Tugas UDP Socket Programming
II2120 - Jaringan Komputer

Project Mata Kuliah II2120 Jaringan Komputer
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Diampu oleh:
Hamonangan Situmorang, S.T, M.T.

Oleh : Kelompok Zepuh Zarkom

Sharon Darma Purta	18223106
Nakeisha Valya Shakila	18223133

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JATINANGOR
2024

KATA PENGANTAR

Puji syukur kami ucapkan kehadirat Allah SWT atas segala rahmat-Nya sehingga kami dapat menyelesaikan tugas UDP Socket Programming.

Laporan ini disusun untuk memenuhi tugas mata kuliah Jaringan Komputer (II2120) yang diampu oleh Bapak Hamonangan Situmorang, S.T, M.T. sebagai dosen di Kelas 03, Sekolah Teknik Elektro dan Informatika (Komputasi), Institut Teknologi Bandung. Kami berharap tugas ini dapat menambah pengetahuan dan pengalaman bagi pembaca dan bagi kami tentunya.

Kami ucapkan terima kasih kepada Bapak selaku dosen mata kuliah Jaringan Komputer (II2120) atas tugas yang telah diberikan dimana tugas besar ini dapat mengasah kemampuan Computational Thinking dalam memecahkan masalah terkait. Kami ucapkan juga terima kasih kepada semua pihak yang telah membantu proses penyusunan tugas besar ini baik berupa tenaga maupun pikiran. Tentunya, laporan ini tidak akan bisa sempurna tanpa dukungan dan bantuan berbagai pihak yang terlibat.

Kami menyadari bahwa kami masih banyak kekurangan dalam pembuatan tugas besar ini karena keterbatasan pengetahuan dan pengalaman kami. Kami mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan tugas besar ini

Jatinangor, 1 September 2024

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Dalam era digital yang terus berkembang, pemahaman tentang protokol komunikasi jaringan, khususnya User Datagram Protocol (UDP), menjadi semakin penting untuk mendukung aplikasi modern seperti streaming video, game online, dan komunikasi real-time yang memerlukan pengiriman data cepat dan efisien. UDP menawarkan kecepatan tinggi dengan mengorbankan beberapa aspek keandalannya, sehingga menjadi pilihan yang menarik untuk berbagai aplikasi. Kajian ini mengeksplorasi penerapan socket programming melalui pembuatan program sederhana yang memanfaatkan UDP sebagai fungsi utamanya, sekaligus menyelidiki mekanisme pengiriman data sederhana lewat jaringan menggunakan protokol pada transport layer. Dengan membahas berbagai aspek, termasuk pengaturan socket, pengiriman dan penerimaan data, serta pengelolaan kesalahan, diharapkan kajian ini dapat memberikan wawasan teoritis dan praktis mengenai cara kerja UDP dan socket programming, serta menjelaskan bagaimana data dapat dikirim secara efisien dalam jaringan dengan memanfaatkan protokol ini, sehingga relevan bagi pengembang dan peneliti di bidang teknologi informasi.

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam era globalisasi saat ini, penggunaan teknologi informasi menjadi sangat penting bagi setiap individu. Perkembangan teknologi memungkinkan berbagai perangkat, termasuk komputer, untuk saling terhubung dengan mudah melalui jaringan lokal maupun internet. Tujuan utama dari konektivitas ini adalah untuk memungkinkan komunikasi dan pertukaran data yang efisien. Namun, kualitas komunikasi yang baik sangat bergantung pada kemampuan transfer data yang akurat dan konsisten, yang dipengaruhi oleh perangkat keras dan perangkat lunak yang digunakan.

Protokol jaringan komputer berfungsi sebagai serangkaian aturan yang mengatur komunikasi antar komputer, memastikan data dapat dikirim dan diterima dengan benar. Tanpa adanya protokol, transfer data antar perangkat akan menjadi tidak konsisten, berpotensi menyebabkan gangguan komunikasi yang dapat mempengaruhi kinerja aplikasi. Salah satu protokol yang dikenal luas adalah UDP (User Datagram Protocol), yang menawarkan kecepatan pengiriman data tanpa memerlukan koneksi awal atau verifikasi pengiriman. Namun, kelemahan utama UDP adalah ketidakpastian dalam jaminan pengiriman dan pengurutan data, yang memerlukan pertimbangan cermat saat digunakan.

Dalam konteks pengembangan perangkat lunak, khususnya socket programming, UDP dapat diimplementasikan dalam berbagai bahasa pemrograman, termasuk Python. Penggunaan socket UDP memungkinkan pengiriman data secara efisien melalui jaringan, terutama untuk aplikasi yang memerlukan respons cepat seperti streaming video. Hal ini menunjukkan pentingnya pemahaman yang mendalam mengenai protokol dan teknologi yang digunakan untuk memenuhi kebutuhan komunikasi dalam aplikasi yang beragam, serta untuk merancang sistem yang dapat berfungsi secara optimal di berbagai lingkungan perangkat keras dan lunak.

1.2. Rumusan Masalah

Dari latar belakang yang ada, terdapat beberapa rumusan masalah yang diharapkan dapat terjawab setelah membaca laporan ini, antara lain sebagai berikut:

- Apa saja keuntungan dan kelemahan protokol UDP?
- Bagaimana cara menggunakan socket programming dengan protokol UDP?
- Apa langkah-langkah pengiriman data menggunakan protokol transport layer?

1.3. Batasan Masalah

Batasan masalah dari pembahasan laporan ini adalah sebagai berikut:

- Pembahasan berfokus pada protokol UDP, termasuk keuntungan dan kelemahannya, serta socket programming dengan pembuatan program sederhana menggunakan UDP dalam bahasa pemrograman Python.

- Pengiriman data akan mencakup metode dasar menggunakan protokol transport layer UDP, terbatas pada konteks komunikasi antar komputer tanpa membahas protokol lain seperti TCP.

1.4. Tujuan dan Manfaat

- Memahami protokol UDP, beserta dengan segala keuntungan dan kelemahannya.
- Mempelajari socket programming dengan membuat program sederhana yang memanfaatkannya sebagai fungsi utamanya.
- Membuat dan memahami cara pengiriman data sederhana lewat jaringan menggunakan protokol transport layer.

BAB II

PEMBAHASAN

1.1. Deskripsi Kerja Program

Program socket programming dengan UDP pada Python ini menggunakan dua sisi, yaitu *client* dan server, karena keduanya memiliki peran penting dalam komunikasi berbasis jaringan yang mengikuti model client-server.

```
1       server_socket = socket.socket(socket.AF_INET, socket.
    SOCK_DGRAM)
2       server_socket.bind((IP, port))
```

Gambar 1.1.1 Setting UDP server socket

Menggunakan python socket library, UDP server socket dapat di set-up sehingga UDP socket dapat mengirim dan menerima data melalui IPv4. Pada kode di atas, .bind() berfungsi untuk menghubungkan socket dengan IP address dan port spesifik sehingga server dapat mendengarkan data.

Client berfungsi sebagai perangkat yang mengirimkan permintaan dan berinteraksi langsung dengan pengguna, sedangkan server bertindak sebagai pengelola komunikasi, menerima pesan dari *client* dan meneruskannya ke semua *client* lain yang terhubung. Implementasi model ini penting karena, seperti yang dijelaskan oleh Kurose dan Ross (2017) dalam *Computer Networking: A Top-Down Approach*, "model client-server memungkinkan server bertindak sebagai pusat kontrol yang menerima dan mengelola permintaan dari *client*, mendukung komunikasi yang terkoordinasi di seluruh jaringan."

```
self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
self.client_socket.sendto(f"{username}:{password}:{port_password}".encode(), (ip, int(port)))
response, _ = self.client_socket.recvfrom(1024)
```

