SecureFastChat Server Program

Khushang Singla, Mridul Agarwal, Arhaan Ahmad

Nov 16, 2022

CONTENTS:

CHAPTER ONE

SERVER-SIDE

1.1 DatabaseRequestHandler module

`DatabaseRequestHandler.`**`checkuid`**(*uid*)

`DatabaseRequestHandler.`**`setpass`**(*uid*, *pwd*)

`DatabaseRequestHandler.`**`check_login_uid`**(*uid*)

`DatabaseRequestHandler.`**`check_login_pwd`**(*uid*, *pwd*)

1.2 Message module

`class` `Message.`**`Message`**(*conn_socket*, *status*, *request*, *sel*)

Bases: `object`

This is the class to handle Encryption of messages. The format in which the message is sent to client is determined in this class

Parameters

- **`task`** (*`str`*) – Task to be done. It can have the values signup, login, send_message
- **`socket`** (*`socket.socket`*) – The socket used for connection with Server
- **`request_content`** (*`dict`*) – Content to include in the request to send to server
- **`_data_to_send`** (*`bytes`*) – Contains the data to send to the server
- **`_recvd_msg`** (*`bytes`*) – Content received from server is stored here

Constructor Object

Parameters

- **`conn_socket`** (*`socket.socket`*) – Socket which has a connection with client
- **`task`** (*`str`*) – Task to do. It can have values: login, signup, send_message
- **`request`** (*`str`*) – Content to send to server

`_send_data_to_client`()

Function to send the string to the client. It sends content of _send_data_to_client to the client

`_recv_data_from_client`(*size*)

Function to recv data from client. Stores the bytes recieved in a variable named _recvd_msg.

Parameters **`size`** (*`int`*) – Length of content to recieve from server

`_send_msg_to_reciever`(*rcvr_sock*)
Function to send message to a reciever

`_json_encode`(*obj*, *encoding*)
Function to encode dictionary to bytes object

Parameters

- **obj** (*`dict`*) – dictionary to encode
- **encoding** (*`str`*) – Encoding to use

Returns Encoded obj

Return type bytes

`_json_decode`(*obj*, *encoding*)
Function to decode bytes object to dictionary

Parameters

- **obj** (*`bytes`*) – Encoded json data
- **encoding** (*`str`*) – Encoding used

Returns Decoded json object

Return type json

`processTask`(*loggedClients*)
Processes the task to do

Returns Returns int to represent result of the process. The details of return values are given in the corresponding functions handling the actions.

Return type int

`_send_msg`(*rcvr_uid*, *msg_type*, *content*, *loggedClients*)

`_send_rcvr_key`(*rcvr_uid*)

`keyex`() → str
Does key exchange. First waits for request from the client, then sends a response with its own public key. Returns a string containing the public key of the client

Returns public key of the client

Return type str

`_process_login`(*uid*)

`_login_failed`()

`_login_successful`()

`_login_uid_not_found`()

`_successfully_found_login_uid`(*token*)

`_signup_failed`()

`_successfully_signed_up`()

`_process_signup_uid`(*uid*)

`_signup_uid_not_available`()

`_signup_uid_available`()

_process_signup_pass(*encrypted_pass: str*)
Process the command for signing up the user and storing the password

Parameters

- **encrypted_pass** – The encoded password
- **client_public_key** – Public key of the client

isOnline()

get_uid_sock()

1.3 db module

db.**checkIfUsernameFree**(*username: str*) → bool
Check if a given username is already in use

Parameters **username** (*str*) – The username to check

Returns Whether the name is in use or not

Return type bool

db.**createUser**(*username: str*, *password: str*) → bool
Adds a user with the given username and password to the database. Assumes that the checkIfUsernameFree has already been called before. We hash the password here. Returns true if the user generation happened without any error

Parameters

- **username** (*str*) – username
- **password** (*str*) – password (hashed)

Returns Whether the user creation happened succesfully

Return type bool

db.**login**(*username: str*, *password: str*) → bool
Checks if a given username password pair is present in the db

Parameters

- **username** (*str*) – username
- **password** – password

Returns True if the user is authenticated by this

Return type bool

db.**storeMessageInDb**(*sender: str*, *receiver: str*, *message: str*)
stores the encrypted message in the database, in case it is not possible to send them the message directly

Parameters

- **sender** (*str*) – sender username (TODO: Do we keep these encrypted?)
- **receiver** (*str*) – receiver username
- **message** (*str*) – the enecrypted message

1.4 startServer module

`startServer.`**`accept`**(*sel*, *sock=None*)

Function to accept a new client connection

`startServer.`**`service`**(*key*, *mask*)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX