Daniel Rodriguez

Daniel's Chat Application

Design Document

Table of Contents

Introduction

The Daniel's Chat Application Design Document outlines the expected architecture and system design of Daniel's Chat Application including expected users and behaviors. The chat application is a program designed to allow two or more users to communicate with each other in real time with the user of a server to relay messages between them. Functionalities include logging in and out of the program, viewing online users available to chat, and sending, receiving and archiving messages.

Purpose and Scope

The Design Document allows the chat application programmer to plan out all of the details of the expected design of the program to ensure that everything assigned is being accounted for in the design process. The Design Documents allows the programmer to create a sort of checklist to compare what is expected versus what is being planned for the design. If all bases are covered, the programmer, and grader, can be assured that the project is on the right path and fewer errors can be expected as the project progresses.

Target Audience

The Design Document is intended to be reviewed by both the programmer himself, to review his understanding and knowledge of the assignment, as well as the professor and teaching assistant, as they will be able to see the programmer's understanding of the assignment and monitor the project's progress.

Terms and Definitions

Chat Application - refers to Daniel's Chat Application, or the main program assigned for this project. It will be hosted on a local computer and communicates with the program server, which contains all the data for the program. Also referred to as "program."

Home Screen - Once logged into the chat application, the default screen is the home screen which includes a list of online users, menu options, as well as a text field--used to send a message out to the entire list of online users.

Program - refers to Daniel's Chat Application, or the main program assigned for this project. It will be hosted on a local computer and communicates with the program server, which contains all the data for the program. Also referred to as "chat application."

Server - a non-local computer that stores the underlying data and data structure for the chat application. A user must successfully login to their account in order to communicate with the server

User - a person at a local computer who is communicating with the chat application, which communicates with a server.

Design Considerations

The following outlines the boundaries that must be maintained in the implementation of this project. Constraints, dependencies as well as the methodology used to execute the project implementation.

Constraints and Dependencies

The Daniel's Chat Application must be written in the Java programming language. There is a project timeline in which different parts of the project are due at different times: the Requirements Document was due 2 weeks after project assignment. This Design Document is due now, 4 weeks after assignment. 7 weeks after assignment, a Test Plan will be due. And, finally, the project report and final deliverables will be due a couple of weeks later. The Chat Application will require an internet connection, in order to use it successfully. And it will require multiple users in order to test its full functionality--one will be able to communication with someone else.

Methodology

A modified Waterfall Model methodology will be used to complete this project. Already the project requirements have been assessed and a Requirements Document produced outlining the programmer's understanding of the project details and requirements. This current document, the Design Document, outlines finer details of the architecture of the program. Next, implementation and coding will begin with ongoing code testing, followed by the launch or submission of the project for review by the class professor and teaching assistant.

System Overview

The Daniel Chat Application will provide users with four overarching functionalities: logging in/registering account, sending & receiving messages between users, logging off, and archiving messages. Below is an overview of these functionalities. In the next section, System Architecture, we take a closer look at how these should be implemented.

Logging In/Registering User Account

A human user will open the chat application program and enter in login information which will be communicated to a server on the other end, which will verify or add login information in the data structured housed within it. It is the server's responsibility to manage all underlying program data.

Sending/Receiving Messages

A logged in human user will select users to send a message to. Upon submission of message, the local computer will send information to the server on the other end, which will be responsible for managing the behavior of that data--sending message to the correct user or verifying receiving user information.

Logging Off

A human user will indicate that they are finished using the chat application, the local computer will communicate this with the server which will mark the user as offline and store user information appropriately.

Archiving Messages

Throughout the messaging experienced, all messages sent or received will be stored in a message history accessible by individual users. They will be able to access all message threads they have been a part of.

System Architecture

The System Architecture section outlines the process by which the functionalities will work, pointing out classes and methods needed to accomplish tasks.