Gambar 1.1.2 Setting UDP client socket

Seperti setting UDP server, kode di atas juga membuat UDP client socket. Pada line selanjutnya terdapat formatted string untuk memisahkan username, password, dan port_password dan memisahkannya dengan colon “:”. Formatting ini dilakukan untuk mengirim sebuah message yang berisi username, password, dan port_password tersebut. Setelah itu terdapat .encode() yang akan melakukan konversi ke format byte karena socket akan mengirimkan data dalam format byte. Setelah itu (ip, int(port)) akan mengembalikan IP dan port dari server sehingga *message* dapat dikirim ke server. Semuanya berada di dalam sendto() untuk mengirimkan informasi login yang disandikan ke IP dan port server. Bagian response akan menunggu respons dari server dengan recvfrom(1024) yang berfungsi untuk menunggu penerimaan data dari server, dengan ukuran buffer 1024 byte.

Dalam program ini, server juga akan memvalidasi login dengan memeriksa username dan password *client*, memastikan hanya *client* yang menggunakan input yang dapat masuk ke dalam chatroom.

```
    if not username or not password or not port_password:
        messagebox.showwarning("OooHHH NoOOOooo", "Username, password, atau port password tidak boleh kosong yaaa (╥﹏╥)")
        return
    if os.path.exists('users.csv'):
        with open('users.csv', mode='r', newline='') as file:
            reader = csv.reader(file)
            if not any(row[0] == username and row[1] == password for row in reader):
                messagebox.showwarning("OooHHH NoOOOooo", "Username atau password kamu masih salah (>Д<)゜゜")
                return
    else:
        messagebox.showwarning("OooHHH NoOOOooo", "Akun kamu belum terdaftar, ayoo registrasi dulu ( ˘ ³˘)")
        return
```

Gambar 1.1.3 Kondisional untuk memvalidasi login

Setelah berhasil login, *client* dapat mengirim pesan yang kemudian akan didistribusikan oleh server ke semua *client* lain. Proses ini disebut broadcasting, yang memungkinkan pesan diterima oleh banyak *client* secara bersamaan. Selain itu, program menggunakan UDP karena lebih ringan dan cepat dibandingkan TCP, meskipun kurang andal. UDP tidak memerlukan koneksi yang persisten, membuatnya ideal untuk aplikasi real-time seperti chatroom, di mana kecepatan lebih penting daripada keandalan penuh. Seperti yang juga dijelaskan oleh Kurose, UDP "cocok untuk aplikasi yang memerlukan low-latency, seperti streaming media atau chatroom, di mana kehilangan beberapa pesan tidak akan merusak pengalaman pengguna." Oleh karena itu, penggunaan dua sisi, *client* dan server, serta protokol UDP, memastikan interaksi yang efisien dalam aplikasi ini.

1.2. Deskripsi Tata Cara dan Lingkungan Pengujian

Pengujian dilakukan di jaringan LAN (Local Area Network) dengan waktu yang dibutuhkan untuk mengirim dan menerima data antara perangkat sangat singkat. Dalam pengujian ini, server dan client pertama dijalankan di laptop Windows 10 dan *client* kedua dijalankan di laptop MacOS Monterey Ver 12.1 yang juga terhubung ke jaringan WiFi yang sama. Dengan cara ini, kedua perangkat dapat saling terhubung untuk berkomunikasi.

Koneksi di jaringan yang sama sangat penting. Jika salah satu perangkat terhubung ke jaringan yang berbeda, *client* tidak akan bisa bergabung atau terhubung dengan server. Oleh karena itu, sebelum melakukan pengujian, semua perangkat perlu diatur untuk terhubung ke jaringan yang sama. Selain itu, untuk menghubungkan *client* di perangkat lain ke server, diperlukan alamat IP dari perangkat server. *Client* di MacBook harus memasukkan alamat IP tersebut saat program dijalankan. Langkah ini penting agar *client* tahu ke mana harus terhubung, sehingga komunikasi bisa berlangsung dengan baik.

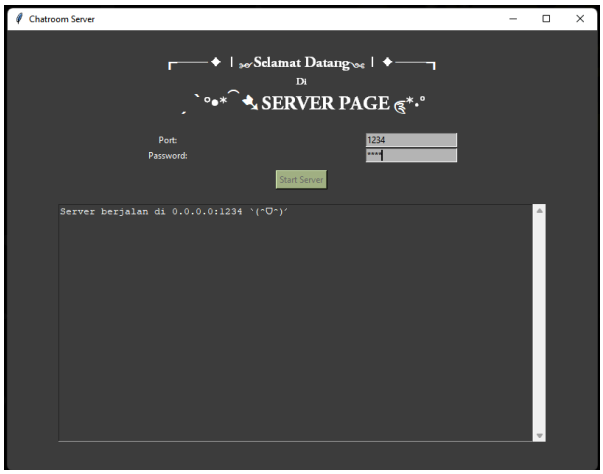
Dengan semua pengaturan ini, pengujian menunjukkan bahwa server dapat mengirim dan menerima pesan dengan *client* di kedua perangkat dengan cepat dan tanpa gangguan. Hal ini

sesuai dengan karakteristik dari protokol UDP (User Datagram Protocol) yang digunakan, yang memang tidak menjamin urutan atau pengiriman ulang pesan jika terjadi masalah. Pengujian ini menunjukkan bahwa pengaturan lingkungan yang tepat sangat berpengaruh pada kinerja aplikasi yang diuji.

1.3. Tampilan Antarmuka dan Hasil Pengujian

Berikut merupakan detail tata cara dalam melakukan pengujian:

1.3.1. Konfigurasi dan Jalankan Server

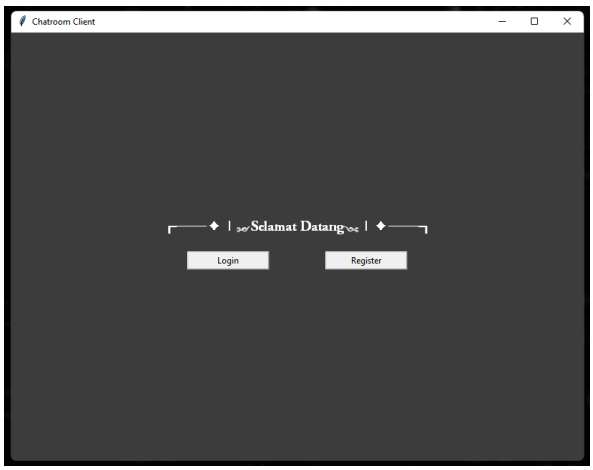


Gambar 1.3.1 Tampilan pada laman server

Setelah lingkungan siap, langkah pertama adalah menjalankan server. Pada perangkat yang bertindak sebagai server, program `server.py` dijalankan dengan mengatur IP server dan port yang diperlukan. Pengaturan IP server dapat dilakukan dengan menggunakan 0.0.0.0 yang memungkinkan server menerima koneksi dari semua antarmuka jaringan yang tersedia ataupun IPv4 Address dari perangkat yang digunakan. Password rahasia juga ditentukan sebagai mekanisme validasi login bagi *client*.

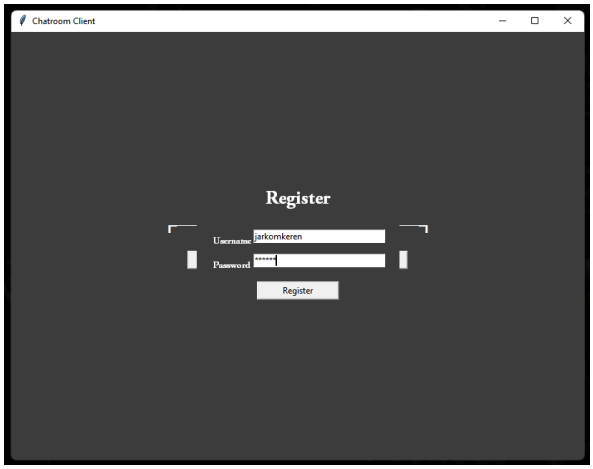
1.3.2. Konfigurasi dan Jalankan Client

Pada perangkat *client*, program `client.py` dijalankan, dan pada tampilan awal akan menampilkan pilihan *homepage* atau laman utama dengan opsi Login atau Register.

