Communications Project

Software Requirements Specification

Revision History

Date	Revision	Description	Author
02/22/24	1.0	Scope, definitions, identify use cases	Herminia
02/23/24	1.1	Use cases & UML diagrams	Herminia & Christopher
02/24/24	1.2	Section 2	Herminia
02/25/24	1.3	Updated use cases and requirements	All
02/25/24	1.4	Updated requirements	Jasan Manimbao
02/26/24	1.5	Added more use cases	Lyca Mangarin
02/27/24	1.6	Added first draft of class diagram	Jasan Manimbao
02/28/24	1.7	Added Message Sequence Diagram	Lyca Mangarin
02/29/24	1.8	Updated class diagram	Jasan Manimbao
02/29/24	1.9	Document revision	All
03/28/24	2.0	Updated all use cases + Added more reqs	All
4/30/24	2.1	Updated Class Diagram	Jasan Manimbao
4/30/24	2.1	Final Revision	All

Table of Contents

Github Link

https://github.com/Vexana/CS401-Communication-Project

1. Purpose

This document outlines the requirements for a company's communication system.

1.1. Scope

This document will catalog the user, system, and hardware requirements for a company's communication system.

1.2. Definitions, Acronyms, Abbreviations

1.2.1. Synchronously: Communication that occurs in real-time, allowing users to send and receive messages while they are connected.

1.2.2. Asynchronously: Communication that occurs when users are not connected to the system. This means users can receive messages even when they are offline, and the messages are stored until the user connects to the system..

1.2.3. Chat privately: Conversation within the system that is exchanged exclusively between two users.

1.2.4. Chat in groups: Conversation that enables multiple users to engage in a conversation simultaneously within a single chat.

1.2.5. IT users: Individuals responsible for managing and maintaining the technical aspects of the system.

1.2.6. "Chat List View":

1.2.7. "Log View"

1.2.8. "Message View"

1.3. References

Use Case Specifications:

Use case ID: CS01
Use case name: Create New User
Relevant requirements: 3.1.3.3
Primary actor: IT users
Pre-conditions: IT user must be signed into their client
Post-conditions: Standar users and new IT users have successfully assigned login credentials.
Basic flow or main scenario:

1. The IT user logs in with their credentials into the system administration interface.
2. The IT user selects the option to assign login credentials.
3. The system verifies the IT user's ability, using their credentials, to assign login credentials to new users.
4. Once verified, the IT user enters the username and password for the new user in the company.
5. The IT user saves the login credentials.
6. The system confirms the successful assignment of the credentials.

Extensions or alternate flows:
If the entered username is already in use, the IT user receives a notification and must choose a different username.
Exceptions:
If an IT user lacks the necessary permissions, the credential assignment process cannot proceed.

Related use cases:

1. Managing user permissions
2. Accessing credentials and chat logs

Use case ID: CS02
Use case name: Login
Relevant requirements: Refer to 3.1.1.3, 3.1.1.4, 3.1.1.5, 3.1.4.3, and 3.1.4.4
Primary actor: All users
Pre-conditions: IT user must have provided authorized credentials to the user.
Post-conditions: The user is successfully logged into the system.
Basic flow or main scenario:

1. Users enter their username and password.
2. The server verifies the entered credentials.
3. Upon successful authentication, the user gains access to the system.
4. Users can start using the system features and functionalities.

Extensions or alternate flows:

1. If the entered username or password is incorrect, the system displays an error message prompting the user to re-enter the credentials.
2. After a certain number of unsuccessful login attempts, the system might lock the user's account for security reasons.
3. If the user is already logged in on another device, the system should reject the login even with correct credentials.

Exceptions:

1. If there is an unexpected system error during the login process, such as server downtime, the system should display an error message informing the user that login is temporarily unavailable and advise them to try again later.
2. If the user's account has been deactivated due any suspicious activity or a violation of terms, the system should inform the user that their account has been disabled and provide instructions for reactivation or contacting support with IT users.

Related use cases: CS03, CS05

Use case ID: CS03
Use case name: Create/Send a message
Relevant requirements: Refer to 3.1.1.2, 3.1.2.1, 3.1.4.1, and 3.1.4.2
Primary actor: All users
Pre-conditions: The user that wants to send a message is logged into the system and has clicked into a specific chat.
Post-conditions: The user sending the chat has their conversation updated with the message they send. The recipient of the message receives a notification of the message and has their conversation updated with the sent message.
Basic flow or main scenario:

1. Users click into the message field and type in a message to send.
2. Users click the "Send" button to initiate the message transaction with the server.
3. The server receives the message, logs it, and then routes the message to the recipient.
4. The recipient receives the message from the server and they are notified. Their UI is also updated to reflect the new message in the chat.

Extensions or alternate flows:
Alternate Flows:

1. If the user sending the message loses connection with the server, an error is sent back to the user and their UI does not reflect the message that was sent.
2. If the recipient is not online, the server will store the message addressed to them and wait until they come online to deliver the message.

Exceptions:

1. If there are technical issues or system errors, users may experience delays or interruptions in messaging functionality.

Related use cases: CS02

Use case ID: CS04
Use case name: IT user queries a chat log
Relevant requirements: Refer to 3.1.3.2, 3.1.3.5, and 3.1.5.2
Primary actor: IT user
Pre-conditions: The IT user has successfully logged into the system.
Post-conditions: The IT user that requested a chat log can now view the specified chat.
Basic flow or main scenario:

1. IT users click the View Logs button
2. The GUI should display a list of all users.
3. IT users should be able to select a user from the list or search for a specific one.
4. After selection, the server returns the conversation.
5. IT users review the messages exchanged within the conversation.
6. IT users may flag or take action on any messages that violate company policies or pose security risks.

Exceptions:

1. In cases where the conversation logs are unusually large or the system experiences high traffic, accessing and loading conversation data may be delayed.
2. If the system encounters a critical error while attempting to retrieve conversation logs, IT users should be notified promptly and the issue should be resolved as soon as possible to ensure continuity of monitoring activities.

Related use cases: CS02

Use case ID: CS05
Use case name: Logout
Relevant requirements: Refer to 3.1.1.6. and 3.1.4.5.
Primary actor: All users
Pre-conditions: User is logged in
Post-conditions: User’s account is logged off and the server no longer contains a connection with the user’s client.
Basic flow or main scenario:

1. A user is logged into their respective profile.
2. The user has a logout button in their UI.
3. The user clicks the logout button.
4. A message is sent to the server to request a logout.
5. The server removes that specific user from the online users list.
6. The client should display the login menu so a user can log into the system.

Extensions or alternate flows: User closes the application.
Alternate Flow:

1. A logout request is sent to the server from a user who is not online - the server should disregard.

Exceptions:

1. User does not have a logout button
2. User chooses to logout but the system does not respond.

Related use cases: CS02

Use case ID: CS06
Use case name: Create a new chat
Relevant requirements: Refer to 3.1.5.2.

Primary actor: All users
Pre-conditions: The user is logged in.
Post-conditions: The user can now send messages within the chat. An associated chat file has been created on the server to log the conversation.
Basic flow or main scenario:

1. User clicks a button to create a new chat.
2. User specifies the members of the chat.
3. The server checks to see if there is an existing chat with those members
4. A chat file is created on the server.
5. The creating user's UI updates to show the chat window.
6. The user can now send messages within the chat.

Extensions or alternate flows:
Alternate flows:

1. If a chat being created already exists, the user receives an error stating "Chat already exists."

Exceptions:

2. The user loses connection with the system while trying to create a new chat. The user will have to reconnect and try to create the chat again.

