# Waterfall Chat

Requirements Document

Author: Tegan Lamoureux

## Table of Contents

# 1 Introduction

This is the requirements document for the Waterfall Chat server and client program, part of the term project for Professor Fei Xie's CS300 class, which is a requirement for the Computer Science program at Portland State University.

## 1.1 Purpose and Scope

This document will give a high level overview of the product, in which we describe the general features of the program. It will also outline the users and stakeholders, describing both who will use it, and also who has a vested interest in its completion. We will also give functional and nonfunctional requirements. Finally, we will describe the milestones and deliverables of the project.

## 1.2 Target Audience

This document is intended for the stakeholders, in order to effectively convey an overview of the functional requirements and future milestones of the project, as well as the graders, in order to convey my overall understanding of writing a requirements document.

## 1.3 Terms and Definitions

- User - A client (user of the program) that logs into the server.
- Server - A program that accept connections from multiple users, and allows them to communicate with each other.
- Chatroom - An online space that allows users to communicate with each other by sending public messages to everyone else in the room.
- Private Messaging - The sending of a message from one user to another user.

# 2 Product Overview

This section is intended to give a high level overview of the project, along with a list of stakeholders, which are those with a vested interest in the project, and use cases, which are typical scenarios a user will encounter in the program and the resulting program flow.

This program will create a chatroom, at its most basic definition. It will consist of a server to which an indefinite number of clients connect, which allows them to chat with the entire server (chat room), or between individual clients (private messaging).

Users (clients) will have accounts stored on the server, with login credentials and logs of all chats participated in. When connecting to the server, the user (client) will enter their credentials. If these do not match any records in the server, they will receive an error message and a prompt to make a new user account. If it matches, they will be logged in, which does multiple things. It notifies all users (clients) on the server that the new user has logged in, it gives them access to account details, and gives them access to chat logs. It also allows them to send chats from their user account to the server, or other users. Along with this, once a user is logged in, they can see a list of all other users logged into the server.

The program will be implemented such that it will run all on the same machine (server and all clients). It will not be capable of communicating across a network other than localhost for the time being. All passwords will be stored in plaintext in a data structure/text file, and therefore should not be sensitive passwords used for other accounts. User logs will also be stored in plaintext on the machine, and therefore should not contain data or information the user would like to keep private.

## 2.1 Users and Stakeholders

Here I will outline the users and stakeholders of the project, their interests in its completion, and typical scenarios users will encounter while running the program, along with the resulting program flow and server-client interactions. Users of the program will be anyone who wishes to communicate using the application, and thereby anyone who opens and runs a client program that connects to the server.

### 2.1.1 Stakeholder 1: Tegan Lamoureux

Tegan has a vested interest, as she is the sole creator and maintainer, and will be directly affected by the success (and thereby grade) of the project. Her role is the design, development, and implementation of the entire project. After design and review, she will modify the project as needed to meet customer requirements. After delivery, it will be her responsibility to maintain the software for any future errors, bugs, or changes required, under the scope of the original contract.

### 2.1.2 Stakeholder 2: Project Grader

The project grader also has an interest in the project completion, as it's their responsibility to determine whether the original project requirements were met, and to determine whether the creator (Tegan) will receive the full amount of payment (grade), or a reduced quantity, determined by her ability to meet the original project specifications.

## 2.2 Use cases

Here I will describe different use cases for the program. These are scenarios in which the program will typically be used, and what the typical client (user) will encounter while running the program.

### 2.2.1 Connection Without an Account

This scenario occurs when a user connects to the server and doesn't already have an account. They start the client program, connect to the server, and are prompted to login by the client. They don't have login credentials, though, so they click on "Create new account.". This allows them to access a sub-form of the client program in which they can enter information to create their new account. They will enter a username and password both of which are checked. The password is entered twice, checked for matching by the client (input errors). The username is checked for repeat usernames by the server. If acceptable (no repeat usernames, password verification matches), this is sent to the server and saved as a new account on the server. The user is now allowed to login at the client login form. Here they enter their credentials, and it's sent to the server and checked against the accounts stored there. If the server returns a match, they are logged in and allowed to send messages to whomever they like (covered in a later use-case), and the server notifies all other clients that the user has logged in (entered the chatroom.. If the server doesn't return a match, they are given an error, and then allowed to try again.

