| UT | Unit test |
| IT | Integrated test |

**1.3 Project Purpose**

Schedule simulation is designed to help in the planning stages of a project so that you can get a visual representation of the required tasks and prevent spatial conflicts.

* 1. **Key Project Objectives**

Client select the algorithm & then send to server. When server gives that to scheduler, scheduler will be running according to that algorithm. Scheduler gives that according output to server .Server giving that to client & client will display.

* 1. **Project Scope and Limitation**

As per the limitation of information scheduler is we can’t handle workflow dependency directly. It does not consider the priority or burst time of the processes.

It suffers from convoy effect i.e. processes with higher burst time arrived before  the processes with smaller burst time.

* + 1. **In Scope**

The scope of the Simulation of Scheduler is to show the process scheduling for different types of algorithms and displaying the running process and processes in queue to user.

We are having user interface to interface the user data to server and from server to scheduler.

* + 1. **Out of scope**

Required to involve techniques such as socket programming in TCP, process management to make the project done completely.

* 1. **Functional Overview**

The application Simulation of Scheduler should display a main menu to select options such as algorithm choice menu, help etc. It should display further sub menu options based on selected menu options. It should validate the menu options at each level. If any incorrect option or entry by the user should display an error.

For FCFS algorithm selection, the given process order is treated as the order of execution. For SJF/LJF algorithm selection, the user should provide total execution time in seconds for each process along with process path in format given below. For priority algorithm selection, the user should provide the priority for each process along with the process path in the format given below. For RR algorithm selection, the user should provide time quantum duration in seconds for each process along with process path

Displaying the current waiting processes from the user-entered queue. Displaying the ready to run processes from the ready-to-run queue. Start the scheduler if the option is chosen. Displaying the current running processes. The application should maintain a data structure to store each algorithm related data which will be taken from the user. The application should maintain a ready to run queue containing all the given processes, sorted as per the selected algorithm. The application should maintain a wait queue containing all the given processes, waiting to be scheduled.

* 1. **Assumptions, Dependencies & Constraints**

Assuming if two types of algorithms to run simultaneously, It needs server to decide which algorithm to run first and also efficiently for two processes. Implementation of this assumption will be depending on the accuracy to switch between algorithms.

* 1. **Risks**

All assumptions, functional overview and design parameters are documented without evaluation  which are to be implemented without missing. To eliminate its struggles with developing an optimized schedule and to properly manage resources, the use of deterministic scheduling was explored but deterministic scheduling was unable to create accurate schedules.

1. **Design Overview**

In this section , we are discussing about the design objectives that are considered, the architecture used to implement the design flow.

**2.1 Design Objectives**

Design objectives include different algorithms like FCFS, RR, SJF, Priority these are corresponding to the user or client.

In the FCFS scheduling algorithm, the job that arrived first in the ready queue is allocated to the CPU and then the job that came second, and so on.

RR is a job-scheduling algorithm that is considered to be very fair, as it uses time slices that are assigned to each process in the queue or line.

SJF is a scheduling algorithm, that is used to schedule processes in an operating system. It is a very important topic in Scheduling when compared to round-robin and FCFS Scheduling.

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority.

**2.1.1 Recommended Architecture**

UML Architecture UML stands for Unified Modeling Language. Any real-world system is used by different users. The users can be developers, testers, business people, analysts, and many more. Hence, before designing a system, the architecture is made with different perspectives in mind. The most important part is to visualize the system from the perspective of different viewers. The better we understand the better we can build the system.

**2.2 Architectural Strategies**

No architectural strategies have been employed.

* + 1. **Design Alternative**

Designed sequence diagram and use case diagram as design paradigm but as an alternative selected class diagram to visualize more data that've used in application.

* + 1. **Reuse of Existing Common Services/Utilities**

Design and development is done from scratch using existing sources star UML for design and VI for development.

* + 1. **Creation of New Common Services/Utilities**

Use the Scheduler to schedule processes and monitor processing. Used existing resources to develop the application specific services.

* + 1. **User Interface Paradigms**
* Client
* Server
  + 1. **System Interface Paradigms**

The system Interface paradigm implemented is a Menu driven interface between user and client. It displays a set of menu, which is to be chosen by user and implementing application accordingly.

