

NAME - GAUTAM KUMAR MAHAR(2103114) AND AUMKAR LOREKAR (2003108)

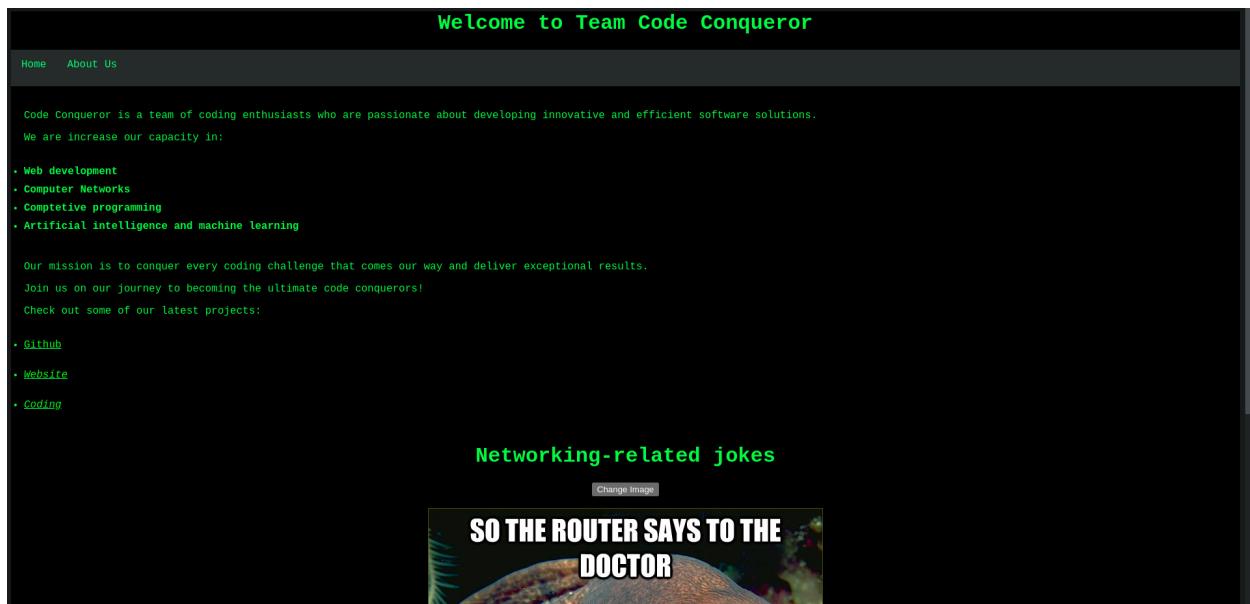
LAB 5 (HTTP, Web and DNS)

Question 1:

Choose a name for your team. Create a website for your team focussed on any theme/topic of your choice, using only raw HTML and a text editor. (Some suggested themes: Internet memes, Networking-related jokes, Networking history...)

- The website should consist of at least 2 webpages, where one of the pages should be named “index.html”
- Each webpage should contain a hyperlink to the other.
- The webpages should contain:
 - at least one link to some external website
 - at least one image
 - some formatted text such as headings and bold text. Do not include inappropriate content on the web pages. Store the objects (.html files, images etc) in a single folder on the local machine and test your web pages by opening the .html files in a browser.

Answer-



Check out some of our latest projects:

- [Github](#)
- [Website](#)
- [Coding](#)

Networking-related jokes

[Change image]

SO THE ROUTER SAYS TO THE
DOCTOR
IT HURTS WHEN IP.

© 2023 Code Conqueror. All rights reserved.

About Code Conqueror

Home About Us

Code Conqueror is a team of coding enthusiasts who are passionate about developing innovative and efficient software solutions. We are a diverse team with expertise in various programming languages and technologies.

Our team members include:

- Gautam Kumar Mahar - Web Developer
- Gautam Kumar Mahar - Competitive Programming
- Aumkar - AI/ML Engineer

Our goal is to provide high-quality software solutions that meet the needs of our clients and exceed their expectations. Get in touch with us to learn more about how we can help you achieve your coding goals!

