Software Requirements Specification

Change History

VERSION	DATE	MODIFIER	DESCRIPTION OF CHANGE
0.1	04 Jan 2020	Damiano Ghisellini	Initial draft.

1. Introduction

1.1 Overview

The chat protocol is a project inside the G. Marconi institute and created by the students.
It is primarily intended as the conclusion of an experience on network programming.

This document lists the requirements that the final program must have.

Planning, development methods, development phases, final results or test procedures are not managed here.

1.2 Goals and Objectives

The main objective of the project is to provide an application that allows authorized users to communicate with each other.

In general, each program must:

- Provide a textual or graphical interface to communicate.
- Send and receive messages and view their content.
- Handle any incorrect packages.

1.3 Scope

The students are divided in client developers and server developers.

- Client: the program must provide the ability to send and receive packets and view related messages.
  Messages can be private, public or multicast. In addition, packages must be implemented to register, log in and log out.

Finally, a package must be provided to request a list of connected users.
- Server: the program must check the correctness of the packets that arrive, and return one with the result of the verification.
  In addition, it must forward messages to its recipients.
  Finally, it must manage users who want to register, log in and log out.

The purpose of the project is to provide a sort of space where you can communicate with other users and get general information about the chat.

1.4 Assumptions

It is assumed that all the computers participating in the project are connected to each other and authorized by the server to communicate and that they all have Python 3.7 or higher installed.
Each user must register first.

There are no additional security or access control mechanisms.

2. General Design Constraints

2.1 Product Environment

In the scope of this project, each program must provide an interface, textual or graphic, to allow the user to read and write messages.
There is no storage space for these messages.

2.2 User Characteristics

Users can be divided in:

- Who manages the server, that is responsible for registering users, blocking them and checking packages.
- Clients, that have the same priority and can only send and receive messages and request information.

2.3 Mandated Constraints

- The program must follow the protocol given to us.

2.4 Potential System Evolution

No future evolution of the project is expected, as it only represents an exercise.
A possible evolution of the program, however, could include message storage and a more developed graphical interface.

3 Nonfunctional Requirements

3.1 Operational Requirements

- The application has to allow different users to view messages, while preserving the accuracy of the data.
- To be accepted, all packets must follow the protocol.

3.2 Performance Requirements

No performance requirements have been identified for this project.

3.3 Security Requirements

The only security mechanism resides on the server, which decides which users to block and which ones not.
Before sending a message, a user must log in. The first time, it is necessary for him to register.

3.4 Safety Requirements

Safety, regarding the correctness of packets with respect to the protocol, is managed by the server.

3.5 Legal Requirements

No legal requirements have been identified for this project.

3.6 Other Quality Attributes

Each host must send packets that follow the protocol and no errors should occur during program execution.

4 System Features

4.1 Feature: Implement a voting system

4.1.1 Description and Priority

Cost: none
Risk: low
Value: medium

4.1.2 Use Case: Voting

Actors: voters, supervisor.

Description: This use case begins when there is a democratic choice to be made.

Basic Path:

1. The supervisor checks that everything is working, even in terms of security.
2. Voters are asked to cast their vote via the client.
3. Once everyone has voted, the votes are counted and a decision can be made.

Alternative Path:
If one of the steps fails, the security of the program must be checked.

4.1.3 Additional Requirements

N/A