LAPORAN TUGAS PRAKTIKUM PEMROGRAMAN JARINGAN

Mata Kuliah: Bahasa Pemrograman Jaringan Komputer

Di Susun Oleh:

Krisnayanty Kesia Patanda (231401008)

Dosen Pengampu: Ucok, S.Kom.,MT

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA TIMUR

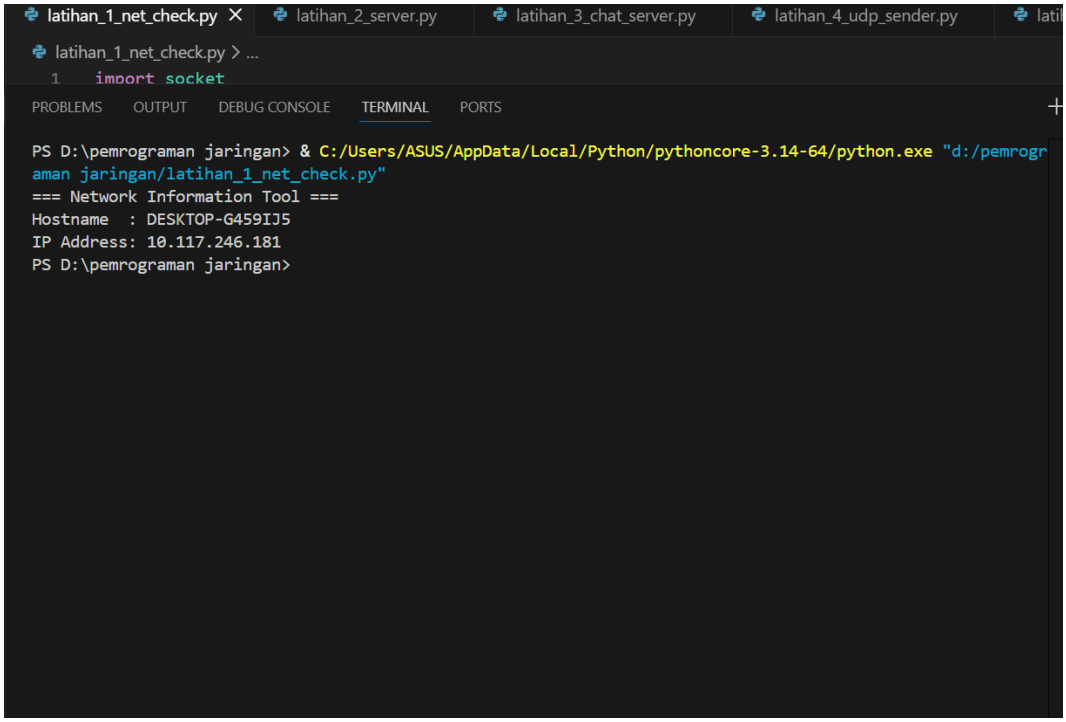
MAKASSAR

2026

BAB 1 KONSEP DASAR PEMROGRAMAN JARINGAN

Program pengecekan informasi jaringan menggunakan Python bertujuan untuk menampilkan hostname dan IP address komputer. Program ini memanfaatkan module socket untuk mengambil data jaringan dari sistem operasi, di mana gethostname() digunakan untuk mengetahui nama komputer dan gethostbyname() untuk memperoleh alamat IP. Hasil informasi kemudian ditampilkan ke layar menggunakan fungsi print(). Program ini melatih pemahaman dasar tentang pemrograman jaringan, penggunaan library bawaan Python, serta pengenalan konsep hostname dan IP address dalam jaringan komputer.

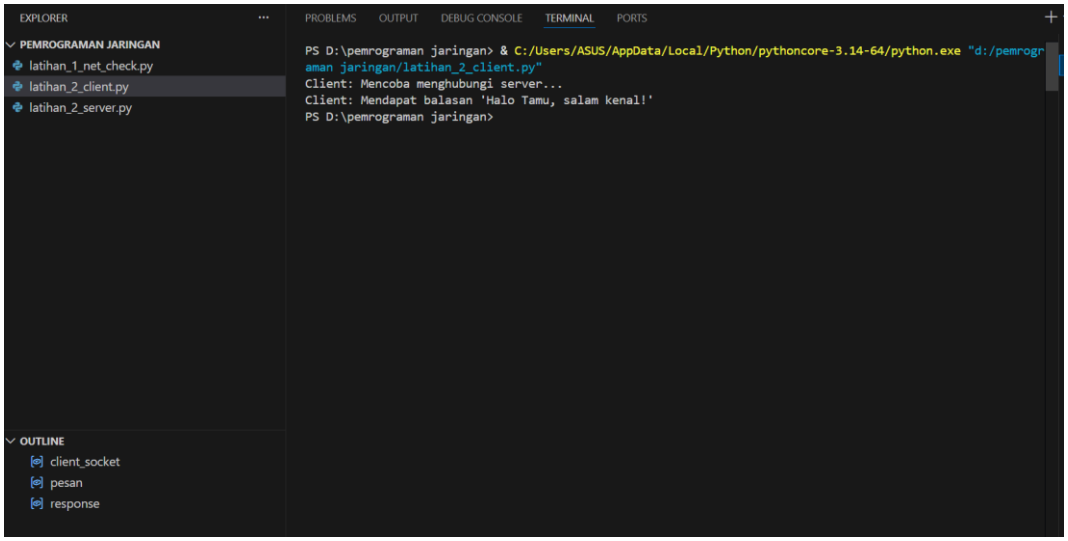
Hasil:

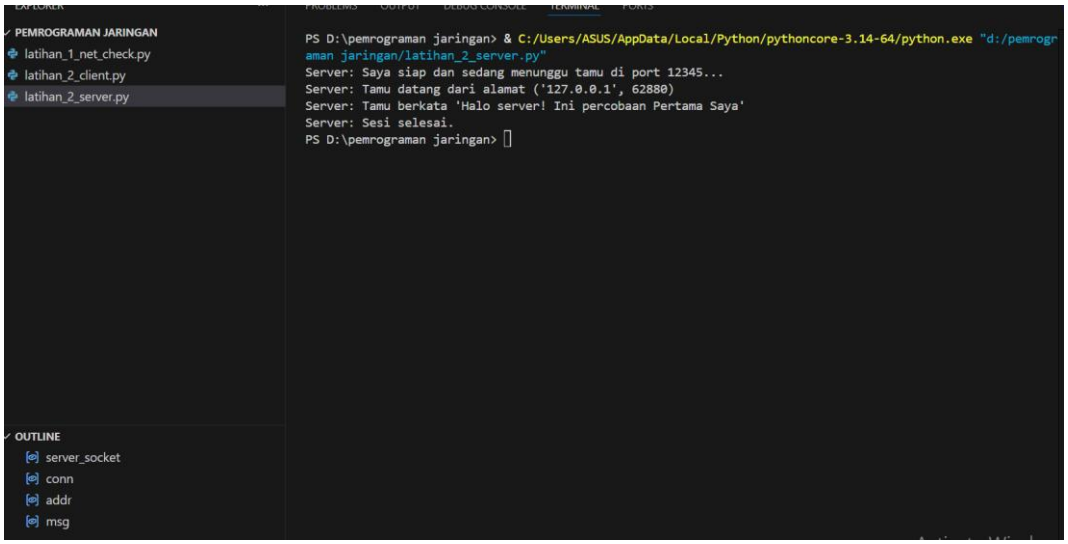


BAB 2 SOCKET API DASAR

Pada percobaan ini, komunikasi jaringan dilakukan antara client dan server yang berjalan pada satu komputer menggunakan alamat localhost . Server dijalankan terlebih dahulu untuk membuka port dan menunggu koneksi masuk, kemudian client melakukan koneksi ke server dan mengirimkan pesan percobaan. Dari hasil eksekusi terlihat bahwa server berhasil menerima pesan dari client, menampilkan alamat client, serta mengirimkan balasan yang diterima kembali oleh client. Hal ini membuktikan bahwa proses koneksi socket, pengiriman pesan, penerimaan respon, dan penutupan koneksi berjalan dengan baik sesuai konsep dasar pemrograman jaringan menggunakan Python.

