CS 401-01

Summer 2022

Software Engineering

Group 7: Communication Systems

Requirements Document

Authors: Ayumi Toki

Emanuel Baca

Revision History

Date	Revision	Description	Author
06/21/2022	1.0	Initial Version	Group 7
06/22/2022	1.1	Addition of Meeting Notes	Emanuel Baca
07/06/2022	1.3	Adding setters and getters	Sandra Torres
07/11/2022	2.0	Now working on Design phase.	Group 7

Table of Contents

1.	**Purpose**	**……**	**4**
2.	**Overall Description**	**……**	**5**
3.	**Specific Requirements**	**……**	**6**
4.	**Non-Functional Requirements**	**……**	**7**
5.	**UML: Use Case Diagram**	**……**	**8**
6.	**Meeting Minutes Log**	**……**	**9**
7.	**Team Schedule & Github Link**	**……**	**12**

1. Purpose

1.1 Scope

This document will catalog the users and the system for a Java platform.

1.2 Definitions, Acronyms, Abbreviations

Java = programming language
IT = information technology
UML = Unified Modeling Language

1.3 References

SRS_Template Word document - Assignment 2
UML Use Case Diagrams

1.4 Overview

A communications system for a large organization running on a Java platform.

2. Overall Description

2.1 Product Perspective

2.1.1 Initial Login module for valid credentials

2.1.2 Group chat module

2.1.3 Chat log only module

2.1.4 Search for users module

2.1.5 Log off module

2.2 Product Architecture

2.2.1 The system will be organized into 2 different modules: the Standard Employee module and the IT Administrator module.

2.3 Product Functionality/Features

2.3.1 The high-level features of the system are the following: send messages within employees in organization, messages can be sent synchronously and asynchronously, create a group chat, search for employees in directory, log all chats in a log.txt file, and create new employees.

2.4 Constraints

2.4.1 The application will only work within an organization and is not accessible from any outside source, in other words employees can only message others within their organization.

2.4.2 Valid credentials must be entered in order to launch the application.

2.5 Assumptions and Dependencies

2.5.1 It is assumed that the maximum number of employees is “infinite” therefore the dependency is on the client's physical hardware capabilities.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Common Requirements:

3.1.1.1 Users should be employees of the large organization.
Users should have their id, username, and password.

3.2 External Interface Requirements

3.2.1 The system must provide an interface where users can navigate group chats. The group chats are visible to the users with IT's permission.
The interface should have the employee id, username, and action. Action is for the employee to create a group chat, add a user to the chat, and remove the user.
Chat log and history will be saved in a text file. After new group chats are created or old ones removed, the file will be overwritten with newer updates.

3.3 Internal Interface Requirements

3.3.1 The system must process the chat log where conversations between employees are held. IT has access to all group chats and the chat log history.

4. Non-Functional Requirements

4.1 Security and Privacy Requirements

4.1.1 The User must have valid credentials in order to log into the application.

4.1.2 Only IT Administrators can create new employees and have access to chat logs.

4.2 Environmental Requirements

4.2.1 System must have Java installed in order to use the application.

4.2.2 Server Application must be deployed and running before using Client Application

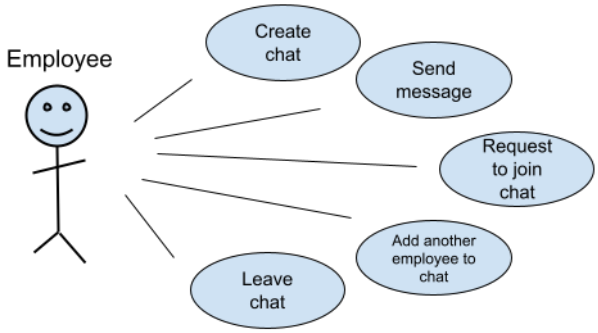
4.2.3 Systems must have a valid IP address in order to communicate via TCP/IP

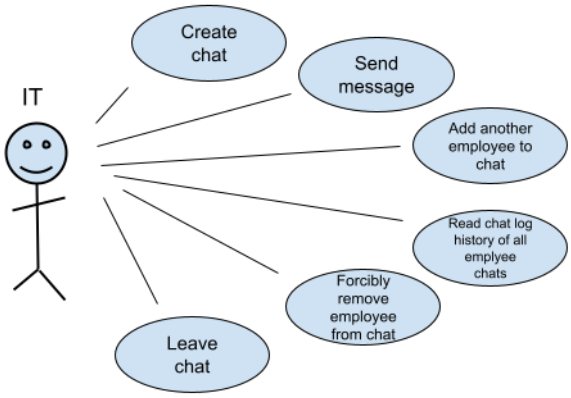
4.3 Performance Requirements

4.3.1 System must be able to have sufficient hard drive space in order to log all chats to a text file where all of the data is stored.

