htl-tk-chat 0.0.1

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 Namespace Index	3
2.1 Namespace List	. 3
3 Hierarchical Index	5
3.1 Class Hierarchy	. 5
4 Class Index	7
4.1 Class List	
5 Namespace Documentation	9
5.1 chat_client Namespace Reference	
5.1.1 Detailed Description	
5.2 chat_server Namespace Reference	. 9
5.2.1 Detailed Description	. 9
5.3 client Namespace Reference	. 10
5.3.1 Detailed Description	. 10
5.3.2 Function Documentation	. 10
5.3.2.1 send_msg()	. 10
5.4 client_functions Namespace Reference	. 10
5.4.1 Detailed Description	. 11
5.4.2 Function Documentation	. 11
5.4.2.1 authenticate()	. 11
5.5 database_functions Namespace Reference	. 11
5.5.1 Detailed Description	. 11
5.6 userstub Namespace Reference	
5.6.1 Detailed Description	. 12
5.6.2 Function Documentation	
5.6.2.1 test_combo_box()	. 12
5.6.2.2 test_connection()	
5.6.2.3 test_user_table()	
5.6.3 Variable Documentation	
5.6.3.1 test_users	
6 Class Documentation	15
6.1 database functions.Database Class Reference	
6.1.1 Detailed Description	
6.1.1 Detailed Description	
U. 1.4 INCHIDEL I UNCLION DOCUMENTATION	. 10

Index	19
6.3.1 Detailed Description	17
6.3 chat_server.ThreadedTCPServer Class Reference	17
6.2.2.1 handle()	17
6.2.2 Member Function Documentation	17
6.2.1 Detailed Description	16
6.2 chat_server.ThreadedTCPRequestHandler Class Reference	16
6.1.2.3 write_message()	16
6.1.2.2 init_db()	16
6.1.2.1 get_message()	15

htl-tk-chat

Watch the development progress at https://gitea.escpe.net/cc69222/htl-tk-chat

1.1 Build documentation

To build the documentation run: doxygen cd Documentation/latex make

After this you can open the refman.pdf.

1.2 Project description

A Server and client for a self-written chat application. The client interface uses Qt 6, 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 6 (via PyQt6)
- · 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.

Namespace Index

2.1 Namespace List

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

hat_client	9
hat_server	9
lient	10
lient_functions	10
latabase_functions	1
ıserstub	1

Namespace Index

Hierarchical Index

3.1 Class Hierarchy

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

socketserver.BaseRequestHandler	
chat_server.ThreadedTCPRequestHandler	16
database_functions.Database	15
socketserver.TCPServer	
chat_server.ThreadedTCPServer	17
socketserver.ThreadingMixIn	
chat server.ThreadedTCPServer	17

6 Hierarchical Index

Class Index

4.1 Class List

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

database_functions.Database	15
chat_server.ThreadedTCPRequestHandler	16
chat server.ThreadedTCPServer	

8 Class Index

Namespace Documentation

5.1 chat_client Namespace Reference

Variables

• **sock** = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

5.1.1 Detailed Description

Chat client for testing purposes

5.2 chat_server Namespace Reference

Classes

- class ThreadedTCPRequestHandler
- class ThreadedTCPServer

Variables

- HOST
- PORT
- database = database_functions.Database("chat.db")
- **server** = ThreadedTCPServer((HOST, PORT), ThreadedTCPRequestHandler)
- ip
- port
- **server_thread** = threading.Thread(target=server.serve_forever)
- daemon

5.2.1 Detailed Description

The Chat Server

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

5.3 client Namespace Reference

Functions

• def send_msg ()

Variables

- **app** = QApplication(sys.argv)
- string ui_file_name = "mainwindow.ui"
- **script_path** = os.path.realpath(<u>__file__</u>)
- **script_path_list** = script_path.split("/")
- string ui_path = "/".join(script_path_list)
- window = uic.loadUi(ui_path)

5.3.1 Detailed Description

The Chat Client

This is the chat client with Qt6 frontend.

5.3.2 Function Documentation

5.3.2.1 send_msg()

```
def client.send_msg ( )
```

Send message function

This function is called when the user presses Enter, to send the message. The function isn't fully implemented at the moment.

Definition at line 16 of file client.py.

5.4 client_functions Namespace Reference

Functions

• def authenticate (sock, str username)

5.4.1 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.

5.4.2 Function Documentation

5.4.2.1 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 13 of file client_functions.py.

5.5 database_functions Namespace Reference

Classes

class Database

5.5.1 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.

5.6 userstub Namespace Reference

Functions

- def test_connection (ip)
- def test_user_table (window)
- def test_combo_box (window)

Variables

list test_users

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 test_combo_box()

Definition at line 51 of file userstub.py.

5.6.2.2 test_connection()

```
def userstub.test_connection ( ip\ ) Test the Connection pings the ip to get the connction status. Not implemented at the moment.
```

Definition at line 14 of file userstub.py.

5.6.2.3 test_user_table()

Definition at line 23 of file userstub.py.

5.6.3 Variable Documentation

5.6.3.1 test_users

 $list\ userstub.test_users$

Definition at line 8 of file userstub.py.

Class Documentation

6.1 database_functions.Database Class Reference

Public Member Functions

- def init_db (self)
- def write_message (self, message, author, receipient, date)
- def get_message (self)

Private Member Functions

• def __dict_factory (self, cursor, row)

6.1.1 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.

6.1.2 Member Function Documentation

6.1.2.1 get_message()

```
\begin{tabular}{ll} def & database\_functions.Database.get\_message & ( & self \end{tabular} \label{fig:parabase}
```

Gets the messages from the database

Definition at line 67 of file database_functions.py.

16 Class Documentation

6.1.2.2 init_db()

6.1.2.3 write_message()

Writes the current message to the message history

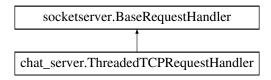
Definition at line 50 of file database_functions.py.

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

server/database functions.py

6.2 chat_server.ThreadedTCPRequestHandler Class Reference

Inheritance diagram for chat_server.ThreadedTCPRequestHandler:



Public Member Functions

• def handle (self)

6.2.1 Detailed Description

Handles the request in a thread

Definition at line 14 of file chat_server.py.

6.2.2 Member Function Documentation

6.2.2.1 handle()

```
def chat\_server.ThreadedTCPRequestHandler.handle ( \\ self )
```

function gets a request offer from the socket and prints the message and the Thread it is in. Then it sends the message back as a response.

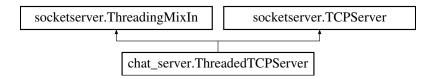
Definition at line 18 of file chat_server.py.

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

server/chat_server.py

6.3 chat_server.ThreadedTCPServer Class Reference

Inheritance diagram for chat_server.ThreadedTCPServer:



6.3.1 Detailed Description

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

Definition at line 33 of file chat_server.py.

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

server/chat_server.py

18 Class Documentation

Index

```
authenticate
    client_functions, 11
chat_client, 9
chat_server, 9
chat\_server. Threaded TCPR equest Handler, \, {\color{blue}16}
     handle, 17
chat_server.ThreadedTCPServer, 17
client, 10
     send_msg, 10
client_functions, 10
     authenticate, 11
database_functions, 11
database_functions.Database, 15
    get_message, 15
     init_db, 15
    write_message, 16
get_message
     database_functions.Database, 15
handle
    chat_server.ThreadedTCPRequestHandler, 17
init_db
     database\_functions. Database, \, 15
send_msg
    client, 10
test_combo_box
     userstub, 12
test connection
     userstub, 12
test_user_table
     userstub, 12
test_users
     userstub, 13
userstub, 11
    test_combo_box, 12
    test_connection, 12
    test_user_table, 12
    test_users, 13
write_message
     database_functions.Database, 16
```