Hasil:

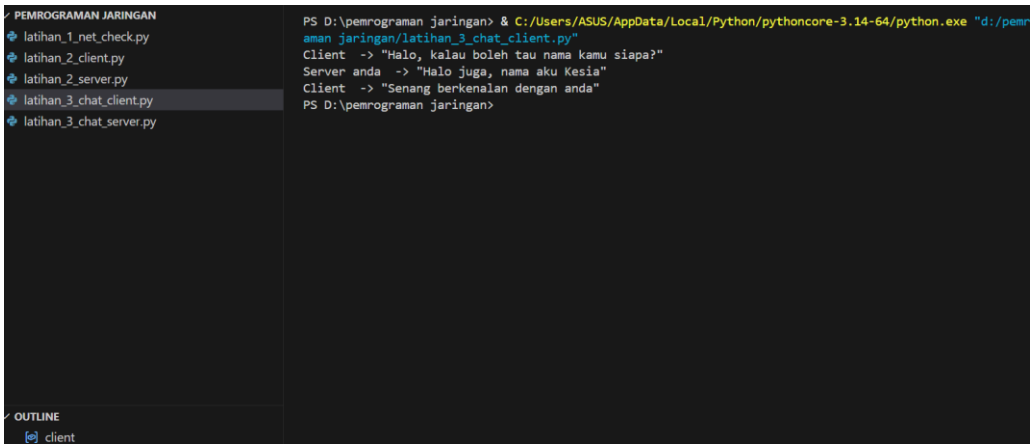


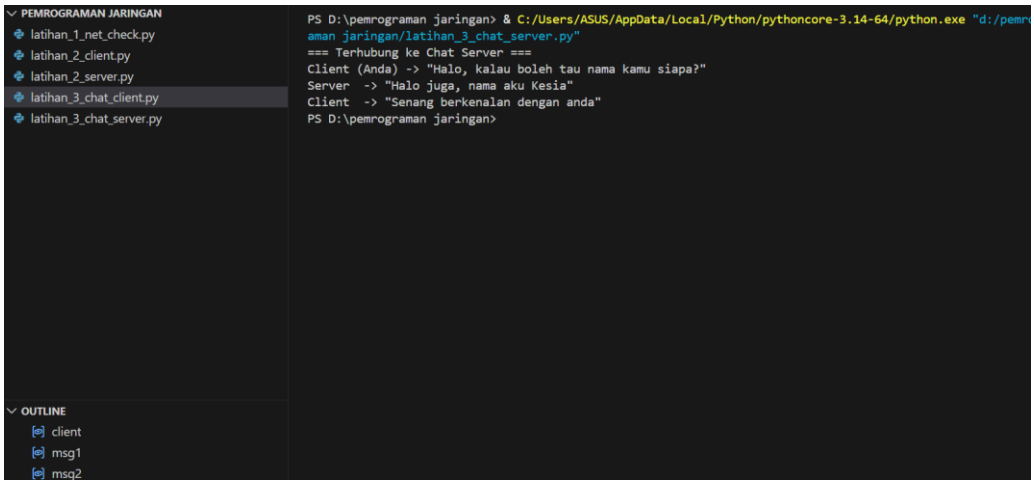


BAB 3 PROTOKOL TCP (APLIKASI CHAT)

Pada materi ini menampilkan implementasi aplikasi chat berbasis protokol TCP menggunakan bahasa pemrograman Python, di mana komunikasi dilakukan antara satu server dan satu client melalui koneksi jaringan yang andal. Server terlebih dahulu dijalankan untuk membuka port dan menunggu koneksi masuk dari client, kemudian client melakukan koneksi ke server untuk memulai sesi percakapan. Setelah koneksi TCP berhasil terbentuk, client mengirimkan pesan pertama yang diterima oleh server, lalu server membalas pesan tersebut sehingga terjadi pertukaran pesan secara dua arah. Seluruh proses komunikasi ini berlangsung dalam satu sesi koneksi yang sama, sehingga setiap pesan dapat dikirim dan diterima secara berurutan tanpa kehilangan data. Penggunaan protokol TCP pada aplikasi chat ini memastikan keandalan komunikasi, menjaga urutan pesan, serta menjamin bahwa data yang dikirim sampai ke tujuan dengan benar, sesuai dengan prinsip dasar pemrograman jaringan aplikasi chat.

Hasil:

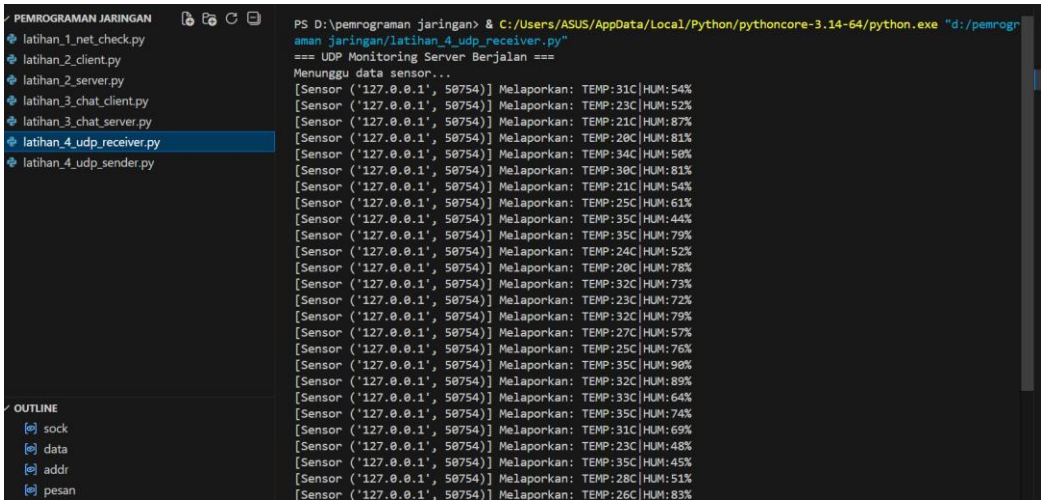


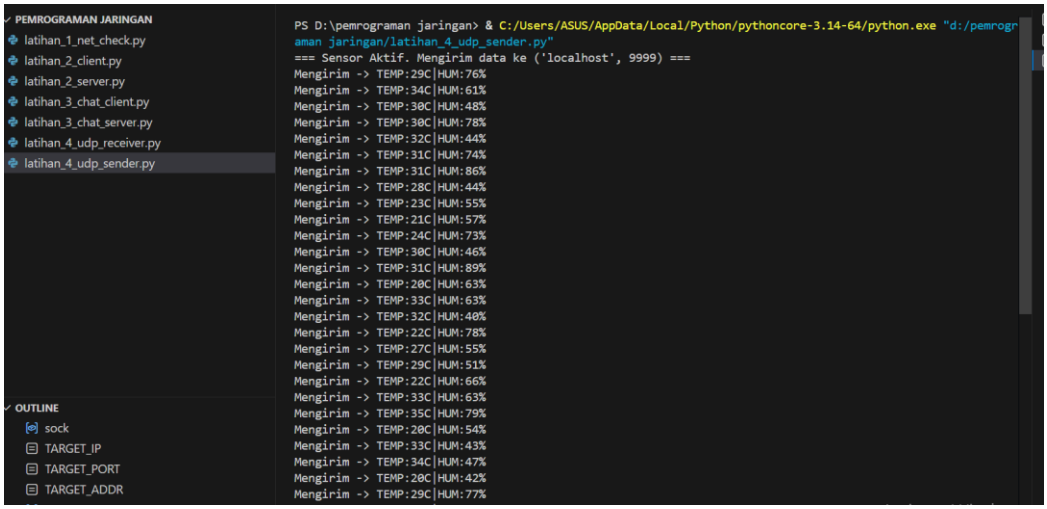


BAB 4 PROTOKOL UDP (STREAMING & BROADCASTING

Program ini menunjukkan penerapan komunikasi jaringan menggunakan protokol UDP pada simulasi pengiriman data sensor. Program sender bertindak sebagai sensor yang secara terus-menerus mengirimkan data suhu (TEMP) dan kelembapan (HUM) ke alamat localhost dan port tertentu tanpa membangun koneksi terlebih dahulu, sedangkan program receiver berfungsi sebagai server monitoring yang menerima dan menampilkan data sensor tersebut. Karena menggunakan protokol UDP, pengiriman data berlangsung cepat dan sederhana tanpa mekanisme pengecekan koneksi, sehingga sesuai untuk aplikasi monitoring sensor yang membutuhkan kecepatan pengiriman data secara real-time.

Hasil:

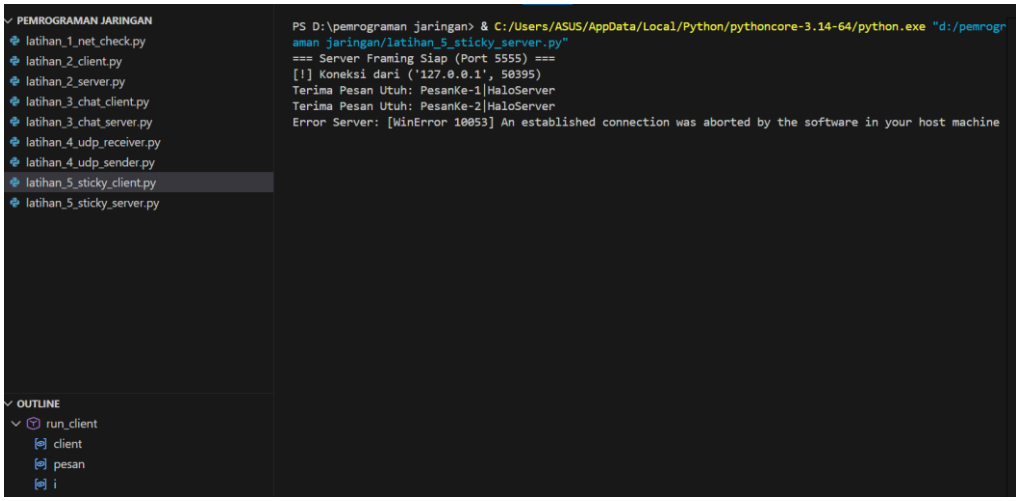


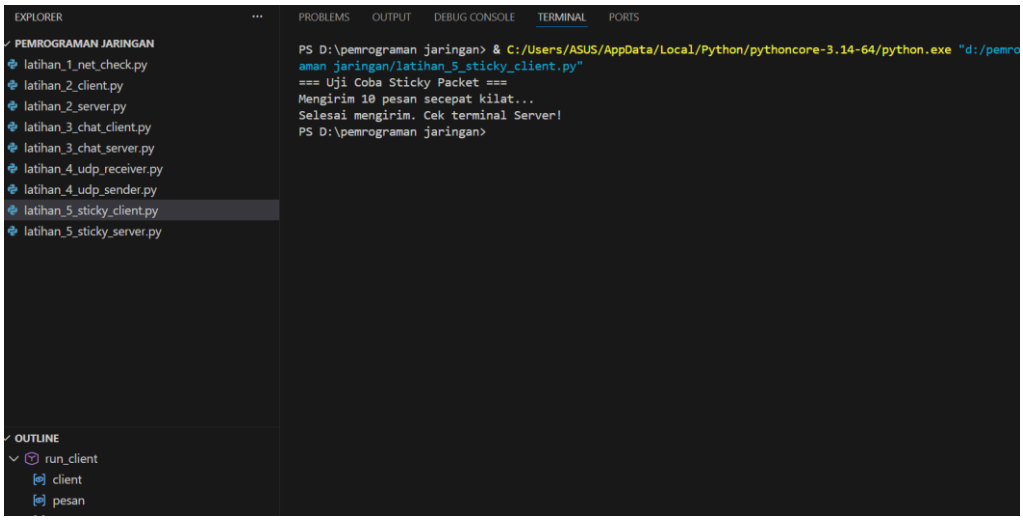


BAB 5 ERROR HANDLING & FRAMING DAT

Materi ini menjelaskan konsep sticky packet (packet lengket) pada komunikasi TCP melalui simulasi client–server menggunakan Python, di mana server melakukan framing data untuk memastikan pesan diterima secara utuh. Client mengirimkan beberapa pesan secara cepat ke server melalui koneksi TCP yang bersifat stream, sehingga data dapat tiba secara beruntun tanpa batas pesan yang jelas. Di sisi server, mekanisme framing digunakan untuk memisahkan dan menyusun kembali data yang diterima agar setiap pesan dapat dikenali dengan benar, seperti terlihat dari keluaran “Terima Pesan Utuh”. Proses ini menunjukkan bahwa pada TCP tidak ada pemisahan paket secara otomatis seperti pada UDP, sehingga programmer perlu menambahkan logika khusus untuk menangani penggabungan atau pemotongan data agar komunikasi client–server tetap akurat dan stabil.

Hasil:

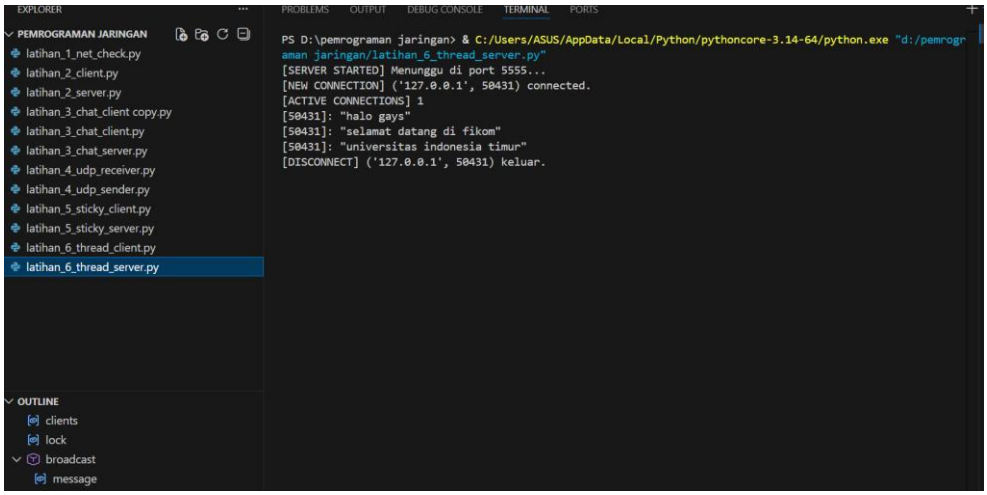


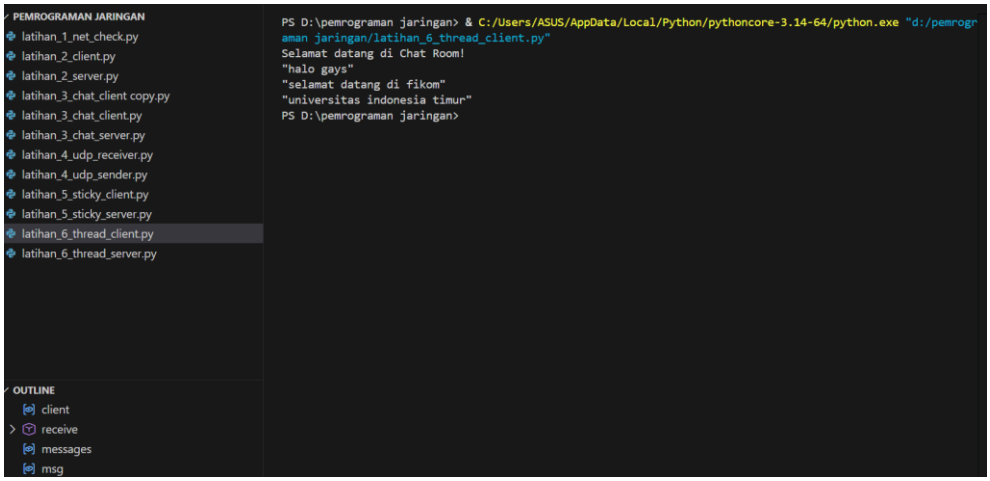


BAB 6 CONCURRENCY PART I-THREADING

Materi ini menunjukan implementasi program chat client–server berbasis TCP dengan multithreading menggunakan Python, di mana server mampu menangani beberapa koneksi klien secara bersamaan. Server dijalankan pada port 5555 dan menampilkan status seperti server dimulai, koneksi baru dari klien, jumlah koneksi aktif, pesan yang diterima, hingga notifikasi klien keluar. Setiap klien yang terhubung dilayani oleh thread terpisah sehingga komunikasi dapat berlangsung tanpa saling mengganggu. Di sisi klien, program menampilkan pesan sambutan "Selamat datang di Chat Room!" dan memungkinkan pengguna mengirim beberapa pesan ke server. Konsep ini menunjukkan penggunaan socket, thread, dan mekanisme komunikasi dua arah (send–receive) yang umum digunakan pada aplikasi jaringan real-time seperti chat room.