Here are some helpful coding resources (I also use these)-

- [Codecademy](#)
- [Udemy](#)
- [Stack Overflow](#)

© 2023 Code Conqueror. All rights reserved.

Question 2

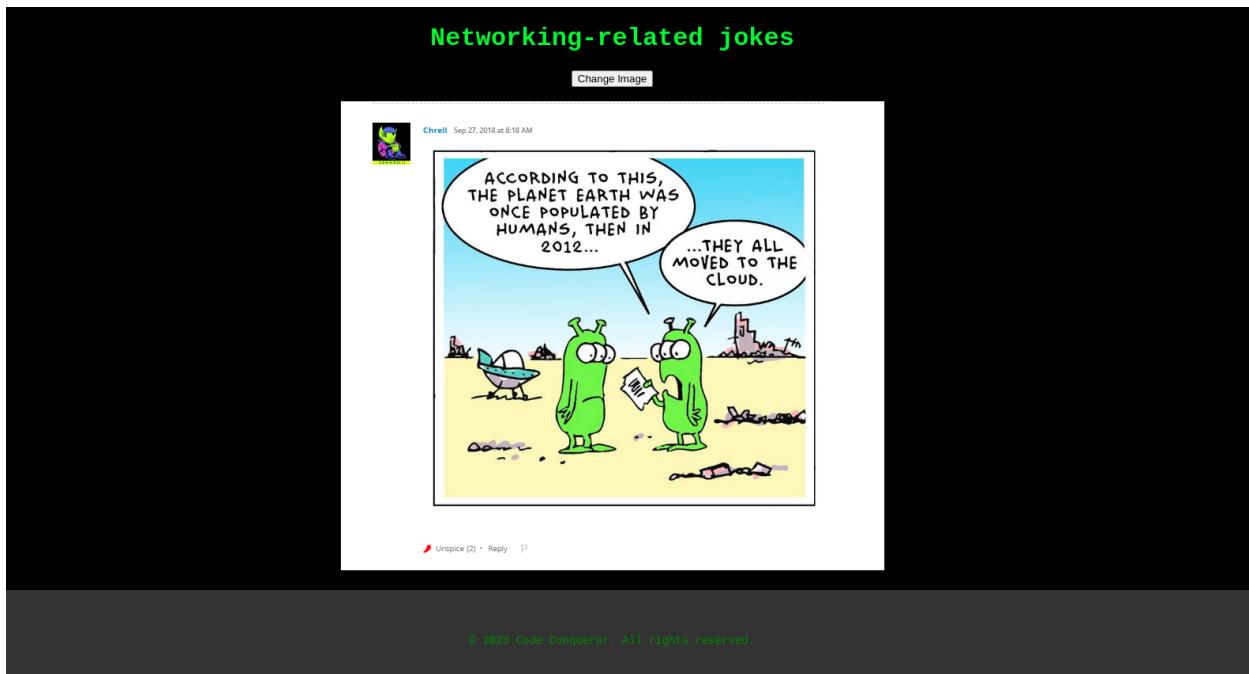
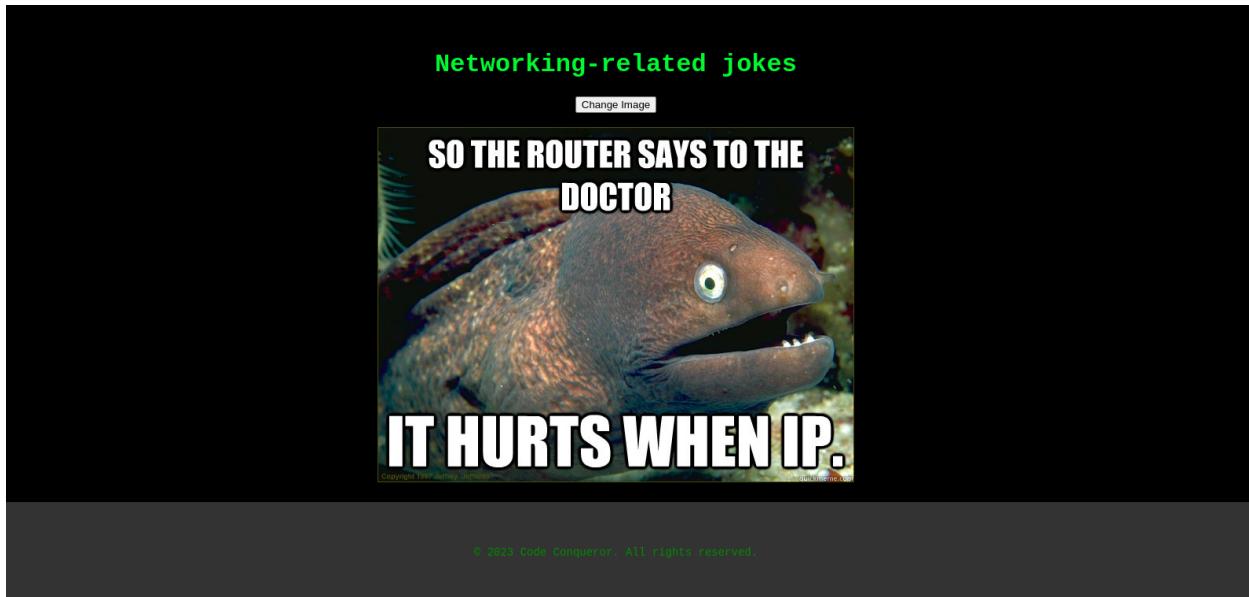
The essential components/technologies that go into a modern webpage are:

- HTML: to define the content of web pages
- CSS: to specify the style/layout of web pages and
- JavaScript: to program the behaviour of web pages

Tutorials and references for all three can be found here: <https://www.w3schools.com/default.asp>.

- Go through the initial tutorials for Javascript at <https://www.w3schools.com/js/default.asp>.
- Using Javascript, add a button to your webpage (created in q1) such that each time the user clicks this button, one of the images in your webpage changes to some other image.

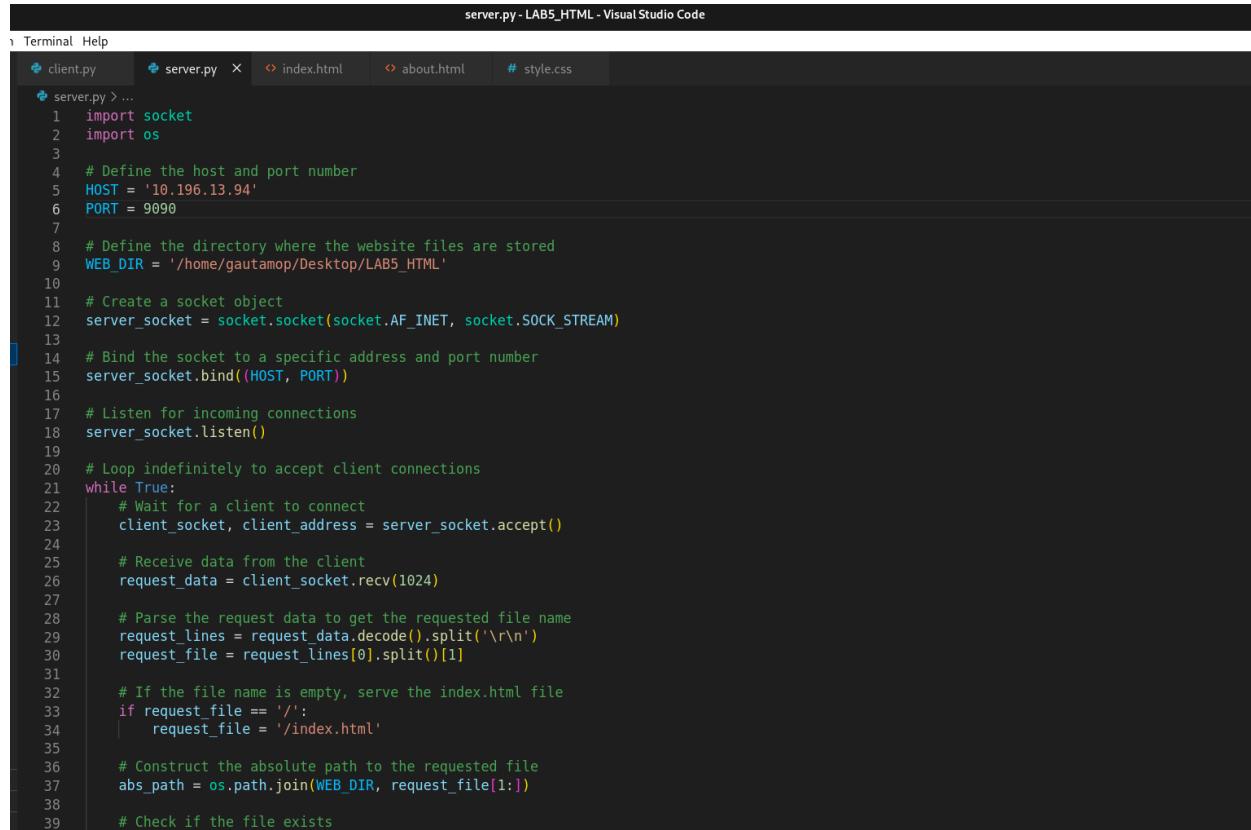
Ans:



