# Group #4 – Design

Design Requirements Specification

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| mm/dd/yyyy | 1.0 | Initial Version | Your Name |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. INTRODUCTION [4](#__RefHeading___Toc19440719)

1.1. Goals and Objectives [4](#__RefHeading___Toc19440720)

1.2. Statement of Scope [4](#__RefHeading___Toc19440721)

1.3. Software Context [4](#__RefHeading___Toc19440722)

1.4. Major Constraints [4](#__RefHeading___Toc19440723)

2. DAta DESIGN [5](#__RefHeading___Toc19440724)

2.1. Client Side [5](#__RefHeading___Toc19440725)

2.2. Server Side [5](#__RefHeading___Toc19440726)

2.3. Chat Room Side [5](#__RefHeading___Toc19440727)

2.4. Message Side [5](#__RefHeading___Toc19440728)

2.5. ??? [5](#__RefHeading___Toc19440729)

3. ARCHITECTURAL and COmPonent-level design [6](#__RefHeading___Toc19440730)

3.1. Program Structure [6](#__RefHeading___Toc19440731)

3.2. Description of Client [6](#__RefHeading___Toc19440736)

3.3. Description of Server [6](#__RefHeading___Toc19440737)

3.4. Software Interface Description [6](#__RefHeading___Toc19440737)

4. User Interface Design [7](#__RefHeading___Toc19440738)

5. Restrictions, LiMitations, and Constraints [8](#__RefHeading___Toc19440738)

6. Testing Issues [9](#__RefHeading___Toc19440738)

7. Appendices [10](#__RefHeading___Toc19440738)

# Introduction

## Goals and Objectives

This document describes important aspects of the implementation of the Communications System.

## Statement of Scope

Decisions in this document are made based on the following priorities (most important first): Maintainability, Usability, Portability, Efficiency

## Software Context

Server information will be maintained within the Server singleton object. The Server objects various commands will handle the receiving and sending of all communications between the clients.

## Major Constraints

**Issue 1**: Where should we store the database?

**Option 1.1**: We can store the database in the Server object.

**Option 1.2**: We can store the database in a separate file that is properly sorted and accessible by the Server object. The Server object will sort in chronological order and within its respected chatroom.

**Decision:** NEED TO DECIDE

I think we should go with 1.1. The database should only be on the “server” so even if we did 1.2, that’s kinda the same thing as being in the Server object (only the Server accesses it because the clients only access the server).

**Issue 2**: Where should we store the information of usernames and passwords?

**Option 1.1**: We can store this information within the Server object.

**Option 1.2**: We can store the information in a separate file that the Server will search through.

**Decision:** NEED TO DECIDE

They should be stored in the database which after Issue 1 I think should be part of the Server object.

**Issue 3**: How does the joinChatRoom() Method work?

**Option 1.1**: Once a chatroom is created, the owner of the chatroom selects the users they want to participate. Then the selected user may join when they see fit. The User has the option to never join.

**Option 1.2**: Once a chatroom is created, the owner of the chatroom selects the users they want to participate. The selected users are automatically joined.

**Decision:** NEED TO DECIDE

I think we should do 1.2 because it’s probably easier to code.

**Issue 4**: If we go down the path that the creator of the chatroom is the host, and so far, there are no methods to add additional users or remove users, is there a purpose for having a setChatLock() method? There is currently no way of altering the number of users as of now.

**Option 1.1**: Add methods to add and remove Users from Chatroom. Have a setChatLock() Method. This then develops ISSUE 5.

**Option 1.2**: Once a Chatroom is initiated and a set of participants users in selected this list can no longer be altered. Furthermore, removing the necessity of a setChatLock() Method.

**Decision:** NEED TO DECIDE

I think he said people should be able to come so we should go with 1.1

Issue 5: If we want the ability to allow users to add and remove other users who should be allowed to do that?

**Option 1.1**: The host only (Seems to negate the purpose of a setChatLock() method.

**Option 1.2**: All Users with the host being able to lock it when they seem fit. (Requiring a setChatLock() Method)

**Decision:** NEED TO DECIDE

I would say 1.2.

# Data Design

## Client Side

The Client will communicate with the Server to create chatrooms with the createChatroom() method. The user that starts the chatroom is automatically the host and is the only one who may change the state of the Chatroom. A response from the Server will be displayed on the UI confirming a successful creation of a Chatroom.

The Client will frequently send Messages to the server with the combination of the createMessage() and sendMessage() methods. SHOULD THERE BE AN INDICATION OF A SUCCESSFUL MESSAGE SENT? 🡪 Yes

## Server Side

The implementation of the validateUser(username, password) will search through a file (the database) of approved personnel.

The Server will receive Messages with the getMessage() method. This message will be stored by calling recordMessage(message, date) where message will be the text message and date is the timestamp.

The Server will redistribute the data to those involved in the Chatroom with the resendMessage() method.

## Chat Room Side

## Message Side

The setMessageContent(String) implementation will require that the textual information to be contained within a String object.

SIDENOTE TO TEAM: My reasoning for purposely excluding the User and Supervisor classes from this section is because when I was going over the example document in the software\_design.pptx from class, I interpreted this section as giving a description to each module that directly handles data in our project. I felt as though the User/Supervisor classes didn’t offer much information to put into this section due to them only being invokers and not data handlers. 🡪 Makes sense

# Architectural and Component-level Design

## Program Structure

### Architecture Diagram:

## Description of Client

## Description of Server

# User Interface Design

### 

# Restrictions, Limitations, and Constraints

# Testing Issues

# Appendices