Hasil:

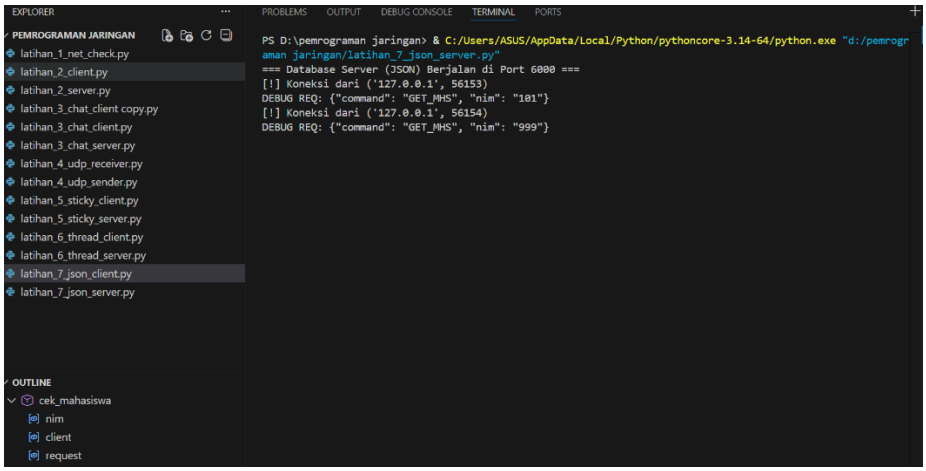


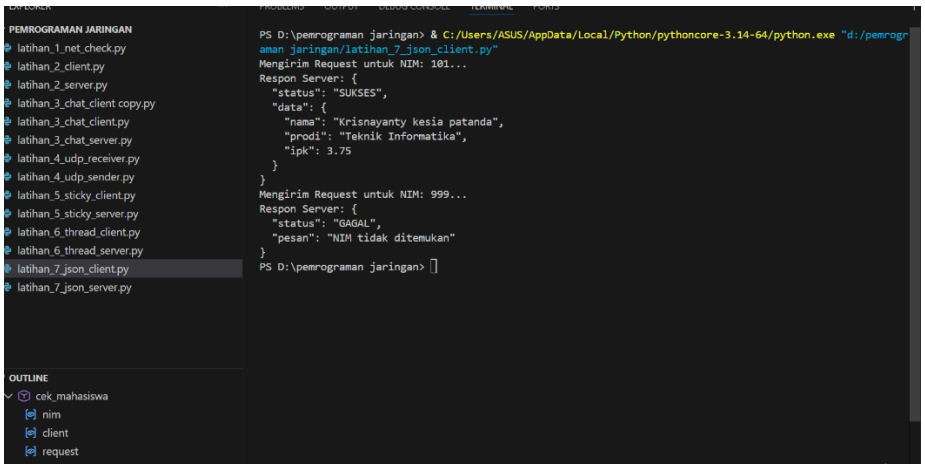


BAB 7 SERIALISASI DATA (JSON & PICKEL)

Pada materi ini ditunjukkan proses pengujian layanan server berbasis JSON pada pemrograman jaringan yang dijalankan melalui terminal. Server dijalankan pada port 6000 dan menampilkan informasi setiap koneksi yang masuk dari client, termasuk alamat IP dan port client serta data request JSON yang diterima. Client kemudian mengirimkan permintaan pengecekan data mahasiswa berdasarkan NIM, dan hasil respon server ditampilkan langsung di sisi client dalam bentuk JSON. Dari hasil tersebut terlihat dua kondisi, yaitu saat NIM valid server mengembalikan status SUKSES beserta data mahasiswa, dan saat NIM tidak valid server mengembalikan status GAGAL dengan pesan kesalahan. Tampilan ini memperlihatkan alur koneksi, proses request–response, serta cara server dan client saling berinteraksi secara real time menggunakan socket dan format JSON.

Hasil:

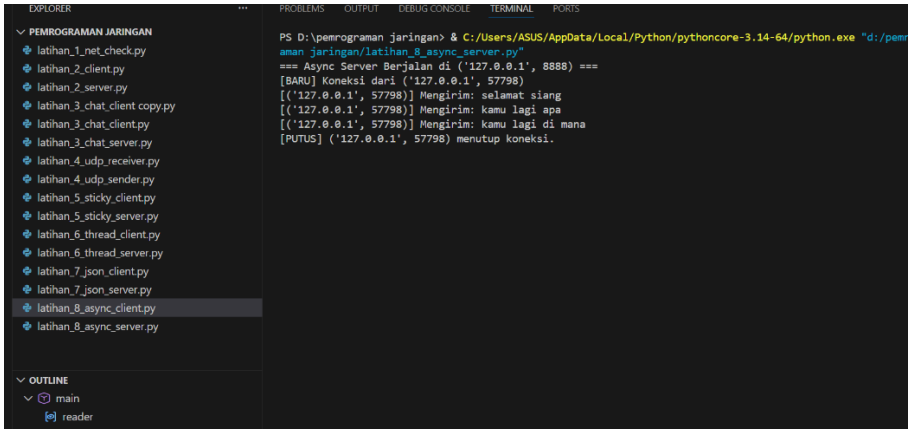


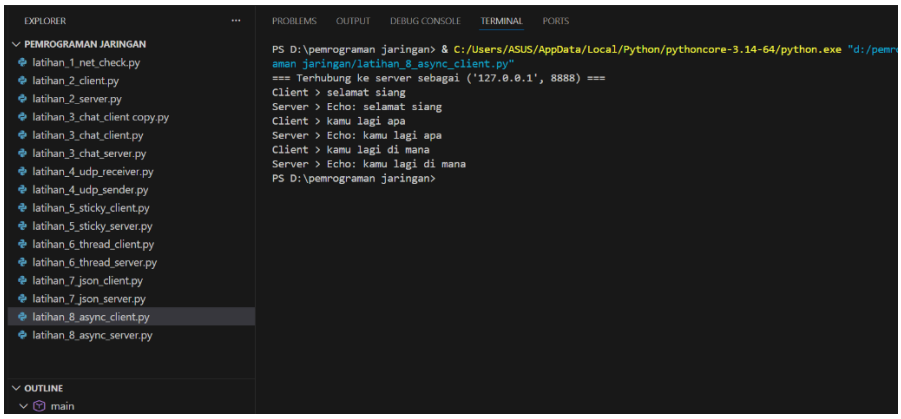


BAB 8 ASYNCHRONOUS I/O (CONCURRENCY PART II)

Program ini menunjukkan penerapan komunikasi client–server asynchronous (asyncio) pada pemrograman jaringan Python, di mana server berjalan non-blocking pada alamat 127.0.0.1 port 8888 dan mampu menangani koneksi client tanpa harus menunggu proses lain selesai. Pada gambar terlihat server async berhasil berjalan, menerima koneksi dari client, lalu membaca pesan yang dikirim seperti *"selamat siang"*, *"kamu lagi apa"*, dan *"kamu lagi di mana"*, kemudian memprosesnya secara berurutan dalam satu koneksi. Di sisi client, pengguna mengirim pesan melalui terminal dan langsung menerima balasan dari server berupa *Echo*, yang menandakan server membaca data melalui reader dan mengirimkan kembali respons menggunakan writer. Setelah komunikasi selesai, client menutup koneksi dan server mendeteksi kondisi tersebut lalu menampilkan status *memutus koneksi*. Alur ini membuktikan bahwa model async memungkinkan komunikasi jaringan berjalan efisien, responsif, dan terstruktur tanpa menggunakan banyak thread.

