High Level Design & Low Level Design

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

**PROCESS CONTROLLER**

**Document Control :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 28/11/2022 | 0.1 | Group-6 | Understanding SRS | | | |  | |
| 2/12/2022 | 0.5 | Group-6 | Design phase | | | |  | |
| 7/12/2022 | 1.0 | Group-6 | Code Implementation | | | |  | |

[1. Introduction 5](#_Toc368912248)

[1.1. Intended Audience 5](#_Toc368912249)

[1.2. Acronyms/Abbreviations 5](#_Toc368912250)

[1.3. Project Purpose 5](#_Toc368912251)

[1.4. Key Project Objectives 5](#_Toc368912252)

[1.5. Project Scope and Limitation 6](#_Toc368912253)

[1.5.1. In Scope 6](#_Toc368912254)

[1.5.2. Out of scope 6](#_Toc368912255)

[1.6. Functional Overview 6](#_Toc368912256)

[1.7. Assumptions, Dependencies & Constraints 6](#_Toc368912257)

[1.8. Risks 6](#_Toc368912258)

[2. Design Overview 6](#_Toc368912259)

[2.1. Design Objectives 6](#_Toc368912260)

[2.1.1. Recommended Architecture 7](#_Toc368912261)

[2.2. Architectural Strategies 7](#_Toc368912262)

[2.2.1. Design Alternative 7](#_Toc368912263)

[2.2.2. Reuse of Existing Common Services/Utilities 7](#_Toc368912264)

[2.2.3. Creation of New Common Services/Utilities 7](#_Toc368912265)

[2.2.4. User Interface Paradigms 7](#_Toc368912266)

[2.2.5. System Interface Paradigms 8](#_Toc368912267)

[2.2.6. Error Detection / Exceptional Handling 8](#_Toc368912268)

[2.2.7. Memory Management 8](#_Toc368912269)

[2.2.8. Performance 8](#_Toc368912270)

[2.2.9. Security 8](#_Toc368912271)

[2.2.10. Concurrency and Synchronization 8](#_Toc368912272)

[2.2.11. Housekeeping and Maintenance 8](#_Toc368912273)

[3. System Architecture 8](#_Toc368912274)

[3.1. System Architecture Diagram. (Not Necessary) 8](#_Toc368912275)

[3.2. System Use-Cases 9](#_Toc368912276)

[3.3. Subsystem Architecture 9](#_Toc368912277)

[3.4. System Interfaces 12](#_Toc368912278)

[3.4.1. Internal Interfaces **Error! Bookmark not defined.**](#_Toc368912279)

[3.4.2. External Interfaces 13](#_Toc368912280)

[4. Detailed System Design 14](#_Toc368912281)

[4.1. Key Entities 14](#_Toc368912282)

[4.2. Detailed-Level Database Design 14](#_Toc368912283)

[4.2.1. Data Mapping Information 14](#_Toc368912284)

[4.2.2. Data Conversion 14](#_Toc368912285)

[4.3. Archival and retention requirements 14](#_Toc368912286)

[4.4. Disaster and Failure Recovery 14](#_Toc368912287)

[4.5. Business Process workflow 15](#_Toc368912288)

[4.6. Business Process Modeling and Management (as applicable) 15](#_Toc368912289)

[4.7. Business Logic 15](#_Toc368912290)

[4.8. Variables 15](#_Toc368912291)

[4.9. Activity / Class Diagrams (as applicable) 16](#_Toc368912292)

[4.10. Data Migration 16](#_Toc368912293)

[4.10.1. Architectural Representation 16](#_Toc368912294)

[4.10.2. Architectural Goals and Constraints 16](#_Toc368912295)

[4.10.3. Logical View **Error! Bookmark not defined.**](#_Toc368912296)

[4.10.4. Architecturally Significant Design Packages 17](#_Toc368912297)

[4.10.5. Data model 17](#_Toc368912298)

[4.10.6. Deployment View 17](#_Toc368912299)

[5. Environment Description 17](#_Toc368912300)

[5.1. Time Zone Support 17](#_Toc368912301)

