2. Compare Laptop Configurations Available Online

Feature	**Laptop 1 (Budget) (50K)** **MSI Thin 15** **B12UCX-1695IN** **Laptop**	**Laptop 2 (MidRange) (100K)** **ROG X13 FLOW GV301**	**Laptop 3 (HighPerformance) (150K)** **MSI Crosshair 16** **HX** **D14VFKG-206IN** **Laptop**
CPU Model	**Intel i5-12450H(12thGen)**	**Ryzen 6900HS**	**Intel i7-14700HX(14thGen)**
CPU Cores/Threads	8 Cores 12 Threads	8 Cores 16 Threads	Performance Cores: 8 Cores, 16 Threads, Efficient Cores: 12 Cores, 12 Threads, 1.5 GHz Base, 3.9 GHz Turbo
Base/Turbo Clock Speed	**2.0 GHz Base** **4.4 GHz Turbo**	**3.3 GHz Base** **4.8 GHz Turbo**	Performance Cores:2.1 GHz Base, 5.5 GHz Turbo, Efficient Cores: 1.5 GHz Base, 3.9 GHz Turbo
L3 Cache Size	**12MB**	**16 MB**	**33MB**
RAM Size	**16GB**	**16GB**	**16GB**
RAM Type & Speed	**DDR4(3200 MHz)**	**DDR5(6400Mhz)**	**DDR5(6400MHz)**
Storage Type (HDD/SSD/NVMe)	**SSD**	**NVMe**	**NVMe**
Storage Capacity	**512GB**	**512GB**	**1TB**
Storage Interface (SATA/PCIe)	**M.2(PCIe)**	**NVMe(M.2)**	**M.2(PCIe)**
Price (Approximate)	**52,000**	**89.900**	**134,400**

GPU	RTX2050(4GB)	RTX 3050(4GB)	RTX 4060(8GB)

Answer the Following Questions
1. Which laptop has the best CPU for multitasking and data science workloads? Why?
Best CPU: Laptop 3 – MSI Crosshair 16 HX (i7-14700HX)
CPU Architecture: 20 cores (8 Performance + 12 Efficiency) and 28 threads, which is significantly better for parallel workloads like training ML models or running simultaneous processes (e.g., Jupyter + ETL + DB + browser).

Base/Turbo Clocks: Up to **5.5 GHz** turbo on performance cores.

L3 Cache: 33 MB — larger cache is great for data-heavy operations

2. Which laptop has the best memory configuration? Can it be upgraded?

Memory: Tie between Laptop 2 and Laptop 3

- **Both have:**
    - **16GB DDR5 @ 6400 MHz**, which is significantly faster than **DDR4 @ 3200 MHz** (Laptop 1).
    - DDR5 improves bandwidth and latency — crucial for large dataset operations and GPU memory paging.
- Laptop 3 has upgradable memory upto 32GB

3. Which storage type is the fastest among the three?

Fastest Storage: Laptop 2 and Laptop 3 – Both use **NVMe (PCIe)** SSDs.

Explanation:

- NVMe (Non-Volatile Memory Express) over PCIe offers **much higher throughput** and **lower latency** than standard SATA SSDs or M.2 SATA.
- All three laptops use M.2 form factor, but **Laptop 2 and 3 explicitly use NVMe**, making them ideal for fast data access, large file loads (e.g., image/video datasets), and fast boot.

4. If you were to buy a laptop for data science, which one would you choose and why?

Recommended: Laptop 3 – MSI Crosshair 16 HX D14VFKG-206IN

Why:

- **CPU:** The i7-14700HX offers exceptional multicore performance with modern hybrid architecture (P+E cores).
- **High-speed DDR5 RAM** and upgrade potential.
- **1TB NVMe SSD** provides ample and fast storage.
- **Powerful GPU (RTX 4060 8GB)**: Useful for training models (CUDA), deep learning, and GPU-accelerated libraries.

3. (Group Assignment) Compare Features:

Pre-requisite:
• The participants have completed the Assignment #2 individually.

Task:
• Sit together and compare the 3 Laptops you just selected.
• Argument & justify why your chosen Laptop is better.

BUDGET:

CPU:

The MSI Thin 15 uses the Intel Core i5-12450H, which is a more powerful processor designed for gaming and high-performance tasks. It has a higher TDP (45W) and better multi-core performance, which is beneficial for data science tasks that require heavy computation.

The Lenovo V14 uses the Intel Core i7-1255U, which is a low-power, efficient processor designed for thin and light laptops. While it is more power-efficient, it may not match the raw performance of the i5-12450H in demanding tasks.

STORAGE:

Both laptops have 512 GB SSDs, which provide fast read and write speeds, essential for data science workflows. The MSI Thin 15 has a PCIe Gen 4 SSD, which offers faster data transfer rates compared to the standard SSD in the Lenovo V14.

GRAPHICS:

The MSI Thin 15 has a dedicated NVIDIA GeForce RTX 2050 GPU, which can significantly speed up tasks that benefit from GPU acceleration, such as machine learning, deep learning, and data visualization.

The Lenovo V14 has Intel Iris Xe integrated graphics, which are suitable for basic tasks but may not provide the same level of performance for GPU-intensive tasks.

MIDRANGE:

CPU:

The i7-13620H offers more physical and logical cores than the Ryzen 9 6900HS, which is beneficial for parallel data processing and machine learning workloads.

GPU:

The RTX 4060 (8GB VRAM) in the TUF F15 is significantly more powerful than the RTX 3050Ti (4GB VRAM) in the Flow X13. For deep learning, GPU memory and compute power are critical, and the RTX 4060 nearly doubles the performance and VRAM.

HIGH-END:

CPU:

The HP Omen's i9-13900HX has more total cores (24 vs 20), which can be advantageous for Data science workloads.

UPGRADABILITY:

MSI supports up to 96GB RAM, which is a major advantage for handling large datasets or running memory-intensive workloads.

QUALITY OF LIFE:

MSI offers a higher resolution (2560x1600 vs 1920x1080), higher refresh rate (240Hz vs 165Hz), and a better 16:10 aspect ratio.

