Group 7: Communications

Software Requirements Specification

Revision History

Date	Revision	Description	Author
9/24/2023	1.0	Initial Version	Tommy Thai
9/29/2023	1.1	Preliminary Requirements Addition	Kat Webb
9/30/2023	1.2	Non-functional Requirements, Modules	Kat, Jon, Tommy, Vansh
10/1/2023	1.3	Functional Requirements, Specific Requirements	Tommy, Jon, Kat, Vansh
10/1/2023	1.4	Use Cases Added	Kat Webb
10/2/2023	1.5	Use Case Diagrams Added	Tommy Thai
10/3/2023	1.6	Sequence Diagrams Added	Jon Vazquez
10/4/2023	1.7	Class Diagram Added	Vansh
10/4/2023	1.8	Completed SRS Document	Tommy, Jon, Kat, Vansh

Table of Contents

1. Purpose

This document outlines the requirements for the Communications System.

1.1. Scope

This document will catalog the user, system, and hardware requirements for a communications system.

1.2. Definitions, Acronyms, Abbreviations

Synchronous - happening at the same time, here meaning all parties are present and messaging in real time

Asynchronous - parties are not present at the same time, their messaging happens at different times

Server - The entity or instance that provides the communication service.

Client - The entity or instance that provides a user the ability to connect to the server.

1.3. References

Use Case Specification Document – See Page 9-11

UML Use Case Diagrams Document – See Page 12-13

Class Diagrams – See page 14

Sequence Diagrams – See page 15-17

1.4. Overview

This Communications Applications, is designed for users to communicate synchronously and asynchronously. Workers need a swift way to communicate with team members.

2. Overall Description

2.1. Product Perspective

This Chat Application will provide the ability to users to communicate with other users over text. This will vary from individual messaging and group messaging. There will also be logs of every chat for company IT users to review.

2.2. Product Architecture

The system will be organized into 5 major modules: Server module, Client module, User module, Chat module, and Log Modules.

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):

- Server Connection
- Client Login
- User Messaging/ Chatting
- Logging

2.4. Constraints

- The network communication has to be over TCP/IP connection
- The service will be coded in the Java Language
- Communication between two users will be limited to text
- Privacy should be minimized.

2.5. Assumptions and Dependencies

- It is assumed that there is no user limit
- It is assumed that accounts are created manually on the server
- It is assumed that roles are assigned to users manually on the server.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Every user should be able to send and receive message synchronously and asynchronously

3.1.2. Server Module Requirements:

3.1.2.1 The Server should be able to handle multiple network connections at once.

3.1.2.2 The Server should be able to authenticate users.

3.1.2.3 The Server should be able to receive messages from the sender client and transmit them to the recipient(s) client(s)

3.1.2.4 The Server should keep track of the port(s) that are currently connected.

3.1.2.5 The Server should have a port listening at all times, barring maintenance.

3.1.3. Client Module Requirements:

3.1.2.1 Each client should be able initiate a connection with the server.

3.1.2.2 Each user will use a client to login using username and password, both of which are an alphanumeric string.

3.1.2.1 The client should toggle users between online and offline depending on connection status

3.1.4. User Module Requirements:

3.1.4.1 There must be roles for users including, standard employee and IT employee.

3.1.4.2 The IT role must have rights to view logs of chats

3.1.4.3 Each user must be able to initiate an instance of the client.

3.1.4.4 Users must be able to send and receive messages synchronously and asynchronously.

3.1.4.5 Users must have the option to create group chats

3.1.5 Chat Module Requirements:

3.1.5.1 Each chat must have sender id, receiver id(s), message, date and time values initialized.

3.1.5.2 Each chat should also have a boolean to tell if the message has been received by the recipient(s) or not.

3.1.5.3 Each chat should use the log module to create logs for the messages.

3.1.6 Log Module Requirements

3.1.6.1 A Unique Log instance should be created when a user chats with another.

3.1.6.2 A Unique Log Instance should also be created for group conversations.

3.1.6.3 All log instances must be stored for IT viewership

3.2. External Interface Requirements

3.2.1 The application must provide an interface to the client where chat room instances can be created.

3.2.2 The application must provide a status for each user indicating the availability of the user.

3.3. Internal Interface Requirements

3.3.1 The system must have an interface for IT users to view logs of all chats.
3.3.2 All log instances will be found in the the internal interface.
3.3.3 Each chat instance will write to a log instance.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 Only IT members can view the logs
4.1.2 Every user must authenticate themselves before gaining access to the service.

