Ian Percy

Chat Application

Requirements Document

Table of Contents

1 Introduction **3**

1.1 Purpose and Scope *3*

1.2 Target Audience *3*

1.3 Terms and Definitions *3*

2 Product Overview **4**

2.1 Users and Stakeholders *4*

2.1.1 Fei Xie 4

2.1.2 Bin Lin 4

2.2 Use cases *5*

2.2.1 Account Authentication 5

2.2.2 Chat 6

2.2.3 View Chat History 6

3 Functional Requirements **7**

3.1 User Authentication 7

3.2 Chat *8*

3.2.1 Group Chat 9

3.2.2 Individual Chat 10

3.3 View Chat History *10*

4 Nonfunctional Requirements **12**

4.1 User Interface *12*

4.1.1 View Before Authentication 12

4.1.2 View After Authentication 13

4.2 Portability *14*

4.3 Security *15*

5 Milestones and Deliverables **16**

5.1 *Requirements Documentation* *16*

5.2 *Design Documentation* *16*

5.3 *Test Plan*

17

5.4 *Final Deliverables* *17*

1- Introduction

This project is aimed at creating a functional chat application. The application will be using Java to create a client-server model for communication between multiple, unique clients. This document will be explaining the steps that will be taken to fulfill this task.

1.1 - Purpose and Scope

The task at hand is to develop a program that will allow clients to communicate via a simple chat interface. This document will explain the steps that will be taken to accomplish this task, what the program will do, and how it will react with users. Documentation following will outline methodical preparation taken to produce an effective solution to our problem.

The final product will be a local program that will allow multiple to communicate with one another through a basic user interface. While the software will be tested locally and with a few external clients, the purpose of this project is to create a program which can communicate from multiple machines.

1.2 - Target Audience

The main audience for this project is Fei Xie and Bin Lin of the CS300 Spring 2017 class at Portland State University. Grading and clarifying communicating will be be done with either of these individuals throughout the course to complete this project. Ultimately, the final product will be demoed to Bin at the end of the term.

1.3 - Terms and Definitions

Client - Server - Creation of both a server and a client with purpose of holding a monitored connection with a distributed workflow.

User - Any individual who interacts with the application

JAR - Java Archive format for storage of many Java class files into one file for distribution.

2 - Product Overview

This chat application will have several key features: Clients will have the ability to login with a user/password combination, chat with an individual client or the group chat, and also view chat history. Clients in this case will be instances of the program which have the ability to send messages. A server will provide the link between clients, maintain user state information, and also a chat history.

2.1- Users and Stakeholders

Active stakeholders for this project will be the instruction staff of CS300 at Portland State and also myself. The users will be any person with access to the executable (JAR) that wishes to chat with other clients connected to the server. Our setup for the system is shown below in Figure 1.

2.1.1- Stakeholder 1

Primary stakeholders for the project CS300 Project Administrators (Fei Xie and Bin Lin). Grading and main contact for clarification will be with either one. The main requirements document was acquired through CS300.

2.1.2 - Stakeholder 2

The secondary stakeholders for the project will be myself and possibly other students of CS300 who will test the program's execution closer to the delivery date near the end of the term.

2.2 - Use cases

This section outlines use cases in broad detail and the functional requirements in Section 3 will explain the main actors for each use case in detail. A client-server interaction is the critical portion of this project. The server's primary function is to provide the link between clients and manage the data of the users. Clients will connect and chat with other clients.