Related use cases: CS02, CS03

Use case ID: CS07
Use case name: Managing credentials
Relevant requirements: Refer to 3.1.3.2, 3.1.3.3, 3.1.3.4, and 3.1.5.3
Primary actor: IT users
Pre-conditions: The IT user is logged into the system, There must be Users in the system to update.
Post-conditions: IT users have successfully managed and updated the credentials of standard users and other IT users.
Basic flow or main scenario:

1. IT users navigate to the user management section of the system.
2. IT users select the option to manage credentials.
3. The system displays a list of all user credentials available for management.
4. IT users select a specific user from the list to manage their credentials.
5. IT users can view, edit, or reset the username and password for the selected user.
6. IT users save the changes made to the credentials.
7. The system confirms the successful update of the user's credentials.

Extensions or alternate flows: N/A
Exceptions:

1. If there is a system error during the credential management process, an error message should be displayed to the IT user.

Related use cases: CS02

Use case ID: CS08
Use case name: Set user status
Relevant requirements: Refer to 3.1.1.4, 3.1.4.5, 3.1.4.8, and 3.1.6.7
Primary actor: All users
Pre-conditions: The user is logged into the system.
Post-conditions: The user's status is updated and visible to other users.

Basic flow or main scenario:

1. Users log in to their accounts.
2. The user's status automatically changes from offline to online.
3. Other users can now see the status next to the user's name.

Extensions or alternate flows:

1. N/A

Exceptions:

1. If there is an unexpected system error or a server shutdown, the user's status may not be updated correctly. An error message should be displayed to the user.

Related use cases: CS02, CS05

Use case ID: CS09
Use case name: GUI updates to show new message notification
Relevant requirements: Refer to 3.1.4.6, 3.1.4.7, 3.1.2.4, and 3.1.1.5
Primary actor: All users
Pre-conditions: The user is logged into the system.
Post-conditions: The user’s GUI should reflect that there is an unread message..
Basic flow or main scenario:

1. Users are actively logged into their respective profiles.
2. Another user sends a message to the logged-in user.
3. If the recipient is not currently viewing the chat in which they are receiving a message, then a notification icon should be visible next to the conversation where the message was sent.
4. Users receive a notification indicating there is a new message.
5. Users can click on the notification to view the new message within the chat interface.
6. After viewing, the notification indicator is cleared from the GUI.

Extensions or alternate flows:

1. While the user is offline, new messages are stored on the server.
2. Upon logging in again, the stored new messages are delivered to the user, and a GUI notification is displayed to indicate the presence of these new messages.

Exceptions:

1. If there is an unexpected system error preventing the display of the GUI notification for new messages, an error message should be displayed to the user.
2. In case of server downtime, the GUI notification for new messages might be delayed.

Related use cases: CS10

Use case ID: CS10
Use case name: Storing a message on the server
Relevant requirements: Refer to 3.1.1.5, 3.1.2.1, 3.1.2.4, 3.1.6.3, and 3.1.6.4
Primary actor: Server
Pre-conditions: A user sends a message to another user who is currently offline.
Post-conditions: The message is written to the associated chat file on the server retrieval once the user logs in again.
Basic flow or main scenario:

1. A user sends a message to another user who is currently offline.
2. The system detects that the recipient is offline.
3. The system stores the new message on the server associated with the recipient’s account.

Extensions or alternate flows:

1. When the offline user logs back in, the stored messages are retrieved from the server and delivered to the user.

Exceptions:

1. During server maintenance or downtime, the message storage functionality may be temporarily unavailable. Messages might not be stored or retrieved, resulting in a "Not sent" notification for users attempting to send messages.

Related use cases: CS03

Use case ID: CS11
Use case name: Searching for a chat
Relevant requirements: Refer to 3.1.2.3, 3.1.2.6, 3.1.4.6, and 3.1.4.7
Primary actor: All users
Pre-conditions: The user is logged into the system and is on the main chat interface.
Post-conditions: The user successfully finds and accesses the desired chat.
Basic flow or main scenario:

2. The user is on the main chat interface and wishes to find a specific chat.
3. The user initiates the search function by clicking on the search bar.
4. The system displays a list of chat results matching the search query in real-time.
5. The user selects the desired chat from the search results.
6. The system navigates the user to the selected chat, displaying the chat history and allowing the user to continue the conversation.

Extensions or alternate flows:

1. If the search query does not match any chat names or participants, the system should display a message indicating no results found.

Exceptions:

1. If there is an unexpected system error during the search process, an error message should be displayed to the user, and the search functionality might be temporarily unavailable.

Related use cases: N/A

Use case ID: CS12
Use case name: Clicking into a specific chat in the chat list
Relevant requirements: Refer to 3.1.1.2, 3.1.2.1, 3.1.4.1, and 3.1.4.2
Primary actor: All users
Pre-conditions: The user is logged into the system and is currently viewing their chats.
Post-conditions: The user should be in the “Chat List View”.
Basic flow or main scenario:

2. The user is on the main chat interface and wishes to find a specific chat.
3. The user clicks on a chat from their chat list
4. The GUI should update and populate a new view with the messages of the selected chat.. The top of the chat should specify the participants. There should be a text-entry field at the bottom of the window for the user to type messages to the chat.
5. As new messages are being sent, the GUI should update.

Extensions or alternate flows: N/A
Exceptions:

1. If there is a problem with the user's network connection preventing the chat list or messages from loading.
2. If there is an unexpected system error preventing the chat list or messages from loading.

Related use cases: CS03

Use case ID: CS13
Use case name: Clicking the “View Chats” Button
Relevant requirements: Refer to 3.1.1.2, 3.1.2.1, 3.1.4.1, and 3.1.4.2
Primary actor: All users
Pre-conditions: The user should be logged into the system and not currently viewing the chat list.
Post-conditions: The user should be able to see a list of all of their chats
Basic flow or main scenario:

1. The user is currently viewing another part of the system e.g. “Chat List View” or “log view”
2. The user clicks the “View Chats” button.
3. The GUI should update to display a list of all the chats a user is currently a member of. There should be a button that allows the user to create a new chat and a button that allows the user to search for a specific chat.

Extensions or alternate flows: N/A
Exceptions:

1. If there is a problem with the user's network connection preventing the chat list from loading.
2. If there is an unexpected system error preventing the chat list from loading.

Related use cases: CS02, CS03

Use case ID: CS14
Use case name: Clicking the “View Logs” Button
Relevant requirements: Refer to 3.1.1.2, 3.1.2.1, 3.1.4.1, and 3.1.4.2
Primary actor: IT User
Pre-conditions: The user should be logged into the system and not currently in the “Log View”.
Post-conditions: The user should be able to see a list of all of their chats and be able to select a user to view the chats for.
Basic flow or main scenario:

1. The IT user clicks on the “View Logs” button in their GUI.
2. The GUI should update to display a list of all the users in the system.
3. The IT User should be able to select a user from this list to display the chats they’re a part of.

Extensions or alternate flows:
Alternate Flows:

1. If the user is already in the “Log View”, do nothing on button click.
2. If the IT user has already clicked the “View Logs” button and selected a user’s chat’s to display, then clicking the “View Logs” button should clear the current view and bring the IT user back to the “Log View.”

Exceptions:

1. If there is an unexpected system error preventing the logs from being displayed.
2. If there are no logs available for the selected user.
3. If the IT user selects an invalid or unauthorized user to view logs for.

Related use cases: CS02, CS03

Use case ID: CS15
Use case name: Clicking a user in the “Log View”
Relevant requirements: Refer to 3.1.3.3, 3.1.4.8, 3.1.5.5, and 3.1.5.6

Primary actor: IT User
Pre-conditions: The user should be logged into the system and not currently in the “Log View”.
Post-conditions: The user should be able to see a list of all of their chats and be able to select a user to view the chats for.
Basic flow or main scenario:

1. The IT user clicks a specific user in the “Log View”
2. The client sends a request to the server for the chats of the specified user.
3. The server returns a list of all of the chats that specific user to the client.
4. The GUI then displays a selectable list of all the chats for the specified user.

Extensions or alternate flows:

1. N/A

Exceptions:

1. In the case where the connection between the IT user’s client and the server is lost. Do nothing.

Related use cases: CS14

Use case ID: CS16
Use case name: Pinning a chat
Relevant requirements: Refer to 3.1.2.7 and 3.1.4.10
Primary actor: All users
Pre-conditions: The user should be logged into the system and the user should not already have the max amount of pinned chats. (3)
Post-conditions: The user’s pinned chats should now reflect the newly pinned chat. The “Chat List View” should display the newly pinned chat at the top of the view.
Basic flow or main scenario:

1. The user needs to be in the “Chat List View.”
2. Upon right clicking a specified chat, the user selects the option to pin the chat.
3. After pinned, the GUI should update reflecting the addition to the pinned chat.

Extensions or alternate flows:
Alternate Flows:

1. If a user already has the maximum number of pinned chats, display a message that notifies the user they’ve reached the limit of pinned chats.

Exceptions:

1. If the specified chat to pin is not found or has been deleted.
2. If there is an unexpected system error preventing the chat from being pinned.
3. If the specified chat is already pinned.

Related use cases:CS02, CS13

Use case ID: CS17
Use case name: Removing a Pinned a chat
Relevant requirements: Refer to 3.1.2.8 and 3.1.4.11
Primary actor: All users
Pre-conditions: The user should be logged into the system and the user must have at least 1 pinned chat.
Post-conditions: The user’s pinned chats should now reflect the newly unpinned chat. The “Chat List View” should update to reflect the changes.
Basic flow or main scenario:

1. The user needs to be in the “Chat List View.”
2. Upon right clicking a specified chat, the user selects the option to unpin the chat.
3. After unpinning a chat, the GUI should update reflecting the removal of the pinned chat.

Extensions or alternate flows:

1. If a user already has the maximum number of pinned chats, display a message that notifies the user they've reached the limit of pinned chats.

Exceptions:

1. If the specified chat is already unpinned.
2. If there is an unexpected system error preventing the chat from being unpinned.

Related use cases:CS13, CS16

2. Overall Description

2.1. Product Perspective

The communication system strives to be user-friendly and reliable. It is designed to work seamlessly with different devices and systems, ensuring ease of use for everyone. Security is a top priority, guaranteeing that messages are kept safe and private. The system is also closely monitored to quickly address any issues, maintaining smooth operation for all users.

2.2. Product Architecture

2.2.1. Standard user module: It is designed to provide a user-friendly interface for non-technical users, allowing them to exchange text messages, view chats on a central homepage, and receive notifications for missed messages.

2.2.2. IT User module:It is designed to equip Information Technology (IT) professionals with exclusive capabilities, offering specialized tools for managing user accounts, overseeing system operations, and ensuring security.

2.2.3. User GUI module:This module ensures a user-friendly interface when interacting with the system, making actions such as logging, sending messages, and viewing chats easy.

2.2.4. IT GUI module: It is a user interface designed specifically for IT users, incorporating all standard UI elements. It features buttons for various functions such as querying log files for chats, viewing and modifying user information (Usernames and permissions), and creating new users within the system.

2.2.5. Server module: Responsible for managing client-server communication, user authentication, message routing, and logging.

2.3. Product Functionality/Features

2.3.1. Chat functionality

1. Synchronous chat: Enable real-time messaging between all users for instant communication.
2. Asynchronous chat: Allow users to send and receive messages even when the recipient is offline, with messages delivered once the recipient reconnects.
3. Private Chat: Facilitates one-on-one conversations between users.
4. Group Chat: Allows multiple users to participate in a single conversation.

2.3.2. Message formatting Supports only text in user communication.

2.3.3. User Management

1. User Registration and Authentication: Provide a registration process for new users and secure authentication mechanisms to verify user identities.
2. User Profiles: Allow users to create and customize their profiles with profile pictures, status messages, and contact information.

2.3.4. Administration and Monitoring

1. User Activity Monitoring: Allow IT users to monitor user activity, track login/logout times, and view user session information.
2. Content Moderation: Give IT users tools to manage user content, like flagging inappropriate messages and handling user bans or suspensions.
3. System Health Monitoring: Monitor system performance, resource utilization, and network traffic to identify and address potential issues proactively.

2.4. Constraints

2.4.1. The system needs to be permanently connected to an internet connection.

2.4.2. The system only accepts text format for communication among all users.

2.4.3. The system only accepts text characters that fall under the ASCII table.

2.4.4. Scalability: The system should be scalable to accommodate future growth in the number of users and messages without needing major changes.

2.4.5. Performance Limitations: The system must handle a large volume of messages and users without sacrificing performance or responsiveness.

2.4.6. Platform Compatibility: Ensuring compatibility across different operating systems (Windows, macOS, Linux) and devices (desktops, laptops, mobile devices) may pose challenges.

2.5. Assumptions and Dependencies

Assumptions:

1. The project assumes that the organization's network infrastructure is reliable and can support the demands of the communication system without significant interruptions.
2. The project assumes that users will readily adopt the new communication system and integrate it into their daily workflows.
3. The project assumes that users will receive adequate training and support to effectively utilize the features and functionalities of the communication system.
4. Assumes that the communication system is compatible with various devices and operating systems commonly used by employees within the organization.
5. Assumes that the data transmitted and stored within the communication system will remain accurate, consistent, and free from corruption or loss.

Dependencies:

1. The project may depend on user feedback and testing to identify areas for improvement and refine the features and usability of the communication system over time.
2. The project may depend on compliance with regulatory requirements, such as data protection laws and industry standards, which may impact the design and implementation of the communication system.
3. The project may depend on the network infrastructure since the organization's network needs to provide sufficient bandwidth, reliability, and security for the transmission of messages and data within the communication system.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1. All users on the system should be able to request a password reset.
3.1.1.2. All messages should show the username, date, and time that a message was sent.
3.1.1.3. All user logins should be limited to one device at a time.
3.1.1.4. All users should be able to login and logout anytime.
3.1.1.5. All users should receive notifications when a new message comes in while they're logged in.

3.1.2. Standard User Module Requirements:

3.1.2.1. Users should be able to send messages.
3.1.2.2. Files, such as pictures, audios, and videos, etc. will not be allowed to be sent.
3.1.2.3. Users should be able to view their chats on a home page or lobby.
3.1.2.4. All users should receive message notifications for any missed messages while they were not signed in.
3.1.2.5. Users should be able to create a new private or group chat.
3.1.2.6. A user cannot have more than one private chat with the same user. Similarly, there can only be one instance of a group with the same participants.
3.1.2.7. Users should be able to "pin" chats so that they remain at the top of their chats list.
3.1.2.8. Users should be able to remove pinned chats from the current list of pinned chats.

3.1.3. IT User Module Requirements:

3.1.3.1. IT users should share all functionality of a standard user.

3.1.3.2. IT users must assign login credentials for a standard user and new IT users.

3.1.3.3. IT users should be able to look up all credentials and view all chat logs between users.

3.1.3.4. IT users should be able to change the account permissions of a standard user to an IT user.

3.1.3.5. IT users should not be able to modify existing conversations in the system.

3.1.4. Users GUI Module Requirements:

3.1.4.1. When inside a specific chat, there should be a "Send" button that allows the users to send their message.
3.1.4.2. When inside a specific chat, there should be a text box where the user types in the desired message.
3.1.4.3. When the client program is loaded, there should be a box for the users to type their login credentials such as username and password.
3.1.4.4. A "Login" button that allows the user to log in should be present.
3.1.4.5. Once a user successfully logs into the system, their status should automatically change to "Online." This update status should be visible to all user's contacts.
3.1.4.6. A "Logout" button that signs a user out of the system should be present.
3.1.4.7. When the user logs into the system they should see a list of the chats they are a

part of.

3.1.4.8. Once logged in, users should be able to see the status of each of their system contacts.The status should indicate whether each contact is 'Online' or 'Offline'.

3.1.4.9. There should be a button that allows the user to search for a particular chat.

3.1.4.10. When a user clicks on a specific chat, the GUI should display the chat in the full client window. This is known as the “Chat List View”

3.1.4.11. There should be a button to create a new chat dialogue with another user.

3.1.4.12. From the “Chat List View” view, a user should be able to right click a specific chat and “Pin” it.

3.1.4.13. From the “Chat List View” view, a user should be able to right click a specific chat and “Unpin” it.

3.1.5. IT GUI Module Requirements:

3.1.5.1. The IT GUI should show all of the same UI elements as referenced in section 3.1.4.*

3.1.5.2. There should be a button that allows the IT users to query the log file for a specific chat.

3.1.5.3. There should be a button that allows the IT users to view and modify user information such as usernames, passwords, and account permissions (standard or IT).

3.1.5.4. There should be a button that allows the IT users to create a new user on the system.

3.1.5.5. The IT user should be able to click a user in the “Log View” and the GUI should display all the chats that user is a member of. This should display the “Specified User Chat List View.”

3.1.5.6. The IT user should be able to click a specific chat in the “Specified User Chat List View” and the GUI should display the chat in the standard “Chat List View”

3.1.6. Server Module Requirements:

3.1.6.1. There should not be a database used.

3.1.6.2. Must be able to handle multiple client requests

3.1.6.3. Must save messages for a recipient who is offline.

3.1.6.4. Must send all undelivered messages to a user when they log into the system.

3.1.6.5. Must store logged chats from all private/group chats in .txt format.

3.1.6.6. Must have a list of user’s credentials for authentication requests.

3.1.6.7. Upon successful authentication and login of a user, the server should send a signal to update the user's status to 'Online'.

3.2. External Interface Requirements

3.2.1. The system must provide a user interface to the users to access the application.

3.2.2. The system must have a user interface that will allow users to send and receive messages.

3.2.3. The system should work with outside notification systems like email or push notifications. This way, users can get quick alerts about important events, new messages, or updates as they happen.

3.3. Internal Interface Requirements

3.3.1. The system needs a strong process to check if users are who they say they are. This means making sure users have the right permissions to use the system and access its features.

3.3.2. The system needs a way to keep track of what users do, system events, and errors. This helps with fixing problems, keeping an eye on things, and following rules.

3.3.3. The system must store and record messages between users.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1. A list of company employees should be provided to the initial IT users to create user accounts.
4.1.2. IT users are the only authorized users to create new users on the system.
4.1.3. System should require users to login to access their profiles.
4.1.4. The system should check for matching credentials before allowing a user to interact with the application..

4.2. Environmental Requirements

4.2.1. Must be Java based
4.2.2. The server must always be accessible for all users.
4.2.3. The system must require only the installation of the program itself to run.
4.2.4. The system must run on systems running the Windows 10 operating system.

4.3. Performance Requirements

4.3.1. The system should be able to handle a large volume of users and simultaneous messages.
4.3.2. A user should be able to receive multiple simultaneous messages.