* + 1. **Error Detection / Exceptional Handling**

Error detection and exception handling is implemented in each case where there is a chance of occurring exceptions. Respective try catch blocks are implemented which throws exception.

* + 1. **Memory Management**

Valgrind application used to check for memory leaks if any and rectified if any memory leaks occur. Valgrind is a programming tool for memory debugging, memory leak detection, and profiling. Valgrind was originally designed to be a free memory debugging tool for Linux.

* + 1. **Performance**

System performance needs to be accurate while scheduling for each algorithm. One way of estimating the performance of a scheduling algorithm is to measure the number of processes completed per unit time. Throughput is dependent on workload shorter jobs yield higher throughput, whereas longer jobs yield lower throughput.

* + 1. **Security**

The source code is available only in read only mode for others who are accessing file and final executable file is only available for further modifications. Restrict local services to the services required for operation. Implement protection for buffer overflow.

**2.2.10 Concurrency and Synchronization**

Application is designed to work concurrently without interfering with the function of other part of code and also work simultaneously.

**2.2.11 Housekeeping and Maintenance**

Not applicable for our application.

1. **System Architecture**

UML design is the shortest form of “Unified Modeling Language”. The purpose of this modeling language is to visualize the design of the system. There are total 14 types of UML diagram. They are:

• Use case diagram

• Sequence diagram

• Class diagram

* 1. **System Architecture Diagram.**

We designed Use case diagram, sequence diagram, class diagram & level diagrams.

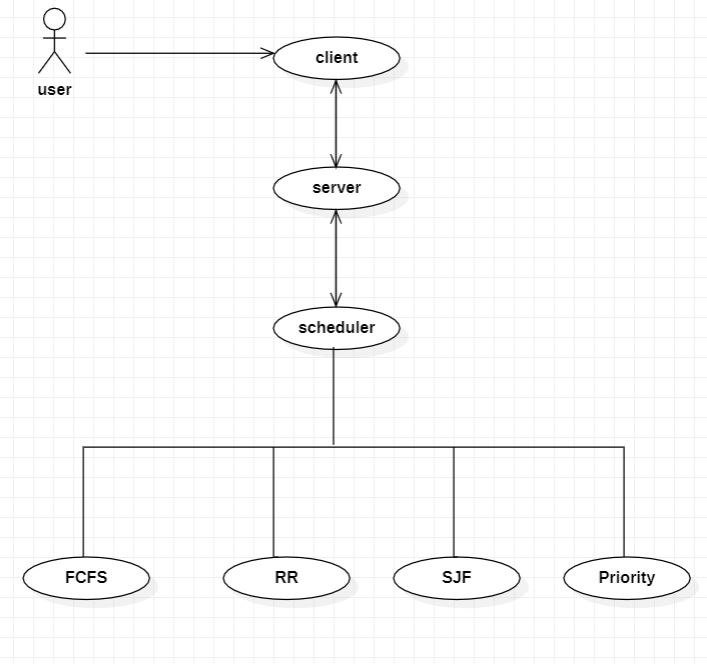
In Use case diagram we have taken actor as a client and all the others such as server, scheduler, FCFS, RR, SJF & Priority are taken as use case. Here we are considering four algorithms FCFS, RR, SJF & PRIORITY.

In sequence diagram we have user, client, server & scheduler acts as a lifelines. Where as user will select one algorithm among four algorithms those are FCFS, RR, SJF, PRIORITY

* 1. **System Use-Cases**

It is also called behavioral UML diagram. It gives a graphic over-view of the actors involved in a system directly. It shows how different functions needed by the actors how they are interacted.

The Simulation of a scheduler Use Case Diagram is one of the methods used to show the processes involved in scheduling. It depicts the system’s supposed structure and behaviour. Additionally, the diagram consists of processes (use cases) and users, or “actors.” It uses defined symbols to describe the overall flow of the system.

