Chris McPherson

mcpher3@pdx.edu

916415689

Requirements Document

4/24/17 10:00

JavaMess

Requirements Document

Table of Contents

1. Introduction

Introduce this document and the project it describes. You should also summarize the remaining content of the document here.
JavaMess is a distributed messaging system written in Java.

1.1. Purpose and Scope

You should describe this document by giving its purpose, scope.
(what it is supposed to do)
JavaMess is a distributed messaging system written in Java. It will have a server and a chat client. Both will have a graphical interphase that supports pictures. JavaMess will support a login system with a username and password. The JavaMess users will be able to chat with other users online and save specific users to their list of other users. Users will also be able to chat with many users all at once.

1.2. Target Audience

Describe the target audience for this document.
This messaging program will be targeted to people who want a simple chat system that people may want to use with friends or other groups of friends.

1.3. Terms and Definitions

Define any terms or acronyms you will be using in the remainder of this document.
JavaMess: the name of the Java Messaging system being proposed.
client: The JavaMess client program

2. Product Overview

Give a high level description of the functionality of the project here. Describe the purpose of this section. It may be useful to give your definition of a user, a stake holder and a use case. If there are scope limitations to the project, i.e. things you will not be doing, or are not required to do, this is a good section to put those.

2.1. Users and Stakeholders

Describe the purpose of this section. Only a few sentences are expected here.

2.1.1. Chris McPherson

List the first stakeholder or class of stakeholders if necessary. Describe, exactly, their role in the development, deployment, use, maintenance, etc. of the software.
The sole stakeholder in the creation of the program is Chris McPherson. I will be handling the development, deployment, use, and maintenance of this program.

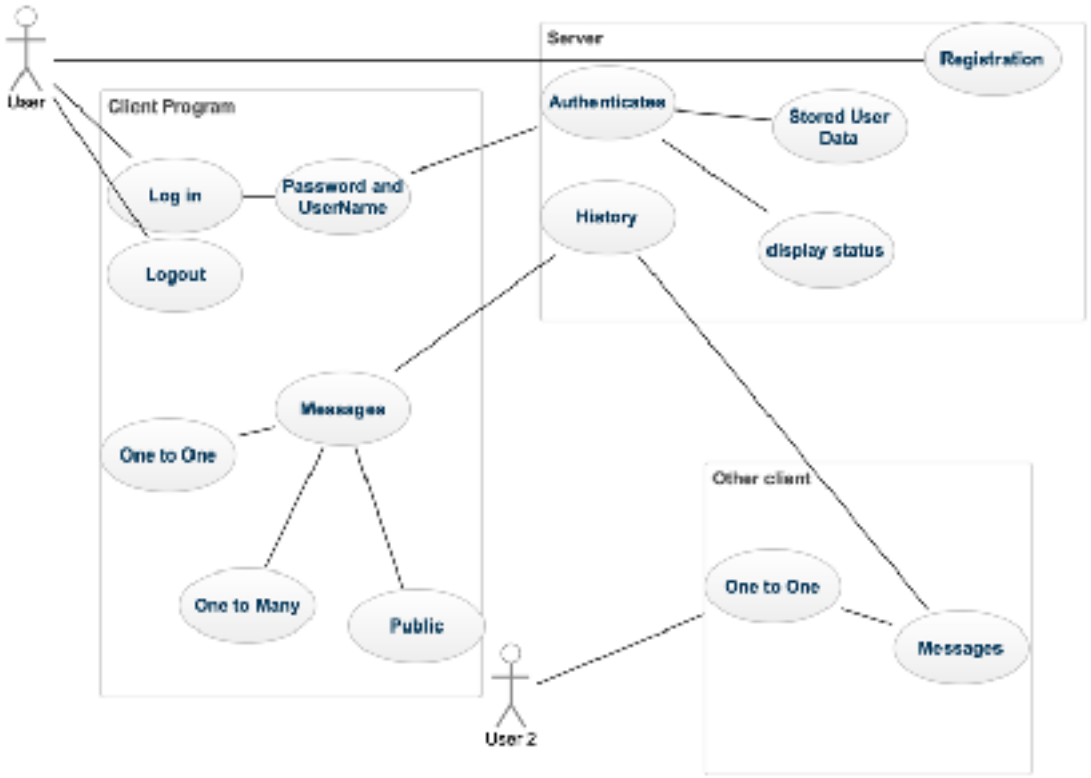
2.1.2. Fei Xie and Bin Lin

Both Fie Xie and Bin Lin will be involved in evaluating the success and usefulness of JavaMess.

2.2. Use cases

Describe the purpose of this section. Only a few sentences are expected here.

The use cases cover the possible actions that may occur with JavaMess. These may between the users, the client program, server, or the different actions between the different classes.



2.2.1. Registration

Describe the first use case here. Be sure to explicitly identify the participants, human or otherwise, and explain their roles. Diagrams may be effective here, particularly a sequence diagram.

New Users will have to register before being able to chat with other users. Users will enter a username. The client will send the username to check to see if it is unique or have the user enter a different name. The client will then enter a password to be sent by client to be stored on the server.

