# Software Requirements Specification

# Server Chat

**November 19, 2019**

### Team Members

- Enrico Bragastini
- Loris Pesarin
- Davide Pizzoli
- Simone Tomelini

### Document Control

#### Change History

| Revision | Change Date | Description of changes |
|---|---|---|
| v0.5 | 05/11/2019 | Project Release Plan Complete |
| v0.7 | 12/11/2019 | Iteration #1 Complete |
| v0.9 | 19/11/2019 | Iteration #2 Complete |
| v1.0 | 19/11/2019 | Project Complete |

#### Document Storage

This document is stored in the project's GIT repository at:
https://github.com/pizidavi/2020_5BI_team3_Pizzoli/tree/master/docs

#### Document Owner

Enrico Bragastini is responsible for developing and maintaining this document.

# Table of Contents

# 1. INTRODUCTION

## 1.1 Overview

Team 3 was commissioned by prof. L. Decarli, a professor at the ITIS G. Marconi instutute, to develop both Client Chat and Server Chat applications to understand the importance of a unique and open protocol and also to learn how to program with network standards and protocols.

## 1.2 Goals and Objectives

The Goals and Objectives of our applications are: 1. Send a private message to another client 2. Send a public message to all the clients connected to the server 3. Send a multicast message to a specific set of users 4. Check the list of all the online users

## 1.3 Scope

The scope of our little project is to understand how different network protocols and being able to program a more complex applications that uses these protocols. Moreover, it was necessary to conceive and create a higher-level protocol in order to create different chat-type packets.

# 2. GENERAL DESIGN CONSTRAINTS

## 2.1 Product Environment

Since we made two different applications, we have to distinguish the two environments where these two applications will work: - ***Client App***: Will work on user's machine. The end user will only need to open the application on his own machine. There will be a client app for each user. - ***Server App***: Will work on a central server machine. There will only be a single server application for each *"chat environment"*

## 2.2 User Characteristics

Since it is a simple Chat Application, all the users will be equated. If we want, there will be a system administrator, in charge of keeping under control the server application.

## 2.3 Potential System Evolution

Since it is a simple Chat Application for learning purposes, no future system update has been planned so far.

# 3. NONFUNCTIONAL REQUIREMENTS

## 3.1 Operational Requirements

There are no operational requirements at this time.

## 3.2 Performance Requirements

No requirements for speed have been set forth.

### 3.3 Security Requirements

The Server Chat application needs to force clients to sign-up the first time they connect, and to sign-in for each future connection. The server application has to keep track of clients' ip addresses in order to trace back users in case of inappropriate uses.

### 3.4 Safety Requirements

No safety requirements were identified for this application.

### 3.5 Legal Requirements

Since it is an application made for learning purposes only and used in a school private network, no legal requirement is needed.

---

# 4. SYSTEM FEATURES

### 4.1 Public Message

A Client can send a public message to every online Client, and this is made possible by a specific *"packet type"* from our custom-made chat protocol.

```
 packet-type 20: Sending Public Message
    CL → SV [20][LEN][Text]

packet-type 21: Receiving Public Message
    SV → CL [21][LEN][Addresser][0][Text]
```

### 4.2 Private Message

A Client can send a private message to another Client, by using these *packets* from our custom-made chat protocol:

```
 packet-type 22: Sending Private Message
    CL → SV [22][LEN][Addressee][0][Text]

packet-type 23: Receiving Private Message
    SV → CL [23][LEN][Addresser][0][Text]
```

### 4.3 Multicast Message

A Client can send a message to a specific subset of clients, by using these *packets* from our custom-made chat protocol:

```
 packet-type 24: Sending Multicast Message
    CL → SV [24][LEN][Addressee][0][Addressee][0]...[Text]

packet-type 25: Receiving Multicast Message
    SV → CL [25][LEN][Addresser][0][Text]
```

### 4.4 Getting the online-users list

A Client can ask the Server to sen him the list of all the online users, by using these *packets* from our custom-made chat protocol:

```
 packet-type 42: Requesting Online Users List
    CL → SV [42][LEN]

packet-type 43: Receiving Online Users List
    SV → CL [43][LEN][User][0][User]...
```