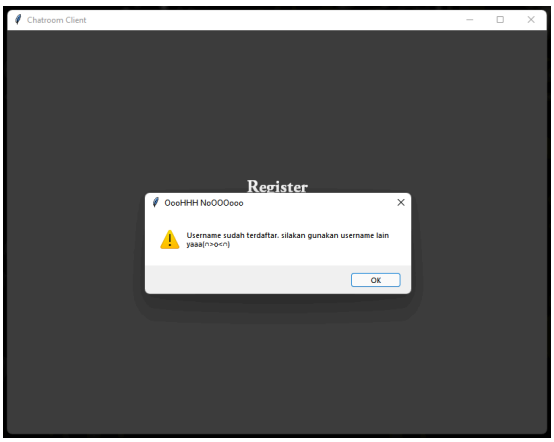


Gambar 1.3.2 Tampilan pada laman utama

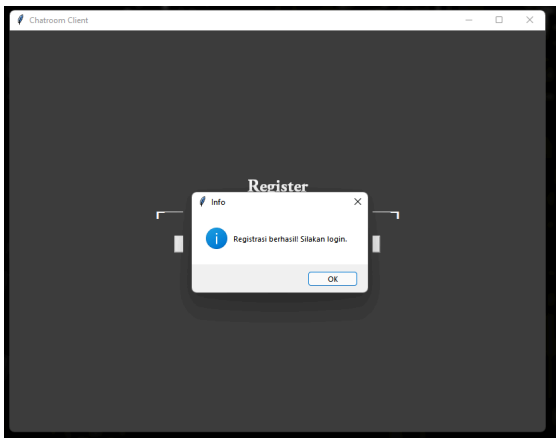
Client diminta untuk melakukan registrasi terlebih dahulu jika belum memiliki akun, dengan memasukkan *username* dan *password* sebelum melakukan login. Setiap akun yang ingin masuk ke *roomchat* harus sudah terdaftar. Jika *username* yang dimasukkan sudah digunakan, akan muncul peringatan bahwa *username* tersebut telah terpakai, dan *client* harus memasukkan *username* yang unik.