• Decide, Agree and Come-up with one Laptop for each Category
• Present the arguments/justifications for your agreement on each Category

BUDGET:Srijan
MIDDLE:Chattresh
HIGH END:Srijan

FINAL TABLE

Feature	**Laptop 1 (Budget) (50K) (Srijan) MSI Thin 15 B12UCX-1695IN Laptop**	**Laptop 2 (MidRange) (100K) (Chhatresh) ASUS TUF Gaming F15**	**Laptop 3 (HighPerformance) (150K) (Srijan) MSI Crosshair 16 HX D14VFKG-206IN Laptop**
CPU Model	**Intel i5-12450H(12thGen)**	**Intel Core i7 13th Gen 13620H**	**Intel i7-14700HX(14thGen)**
CPU Cores/Threads	8 Cores 12 Threads	10 Cores 16 Threads	Performance Cores: 8 Cores, 16 Threads, Efficient Cores: 12 Cores, 12 Threads, 1.5 GHz Base, 3.9 GHz Turbo
Base/Turbo Clock Speed	2.0 GHz Base 4.4 GHz Turbo	Performance Cores: 2.4 GHz	Performance Cores:2.1 GHz

		Base, 4.9 GHz Turbo. Efficient Cores: 1.8 GHz Base, 3.6 GHz Turbo	Base, 5.5 GHz Turbo, Efficient Cores: 1.5 GHz Base, 3.9 GHz Turbo
L3 Cache Size	**12MB**	**24 MB**	**33MB**
RAM Size	**16GB**	**16GB**	**16GB**
RAM Type & Speed	**DDR4(3200 MHz)**	**DDR5(4800Mhz)**	**DDR5(6400MHz)**
Storage Type (HDD/SSD/NVMe)	**SSD**	**NVMe**	**NVMe**
Storage Capacity	**512GB**	**512GB**	**1TB**
Storage Interface (SATA/PCIe)	**M.2(PCIe)**	**NVMe(M.2)**	**M.2(PCIe)**
Price (Approximate)	**52,000**	**99.900**	**134,400**
GPU	**RTX2050(4GB)**	**RTX 4060(8GB)**	**RTX 4060(8GB)**

4. Real-World Network Analysis

Steps:

1. Install a Packet Sniffing Tool

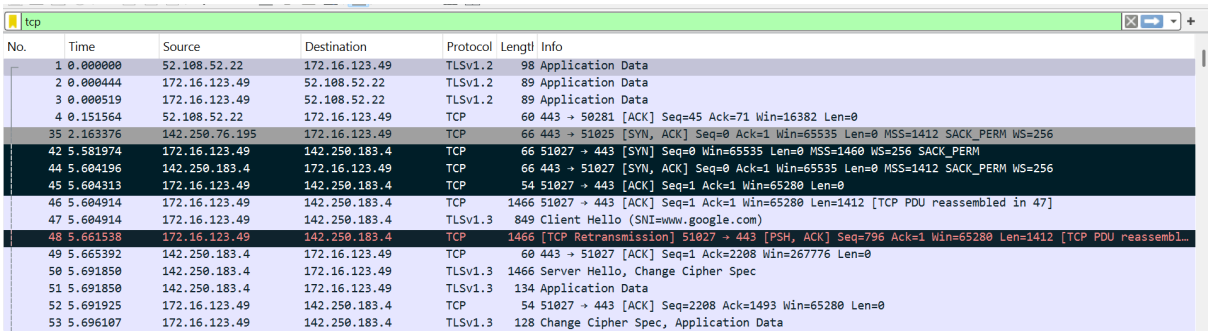
a. Use Wireshark (Windows/Linux/macOS) or tcpdump (Linux/macOS) to capture network traffic.

2. Capture TCP/IP Traffic

a. Open Wireshark and start capturing packets.

b. Visit any website (e.g., http://example.com).

c. Stop capturing packets and filter only TCP packets.



tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	52.108.52.22	172.16.123.49	TLSv1.2	98	Application Data
2	0.000444	172.16.123.49	52.108.52.22	TLSv1.2	89	Application Data
3	0.000519	172.16.123.49	52.108.52.22	TLSv1.2	89	Application Data
4	0.151564	52.108.52.22	172.16.123.49	TCP	60	443 → 50281 [ACK] Seq=45 Ack=71 Win=16382 Len=0
35	2.163376	142.250.76.195	172.16.123.49	TCP	66	443 → 51025 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
42	5.581974	172.16.123.49	142.250.183.4	TCP	66	51027 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
44	5.604196	142.250.183.4	172.16.123.49	TCP	66	443 → 51027 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
45	5.604313	172.16.123.49	142.250.183.4	TCP	54	51027 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
46	5.604914	172.16.123.49	142.250.183.4	TCP	1466	51027 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1412 [TCP PDU reassembled in 47]
47	5.604914	172.16.123.49	142.250.183.4	TLSv1.3	849	Client Hello (SNI=www.google.com)
48	5.661538	172.16.123.49	142.250.183.4	TCP	1466	[TCP Retransmission] 51027 → 443 [PSH, ACK] Seq=796 Ack=1 Win=65280 Len=1412 [TCP PDU reassembl…
49	5.665392	142.250.183.4	172.16.123.49	TCP	60	443 → 51027 [ACK] Seq=1 Ack=2208 Win=267776 Len=0
50	5.691850	142.250.183.4	172.16.123.49	TLSv1.3	1466	Server Hello, Change Cipher Spec
51	5.691850	142.250.183.4	172.16.123.49	TLSv1.3	134	Application Data
52	5.691925	172.16.123.49	142.250.183.4	TCP	54	51027 → 443 [ACK] Seq=2208 Ack=1493 Win=65280 Len=0
53	5.696107	172.16.123.49	142.250.183.4	TLSv1.3	128	Change Cipher Spec, Application Data

3. Analyze the TCP Connection

a. Identify the three-way handshake (SYN, SYN-ACK, ACK).