Question 3

Create a web server for your website using Python. The web server process should accept a TCP connection from a Client process on port 9090 and generate responses to GET requests for objects contained in your website's folder. (You can modify and use the template for a simple Python Webserver provided on Google Classroom.)

Answer



The screenshot shows a Visual Studio Code interface with the title bar "server.py - LAB5_HTML - Visual Studio Code". The left sidebar shows files: client.py, server.py (which is the active tab), index.html, about.html, and style.css. The main editor area contains the following Python code:

```
1 Terminal Help
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```

```
server.py > ...
1 import socket
2 import os
3
4 # Define the host and port number
5 HOST = '10.196.13.94'
6 PORT = 9090
7
8 # Define the directory where the website files are stored
9 WEB_DIR = '/home/gautamop/Desktop/LAB5_HTML'
10
11 # Create a socket object
12 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13
14 # Bind the socket to a specific address and port number
15 server_socket.bind((HOST, PORT))
16
17 # Listen for incoming connections
18 server_socket.listen()
19
20 # Loop indefinitely to accept client connections
21 while True:
22     # Wait for a client to connect
23     client_socket, client_address = server_socket.accept()
24
25     # Receive data from the client
26     request_data = client_socket.recv(1024)
27
28     # Parse the request data to get the requested file name
29     request_lines = request_data.decode().split('\r\n')
30     request_file = request_lines[0].split()[1]
31
32     # If the file name is empty, serve the index.html file
33     if request_file == '/':
34         request_file = '/index.html'
35
36     # Construct the absolute path to the requested file
37     abs_path = os.path.join(WEB_DIR, request_file[1:])
38
39     # Check if the file exists
```

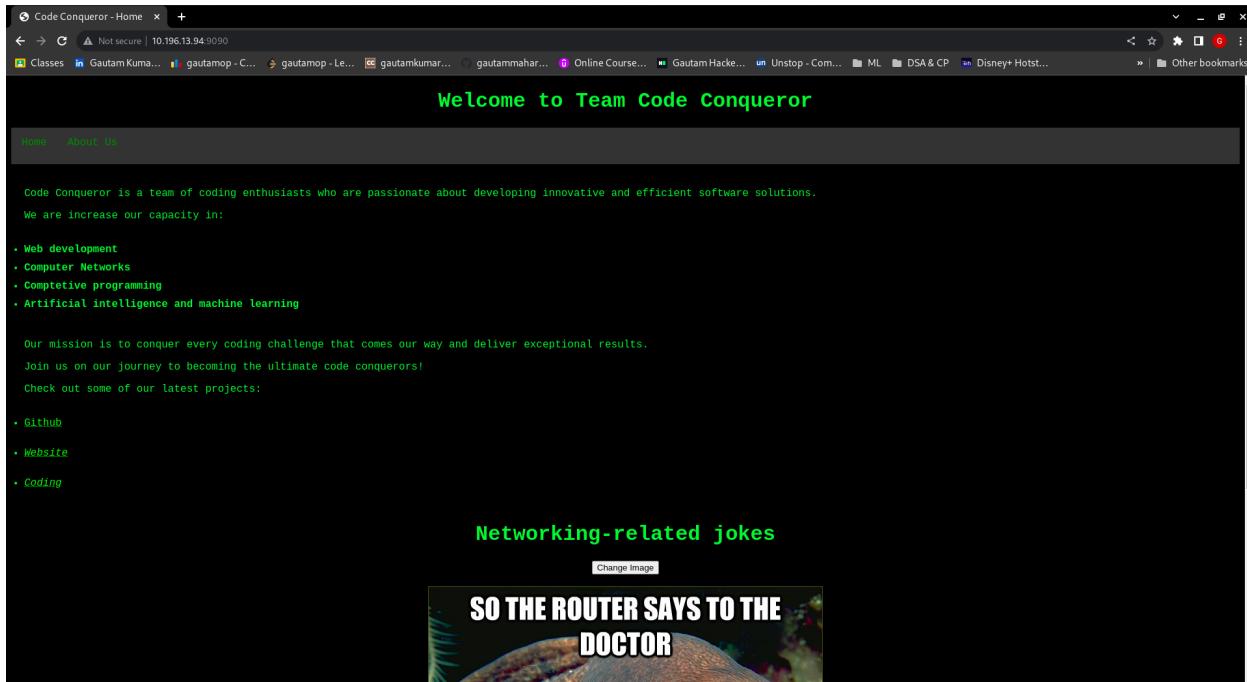
```

