

**Custom Protocol Implementation**

**Software Requirement Specification (SRS) Document**

**Sprint Implementation**

**Project Timeline: 02.01.2023 to 14.01.2023**

**INDEX**

1. Introduction

1.1 Purpose -------------------------------------------------- 4

1.2 Intended audience -------------------------------------------------- 4

1.3 Intended use -------------------------------------------------- 4

1.4 Scope -------------------------------------------------- 4

1.5 Definition -------------------------------------------------- 4

2. Overall description -------------------------------------------------- 4

2.1 User needs -------------------------------------------------- 5

2.2 Assumptions and dependency -------------------------------------------------- 5

3. System feature and requirements -------------------------------------------------- 5

3.1 Functionality -------------------------------------------------- 5

3.1.1 G2\_FR01 -------------------------------------------------- 5

3.1.2 G2\_FR02 -------------------------------------------------- 5

3.1.3 G2\_FR03 -------------------------------------------------- 5

3.1.4 G2\_FR04 -------------------------------------------------- 5

3.1.5 G2\_FR05 -------------------------------------------------- 5

3.1.6 G2\_FR06 -------------------------------------------------- 6

3.1.7 G2\_FR07 -------------------------------------------------- 6

3.1.8 G2\_FR08 -------------------------------------------------- 6

3.1.9 G2\_FR09 -------------------------------------------------- 6

3.1.10 G2\_FR10 -------------------------------------------------- 6

3.2 Technical requirements

**3.2.1 G2\_TR01 --------------------------------------------------** 6

**3.2.2 G2\_TR02 --------------------------------------------------** 6

**3.2.3 G2\_TR03 --------------------------------------------------** 6

3.3 System requirement -------------------------------------------------- 7

3.3.1 Tools to be used -------------------------------------------------- 7

3.3 System feature -------------------------------------------------- 7

4. Reference -------------------------------------------------- 7

### 1.INTRODUCTION: -

The introduction of the software requirement specification provides an overview of the entire

software. The entire SRS is an overview description purpose, scope, tools used and basic

description. The aim of this document is to gather, analyze and give an in-depth insight into

the complete custom message protocol system by defining the problem statement in detail.

The detailed requirements of the custom message protocol are provided in this document.

* 1. **Purpose**: **-** The main purpose of this project is to take custom action according to the message type between the client and server and. Both agree to a set of protocols and perform actions
  2. **Intended Audience: -**
* Development Team
* Maintenance Team
* Clients

Since this a general-Purpose Software Thus any one Can access it.

* 1. **Intended Use: -** This application is carried out between the client and server.

The client/server(sender) sends the request, and the receiver processes the request.

* 1. **Scope: -** The purpose of this document is to describe the requirements to track all messages between the client and server and keep an organized history of limited past messages. The request might be placed from both client and server side. All the information is stored in a retrievable manner in a suitable data structure.
  2. **Definitions: -**

1. Client: The user who places the message request

2. Server: The receiver who processes the request sent by client

In this project the client and server work as both sender and receiver

**2. OVERALL DESCRIPTION: -**

A custom protocol application tracks action information. The client and server together agree to a message protocol to carry out the corresponding actions. It is an application for the communications you make between client and server to carry out actions to be performed. |Here, if the client acts as user, then the server becomes the receiver and if server becomes user client becomes the receiver. The receiver gets the request and processes the request sent by user. The message actions can be creating/modifying/updating files. You can use a suitable data structure to store the message type for easy retrieval of data and custom action. Both client and server maintain a record of 10 older messages and if one side automatically exits the other one also gets closed.

**2.1 User Needs: -**

* Performing actions according to selection of message type
* Message type has a basic menu to select the operation we want to perform
* The actions which can be carried out are creating and writing contents into a file, deleting a file, editing the contents of file
* The command operations carried out are ls command and cat command

**2.2 Assumptions and Dependency: -**

* User must give command line input in specified format
* User has the latest version of Ubuntu Linux installed.
* User should have the latest version of cloud.
* The client has either 4GB or more RAM.
* The service is used preferably on a desktop or laptop.

**3.SYSTEM FEATURES AND REQUIREMENTS: -**

**3.1 Functionality Requirements: -**

**3.1.1 G2\_FR01**: The application should be client-server based. Between the two a custom messaging protocol will be implemented. If client sends a request, then server processes the request and vice versa.

**3.1.2 G2\_FR02**: There will be message header, message data. Message header relates to the one who is sending the request either client or server. Message data refers to type of message. List of messages should be maintained in an appropriate data structure and will be common between server and client.

**3.1.3 G2\_FR03**: For each message, there will be a custom action to be taken by the receiver (server or client). Each message type has its own specified action. The list of messages and corresponding actions will be maintained in appropriate data structure.

**3.1.4 G2\_FR04**: Message parsing also must be implemented as per the message header and appropriate action should be performed.

**3.1.5 G2\_FR05**: Proper error handling to be done for incorrect messages, messages with invalid data. Only specified action for specified message should be done. No other action should take place except the corresponding action.

**3.1.6 G2\_FR06**: Both server and client should maintain history of up to 10 older messages. This is an optional testcase.

**3.1.7 G2\_FR07**: Support for statistics both on server and client (number of messages, errors, etc.)

**3.1.8 G2\_FR08**: Logging mechanism with various log levels like info, debug, error etc.

**3.1.9 G2\_FR09**: The message actions can be creating/modifying/updating files or can be running any Linux commands such as “ls”, “cat” etc. It must perform certain action respective to message and should follow protocol

**3.1.10 G2\_FR10**: Both server and client should exit automatically if other side is exited. The client should not exist if server exits and vice versa

**3.2 Technical Requirements: -**

**3.2.1 G2\_TR01**: **Process Synchronization:**

It is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

**3.2.2 G2\_TR02: Shared Memory in Linux:**

Each process has its own address space, if any process wants to communicate with some information from its own address space to other processes, then it is only possible with IPC techniques and shared memory is one such popular technique which is efficient and fastest.

**3.2.3 G2\_TR03: Socket Programming in C – TCP**

A socket is a communications connection point (endpoint) that you can name and address in a network. Socket programming shows how to use socket APIs to establish communication links between remote and local processes.

### 

### ****3.3 System Requirements: -****

### ****3.3.1.**** Tools to be used:

* C File Handling
* C Language
* System Programming
* Port no: - 4444
* Port address: **127.0.0.1**

**3.4 System Features: -**

### Supportability: The system is easy to maintain.

* Design Constraints: The system is built using only C language and System programming.

### Usability: Custom message protocol is essential for keeping digital records of

### Limited previous history. Custom protocol is the process of storing and

### processing data to act related to the message type. The software can handle

### information in modules or functions using suitable data structure

### Reliability & Availability: The system is available when the user is requested for service. The system is available 24/7.

### Performance: The system will work on the user’s terminal.

**4.REFERENCE: -**

1. <https://www.ibm.com/docs/en/i/7.1?topic=programming-how-sockets-work>
2. <https://www.geeksforgeeks.org/socket-programming-cc/>
3. https://www.geeksforgeeks.org/client-server-model/