### 2.2.2 Connection With an Account

This scenario occurs when a user connects to the server and already has an account. They start the client program, connect to the server, and are prompted to login. Here they enter their credentials, and it's sent to the server and checked against the accounts stored there. If the server returns a match, they are logged in and allowed to send messages to whomever they like (covered in a later use-case), and the server notifies all other clients that the user has logged in (entered the chatroom.. If the server doesn't return a match, they are given an error, and then allowed to try again.

### 2.2.3 Sending a message to the room

This scenario occurs when a user is already connected and logged into the server with their account. At the client screen, they are given three main areas to interact with: the user list, the messages list, and the message send area. The user list is a list of users currently connected to the server, which is selectable. The messages list is a list of all messages sent to the user (via server-wide messages or private message). The message send area is where the user can enter a message to send to someone (or everyone). In order to send a message to the entire server (room), the user simply needs to de-select any users in the user-list (so that none are selected), and then enter a message into the message send area. When they press send, it will be sent to the entire server, as well as logged to the server for all accounts.

### 2.2.4 Sending a message to a single user

This scenario occurs when a user is already connected and logged into the server with their account. At the client screen, they are given three main areas to interact with: the user list, the messages list, and the message send area. The user list is a list of users currently connected to the server, which is selectable. The messages list is a list of all messages sent to the user (via server-wide messages or private message). The message send area is where the user can enter a message to send to someone (or everyone). In order to send a message to a single user (private message), the user simply needs to select the desired user from the user list, and then enter a message into the message send area. When they press send, it will be sent to that user alone, as well as logged to the server for their account.

### 2.2.5 Checking Account Records

This scenario occurs when the user is logged in. If they wish to check their log of previous chats, they can do so on the login screen. Here, if connected to the server and logged in, they can select the button “view account logs”. This will open a form which requests all logs for that user from the server. It will then display them, and allow the user to scroll through and then exit.

# 3 Functional Requirements

Here we will outline the functional requirements of the program. These are methods of functionality that must be implemented to ensure the customer is satisfied with the outcome of the project. Therefore, these are also a list of requirements that the project author must accomplish.

## 3.1 Client Connection to the Server

All client program instances must have the ability to connect to a server, which allows for account creation, storage and validation, and the ability to communicate between the clients. This is a two way communication in which the client program sends a request to the server to connect, and then server sends a response that it recognises the connection. In this way, the client is now connected, and may communicate other actions with the server.

### 3.1.1 Client-Server Connection Protocol

This is a message or packet that’s sent initially from the client to the server. If the server receives this, it internally assigns a unique ID to the client, and sends a response back to the client, recognising that the client is now connected, and also containing the client ID the server assigns. This is required before any other communication can proceed, and the client ID is a required part of future communication between the client and server.

## 3.2 Account Creation

This is an action that a connected client can perform. On the client side, new account information (username, password) will be taken in from the user, and checked for input validity. If valid on the client side, the information will be sent to the server for further validation. If the account is created successfully, the server responds with true, if there was an error and the account wasn't created, it returns false.

### 3.2.1 Client Side Input Validation

This is a check that's performed on the client side during the initial stages of account creation. The user is asked to input a username and a password. The password is entered twice, however, into two text fields. If the passwords entered don't match, the client stops further account creation and notifies the user of an error. If they match, however, the account creation process proceeds.

### 3.2.2 Client-Server Account Creation Protocol

Once the client has validated the input from the user, the information is sent to the server. Here the server will check to see if the username is already in use for any other clients. If not, the new account is created on the server, and a response of true is sent back to the client, signalling successful account creation. If the username is already taken, however, the server responds with false, indicating the account wasn't created successfully, and that the client should try again. Once a response of true is received, the client can safely assume the account is now valid and resides on the server, ready to be logged into.