5. UML: Use Case Diagram

5.1 Use Cases



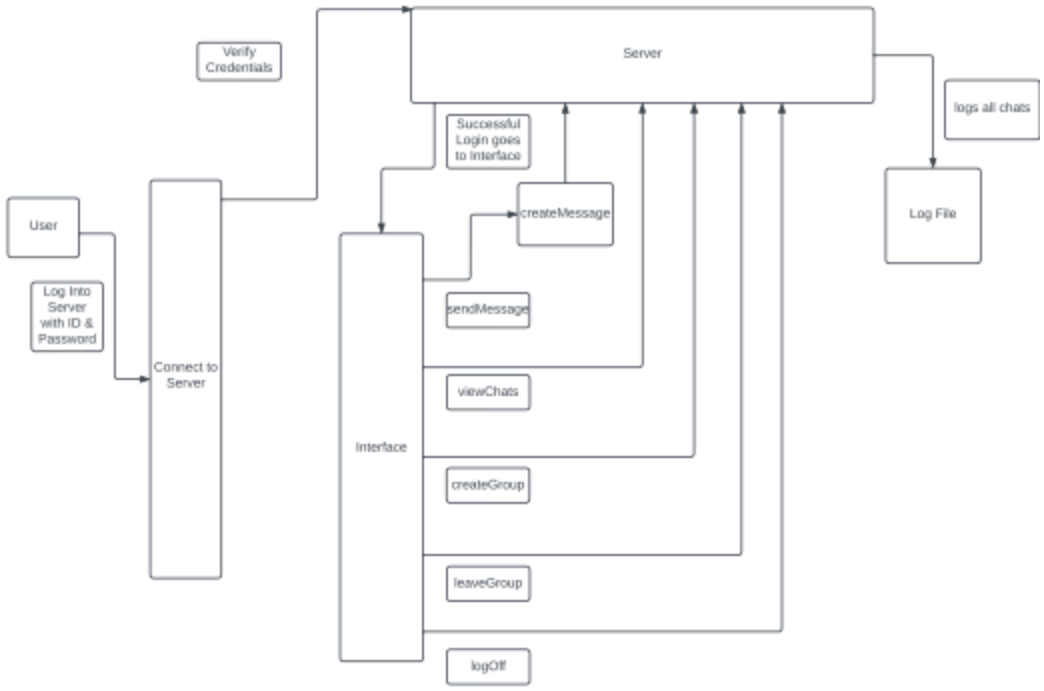


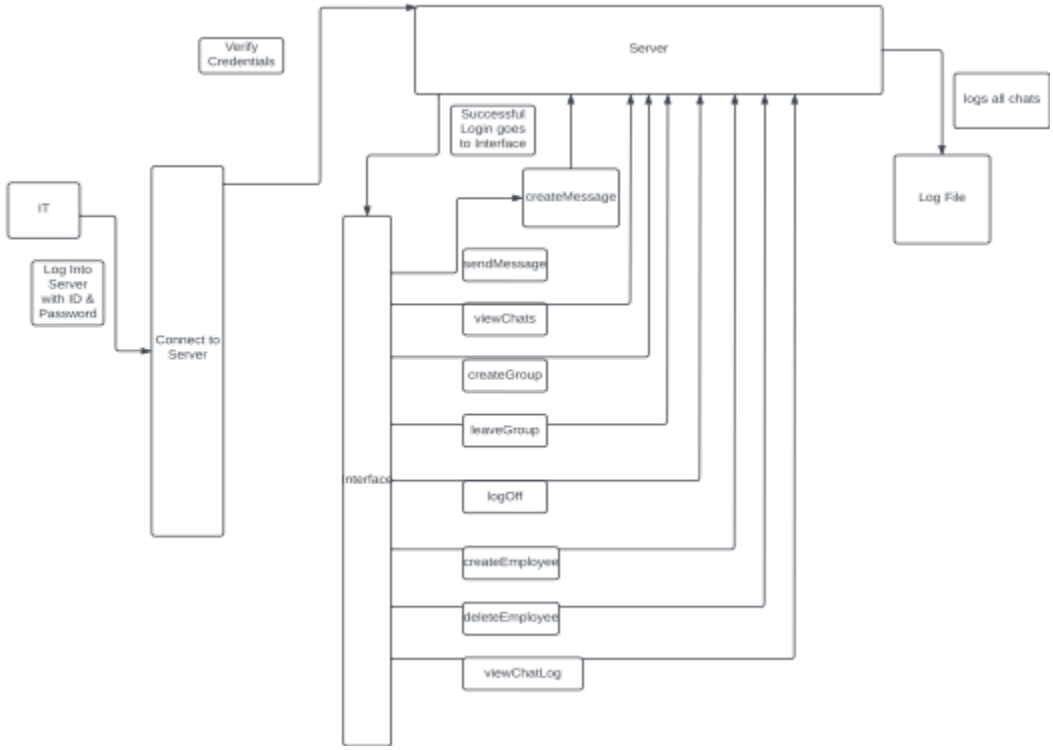
Setters & Getters:

