fastChat Documentation

210050126_210050125_210050105

Nov 25, 2022

CONTENTS:

CHAPTER

ONE

DOCS

1.1 client module

client.**add_member**(*groupname*, *member*, *username*)
Sends add member request to the server.

Parameters

- **groupname** (*str*) – name of the group in which a new member is to be added
- **member** (*str*) – username of the new member to be added
- **username** (*str*) – the client who is sending the command for adding member(the admin of the group)

client.**create_group**(*groupname*, *username*)
Sends group creation request to the server.

Parameters

- **groupname** (*str*) – name of the group
- **username** (*str*) – the username of the client who is creating the group

client.**delete_member**(*groupname*, *member*, *username*)
Sends request to the server to remove a client from a group.

Parameters

- **groupname** (*str*) – name of the group from which the member is to be removed
- **member** (*str*) – username of the member to be removed
- **username** (*str*) – the client who is sending the command for deleting the member(the admin of the group)

client.**disconnect**()
Sends logout request to the servers.

client.**getServer**()
Requests a server from the main_server to which the client can send command to.

Returns connection object to the server

Return type connection object

client.**leave_group**(*groupname*, *username*)
Sends request to the server for leaving a group.

Parameters

- **groupname** (*str*) – the groupname of the group the client wants to leave
- **username** (*str*) – the username of the client who wants to leave the group

client.**loGin**()
Sends login request to the main_server which authorizes the client based on the given username and password.

Return type bool

client.**rec_image**(*file_type*, *server_number*)
Receives and saves the image sent from a client through a server. It saves the image in the currect working directory.

Parameters

- **file_type** (*str*) – The file type of the image which helps in correctly saving the image
- **server_number** (*int*) – the number representing which server to listen to for receiving the image

client.**rec_text**(*server_number*)
Receives the message from the servers in string format and converts it into python dictionary.

Parameters **server_number** (*int*) – the server from which message is to be received

Returns message

Return type python dictionary

client.**save_group_keys**(*msg*)
Saves the group keys that are used for encryption of messages of the group

Parameters **msg** (*python dictionary*) – contains the required keys fo storing

client.**send_group_image**(*msg*, *client*)
Sends a group image to be sent to each member of the group.

Parameters

- **msg** (*str*) – image name
- **client** (*connection object*) – the server connection to which the message is to be sent

client.**send_group_text**(*groupname*, *msg*, *username*)
Sends a groupmessage to be broadcasted in the group

Parameters

- **groupname** (*str*) – name of the group in which message is to be broadcasted
- **msg** (*str*) – the message to be broadcasted
- **username** (*str*) – the sender

client.**send_image**(*msg*, *client*)
Sends image to a different client through intermediate server

Parameters

- **msg** (*str*) – image name
- **client** (*connection object*) – connection to the server through which image is to be sent

client.**send_text**(*msg*, *client=0*)
Sends a message to the server. The message functionalities vary according to the request the client has made. It first asks the main_server to allot some server out of the many given servers.

Parameters

- **msg** (*dictionary*) – the message to be processed and sent
- **client** (*connection object*) – the server to which the message is to be sent. If it is 0, then main_server is asked to return a server to which message is to be sent

client.**signup**()
Sends signup request to the main_server which checks whether the signup credentials are valid and if yes creates the user in the server's database

Return type bool

client.**talk**()
The main loop function which makes our application a continuous one which can send messages and receive messages anytime

1.2 grpencrypt module

grpencrypt.**genkeys**(*groupname*)
It generates and stores the private and public keys for encryption decryption purposes for the group.

Parameters **groupname** (*str*) – name of the group for which keys are generated

grpencrypt.**group_decrypt**(*encryptedmsg*, *groupname*)
Converts the given encrypted message into decrypted form after transmission

Parameters

- **encryptedmsg** (*str*) – message to be encrypted
- **groupname** (*str*) – name of the group according to which encryption is to be done

Returns decrypted message

Return type str

grpencrypt.**group_encrypt**(*msg*, *groupname*)
Converts the given message into encrypted form for transmission

Parameters

- **msg** (*str*) – message to be encrypted
- **groupname** (*str*) – name of the group according to which encryption is to be done

Returns encryptedstring

Return type str

grpencrypt.**private_key_getting**(*groupname*)
Returns private key of the group as a string.

Parameters **groupname** (*str*) – name of the group for which key is returned

Returns pubkeystring

Return type str

grpencrypt.**private_key_storing**(*groupname*, *privatestring*)
Stores the private key of the group with name as groupname in the client's system.

Parameters

- **groupname** (*str*) – name of the group
- **privatestring** (*str*) – private key of the group

grpencrypt.**public_key_getting**(*groupname*)
Returns public key of the group as a string.

Parameters **groupname** (*str*) – name of the group for which key is returned

Returns pubkeystring

Return type str

grpencrypt.**public_key_storing**(*groupname*, *publicstring*)
Stores the public key of the group with name as groupname in the client's system.

