# SecureFastChat Server Program

Khushang Singla, Mridul Agarwal, Arhaan Ahmad

# **CONTENTS:**

1	er-Side	1	
	1.1	DatabaseRequestHandler module	1
	1.2	Message module	
	1.3		
	1.4	startServer module	4
2 Indices and tables			
Ру	thon	Module Index	7
In	dex		9

**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 (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)
\mathbf{keyex}() \rightarrow \mathbf{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)  $\rightarrow$  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

 $db.createUser(username: str, password: str) \rightarrow bool$ 

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 successfully

Return type bool

db.login (username: str, password: str)  $\rightarrow$  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

1.3. db module 3

# 1.4 startServer module

 $\begin{tabular}{ll} {\tt startServer.accept} (sel, sock=None) \\ {\tt Function} \ to \ accept \ a \ new \ client \ connection \\ \end{tabular}$ 

startServer.service(key, mask)

## **CHAPTER**

# TWO

# **INDICES AND TABLES**

- genindex
- modindex
- search

# **PYTHON MODULE INDEX**

# d DatabaseRequestHandler, 1 db, 3 m Message, 1 S startServer, 4

8 Python Module Index

# **INDEX**

Symbols	D		
_json_decode() (Message.Message method), 2 _json_encode() (Message.Message method), 2 _login_failed() (Message.Message method), 2 _login_successful() (Message.Message method), 2	DatabaseRequestHandler module, 1 db module, 3		
_login_uid_not_found() (Message.Message method), 2	G get_uid_sock() (Message.Message method), 3		
_process_login() (Message.Message method), 2 _process_signup_pass() (Message.Message	is Online () (Massage Massage method) 2		
_process_signup_uid() (Message.Message method), 2	isOnline() (Message.Message method), 3		
_recv_data_from_client() (Message.Message method), 1	keyex() (Message.Message method), 2		
_send_data_to_client() (Message.Message method), 1 _send_msg() (Message.Message method), 2	L login() (in module db), 3		
_send_msg_to_reciever() (Message.Message method), 1	M		
_send_rcvr_key() (Message.Message method), 2 _signup_failed() (Message.Message method), 2 _signup_uid_available() (Message.Message method), 2	Message module, 1 Message (class in Message), 1 module		
_signup_uid_not_available() (Mes- sage.Message method), 2 _successfully_found_login_uid() (Mes- sage.Message method), 2	DatabaseRequestHandler, 1 db, 3 Message, 1 startServer, 4		
_successfully_signed_up() (Message.Message method), 2	P		
A	processTask() (Message.Message method), 2		
accept() (in module startServer), 4	service() (in module startServer), 4		
C	setpass() (in module DatabaseRequestHandler), 1		
check_login_pwd() (in module  DatabaseRequestHandler), 1	startServer module, 4		
check_login_uid() (in module  DatabaseRequestHandler), 1	storeMessageInDb() (in module db), 3		
<pre>checkIfUsernameFree() (in module db), 3 checkuid() (in module DatabaseRequestHandler), 1 createUser() (in module db), 3</pre>			