!

!thiscord

Group #2 – Communications Project

Software Requirements Specification

Revision History

Date	Revision	Description	Author
06/14/2021	1.0	Initial Version	John Kyle Lintao
06/21/2021	1.1	Updated Purpose and Scope	John Kyle Lintao
06/22/2021	1.2	Organized References (Use Cases)	Haoxiang Hu
06/23/2021	1.3	Added Acronyms / Definitions, and References (Use Case Diagrams)	Manisha Yonjan
06/23/2021	1.4	Updated Specific Requirements and Non-functional Requirements	Mitchee Costelo
06/23/2021	1.5	Updated References (Sequence Diagram)	Manjesh Prasad
06/23/2021	1.6	Added References (Class Diagram)	John Kyle Lintao
06/29/2021	1.7	Cleaned up and condensed sentence structure throughout SRS document	Mitchee Costelo
06/29/2021	1/8	Cleaned up the sequence diagram to get rid of narrow scops	Manjesh Prasad

Table of Contents

1. Purpose

This document outlines the requirements for the communication program "!thiscord" (pronounced as "not this cord"). The purpose of !thiscord is to provide an intuitive instant messaging system catered towards businesses and corporations of various sizes.

1.1. Scope

This document will catalog the user and system requirements for !thiscord system. Additionally, it will give guidelines on what features are included and how they interact. This document is intended to assist in the development process of !thiscord. It will not, however, document how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

Acronyms

- !TC: !thiscord communication program
- CLS: Client Software
- SES: Server Software
- GRC: Group Chat
- OOC: One on One Chat
- GUI: Graphical User Interface

1.3. References

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

Client Notes

1.4. Overview

!TC is a new chat application designed as an easy communication medium for the workplace environment. This application is built to allow users to use GRC and OOC to communicate with one another. !TC was built around two core systems, the CLS and the SES. The user will directly interact with CLS which as a result will communicated with the SES where there it will handle tasks like directing messages or changing user data.

2. Overall Description

2.1. Product Perspective

2.2. Product Architecture

The system will be organized into 2 major modules: the CLS module and the SES module.

Note: System architecture should follow standard OO design practices.

2.3. Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

2.4. Constraints

- Due to time constraints, additional features like profile pictures and file sharing can only be added if deadlines are met ahead of schedule.
- The system must be coded strictly in java and not require any additional downloads to function.
- The GRC needs to handle an “infinite” number of users.

2.5. Assumptions and Dependencies

- It is assumed that the server can handle an infinite number of employees stored in its userbase.
- An infinite number of messages and message history can be stored within the server’s message logs.
- The system does not account for connections lost to the server, so it is dependent on stable internet connections.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Messaging is done through text-based communications.

3.1.1.2 The system should be able to handle single recipient messages as well as group-based ones.

3.1.1.3 The CLS needs to connect over network to the SES through an IP address and port number.

3.1.2. CLS Module Requirements:

3.1.2.1 Upon startup, the CLS needs to check to see if it can connect to the SES. If unable to connect, the software will terminate.

3.1.2.2 After establishing a connection to the server, the CLS immediately asks for user to login using a username and password. The user will have 4 attempts to login. After the 4th failed attempt, the software will terminate.

3.1.2.3 Once logged in, the user will have the ability to create a new chat room with any other user located in the userbase. In the chat room, the user can send messages to one or more people.

3.1.2.4 Depending on their account status, the user might have Admin privileges which would allow them to do things such as add or remove users, change passwords, or restart the SES.

3.1.3. SES Module Requirements:

3.1.3.1 The server needs to be able to have a database containing the information of all users, and message history.

3.1.3.2 The SES is responsible for directing and distributing messages to the CLS of the correct clients.

3.1.3.3 The SES needs to create a single thread when a client connects to the server. When multiple clients connect, the server needs to create a multithread that allows different people to connect to different ports.

3.1.3.4 The server needs to be able to reboot and successfully log everybody out when instructed by an admin to reset.

3.2. External Interface Requirements

3.2.1 The system needs to have an interface that can handle the following tasks: Check to see if the client can successfully connect to the server, log in, change user details, create chat rooms, and browse message history.

3.2.2 Additionally, the GUI needs to be adaptive so if the user has admin privileges it can handle additional functions such as adding or removing a user or resetting the server.

3.2.3 The GUI needs to be designed in a way to handle an “infinite” number of users or messages. That means designing it to be able to scroll through a user list or having pages needs to be implemented.

3.3. Internal Interface Requirements

3.3.1 The system needs to be designed in a way to handle an infinite number of users or messages. As a result, that means everything needs to be coded in a way that future proofs expanding sizes.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The System must encrypt data being transmitted over the Internet.

4.1.2 If a user gets deleted, to ensure the privacy of the user and the company, all logs and records need to be deleted. Any message history an active user has with that deleted profile need to be removed as well.

4.2. Environmental Requirements

4.2.1 System cannot require any additional downloads or addons in order to function properly.

4.2.2 The application is required to be coded completely in the Java language.

4.3. Performance Requirements

4.3.1 The system still needs to send and receive messages at similar speeds to every recipient.