****

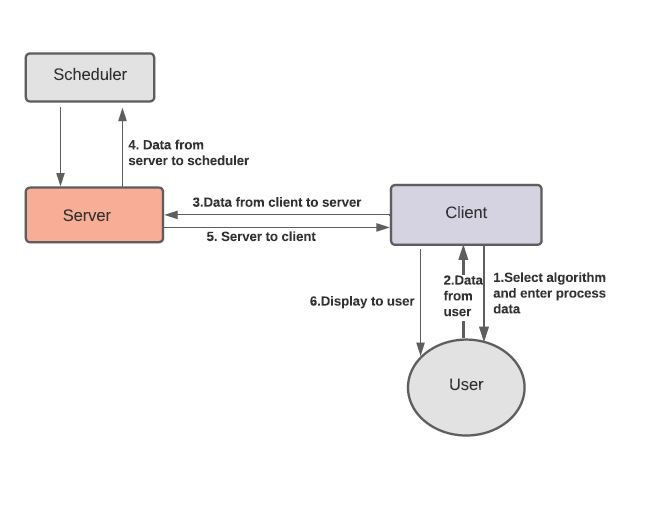
Here we have taken actor as a client and all the others such as server, scheduler, FCFS, RR, SJF & Priority are taken as use case. Here we are considering four algorithms FCFS, RR, SJF & PRIORITY. In usecase diagram client select the algorithm & then send to server. When server gives that to scheduler, scheduler will be running according to that algorithm. Scheduler gives that according output to server .Server giving that to client & client will display.

* 1. **Subsystem Architecture**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Level 0 Diagram:

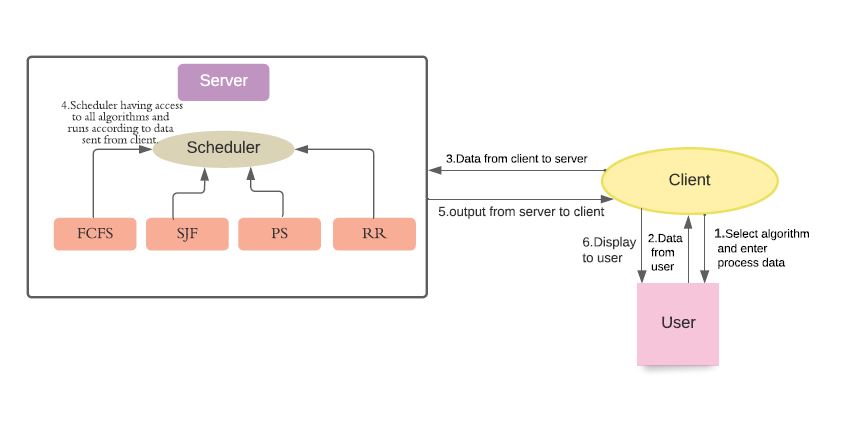
It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities.

****

We have designed DFD level diagrams for our project where we have client who will select the algorithm and enter the process data where as user will send the data and client will process these data to the server and similarly server will process the data to scheduler and at last client will display the output to the user.

**Level 1 Diagram:**

In level 1 DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of level 0 DFD into sub-processes.