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000444	172.16.123.49	52.108.52.22	TLSv1.2	89	Application Data
3	0.000519	172.16.123.49	52.108.52.22	TLSv1.2	89	Application Data
4	0.151564	52.108.52.22	172.16.123.49	TCP	60	443 → 50281 [ACK] Seq=45 Ack=71 Win=16382 Len=0
35	2.163376	142.250.76.195	172.16.123.49	TCP	66	443 → 51025 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
42	5.581974	172.16.123.49	142.250.183.4	TCP	66	51027 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
44	5.604196	142.250.183.4	172.16.123.49	TCP	66	443 → 51027 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
45	5.604313	172.16.123.49	142.250.183.4	TCP	54	51027 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
46	5.604914	172.16.123.49	142.250.183.4	TCP	1466	51027 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1412 [TCP PDU reassembled in 47]
47	5.604914	172.16.123.49	142.250.183.4	TLSv1.3	849	Client Hello (SNI=www.google.com)
48	5.661538	172.16.123.49	142.250.183.4	TCP	1466	[TCP Retransmission] 51027 → 443 [PSH, ACK] Seq=796 Ack=1 Win=65280 Len=1412 [TCP PDU reassembl…
49	5.665392	142.250.183.4	172.16.123.49	TCP	60	443 → 51027 [ACK] Seq=1 Ack=2208 Win=267776 Len=0
50	5.691850	142.250.183.4	172.16.123.49	TLSv1.3	1466	Server Hello, Change Cipher Spec
51	5.691850	142.250.183.4	172.16.123.49	TLSv1.3	134	Application Data
52	5.691925	172.16.123.49	142.250.183.4	TCP	54	51027 → 443 [ACK] Seq=2208 Ack=1493 Win=65280 Len=0
53	5.696107	172.16.123.49	142.250.183.4	TLSv1.3	128	Change Cipher Spec, Application Data
54	5.718317	142.250.183.4	172.16.123.49	TCP	60	443 → 51027 [ACK] Seq=1493 Ack=2282 Win=267776 Len=0

Filter for SYN packets only

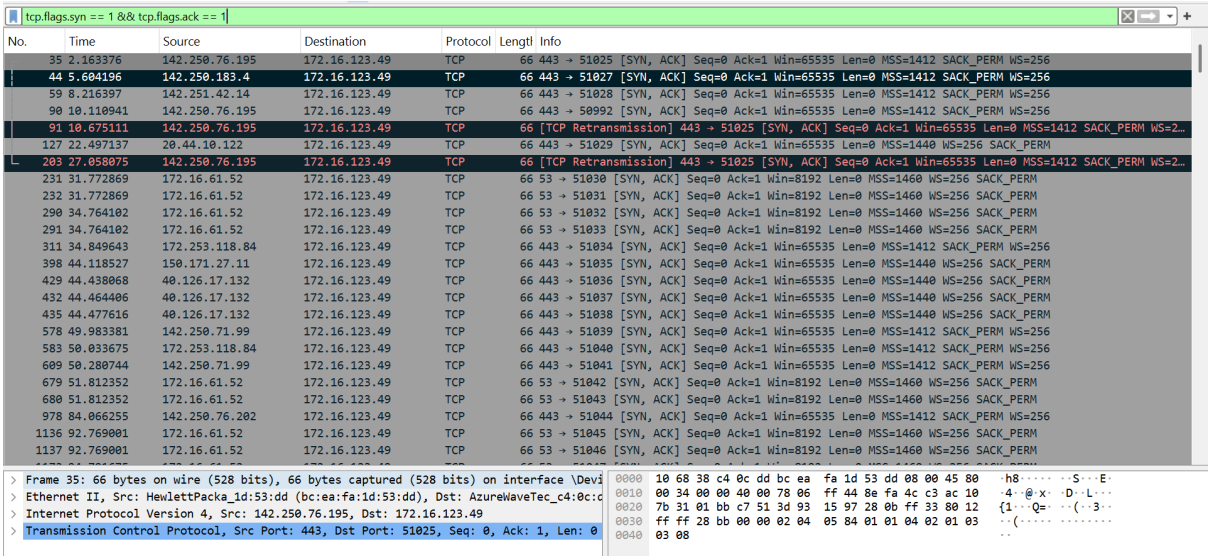
tcp.flags.syn == 1 && tcp.flags.ack == 0

No.	Time	Source	Destination	Protocol	Length	Info
42	5.581974	172.16.123.49	142.250.183.4	TCP	66	51027 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
58	8.193257	172.16.123.49	142.251.42.14	TCP	66	51028 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
126	22.249717	172.16.123.49	20.44.10.122	TCP	66	51029 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
226	31.768860	172.16.123.49	172.16.61.52	TCP	66	51030 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
227	31.769065	172.16.123.49	172.16.61.52	TCP	66	51031 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
288	34.760105	172.16.123.49	172.16.61.52	TCP	66	51032 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
289	34.760315	172.16.123.49	172.16.61.52	TCP	66	51033 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
303	34.769170	172.16.123.49	172.253.118.84	TCP	66	51034 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
397	44.080903	172.16.123.49	150.171.27.11	TCP	66	51035 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
426	44.399062	172.16.123.49	40.126.17.132	TCP	66	51036 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
427	44.425586	172.16.123.49	40.126.17.132	TCP	66	51037 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
428	44.437430	172.16.123.49	40.126.17.132	TCP	66	51038 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
573	49.954086	172.16.123.49	142.250.71.99	TCP	66	51039 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
574	49.954328	172.16.123.49	172.253.118.84	TCP	66	51040 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
608	50.254252	172.16.123.49	142.250.71.99	TCP	66	51041 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
675	51.808084	172.16.123.49	172.16.61.52	TCP	66	51042 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
676	51.808330	172.16.123.49	172.16.61.52	TCP	66	51043 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
977	84.042614	172.16.123.49	142.250.76.202	TCP	66	51044 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1127	92.764652	172.16.123.49	172.16.61.52	TCP	66	51045 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1128	92.764818	172.16.123.49	172.16.61.52	TCP	66	51046 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1171	94.787415	172.16.123.49	172.16.61.52	TCP	66	51047 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1172	94.787685	172.16.123.49	172.16.61.52	TCP	66	51048 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1189	94.802604	172.16.123.49	3.165.190.15	TCP	66	51049 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
1256	112.803952	172.16.123.49	172.16.61.52	TCP	66	51050 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM

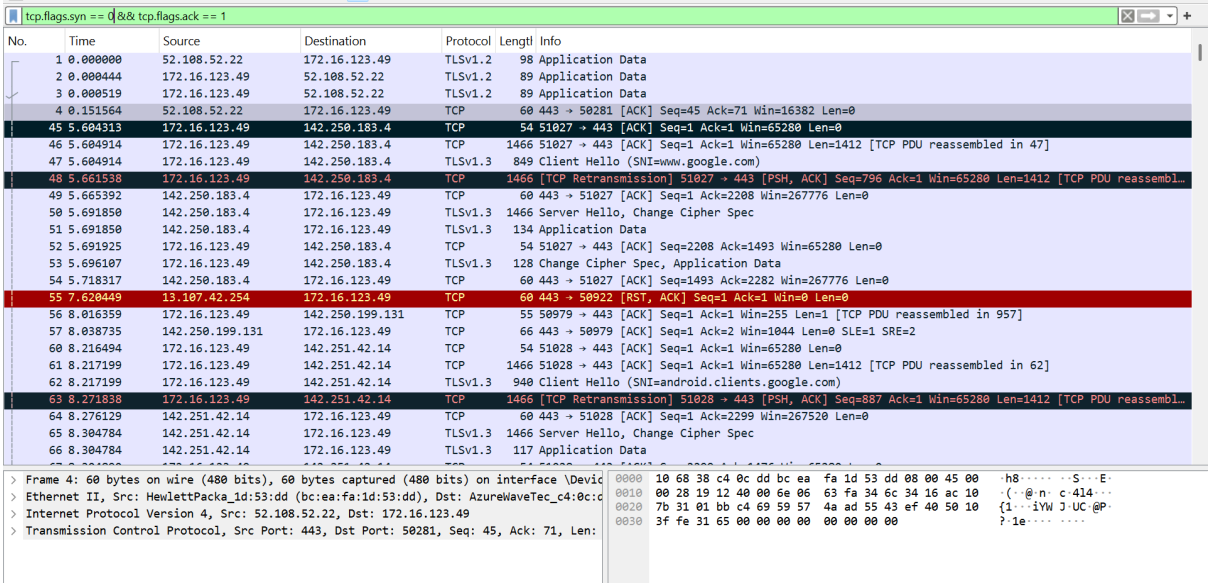
```
> Frame 42: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Devi
> Ethernet II, Src: AzureWaveTec_c4:0c:dd (10:68:38:c4:0c:dd), Dst: HewlettPacka_1d:53:d
> Internet Protocol Version 4, Src: 172.16.123.49, Dst: 142.250.183.4
> Transmission Control Protocol, Src Port: 51027, Dst Port: 443, Seq: 0, Len: 0

0000  bc ea fa 1d 53 dd 10 68  38 c4 0c dd 08 00 45 00   ····S··h 8·····E·
0010  00 34 07 68 40 00 80 06  86 1b ac 10 7b 31 8e fa   ·4·h@··· ····{1··
0020  b7 04 c7 53 01 bb a7 f5  b2 6f 00 00 00 00 80 02   ···S···· ·o******
0030  ff ff de 5b 00 00 02 04  05 b4 01 03 03 08 01 01   ···[···· ********
0040  04 02                                              ··
```

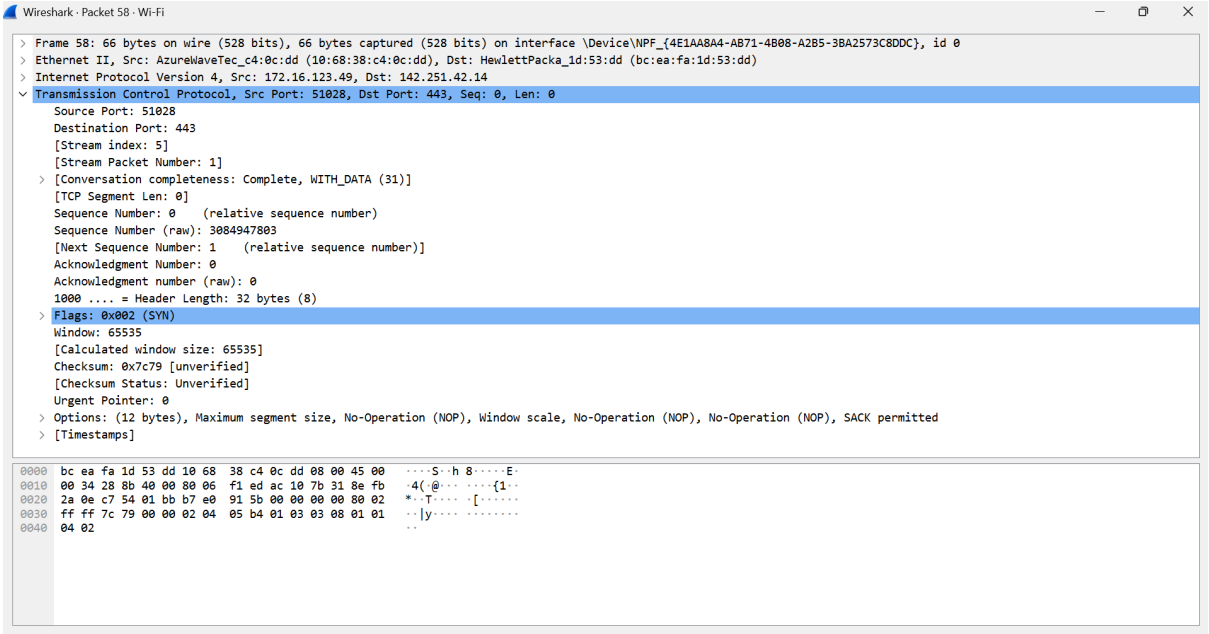
FILTER for SYN,ACK



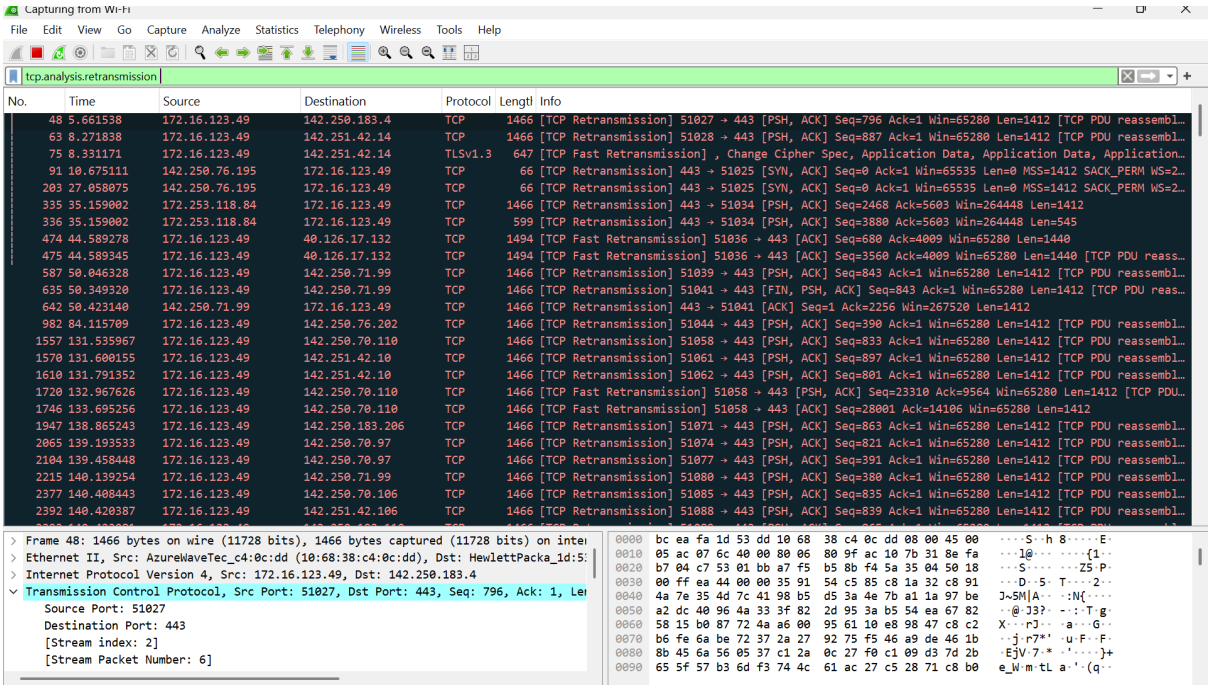
FILTER for ACK

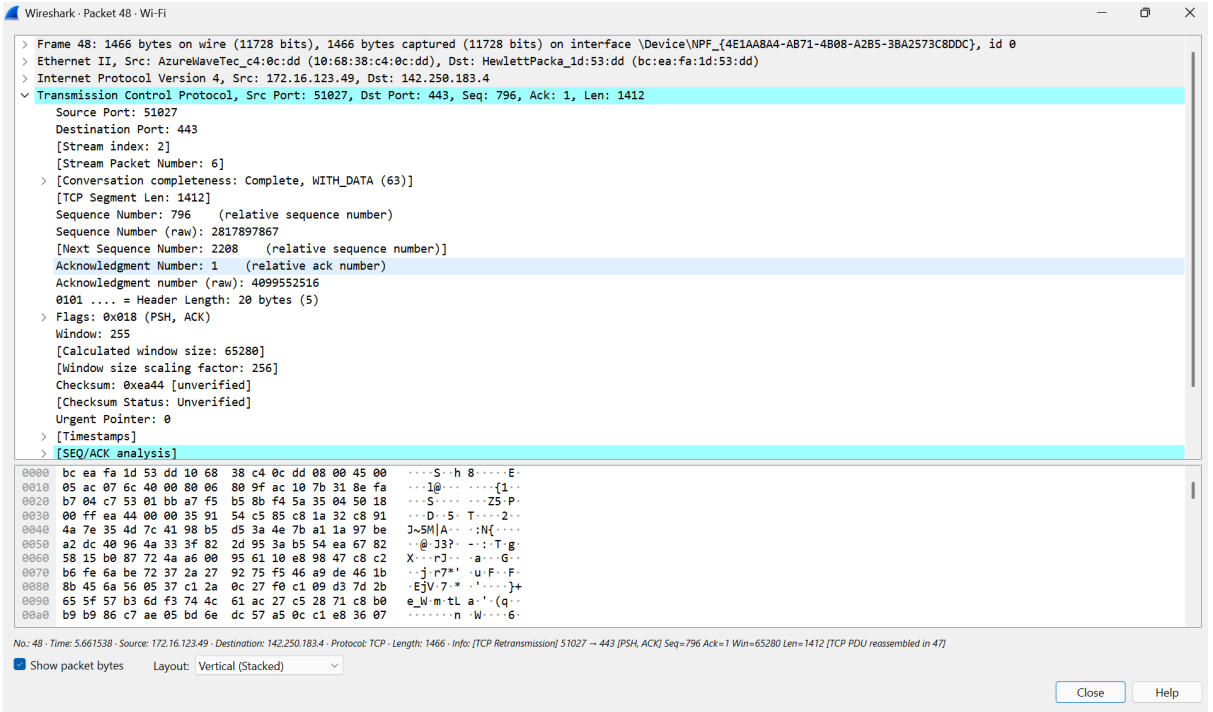


b. Observe the packet sequence numbers.



c. Find a TCP retransmission or dropped packet.