[5.2. Language Support 17](#_Toc368912302)

[5.3. User Desktop Requirements 17](#_Toc368912303)

[5.4. Server-Side Requirements 17](#_Toc368912304)

[5.4.1. Deployment Considerations 17](#_Toc368912305)

[5.4.2. Application Server Disk Space 17](#_Toc368912306)

[5.4.3. Database Server Disk Space 17](#_Toc368912307)

[5.4.4. Integration Requirements 17](#_Toc368912308)

[5.4.5. Jobs 17](#_Toc368912309)

[5.4.6. Network 18](#_Toc368912310)

[5.4.7. Others 18](#_Toc368912311)

[5.5. Configuration 18](#_Toc368912312)

[5.5.1. Operating System 18](#_Toc368912313)

[5.5.2. Database 18](#_Toc368912314)

[5.5.3. Network 18](#_Toc368912315)

[5.5.4. Desktop 18](#_Toc368912316)

[6. References 19](#_Toc368912317)

[7. Appendix 19](#_Toc368912318)

# 

# Introduction

The aim of the project “PROCESS CONTROLLER” is basically an application that controls and monitors different processes Basically this is a menu driven application that works on user input. It contains a list of 4 to 5 different operations that are given by the user like for adding, resuming, deleting and stopping a process. And after the execution of each process, it will also display the current status of process in the console window.

## 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.

## Acronyms/Abbreviations

|  |  |
| --- | --- |
| UT | Unit Test |
| PM | Process Manager |
| IT | Integrated Test |

## Project Purpose

Process Controller application is designed to manage and control two or more process. It performs different operations on the processes which are given as input by the user

## Key Project Objectives

1.It allows to make connection between the client and the server.

2.It allows the client and server to send and receive messages once the connection

is established.

3.It allows the user to select various options as it is a menu driven process.

Option 1: adds the process

Option 2: stop the process

Option 3: resume the process

Option 4: kills the process

Option 5: Display the statistics information

Option 6: Delete PIDs from PM list

Option 7: Exit.

4. It allows to close the client server connection when all the processes present

in the process manager are terminated.

## Project Scope and Limitation

### In Scope

This application cannot be used as real-time application as this is a simpler prototype of real-time use

### Out of scope

Required to involve techniques such as Socket Programming, Signals, I/O Multiplexing to complete the project successfully.

## Functional Overview

The menu selection feature is provided for the client. Single client or multiple clients can login into the system and select their required option. The server side validates the client and processes the information sent by the clients. This processing can done by message being sent and received by both client and server. The server sends responses for respective received messages. The server also provides error messages in few instances. The whole project will be built on TCP client-server model.

## Assumptions, Dependencies & Constraints

* Should be UNIX based/ should have any linux/putty installed.
* Assume that source code occupies more than 250 kb and less than 600kb

**Constraints:** Not applicable for this project

## Risks

The potential risks are changes in market, customer satisfaction.

# Design Overview

This section gives a brief description of flow of data and application. It gives an idea of features required by the application and can be described as visual aid of the project flow. It is a roadmap of a contingency plan of the project.

## Design Objectives

Design objectives include different operations like add a process, resume a process, kill the process and display the statistics.

If user selects option1 then it will add a process and inside that firstly the pids are been provided by the user to the server. User cannot select any other option until unless he enter option 1.

If the user selects option 2 then it will stop the process.

If the user selects option 3 then it will resume the process and the stopped process will start working after choosing this option.

If the user selects option 4 then it will kill the process and automatically the process is terminated, and the program stops.

If the user selects option 5 then it will display all the statistical information such as cpu usage and current status of program whether the program is running or not.

If the user selects option 6 then it will delete the process from PM list.

### Recommended Architecture

DFD,UML Architecture. Any real-world system is used by different users. The users can be developers, testers, businesspeople, 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.

## Architectural Strategies

DFD-0, DFD-1, ER diagram, flow diagram and class diagram can be considered as main architectural strategies.

* **DFD-0:** It gives information of the top-level flow of an application. In this case, it describes general flow of work done in the PM
* **DFD-1:** It gives information on the next level flow of the application. In this case, it gives information on flow of work in PM, how can a client connect to server and perform operations.
* **Class Diagram:** It gives an overall description of various classes involved in the development phase of the project.