HASIL:

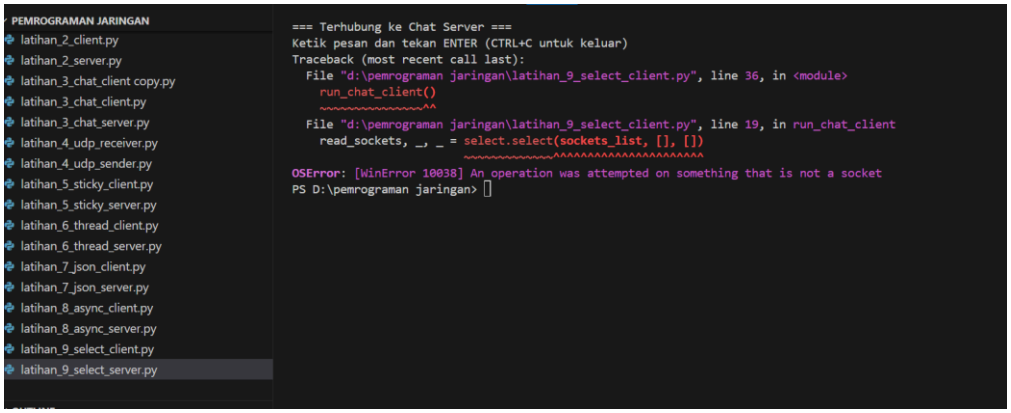


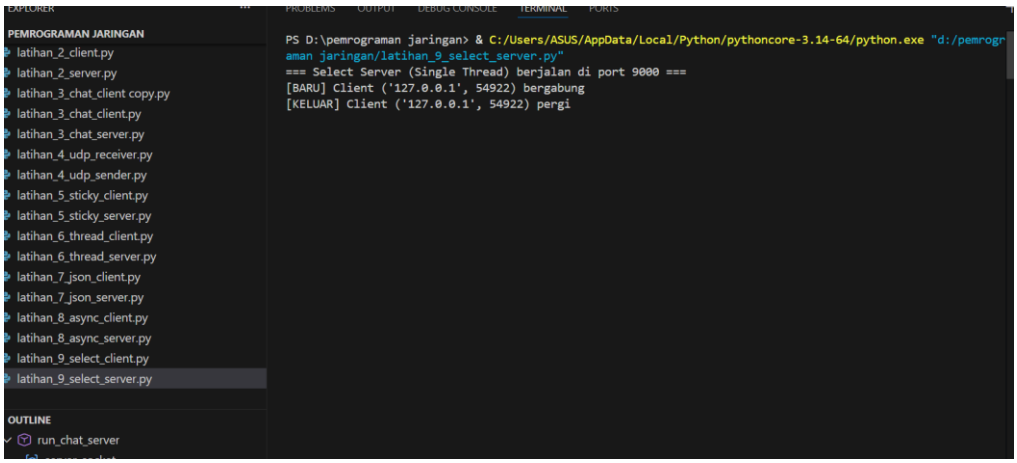


BAB 9 I/O MULTIPLEXING (SELECT&POLL)

Program ini menjelaskan cara kerja chat client–server berbasis select() pada Python yang memungkinkan server memantau banyak koneksi secara efisien dalam satu proses. Pada tampilan server terlihat bahwa socket utama berhasil membuka layanan di port 9000, menerima client baru, lalu mendeteksi saat client tersebut keluar tanpa membuat server berhenti. Sebaliknya, pada sisi client muncul pesan error yang menandakan adanya kesalahan saat pemanggilan select(), yaitu ketika daftar socket yang dipantau berisi objek yang tidak valid sebagai socket. Dari sini dapat dipahami bahwa select() hanya dapat bekerja dengan descriptor socket yang aktif, sehingga pengelolaan daftar socket menjadi bagian penting dalam arsitektur program. Materi ini menekankan konsep I/O multiplexing, di mana satu thread dapat menangani banyak koneksi sekaligus dengan tetap menjaga stabilitas dan kontrol alur komunikasi jaringan.

HASIL:

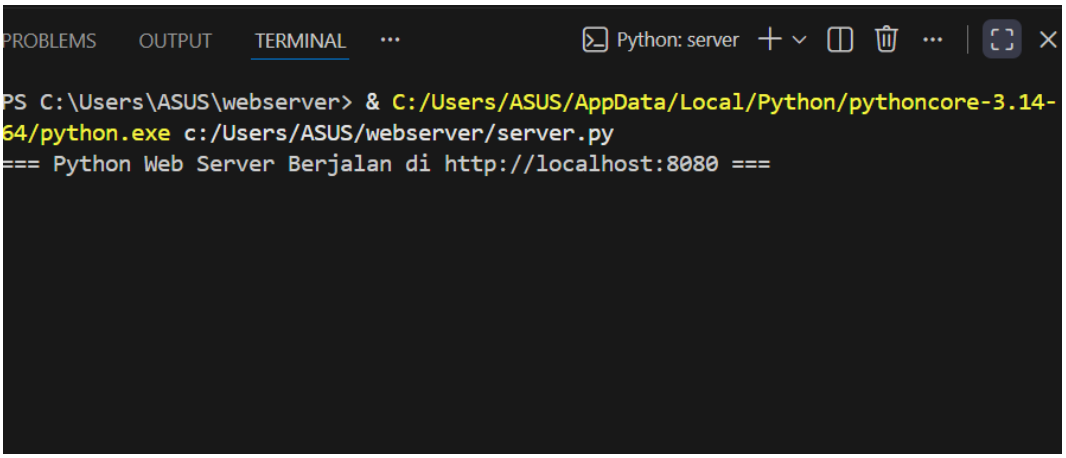




BAB 10 PROTOKOL HTTP & WEB SERVER

Program ini menunjukkan implementasi web server sederhana menggunakan Python Socket yang berjalan secara lokal pada alamat http://localhost:8080. Pada terminal terlihat server berhasil dijalankan dan siap menerima permintaan dari browser, sedangkan pada tampilan browser ditunjukkan halaman HTML sederhana dengan pesan *"Web Server Berhasil"* dan keterangan bahwa server berjalan menggunakan Python Socket. Alur kerjanya adalah server membuka socket, melakukan binding ke port 8080, lalu menunggu request HTTP dari client (browser), setelah itu server mengirimkan respons berupa header HTTP dan konten HTML yang kemudian dirender oleh browser. Materi ini memperkenalkan konsep dasar komunikasi HTTP di atas TCP, hubungan antara browser sebagai client dan Python sebagai server, serta membuktikan bahwa tanpa framework tambahan pun Python dapat digunakan untuk membangun web server dasar.

