- # MEPHA Chat-Application

Introduction:

This is a simple chat-application which supports a server and many clients. I’ve used “C” programming language to develop this application and I will explain more about details and implantations.

Main Tools:

- *cJSON Library which is available at https://github.com/DaveGamble/cJSON.*
- *Winsock2 library for socket programming.*

Client :

- **Receiving string from the server:**
  After sending a message, a response will be received in string format which must be parsed into json object. This response includes the situation of the process, whether it was successful or not .
- **Defined Functions:**
  int connect_to_server():
  This function create a socket in client side and connects to server.This function is called in all other functions before everything else happens.

  void account_menu():
  Managing user account options interface. These options are available here:
    - *Register*
    - *Login*
    - *Exit*

void regist():

This function gets username and password from the user and send them to server in this format:(Using built-in **send()** function)

"register <username>, <password>"

void login():

This function gets username and password from the user and send them to the server in this format, just like the previous one:

"login <username>, <password>"

If everything goes fine, the server send a unique auto-token to the client to identify the client in next requests. The token will be saved in an array(*char auto_token [100]).*

- *void main_menu():*

  When the user logs in successfully, he/she can manage the tasks from this panel. The user can access these options here:

  - Create channel
  - Join a channel
  - Logout
- *void create_channel():*

  After connecting to the server, this function gets a channel name from the user and send it in this format:

  "create channel <channel name>, <AutoToken>"

- *void join_channel():*

  The user can join to an existing channel here. The function gets an existing channel name and send it and the token to the server in this format:

  ”join channel <channel name>, <AuthToken>”

- *void logout():*
  Calling this function result in users logging out. The request format is:

  "logout <AutoToken>"

- *void chat_menu():*

  After the user joins a channel successfully, he/she can manage the tasks from this panel. The user can access these options here:

  - Send message
  - Refresh
  - Members list
  - Leave channel
  - Back to main menu

  After choosing any option here, the corresponding function will be called.

- *void send_message() :*
  This function gets a message from the user, make a string in the following format and send it to the server:

  "send <message>, <AuthToken>"

- *void refresh():*
  After sending a request to the server in proper format , a string will be received that includes member messages.Then it must be parsed into array element and the message and the messages will be printed. The format is :

```
{
“type”: “List”,
“content”: [
{“sender”: “<username>”, “content”: “<message_content>”},
{“sender”: “<username>”, “content”: “<message_content>”},
{“sender”: “<username>”, “content”: “<message_content>”},

...
]
        }
```

- *void members_list():*

After sending a request to the server in proper format, a string will be received that includes member messages.Then it must be parsed into array element and the message and the messages will be printed. The format is :

```
{“type”: “List”, “content”: [“<username1>”, “<username2>”, “<username3>”, ...]}
```

- *void leave_channel():*
  For leaving the channel which the user was joined , a request is sent in this format:
  "leave <AuthToken>"

- **Main function() :**
  In the main() function, *account_menu()* is called.

Server:

- **Structures**:
  struct users:

  Which consists of two variables: username and auto token. All usernames and their unique auto token save in an array (online_users).

  struct channel_members:

  This one saves the auto token and the channel name which the user has joined. All auto tokens and their channels save in an array(channel_members).

- **Helping functions:**

  *Verify_user (char*auto_token):*

  Returns the index of an online user in the online_users array if exists, while -1 will be returned if not found.

  int check_member(char auto_token[]):

  Returns the index of a joined user in the channel members array if exists, while -1 will be returned if not found.

Other functions:

- *void regist(int client_socket,char *username,char *password):*

  This function gets the username and password, create a new text file in database and save the password in that file.

  Response sends to the client in proper format.

- *void login(int client_socket,char*username,char*password):*

  Checks if the username exists in the database. Also, checks the password correction. Then call fhandle function()(written in fhandle c file) and gets a generated auto token.

  Response sends to the client in proper format.

- *void create_channel(int client_socket,char *channel_name,char *auto_token):*

  After calling *Verify_user* for verification, it creates a folder in the database with the same name as the channel name. Then create two text files in it:

  One for members and the other for messages.

  *void join_channel(int client_socket,char *channel_name,char *auto_token):*

  Add the auto token to a the messages text file which exists in the channel name's folder.

- *void receive_message(int client_socket,char*message,char*auto_token):*

  After calling *Verify_user* for verification, it add the message and the username to an array in cJSON object. Then print it to an string in special format and send it to the client.

- *void refresh(int client_socket,char*auto_token) :*

  This function reads the messages written in the messages file in channel name folder. Then send them so the client.

- *void show_members(int client_socket,char*auto_token):*

  This function read the auto tokens from the members text file in channel name folder line by line ,and find the corresponding username saved in channel_members array. Then sends the usernames to the client in proper format ,created using cJSON functions.

  *void leave_channel(int client_socket,char *auto_token):*

  This function deletes the auto token from the messages text file and send the proper response to the client.

  *void logout(int client_socket,char*auto_token):*

  After finding the index of the username in online_users array, it replaces the username with NULL string.

Main() function :

First, the socket is created and listens until a client connects to it. After that, it receives commands from the client and call the corresponding functuin in an infinite while loop. And finally, we close the socket.

Fhandle c file:

This file includes a single function which generate a 32-bit random auto token, consists of capital letters, small letters, numbers and special characters.