4. Answer the following questions:

a. What are the source and destination IP addresses of the captured packets?

```
Transmission Control Protocol, Src Port: 443, Dst Port: 50281, Seq: 45, Ack: 71, Len: 0
    Source Port: 443
    Destination Port: 50281
```

b. What is the port number used for the connection?

```
Internet Protocol Version 4, Src: 52.108.52.22, Dst: 172.16.123.49
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x1912 (6418)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 110
    Protocol: TCP (6)
    Header Checksum: 0x63fa [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 52.108.52.22
    Destination Address: 172.16.123.49
    [Stream index: 0]
Transmission Control Protocol, Src Port: 443, Dst Port: 50281, Seq: 45, Ack: 71, Len: 0
    Source Port: 443
    Destination Port: 50281
```

c. What happens when a packet is lost? How does TCP handle it?

TCP detects loss (due to missing ACK or timeout).

It **retransmits** the lost packet automatically.

Mechanisms used:

- **Timeouts**
- **Duplicate ACKs**
- **Fast Retransmit**

2580	140.708796	172.16.123.49	142.250.70.106	TCP	1466	[TCP Retransmission] 51097 → 443 [PSH, ACK] Seq=931 Ack=1 Win=65280 Len=1412 [TCP PDU reassembl…
2632	140.786632	142.250.70.106	172.16.123.49	TLSv1.3	1466	[TCP Fast Retransmission] , Server Hello, Change Cipher Spec
2672	140.909862	172.16.123.49	142.250.70.106	TCP	1466	[TCP Retransmission] 51098 → 443 [PSH, ACK] Seq=867 Ack=1 Win=65280 Len=1412 [TCP PDU reassembl…
2716	140.965646	172.16.123.49	142.250.70.106	TCP	1466	[TCP Spurious Retransmission] 51098 → 443 [ACK] Seq=7215 Ack=2555 Win=64256 Len=1412

5. HTTP Request using TELNET

Objective:
Understand how HTTP works at a low level by manually sending an HTTP request.
Steps:
1. Open Command Line (Windows/Linux/macOS)

```
C:\Windows\System32>dism /online /Enable-Feature /FeatureName:TelnetClient

Deployment Image Servicing and Management tool
Version: 10.0.26100.1150

Image Version: 10.0.26100.3775

Enabling feature(s)
[==========================100.0%==========================]
The operation completed successfully.

C:\Windows\System32>_
```

a. Run the command: telnet example.com 80

```
Administrator: Command Prompt

HTTP/1.0 400 Bad Request
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 312
Expires: Wed, 30 Apr 2025 10:05:29 GMT
Date: Wed, 30 Apr 2025 10:05:29 GMT
Connection: close

<HTML><HEAD>
            <TITLE>Invalid URL</TITLE>
                                    </HEAD><BODY>
                                                 <H1>Invalid URL</H1>
                                                                     The requested URL "&#91;no&#32;URL&#93;", is invalid.<p>
      Reference&#32;&#35;9&#46;c6380760&#46;1746007529&#46;20f79d27
                                                                  <P>https&#58;&#47;&#47;errors&#46;edgesuite&#46;net&#47;9&#46;c6380760&#46;1746007529&#46;20f79d27</P>
                                                 </BODY></HTML>


Connection to host lost.

C:\Windows\System32>
```

b. If telnet is not installed, install it first.
2. Send an HTTP Request Manually
a. Type the following HTTP request and press Enter twice:
i. GET / HTTP/1.1

ii. Host: example.com

```
      HTTP/1.1 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=2505
Date: Wed, 30 Apr 2025 10:08:31 GMT
Content-Length: 1256
Connection: keep-alive

<!doctype html>
               <html>
                     <head>
                                <title>Example Domain</title>

                                                                                <meta charset="utf-8" />
       <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
                                                                                                <meta name="viewport" content="width=device-width, initial-scale=1" />
                                                                                        <style type="text/css">
             body {
                                background-color: #f0f0f2;
                                                                                 margin: 0;
                                                                                                 padding: 0;
              font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

                                                                                        }
                                                                                             div {
       width: 600px;
                                 margin: 5em auto;
                                                                        padding: 2em;
                                                                                                 background-color: #fdfdff;
                          border-radius: 0.5em;
                                                                        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
                  }
                       a:link, a:visited {
                                                                      color: #38488f;
                                                                                                 text-decoration: none;
               }
                    @media (max-width: 700px) {
                                                                                 div {
                                                                                                 margin: 0 auto;
               width: auto;
                                          }
                                               }
                                                    </style>
                                                                                    </head>

                                                                                          <body>
                                                                                                <div>
                                                                                                         <h1>Example Domain</h1>
                       <p>This domain is for use in illustrative examples in documents. You may use this
                        domain in literature without prior coordination or asking for permission.</p>
                      <p><a href="https://www.iana.org/domains/example">More information...</a></p>
             </div>
                   </body>
                          </html>
                                 xam
```

b. Observe the HTTP response headers and content.

3. Answer the following questions:

a. What is the HTTP status code returned? What does it mean?

HTTP/1.1 200 OK

- The server successfully processed the HTTP request and returned the requested resource (in this case, the HTML page of example.com).

b. What are the HTTP headers received?

```
      HTTP/1.1 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=2505
Date: Wed, 30 Apr 2025 10:08:31 GMT
Content-Length: 1256
Connection: keep-alive
```

Content-Type: Specifies that the content is HTML.

ETag: A unique identifier for the version of the resource.

Last-Modified: When the content was last changed.

Cache-Control: How long (in seconds) clients can cache the response.

Date: The date and time the response was sent.

Content-Length: The size of the response body in bytes.

Connection: keep-alive: Keeps the connection open for further requests.

c. How does this method compare to using a browser?

c. How does this method compare to using a browser?

Feature	Telnet	Browser
User Interface	Command-line only, no rendering	Graphical, fully rendered web pages
Manual Request	You must type raw HTTP requests	Browser builds and sends them automatically
Headers/Body View	Headers and HTML visible as raw text	Headers are hidden; body is rendered as a webpage
HTTPS Support	Not supported via Telnet (use `openssl` instead)	Fully supports HTTPS, TLS, certificates, etc.
Purpose	Debugging, learning protocol behavior	User experience, browsing the web

6. Compare HTTP vs. HTTPS Request

Objective:

Understand how HTTPS encrypts communication compared to HTTP.