.

### Design Alternative

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

### Reuse of Existing Common Services/Utilities

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

### Creation of New Common Services/Utilities

Used existing resources to develop the application specific services

### User Interface Paradigms

1. Client
2. Server

### 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.

### 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.

### 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 leak detection, and profiling. Valgrind was originally designed to be a free memory debugging tool for linux.

### Performance

System performance needs to be accurate while checking for the running processes and the performed operations.

### Security

The source code is designed in that way so that only 4 pid’s are given to the process manager by the user. Client cannot use any of the other pid’s which is not present in that 4 pids by its own and even will not able to do any modifications in the pid’s.

### Concurrency and Synchronization

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

### Housekeeping and Maintenance

NOT APPLICABLE

# System Architecture

UML design is the shortest form of “Unified Modelling Language”. The purpose of this modelling 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

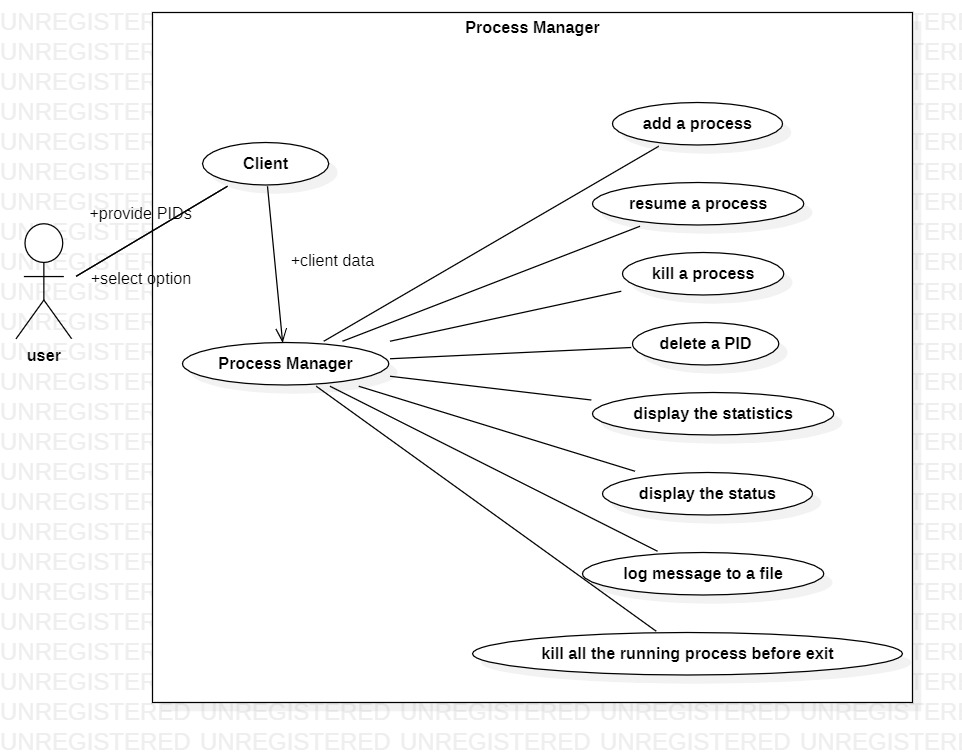
# System Architecture Diagram. (Not Necessary)

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, and processes, All the different performed operations are taken as use-case. Here we are considering four to five options as use-cases.

In sequence diagram we have user, client, server & program manager acts as a lifelines. Whereas user will select one option among four options those are add the process, resume the process, kill the process, delete a pid , display the status .