HASIL:



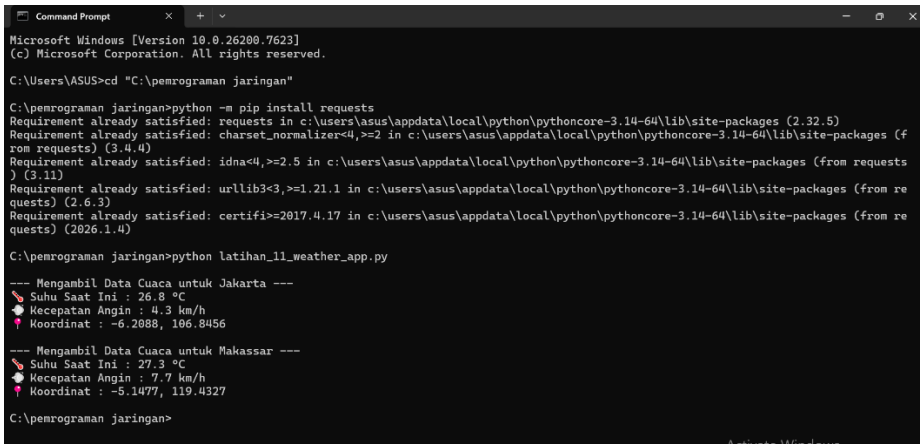
Web Server Berhasil

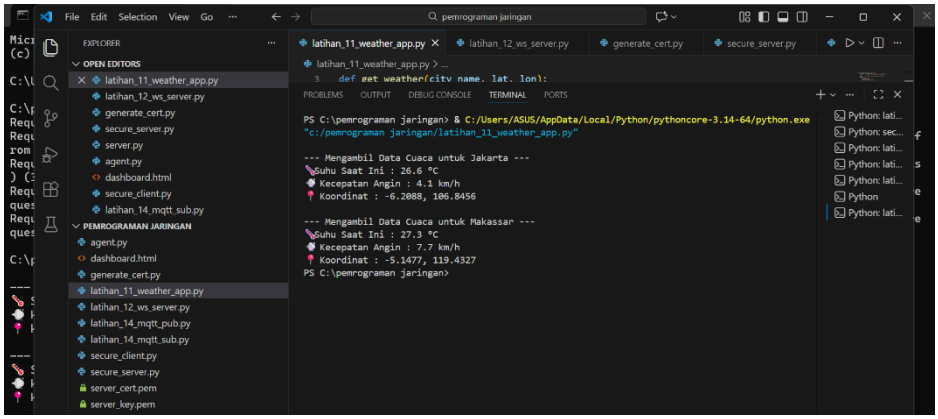
Ini berjalan menggunakan Python Socket

BAB 11 REST API & WEB SERVER

Program ini merupakan aplikasi client jaringan berbasis HTTP menggunakan library requests pada Python untuk mengambil data cuaca secara real-time dari layanan web (API). Pada gambar terlihat proses instalasi library requests yang memastikan dependensi tersedia, lalu program latihan_11_weather_app.py dijalankan melalui terminal dan berhasil menampilkan informasi cuaca untuk beberapa kota seperti Jakarta dan Makassar. Aplikasi ini bekerja dengan mengirim permintaan HTTP ke server cuaca menggunakan koordinat lintang dan bujur, kemudian menerima respons dalam format data yang diproses untuk menampilkan suhu saat ini, kecepatan angin, serta koordinat lokasi. Materi ini memperkenalkan konsep komunikasi client–server berbasis web service, pemanfaatan API eksternal dalam pemrograman jaringan, serta bagaimana Python dapat digunakan sebagai client untuk mengambil dan mengolah data dari internet secara terstruktur.

HASIL:





BAB 12 REAL-TIME COMMUNICATION (WEBSOCKET)

Program ini merupakan aplikasi pemantauan harga saham BBCA secara real-time yang menggunakan WebSocket sebagai media komunikasi antara server dan client. Pada sisi server (Python), terlihat server berjalan di ws://localhost:6789 dan secara kontinu melakukan broadcast data harga saham dalam format JSON (berisi simbol saham, harga, dan timestamp) ke client yang terhubung. Setiap kali ada client baru bergabung, server langsung mengirimkan update harga terbaru. Pada sisi client (dashboard HTML), halaman web menampilkan judul *Pantauan Saham BBCA (Real-time)* dan memperbarui harga saham (misalnya Rp 8066) secara otomatis tanpa perlu refresh halaman, menandakan koneksi WebSocket berhasil dengan status Connected to Server. Dengan mekanisme ini, perubahan data dari server dapat langsung diterima dan ditampilkan secara instan di browser.

HASIL:

```
:\pemrograman jaringan>python latihan_12_ws_server.py
== WebSocket Server running on ws://localhost:6789 ===
NEW] Client bergabung.
BROADCAST] {"symbol": "BBCA", "price": 8125, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8240, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8328, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8002, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8436, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8375, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8303, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8108, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8182, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8066, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8450, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8148, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8139, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8333, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8263, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8124, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8362, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8265, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8136, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8288, "timestamp": "Live"} -> ke 1 clients
```

Pantauan Saham BBCA (Real-time)