Gambar 1.3.3 Tampilan pada halaman Register

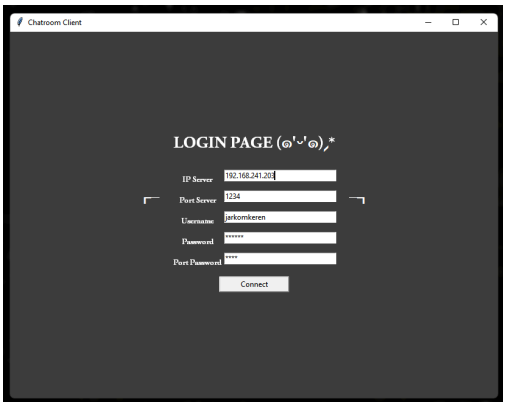


Gambar 1.3.4 Tampilan ketika username telah digunakan

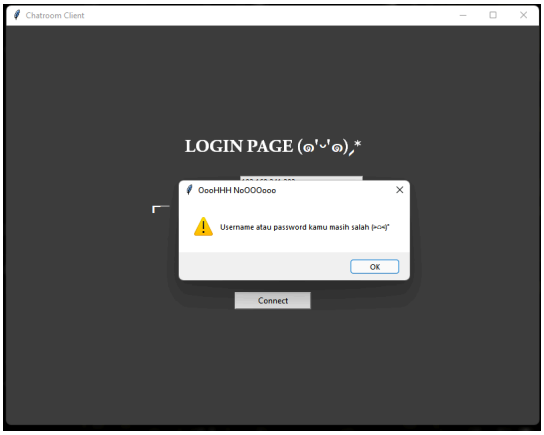


Gambar 1.3.5 Tampilan ketika registrasi berhasil dan username unik

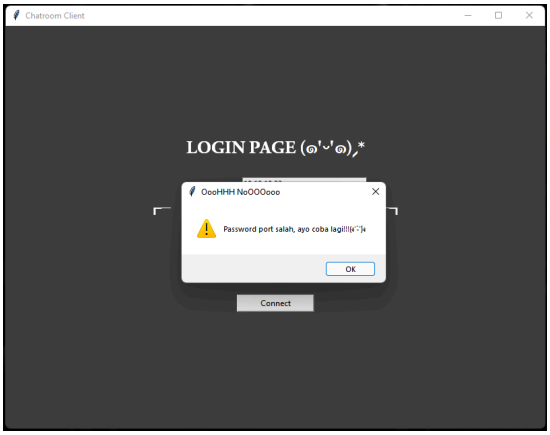
Setelah berhasil registrasi, client harus melakukan login dengan memasukkan username dan password yang telah didaftarkan pada registrasi, serta mengisi alamat IP server, port, dan kata sandi ruang obrolan yang ditentukan dalam program `server.py`.



Gambar 1.3.6 Tampilan pada laman login

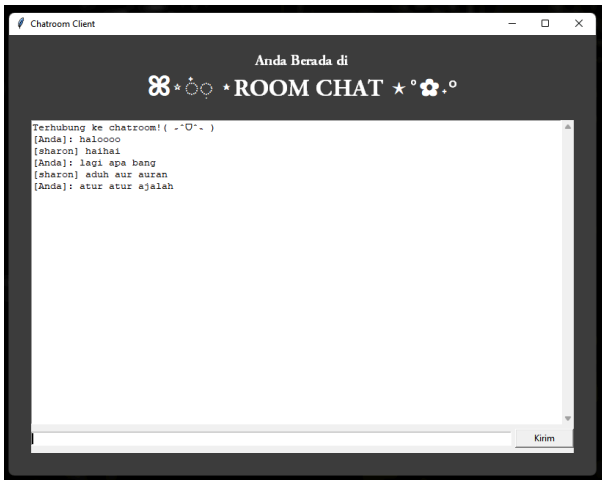


Gambar 1.3.7 Tampilan ketika username atau password user tidak sesuai



Gambar 1.3.8 Tampilan ketika port password tidak sesuai

1.3.3. Pengujian Pengiriman Pesan



Gambar 1.3.9 Tampilan pada roomchat

Setelah berhasil terhubung, *client* dapat mulai mengirim pesan yang akan diteruskan oleh server ke semua *client* lain yang terhubung. Pengujian dilakukan dengan mengirimkan pesan dari satu *client* dan memeriksa apakah *client* lain menerima pesan tersebut dalam waktu yang singkat dan tanpa keterlambatan signifikan. Server juga mencatat setiap pesan yang diterima dan melakukan proses broadcasting agar pesan tersebut dapat diterima oleh semua *client* kecuali pengirim. Penggunaan UDP memungkinkan pengiriman pesan secara cepat tanpa memerlukan koneksi persisten, sehingga latensi pada pengiriman pesan minimal. Hal ini menunjukkan bahwa mekanisme pengiriman pesan berjalan sesuai harapan.

1.3.4. Pengujian Penghentian Program

Penghentian program juga diuji dengan menutup *client* secara manual menggunakan Ctrl + Q pada windows dan Command + Q pada MacOS. Penutupan program ini dapat dilakukan karena adanya implementasi kondisional jika melakukan KeyboardInterrupt. Pemberhentian program juga dapat dilakukan dengan keluar dari terminal secara langsung. Pengujian ini memastikan bahwa server tetap berjalan meskipun satu *client* keluar dari jaringan, dan *client* lain masih dapat terus mengirim pesan. Jika semua *client* keluar dari chatroom, server tetap aktif dan siap menerima koneksi baru tanpa harus di-restart. Pengujian ini bertujuan untuk menguji ketahanan server dalam menangani *client* yang keluar dari jaringan tanpa mengganggu jalannya program secara keseluruhan.