## 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 processes needed by the actors how they are interacted.

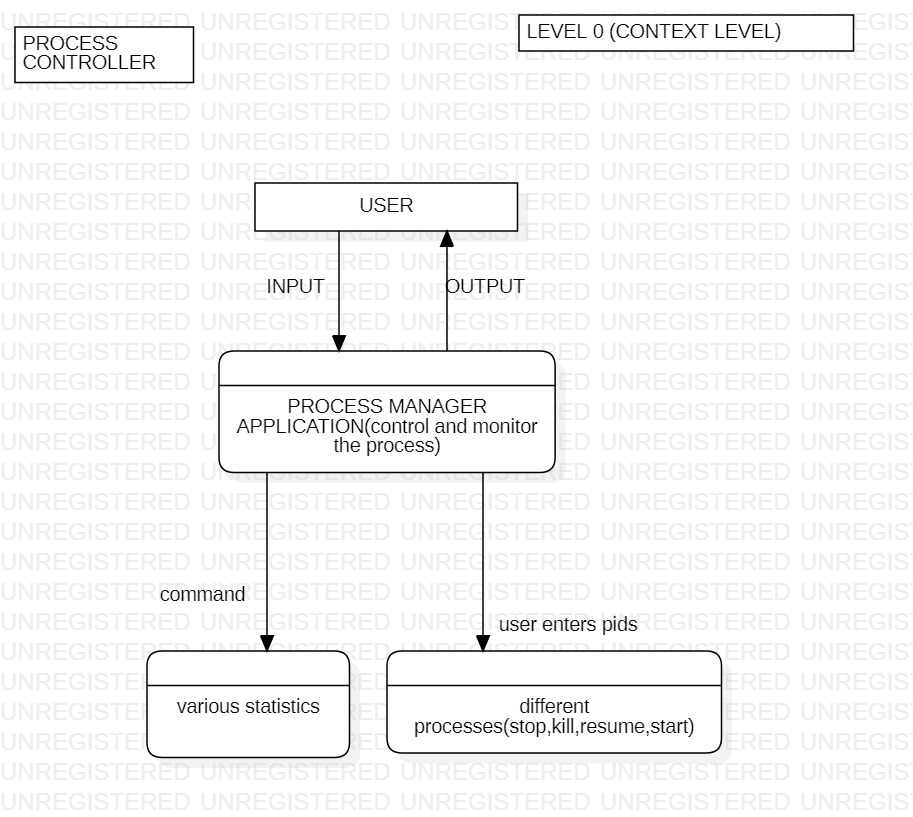


## Subsystem Architecture

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 is 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 user who will select among the 4 to 5 options provided and enter the option data whereas user will send the data and client will process these data to the server and similarly server will process the data to program manager and at last client will display the output to the user.

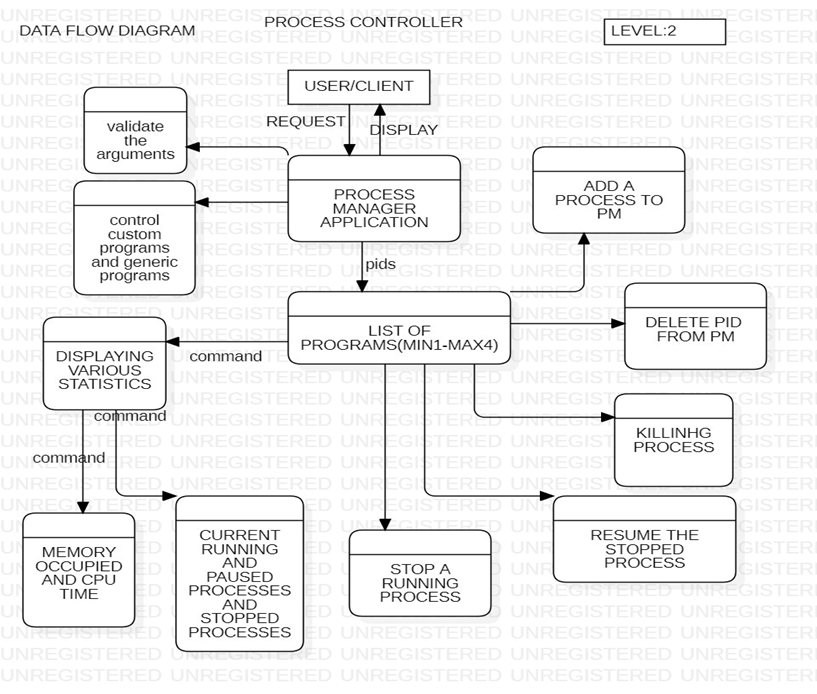
Level 1 Diagram:

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 subprocesses.