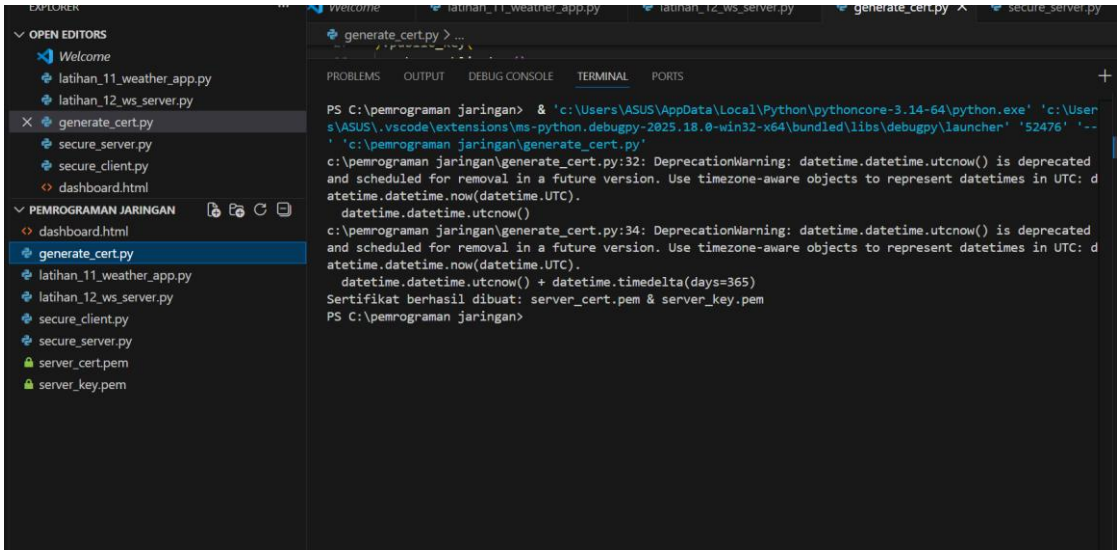
Rp 8066

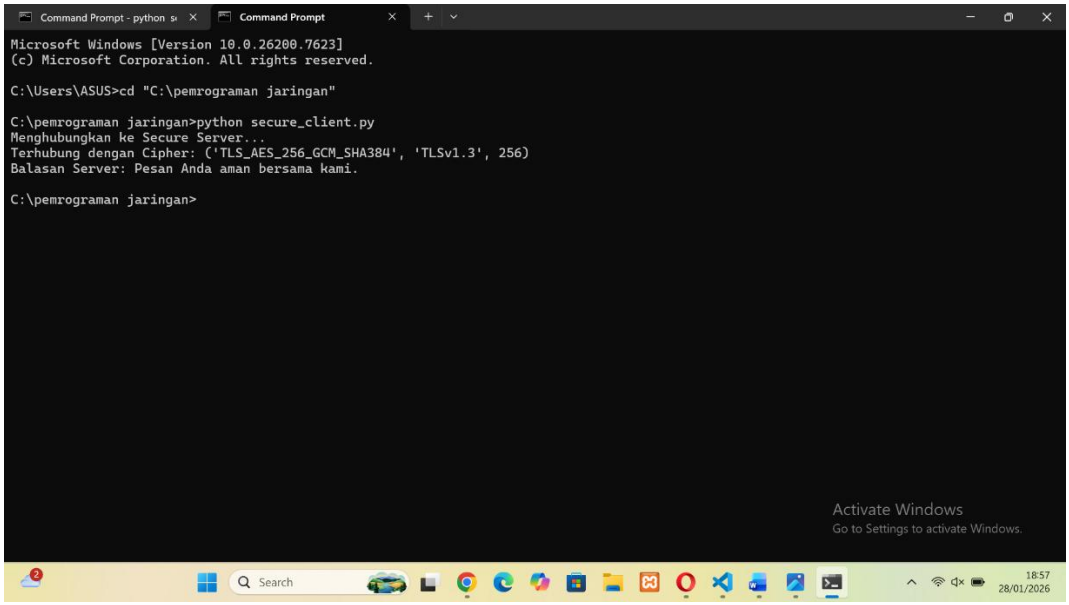
Connected to Server

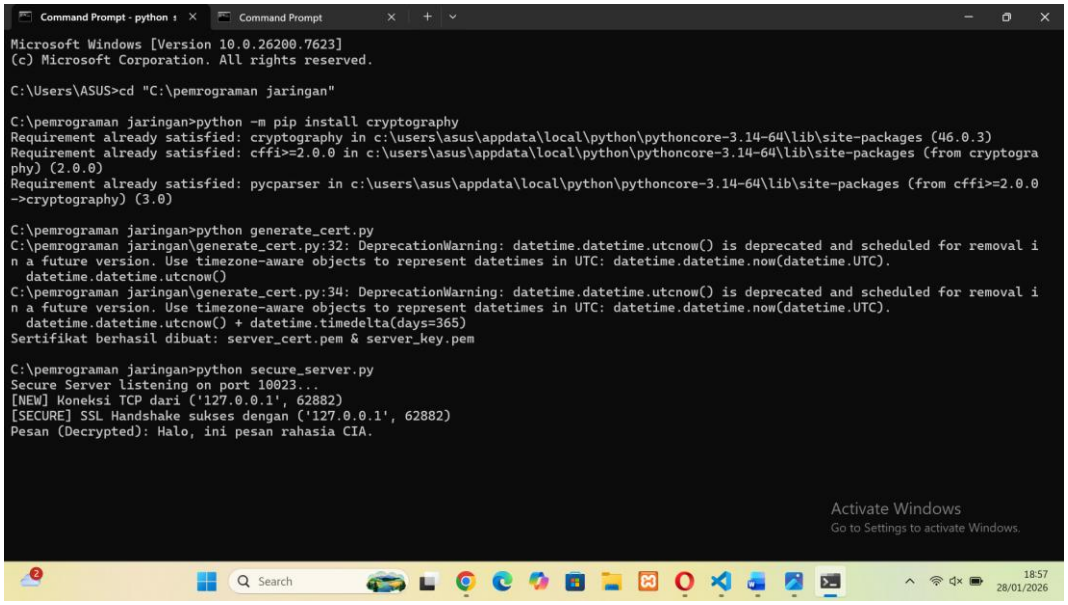
BAB 13 KEAMANAN JARINGAN (NETWORK SECURITY)

Materi menunjukkan proses implementasi komunikasi client–server yang aman menggunakan SSL/TLS dengan Python. Dimulai dari instalasi library cryptography, pembuatan sertifikat digital dan private key melalui file generate_cert.py, lalu menjalankan secure_server.py sebagai server yang mendengarkan koneksi pada port tertentu. Server berhasil melakukan SSL handshake dengan klien, menandakan proses enkripsi TLS berjalan dengan baik menggunakan cipher modern (TLS 1.3). Selanjutnya, klien (secure_client.py) berhasil terhubung dan mengirim pesan yang terenkripsi, kemudian server menampilkan pesan tersebut dalam bentuk terdekripsi. Hal ini membuktikan bahwa data yang dikirimkan melalui jaringan terlindungi dari penyadapan dan komunikasi berlangsung secara aman.

HASIL:



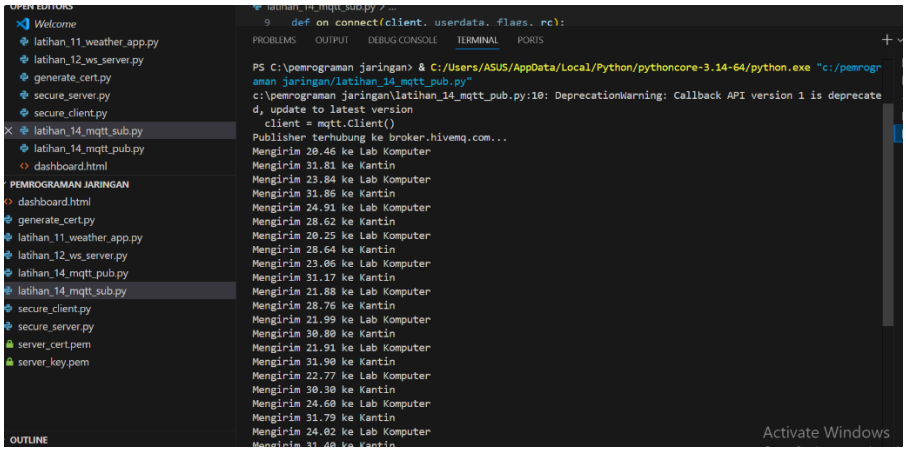

Command Prompt - python
Command Prompt
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ASUS>cd "C:\pemrograman jaringan"
C:\pemrograman jaringan>python secure_client.py
Menghubungkan ke Secure Server...
Terhubung dengan Cipher: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
Balasan Server: Pesan Anda aman bersama kami.
C:\pemrograman jaringan>
Activate Windows
Go to Settings to activate Windows.
Search
18:57
28/01/2026

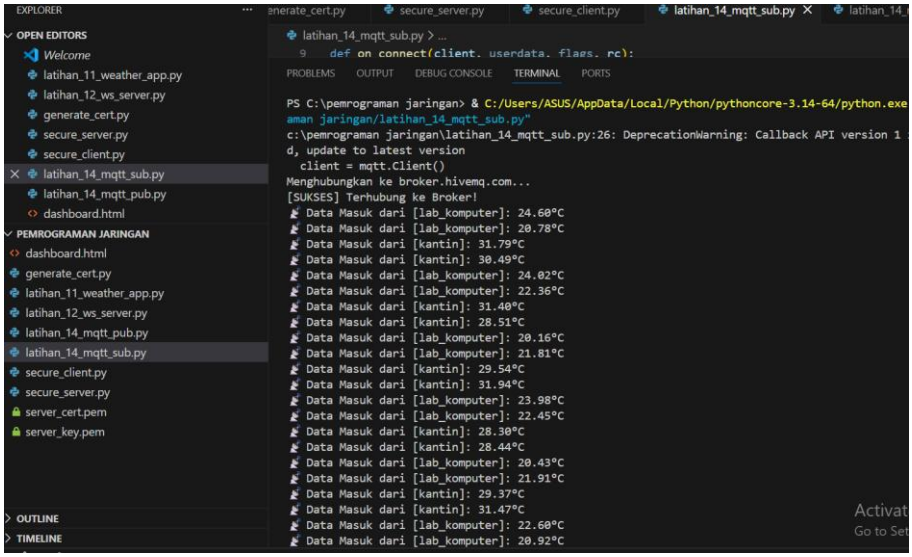

Command Prompt - python
Command Prompt
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ASUS>cd "C:\pemrograman jaringan"
C:\pemrograman jaringan>python -m pip install cryptography
Requirement already satisfied: cryptography in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (46.0.3)
Requirement already satisfied: cffi>=2.0.0 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from cryptography) (2.0.0)
Requirement already satisfied: pycparser in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from cffi>=2.0.0->cryptography) (3.0)
C:\pemrograman jaringan>python generate_cert.py
C:\pemrograman jaringan\generate_cert.py:32: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
datetime.datetime.utcnow()
C:\pemrograman jaringan\generate_cert.py:34: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
datetime.datetime.utcnow() + datetime.timedelta(days=365)
Sertifikat berhasil dibuat: server_cert.pem & server_key.pem
C:\pemrograman jaringan>python secure_server.py
Secure Server listening on port 10023...
[NEW] Koneksi TCP dari ('127.0.0.1', 62882)
[SECURE] SSL Handshake sukses dengan ('127.0.0.1', 62882)
Pesan (Decrypted): Halo, ini pesan rahasia CIA.
Activate Windows
Go to Settings to activate Windows.
Search
18:57
28/01/2026

BAB 14 ARSITEKRUT SISTEM TERDISTRIBUSI & IoT (MQTT)

Materi ini menampilkan penerapan komunikasi data menggunakan protokol MQTT dengan metode publish–subscribe berbasis Python. Program latihan_14_mqtt_pub.py berperan sebagai publisher yang mengirimkan data suhu secara berkala ke broker publik HiveMQ, dengan topik berbeda untuk lokasi seperti lab_komputer dan kantin. Sementara itu, program latihan_14_mqtt_sub.py bertindak sebagai subscriber yang berhasil terhubung ke broker dan menerima data yang dipublikasikan sesuai topik yang di-subscribe, lalu menampilkannya di terminal. Alur ini menunjukkan bagaimana MQTT memungkinkan pertukaran data secara real-time, ringan, dan efisien, sehingga sangat cocok digunakan pada sistem IoT seperti pemantauan suhu di berbagai lokasi.

