

CN Messenger Documents

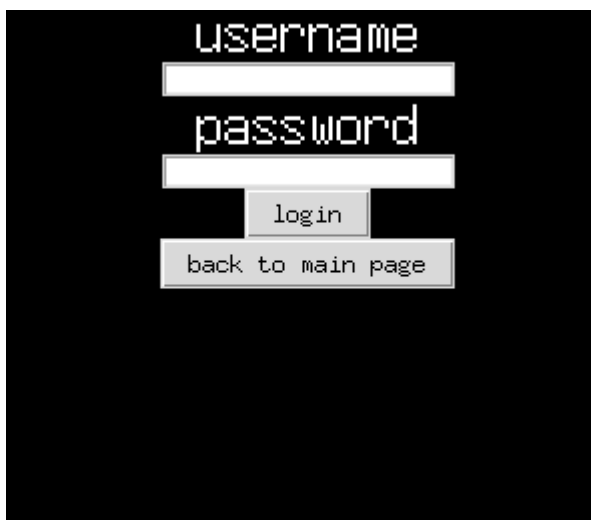
1. User & Operator Guide

GUI:



Initial landing page, can select different mode to decide **LOGIN** or **REGISTER**

LOGIN



Once we have typed in username, and password, we can then press the login button to login.

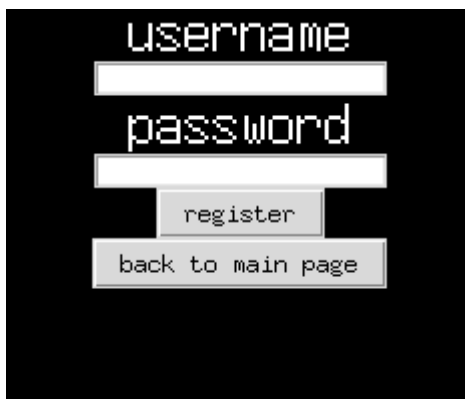
If successfully logged in, it will show:



If it fails, it will show:



REGISTER

A registration form on a black background. It features two white input fields labeled "username" and "password". Below the fields are two buttons: "register" and "back to main page".

If successfully creates, it shows:



If it fails, it will show:



USER page



Protocol:

“->” means message from Client to Server. In contrast, “<-”.

- Register: -> REGISTER \$name
 - If User Not registered: <- YES
 - Input password: -> \$password
 - If Register Success: <- GOODJOB
 - else: <- FAIL
 - else: <- FAIL
- Login: -> LOGIN \$username
 - If User Not Exist: <- NO
 - elif User has Logged in: <- NO, user: \$Steve is already online
 - elif User Exist: <- HI
 - Input password: -> \$password
 - If password match: <- WELCOME
 - else: <- GO AWAY
- Get Message:-> GETMSG \$name
 - If Receiver not Exist: <- NOTEXIST
 - elif: <- \$ChatFile.csv
- Send Message:-> SENDMSG \$receiver \$message

- <- Done
- Get File: ->``
- Send File: ->

2. Instructions on how to run server & clients

- Run server: `python main.py`
- Run client: `python client.py`

3. System & Program Design

- Encryption:
 - We use openssl to generate cert and key and use `ssl` in python standard library to make the connection secure.
 - We hash password with `hashlib.md5` .
- Message & Information Storage
 - We use csv file and package `panadas` to retrieve, store chat & account's information.
 - There are 2 csv files storing

Accounts.csv

| Username | hashed_pwd | IP | active |
|----------|----------------------------------|-----------------|--------|
| Bob | 1f3870be274f6c49b3e31a0c6728957f | 140.112.106.88 | 1 |
| Apple | 1f3870be274f6c49b3e31a0c6728957f | 140.112.107.210 | 0 |

The chat room file for *Bob* and *apple* will be stored like,
Bob_apple.csv

| No | Sender | Msg | A_read | B_read |
|----|--------|----------------|--------|--------|
| 1 | apple | "Hello" world | 1 | 1 |
| 2 | Bob | "Hello, too" | 1 | 0 |
| 3 | Bob | "This is hard" | 1 | 0 |
| 4 | Bob | "I am tired" | 1 | 0 |

Server side implementation:

- Client Register

- param: (name,passwd)

1. Server checks for the following:

- Existence → reject if already exist.
- isOnline → Reject if already online.
- Special characters → Reject if contains illegal chars.

2. If the above checking passes, create an entry within the *Accounts.csv*.

- **Client Log in**

- param: (name,passwd)

1. Server checks for the following:

- isOnline → Reject if true.
- NotExist → Reject if true.

2. If the checking passes, Modify the *active* column in *Accounts.csv* for entry name.

The following actions assumes the client has logged in already.

- **Client Gets chat**

- param: (receiver)

1. Server checks (receiver) for the following property:

- Existence → reject if not exist.

2. If the above checking passes, a file called "sender_receiver.csv" will be created and stored in "storing" directory (replace sender, receiver with the actual name).

- This will not happen, if the file exists already.

3. Returns the file in json format as follows:

```
[
  {
    "NO" : int()
    "msg" : "hello",
    "Sender": "Bob",
    "A_read" : 1,
    "B_read" : 0
  },
  {...}
  ...
]
```

- **Client Update chat**

- param: (receiver,msg)

1. Server checks (receiver) for the following property:

- Existence → reject if not exist.

2. If the above checking passes, “sender_receiver.csv” will be opened. Append the record into the file.

[msg, sender, A_read, B_read]

3. Set the sending side read state, either *A_read* or *B_read* to 1.

- **Client Gets file**

- param: (name)

- 1. Server checks (receiver) for the following property:

- Existence → reject if not exist.

- 2. If the above checking passes, server will look for file names with name being enclosed by underscores. Send the file(s) in byte stream if there exists one.

- If name is BOB, then files like “S_BOB_something.blah” will all be returned.

- **Client Sends file**

- param: (receiver,filename)

- 1. Server checks (receiver) for the following property:

- Existence → reject if not exist.

- isOnline → Reject if receiver is not online.

- 2. If the above checking passes, server will store the file(s) with “S_name” appended to the front of the filename.

- For instance, if “Dan” wants to send file to “Bob” a file called “Rain.mp3”, then there will be a file “S_Bob_Rain.mp3” stored in the server.

4. Other things you want to say, if any