27
28     # Parse the request data to get the requested file name
29     request_lines = request_data.decode().split('\r\n')
30     request_file = request_lines[0].split()[1]
31
32     # If the file name is empty, serve the index.html file
33     if request_file == '/':
34         request_file = '/index.html'
35
36     # Construct the absolute path to the requested file
37     abs_path = os.path.join(WEB_DIR, request_file[1:])
38
39     # Check if the file exists
40     if os.path.exists(abs_path):
41         # Open the file and read its contents
42         with open(abs_path, 'rb') as file:
43             file_contents = file.read()
44
45         # Send a response back to the client
46         response_data = b"HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n" + file_contents
47         client_socket.sendall(response_data)
48     else:
49         # If the file doesn't exist, send a 404 error response
50         response_data = b"HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n<html><body><h1>404 Not Found</h1></body></html>"
51         client_socket.sendall(response_data)
52
53     # Close the connection
54     client_socket.close()
55
56 # Close the server socket
57 server_socket.close()
58

```

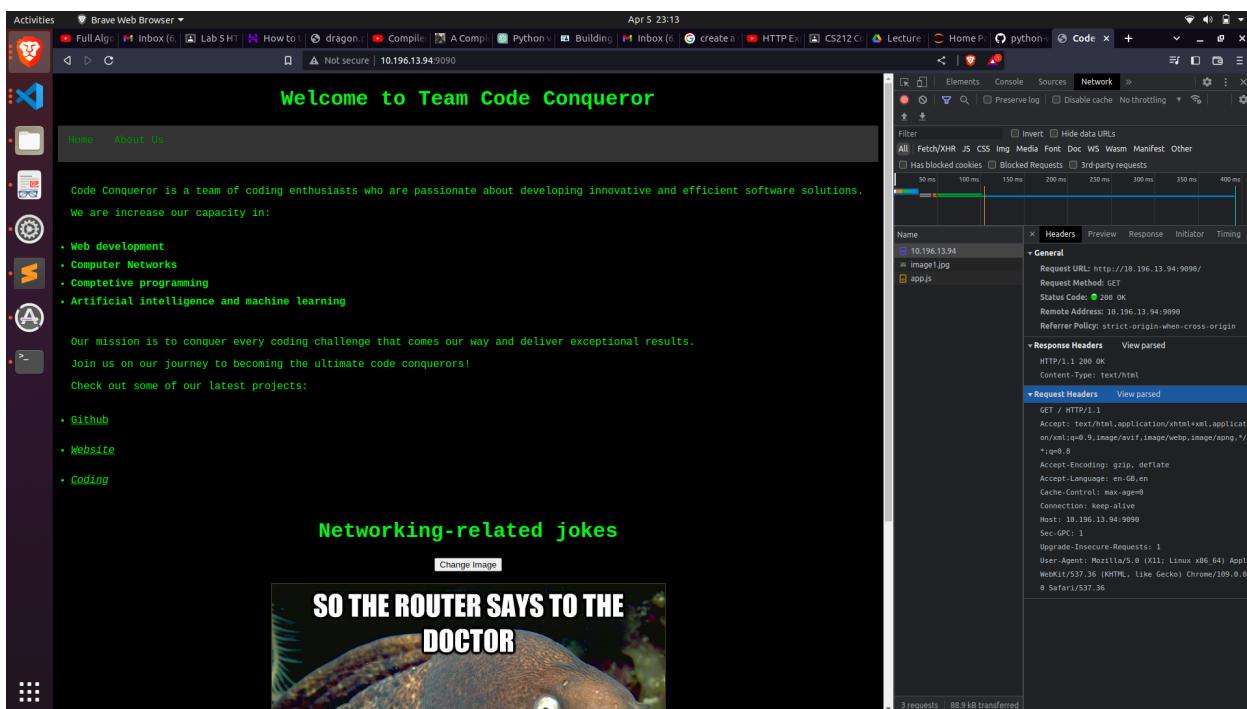
- Test your web server by entering the following URL in a browser’s address bar: “<http://localhost:9090>” on the same machine