setConnection

connectionToServers

ConnectionToClient


Server
Verify Credentials
logs all chats
Successful Login goes to Interface
createMessage
Log File
User
Log Into Server with ID & Password
Connect to Server
Interface
sendMessage
viewChats
createGroup
leaveGroup
logOff


Server
Verify Credentials
logs all chats
Successful Login goes to Interface
createMessage
Log File
IT
Log Into Server with ID & Password
Connect to Server
Interface
sendMessage
viewChats
createGroup
leaveGroup
logOff
createEmployee
deleteEmployee
viewChatLog

6. Meeting Minutes Log

1st Meeting: 6/8/22 - Requirement Phase

- **Requirement List:**
- **Attendance: Sandra Torres, Ayumi Toki, Son Phil, Emanuel Baca**
    - **Emanuel Note Taker**
    - **Talk about Requirements**
    - **Sandra is typing our Prototype Req List=**

    Communications Project

    Projects will include

    - Project documentation

    - Java source code

    - Junit test suite

    - Git source control repository

    Brief presentation

    Create a communications system for a very large organization.

    This system should allow employees to communicate over chat both synchronously and asynchronously.

    Users should be able to chat privately and in groups.

    All conversations should be logged and viewable by the IT users.

    Privacy should be minimized. Only text is required at this time.

    This is a Java application with a GUI that operates over TCP/IP.

    This system requires a server application and client application.

    There is no web or HTML component.

    No databases, libraries, frameworks, or other technologies may be used without approval.

    Requirements:

Employees, employers, and IT teams are the users of the communication system.

The IT team has access to all chat of employees.

# of users

Only for Java platform

Size is the size of chat, as in the amount of users in a chat. This includes 1-on-1 or group.

There is a maximum and minimum number of users per chat.

Minimum is 2 users.

Maximum is defined by the client.

the user must be an employee or client.

And only IT can add another user.

Access: IT team

action: add users, remove users

actions: action[]

role: action

Roles: role[]

type: voice, chat

Team: user, role[]

User: role, team, id

Users: users[]

size: min, max

- **Meeting at 8:00pm on 6/8/22**
- **Discord Meeting: Nothing to follow up**

2nd Meeting 6/13/22 - Requirements Phase (Meeting with Client)

- **Meeting with Client**
- **Attendance: Sandra Torres, Ayumi Toki, Emanuel Baca**
    - **Time 7:21 pm client meeting 6/13th**

    - **No client group, just employees**
    - **Standard Users and IT Users**
    - **IT Users can read logs, add new users**
    - **Users interface, look at DVD Collection Gui**
    - **User properties: Employee ID, First Name, Last Name, Password**

3rd Meeting 6/20/22

- **Attendance: Sandra Torres, Ayumi Toki, Emanuel Baca**
- **Assign tasks/roles : Ayumi - Scope (Section 1) , Sandra (Section 3), Emanuel (Section(s) 2, 4)**
- **Working on Requirements Document Draft and getting a rough draft down**

4th Meeting 6/21/22

- **Attendance: Sandra Torres, Ayumi Toki, Emanuel Baca**
- **Follow up on Assigned tasks**
- **Sandra made a Github Repository called CS401Group7**
- **Only one question about 2.1 by Emanuel on what to do**
- **Confirmation that Son was removed from our Discord Group Chat**
- **No other updates, possible meeting tomorrow 6/22 depending on how class lecture goes**