4.2. Environmental Requirements

4.2.1 Application must run on Windows 10 and above.

4.3. Performance Requirements

4.3.1 The server must be able to handle multiple connections without crashing.
4.3.2 The logs must contain all historical data of the chat

Use Case ID: 0001
Use Case Name: Client Connects to Server for authentication
Relevant Requirements: 3.1.3, 3.1.2, 3.1.1
Primary Actor: Client
Pre-conditions: The client is not connected to the server, the server has not authenticated the user for use of the application.
Post-conditions: The client has been authenticated, the server is connected to the client. The user using the client is marked as online.
Basic Flow or Main Scenario:
1. The client attempts to connect to the server.
2. The server requests authentication from the client.
3. The user enters their credentials.
4. The server accepts their credentials and allows them to interact with their messaging content.
5. The user is marked as online.
Extensions or Alternate Flows: Login authentication fails; the server checks for unread messages; the server fails to mark the user as online.
Exceptions: The server fails to authenticate a registered user, the client fails to connect to the server
Related Use Cases: 0002, 0003

Use Case ID: 0002
Use Case Name: Server Updates Client Messages
Relevant Requirements: 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5
Primary Actor: Server
Pre-conditions: The client is authenticated,but has not received unread messages.
Post-conditions: The client has received any messages sent while the user was not logged into a client. All messages sent to the user asynchronously are now marked “received”.
Basic Flow or Main Scenario:

1. The server checks records for any message objects addressed to the user that had not been marked “sent”.
2. The server sends any messages addressed to the user but not marked “received” to the client
3. All messages that had not been marked received addressed to the client are now marked received, and the client can check them.
Exceptions: The server loses connection with the client, the server fails to retrieve the unreceived messages, the server fails to send the unreceived messages to the client
Related Use Cases: 0001

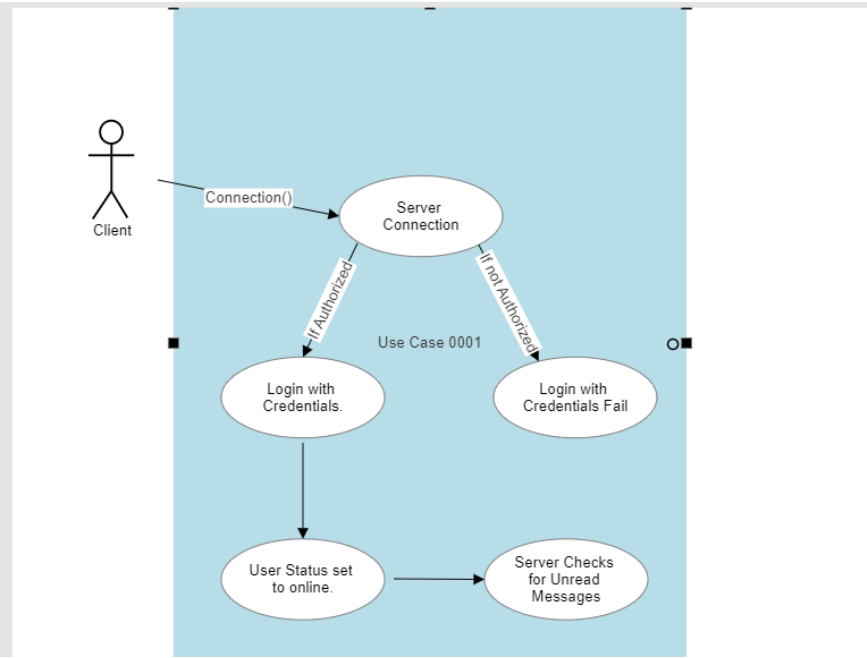
Use Case ID: 0003
Use Case Name: User Creates a Message Instance
Relevant Requirements: 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5
Primary Actor: User
Pre-conditions: The sending party is authenticated on a client that is connected to the server. The server is prepared to receive and redirect messages.
Post-conditions: The user has created and sent a message to the .
Basic Flow or Main Scenario:
1. The user creates a list of intended recipients, through user IDs.
2. The user creates a message that they want to send to the user IDs in the list.
3. The user sends this message to the server, which in turn repackages the message and sends it to the intended recipient(s)
4. The server creates a log that tracks the date sent, time sent, subject matter of the message, and whether or not the message has been received
Exceptions: The server and client lose connection, one or more of the recipients does not exist, the connection does not stay established and the traffic is not transmitted, the server does not handle the message

