**Group Project - Communications**

**!thiscord**

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 |
| 6/22/2021 | 1.2 | Updated Use Cases | Haoxiang Hu |
| 6/23/2021 | 1.3 | Definitions/Acronyms, UML Use Case diagram and overview | Manisha Yonjan |
| 6/23/2021 | 1.4 | Updated Specific Requirements and Non-functional Requirements | Mitchee Costelo |
| 6/23/2021 | 1.4 | Sequence Diagram and flow of the applications. | Manjesh Prasad |
| 6/23/2021 | 1.4 | Class diagrams | John Kyle lintao |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Purpose 4**

1.1. Scope 4

1.2. Definitions, Acronyms, Abbreviations 4

1.3. References 4

1.4. Overview 4

**2.** **Overall Description 5**

2.1. Product Perspective 5

2.2. Product Architecture 5

2.3. Product Functionality/Features 5

2.4. Constraints 5

2.5. Assumptions and Dependencies 5

**3.** **Specific Requirements 6**

3.1. Functional Requirements 6

3.2. External Interface Requirements 6

3.3. Internal Interface Requirements 7

**4.** **Non-Functional Requirements 8**

4.1. Security and Privacy Requirements 8

4.2. Environmental Requirements 8

4.3. Performance Requirements 8

# Purpose

This document outlines the requirements for !thiscord(Pronounced “not this cord”). The purpose of the !thiscord is to provide an intuitive instant messaging and file transfer system that can be deployed in a given professional environment.

## Scope

This document will catalog the user, system, and hardware requirements for the !thiscord system. It will not, however, document how these requirements will be implemented. This system is designed to improve workplace communication and productivity, through a dedicated and secure instant messaging platform. Using a dedicated server, clients will be able to establish a stable one-on-one or group connection to collaborate on projects and improve … (need to ask about other functions. UML will give guidance on what else to emphasize in scope).

## Definitions, Acronyms, Abbreviations

Acronyms

* SRS : Software Requirements Specification
* UML: Unified Modeling Language

Definitions

* Software requirements Specification: Documentation that consists of all the functional, non functional, Use cases, features and designing process of the application.
* Use case Specification: Use case is the illustration of how the application behaves when the user interacts with the service in various different conditions.
* UML use case diagram: It is a visual representation of the functionality and the relationship of use cases, employee, admin and system.
* Class diagram: Class diagram is the structure where different classes, objects, its attributes, methods and its relationships are described.
* Sequence diagram: It is the sequential description of how different classes or objects work together in different scenarios.

## References

Use Case Specification Document – Step 2 in assignment description

Use Case ID: CASE\_1

Use Case Name: User Account join the 1 to 1 Chat room.

Relevant Requirements: User Class, Chat Class, Notification Class, Chat History Class

Pre-conditions: When users need to join in a 1 to 1 Chat Room.

Post-conditions: Users join a Chat Room and send text messages, they can search users by ID number and send chat requests. Also, they can manage chat history and notifications.

Extensions or Alternate Flows: Admin account can stop chatting at any time, clean the chat history and kick someone from the chat.

Exceptions: Two admin accounts in the chat room, permission conflict.

Related Use Cases: The system generates a chat room and a room number for users to use.

Use Case ID: CASE\_2

Use Case Name: User Account join the Group Chat room.

Relevant Requirements: User Class, Chat Class, Notification Class, Chat History Class, Group Chat Class.

Pre-conditions: When users need to join a Group Chat Room.

Post-conditions: Users join a Group Chat Room and send text messages, they can search users by ID number and send chat requests. Also, they can manage chat history and notifications.

Extensions or Alternate Flows: Admin account can stop chatting at any time, clean the chat history and kick someone from the chat.

Exceptions: Two admin accounts in the chat room, permission conflict.

Related Use Cases: The system generates a group chat room and room number for users to use and it can be searched.

Use Case ID: CASE\_3

Use Case Name: User Account Delete chat history in Chat room(Group Chat Room).

Relevant Requirements: User Class, Chat Class, Notification Class, Chat History Class, Group Chat Class, Delete Class.