Logging In/Registering User Account

The login screen will be very plain, featuring two text fields labeled Username and Password, respectively. Underneath the text fields, there will be a button labeled “Enter” and a link labeled “Register.” A user should enter in a registered username and password or a username and password they would like to register and press the appropriate button or link.

If no information is entered or has incorrect symbols (only text characters (letters) are allowed) and the “Register” link was clicked on, the login screen will reopen with an indicator stating, “please enter a username and password you wish to register.”

If the user has inputted text and selects the “Register” link, the program will send user information to the user database, where the **user** class will call a function to check for an existing user of the same username. If a match is found, the server side program will return an integer to the local program indicating, “The username you have entered has already been registered. Please use the corresponding password or select a new username.” This will be displayed on the login screen, where the user will have another opportunity to register an account. If a match is not found in the user database, an add user function will be called and the information will be added to the user database.

If the user enters in information and selects the “Enter” button on the login screen, the local program will send this information to the server where the user class will call a function to check for an existing user of the same username. If a match is not found, the server side program will return an integer indicating, “This username has not been registered. Please select the register link or enter a registered account information.” If a username match is found, the password entered will be checked against the matching user information found. If the passwords do not match, an integer will be returned to the local program indicating “the password you entered does not match our records. Please try again,” and the user will remain at the login screen. Once a username and password have matched a user’s information in the user database, a home screen will be displayed to the user.

Home Screen

Upon successfully logging in, a “home screen” will be displayed with two main windows inside it: a menu window and a user's’ window.

Menu

The menu will only have two options for the user to choose from: “Message All Users” and “Logoff.” Clicking the “Message All Users” option will open a new window with a text field where the user may type a message they intend to send to all users on the chat application. Selecting the “Logoff” option will use the user class’s logoff function to disconnect the user from the chat application server, and thus their access to the chat application services--including the ability to send, receive, or view past messages.

User’s Window - Sending & Receiving Messages

In the user’s window, a list of all chat application users’ usernames will be displayed. The user can double click on a username and a new chat window will open with a text field that the user can type their message in to send to the selected user. Once the message is entered and the user hits <Enter> to send, the local program will use the Message class’s send function to send the message to the chat application server, which will traverse the user database, find the appropriate user the message is being sent to and then use the server’s send message function to send the message to the appropriate user. If the user is online, they should receive a notification by a new open window, if it is a new message/there are no previously opened windows for a direct chat with the sending user. If there is an existing window open for the sending user, then the chat window will blink in the taskbar to indicate that a new message has been received. If the receiving user is offline, the message will be sent to the Archive class--discussed below.

Archiving Messages

Every time a user sends a message, the message is sent to the Archive class which will call a function to append that message to the chat history between the two users. The chat history will be stored in some node which contains the chat history as well as information regarding the users involved. When a user opens a new chat window by selecting a user to send a message to, a function is called to the archive class to find a node that contains both users. If found, the chat history text stored within it is sent to the local program to display in the newly opened chat window between the two users. If no matching node is found with both users name--meaning this is the first time the two users have chatted, an add archived message function will be called to create a new node with the two users. Any messages sent between them will be added as they are sent.

Detailed System Design

Logging In/Registering User Account

The username and password text that the user enters to register or login will be saved in a class object which is passed to the server program which will pass the object into a function that will traverse a binary search tree of users--the user database--sorted by username. The functions mentioned above will be used to find a matching user in the user database, and to return an appropriate integer indicating the status of the user request.

Home Screen

User's Window - Sending, Receiving, and Archiving Messages

Messages sent by the user will be saved as a dynamically allocated character array which will be sent to the server program which then calls both the message and archive classes. The message class manages what happens to the message--it will search the user database for the matching receiving user and pass that character array to its send function which will notify the receiving user of a new incoming message. The archive class will call its search function, find a matching node in a binary tree of archived messages--by finding the matching combination of sending and receiving users. Once found, an add function will append the newly sent message to the existing chat history between the users. If the message is to all users, the receiving user will be stored as "All Users" instead of listing each one out one by one.