properly and it does not get sent to intended recipients
Related Use Cases: 0001

Use Case ID: 0004
Use Case Name: IT User Accesses Logs
Relevant Requirements: 3.1.1, 3.1.2, 3.1.4, 3.1.5
Primary Actor: IT User
Pre-conditions: The IT user needs to view a message log, the server is prepared to display logs but has not authenticated the IT user.
Post-conditions: The IT user has accessed the message log via the server's file database, for the duration of their authenticated login.
Basic Flow or Main Scenario:
1. An IT user logs into the server using their authentication
2. Once the server authenticates the IT user, the IT user may navigate to and access the logs in chronological order or may look up logs by date or participating users.
Exceptions: The server cannot authenticate the IT user, the server does not store the logs properly/cannot search the logs

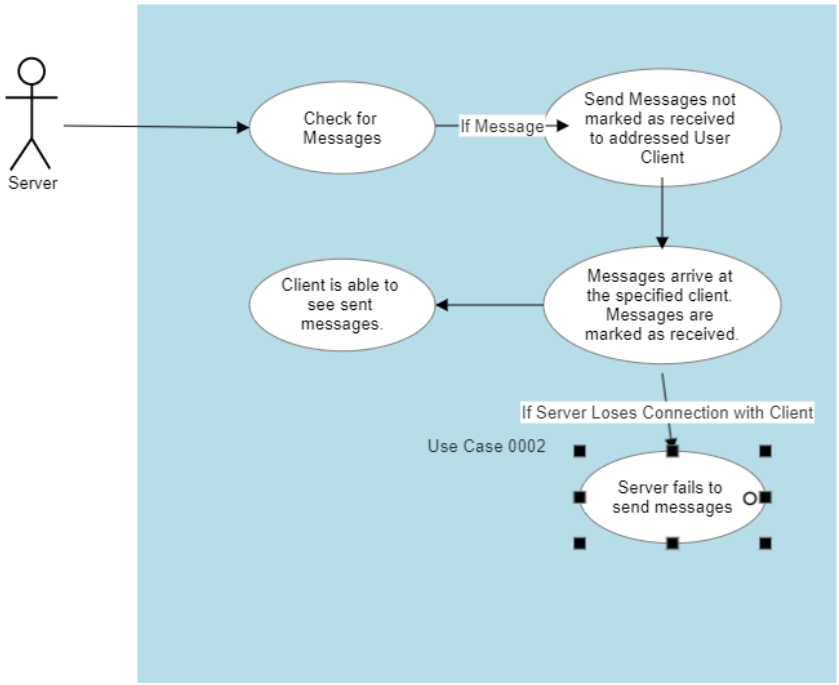
Use Case ID: 0001

Use Case Name: Client Connects to Server for authentication



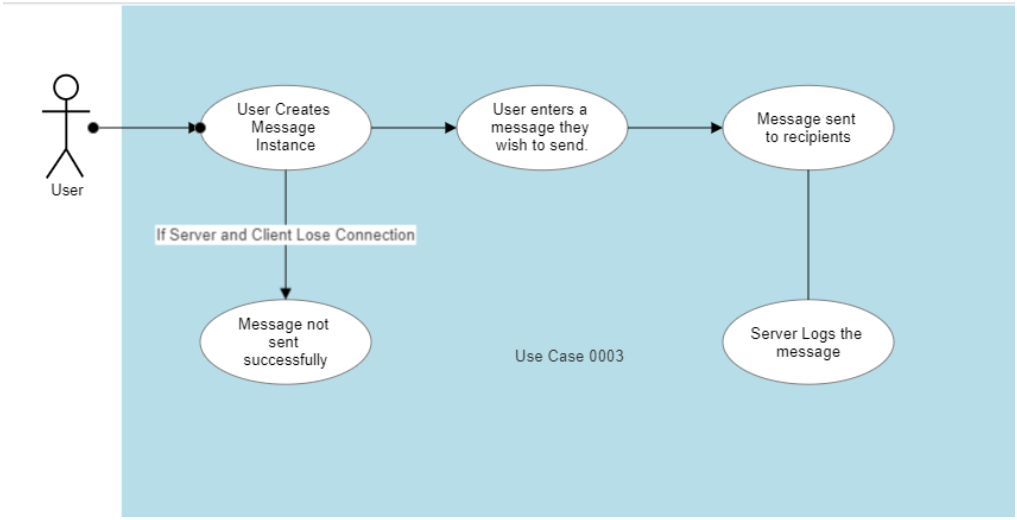
Use Case ID: 0002

Use Case Name: Server Updates Client Messages



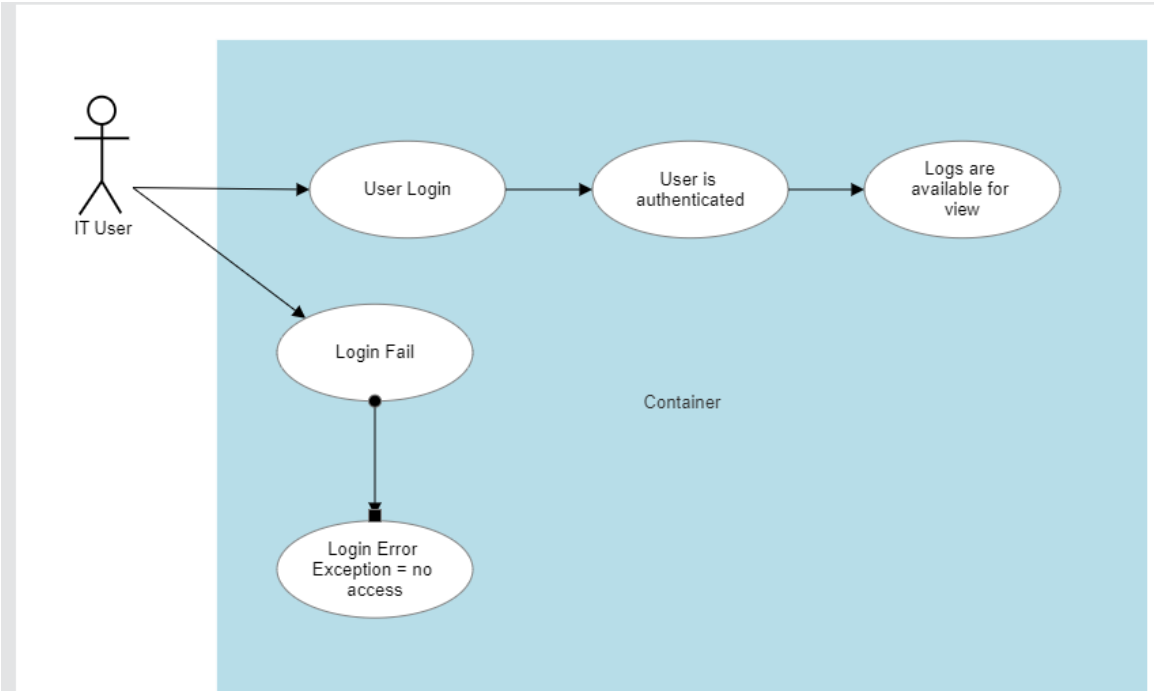
Use Case ID: 0003

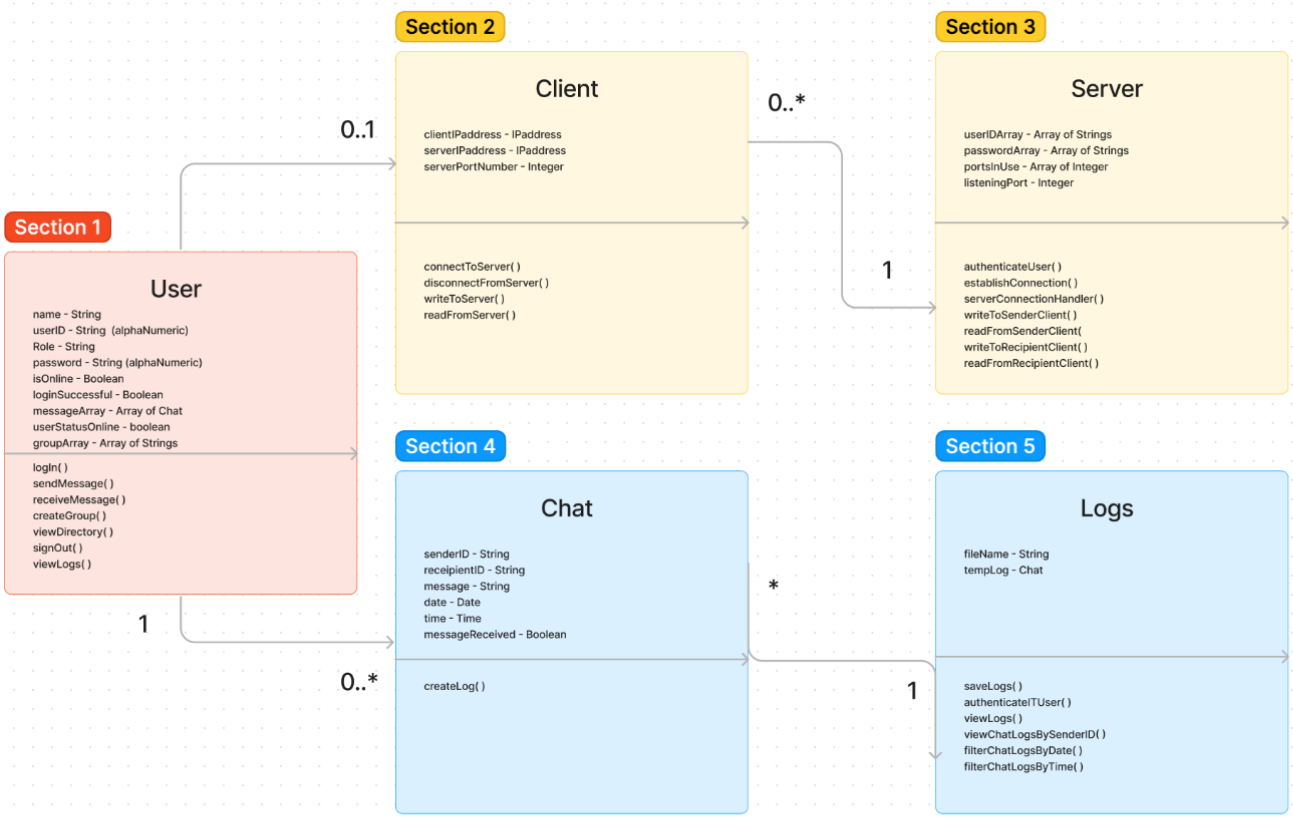
Use Case Name: User Creates a Message Instance

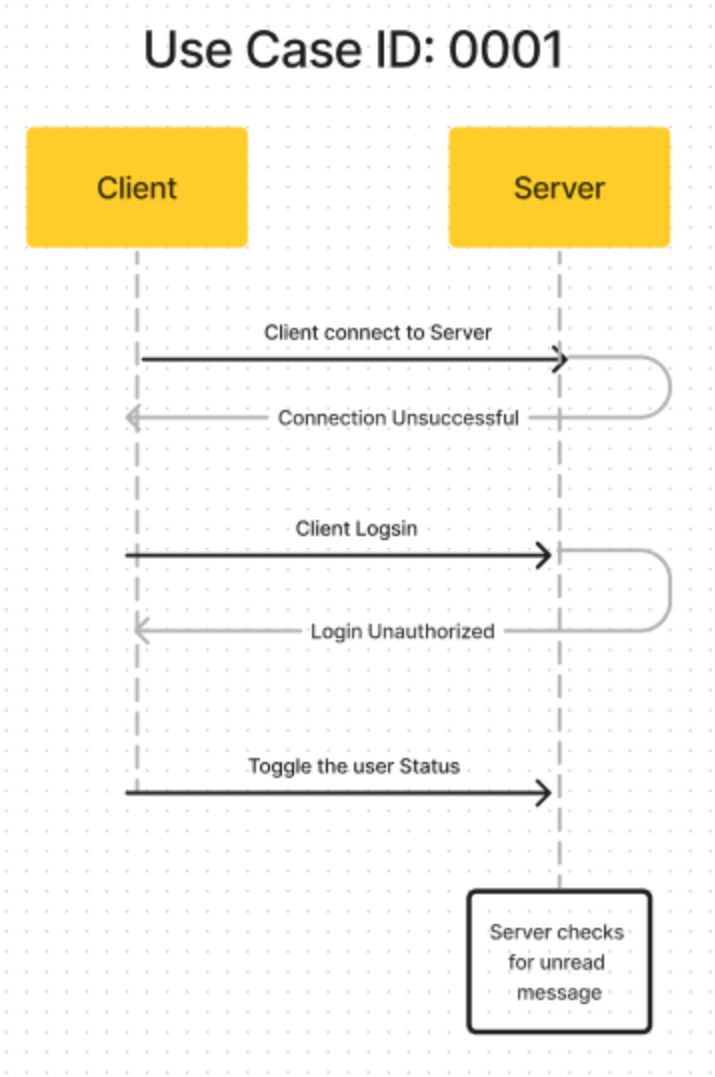


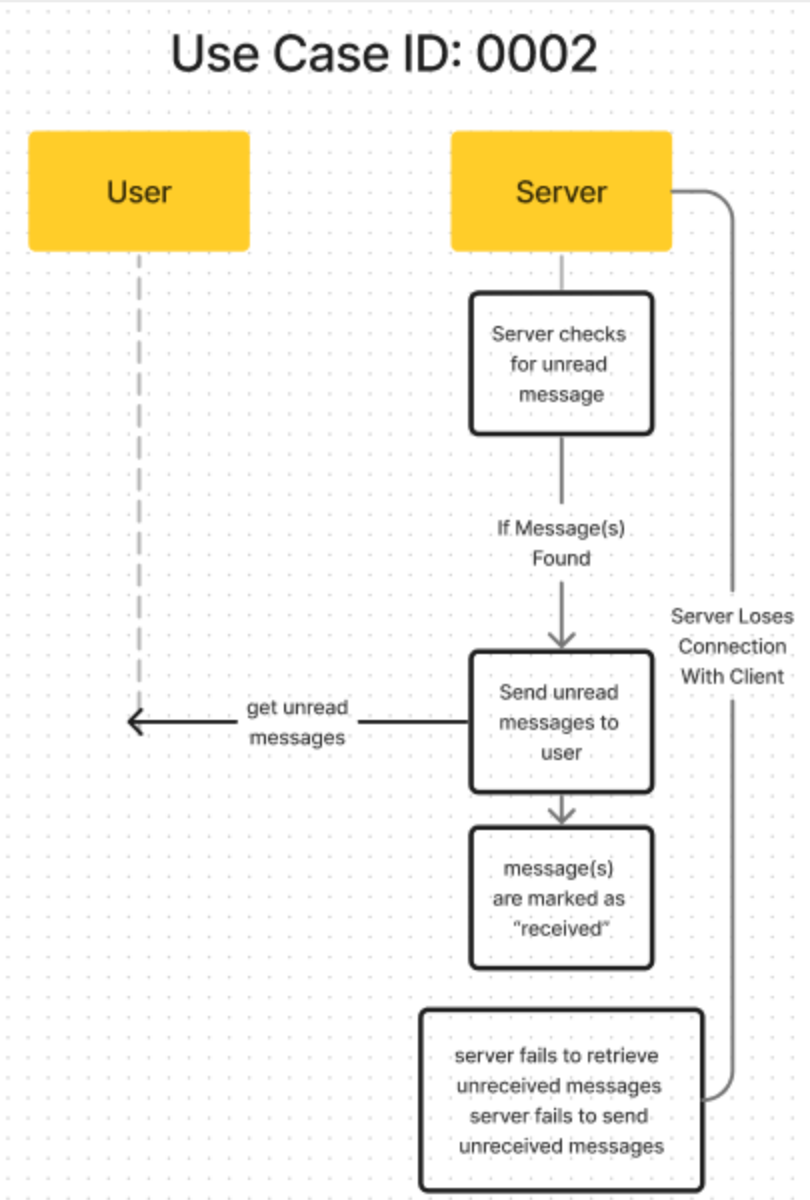
Use Case ID: 0004

Use Case Name: IT User Accesses Logs

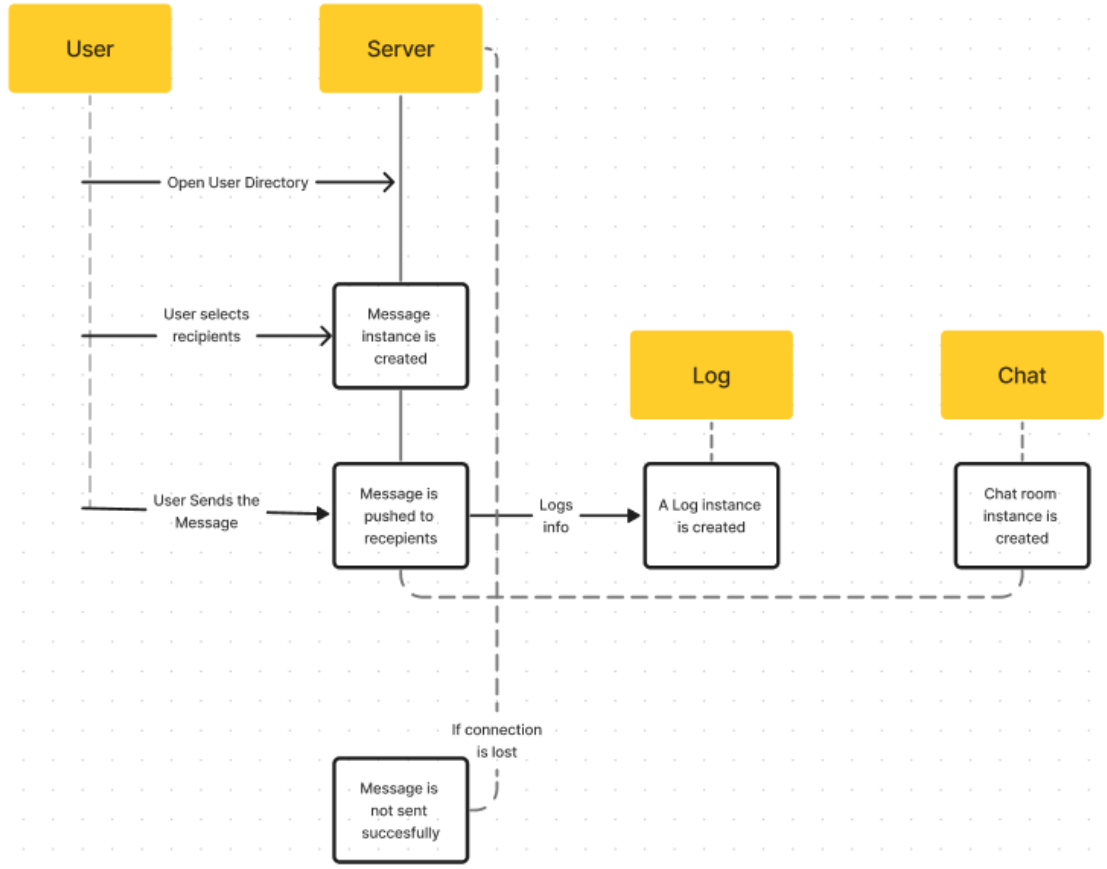



Section 1
User
name - String
userID - String (alphaNumeric)
Role - String
password - String (alphaNumeric)
isOnline - Boolean
loginSuccessful - Boolean
messageArray - Array of Chat
userStatusOnline - boolean
groupArray - Array of Strings
logIn( )
sendMessage( )
receiveMessage( )
createGroup( )
viewDirectory( )
signOut( )
viewLogs( )
Section 2
Client
clientIPaddress - IPaddress
serverIPaddress - IPaddress
serverPortNumber - Integer
connectToServer( )
disconnectFromServer( )
writeToServer( )
readFromServer( )
Section 3
Server
userIDArray - Array of Strings
passwordArray - Array of Strings
portsInUse - Array of Integer
listeningPort - Integer
authenticateUser( )
establishConnection( )
serverConnectionHandler( )
writeToSenderClient( )
readFromSenderClient(
writeToRecipientClient( )
readFromRecipientClient( )
Section 4
Chat
senderID - String
receipientID - String
message - String
date - Date
time - Time
messageReceived - Boolean
createLog( )
Section 5
Logs
fileName - String
tempLog - Chat
saveLogs( )
authenticateITUser( )
viewLogs( )
viewChatLogsBySenderID( )
filterChatLogsByDate( )
filterChatLogsByTime( )
0..1
0..*
1
1
0..*
*
1


Use Case ID: 0001
Client
Server
Client connect to Server
Connection Unsuccessful
Client Logsin
Login Unauthorized
Toggle the user Status
Server checks for unread message


Use Case ID: 0002
User
Server
Server checks for unread message
If Message(s) Found
Server Loses Connection With Client
Send unread messages to user
get unread messages
message(s) are marked as "received"
server fails to retrieve unreceived messages
server fails to send unreceived messages

Use Case ID: 0003



Use Case ID: 0004