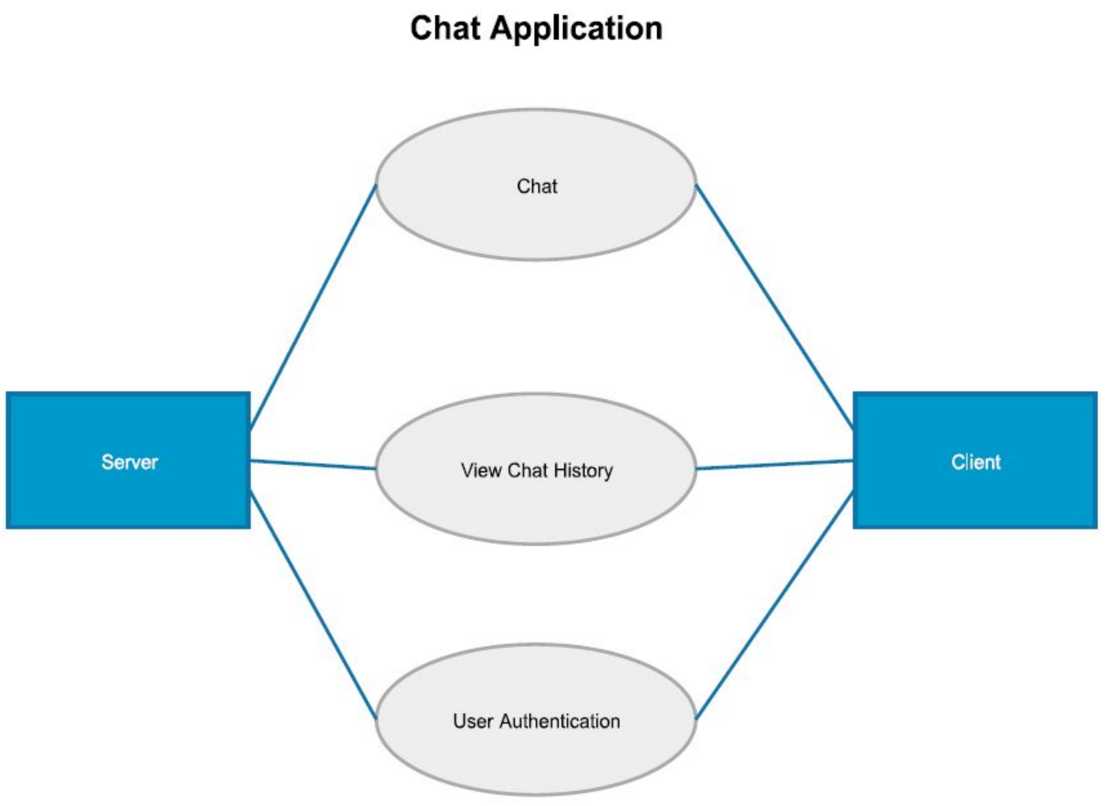

Figure 1: Use Cases for Chat Application

2.2.1 - User Authentication

Clients will connect to the server via a username and password. This authentication process requires each new client to create an account. If an account has already been created, the user can connect using their proper credentials.

2.2.2 -Chat (Group or Pair)

The main use case is for a client to have the ability to chat with other clients. Options for this use case are for a general group chat or for starting a chat with an individual online user. Clients, once connected to the server, will have communication ability between themselves and other clients.

2.2.3 - Chat History

A persistent chat history will be maintained for each user that has connected successfully to the server through an account creation process. Management and storage will be done exclusively by the server. Clients will have the ability to view their chat history for any previous sessions.

3 - Functional Requirements

Use cases focus on three major requirements: Clients having the ability to connect with the server, Clients communicating with other Clients, and users being able to view their chat history. These functional requirements impact both the server and the client in some way or another so each will contain their role for the functional requirements.

3.1 - User Authentication Use Case

Brief Description:

Users/clients are given the ability to connect with the application via a username and password. First time users will create a login with the server.

Diagram:

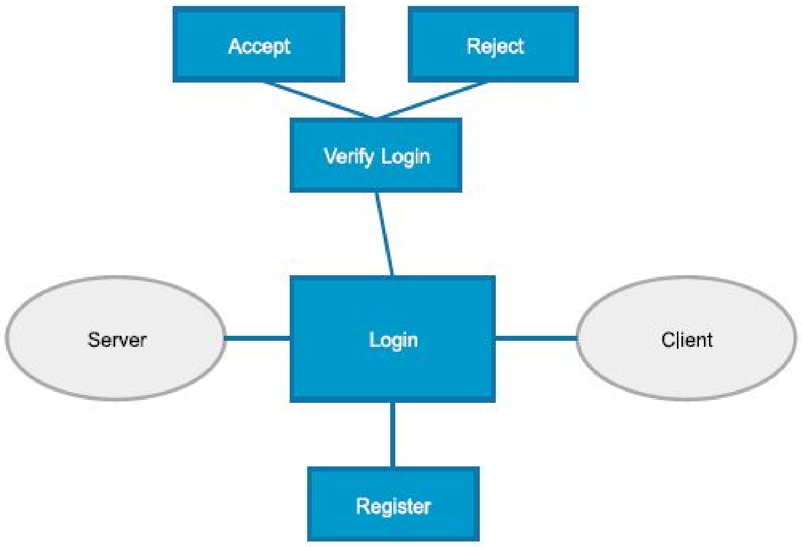

Figure 2: User Authentication Use Case

Initial Step-By-Step Description:

This use case will have two main components. Users are able to login with a previously used username and password. If the user does not have an account, they can register for one.

1. User will be presented with interface for the chat application
    a. User can then will type username and password into the proper fields
    b. Otherwise, user will click on a register button and create account
2. Server verifies login credentials with saved users
3. User will then be either be allowed into the system and progress forward or repeat step 1

This use case is the first major requirement for the project, as users will be unable to interact with any of the other aspects without proper connection with the sever. Users will be required to have a unique login already. Any attempt to register a current username will return an error.

3.2 - Chat Use Case

Brief Description:

Once clients have connected to the server, the application will open to a general group chat. Users then have the choice to select any individual logged in users and start a chat with that individual.

Diagram:

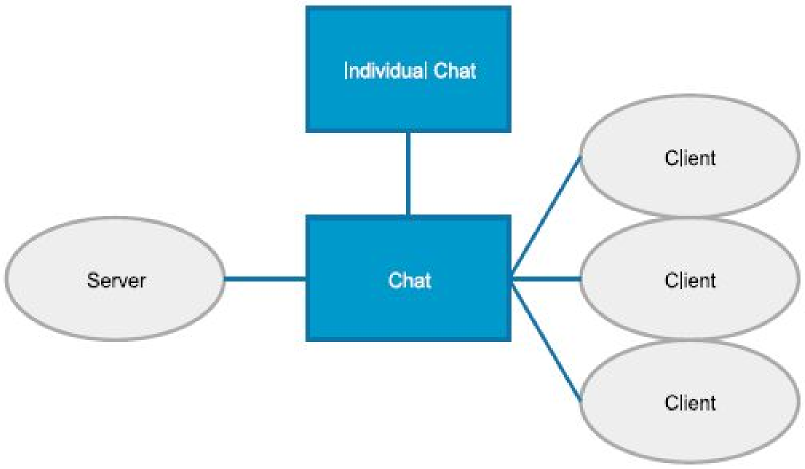

Figure 3: Chat Use Case

This use case gives clients two choices. Once a client is connected, users will be presented with a interface where majority of the view is dedicated to a group chat. Users are automatically connected to this chat which is why Figure 3 only has one choice off of the chat box in the middle.

3.2.1 - Chat Use Case : Group Chat

Initial Step-By-Step Description:

1. Assumption: User has connected to system
2. User will be presented with the main interface for the chat application
3. Each user will be connected automatically is added to the group chat
4. Every user is part of the same group chat and messages are seen by all

3.2.2. - Chat Use Case: Individual Chat

Initial Step-By-Step Description:

1. Assumption: User has connected to system
2. User will be presented with the main interface for the chat application
3. List of the users who have connected to the system is displayed
4. If a user wishes to start a chat with an individual user
    a. User must select the user from the list
    b. If user is logged in, a new chat will be started between the user and the target user
    c. If user is not logged in, a message will be displayed to the user to reflect an error.
5. Any individual chats will be saved according to Use Case 3.2

As this project is centered around providing users the ability to chat, the bulk of design and implementation is devoted to this aspect. The requirements for this use case state we need the ability to have a group chat upon login and provide the ability for individual chats to happen between active users.

3.3 - Chat History

Brief Description:

Once a user has completed a chat and disconnected from the system, the server will maintain a record of their chats. A user, upon a subsequent login, will be able to view their chat history.

Diagram:

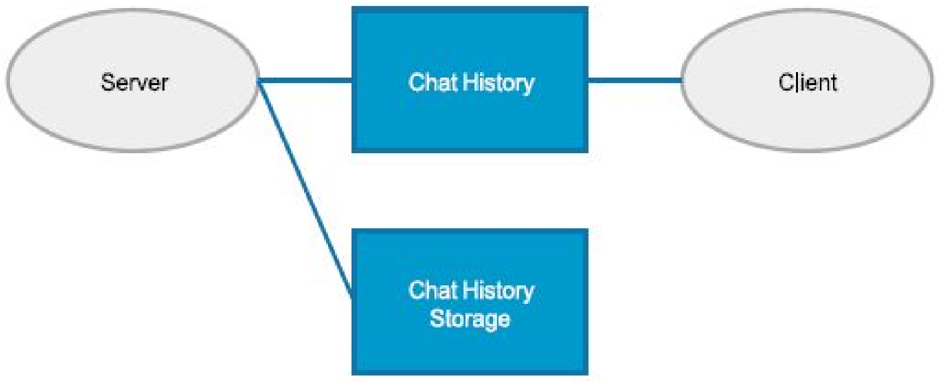

Figure 4: Chat History Use Case

Initial Step-By-Step Description:

1. Assumption: User has connected with system, participated in a chat, and logged out of the system
2. User will be presented with the main interface for the chat application
3. Users will have an option on the screen to view their chat history
4. If a user wishes to view their history, this option will open a new window that will display their chat by day and by user
5. Once a user has finished with viewing the history, closing the screen will return them back to the primary group chat view

As stated before, the server will handle the storage and recovery of any chat history. The interface will help display the history in a user-friendly format. A client will attempt to read their history from a server, which will either display the history or an error if empty history. Chat history is the third major functional requirement for this project.

4 - Nonfunctional Requirements

Additional requirements have been found as necessary for a properly functioning project. While the functional requirements above outline the proper way to handle the use case implementations, these following requirements help with creating an application which is user friendly and reliable. These non-functional requirements are user interface, portability, and security.

4.1 - User Interface

A functioning user interface will allow any client to interact properly with the system. There are a couple essential views that have been identified: the interface view before authentication and the interface view after authentication. These are split into two different categories as the view before authentication will not contain as many features as after authentication.

4.1.1 - View Before Authentication

Diagram:

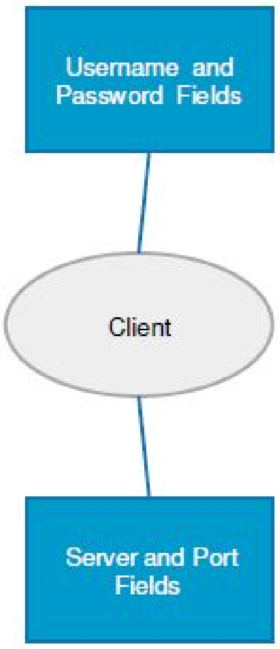

Figure 5: View Before Authentication

The above diagram shows the options that will be available to the client before they have connected to the server. The interface will be displayed through the application. Server and port will be automatically filled by the program as to not create any confusion. Username and password fields become the only part of the interface a user will need to interact with before connection. After successful authentication, the second view will be displayed.

4.1.2 - View After Authentication

Diagram:

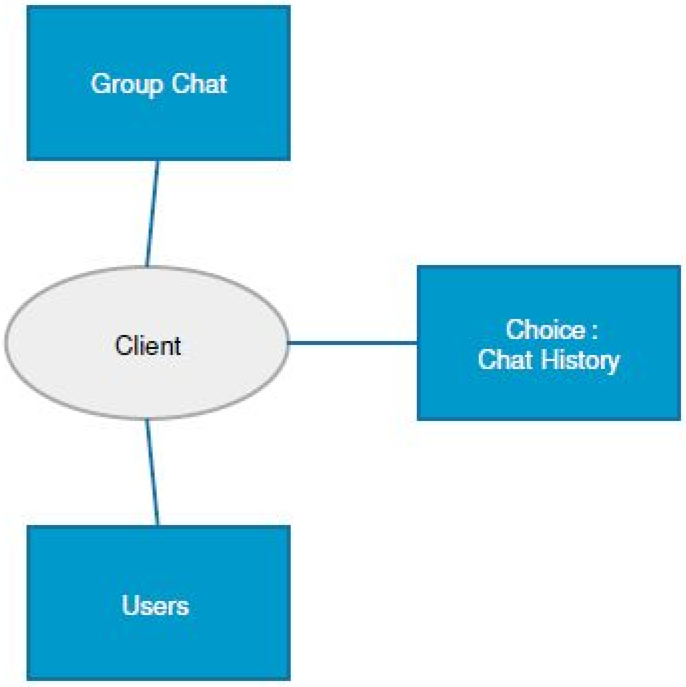

Figure 6: View After Authentication

Once a client has connected to the server, the view will become more active with live components. First, the user will be entered into the group chat which will dominate the display. From here the user can choose to interact with the chat or be a spectator. Second, the chat history button will become available, however this does not mean there will be any data if this is the user's first time connecting to the system. Third, the users will be displayed to the client. This will allow the client to then choose another user to start an individual chat with or not.

Server messages will also serve an important part of this connected view as well. Active users will be highlighted in the list of users who have connected to the system and also moved to the top of the list view. If a user logs in or out, any active user will see a message of this on their individual view.

To summarize the user interface non-functional requirement, the view has been split into a pre-authentication phase and a post-authentication phase. Views after a successful connection will allow the user more choices such as starting a chat with another individual.