5. UML Diagrams

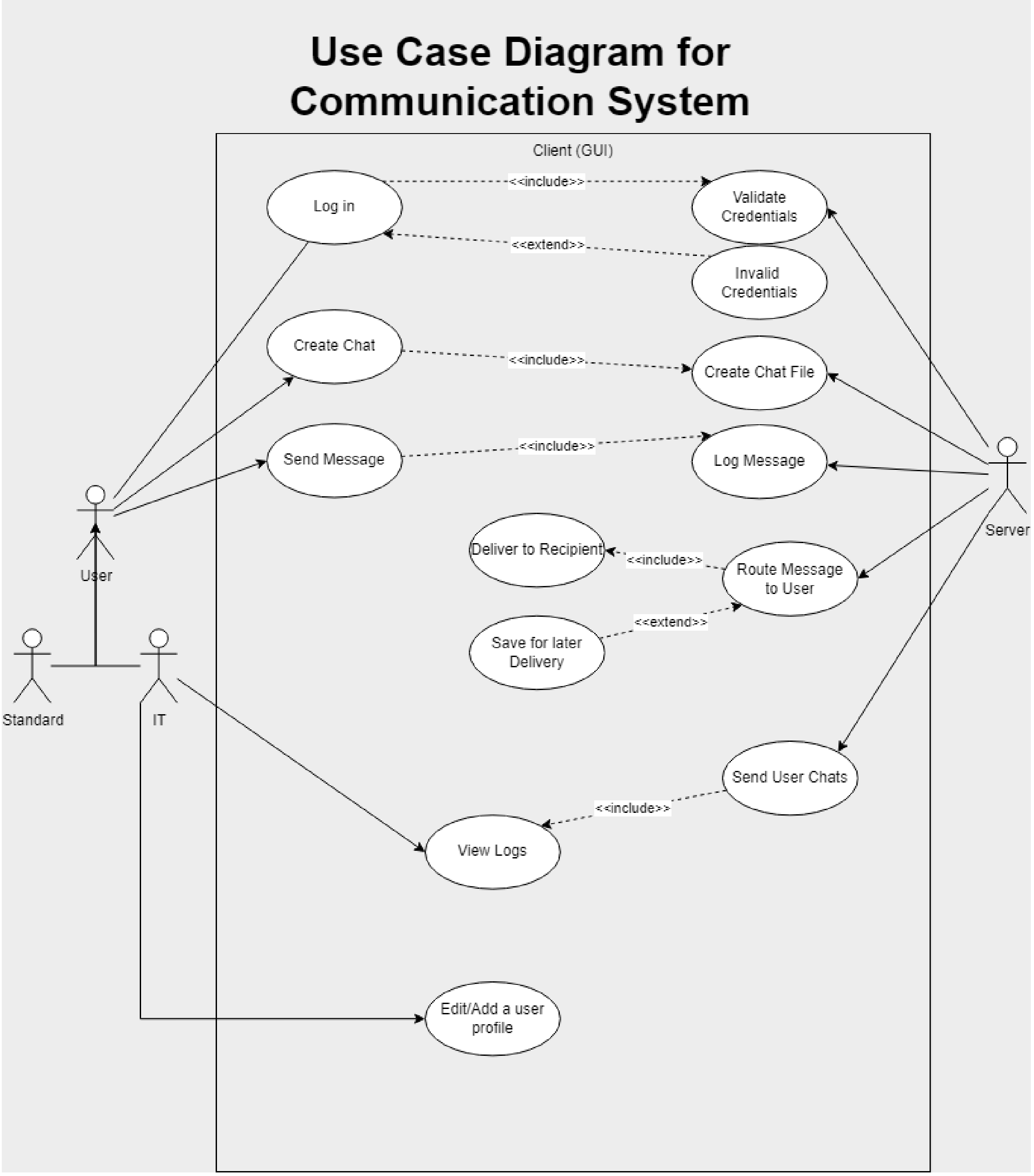

Figure 5.1 Use Case Diagram

Sequence diagrams

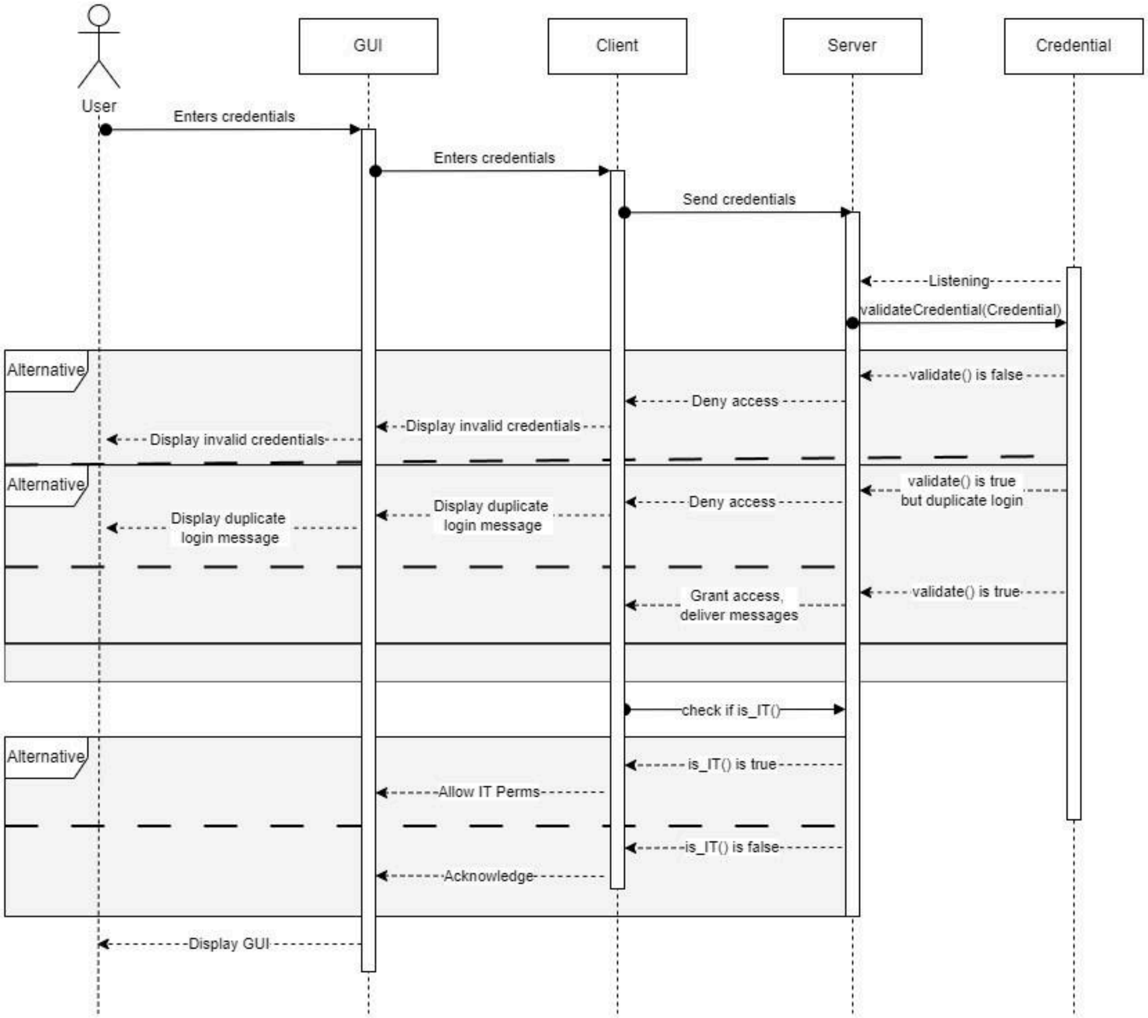

Figure 5.2 User Logs In - Sequence Diagram

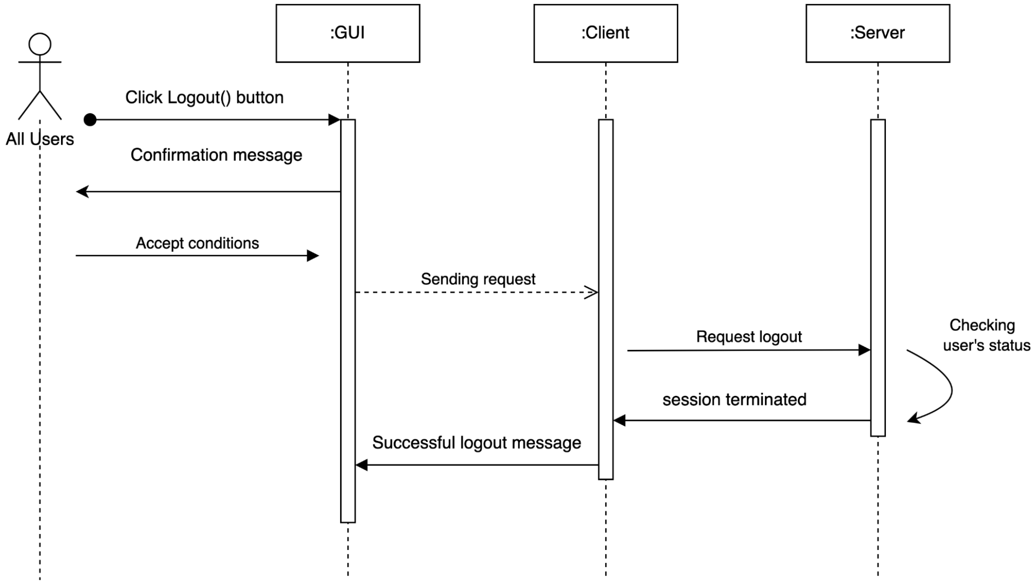

Figure 5.3 User Logs Out - Sequence Diagram

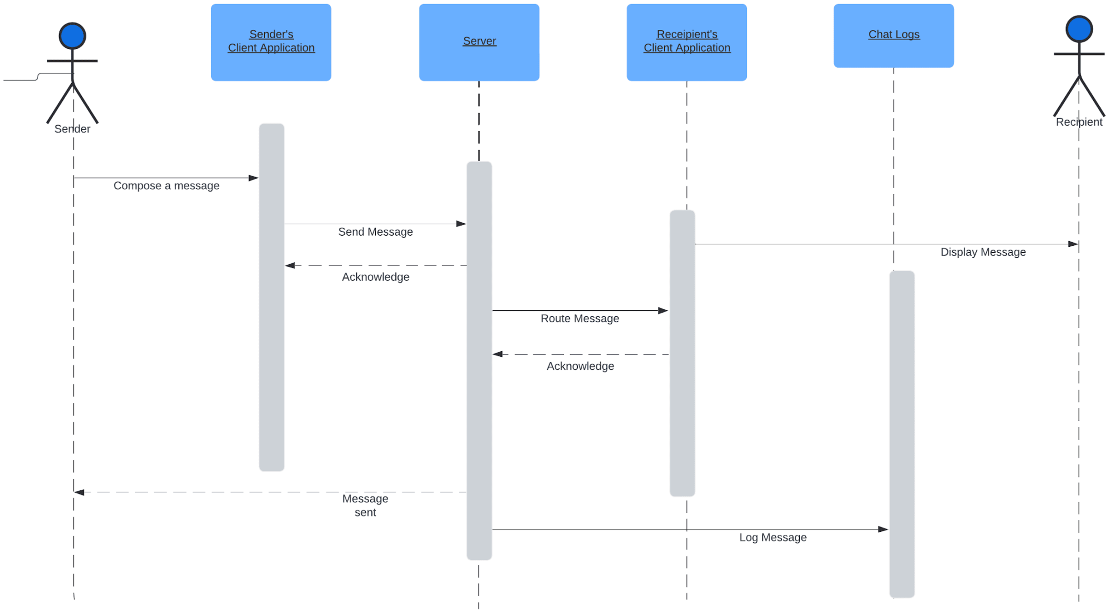

Figure 5.4 Compose a Message - Sequence Diagram

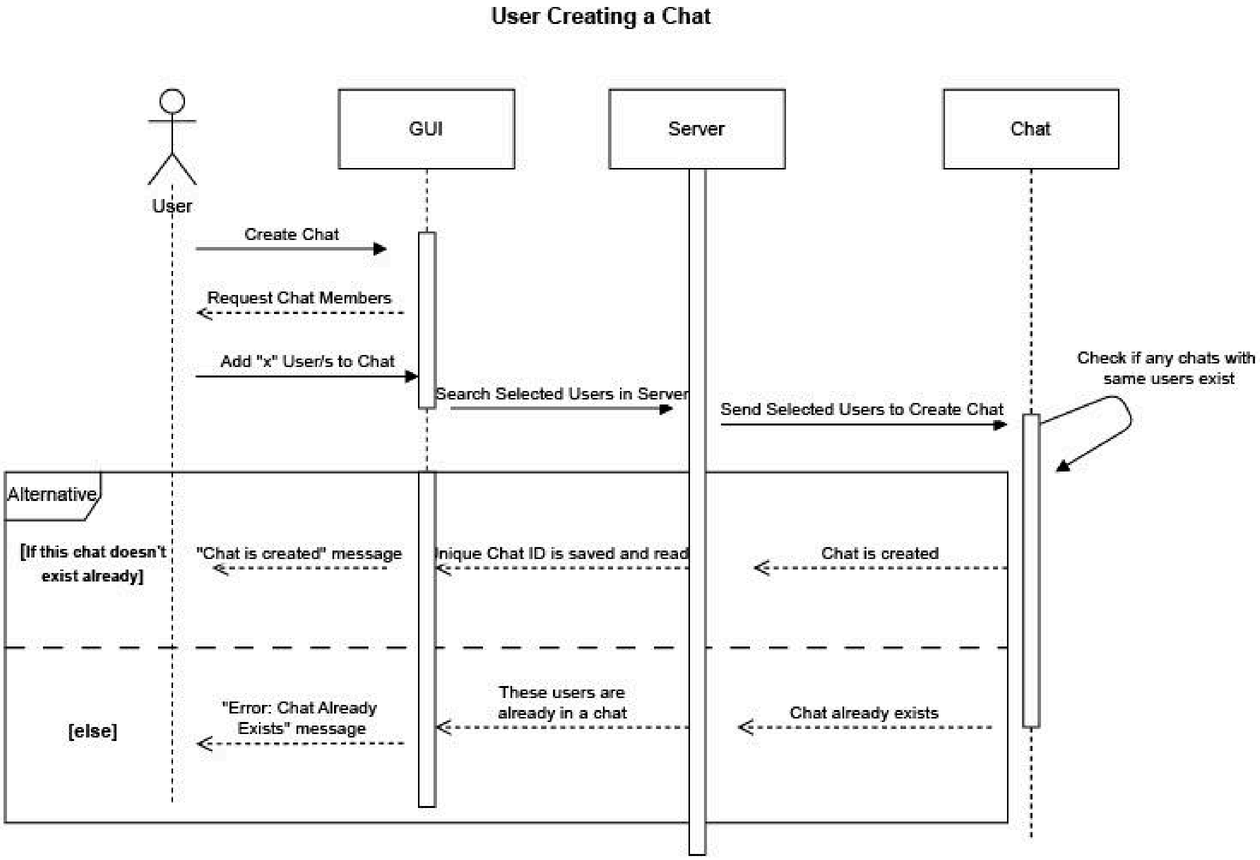
User Creating a Chat
User
GUI
Server
Chat
Create Chat
Request Chat Members
Add "x" User/s to Chat
Search Selected Users in Server
Send Selected Users to Create Chat
Check if any chats with same users exist
Alternative
[If this chat doesn't exist already]
"Chat is created" message
Inique Chat ID is saved and read
Chat is created
[else]
"Error: Chat Already Exists" message
These users are already in a chat
Chat already exists