HASIL:

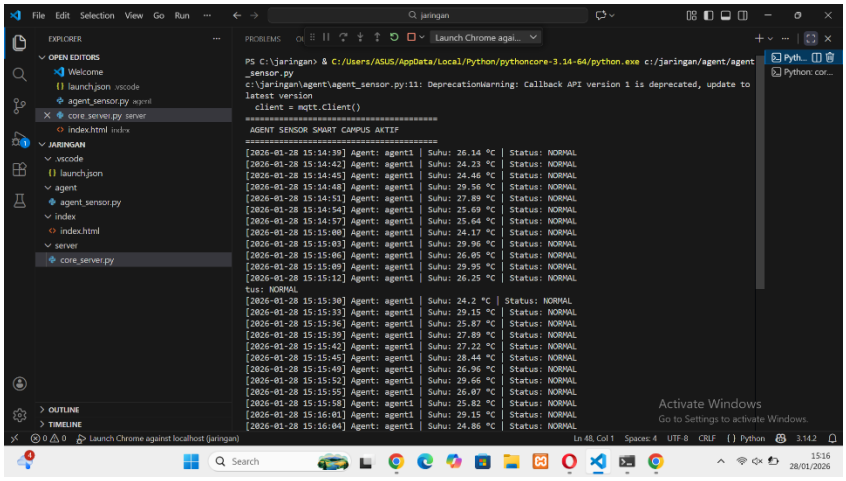


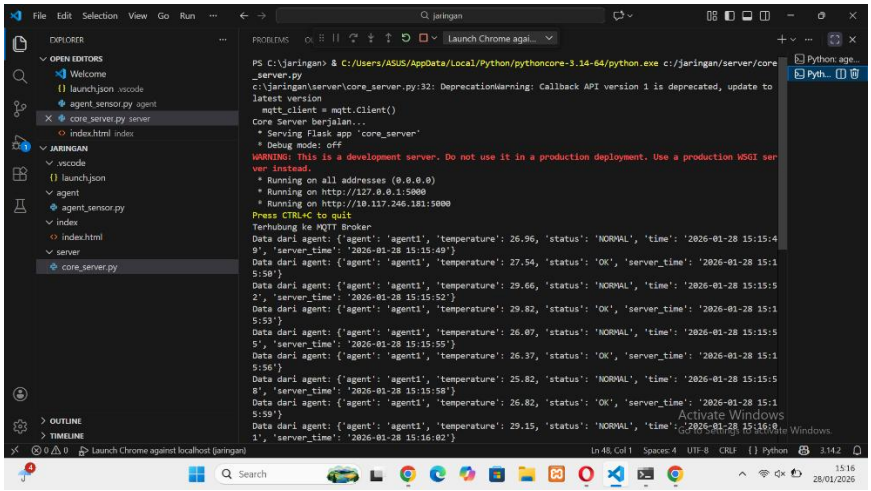


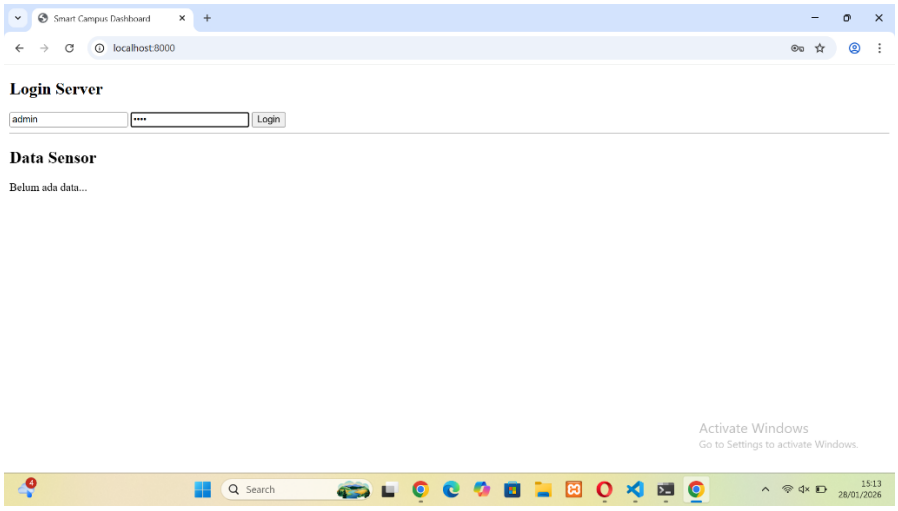
BAB 15 PROYEK AKHIR (CAPSTONE PROJECT)

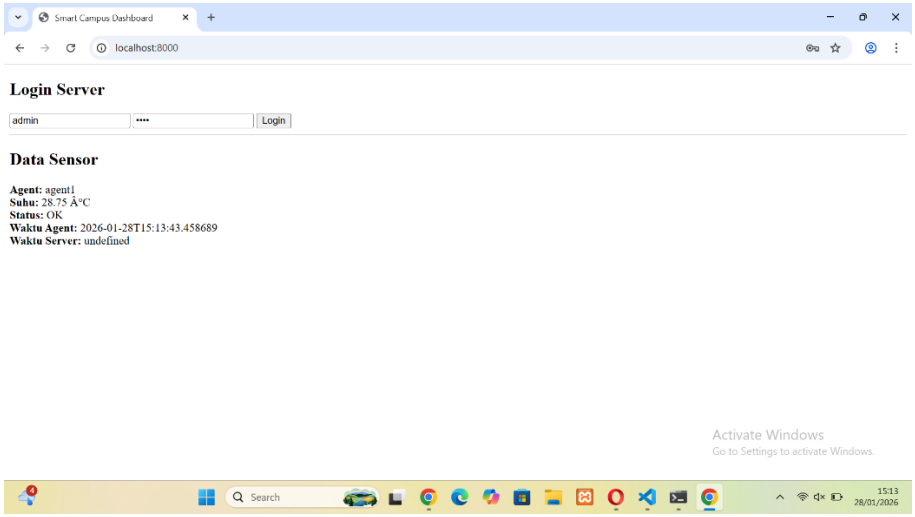
materi sistem yang ditampilkan merupakan implementasi arsitektur IoT berbasis MQTT yang terdiri dari agent/sensor (publisher) broker MQTT dan core server (subscriber + web server). Program agent_sensor.py berperan sebagai agent yang mensimulasikan pembacaan suhu, menentukan status (NORMAL/OK), lalu mengirimkan data secara periodik ke broker MQTT. Selanjutnya core_server.py. berfungsi sebagai subscriber yang menerima data dari agent melalui broker, menambahkan waktu server, dan menampilkannya di terminal sekaligus menjalankan Flask web server agar data dapat diakses melalui browser.

HASIL:






Smart Campus Dashboard
localhost:8000
Login Server
admin
Login
Data Sensor
Belum ada data...
Activate Windows
Go to Settings to activate Windows.
Search
15:13
28/01/2026


Smart Campus Dashboard
localhost:8000
Login Server
admin
Login
Data Sensor
Agent: agent1
Suhu: 28.75 Â°C
Status: OK
Waktu Agent: 2026-01-28T15:13:43.458689
Waktu Server: undefined
Activate Windows
Go to Settings to activate Windows.
Search
15:13
28/01/2026