1.4. Source Code

SOURCE CODE
https://github.com/sharondarmap/06_Tugas-Socket-Programming.git

BAB III
PENUTUP

3.1. Kesimpulan

1. UDP merupakan protokol layer transport tanpa connection. UDP bersifat simpel dan cepat, tetapi tidak reliable. Bisa terjadi data loss tanpa diketahui.
2. UDP bekerja dengan merangkum data ke dalam datagram dan mengirimkannya langsung ke alamat IP dan port tujuan. UDP tidak melakukan pemeriksaan atau transmisi ulang, sehingga lebih cepat tetapi kurang dapat diandalkan dibanding TCP.

3.2. Lampiran Link Terkait

Link Repository GitHub:
https://github.com/sharondarmap/06_Tugas-Socket-Programming.git
Link Youtube Demo:
Demonstrasi ZepuhZarkom Chatroom

3.3. Keterangan Spesifikasi Tugas

Tabel 3.1 : Tabel Penyelesaian Spesifikasi Wajib

Spesifikasi	Selesai
Server mampu menerima pesan yang dikirim client dan mencetaknya ke layar.	☑
Server mampu meneruskan pesan satu client ke client lain.	☑
Client mampu mengirimkan pesan ke server dengan IP dan port yang ditentukan pengguna.	☑
Client mampu menerima pesan dari client lain (yang diteruskan oleh server), dan mencetaknya ke layar.	☑
Client harus memasukkan *password* untuk dapat bergabung ke chatroom.	☑
Client memiliki username yang unik.	☑

Tabel 3.2 : Tabel Penyelesaian Spesifikasi Opsional

Spesifikasi	Selesai
Aplikasi mengimplementasikan TCP over UDP. **Note: asisten sangat menyarankan untuk mengerjakan spesifikasi ini, karena akan memberikan pemahaman kuat terkait TCP.**	☐
Seluruh pesan dienkripsi menggunakan algoritma kriptografi klasik simetris,	☐

misal cipher Vigenere atau Caesar.	
Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern simetris, misal cipher RC4.	☐
Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern asimetris, misal cipher RSA, atau kombinasi algoritma kriptografi modern asimetris dan modern simetris.	☐
Seluruh pesan dienkripsi menggunakan algoritma Double-Rachet atau MLS.	☐
Aplikasi memiliki GUI.	☑
Aplikasi mampu digunakan untuk mengirimkan dan menerima pesan bertipe file biner.	☐
Aplikasi mampu menunjukkan apabila integritas pesan telah rusak, baik dengan memanfaatkan *checksum* ataupun *digital signature*.	☐
Aplikasi mampu menyimpan pesan-pesan lampau meskipun telah ditutup; mekanisme dan tempat penyimpanan bebas, baik di *client* maupun di *server*.	☐
Aplikasi mampu mengotentikasi pengguna.	☑
Aplikasi diprogram menggunakan paradigma *object oriented programming* atau pemrograman berorientasi objek	☑

3.3. Pembagian Tugas

Tabel 3.3 : Tabel Pembagian Tugas

Nama	Detail Tugas
Sharon Darma Putra	Spek wajib 1 Spek wajib 2 Spek wajib 3 Spek wajib 4 Spek opsional OOP Menyusun dokumen
Nakeisha Valya Shakila	Spek wajib 5 Spek wajib 6 Spek opsional GUI Spek opsional user auth Menyusun dokumen Mengedit video

DAFTAR PUSTAKA

Redaksi Jagoan Hosting. (2023, September 19). *Pengertian OOP (Object Oriented Programming) dan 4 Prinsipnya*. Blog Jagoan Hosting; Jagoan Hosting Indonesia. https://www.jagoanhosting.com/blog/oop-adalah/

Serhii Orlivskyi. (2023, August 18). *A Complete Guide to Socket Programming in Python*. Datacamp.com; DataCamp. https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python

Jennings, N. (2023, November 16). *Socket programming in Python (Guide)*. https://realpython.com/python-sockets/

Adjie Gery Ramadhan. (2024, June 24). Apa itu Protokol UDP (User Datagram Protocol) ? *D3 Teknologi Telekomunikasi*. https://dte.telkomuniversity.ac.id/apa-itu-protokol-udp-user-datagram-protocol/