

Kyle Russell - 13831056
JSock Chat
Documentation

CONTENTS

1.0 About

2.0 User Guide

2.1 Connecting to a JSockChat server

2.2 Account management

2.3 Friend management

2.4 Rooms and Chatting

3.0 Technical notes

3.1 Installation and Project structure

3.2 Protocol

1.0 ABOUT

JSockChat is a client and server chat application aimed at demonstrating message passing via TCP communication as well as keeping client data up to date by frequently pushing updates through UDP. The chat application offers both private and group chatting where users can be involved in many conversations simultaneously. JSockChat users maintain an account which can be created and signed into when using the application. Additionally, users may have friends in a JSockChat server which they can track, manage and private message.

2.0 USER GUIDE

This section will offer a general overview of the JSockChat client and its features. It includes instructions on connecting to a server, creating and logging into a JSockChat account, managing user friends and chatting rooms.

2.1 Connecting to a JSockChat server

Before connecting, start up the JSockChat client from the executable in the 'jsockchat-client' directory. Note: this executable requires elevated privileges in some systems and it may be necessary to start the executable from a terminal using `java -jar jsockchat-client.jar`

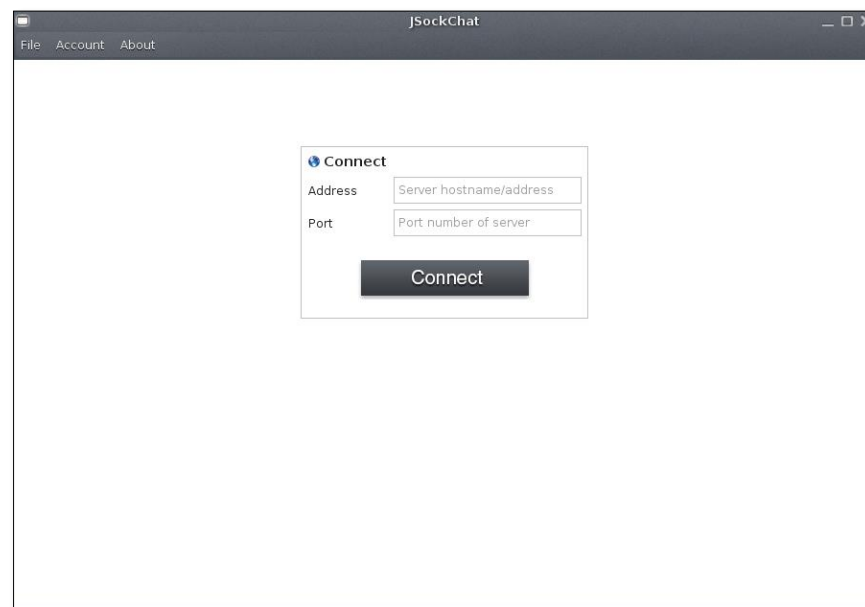


Figure 1: JsckChat connection view

Once the client is started, the connection interface shown in fig.1 will be displayed. In the form provided, you will be required to enter the address and port details of the JSockChat server you wish to connect to. The address of the server is the servers IP or hostname while the port is the port number the connecting server is listening for. Note: if the server is outside of your local

network, these ports will need to be forwarded. Once you have entered the server's address and port details in the fields provided, you can attempt to connect to the server by clicking the 'Connect' button below the form. If the connection is successful you will be transferred to a new view for logging in. Otherwise, the corresponding server will issue an error message which may suggest the server details entered are incorrect or the server is unavailable.

2.2 Account management

Once connected to a JSockChat server, you will need to sign-in to an account created on that server. This can be done through the login view shown in fig.2. Once you have entered your credentials, click the 'Login' button to attempt to sign-in. If the authentication is successful, you will now be connected to the server under this account and you will be forwarded to the home view. In the event that authentication fails, you will be notified and prevented from entering the home view. If this occurs, your credentials may be incorrect, the server could be offline or you may already be signed-in.

To logout of your JSockChat account, select the 'Logout' option from the 'Account' menu. You will be transferred back to the login view.

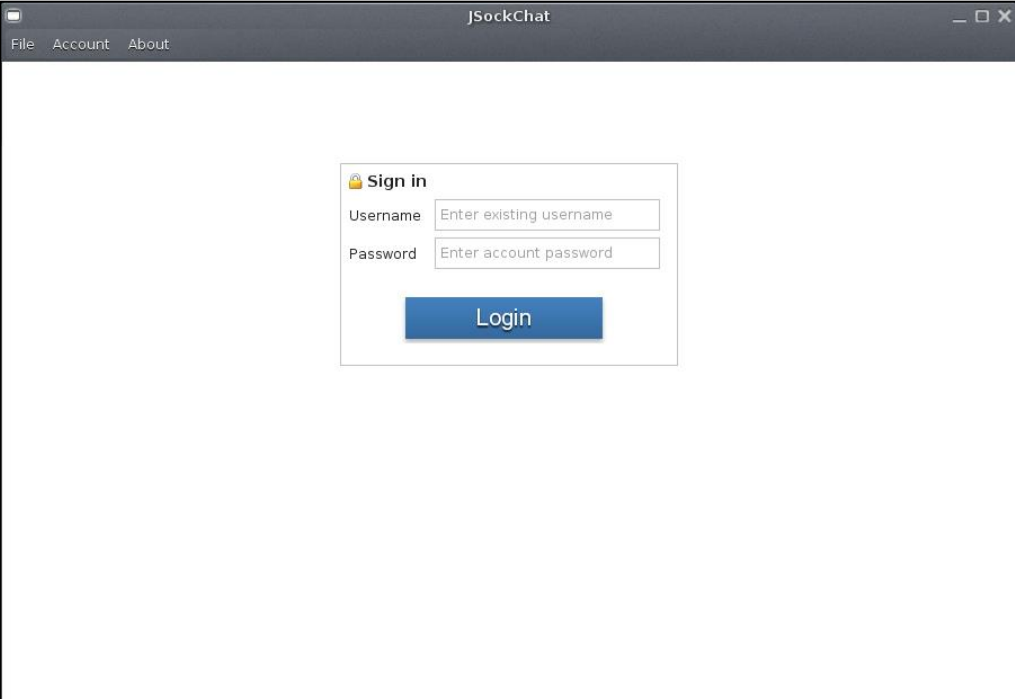
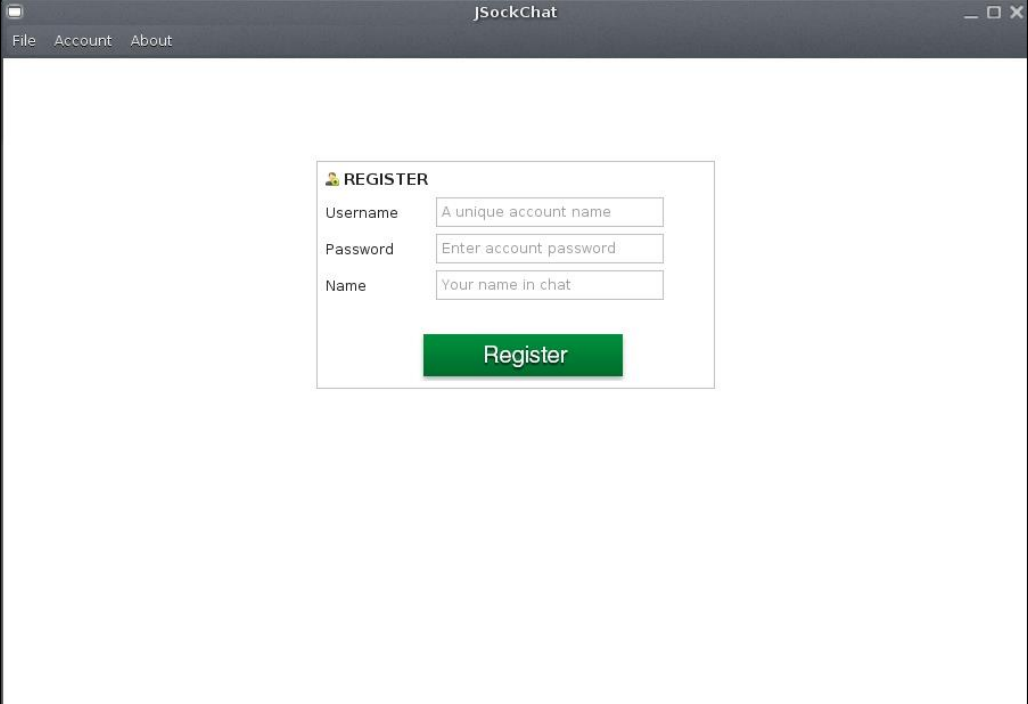
The image shows a screenshot of a web application window titled "JSockChat". The window has a dark grey title bar with standard window controls (minimize, maximize, close) on the right. Below the title bar is a menu bar with three items: "File", "Account", and "About". The main content area is white and contains a "Sign in" form. The form has a title "Sign in" with a small lock icon. It contains two input fields: "Username" with the placeholder text "Enter existing username" and "Password" with the placeholder text "Enter account password". Below these fields is a blue button with the text "Login".

Figure 2: Login view

To create a new account you must use the register view by going to the top 'Account' menu and selecting the 'Register' item. The register view is shown in fig.3. In order to complete the registration, the fields 'Username', 'Password' and 'Name' must be entered. Your username will be a unique ID that is used on sign-in and similarly, your account password should be strong and noted for sign-in. The account name is the name viewed by other users in the chat and is not unique. Once all fields have been filled out, click the 'Register' button to complete the registration. Should the registration be successful, you can now log in via the login view page.



The screenshot shows a web application window titled "JSocketChat". The window has a dark grey header bar with a menu containing "File", "Account", and "About". The main content area is white and contains a registration form. The form is titled "REGISTER" with a small user icon. It has three input fields: "Username" with the placeholder text "A unique account name", "Password" with the placeholder text "Enter account password", and "Name" with the placeholder text "Your name in chat". Below these fields is a green "Register" button.

Figure 3: Register view

2.3 Friend management

JSocketChat allows users to have friends, track their online status and chat directly via private message/chat. To add a friend click the 'Add' button in the 'Friends' tab in which you will be prompted a dialog for a friend's username as shown in fig.4

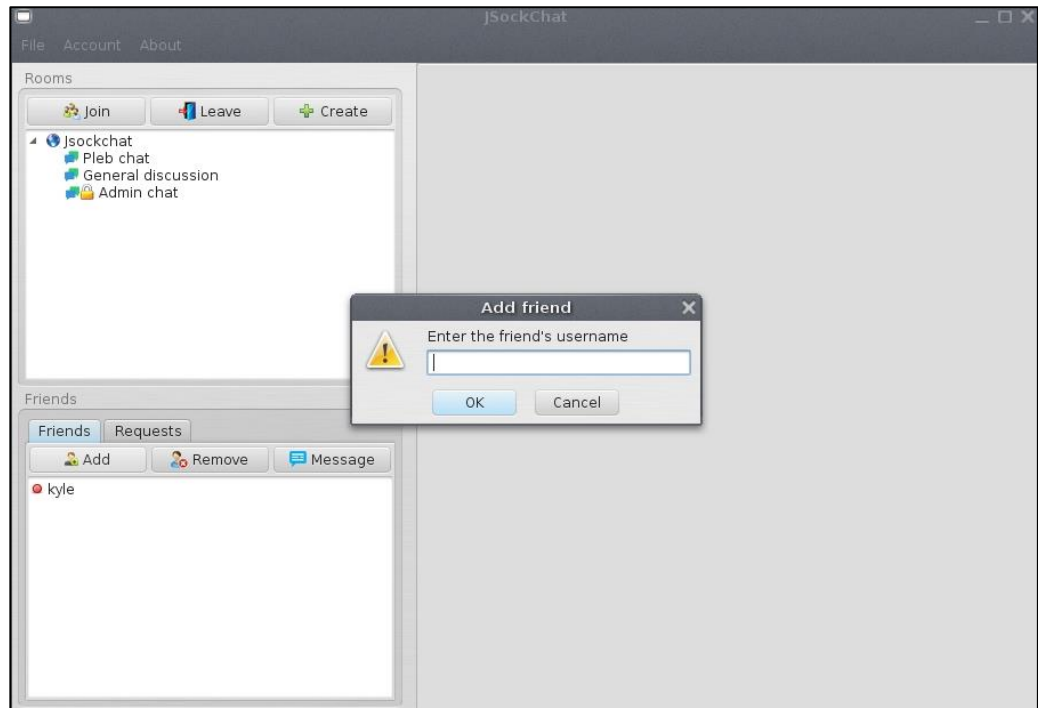


Figure 4: Home view, adding friend

In this dialog, enter your friend's unique ID. You cannot add a friend where a friendship request exists, user ID doesn't exist or the user ID is of your own. Once entered, the requested friend will be notified of your friend request if they are online, otherwise they will see the request on their next login. To view current friend requests click the 'Requests' tab.

To remove a friend, select a friend from the friends list under the 'Friends' tab and click the 'Remove' button. If successful, the friend will be removed from your friend list.

To directly chat with a friend, click the 'Message' button in the 'Friends' tab. If the user is online, a new chat window will be opened on the right pane. If you are already chatting with the friend then the current chat will move to the corresponding private chat. While in a private chat with a friend, the room is not visible to others and the room is available only to the pair of friends.

2.4 Rooms and Chatting

In previous sections it has been mentioned the functionality of private messaging. JSockChat also offers public chat rooms for communication of more than two users. Rooms can be created, left and joined. Additionally, JSockChat offers protected rooms in which rooms may have a password, blocking unwanted users.

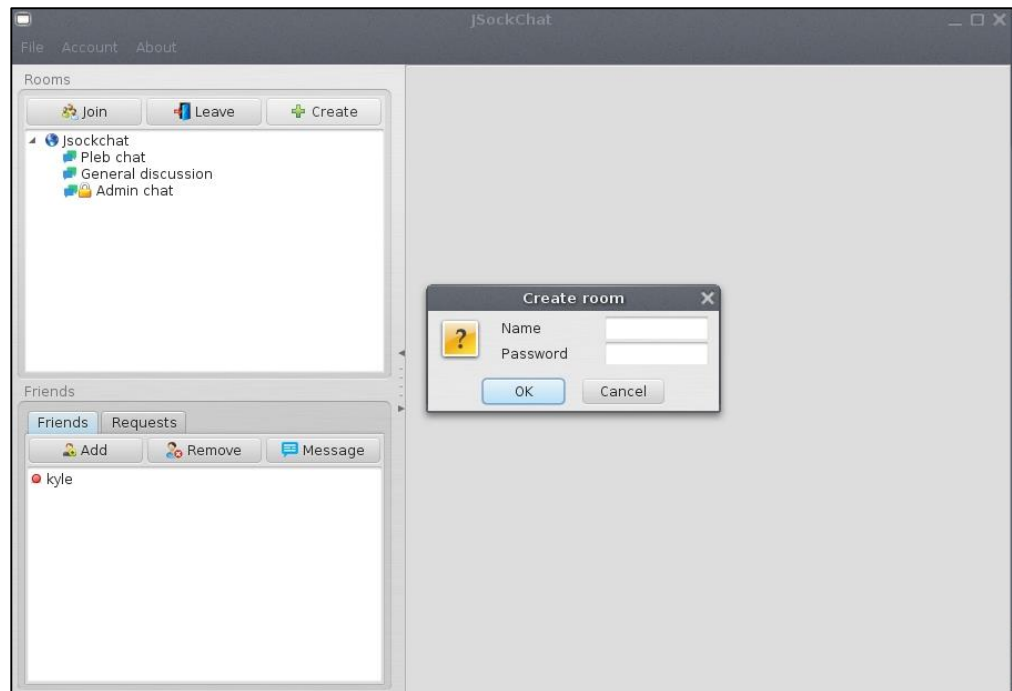


Figure 5: Creating room

To create a room as shown in fig.5, click the 'Create' button in the 'Rooms' panel. You will be prompted with a dialog requesting the room name and password. The room name is visually displayed and uniquely identifies the room. The password field is optional, and omitting the password will allow all users to enter. By specifying a password, users will be required to enter the correct password in order to join the room.

To join a room, select a room from the room tree and click the 'Join' button on the 'Rooms' panel. If the room is protected, you will be prompted to enter the room's password. If you are successful in joining the room, a new chat window will be added to the right pane of the window as shown in fig.6. Users in the room will be notified and their lists will be updated when you join.

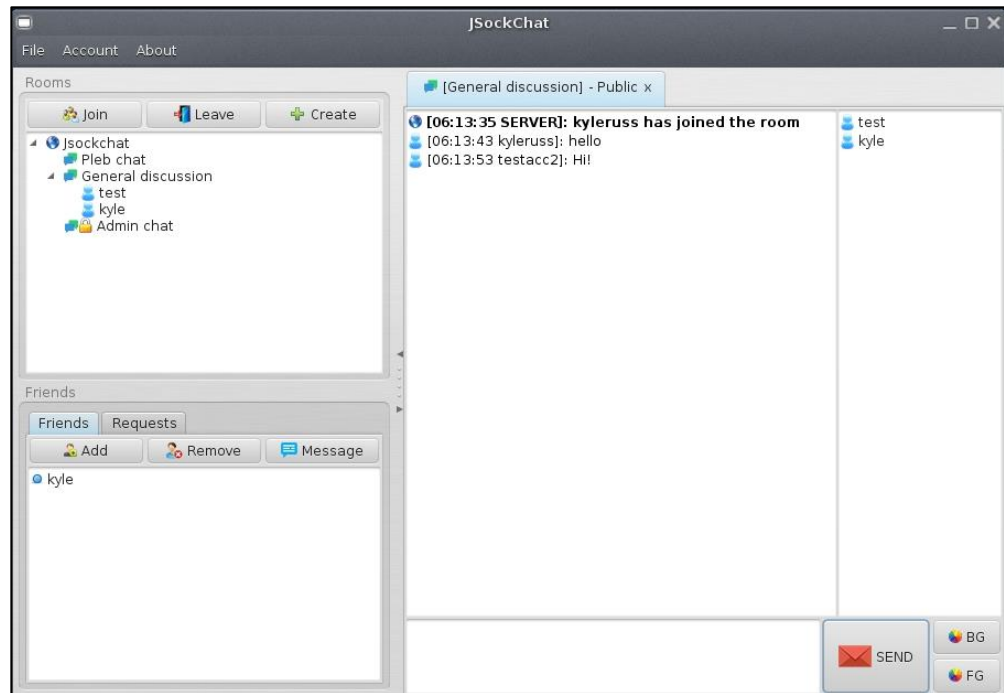


Figure 6: Room chat

To send a message in the chat room, enter your message in the area left of the send button. When you are ready to send the message, click the 'Send' button and users of the room will observe your message.

Chat rooms may also be customized to your preference of text foreground & background. To change these click either the 'FG' or 'BG' buttons right of the 'Send' button where you will be prompted to select a new colour.

To leave a room select the room in the room tree and click the 'Leave' button on the room's panel. Additionally, you can click the 'x' button on the right of chat tabs which will close the chat window and remove you from the room.

3.0 TECHNICAL NOTES

3.1 Installation and Project structure

JSockChat is broken up into three projects, 'jsockchat-server', 'jsockchat-client' and 'jsockchat-commons'. The server component in the 'jsockchat-server' artefact is responsible for listening and handling all communication from clients as well as managing the database of the application. The client project is the face of the application that user's interact with. The 'jsockchat-commons' project builds the foundation of the messaging protocol described in 3.2 and is used by both server and client components as a dependency. Project libraries for local dependencies and repositories can be found in the data/lib directories. Additionally, project resources are located in the data/resources directory for client and server projects.

The server component utilizes a SQLite database to manage user and friend data. The database maintains two tables 'Users' and 'Friends' that are in a 1:M relationship where the 'Users' table holds user account records that includes fields for username, password and display name while the friends table manages friendships between any two users.

Each of the three projects are offered as maven projects rather than class libraries. As such, building either of the projects requires maven and if it is being built in an IDE, then the IDE must support maven projects (i.e. Netbeans maven plugin). Additionally, the projects can be executed from their pre-built jars that have their dependencies packed inside.

3.2 Protocol

JSockChat client/server communication is achieved by passing a MessageBean to either ends. MessageBean's are JavaBean's in the jsockchat-commons 'message' package that carry communication data such as room names, users etc. Additionally, they serve as a 'marker' such that the receiver can identify the type of MessageBean and handle it appropriately. Since jsockchat-commons is a dependency for the client and server, each side only needs to interpret an incoming MessageBean and handle it. Outgoing messages from the client to the server are created from a RequestMessage which wraps a MessageBean that carries the data and gives context to the client's request. When a client sends a request to the server and expects some result, the server will reply with a response. Responses are created in the server as a ResponseMessage which wraps a RequestMessage and maintains the status of the server's response and the response content for the request.

The server and client(s) listen for incoming messages and then handle the interpreted messages. For the client, classes in the client's message package perform an action when given a ResponseMessage. Additionally, an action can be performed for witnesses (users that aren't the sender of the request). The server follows a similar approach but handles requests directly from the

request source and the server's action for this response may be to perform some task (e.g. register an account) and send a reply to the client.

The server also periodically pushes updates to connected clients via UDP. The updates include the online user list, friend lists and room lists. Similar to the MessageBean, the update server pushes UpdateBean's to clients which contains the updated data and timestamps. When a client receives an UpdateBean from the server it then initializes it and performs an action relevant to the type of UpdateBean sent.