Steps:

1. Use OpenSSL to Connect to an HTTPS Server

a. Run the following command:

openssl s_client -connect www.google.com:443

```
C:\Windows\System32>openssl s_client -connect www.google.com:443
Connecting to 142.250.192.100
CONNECTED(00000200)
depth=2 C=US, O=Google Trust Services LLC, CN=GTS Root R4
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=1 C=US, O=Google Trust Services, CN=WE2
verify return:1
depth=0 CN=www.google.com
verify return:1
---
Certificate chain
 0 s:CN=www.google.com
   i:C=US, O=Google Trust Services, CN=WE2
   a:PKEY: EC, (prime256v1); sigalg: ecdsa-with-SHA256
   v:NotBefore: Mar 31 08:56:27 2025 GMT; NotAfter: Jun 23 08:56:26 2025 GMT
 1 s:C=US, O=Google Trust Services, CN=WE2
   i:C=US, O=Google Trust Services LLC, CN=GTS Root R4
   a:PKEY: EC, (prime256v1); sigalg: ecdsa-with-SHA384
   v:NotBefore: Dec 13 09:00:00 2023 GMT; NotAfter: Feb 20 14:00:00 2029 GMT
 2 s:C=US, O=Google Trust Services LLC, CN=GTS Root R4
   i:C=BE, O=GlobalSign nv-sa, OU=Root CA, CN=GlobalSign Root CA
   a:PKEY: EC, (secp384r1); sigalg: sha256WithRSAEncryption
   v:NotBefore: Nov 15 03:43:21 2023 GMT; NotAfter: Jan 28 00:00:42 2028 GMT
---
```

```
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDlTCCAzygAwIBAgIQAnGD1KgleQcKPY9gnifN5TAKBggqhkjOPQQDAjA7MQsw
CQYDVQQGEwJVUzEeMBwGA1UEChMVR29vZ2xlIFRydXN0IFNlcnZpY2VzMQwwCgYD
VQQDEwNXRTIwHhcNMjUwMzMxMDg1NjI3WhcNMjUwNjIzMDg1NjI2WjAZMRcwFQYD
VQQDEw53d3cuZ29vZ2xlLmNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABBHj
B5fkcQxfYTjDVmvM4Jpr4RhjL+mH4yyk8lTvodX9BsFwTMwbaZ3AH7rPf9Pv6s3v
M9CBGWcwDkVZbDXS4NSjggJCMIICPjAOBgNVHQ8BAf8EBAMCB4AwEwYDVR0lBAww
CgYIKwYBBQUHAwEwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQU3KP2XkzycUZRpFCY
kYv8IO8pcEUwHwYDVR0jBBgwFoAUdb7Ed66J9kQ3fc+xaB8dGuvcNFkwWAYIKwYB
BQUHAQEETDBKMCEGCCsGAQUFBzABhhVodHRwOi8vby5wa2kuZ29vZy93ZTIwJQYI
KwYBBQUHMAKGGWh0dHA6Ly9pLnBraS5nb29nL3dlMi5jcnQwGQYDVR0RBBIwEIIO
d3d3Lmdvb2dsZS5jb20wEwYDVR0gBAwwCjAIBgZngQwBAgEwNgYDVR0fBC8wLTAr
oCmgJ4YlaHR0cDovL2MucGtpLmdvb2cvd2UyL3h1enQzUFU5Rl93LmNybDCCAQUG
CisGAQQB1nkCBAIEgfYEgfMA8QB2AM8RVu7VLnyv84db2Wkum+kacWdKsBfsrAHS
W3fOzDsIAAABleuhkB4AAAQDAEcwRQIhAMf3jwButHnFnHo1aUx9e+EbNsgP2WzC
YyhM3o9H13J0AiAgLZv1kFt0po07tpllOvk/LAzx8RtO9l3IDHxs0q7AXQB3AKLj
CuRF772tm3447Udnd1PXgluElNcrXhssxLlQpEfnAAABleuhk+MAAAQDAEgwRgIh
AIu72/WhD+8tyuBXYyJ7sqUhTXuurs4MLJIqDcT2Y6USAiEA0Dmz78Ap+gPbnUhJ
+UifxR8jQ2tBX7J27wfH6sbfl3YwCgYIKoZIzj0EAwIDRwAwRAIgZOShqs9njXez
5Wen/buqZWKZsXw57BPidSojHUJ5IhoCICRd5uCEGApzR5sG506AnVswrPKdMtiX
H7RqdOGixbXu
-----END CERTIFICATE-----
subject=CN=www.google.com
issuer=C=US, O=Google Trust Services, CN=WE2
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: ecdsa_secp256r1_sha256
Negotiated TLS1.3 group: X25519MLKEM768
---
SSL handshake has read 3892 bytes and written 1627 bytes
Verification error: unable to get local issuer certificate
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Protocol: TLSv1.3
Server public key is 256 bit
This TLS version forbids renegotiation.
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 20 (unable to get local issuer certificate)
---
```

b. Observe the TLS handshake and certificate details.

2. Answer the following questions:
a. What TLS version is being used?

Protocol: TLSv1.3

b. What is the certificate authority (CA) issuing the SSL certificate?

Google Trust Services LLC

c. What encryption algorithm is used for securing communication?

TLS_AES_256_GCM_SHA384

d. How is HTTPS different from HTTP in terms of security?

Feature	HTTP	HTTPS (HTTP over TLS)
Encryption	No	Yes (TLS/SSL encryption)
Data Privacy	Plaintext (can be sniffed)	Encrypted end-to-end
Authentication	No identity verification	Verified via CA certificates
Data Integrity	Can be modified	Protected from tampering
Port	80	443

7. API Communication using HTTP Methods

Objective:

Understand different HTTP methods (GET, POST, PUT, DELETE) using a REST API.