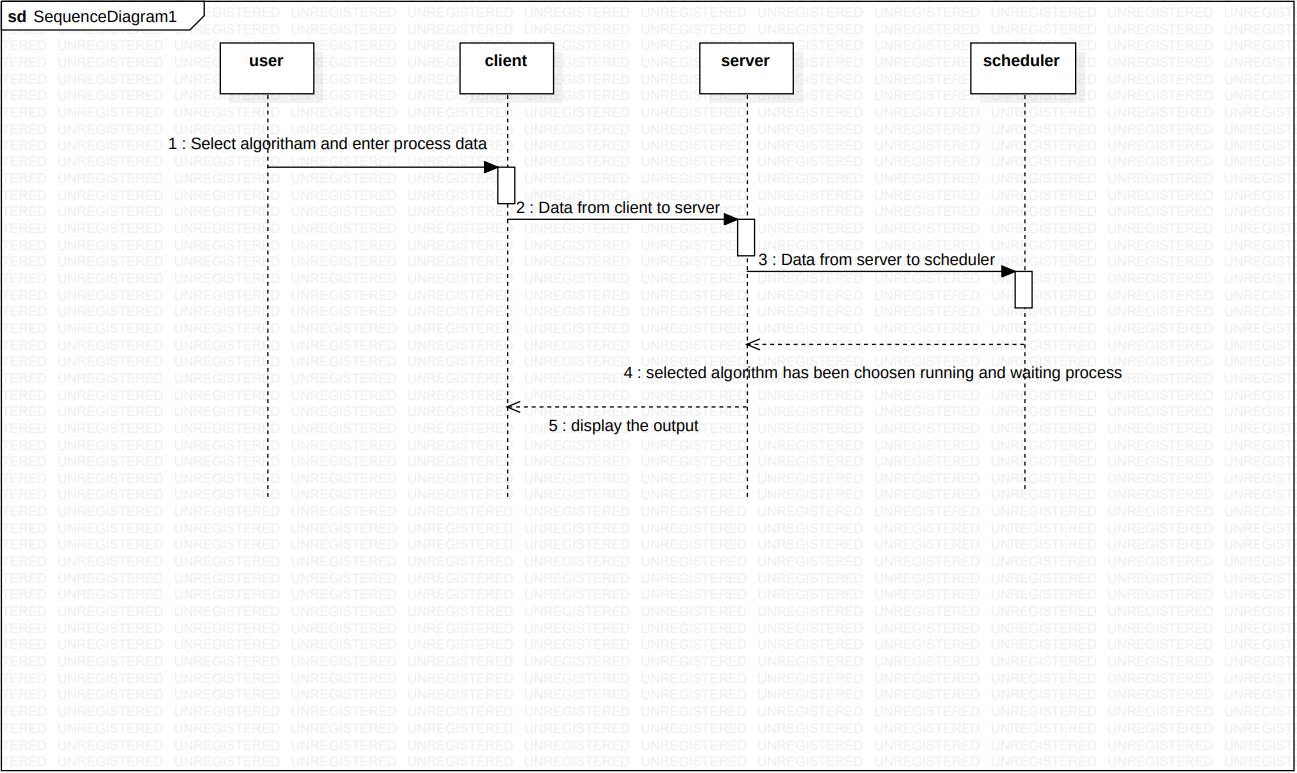
We have designed DFD level diagrams for our project where we have client who will select the algorithm and enter the process data where as user will send the data and client will process these data to the server, inside server there will be a scheduler having access to all the algorithms (FCFS, RR, SJF, Priority) and runs according to data sent from client. And it gives the output and that output is sent from server to client and it displays it to the user.

* 1. **System Interfaces**

The key system interfaces implemented are client-server using socket IPC. The sequence of interfacing is , client to server and from to server to scheduler and scheduled output from scheduler back to server.

* + 1. **Internal Interfaces**

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. Below is the “SEQUENCE DIAGRAM” of our new proposed system.

**

A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.

In sequence diagram we have user, client, server & scheduler acts as a lifelines. Where as user will select one algorithm among four algorithms those are FCFS, RR

SJF, PRIORITY and enter the process data and this data is been send to server from client. The data which is sent from client to server is now passed to scheduler according to it the scheduler will select the algorithm and process the data.

* + 1. **External Interfaces**

Hardware Interfaces

• GHz processor, 2 GB RAM or more (system memory)

• 20 GB of hard-drive space or more

• VGA capable of 1024×768 screen resolution

• Necessary computer peripherals such as keyboards etc.

• Internet Connectivity (Wired/ Wireless)

Software Interfaces

• Windows/ Linux Based OS/ Mac OS/ Any OS capable of running c ++

• Database

• Server

1. **Detailed System Design**

In this section we are going to represent the key sections that have been implemented while implementing low level design. Low level design includes the parameters, attributes and particulars that are used while implementing application.

* 1. **Key Entities**

The key entities are Socket programming which involves communication between server and client. Another important entity is scheduler which is to be implemented in server side helps to implement corresponding algorithm as selected by user.

* 1. **Detailed-Level Database Design**

Not applicable for our Application.

**4.2.1 Data Mapping Information**

Not applicable for our application.

* + 1. **Data Conversion**

Not applicable for our application.

* + 1. **Archival and retention requirements**

Not applicable for our application.

* 1. **Disaster and Failure Recovery**

It needs server to decide which algorithm to run first and also efficiently for two processes. Implementation of this assumption will be depending on the accuracy to switch between algorithms. This will assure that application will not affected by complexity as it is switching between every algorithm.

* 1. **Business Process workflow**

The workflow of our application mainly involves socket IPC in between client and server. Here client acts as user-interface and helps user to decide whether to start or stop scheduler and also decide the scheduling algorithms to be implemented as per user interest.