4.2 - Portability

Second non-functional requirement chosen was to emphasize creation an application that will work at least minimally between different types of systems. Having a chat application that only works on one computer means that the application is obsolete. This will be accomplished in the following way:

1. Assumption: Code is at least functioning to a certain level where communication is possible
2. Compile the .java files and any external files that are necessary
3. Convert these files into an executable JAR file by following documentation provided by Oracle

JAR is my current choice due to it being portable with Windows environments and portable due to Java's core API handling these files easily (Reference 1.)

4.3 - Security

Another non-functional requirement that will be added is the idea of creating some sense of security for the user passwords. Security is not a major concern as this application will not be mass distributed. Passwords are processed over a hash value to create an encrypted key:value relationship for each user.

Diagram:

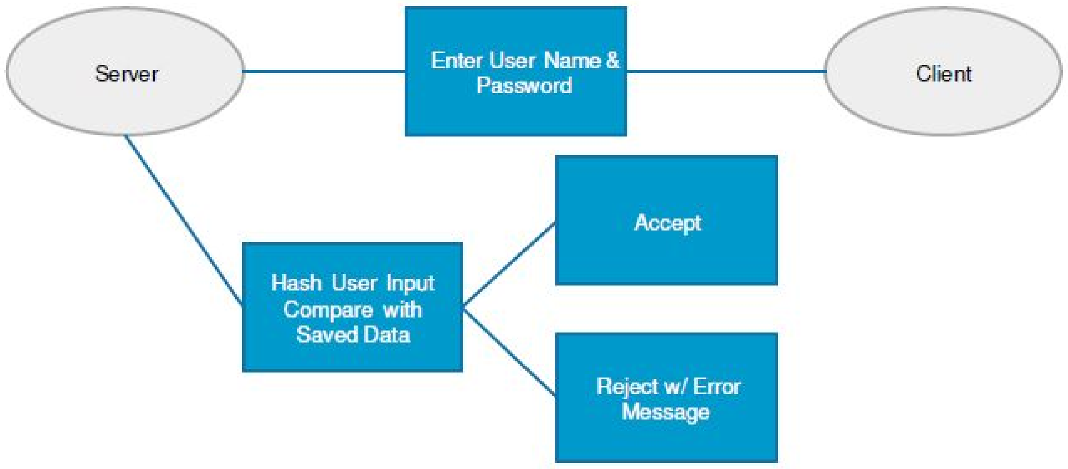
Server
Enter User Name & Password
Client
Accept
Hash User Input Compare with Saved Data
Reject w/ Error Message

Figure 7: Security for User Information

While security was not explicit requirement for this project, security should be a consideration for every program that will store user data. The server will manage all user related information with the passwords being hashed over a particular value.

Milestones and Deliverables

Development of this project will be completed in several stages. The development will be following a simple software development life cycle. Sequence of steps and deliverables will be requirements defined, design development, test plan, and final project submission/deliverables.

	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7	Wk 8
Requirements Due April 25th, 2017	🟦	🟦						
Design Due May 8th, 2017			🟦	🟦				
Test Plan Due May 30th, 2017			▧	▧	🟦	🟦	▧	▧
Final Deliverables Due June 8th, 2017							🟦	🟦

Requirements Documentation (2 Weeks)

This document has outlined the proper steps taken towards handling the requirements documentation in a sufficient manner. Use cases have been defined along with functional and nonfunctional requirements. This is to be submitted April 25, 2017.

Design Documentation (2 Weeks)

Design for the project will begin after the requirements documentation has been submitted. Architecture for the system will be planned during this phase. Design will be focused around three major components: client-server interaction, chat, and storage-retrieval of user chat history. The documentation will outline the plan to implement the requirements set forth in this document. Design documentation will be completed by May 9th, 2017.

Test Plan (~6 Weeks)

Once requirements have been understood and the design for the system is complete, explicit testing of the system can commence. While testing will be occurring throughout each stage of the project, this phase will solidify the steps needed to demonstrate proper functioning of the project. A formal test plan is due on May 30th, 2017.

Final Deliverables (2 Weeks)

Last stage of deliverables will be the submission of a final project report. The application will be demonstrated with Bin near the end of the term to show that all requirements have been met. Final date for submission and end of the project is June 8th, 2017.

References

1. Oracle. Lesson: Packaging Programs in JAR Files
   http://docs.oracle.com/javase/tutorial/deployment/jar/index.html