Ian Percy

Chat Application

Test Plan Document

Table of Contents

1 Introduction

This is document will show how the application created for this project will be tested before final deliverable date. Unit testing and integration testing will be done to demonstrate the functionality of the project meets the requirements.

1.1 Purpose and Scope

Purpose of this document is to outline testing strategy and testing approach. This will include unit tests, integration tests, and estimated schedule. Scope is to develop the test plan for the local chat application developed for this project.

1.2 Target Audience

Target audience for this document is the instruction staff of CS300 and also myself as the developer.

1.3 Terms and Definitions

Socket- internal endpoint for sending and receiving data through a single node

2 Test Plan Description

This section will outline the objectives of the tests to be performed. The goal for this document is to create a template that can be used by myself to execute the necessary tests in a time conscience manner.

2.1 Scope of Testing

Testing will cover the use cases (functional and non-functional requirements) described in the requirements document and the data flow between the system architecture. Testing will be done throughout implementation to test object behavior of the architecture but more thorough testing is required before project release. Testing for the use cases will include as many boundary cases as possible. The main functionality that will be tested is as follows: register user, login user, chat (one-to-one and one-to-many), view chat history, view active users, and receive status updates of client login/logout.

2.2 Testing Schedule

Testing will be done from 5/30/2017 to 6/8/2017 (release date). Basic testing has been done since implementation started after design. Login and chat will be tested first. Then all other functionality will be confirmed.

2.3 Release Criteria

Criteria for the project is to allow login/register, one-to-one chat, one-to-many chat, view chat history and also have activity messages/updates regarding when users login and logout. Must also be complete by the final deliverables date of 6/8/2017.

3 Unit Testing

Unit tests for this project will correspond to the main modules for which the project was required to implement. These main modules include: register, login, chat, and chat history.

3.1 Register Module

Overview:

The project requirements specify the need for users to be able to register a user that is not currently in the application. Minimum criteria for release is for users to be able to register on first use of application.

Test Data:

Register different types of users with the application which must be for both users in the system and not. Input will involve a UI with both a username and password text fields which need to be populated for successful connection.

Positive Test Cases:

Input data: Register user not currently connected/stored in the system.
Expected result: Successful connection, user saved into application.

Negative Test Cases:

Input Data: Register user who is already in the system
Expected Result: Failure to connect, message to user

Input Data: Attempt to connect/register with empty text fields.
Expected Result: Failure to connect, message to user

This unit will pass the test case when a user is able to successfully connect with new credentials to the system (not a current user).

3.2 Login Module

Overview:

The project requirements specify the need for users to login to the system. (Will also cross with the register test cases partially from the unit above). Minimum criteria for release are users will only be able to log in with proper credentials.

Test Data:

Login different users with the application. The test cases will need to involve users in the system and not in the system along with improper data. Input will involve a UI with both a username and password text fields which need to be populated for successful connection.

Positive Test Cases:

Input Data: Login user who has successfully connected/stored in system
Expected result: Successful connection

Input Data: Login user who has not connected to system
Expected Result: Successful connection, user registered and stored into system

Negative Test Cases:

Input Data: Login user that is in the system, but attempt with incorrect credentials
Expected Result: Failure to connect, message to user (incorrect password message)

Input Data: Attempt to login with empty text fields (either password, username or both).
Expected Result: Failure to connect, message to user

This unit will pass the test case when a user is able to successfully connect with correct credentials (current user).

3.3 Chat Module

Overview:

The project requirements specify the need for users to be able to chat with other clients. The two main parts of this unit are one-to-one chat and one-to-many (group) chat. Minimum criteria for release are users will be able to send to all other clients at once through one-to-many or to select a user and send only them a message.

Test Data:

Use the text box from the client GUI to send messages from one client to another. The test cases will need to involve sending messages to the group chat (one-to-many) and to individual clients (one-to-one). Input will involve an interaction with a primary GUI and a secondary one GUI for the individual chat.

Positive Test Cases:

Input Data: Send message from one client to others via the group chat
Expected result: Successful message delivery to connected clients

Input Data: Start chat with a logged in user by double clicking name and send them a message.
Expected Result: Successful connection, window opens and message only sent to that one user

Negative Test Cases:

Input Data: Attempt to send message to user not connected
Expected Result: Impossible, only can chat with active users.

Input Data: Attempt to send message with empty text fields (either to group or individual).
Expected Result: No action occurs

This unit will pass the test case when a user is able to chat to the group and to another individual active client and also view replies.

3.4 Chat History

Overview:

The project requirements specify the need for users to view their previous chat history in the application. History is only saved once a user logs out of the system. Minimum criteria for release is that a user will only be able to view history if they have connected previously.

Test Data:

View history of clients who have connected to the system. The test cases will need to involve users in the system and not in the system. History will be displayed after selection from the GUI into a new frame.

Positive Test Cases:

Input Data: View history of user who has connected and disconnected at least once
Expected result: History window opens and displays file

Negative Test Cases:

Input Data: View history of user who has not connected to the system and disconnected at least once (first time logged in)
Expected Result: Failure to retrieve history, display message

This unit will pass the test case when a user is able to successfully view their previous chat history if it exists and display appropriate message if not.

3.5 Multiple Clients

Overview:

The project requirements specify the need for multiple clients to be simultaneously connected to the server. This should be evident through the other test cases of login and chat, but can be demonstrated further. Minimum criteria for release is more than one client can connect.

Test Data:

Connect with multiple clients. Send chats to the group chat to demonstrate that simultaneously connection is displayed.

Test Cases:

Input data: Login with 3 or more users. Send messages to the group chat.

Expected result: Successful connections, messages sent to all client group chat text areas.

This unit will pass the test case when multiple clients are able to connect simultaneously.

4 Integration Testing

The purpose of this section is to demonstrate proper communication between each object of the project. Testing will involve interaction with the GUIs, connection with the server, user login data, and chat history data.

4.1 Login Frame

Overview:

The application has a class dedicated to the frame displayed to the user once the application has begun running. This login frame will allow the user to connect to the server. Minimum criteria for release is that the client will be able to login with new or saved credentials that the server will verify.

Process:

The login frame consists of:

Text frame for username

Text frame for password

Login button

Username and password boxes must be non-empty strings. Once the login button has been selected the expected result is to either accept connection then show the main chat frame or to reject connection and display message. This integration will be further explained in 4.4 for the login database.

Login button will attempt to connect to the server if the text fields are populated. Successful connection will then start a new client object which has been connected to the server and show that there is proper data flow from client to server and login database.

Test: Enter proper credentials (connection success), enter improper credentials (error message), and attempt to login with empty fields (no action).

This integration test will demonstrate proper communication between the server, client and login database when a user registers or connects with correct credentials.

4.2 Main Chat Frame

Overview:

The application has a class dedicated to the frame displayed to the user once successfully connected to the server. This frame will allow the user to select other clients to chat with, send message to all connected clients, or to view their chat history. Minimum criteria for release is that the group chat displays messages from all users; can select any active users in the user frame; can view chat history if it exists and also logout.

Process:

The main chat frame consists of:

- Text frame for message
- Text frame for current chat state
- Display of users currently connected to server
- Button to send message
- Button to view chat history

Button to send message must be paired with a non-empty string in the message text field. Once this is met the message should be successfully sent to all other clients connected and displayed in the text frame for the chat state.

Test: Sending non-empty string (server delivers to other clients). Sending empty-string (no action).

The chat history button will open another frame if the user has a chat history in the system (if not, an error message will be displayed when selected).

Test: Selecting this button when a user has a history (successful display on client). Selecting when the user does not have a current history (error message sent from server).

Display of users will update with the username once a user connects and disconnects (communication with the server).

Test: Login with user (server updates the list for each client). Logout with user (server updates the list for each client).

This integration test will demonstrate proper communication between the server and other clients. When chats are sent from standard input of one client to all others communication has been proven to be a success.

4.3 Chat History

Overview:

The application has a class dedicated to storing/retrieving the chat history for each user. This will allow the server to respond to this selection and then display the chat to the user. Minimum criteria for release are users can view their chat history when the server finds a file that matches the user profile.

Process:

The history will be displayed in its own chat frame with no elements besides the text area for display of the history.

From the main chat frame, users can select the chat history button.
Test: View history when user has logged in and logged out once (history class will check for history, send to server then server will send to the client). View history when the user has not logged in previously (history class will check for history, server will send message to the client that history does not exist).

This integration test will demonstrate proper communication between the server and client. When history exists and a user selects, then a confirmation will be sent to the client and history will be displayed.

4.4 Login Database

Overview:

The application has a class dedicated to storing/retrieving the login information for users. This will allow the server to respond to username and password entries from the login frame. Minimum criteria for release are users will only be able to log in with proper credentials.

Process:

Login information will be entered into the login frame from the client, sent to the databased to be checked.

Test: Enter correct credentials (client sends data to server, server calls the login database, and confirms entry, server accepts client). Enter incorrect/empty string (client sends data to server, server calls the login database, denies entry, server does not accept client).

This integration test will demonstrate proper communication between the server, client, and login database. When a user enters correct credentials then the server will allow the client connection.

4.5 One to One Chat Frame

Overview:

Once a user has selected to start a chat with another active client, a new chat frame will open allowing communication between the two. Messages can be sent back and forth until window is closed. Minimum criteria for release are users will be able to have a new chat frame for chat between only one other client through the server.

Process:

The chat frame will consist of a text field for sending the messages.

Test: Enter message into text field and enter (client will send message, server will parse and send to correct client, other client will display if chat window is open). Enter empty string into text field and enter (no action).

Successful tests of this will show integration between the clients and servers.

This integration test will demonstrate proper communication between client to client. When another user is logged in and they are double clicked, a message can be sent from one user and only displayed when they also have opened that connection. Successful result will be messages only visible between those two clients.

4.6 Login/Logoff Messages

Overview:

When a user connects or disconnects from the system, a message will be displayed to other active clients. Minimum criteria for release is that when a user logs in or out a message will be displayed in the primary chat for each client.

Process:

The main chat frame will update in the text field with a message (LOGIN: <user>, LOGOUT: <user>) when an action occurs.

Test: Login with proper credentials (client sends to server which will check login database, once connection accepted the login message will be sent to all other active users. Logout (client sends message to server which disconnects and server sends message to all other active clients.

Successful tests of this will show integration between the clients, login database, and server. The login message will be sent to each of the clients and their proper text frames if passed.

4.7 List of Users Currently Connected

Overview:

When a user connects or disconnects from the system, the lists in each client's main chat frame of active users will be updated. Minimum criteria for release is that when a user connects or disconnects each user's active list in the primary view will update accordingly.

Process:

The main chat frame will update in the user list field with a username when user connects or disconnects. Login will add the username; logout will remove it.

Test: Login with proper credentials (client sends to server which will check login database, once connection accepted the server will notify other clients and those clients will update their user list field). Logout (client sends message to server which disconnects and server sends message to all other active clients which then will update their user list fields).

Successful tests of this will show integration between the clients, login database, and server. The login message will be sent to each of the clients and their proper user list frames if passed.