Design Specification Document

Group 1

Samuel Hunt, Xiaoming Liang, Sadhana Chhetri, Tianzheng Cui

**1.1 Introduction**

**1.1. Goals and Objectives**.

This document describes the important aspects of the Communications System, where the employees of a certain institution will use it to send text messages to each other individually or in a group over TCP/IP that uses a Server application and a Client application.

**1.2. Statement of Scope**

To fulfil the goal of creating a chat system for an institution, the system will have a Graphical User Interface where the employees can log into the system and start chatting with others. A secure network connection must be established with the server from the client each time an employee logs in. The system will automatically save the chat logs in a separate file and only allow employees from IT Department to access those message logs.

**1.3 Software Context**

Java will be the language of choice for the implementation of the design of the system. Messages communicated in the system will be maintained in the **MessageLog** class and the messages could be retrieved using the **Message** class. The **UserVerify** class will store the simultaneous login username and password of each employee in a separate file that will facilitate smooth login operations.

**1.4. Major Constraints**

**Issue 1:** High memory usage might make the program slower.

**Option 1.1**: Use of Arrays should be made instead of using vectors to store the list of message logs, list of username and password pairs, user contacts and group lists.

**Option 1.2**: Using dynamic arrays instead of static arrays to store the list of message logs, list of username and password pairs, user contacts and group lists.

**Issue 2:** Very unreliable Internet connection could delay the connection between the client and the server.

**Option 1**: There should be no time limit set for the connection over TCP/IP so that a timeout does not occur when the connection is being set.

**2.0 Data Design**

DataFiles

1. User Login

2. User Contacts

3. MessageLogs

4. GroupList

**2.1 Client Side**

The five commands; sendMessage, createGroup, createAccount, exit, and deleteAccount will be accepted by the Client class and will be passed to the Server. The data passed to the server should require no changes. Response from the server will be displayed on console and UI. The display on console will be used solely for debug purposes.

**2.2 Server Side**

Server will pass socket gained from the client to the serverWorker class to display the proper GUI according to the user permissions. The possible users are Employee user and IT user. To verify the users, the serverWorker would load all the datafiles into temporary data storage such as a vector. This information would be stored by calling Load\_\_\_\_ ( datafile, arg ). Where arg would be the vector. The empty space after the Load would represent the apporiate data being stored. The implementation of ; sendMessage createGroup, deleteAccount, createAccount, displayChat, and displayLog would require the take in arguments of User and data storage. Then would change the data according to the method's purpose. After all changes are confirm, the data would be overwritten to the data file.

**3.1 Program Structure**

The Chat system runs as a client and server application over the TCP/IP network.

**3.1.1 Architecture Diagram**

DataFiles <-> ServerWorker <-> Server ----TCP/IP Netowrking---- Client <-> GUI

**3.2 Description of Client**

**3.2.1 Client Processing Narrative**

* Employee\_GUI, IT\_GUI these GUI will be dispalyed depening on User permissions.
* The User class will have a bool attribute called IT which will determine the methods it may call, such as the delete and createAccount which are strictly only avaiable for IT users.
* Message class will only have getter functions since there is no need to change any messages after creation.
* There are objects for the storage of the data loaded from the datafiles.These objects will have load, save and add methods to them. This allows for access to the data files and changes them when necessary.
* The Client class will need to have a function which allows them to request to the server

**3.2.2 Client Interface Description**

* Client only necessary inputs are sending a message and creating an account. These inputs will not be changed when sent to the server. Outputs will be obtained from the serverWorker

**3.2.3 Client Procesing Details**

* The processing of a client will be something of this sort. Client request to the server.Client is prompted to enter password and username to verify login credentials. Client will be prompted with a contact list to choose who to communicate with. In addition to that prompt depending on the user's permissions there may be extra buttons which allow for creation and messageLog of the entire chat system. After client is finish with their actions client can choose to exit out of the chat system.
  1. **Description of Server**

The Server's purpose would be to "listen" for the request made by the client class and will pass the work to another class serverWorker. For each request the server receieves it will create a new thread and pass that thread to the serverWorker. This allows for the server to remain open and enables multithreading.

**3.4 Software Interface Description**

**3.4.1 External Interface**

* External Interface would be moniter and keyboard which will allow for users to access the chat system.

**3.4.2 Internal Interface**

* There will be client and server connection and all actions done by client will be passed to server for a serverWorker to manage. Whether it be for chatting or accessing messages, this will be done by serverWorker and be displayed by a GUI.

**3.4.3 Human Interface**

* Login Display -> ContactList -> ChatDisplay (Employee User)
* Login Display -> ContactList and buttons for Account Creation/Deletion and MessageLog -> DisplayCurrentAction (IT)

**4.0 User Interface Design**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**5.0 Restrictions, Limitations, and constraints**

* The Java code has limitations with certain methods.
* The memory is very limited and could get in the way.
* If we use an array, it can only hold a certain number of items at once.

**6.0 Testing Issues**

* Socket could be hard to test, there are a lot of sockets.
* Server may be difficult to get working, port numbers could get in the way.
* There may be other potential problems with the code itself.

**7.0 Appendices**

* Appendix A: Message Logs in Depth
* Appendix B: More on Client and Server