Steps:

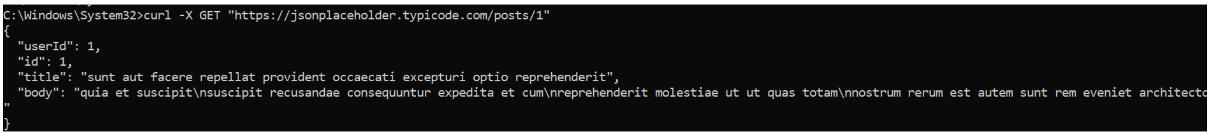
1. Use an API Testing Tool (cURL)

a. Try the following HTTP requests on a public API (e.g., JSONPlaceholder).

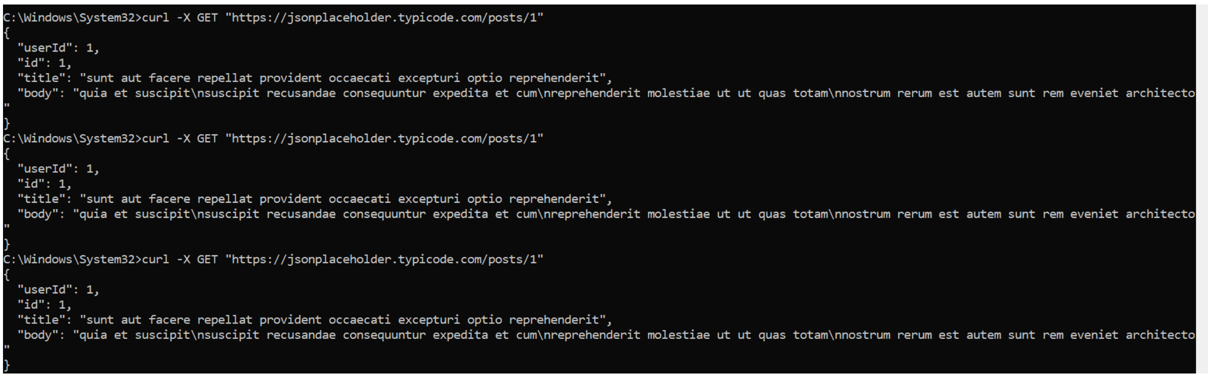
2. Make API Calls:

a. GET Request:

```
curl -X GET
"https://jsonplaceholder.typicode.com/posts/1"
```



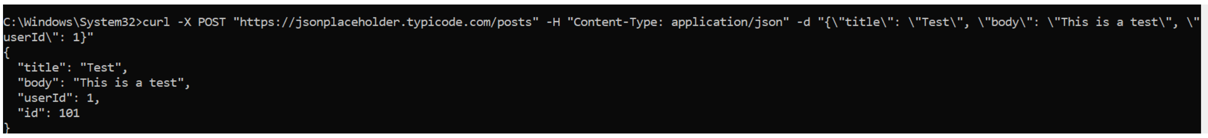
```
C:\Windows\System32>curl -X GET "https://jsonplaceholder.typicode.com/posts/1"
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto
"
}
```



```
C:\Windows\System32>curl -X GET "https://jsonplaceholder.typicode.com/posts/1"
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto
"
}
C:\Windows\System32>curl -X GET "https://jsonplaceholder.typicode.com/posts/1"
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto
"
}
C:\Windows\System32>curl -X GET "https://jsonplaceholder.typicode.com/posts/1"
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto
"
}
```

b. POST Request (Create New Data):

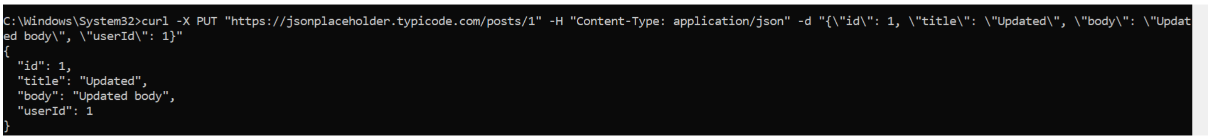
```
curl -X POST "https://jsonplaceholder.typicode.com/posts"
-H "Content-Type: application/json" -d '{"title": "Test",
"body": "This is a test", "userId": 1}'
```



```
C:\Windows\System32>curl -X POST "https://jsonplaceholder.typicode.com/posts" -H "Content-Type: application/json" -d "{\"title\": \"Test\", \"body\": \"This is a test\", \"
userId\": 1}"
{
  "title": "Test",
  "body": "This is a test",
  "userId": 1,
  "id": 101
}
```

c. PUT Request (Update Data):

```
curl -X PUT
"https://jsonplaceholder.typicode.com/posts/1" -H
"Content-Type: application/json" -d '{"id":1, "title":
"Updated", "body": "Updated body", "userId": 1}'
```



```
C:\Windows\System32>curl -X PUT "https://jsonplaceholder.typicode.com/posts/1" -H "Content-Type: application/json" -d "{\"id\": 1, \"title\": \"Updated\", \"body\": \"Updat
ed body\", \"userId\": 1}"
{
  "id": 1,
  "title": "Updated",
  "body": "Updated body",
  "userId": 1
}
```

d. DELETE Request:
curl -X DELETE
https://jsonplaceholder.typicode.com/posts/1

```
C:\Windows\System32>curl -X DELETE "https://jsonplaceholder.typicode.com/posts/1"
{}
```

3. Answer the following questions:
a. What response do you get for each HTTP method?

Method	Response Example (JSON)
GET	Fetched resource data
POST	Echoes created object with new id
PUT	Echoes updated object
DELETE	No content returned (empty body)

b. What HTTP status codes are returned?

Method	Status Code	Meaning
GET	200 OK	Successful fetch
POST	201 Created	Resource created
PUT	200 OK	Resource updated
DELETE	200 OK	Resource deleted

c. What is the difference between PUT and POST?

Feature	POST	PUT
Purpose	Create a new resource	Update an existing resource (or create if not exists)
URL Target	Collection endpoint (/posts)	Specific resource (/posts/1)
Response	New id created	Existing resource overwritten