Figure 5.5 User Creates a Chat Room - Sequence Diagram

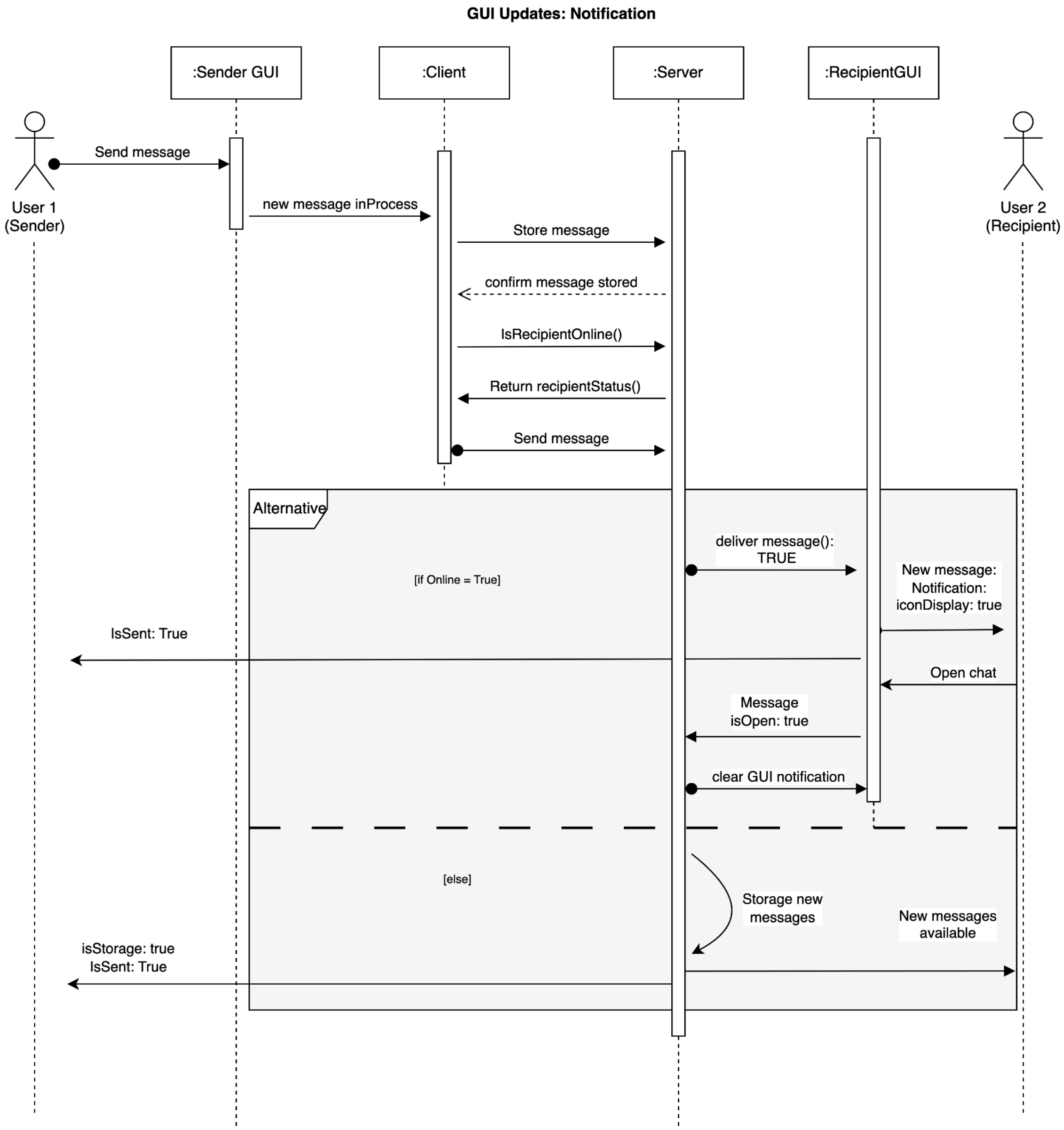

Figure 5.6 - GUI updates to show new message notification - Sequence Diagram

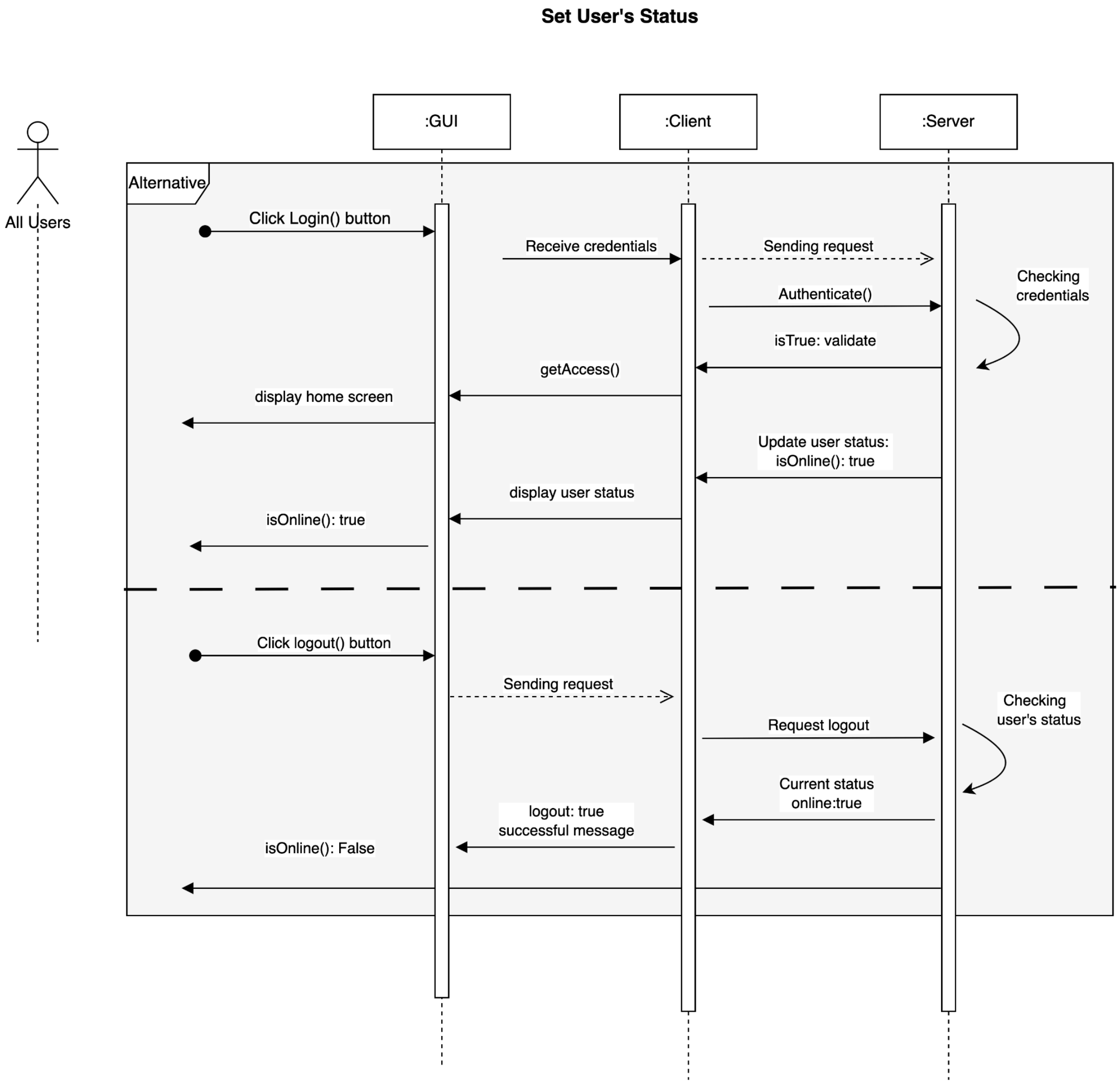

Figure 5.7 - Set User’s Status - Sequence Diagram

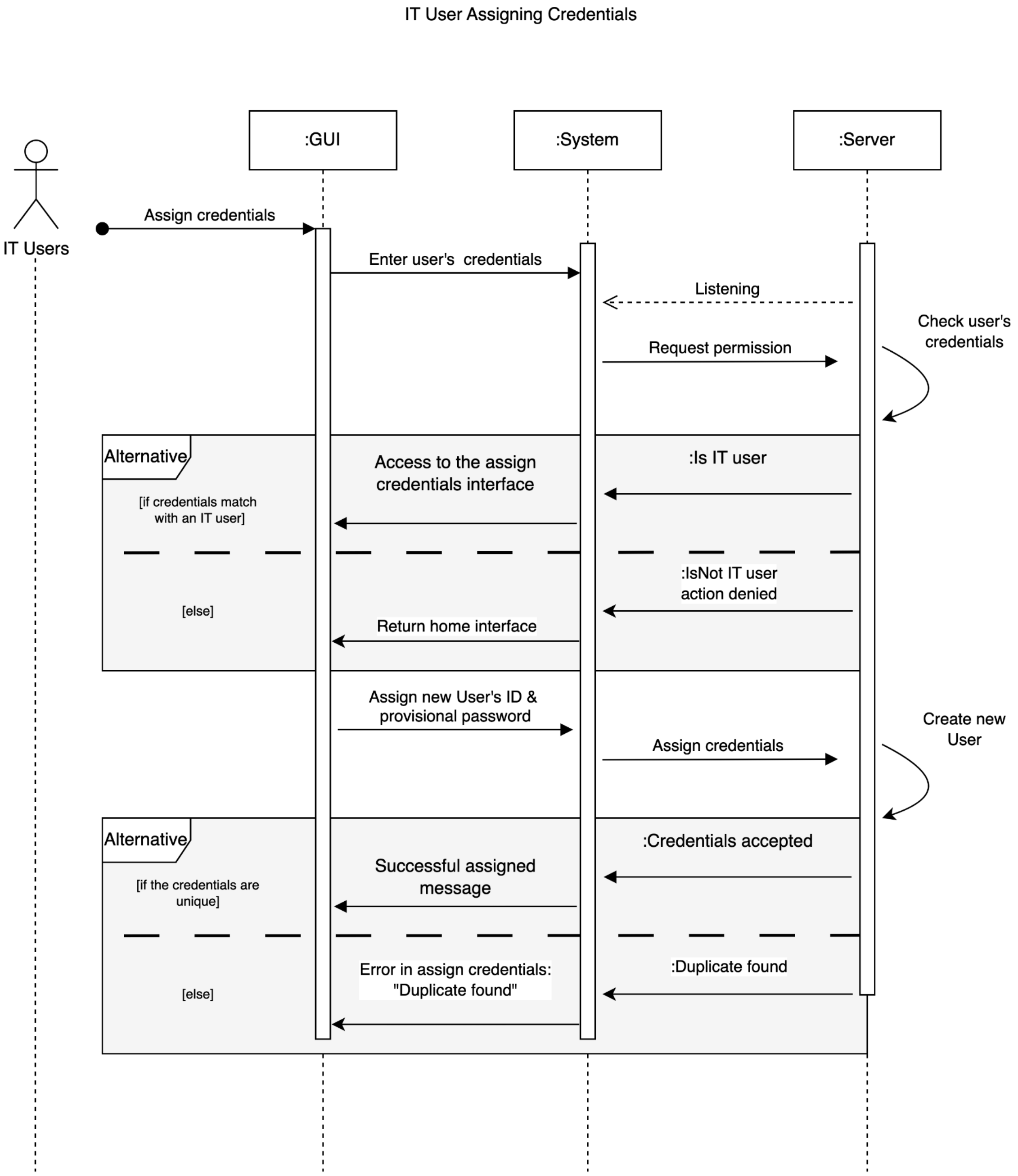
IT User Assigning Credentials
:GUI
:System
:Server
IT Users
Assign credentials
Enter user's credentials
Listening
Request permission
Check user's credentials
Alternative
[if credentials match with an IT user]
Access to the assign credentials interface
:Is IT user
[else]
:IsNot IT user action denied
Return home interface
Assign new User's ID & provisional password
Assign credentials
Create new User
Alternative
[if the credentials are unique]
Successful assigned message
:Credentials accepted
[else]
Error in assign credentials: "Duplicate found"
:Duplicate found