Parameters

- **groupname** (*str*) – name of the group
- **publicstring** (*str*) – public key of the group

1.3 main_server module

main_server.**acceptClient**()
Accepts connection from the client

main_server.**bindSocket**()
Binds the socket object of the load balancing server to the required IP address and PORT number

main_server.**createServer**()
Creates a socket object and a database connection for the load balancing server.

main_server.**disconnect**(*username*, *conn*)
Removes the connection of a client on his request for disconnection.

Parameters

- **username** – username of the client who is disconnecting
- **conn** – connection object from the client to the server

main_server.**loginpage**(*msg*)
Authenticates a user on his login into the application

Parameters **msg** (*python dictionary*) – contains username and password of the client that he has entered

Returns a bool object depending on whether the login is correct or not

Return type bool

main_server.**random**() → x in the interval [0, 1).

main_server.**rec_query**(*conn*)
Listens for each client whatever message they have sent

Parameters **conn** (*connection object*) – connection from the server to the client

Returns message received

Return type python dictionary

main_server.**return_a_server**(*msg_type*, *conn*, *algo=1*)
Contains algorithms based on which a server is sent to the requesting client

Parameters

- **msg_type** (*str*) – what is the message type that needs to be sent
- **conn** (*connection object*) – connection object from the load balancing server to the client
- **algo** (*int*) – represents which algorithm is to be used for server selection

main_server.**signuppage**(*msg*)
Updates the database when a new user signsup in the application

Parameters **msg** (*python dictionary*) – contains the username and password of the new client

Returns bool object depending on whether the username was unique or not

Return type bool

main_server.**store_key**(*msg*)
Stores public key of a client in the database.

Parameters **msg** (*pytohn dictionary*) – contains key

main_server.**talk**(*username*, *conn*)
The continuous loop that enables the server to listen from a client and take action accordingly.

Parameters

- **username** (*str*) – username of the client
- **conn** (*connection object*) – connection object from the client to the server

1.4 newencrypt module

newencrypt.**decrypt**(*encryptedtext*, *username*)
Creates decrypted message from the encrypted message.

Parameters

- **encryptedtext** (*str*) – message to be decrypted
- **username** (*str*) – receiver's username

Returns decrypted string

Return type str

newencrypt.**encrypt**(*msg*, *otherUserPublicKey*)
Creates encrypted string for the message and returns it.

Parameters

- **msg** (*str*) – message to be encrypted
- **otherUserPublicKey** (*str*) – recevier's public key

Returns encrypted string

Return type str

newencrypt.**generatekeys**(*username*)
Generate keys for encryption and returns public key for storing in database.

Parameters **username** (*str*) – username for which keys are generated

Returns publickeystring

Return type str

1.5 server module

server.**acceptClient**()
Accepts connection from the client

server.**add_member**(*msg*)
Updates the database when a new user is added in a group

Parameters **msg** (*python dictionary*) – contains groupname, username of the added member

server.**bindSocket**()
Binds the socket object of the server to the required IP address and PORT number

server.**createServer**()
Creates a socket object and a database connection for the server.

server.**create_group**(*msg*)
Updates database when a new group is created

Parameters **msg** (*python dictionary*) – the message containing groupname, admin

server.**delete_member**(*msg*)
Updates the database when a member is deleted from the group

Parameters **msg** (*python dictionary*) – contains groupname, username of the deleted member

server.**disconnect**(*username*, *conn*)
Removes the connection of a client on his request for disconnection.

Parameters

- **username** – username of the client who is disconnecting
- **conn** – connection object from the client to the server

server.**leave_group**(*msg*)
Updates the database when a member leaves a group

Parameters **msg** (*python dictionary*) – contains groupname, username of the member left

server.**loginpage**(*msg*)
Authenticates a user on his login into the application

Parameters **msg** (*python dictionary*) – contains username and password of the client that he has entered

Returns a bool object depending on whether the login is correct or not

Return type bool

server.**message_retrieval**(*msg*, *conn*)
Retrieves the messages that were sent to a client when he was offline and sends them on his login.

Parameters

- **msg** (*python dictionary*) – contains username of the client
- **conn** (*connection object*) – connection object from the client to the server

server.**rec_query**(*conn*)
Listens for each client whatever message they have sent

Parameters **conn** (*connection object*) – connection from the server to the client

Returns message received

Return type python dictionary

server.**send_group_image**(*msg*, *conn_send*)
Sends image to all the members of the group except the one who has sent it

Parameters

- **msg** (*python dictionary*) – contains information about the groupname, sender, image type etc
- **conn_send** (*connection object*) – connection object from the sender client to the server

server.**send_group_text**(*msg*)
Sends a broadcasting group message to all the members of a group except the user who sent that message

Parameters **msg** (*python dictionary*) – the message sent by the user

server.**send_image**(*rec*, *msg*, *conn_send*)
Receives and sends image from sender client to receiver client

Parameters

- **rec** (*str*) – username of the receiver client
- **msg** (*str*) – the initial message to be sent before image is sent
- **conn_send** (*connection object*) – connection object from the sender client to the server

server.**send_key**(*msg*, *conn*)
Sends the public key of a client to another client who wants it for communication

Parameters

- **msg** (*python dictionary*) – the message sent by the client who wants the key
- **conn** (*connection object*) – connection object from the requesting client to the server

server.**send_text**(*rec*, *msg*)
Sends message to the client

Parameters

- **rec** (*str*) – username of the client to which message is to be sent
- **msg** – message
- **msg** – python dictionary

server.**signuppage**(*msg*)
Updates the database when a new user signsup in the application

Parameters **msg** (*python dictionary*) – contains the username and password of the new client

Returns bool object depending on whether the username was unique or not

Return type bool

server.**store_key**(*msg*)
Stores public key of a client in the database.

Parameters **msg** (*pytohn dictionary*) – contains key

server.**talk**(*username*, *conn*)
The continuous loop that enables the server to listen from a client and take action accordingly.

Parameters

- **username** (*str*) – username of the client
- **conn** (*connection object*) – connection object from the client to the server

1.6 sql module

class sql.**Database**
Bases: object

add_user_to_group(*username*, *groupname*, *isAdmin=0*)
Adds the user with name as username to the group whose name is groupname.

Parameters

- **username** (*str*) – username of the user
- **groupname** (*str*) – groupname of the user

authenticate(*username*, *password*)
Authenticates the user with name given as username and password given as password. Used in login purposes.

Parameters

- **username** (*str*) – username of the user whose authentication is to be done
- **password** (*str*) – password given by the user

Returns 1 on incorrect credentials and 2 on successful authentication

Return type int

change_status(*username*, *isOnline*)
Updates the isOnline value of a user in the database which tells whether a user is Online or Offline

Parameters

- **username** (*str*) – username of the user
- **isOnline** (*int*) – 1 if online and 0 if offline

check_groupname(*groupname*)
Checks whether the there exists a group with its name given exists in the databse or not.

Parameters **groupname** (*str*) – groupname whose existence in the databse is to be checked

Returns bool object representing whether it exists or not

Return type bool

check_user_in_group(*username*, *groupname*)
Check whether a user with name given as username is present in the group whose name is groupname.

Parameters

- **username** (*str*) – username whose existence in the group database is to be checked
- **groupname** (*str*) – groupname of the group in which existence is to be checked

Returns bool object representing whether it exists or not

Return type bool

check_username(*username*)
Checks whether the username given exists in the database or not.

Parameters **username** (*str*) – username whose existence in the databse is to be checked

Returns bool object representing whether it exists or not

Return type bool

close_connection()
Closes the connection with the database

create_group_database(*groupname*, *adminuser*, *publickey=''*)
Updates the database and adds a new table for the group with name as groupname. Also adds the adminuser to the group's table and adds the publicKey of the group

Parameters

- **groupname** (*str*) – groupname of the group
- **adminuser** (*str*) – username of the admin
- **publickey** (*str*) – publickey of the group

create_user(*username*, *password*, *publicKey=''*)
Checks whether the given username is present in the Users table or not. If it is not present then it adds that user to the database, otherwise it doesn;t update the database.

Parameters

- **username** (*str*) – Username of the client to be added
- **password** (*str*) – encrypted for of the password of the user
- **publicKey** (*str*) – publicKey of the user for E2E encryption

delete_user_in_group(*username*, *groupname*)
Removes the user with name as username from the group whose name is groupname.

Parameters

- **username** (*str*) – username of the user
- **groupname** (*str*) – groupname of the user

get_all_users(*groupname*)
Returns a list of users in the group with name as groupname.

Parameters **groupname** (*str*) – name of the group

Returns list of usernames present in the group

Return type list

get_key(*username*)
Returns publicKey of the user with name as username.

Parameters **username** (*str*) – username of the user

Returns publicKey of the user

Return type str

get_status(*username*)
Returns the curent status of the user stating whether the user is online or offline

Parameters **username** (*str*) – username of the user

Returns isOnline (1 if online and 0 if offline)

Return type int

isAdmin(*username*, *groupname*)
Checks whether the user with name as username is the admin of the group with name as groupname.

Parameters

- **username** (*str*) – username of the user
- **groupname** (*str*) – name of the group

Returns 0 if it is not an Admin else 1

Return type int

make_groups_table()
It creates the table Groups in the database if it wasn't there. If it was already present, it drops that and creates a new Groups table in the database.

make_users_table()
It creates the table Users in the database if it wasn't there. If it was already present, it drops that and creates a new Users table in the database.

message_ret(*username*)
Gives a list of messages stored in a user's database for transmission of messages by the server to clients who missed these messages and switched back to online from offline

Parameters **username** (*str*) – username of the user

Returns list of messages

Return type list

message_sent_to_group(*username*, *groupname*, *message*)
Stores the messages sent to the group.

Parameters

- **username** (*str*) – username of the user
- **groupname** (*str*) – name of the group
- **message** (*str*) – message to be stored

message_sent_to_user(*username2*, *message*)
Saves the message of the sent by a client to user with username as username2 and hence helps in message retrieval when the user switches back to online from offline

Parameters

- **username2** (*str*) – username of the user

- **message** (*str*) – message to be stored

update_key(*username*, *key*)

Adds publicKey of the user with name as username. :param username: username of the user :type username: str

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

R

S

T

U