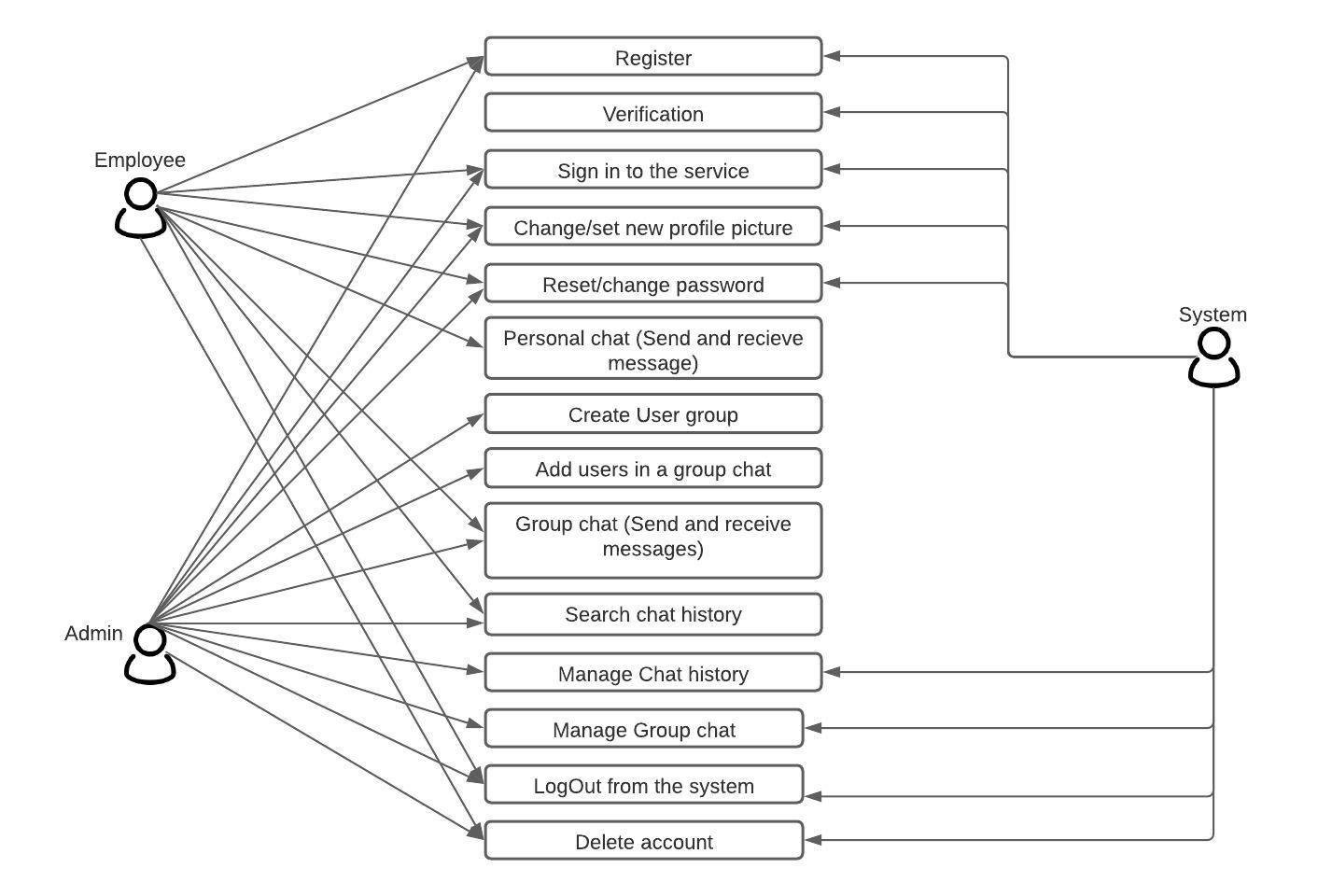
Pre-conditions: When users delete chat history in a chat room.

Post-conditions: Users want to delete chat history in chat rooms and send text messages, but it is only deleted in users’ clients. Server still saves the history. Only Admin accounts can be deleted on the server. Users can search users by ID number and send chat requests.