5th Meeting 7/6/22

- **Attendance: Sandra Torres, Ayumi Toki, Emanuel Baca**
- **Follow up on Group Presentation take aways**
- **Finish UML Design Phase specifically the getters and setters of all classes**
- **Sketch up a detailed schedule of what we need to work on**
- **Begin thinking about implementation phase**
- **Revision of Requirements Document on UML and Schedule**

6th Meeting 7/13/22

Teachers comments on our Group Checkins as of July 6:

- **Work on UML**
- **Consider classes to design**
- **Begin design work**
- **Not ready for implementation**

Section ONE(Emmanuel):

Comments:

- **Introduction**
- **Scope of the client and servers**

Section TWO (Emmanuel):

Section THREE (Ayumi):

Comments:

- **For Architecture diagram, Use PowerPoint, Google Draw, Canva**
- **Now we can describe client interface, describe servers, internal external interfaces,**
-

Section FOUR(Sandra): user interface

Section FIVE (Emmanuel if you don't overwork): Restrictions, Limitations, And Constraints

Section SIX: Testing Issues

- **Connection issue with test scenario**
-

Section Seven

 # Experiment with GUI for future job experience

Reference for all teammates:

Clients = employees of the large organizations

Server = server for the Java program. This server has the role of messages, chats, and access to chat history.

Class candidates: User, IT, Message

Data Management: text file just for storing chat logs, No data saved on the client side but on the server,

Array/vector database on server side.

Make UML for everything

Large company 10000

For standard user, we will have first name, last name, role, active/inactiveMessage type = Synchronous (when both clients are online) and Asynchronous (when one of the party of offline)

Methods will be public since all users are employees of the organization.

Viewing Messages: "View Personal Messages (with group size 2 users)" and "View Group Messages (with group MORE THAN two users, at least 3)"

From-To search function and populate and reverse text file

To-Dos:

Include the server.

Timeline

- **Client must log in to system,**
- **Client must connect to server.**
- **Message would depend on the connection. (Successful/Error)**
- **Client is now in the system.**
- **Client is an employee.**
- **Client can see list of employees and chat rooms.**
- **Client can send a message, create new group chat, leave group chat, and search for employee.**
- **IT can do what client can do and more: access the chat log.**
- **The server holds the employee credentials and information, messages, chat rooms, and chat logs.**
- **Client can log out. There will be a message that client has logged out of the system.**
- **If client is no longer an employee, then the Server will set the status as inactive.**
- **Emanuel: Client and Server**
- **Ayumi: Message and User Class**

7th Meeting 7/20/2022

- **Focus on taking over methods and classes that are not finished by group mates who dropped the course**
- **Start working on new class EmpDataBase and implement Server and Client**
- **Touch bases with Ayumi on accessing and modifying a text file**

8th Meeting 7/25/2022

- **Start working on GUI now that client and server work together**
- **Start implementing J-Unit testing if possible**

7. Team Schedule & Github Link

7.1 Team Schedule

06/13/2022	After Class FIrst Group Meeting on Discord
06/20/2022	Group Meeting on Assigning Roles for Requirements Document
06/21/2022	Group Meeting on Requirements Document
06/22/2022	Revise the Requirements Document before BlackBoard submission
06/27/2022	Review Documents based on Professor's notes
- 07/01/2022	And Start More on the Design Process. Add content to Github.
07/04/2022 - 07/08/2022	Review Revisions on Requirement Document and Interface Design. Add content to Github.
… 07/29/2022	Continue weekly group meetings after class. Project should be coded by now and we collected feedback from the Professor.

Week number	**Date**	**Action**
Week 1		
Week 2		
Week 3		
Week 4		
Week 5	**07/04/2022**	**Class canceled.**
	07/06/2022	**Adding setters and getters to UML. Identify class candidates, interactions, and data and**

		messages.
Week 6	**July 11**	**Finish Revising Requirements and start Design Document**
	July 13	**Polish & Submit Design Document**
Week 7	**July 18**	**Start implementing**
Week 8	**July 25**	**Implementing**
	July 27	**Finish implementing and Final presentation**
Week 9		

7.2 Github Repository

The following link is the github repository where the Requirements Documents is posted.

~~https://github.com/storres97/CS401Group7~~

https://github.com/emanuelbaca22/CS401Group7