

**Process Controller**

Software Requirements

Specification Version -0.1

Document Control:

|  |
| --- |
| Project Revision History |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Author | Brief Description  Of changes | Approver signature |
| 29/11/2022 | 0.1 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |
| --- |
| Team Members |

|  |  |
| --- | --- |
| **Employee Id:** | **Name** |
| 46281548 | Ruhi Shaik |
| 46281286 | Alekhya Saripella |
| 46279283 | Pahuljeet Kaur |
| 46279281 | Mary Gudise |
| 46281273 | Anwesha Anurakta |

**Table of Contents**

1. Introduction

1.1 Purpose

2. Overall Description

3. Specific Requirements

3.1 Functional Requirements

3.1.1. Process Manager

3.1.2. Menu Interface

3.1.3. List of PIDs

3.1.4. Supporting Statistics

3.1.5. Killing Process before Exit

3.1.6. Same PIDs not allowed twice

3.1.7. Logging messages to file

3.1.8. Error Handling

3.2 Non-functional Requirements

3.2.1. Multifile-Multidirectory

3.2.2. UML diagrams

3.2.3. HLD, LLD

3.2.4. RTM

3.2.5. Modularity of Code

3.2.6. Reusability of Code

3.2.7. Maintainability of Code

4. Technical Requirements

5. Technologies and System Environment

**Software Requirement Specification**

**1.Introduction**

The introduction of the Software Requirements specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. This document lays out a project plan for the development of the network switching to determine where to send each incoming message frame. The intended audience includes developers, testers, project managers and the client. The detailed requirement of the process controller is provided in this document.

* 1. **Purpose**

The purpose of this document is to describe the system requirements needed for a process controller.

**2.Overall Description**

Process controller is a Linux based application to control and manage the different processes such as starting a process, pausing a process and killing the process, etc…In this application a list of PIDs is provided by client to the server and the server can execute some operations like starting and stopping etc for each process.

The Process Manager is an application that monitors, and controls different processes and it has a menu-based solution for accepting user input and displaying output. A list of programs can be given as input by the user and the application should display various statistics information. The PM application should be able to exit only after stopping all the current running processes within PM.

**3. Specific Requirements**

**3.1 Functional requirements**

**3.1.1. Process Manager**

* The Process Manager is an application that monitors and controls different processes.

**3.1.2. Menu-Interface**

* Options are provided in the menu interface for the user to select an option according to his choice

**3.1.3. List of PIDs**

* PIDs are entered by the user, minimum 1 and maximum of 4 PIDs and are sent to the PM to maintain the list of PIDs.

**3.1.4. Supporting Statistics**

* PM should display various stats for each process based on given process PID (ex: memory occupied by that process, CPU time)
* Also displays Various statistics information (current process, stopped process, paused process etc...)

**3.1.5. Killing Process before Exit**

* PM should be able to exit only after killing all the current running process.

**3.1.6. Same PIDs Not Allowed Twice**

* User is not allowed to enter the same PID twice

**3.1.7. Logging messages to a File**

* Various messages are logged to a file by the Process Manager

**3.1.8. Error Handling**

* This requirement refers to the response and recovery procedures from error conditions for all scenarios present in the application.

**3.2 Non-Functional Requirements**

**3.2.1. Multi file-Multidirectory**

* This requirement refers to that PM application is developed in Multifile-Multidirectory environment such that creating set of directories and making easy to demonstrate and create files in nested directories which in result maintains the reusability, modularity and maintenance of code

**3.2.2. UML diagrams**

* Unified Modelling language which is intended to provide a standard way to visualize design of a system. Designed use-case and sequence diagram for this PM application to get the detailed view of its purpose.

**3.2.3. HLD\_LLD**

* High Level and Low-Level design requirement specify the whole project system architecture and design constraints used, Interfaces used etc… HLD describes the overall architecture and application as whole whereas LLD describes the micro-level design of the system

**3.2.4. RTM**

* Requirement Traceability Matrix is a document used to verify that all the requirements are connected to test cases. The principal use of RTM is to confirm that all the requirements will be accounted for in the testing phase

**3.2.5. Modularity of code**

* In order to write solutions for bigger problems, it’s important to divide your projects into smaller portions, where each smaller problem is implemented in a separate subdirectory in your project. This will enable you to concentrate on smaller subproblems, which are less overwhelming and easier to deal with.

**3.2.6. Reusability Of Code**

* When you abstract your code by implementing single or related logic into individual modules and sub packages, you can then use them anywhere in your project by just importing them rather than trying to rewrite them elsewhere in your project.

**3.2.6. Maintainability of Code**

* This is perhaps the most important reason to do code abstraction. It’s very important that portions of your project code be loosely coupled together. Good project code should enable anyone without knowledge outside the project to easily make changes to it. This will help during team collaboration. When you use modules well, you only must make changes to the specific module you need to change.

**4. Technical Requirements**

* P thread
* C pp File handling
* Socket Programming
* CPP Language
* CPPUNIT to automate unit testing
* Val grind to detect memory leak
* Make file
* Multi file multi directory solution with two step compilation process.

**5. Technologies and System Environment**

* We will be using IPC mechanisms, signalling and C++ language to develop this application in the given time period.
* We will be working on a Linux environment with gcc compiler and vim editor.
* Developed this application using tools like Valgrind for debugging, gdb and WinSCP for transferring of files.