2.2.2. Logging in

The User enters their username and password. If the username does not exist or the password doesn't match, the client will give an error message and prompt the user to register. If the username and password are matched on the server side, The online status is set to true and the client program will get messaging information from the server.

2.2.3. One to One Messaging

If a client has logged in a client may send messages to an other user who is logged in. A message will be sent from the client to the server. That message will be copied into the message history for the two users and then be sent to the recipient's client program.

2.2.4. One to Many Messaging

If a client has logged in a client may send messages to other users who are also logged in. A message will be sent from the client to the server. That message will be copied into the message history for all the users involved and then be sent to all the recipients' client program.

3. Functional Requirements

Describe the purpose of this section and outline its contents. Only a few sentences are expected here. It may help to define a functional requirement.

In order for the program to successfully function it will need to fulfill a set of functional requirements. These functional requirements are a set of actions that either the client and server programs need to be able to perform. Many of these functions require bother the client and server to work together.

3.1. Storing Client Records

The server must be able to store client user information and chat history.

3.1.1. Registration

The user will enter a username into the client. If the username is not stored on the server, a user will be able to enter a password associated with that username. The username and password will then be stored on the server until the client decides to delete their account.

3.1.2. Logging in

The user will enter their username and password, if the username is not found or the password doesn't match the password on the server, an error message will be displayed. If the username and password match, that users status will be set to online and then messages will be delivered. Once a user is logged in, they will be able to send messages to other users and have their friend list displayed

3.1.3. Storing History

All messages sent from client to the server will be copied into the chat history. Each chat history will be associated with the usernames who are part of the conversation. The chat history can be accessed by any user who is a part of the chat conversation.

3.2. Messaging

The main function of this chat program it to have one client be able to send messages to other client through the server. A user will send one message to the server, which is copied into the history, and then sent to recipients.

3.2.1. One to One messaging

One to one messaging will specifically be addressed to one specific user and be functionally the same as one to many messaging. Each message will have the sender and recipient, as well as the time sent. The message will be sent out, copied to the server, and the sent to the recipient.

3.2.2. One to One messaging

One to many messages will be addressed a set of users. Each message will have the sender and recipients, as well as the time sent. The message will be sent out, copied to the server, and the sent to each recipient.

3.2.3. Public Messaging

A client can send a public message by not entering a recipient. The message will be copied into public history and displayed in the public window.

3.3. Friend List

The friend list is a linked list of stored usernames of other users. This will allow a user to easily message users that are commonly messaged.

3.4. Logging Out

A user can set their logged in status to false. Once that value is set to false they are sent back to login screen. They are no longer able to send messages or access histories.

4. Nonfunctional Requirements

Describe the purpose of this section and outline its contents. Only a few sentences are expected here. It may help to define a nonfunctional requirement.

The nonfunctional requirements are the qualities that determine the qualities of the program. These are specific goals with expectations of how the program should run with respect to ease of use.

4.1. Security

Each of the users must have a password to prevent unauthorized from accessing the history of a conversation.

4.1.1. Password

The password will have a minimum requirement of 8 characters and must contain at least one number and character.

4.1.2. Password Storage

The passwords stored in the system will be salted and hashed while stored on the server side.

4.2. Response Times

The response times between the client and the server must be quick enough to allow for synchronous communication. A message must be uploaded to the server, copied into history, and delivered in a few seconds.

4.3. Usability

The chat program will support a graphical interface for the client that will streamline user interaction. Users will be able to use windows to open message windows manually or from the friends list.

5. Milestones and Deliverables

The milestones is a set of internal goals to make sure that a satisfactory level of progress is being met. In order to have a strong sense of direction for the project, the final deliverable must be broken up into a set of smaller milestones.

5.1. Server Storage

The server must be able to store usernames, secure passwords, and all of the chat histories. Each of these roles will need different classes as well as different data structures. This will be done by writing the different classes and the different data structures that will be storing those classes.

5.1.1. User Class

The user class will contain username, password, status, and be a linked list of conversations.

5.1.2. Users Data Structure

The user Data Structure will be an array of binary search trees. Each user will be organized into the array based on the first character of the username. After that it will be organized by user name into the binary search tree.

5.1.3. History Classes

The history class will be a dynamically bound hierarchy that will store previous conversations

5.2. Server Connectivity

In order for the chat program to work effectively the client programs must be distributed across multiple computers and handle a large number of users.

5.2.1. Test Connection.

Strings must be able to be sent back and forth between the client and server in order to demonstrate information can be passed back and forth.

5.3. Client Program

The client program will must be implemented to take advantage of the server and send messages.

5.3.1. Registration Function

The client program will sends usernames in passwords in order for them to be stored on the server. The server checks to make sure they are suitable.

5.3.2. Log in

Check to see that the username and password are matched when appropriately and display error messages when not appropriate. When the password and username are matched the logged in status is set to true.

5.3.3. Send Messages

Check to make sure the client can send and receive messages, one to one, one to many, and public messages. All of the messages must be stored into the appropriate history.

5.3.4. Log out

A user can set the logged in status to false, and can no longer send messages or view history.