**STEVECHAT CHAT RELAY SYSTEM - USER MANUAL**

**Calvin Nefdt - 207524322**

**Darren Osterloh - 209501289**

**Peter Pretorius - 210543717**

**Aimee Booysens - 210501411**

**Tristan Goring - 209520971**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**INDEX**

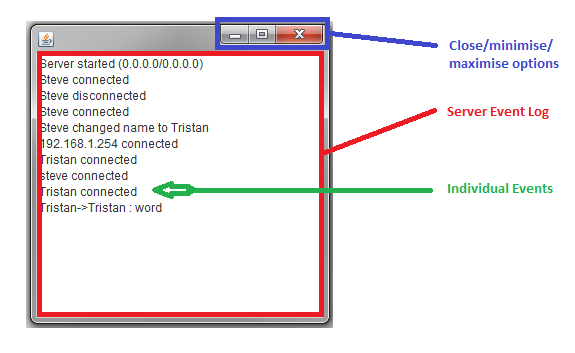
HOW TO USE ... Page 3

OTHER FEATURES ... Page 4

CLASS DESCRIPTIONS ... Page 5

**HOW TO USE**

Clients connect to a server, either hosted locally, or on another computer on the network/web. Essentially, the server has to be live before any clients can connect. Once a server is running, a server event log is started that holds information on all events that take place on the server (i.e. Name changes, clients connecting/disconnecting, whispers(private Messages, etc).



Once the client launches his/her own client instance, a UI pops up, and the client must connect to a server before chatting can begin. The client side chat log will let the client know whether or not he/she connected to the desired server. The clients name is defaulted to "Steve" and can be changed, using the "/nick" command. Once connected, the client chat log will show the client all events from/to any other client, beside that of whispers (private messages) from other clients to any other client.

New clients can learn more about the version of the chat system they are using, by clicking on the "Help" tab, and selecting the "About" option. Clients too can learn more about how to use the chat system by going to the "Help" tab, and selecting the "Help" option. "Connect/Disconnect", "Configure" and "Exit", can be found in the "File" tab.

All online clients can view public messages from every online client, from the time the client connects to the server, as well as notifications of all other clients that connect to the server.

Clients, other than public messages, can use the available commands to utilise the functionality of the chat system. Here is a list of all available commands:

* /connect: Connects to the server.
* /disconnect: Disconnects from the server.
* /nick {new name}: Changes the clients name to the one specified. Note that no duplicate names are allowed.
* /msg {user} {message}: Sends a private message to another user
* /me {action}: Broadcasts a message to all users emoting the specified action
* /whois {user}: Returns the IP Address of target user
* /help: Opens the readme.txt
* This list can be viewed in the client by typing /cmdlist

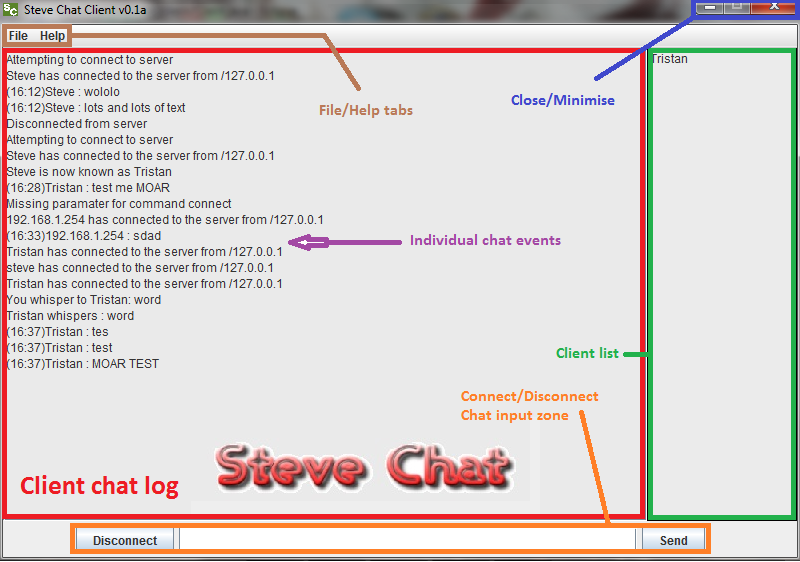
**OTHER FEATURES**

Message history

The client stores all previously sent messages. This list can be traversed by the use of the UP/DOWN arrow keys while the message field has focus. UP will go back through previous messages, and DOWN will iterate forward through past messages.

Tab completion

The client can attempt to complete partial usernames with the use of the TAB key while the message field has focus. For example, if a user has the username “Steve7912401250” and you wish to message them, then simply typing “/msg Ste” and pressing TAB will auto complete the input to “/msg Steve7912401250 ”. Note that there is a 3 character minimum for the partial name.



**CLASS DESCRIPTIONS**

* **ChatRelayServer :** This class handles all server side aspects of the program, deals with incoming messages and commands and sends the appropriate response to all clients currently connected. All input that the server receives through the input streams are in the form of a serializable object and everything that it outputs are in the form of Strings. All error handling is displayed in a debugger window for the server to monitor and all chat logs are stored as text files.
* **MessageHandler :** This class handles the messages that are sent from the client to the server as well as those sent from the server to the client. The message that is sent from the client is checked to see if it is a normal message or a command. If it is a command it is sent to the server which will implement the command instruction relative to the command call. Otherwise, if it is a normal message then a string representation of both your username and message is sent to the server to be broadcast to the clients who are currently logged into the server.
* **Client**: This class provides a GUI interface for client-side activity that sends and receives messages via the MessageHandler. A list of connected users is kept client-side and updated by the server whenever another user connects or disconnects. Supports a multitude of commands, all of which need to be validated by the server.