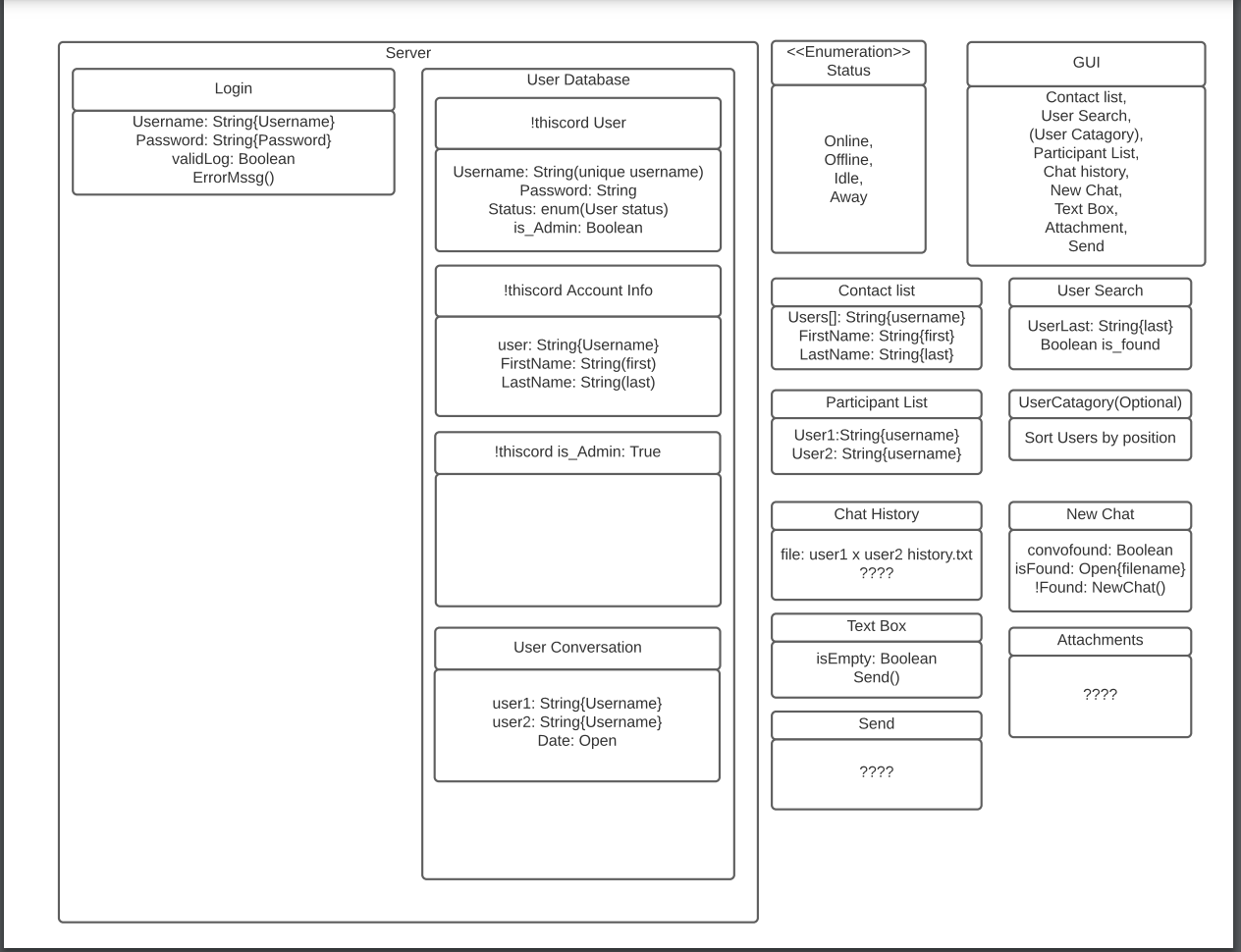
Exceptions: Admin account can delete history in server at any time.

Related Use Cases: Delete chat history.

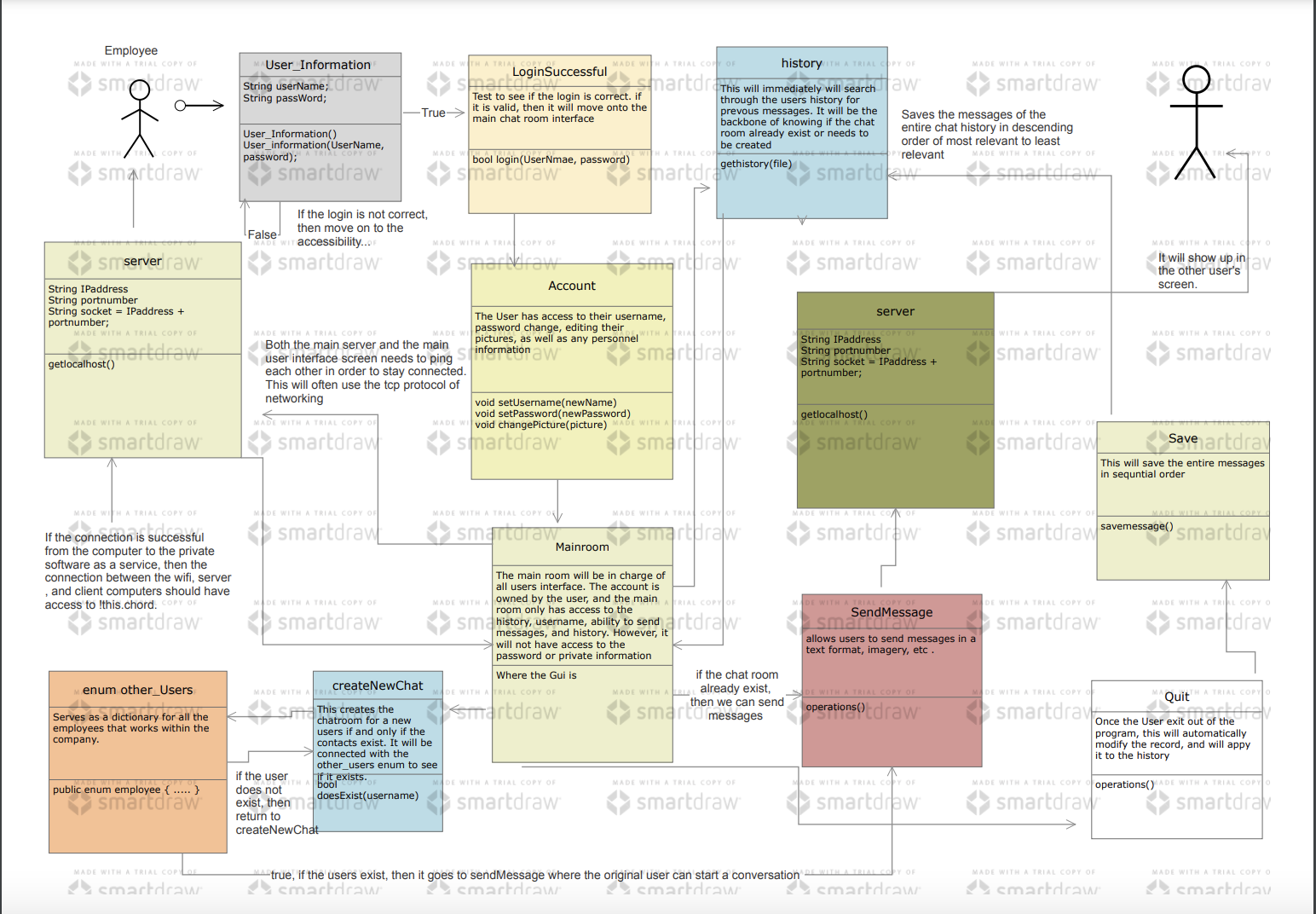
UML Use Case Diagrams Document – Step 3 in assignment description