## 3.3 Account Validation

In this stage, account information (username and password) is inputted on the client side, and sent to the server. The server reviews this information, and determines whether it matches any accounts stored there. It then responds with a message on whether the account was valid or not, and if so, whether the user is now logged in.

### 3.3.1 Client-Server Account Validation Protocol

The client sends a message to the server containing the account information. Once received, the server checks all of the accounts it has stored, and determines whether there's a match. If there is, it sends out a message to all users that the new user has logged in, and matches the client ID (assigned to the client on connection to the server) with that user. This is used in future communication to determine which user is sending what. Once this is done server side, it responds with true, and the client knows the user is logged in. If there isn't a match, either with the username or password, the server responds with false, signalling the user is not logged in, and should try again.

## 3.4 Sending Public Messages

Once the client is connected and logged in, all that's required to send a public message is a communication to the server containing the client ID (matched to the user server-side), a message, and a flag stating the message is public. The server will then broadcast this message to all other logged in clients.

## 3.5 Sending Private Messages

Once the client is connected and logged in, all that's required to send a private message is a communication to the server containing the client ID (matched to the user server-side), a message, a flag stating the message is private, and the username to send the message to. The server will then broadcast this message to only the specified user.

## 3.6 Viewing Chat Logs

Once the client is connected and logged in, they have the option through the client program to view past conversations. This is done via a communication to the server to view past logs (which are stored server-side). The server responds with a series of messages containing past logs, which are then displayed client side.

# 4 Nonfunctional Requirements

Here we will outline the nonfunctional requirements of the program. These are general requirements of the program's performance, and capabilities it will (and will not) be able to meet.

## 4.1 System Size

The system should be able to handle a reasonable amount of connections from clients to the server. Reasonable, here, will be defined as less than or equal to 100 clients. For this amount, the server will handle all communications in a timely manner and will not malfunction because of user load. Connections above this amount may or may not work, and performance is not guaranteed.

## 4.2 Reliability

The server should be reliable for connections within the client volume limit specified above. Individual aspects of this reliability definition will be defined below.

### 4.2.1 Message Log Retention

The server should retain accurate logs of all communications for all users, within the client volume limit. Each user should expect the server to keep logs of all of their sent messages, as well as all of their received messages, and that these logs will be ready to view at the user's discretion.

### 4.2.2 Expectation of Delivery

The server should deliver all messages sent by connected and logged in clients to their intended recipients. Users should expect that once a message was sent, it will be delivered to the recipient(s), and logged in their history.

#### 4.2.3 Expectation of Connection Stability

The client should expect to remain connected to the server, barring any outside communication errors. Likewise, the server should maintain connection with all connected clients, in order to achieve reliable operation (within the client volume limit specification). If this communication becomes compromised for whatever reason (client stops receiving replies from the server), the user should be notified that their messages aren't being delivered, and that logs of their chats are no longer being recorded.

### 4.3 Security

The users of the program have a right to expect privacy in their communications and account records. Following this, account validation should work properly, and the server should not allow connections for a user unless that user's credentials were a match with the server-stored identity. Further, access to logs and records of chat for a user should only be available to that user once successfully logged in, and not otherwise.

# 5 Milestones and Deliverables

Here we will outline milestones and deliverables for the project, along with a high level overview of our workflow, and the intended outcome.

## 5.1 Requirements Document

In this stage of development, requirements of the program will be explored and finalized into a cohesive document. Because the project description was vague, this is where formal requirements for development will be outlined. The deliverable for this milestone will be a formal requirements document (this document), due on April 25th.

## 5.2 Design Document

In this stage of development, the in depth design of the project will be explored. A system overview first, at a high level, and then going lower until we reach system level components. This should be very close to the final system design of the project, and should aid in making a complete application. This document is a deliverable, and will be due May 9th.

## 5.3 Test Plan

In this stage of development, a test plan will be created to effectively determine whether the project meets the original requirements and functions in its intended way. This should be comprehensive, and cover all cases that will come up in the standard operation of the program. This deliverable document is due on May 30th.

## 5.4 Project Report & Final Submission

Here the final working project will be due as a deliverable, along with a short report on the development process, and how the goals were met/not met. This milestone is due June 8th.