Figure 5.8 IT Assigns Credentials - Sequence Diagram

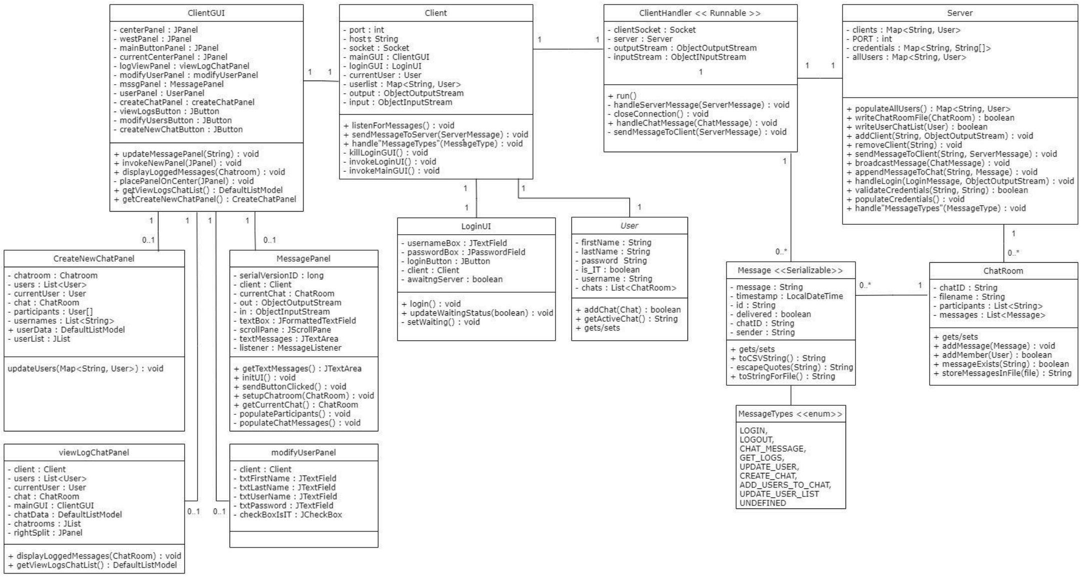
ClientGUI
- centerPanel : JPanel
- westPanel : JPanel
- mainButtonPanel : JPanel
- currentCenterPanel : JPanel
- logViewPanel : viewLogChatPanel
- modifyUserPanel : modifyUserPanel
- mssgPanel : MessagePanel
- userPanel : UserPanel
- createChatPanel : createChatPanel
- viewLogsButton : JButton
- modifyUsersButton : JButton
- createNewChatButton : JButton
+ updateMessagePanel(String) : void
+ invokeNewPanel(JPanel) : void
+ displayLoggedMessages(Chatroom) : void
- placePanelOnCenter(JPanel) : void
+ getViewLogsChatList() : DefaultListModel
+ getCreateNewChatPanel() : CreateChatPanel
1
1
Client
- port : int
- host : String
- socket : Socket
- mainGUI : ClientGUI
- loginGUI : LoginUI
- currentUser : User
- userlist : Map<String, User>
- output : ObjectOutputStream
- input : ObjectInputStream
+ listenForMessages() : void
+ sendMessageToServer(ServerMessage) : void
+ handle"MessageTypes"(MessageType) : void
- killLoginGUI() : void
- invokeLoginUI() : void
- invokeMainGUI() : void
1
1
ClientHandler << Runnable >>
- clientSocket : Socket
- server : Server
- outputStream : ObjectOutputStream
- inputStream : ObjectINputStream
1
+ run()
- handleServerMessage(ServerMessage) : void
- closeConnection() : void
+ handleChatMessage(ChatMessage) : void
- sendMessageToClient(ServerMessage) : void
1
1
Server
- clients : Map<String, User>
- PORT : int
- credentials : Map<String, String[]>
- allUsers : Map<String, User>
+ populateAllUsers() : Map<String, User>
+ writeChatRoomFile(ChatRoom) : boolean
+ writeUserChatList(User) : boolean
+ addClient(String, ObjectOutputStream) : void
+ removeClient(String) : void
+ sendMessageToClient(String, ServerMessage) : void
+ broadcastMessage(ChatMessage) : void
+ appendMessageToChat(String, Message) : void
+ handleLogin(LoginMessage, ObjectOutputStream) : void
+ validateCredentials(String, String) : boolean
+ populateCredentials() : void
+ handle"MessageTypes"(MessageType) : void
1
1
1
1
1
0..1
0..1
1
1
1
1
1
0..*
1
0..*
CreateNewChatPanel
- chatroom : Chatroom
- users : List<User>
- currentUser : User
- chat : ChatRoom
- participants : User[]
- usernames : List<String>
+ userData : DefaultListModel
- userList : JList
updateUsers(Map<String, User>) : void
MessagePanel
- serialVersionID : long
- client : Client
- currentChat : ChatRoom
- out : ObjectOutputStream
- in : ObjectInputStream
- textBox : JFormattedTextField
- scrollPane : JScrollPane
- textMessages : JTextArea
- listener : MessageListener
+ getTextMessages() : JTextArea
+ initUI() : void
+ sendButtonClicked() : void
+ setupChatroom(ChatRoom) : void
+ getCurrentChat() : ChatRoom
- populateParticipants() : void
- populateChatMessages() : void
LoginUI
- usernameBox : JTextField
- passwordBox : JPasswordField
- loginButton : JButton
- client : Client
- awaitngServer : boolean
+ login() : void
+ updateWaitingStatus(boolean) : void
- setWaiting() : void
User
- firstName : String
- lastName : String
- password String
- is_IT : boolean
- username : String
- chats : List<ChatRoom>
+ addChat(Chat) : boolean
+ getActiveChat() : String
+ gets/sets
Message <<Serializable>>
- message : String
- timestamp : LocalDateTime
- id : String
- delivered : boolean
- chatID : String
- sender : String
+ gets/sets
+ toCSVString() : String
- escapeQuotes(String) : String
+ toStringForFile() : String
0..*
1
ChatRoom
- chatID : String
- filename : String
- participants : List<String>
- messages : List<Message>
+ gets/sets
+ addMessage(Message) : void
+ addMember(User) : boolean
+ messageExists(String) : boolean
+ storeMessagesInFile(file) : String
MessageTypes <<enum>>
LOGIN,
LOGOUT,
CHAT_MESSAGE,
GET_LOGS,
UPDATE_USER,
CREATE_CHAT,
ADD_USERS_TO_CHAT,
UPDATE_USER_LIST
UNDEFINED
viewLogChatPanel
- client : Client
- users : List<User>
- currentUser : User
- chat : ChatRoom
- mainGUI : ClientGUI
- chatData : DefaultListModel
- chatrooms : JList
- rightSplit : JPanel
+ displayLoggedMessages(ChatRoom) : void
+ getViewLogsChatList() : DefaultListModel
0..1
0..1
modifyUserPanel
- client : Client
- txtFirstName : JTextField
- txtLastName : JTextField
- txtUserName : JTextField
- txtPassword : JTextField
- checkBoxIsIT : JCheckBox

Figure 5.9 Class Diagrams