htl-tk-chat 1.0.0-rc.1

Generated by Doxygen 1.9.1

1 htl-tk-chat	1
1.1 Build documentation	. 1
1.2 Project description	. 1
1.2.1 Planned features:	. 1
1.3 What we use, what we do	. 2
1.4 Rules	
1.5 Milestone 1	. 2
2 Versioning scheme	3
2.1 Majour	. 3
2.2 Minor	. 3
2.3 Release	. 3
3 Namespace Index	5
3.1 Namespace List	. 5
4 Hierarchical Index	7
4.1 Class Hierarchy	. 7
5 Class Index	9
5.1 Class List	_
	. ,
6 Namespace Documentation	11
6.1 backend Namespace Reference	
6.1.1 Detailed Description	
6.2 chat_client Namespace Reference	
6.2.1 Detailed Description	
6.2.2 Function Documentation	
6.2.2.1 main_read_loop()	
6.2.2.2 main_write_loop()	. 12
6.2.3 Variable Documentation	
6.2.3.1 read_thread	
6.2.3.2 write_thread	
6.3 chat_server Namespace Reference	. 13
6.3.1 Detailed Description	. 14
6.3.2 Function Documentation	. 14
6.3.2.1 image_message()	. 14
6.3.2.2 send_connected_users()	. 14
6.3.2.3 send_encoded()	. 14
6.3.2.4 setup_server()	. 15
6.3.2.5 text_message()	. 15
6.3.3 Variable Documentation	. 15
6.3.3.1 server	. 15
6.3.3.2 ssl_server	. 15
6.4 client Namespace Reference	. 16

6.4.1 Detailed Description	 16
6.4.2 Function Documentation	 16
6.4.2.1 main_read_loop()	 16
6.4.2.2 send_msg()	 17
6.4.2.3 start_read_loop()	 17
6.4.2.4 stop_read_loop()	 17
6.5 client_functions Namespace Reference	 17
6.5.1 Detailed Description	 18
6.5.2 Function Documentation	 18
6.5.2.1 authenticate()	 18
6.5.2.2 check_ssl()	 18
6.5.2.3 close_connection()	 18
6.5.2.4 get_message()	 19
6.5.2.5 get_ssl_cert()	 19
6.5.2.6 image_message()	 19
6.5.2.7 recv_msg()	 19
6.5.2.8 recvall()	 20
6.5.2.9 send_encoded()	 20
6.5.2.10 text_message()	 20
6.6 database_functions Namespace Reference	 20
6.6.1 Detailed Description	 21
6.7 settings_functions Namespace Reference	 21
6.7.1 Detailed Description	 21
6.7.2 Function Documentation	 21
6.7.2.1 connect_server()	 21
6.7.2.2 delete_frontend_config()	 22
6.7.2.3 disconnect_server()	 22
6.7.2.4 init_settings_window()	 22
6.7.2.5 load_frontend_config()	 22
6.7.2.6 save_frontend_config()	 23
6.8 user_functions Namespace Reference	 23
6.8.1 Detailed Description	 23
6.8.2 Function Documentation	 23
6.8.2.1 display_message()	 23
7 Class Documentation	25
7.1 backend.Client Class Reference	
7.1.1 Detailed Description	
7.2 database_functions.Database Class Reference	
7.2.1 Detailed Description	
7.2.2 Member Function Documentation	
7.2.2.1 get_message()	
7.2.2.2 init_db()	
7.2.2.2 IIIIL_UD()	 ۷۷

7.2.2.3 write_message()	26
7.3 client.MainWindow Class Reference	27
7.3.1 Detailed Description	27
7.4 backend.SSL_Client Class Reference	27
7.4.1 Detailed Description	27
7.5 chat_server.SSL_TCPServer Class Reference	28
7.5.1 Detailed Description	28
7.6 chat_server.SSL_ThreadedTCPServer Class Reference	28
7.6.1 Detailed Description	29
7.7 chat_server.ThreadedTCPRequestHandler Class Reference	29
7.7.1 Detailed Description	29
7.7.2 Member Function Documentation	29
7.7.2.1 handle()	30
7.8 chat_server.ThreadedTCPServer Class Reference	30
7.8.1 Detailed Description	30
7.9 mainwindow.Ui_MainWindow Class Reference	30
7.9.1 Detailed Description	31
7.10 settingswindow.Ui_SettingsWindow Class Reference	31
7.10.1 Detailed Description	32
Index 3	33