The data which is given by user includes the type of algorithm and arguments of processes to be scheduled. The data given by user is stored in Map STL and sent to the server , where in between Message queue IPC ‘s are implemented. Based on the selection of user the server runs the respective algorithm to schedule the given process.

* 1. **Business Process Modeling and Management (as applicable)**

The business model used is Agile methodology, where the application is iterated after implementing each of processes in between the actual implementation of application. The steps implemented are planning phase, which involves the detailed understanding of SRS and further documenting the system requirement specification.

The next step analyzing requirements and resources and then designing.UML designing approach is used to complete this step. The next step was actual application implementation. The application was iterated for each phase of implementing it.

* 1. **Business Logic**

**4.6.1 printMenu()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **printMenu()** | | |
| **Input** | **none** |  |  |
|  |  |  |  |
| **output** | Prints menu |  |  |
| **process** | Prints menu which is to be selected by the user | | |

**4.6.2 handleMenuSelection()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **handleMenuSelection()** | | |
| **Input** | selection | int |  |
|  |  |  |  |
| **output** | Selected input | int | Takes input from user as selection and validates |
| **process** | Validates the input number taken by the user and calls further process accordingly | | |

**4.6.3 fcfs()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fcfs()** | | |
| **Input** | struct process | structure | A structure that consists of parameters that are given by user |
| **Output** |  |  | Scheduled output according to fcfs algorithm |
| **process** | Stores algorithm related input in map and gives scheduled output | | |

**4.6.4 sjf()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **sjf()** | | |
| **Input** | struct process | structure | A structure that consists of parameters that are given by user |
| **Output** |  |  | Scheduled output according to SJF algorithm |
| **process** | Stores algorithm related input in map and gives scheduled output | | |

**4.6.5 priority()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **priority()** | | |
| **Input** | struct process | structure | A structure that consists of parameters that are given by user |
| **Output** |  |  | Scheduled output according to priority algorithm |
| **process** | Stores algorithm related input in map and gives scheduled output | | |

**4.6.6 rr()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **rr()** | | |
| **Input** | struct process | structure | A structure that consists of parameters that are given by user |
| **Output** |  |  | Scheduled output according to Round Robin algorithm |
| **process** | Stores algorithm related input in map and gives scheduled output | | |

**4.6.7 scheduler()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **scheduler()** | | |
| **Input** | 1 | int |  |
| **Output** | Starting scheduler |  | Intimating user that scheduler is starting |
| **process** | According to input ,it calls scheduler function to run | | |

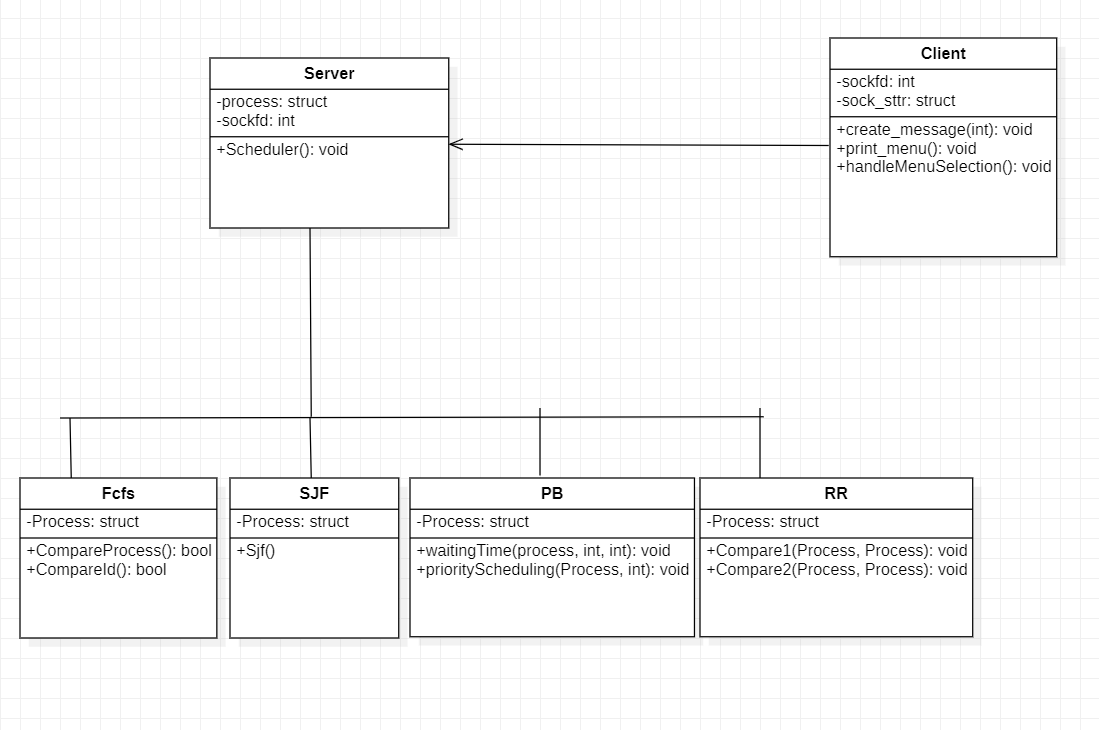
**4.6.8**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **createMessage(option)** | | |
| **Input** | option | string | The option selected by user is selected in Map STL |
| **Output** |  |  |  |
| **process** | Stores option choosen by user in map STL | | |

* 1. **Variables**

The variables used for implementing application are all instance variables of class and important variables used are process\_path , arrival\_time, burst\_time, for each of algorithm and quantum\_time for RR ,priority for priority scheduling algorithm. These are the important variables used. Some default variables that are needed to implement server-client mechanism are included.

* 1. **Activity / Class Diagrams (as applicable)**

**

Class Diagram inserted above represents the key entities involved in implementing the application. Here client acts as user Interface and gives the user given input to the server and server to scheduler. The function of algorithms are called based on the input given by the user.

* 1. **Data Migration**

Not applicable for our application.

**4.10.1 Architectural Representation**

The architectural representation such as use case, sequence ,data flow diagrams of different levels are designed and represented in design sections accordingly.

**4.10.2 Architectural Goals and Constraints**

The main goal of designing different UML is to understand the key interties that are to be implemented and understand the workflow while implementing the application. They helped us to understand the step by step process to be implemented. The constraints are, sometimes the implemented UML doesn’t represented the system functionality perfectly.

**4.10.4 Architecturally Significant Design Packages**

*Not applicable for our application*

**4.10.5 Data model**

*Not applicable for our application*

**4.10.6 Deployment View**

*Not applicable for our application*

**5.Environment Description**

This section involves the hardware environment used and required to implement our application.

* 1. **Time Zone Support**

Any timezone is applicable.

* 1. **Language Support**

C++, System Programming, IPC.

* 1. **User Desktop Requirements**

We use cloud machine and WinSCP as desktop Requirements.

* 1. **Server-Side Requirements**

*Not applicable for our application*

* + 1. **Deployment Considerations**

Hardware, Software, File Storage, Session Storage

* + 1. **Application Server Disk Space**

Not applicable for our application

* + 1. **Database Server Disk Space**

Not applicable for our application

* + 1. **Integration Requirements**

An Integration is connecting systems, applications and devices together so that you have a better flow of data and processes.

* + 1. **Jobs**

Not applicable for our application

* + 1. **Network**

Network used to implement this application is TCP LAN

* + 1. **Others**
  1. **Configuration**

Operating system, Processor.

* + 1. **Operating System**

An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface.

4GB RAM, Processor Intel (R) Core (TM) I3-7020U CPU @, 64 bit operating

system, x 64-based processor.

* + 1. **Database**

Not applicable for our application

* + 1. **Network**

Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation by which applications can exchange data.TCP works with the Internet Protocol, which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules that define the internet. The Internet Engineering Task Force defines TCP in the Request for Comment.

* + 1. **Desktop**

Linux OS

Desktops are used extensively by lodging managers and their assistants to keep track of guests bills, reservations, room assignments, meetings and special events. And also used to order food, beverages and supplies as well as to prepare reports for hotel owners and top-level managers.

**6. References**

<https://www.researchgate.net/publication/3878121_Simulation_of_CPU_scheduling_algorithms>

<https://www.ukessays.com/essays/computer-science/simulation-of-scheduling-algorithms-computer-science-essay.php>

<https://www.chegg.com/homework-help/questions-and-answers/1-introduction-programming-project-simulate-cpu-scheduling-algorithms-discussed-class-writ-q34107104>

**7. Appendix**

Not applicable