Diagram

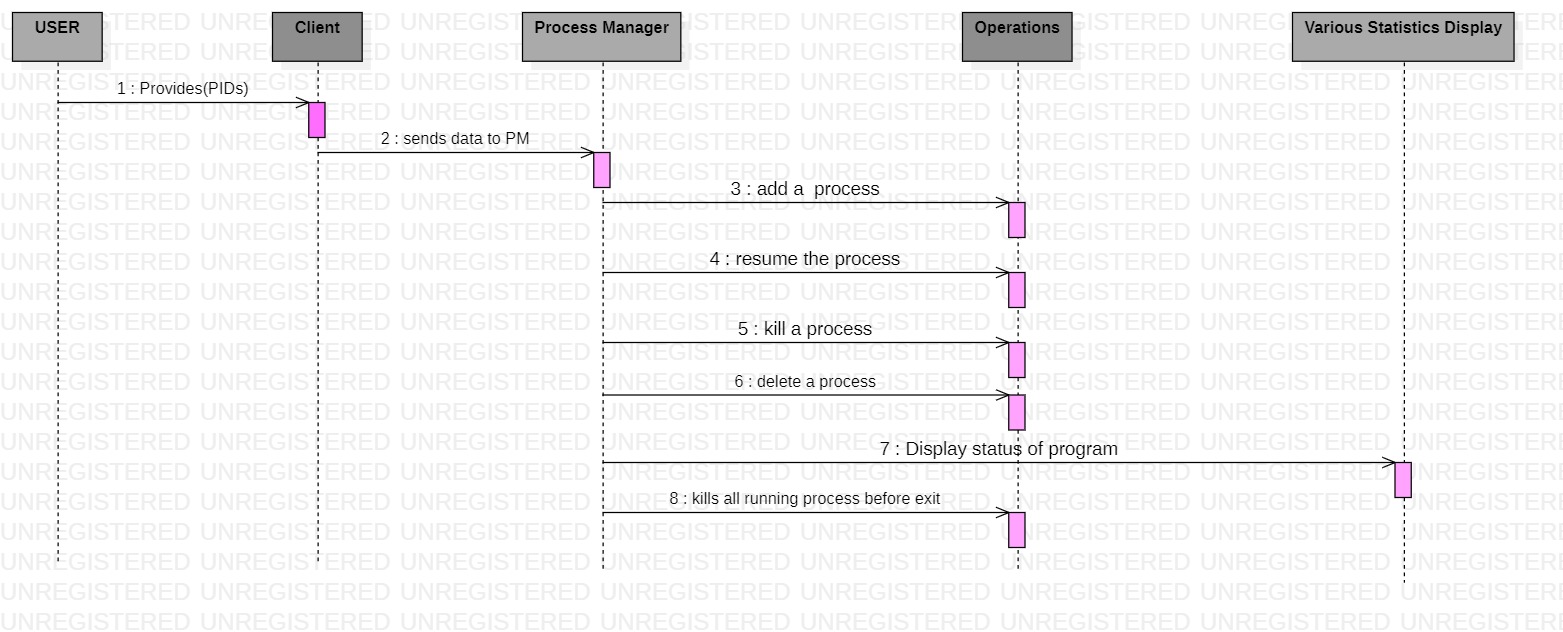
Description automatically generated

Level 2 Diagram:



## 3.4 System Interfaces

A sequence diagram is a Unified Modelling 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, process manager, operations, various statistics Display acts as a life-lines. Whereas user will select one of the options among six options those are add a process, resume the process, kill the process, delete a process, display status of program, kills all running process before test.

User provides the pid’s to the client will send the data to the process manager then the process manager contains some operations that are performed given as input by the user to the client. On the basis of the option entered by the user after performing the operations the output is been displayed in the console.