Class Diagrams – Step 5 in assignment description



Sequence Diagrams – Step 6 in assignment description



## Overview

!thiscord is a new chat application designed as an easy communication medium for the workplace environment. As per the request of the client, this application is specifically designed only for the client’s private company. The application is built to conduct any group chats and one on one chats where different features allow for file transfer, manage user profile and various private settings.

# Overall Description

## Product Perspective

## Product Architecture

The system will be organized into 3 major modules: the Profile module, the Messaging module, and the Server module.

Note: System architecture should follow standard OO design practices.

## 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):

## Constraints

2.4.1 System must be coded exclusively in Java

2.4.2 Software should not require any download.

## Assumptions and Dependencies

2.5.1 We should assume that the system has the capacity to hold an infinite number of users.

2.5.2 We should assume that the system can handle an infinite amount of users in a group chat.

# Specific Requirements

## Functional Requirements

### 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 messages.

3.1.1.3 Clients need to be able to connect and communicate with the server to publish a communication protocol.

### Profile Module Requirements:

3.1.2.1 Users should be allowed to create their own user profile. The profile contains their username and password as well as an internal ID.

3.1.2.2 Users should have access to all users in the server.

3.1.2.3 Each user should be able to access their message history. If a user's profile is deleted, their message history is deleted as well.

### Message Module Requirements:

3.1.2.1 Message carries a list of all the recipients who are to receive the message.

3.1.2.2 The message should be added to a message history between the group or recipients.

### Server Module Requirements:

3.1.2.1 Server must store a list containing all users. It also needs to have all messages saved internally.

3.1.2.2 Server needs to be able to receive messages and use the info of the message to search through the user list and send it out accordingly.

## External Interface Requirements

3.2.1 There needs to be a functional GUI that asks for a login, password, or to sign up upon booting up. Upon successful login, they will be directed to a list with all users stored in the system. By selecting one of the users, a message log will appear where all previous and future messages will be stored.

3.2.2 A menu to adjust settings to a profile such as change username or delete the profile needs to be available in the main menu.

## Internal Interface Requirements

3.3.1 All users need to have an ID number attached to their profile so in the case that we allow for username changing in the future, their profile isn’t attached to their username.

3.3.2 The system must process a data-feed from the University billing system that contains new student records. The feed will be in the form of a comma-separated text file and will be exported from the billing system nightly with new student records. The fields included in the file are student name, student id, and student pin number.

# Non-Functional Requirements

## Security and Privacy Requirements

Example:

4.1.1 When a profile gets deleted, the system needs to be able to delete all messages and logs from everyone’s history.

## Environmental Requirements

4.2.1 System should not require any sort of download.

4.2.2 No coding language should be used outside of Java.

4.2.3 System should hypothetically be able to have an unlimited group size.

## Performance Requirements

Example:

4.3.1 System needs to handle or be coded in a way to handle unlimited group size and store an infinite list of message logs.