# htl-tk-chat

Watch the development progress at <a href="https://gitea.escpe.net/cc69222/htl-tk-chat">https://gitea.escpe.net/cc69222/htl-tk-chat</a>

If you are seeing this repo on Github note that this is only a mirror.

### 1.1 Build documentation

To build the documentation run: make doc

After this you can open the documentation.pdf.

For lazy people or Windows folks, a precompiled version of the documentation can be found here.

## 1.2 Project description

A Server and client for a self-written chat application. The client interface uses Qt 5, the server uses sockets and msgpack.

### 1.2.1 Planned features:

- Selecting users to send messages to ("Public", "Private")
- Notify users when new messages are available (for them)
- · database to store message history
- · attachments (files)
- · embedded pictures in chat
- authentification / identification of users via username
- encryption ? (GPG) or SSL -> Let's Encrypt for server

2 htl-tk-chat

### 1.3 What we use, what we do

- · sockets, tcp streams
- · JSON for message content and metadata
- · msgpack for message content and metadata, because it is faster and encodes in binary instead of strings.
- Server, self written, hosted at home (port forwarding)
- · Database server-side for message-history, users, authentification
- For the Database SQLite, cause it is a lightweight disk-based database and doesn't require a separate server process. Also the sqlite3 python module is in The Python Standard Library.
- We don't care about networking failures, we let tcp handle that.
- Interface in tkinter or Qt 5 (via PyQt5)
- · Doxygen for documentation

### 1.4 Rules

- · Commit messages should make sense
- If you're planning to implement a new feature, open an issue with the correct label
- Working on the development branch and after the one week sprint make a pull request, to merge it back to the master.
- · Write sensible reviews and document your Code

### 1.5 Milestone 1

- · Server is running and accessible from the internet
- Clients send messages to Server, the Server broadcasts the message to all Clients.

# Versioning scheme

The htl-tk-chat uses a 3 Part Decimal Versioning Sytem, consisting of majour, minor and release. The version is incremented as follows:

## 2.1 Majour

A Majour release is announced when majour rebuilds ocur. This indicates a change may break compatiability in one way or another.

### 2.2 Minor

A minor release happens when the host gains a new feature. This does not indicate any sort of regression and regressions within a majour release are most likely accidental. If backwards compatiability is not given, please create a bug-report.

### 2.3 Release

A release is pushed out when bug-fixes occur, or already existing features are improved in a way that doesn't affect their function.

Versioning scheme

# Namespace Index

# 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

ackend	11
nat_client	11
nat_server	13
ient	16
ient_functions	17
atabase_functions	20
ettings_functions	21
ser functions	23

6 Namespace Index

# Hierarchical Index

# 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ocketserver.BaseRequestHandler	
chat_server.ThreadedTCPRequestHandler	<b>2</b> 9
packend.Client	25
database_functions.Database	25
object	
mainwindow.Ui_MainWindow	30
settingswindow.Ui_SettingsWindow	31
packend.SSL_Client	27
socketserver.TCPServer	
chat_server.SSL_TCPServer	28
chat_server.SSL_ThreadedTCPServer	28
chat_server.ThreadedTCPServer	30
socketserver.ThreadingMixIn	
chat_server.SSL_ThreadedTCPServer	28
chat server.ThreadedTCPServer	
QMainWindow	
client.MainWindow	27

8 Hierarchical Index

# Class Index

# 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

backend.Client
database_functions.Database
client.MainWindow
backend.SSL_Client
chat_server.SSL_TCPServer
chat_server.SSL_ThreadedTCPServer
chat_server.ThreadedTCPRequestHandler
chat_server.ThreadedTCPServer
mainwindow.Ui_MainWindow
settingswindow.Ui_SettingsWindow

10 Class Index

# Namespace Documentation

# 6.1 backend Namespace Reference

### Classes

- class Client
- class SSL\_Client

### **Functions**

- def read\_config ()
- def init\_backend (ip=host, port=port, username=user, en\_ssl=en\_ssl)

### **Variables**

- **dir** = os.path.dirname(\_\_file\_\_)
- my\_client = None
- cconfig = None
- string **host** = "ehw12.ddns.net"
- string **port** = "9999"
- **user** = getpass.getuser()
- bool en\_ssl = False
- · \_create\_default\_https\_context

### 6.1.1 Detailed Description

Chat Client backend

# 6.2 chat\_client Namespace Reference

### **Functions**

- def main\_read\_loop (sock)
- def main\_write\_loop (sock, user)

### **Variables**

```
• dir = os.path.dirname(__file__)
```

- bool **shutdown** = False
- string config\_file = "client.conf"
- **cconfig** = configparser.ConfigParser()
- **host** = sys.argv[1]
- **port** = int(sys.argv[2])
- **user** = sys.argv[3]
- read\_thread
- write\_thread

### 6.2.1 Detailed Description

CLI Chat Client

### 6.2.2 Function Documentation

### 6.2.2.1 main\_read\_loop()

Reads the socket in a loop

Definition at line 20 of file chat\_client.py.

### 6.2.2.2 main\_write\_loop()

Sends messages in a loop

Definition at line 38 of file chat\_client.py.

### 6.2.3 Variable Documentation

#### 6.2.3.1 read\_thread

Definition at line 93 of file chat\_client.py.

#### 6.2.3.2 write thread

```
chat_client.write_thread
```

### Initial value:

Definition at line 96 of file chat\_client.py.

## 6.3 chat\_server Namespace Reference

#### Classes

- · class ThreadedTCPRequestHandler
- class SSL\_TCPServer
- · class ThreadedTCPServer
- class SSL\_ThreadedTCPServer

### **Functions**

- def unpack\_message (message)
- def text\_message (sock, str text, str author, str recipient="all")
- def send\_connected\_users (sock, usernames)
- def image\_message (sock, str image, str author, str recipient="all")
- def recv\_msg (sock)
- def recvall (sock, msglen)
- def send\_encoded (sock, message)
- def setup\_server (server)

### **Variables**

- dir = os.path.dirname(\_\_file\_\_)
- list connected\_clients = []
- string log\_conf = "logger.conf"
- config\_file = os.path.join(dir, "server.conf")
- **config** = configparser.ConfigParser()
- **HOST** = config["SERVER"]["listen\_address"]
- **PORT** = int(config["SERVER"]["listen\_port"])
- ssl\_server
- server

### 6.3.1 Detailed Description

The Chat Server

This is the Chat server, it is the backbone of the communication between the clients.

### 6.3.2 Function Documentation

### 6.3.2.1 image\_message()

Image Message function

Takes a socket, image, author and recipient as arguments and sends it to the server. The image will be base64 encoded.

Definition at line 295 of file chat\_server.py.

### 6.3.2.2 send\_connected\_users()

Text Message function

Takes a socket and a iterable usernames as arguments and sends it to the client.

Definition at line 281 of file chat\_server.py.

### 6.3.2.3 send\_encoded()

Helper function to send encoded the message

Definition at line 336 of file chat\_server.py.

### 6.3.2.4 setup\_server()

### 6.3.2.5 text\_message()

Text Message function

Takes a socket, text, author and recipient as arguments and sends it to the client.

Definition at line 265 of file chat\_server.py.

### 6.3.3 Variable Documentation

### 6.3.3.1 server

```
chat_server.server
```

#### **Initial value:**

```
00001 = ThreadedTCPServer(
00002 (HOST, PORT),
00003 ThreadedTCPRequestHandler
00004 )
```

Definition at line 400 of file chat\_server.py.

### 6.3.3.2 ssl\_server

chat\_server.ssl\_server

### Initial value:

Definition at line 389 of file chat\_server.py.

## 6.4 client Namespace Reference

### **Classes**

• class MainWindow

#### **Functions**

- def send msg ()
- def send\_image\_file ()
- def start\_read\_loop ()
- def stop\_read\_loop ()
- def main\_read\_loop (sock)

### **Variables**

- **dir** = os.path.dirname(\_\_file\_\_)
- display\_thread = None
- string **log\_conf** = "logger.conf"
- **app** = QApplication(sys.argv)
- mainwindow = MainWindow()
- mainwindowui = Ui\_MainWindow()
- **settingswindow** = QMainWindow()
- settingswindowui = Ui\_SettingsWindow()

### 6.4.1 Detailed Description

```
The Chat Client
```

This is the chat client with Qt5 frontend.

### 6.4.2 Function Documentation

### 6.4.2.1 main\_read\_loop()

Reads the socket in a loop

Definition at line 120 of file client.py.

# 6.4.2.2 send\_msg()

```
def client.send_msg ( )
Send message function
This function is called when the user presses Enter,
to send the message.
```

Definition at line 45 of file client.py.

### 6.4.2.3 start\_read\_loop()

```
def client.start_read_loop ( )
starts the read loop in a new thread
```

Definition at line 96 of file client.py.

### 6.4.2.4 stop\_read\_loop()

```
def client.stop_read_loop ( )
stops the read loop thread
```

Definition at line 107 of file client.py.

# 6.5 client\_functions Namespace Reference

#### **Functions**

- def authenticate (sock, str username)
- def text\_message (sock, str text, str author, str recipient="all")
- def image\_message (sock, str image, str author, str recipient="all")
- def get\_message (sock)
- def close\_connection (sock)
- def recv\_msg (sock)
- def recvall (sock, msglen)
- def send\_encoded (sock, message)
- def check\_ssl (sock)
- def get\_ssl\_cert (sock)

### 6.5.1 Detailed Description

client\_functions
Adds functions for the client to use

This module serves as some kind of API for the chat client. The client should never send messages to the server directly, but rather through this api. This makes it easier to programm the GUI later.

### 6.5.2 Function Documentation

#### 6.5.2.1 authenticate()

Authentication function

Takes a socket and a username as arguments and sends it to the server as an auth type message. The auth message has the values user for the username identification and time, to know when the user logs in. A value for the ip address is not necessary, since the socket already knows this.

Definition at line 16 of file client\_functions.py.

### 6.5.2.2 check\_ssl()

Check if the server supports ssl

Definition at line 132 of file client\_functions.py.

### 6.5.2.3 close\_connection()

Close function

Closes the server connection and notifies the server about it.

Definition at line 92 of file client\_functions.py.

### 6.5.2.4 get\_message()

```
\begin{tabular}{ll} def & client\_functions.get\_message & ( \\ & sock & ) \end{tabular}
```

Get Message function

Takes a socket argument and returns the message in the pipe.

Definition at line 78 of file client\_functions.py.

### 6.5.2.5 get\_ssl\_cert()

Check if the server supports ssl

Definition at line 144 of file client\_functions.py.

### 6.5.2.6 image\_message()

Image Message function

Takes a socket, image, author and recipient as arguments and sends it to the server. The image will be base64 encoded.

Definition at line 54 of file client\_functions.py.

### 6.5.2.7 recv\_msg()

Receive a whole message

Definition at line 104 of file client\_functions.py.

### 6.5.2.8 recvall()

helper function which receives all up to msglen

Definition at line 114 of file client\_functions.py.

#### 6.5.2.9 send\_encoded()

Helper function to send encoded the message

Definition at line 124 of file client\_functions.py.

### 6.5.2.10 text\_message()

Text Message function

Takes a socket, text, author and recipient as arguments and sends it to the server.

Definition at line 34 of file client\_functions.py.

# 6.6 database\_functions Namespace Reference

### Classes

• class Database

### 6.6.1 Detailed Description

database functions

Adds database functions for the server to use

This library adds easy to use functions to store message data in a database for (somewhat) persistent data storage.

## 6.7 settings\_functions Namespace Reference

### **Functions**

- def save\_frontend\_config (settings)
- def delete\_frontend\_config (settings)
- def load\_frontend\_config (settings)
- def connect\_server (settings)
- def disconnect\_server (settings)
- def quickConnect (mainui, settingsui)
- def init\_settings\_window (settings)

#### **Variables**

- **dir** = os.path.dirname(\_\_file\_\_)
- string config\_file = "../client/client.conf"
- **config** = configparser.ConfigParser()
- bool connected = False

### 6.7.1 Detailed Description

```
Settings Module functions behind the settings window ui-elements
```

### 6.7.2 Function Documentation

### 6.7.2.1 connect\_server()

Definition at line 78 of file settings\_functions.py.

#### 6.7.2.2 delete\_frontend\_config()

Definition at line 49 of file settings\_functions.py.

### 6.7.2.3 disconnect\_server()

The function is a stub.

button for server connection in the settings window.

Definition at line 105 of file settings\_functions.py.

### 6.7.2.4 init\_settings\_window()

```
{\it def settings\_functions.init\_settings\_window} \ ( \\ {\it settings} \ ) 
 {\it Initialize settings-window} \\ {\it Actions and signals are being connected to their Slots}
```

Definition at line 142 of file settings\_functions.py.

### 6.7.2.5 load\_frontend\_config()

```
\begin{tabular}{ll} def & settings\_functions.load\_frontend\_config (\\ & settings \end{tabular}) \end{tabular}
```

Load the values stored in the .config file for frontend settings

Definition at line 58 of file settings\_functions.py.

### 6.7.2.6 save\_frontend\_config()

## 6.8 user\_functions Namespace Reference

#### **Functions**

- def set\_user\_table (window, users)
- def set\_combo\_box (window, users)
- def display\_message (window, str sender, str recipient, str message)
- def add\_image (window, image)
- def get\_image\_file ()

### 6.8.1 Detailed Description

This file provides functions for the mainwindow to use

### 6.8.2 Function Documentation

### 6.8.2.1 display\_message()

Definition at line 33 of file user\_functions.py.

# Class Documentation

### 7.1 backend.Client Class Reference

### **Public Member Functions**

- def \_\_init\_\_ (self, str ip, int port)
- def connect (self, ip=None, port=None)

### **Public Attributes**

sock

### **Private Attributes**

- \_\_ip
- \_\_port

### 7.1.1 Detailed Description

Intializes the Socket which is available as self.sock

Definition at line 27 of file backend.py.

The documentation for this class was generated from the following file:

client/backend.py

## 7.2 database\_functions.Database Class Reference

### **Public Member Functions**

- def init\_db (self)
- def write\_message (self, message, author, receipient, date)
- def get\_message (self)

26 Class Documentation

### **Private Member Functions**

• def \_\_dict\_factory (self, cursor, row)

### 7.2.1 Detailed Description

This class serves as a de facto database providing functions for simple access to the database.

Definition at line 14 of file database\_functions.py.

### 7.2.2 Member Function Documentation

```
7.2.2.1 get_message()
```

Gets the messages from the database

Definition at line 68 of file database\_functions.py.

### 7.2.2.2 init\_db()

```
\begin{tabular}{ll} \tt def \ database\_functions.Database.init\_db \ (\\ & self \ ) \end{tabular}
```

Initalizes the database

Checks if all the needed tables exist or creates them if not.

Definition at line 28 of file database\_functions.py.

### 7.2.2.3 write\_message()

Writes the current message to the message history

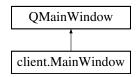
Definition at line 51 of file database\_functions.py.

The documentation for this class was generated from the following file:

 $\bullet \ server/database\_functions.py$ 

### 7.3 client.MainWindow Class Reference

Inheritance diagram for client. Main Window:



### **Public Member Functions**

- def \_\_init\_\_ (self)
- def closeEvent (self, event)

### 7.3.1 Detailed Description

Definition at line 31 of file client.py.

The documentation for this class was generated from the following file:

• frontend/client.py

## 7.4 backend.SSL\_Client Class Reference

### **Public Member Functions**

- def \_\_init\_\_ (self, str ip, int port, ssl\_version)
- def connect (self, ip=None, port=None)

#### **Public Attributes**

- tmpfile
- sock

### **Private Attributes**

- \_\_ssl\_version
- \_\_ip
- \_\_port

### 7.4.1 Detailed Description

The same Client but with SSL support

Definition at line 40 of file backend.py.

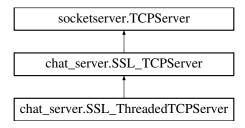
The documentation for this class was generated from the following file:

client/backend.py

28 Class Documentation

### 7.5 chat\_server.SSL\_TCPServer Class Reference

Inheritance diagram for chat\_server.SSL\_TCPServer:



### **Public Member Functions**

def \_\_init\_\_ (self, str certfile, str keyfile, tuple server\_address, RequestHandlerClass, ssl\_version, bind
 \_and\_activate=True)

### **Public Attributes**

socket

### **Private Attributes**

- \_certfile
- \_\_keyfile

### 7.5.1 Detailed Description

A TCP Server with SSL support

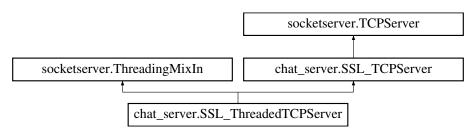
Definition at line 218 of file chat\_server.py.

The documentation for this class was generated from the following file:

server/chat\_server.py

## 7.6 chat\_server.SSL\_ThreadedTCPServer Class Reference

Inheritance diagram for chat\_server.SSL\_ThreadedTCPServer:



### **Additional Inherited Members**

### 7.6.1 Detailed Description

SSL\_TCPServer with threading support. Spawns a new Thread for every response. Has SSL support

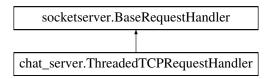
Definition at line 252 of file chat\_server.py.

The documentation for this class was generated from the following file:

• server/chat\_server.py

## 7.7 chat\_server.ThreadedTCPRequestHandler Class Reference

Inheritance diagram for chat\_server.ThreadedTCPRequestHandler:



### **Public Member Functions**

- def handle (self)
- def remove\_user (self, user)

### **Public Attributes**

user

### 7.7.1 Detailed Description

Handles the request in a thread

Definition at line 26 of file chat\_server.py.

### 7.7.2 Member Function Documentation

30 Class Documentation

### 7.7.2.1 handle()

```
{\tt def \ chat\_server.ThreadedTCPRequestHandler.handle \ (} \\ {\tt self \ )}
```

function gets a request offer from the socket and prints the message and the Thread it is in. It handles the authentication and the messages. At the moment it just broadcasts the messages

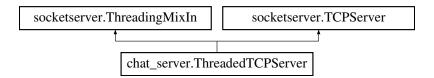
Definition at line 33 of file chat\_server.py.

The documentation for this class was generated from the following file:

server/chat\_server.py

## 7.8 chat\_server.ThreadedTCPServer Class Reference

Inheritance diagram for chat\_server.ThreadedTCPServer:



### 7.8.1 Detailed Description

TCPServer with threading support. Spawns a new Thread for every response.

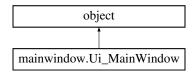
Definition at line 247 of file chat\_server.py.

The documentation for this class was generated from the following file:

• server/chat\_server.py

## 7.9 mainwindow.Ui\_MainWindow Class Reference

 $Inheritance\ diagram\ for\ mainwindow. Ui\_MainWindow:$ 



### **Public Member Functions**

- def setupUi (self, MainWindow)
- def retranslateUi (self, MainWindow)

### **Public Attributes**

- centralwidget
- gridLayout
- horizontalLayout
- userSelect
- InputBar
- addFileButton
- horizontalLayout\_2
- SettingButton
- ConnectButton
- chatList
- · msgList
- actionExit
- actionExport
- actionServer

### 7.9.1 Detailed Description

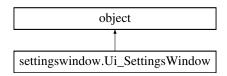
Definition at line 14 of file mainwindow.py.

The documentation for this class was generated from the following file:

· frontend/mainwindow.py

# 7.10 settingswindow.Ui\_SettingsWindow Class Reference

Inheritance diagram for settingswindow.Ui\_SettingsWindow:



### **Public Member Functions**

- def **setupUi** (self, SettingsWindow)
- def retranslateUi (self, SettingsWindow)

**32 Class Documentation** 

### **Public Attributes**

- centralWidget
- gridLayout
- MainButtonsLayout
- ButtonSaveProfile
- ButtonLoadProfile
- ButtonDeleteProfile
- MainTabBar
- ServerTab • gridLayout\_2
- AuthLayout
- InputServerAddress
- InputPort
- InputUsername
- horizontalLayout
- SSLCheckBox
- ConnectionButtons
- ButtonStartConnection
- ButtonEndConnection

### 7.10.1 Detailed Description

Definition at line 14 of file settingswindow.py.

The documentation for this class was generated from the following file:

• frontend/settingswindow.py

# Index

authenticate	database_functions.Database, 25		
client_functions, 18	get_message, 26		
	init_db, 26		
backend, 11	write_message, 26		
backend.Client, 25	delete_frontend_config		
backend.SSL_Client, 27	settings_functions, 21		
	disconnect_server		
chat_client, 11	settings_functions, 22		
main_read_loop, 12	display_message		
main_write_loop, 12	user_functions, 23		
read_thread, 12			
write_thread, 13	get_message		
chat_server, 13	client_functions, 18		
image_message, 14	database_functions.Database, 26		
send_connected_users, 14	get_ssl_cert		
send_encoded, 14	client_functions, 19		
server, 15			
setup_server, 14	handle		
ssl_server, 15	chat_server.ThreadedTCPRequestHandler, 29		
text_message, 15			
chat_server.SSL_TCPServer, 28	image_message		
chat_server.SSL_ThreadedTCPServer, 28	chat_server, 14		
chat_server.ThreadedTCPRequestHandler, 29	client_functions, 19		
handle, 29	init_db		
chat_server.ThreadedTCPServer, 30	database_functions.Database, 26		
check_ssl	init_settings_window		
client_functions, 18	settings_functions, 22		
client, 16	load frontand config		
main_read_loop, 16	load_frontend_config settings_functions, 22		
send_msg, 16	settings_runctions, 22		
start_read_loop, 17	main_read_loop		
stop_read_loop, 17	chat_client, 12		
client.MainWindow, 27	client, 16		
client_functions, 17	main_write_loop		
authenticate, 18	chat_client, 12		
check_ssl, 18	mainwindow.Ui_MainWindow, 30		
close_connection, 18	mamwindow.or_mamwindow, or		
get_message, 18	read_thread		
get_ssl_cert, 19	chat_client, 12		
image_message, 19	recv_msg		
recv_msg, 19	client_functions, 19		
recvall, 19	recvall		
send_encoded, 20	client_functions, 19		
text_message, 20	_ ,		
close_connection	save_frontend_config		
client_functions, 18	settings_functions, 22		
connect_server	send_connected_users		
settings_functions, 21	chat_server, 14		
	send_encoded		
database_functions, 20	chat server. 14		

34 INDEX

```
client_functions, 20
send_msg
    client, 16
server
    chat_server, 15
settings_functions, 21
    connect_server, 21
     delete_frontend_config, 21
     disconnect_server, 22
    init_settings_window, 22
    load_frontend_config, 22
    save_frontend_config, 22
settingswindow.Ui_SettingsWindow, 31
setup_server
    chat_server, 14
ssl_server
    chat_server, 15
start_read_loop
    client, 17
stop_read_loop
    client, 17
text_message
    chat_server, 15
    client_functions, 20
user_functions, 23
     display_message, 23
write_message
     database\_functions. Database, {\color{red}26}
write\_thread
     chat_client, 13
```