### 

### 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

# 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.

## Key Entities

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

## Detailed-Level Database Design

Not applicable for our application

### Data Mapping Information

Not applicable for our application

### Data Conversion

Not applicable for our application

## Archival and retention requirements

Not applicable for our application

## Disaster and Failure Recovery

Not applicable for this application

## 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 process manager and decide whether to manage processes to be implemented as per client interest.

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

## 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.

## Business Logic

Q In this section, we are going to represent the important entities implemented in business logic while implementing application.

### client main.cpp

|  |  |  |
| --- | --- | --- |
| Name | main |  |
| Input | Pids, option, process number | Type-int |
| Output | Call functions and pass parameters | Type-int |
| Process | Prints menu and takes input from user and call respective functions |  |

### client.cpp

**4.7.2.1.**

|  |  |  |
| --- | --- | --- |
| Name | sendPid |  |
| Input | Pids | Type-int |
| Output | sendPids | Type-int |
| Process | It takes pids as arguments and send it to server |  |

**4.7.2.2.**

|  |  |  |
| --- | --- | --- |
| Name | sendOption |  |
| Input | Option and process number | Type-int |
| Output | Send data to server | Type-int |
| Process | It sends user selected option to server |  |

**4.7.2.3.**

|  |  |  |
| --- | --- | --- |
| Name | receiveData |  |
| Input | Receive message | Type-string |
| Output | Print message | Type-string |
| Process | It receives message and print the received message |  |

### server.cpp

### 4.7.3.1.

|  |  |  |
| --- | --- | --- |
| Name | receivePid |  |
| Input | It receieves Pid from client | Type-int |
| Output | Stores the pids in map | Type-int |
| Process | It receives pid and stores in a map |  |

**4.7.3.2.**

|  |  |  |
| --- | --- | --- |
| Name | receievData |  |
| Input | Receives data and process number | Type-int |
| output | Perform operations and send message to client | Type-string |
| Process | It receives data, process number and perform operation |  |

**4.7.3.3.**

|  |  |  |
| --- | --- | --- |
| Name | acceptClient |  |
| Input | It accepts client connection |  |
| output | Client file descriptor | Type-int |
| Process | It accepts client connection and creates client file descriptor |  |

## Variables

### Server class

sockfd(int)

clientfd(int)

n(int)

socketAddr(struct sockaddr\_in)

clientAddr(struct sockaddr\_in)

op1(ofstream)

thread(pthread\_t)

### Client class

### sockfd(int)

serveraddr(struct sockaddr\_in)

op1(ofstream)

Server file descriptor, Client file descriptor, pid’s , op for menu driven inputs

## Activity / Class Diagrams (as applicable)

Table

Description automatically generatedWe have created a client server connection between the client and the server using sockets.

## Data Migration

Not applicable for this application

### Architectural Representation

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

### 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 represent the system functionality perfectly.

### Architecturally Significant Design Packages

Not applicable for this application

### Data model

Not Applicable for our project.

### Deployment View

Not applicable for this application

# Environment Description

This section provides all the descriptions of the environment required by the application such as hardware requirements, software requirements, time zones etc.

### Time Zone Support

Any time zone is applicable

## 5.1.2. Language Support

CPP on Linux, System Programming.

## User Desktop Requirements

We use putty and WinSCP as desktop requirements.

## Server-Side Requirements

We use cloud machine and WinSCP as desktop Requirements.

### Deployment Considerations

Hardware, Software, File Storage, Session Storage

### Application Server Disk Space

Not applicable for our application

### Database Server Disk Space

Not applicable for our application

### Integration Requirements

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

### Jobs

NOT APPLICABLE

### 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.

### Others

## Configuration

Operating system, Processor.

### 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 bits operating

system, x 64-based processor.

### Database

Operating system, Processor, disk place, memory.

### Network

Network is a process of assigning network settings, policies, flows and controls. In a virtual network, its easier to make network configuration changes because physical network devices appliances are replaced by software removing the need for extensive manual configuration.

### Desktop

LINUX, OS.

# References

# Appendix

**Change Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QMS Template Version Control (Maintained by QA)** | | | | | |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
| 28/11/2022 | 1.0 | Whole team | | Initial Version | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |