fastchat Documentation

prachothan, vinu, tulsiram

Nov 25, 2022

CONTENTS:

CHAPTER

ONE

FASTCHAT

1.1 chat_client module

class chat_client.**Client**
Bases: object

Client of Client-Server Program connected via sockets, allowing clients to send or read messages personally or through groups

authenticator()

Allow user to either login or signup

listener(*recv_sock*)

Listens to any data from servers

Parameters **recv_sock** (*socket*) – socket that is connected to the server

refresh(*refr*)

Refreshes the chat page

Parameters **refr** (*socket*) – socket that is connected to the server

run_client()

Connects to server having minimum no of clients connected to it, authenticates the user and allow user to send message, create groups

1.2 chat_server module

class chat_server.**Server**
Bases: object

Server of Client-Server Program connected via sockets, gets inputs from clients, to allow them to send and receive messages, authenticate them and save data in database

getPort()

Returns the port that has minimum number of clients connected to it

Returns Port number

Return type int

run_server()

Connects client to a port named as superport and return port number having minimum number of clients and allow clients to send or receive messages

1.3 checkname module

checkname.**checkname**(*name*, *password*, *is_log*, *cursor*)

Searches if user already exists and returns True if found else False

Parameters

- **name** (*str*) – name of user
- **password** (*str*) – password of user
- **is_log** (*bool*) – login or signup
- **cursor** (*connection*) – object that allows us to access postgresql database

Returns True if user is found, False if not found

Return type bool

1.4 db module

1.5 encryp module

encryp.**decrypt**(*ciphertext*, *key*)

decrypts the given ciphertext(which is in string from of byte list) with the private key of user to return the actual message

Parameters

- **ciphertext** (*str*) – text which is to be decrypted
- **key** (*rsa.key.PrivateKey*) – PrivateKey of the user which is used to decrypt the aired message

Returns decrypted message

Return type str

encryp.**encrypt**(*msg*, *key*)

Encrypts the the given message with the given key string(which is PublicKey string). Encrypts string to string.

Parameters

- **msg** (*str*) – Message which is to be encrypted
- **key** – Public key string which is used to encrypt the message
- **key** – str

Returns encrypted message

Return type str

encryp.**loadPrivateKey**(*usrname*)

loads the stored Private Key(in .pem file inside pems folder) whenever the user logins. This is must for the messages to be decrypted

Parameters **usrname** (*str*) – username of user for whom we require private key

Returns Private Key of that user

Return type rsa.key.PrivateKey

encryp.**storePrivateKey**(*usrname*, *prikey*)

stores the private key of user into a local directory (./pems) in a *.pem file so that it canbe extracted for further use. This is called whenever user signups for the first time

Parameters

- **usrname** (*str*) – username of user whose private key is being stored
- **prikey** (*rsa.key.PrivateKey*) – private key of user which is to be stored

encryp.**strToPublicKey**(*str*)

Extracts Publickey from Public key string . server sends public key in the form of public key string only

Parameters **str** (*str*) – Public key string or string of the form "PublicKey(<PublicKey>)

Returns extracted Public key

Return type rsa.key.PublicKey

1.6 generator module

generator.**generateID**(*table*, *cursor*)

Generates unique ID for user or group based on user's request

Parameters

- **table** (*str*) – Generate ID for user or group
- **cursor** (*connection*) – object that allows us to access postgresql database

Returns unique ID

Return type str

1.7 get_json module

get_json.**json_splitter**(*s*)

Splits multiple jsons so that we can process each one by one

Parameters **s** (*str*) – a string containing multiple jsons

Returns list of jsons

Return type list

1.8 handler module

handler.**handler**(*s*, *cursor*, *id_dict*, *getPort*)

handles user inputs, login, signup, create group, send messagem etc and saves data in database

Parameters

- **s** (*socket*) – socket to which user is connected
- **cursor** (*connection*) – object that allows us to access postgresql database
- **id_dict** (*dictionary*) – maps sockets to user IDs
- **get_Port** (*int*) – Returns Port number having minimum number of clients

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX