**PROJECT 2**

**NETWORK PROGRAMMING COURSE**



**FreeChat sebuah alat komunikasi antar client menggunakan protocol FTP**

**Group 2**

Narendra Haryo B. NRP. 5115100009

Andrea P. Janardana NRP. 5115100047

Maulana sechan NRP. 5115100059

Aditya Dwicahyo NRP. 5115100087

Faturochman Pranacahya A. NRP. 5115100103

M. Faris abdurasyid NRP. 5115100147

**Lecturer :**

Royyana M Ijtihadie, S.Kom.,M.Kom.,Ph.D

**DEPARTMENT OF INFORMATICS, FACULTY OF INFORMATION TECHNOLOGY**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA 2017**

Spesifikasi Sistem :

* Kegunaan : - Bisa Komunikasi dua arah secara langsung.

- Bisa komunikasi group discussion

* Port : 123452
* Protocol : TCP
* Dasar Operasi :

1. User Management : terdapat autentikasi untuk user agar bisa mengakses FreeChat
2. Sistem Management :
   1. Sistem memberi pemberitahuan bahwa user sudah login (jika ingin chat)
   2. Sistem dapat mengetahui ID client.
   3. Group Discussion Maksimal 3 orang.

* Batasan Penggunaan :

1. Sistem :

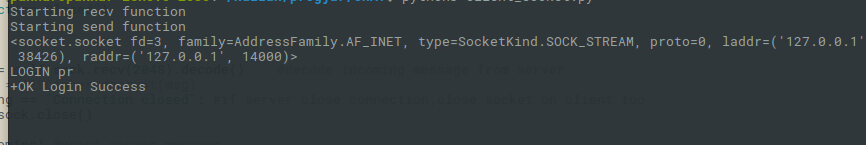
* 1. Tidak dapat mengirim lebih dari 255 kata.
  2. Tidak bisa mengirim file
  3. Username tidak boleh lebih dari 12 karakter.

2. User :

* 1. Sistem memberi pemberitahuan bahwa user sudah login (jika ingin chat)
  2. Sistem dapat mengetahui ID client.
  3. Group Discussion Maksimal 3 orang.
* Protocol Freechat

1. User :

* 1. Mengetik LOGIN ‘USERNAME’



* 1. Mengetik SHOW untuk melihat daftar client yang mengakses server tersebut.

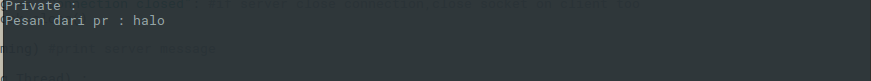


* 1. Mengetik PRIV ‘USERNAME’ untuk akses private chat.

“mengirim pesan secara private”



“menerima pesan dari private chat”



* 1. Mengetik PUB ‘KALIMAT’ untuk membroadcast pesan ke semua user.



* 1. Mengetik LOGOUT untuk keluar program.
  2. Mengetik ‘KALIMAT’ dan diakhiri enter untuk mengirim pesan.
  3. (bisa membuat multichat sendiri)
  4. (yang baru join, bisa masuk ke multichat orang lain yang maks 3 client)

2. Sistem :

1. Mengembalikan kata “ERR” dan “OK” untuk input salah atau benar.





1. Menyimpan USERNAME client dari jika menerima inputan LOGIN.

C:\Users\Pranacahya\Downloads\Server report.png

1. Menampilkan semua USERNAME jika mendapat input SHOW.



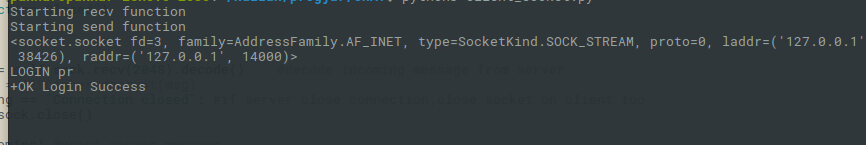
1. Menghapus data client jika menerima inputan LOGOUT.



* Protocol Freechat ‘User’

1. Protocol Login :

* Untuk masuk ke dalam server agar bisa chatting user bisa mengetik :
* LOGIN “USERNAME” ex: LOGIN Prana
* Server akan mencatat Prana sebagai client di server tersebut dan berhasil login dengan menampilkan layer kosong.
* Maksimal karakter dalam username adalah 12.
* Jika melebihi dari 12 maka sistem akan menolak client dan mengeluarkan kalimat “Username Melebihi 12 Karakter!”, setelah itu client bisa memasukan ulang usernamenya.
* Jika terdapat username yang sama akan mengeluarkan kalimat “Username sudah tersedia di server!”, setelah itu client bisa memasukan ulang usernamenya.



1. Protocol Melihat User yang Online :

* Untuk melihat semua client yang terdaftar dalam server, user bisa mengetikan “SHOW\_USER” pada terminal.
* Maka bisa melihat username yang tersedia dengan format :

1 Prana

2 Adit

3 Rendro

* Untuk mengakses command SHOW user harus bias LOGIN terlebih dahulu

1. Protocol Pesan Private :

* Untuk Mengechat secara private, user bisa menulis dengan format “PRIV” “USERNAME\_PENERIMA” “PESAN”
* Contoh : PRIV Adit Halo apa kabar?
* Saat masuk ke dalam server pesan akan dienkripsi terlebih dahulu dan disimpan di server.
* Server akan mengecek jenis pesan yang terkirim adalah PRIV (Private Chat). Server akan otomatis membutuhkan data lagi yaitu USERNAME\_PENERIMA.
* Server mengecek USERNAME untuk mengetahui kemana pesan akan dikirim.
* Sebelum meneruskan pesan ke penerima, server mendekrip terlebih dahulu pesan yang sebelumnya dienkripsi.
* Pesan sampai ke penerima dengan format
* Private:
* Pesan dari “USERNAME\_PENGIRIM” : “PESAN”

1. Protocol Pesan Public :

* Untuk Mengechat secara public, user bisa menulis dengan format “PUB” “PESAN”
* Contoh : PUB Halo apa kabar?
* Saat masuk ke dalam server pesan akan dienkripsi terlebih dahulu dan disimpan di server.
* Server akan mengecek jenis pesan yang terkirim adalah PUB (Public Chat).
* Sebelum meneruskan pesan ke penerima, server mendekrip terlebih dahulu pesan yang sebelumnya dienkripsi.
* Pesan sampai ke penerima dengan format

Public:

Pesan dari “USERNAME\_PENGIRIM” : “PESAN”

1. Terdapat system multichat dengan maksimal 3 client.

* Untuk membuat multichat, user harus berhasil login. Lalu menuliskan “CREATE\_MULTICHAT” “Nama multichat”,

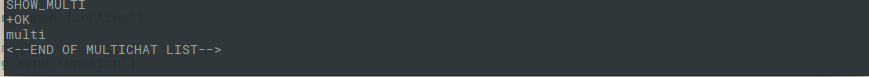
Ex : CREATE\_MULTICHAT ProgjarB



* Sistem akan menyimpan multichat di dalam server.
* Untuk melihat daftar multichat bisa menuliskan “SHOW\_MULTI”, akan mengeluarkan daftar multichat yang tersedia dan status apakah sudah maksimum atau belum dengan format
* “Nama Multichat” “Status”

Ex :

Jika berhasil membuat multichat akan mengeluarkan feedback “Multichat berhasil dibuat”



1. Protocol Join Multichat :

* Untuk masuk ke dalam multichat ketik “JOIN\_MULTI” “Nama Multichat”
* Kemudian server akan mengecek :
  + - Jika sudah penuh dan client meminta masuk maka sistem memblock user dan memberi feedback kalimat “Maaf multichat sudah penuh”
    - Jika masih kosong dan client meminta masuk maka system akan memasukan client ke dalam multichat tersebut dan meng-increment jumlah client pada multichat tersebut





1. Protocol Chat Multichat :

* Untuk mengechat di dalam multichat, user perlu mengetikan “MULTI” “KALIMAT”.

Ex :





* Makan client yang didalam multichat tersebut akan menerima pesan



* Batasannya, client hanya bisa join satu multichat sajah. Jika ingin masuk ke multichat lain atau membuat multichat lain harus keluar dalam multichat tersebut dengan menulis syntax “OUT\_MULTI”.

Fungsi-Fungsi dari Source Code yang ada :

Pembagian Tugas tiap Kelompok :

Perbedaan Synchronous dan Asynchronous :

Synchronous:

object => request 1 => response 1 => proses => request 2 => response 2 => proses

Synchronous mengharuskan object menunggu masing2 request memberikan respon

sebelum melanjutkan ke proses dan kemudian request selanjutnya.

Sedangkan Asynchronous:

object => request 1 => response 1 => proses

|=> request 2 => response 2 => proses

|=> request 3 => response 3 => proses

Asynchronous memungkinkan untuk mengirimkan request dan melakukan hal lainnya

sementara response belum dikirim. Setelah response dikirim, sebuah event akan

ter-invoke dan object dapat melakukan proses terhadap response itu.

Contoh paling mudah adalah mengirimkan pesan dengan form. Dengan metode

synchronous (metode POST seperti biasa), browser harus mengirimkan isi form ke

server dan harus menunggu respon dari server baru kemudian menampilkan hasilnya.

Sementara menunggu, halaman tidak akan dapat diutak atik.

Dengan menggunakan XHR/AJAX yang asynchronous, sementara form dikirim, kita

tetap dapat melakukan ini dan itu. Ketika browser menerima kembali respon dari

server, sebuah event akan ter-invoke, respon akan diproses, dan hasilnya akan

ditampilkan, misalnya dengan menampilkan pesan error/sukses.

Fitur – Fitur yang ada