The screenshot shows two windows side-by-side. The left window is a terminal session titled 'gautamop@gautamop: ~/Desktop/LAB5_HTML'. It displays the command '\$ python3 server.py' being run. The right window is a web browser titled 'New Tab' showing the URL '10.196.13.94:9090'. The browser interface includes a back/forward button, a search/address bar, and a tab bar with several other open tabs.



- Check if your website can be accessed from another machine in the lab. Use the private IP address of the webserver (10.xx.xx.xx) instead of ‘localhost’.

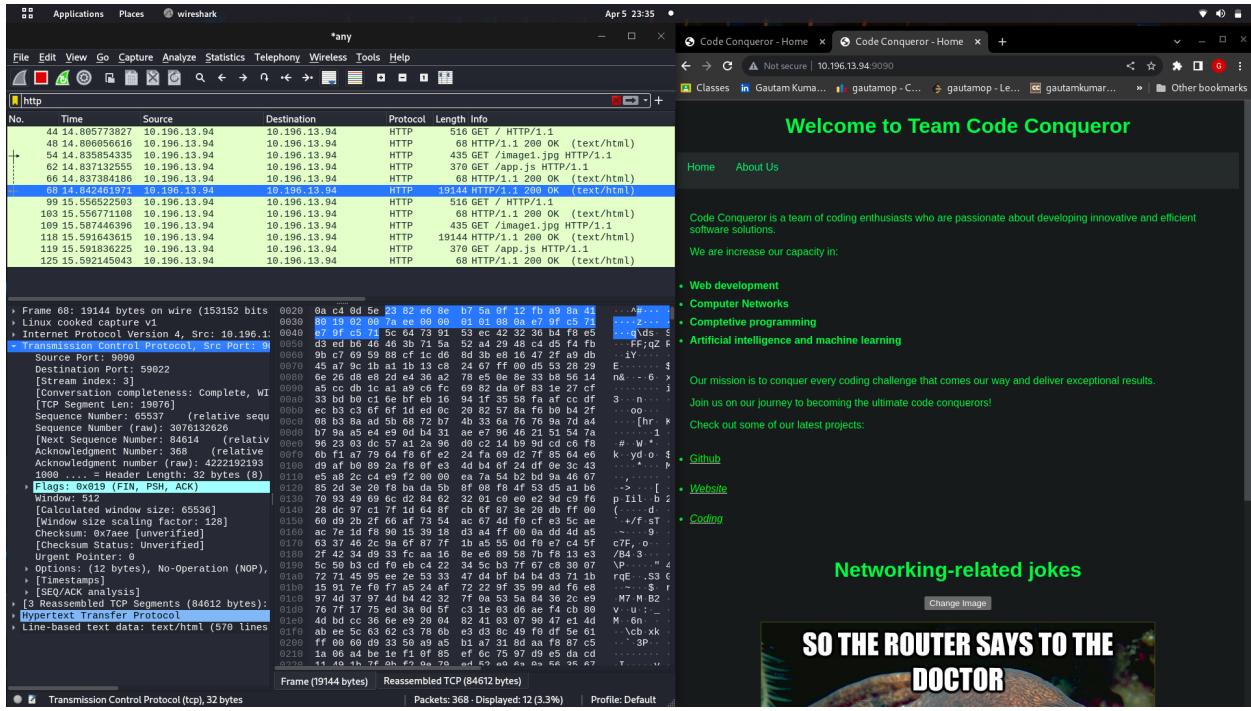
Other Machine POV:



Question 4 -

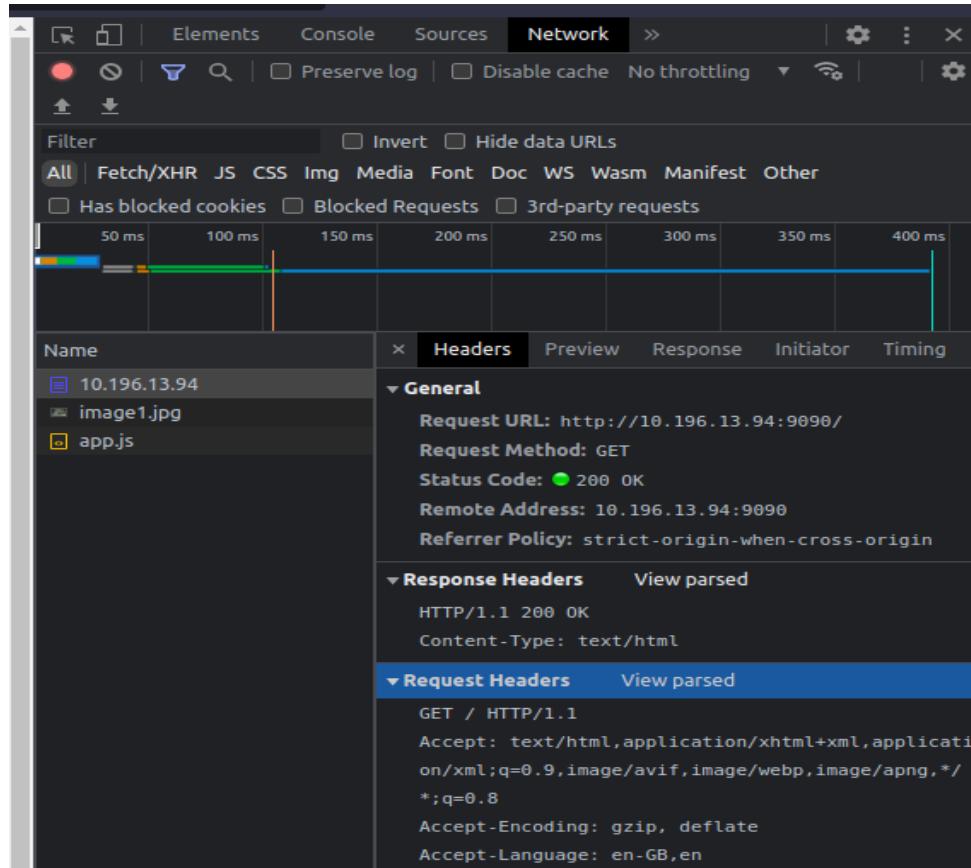
View the HTTP Request and Response traffic generated by the web browser using the browser's developer/Network tool. (For Google Chrome, press F12 to view the Network Tools)

- Check if you can see the “raw” HTTP requests and responses



- Find out whether the persistent or non-persistent connection was used by the browser for viewing your webpages

Ans - The connection is persistent Because

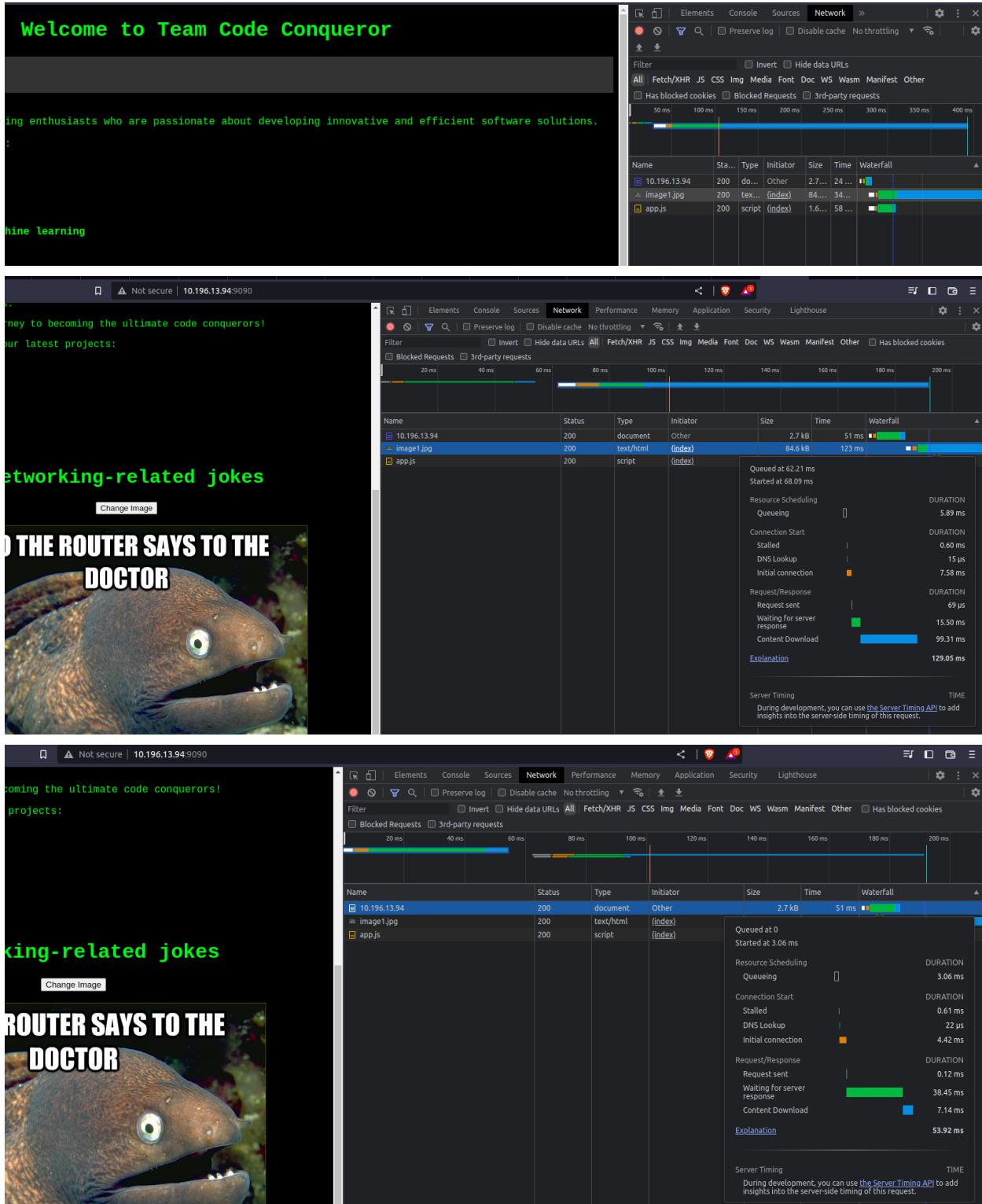


Here we can see HTTP/1.1 and

HTTP 1.1. In HTTP 1.1, all connections are considered persistent unless declared otherwise. The HTTP persistent connections do not use separate keepalive messages, they just allow multiple requests to use a single connection.

So, Now we can say that connect the connection is persistent.

- Open another webpage hosted remotely, such as <https://www.iitgoa.ac.in/~nehak/example.html>. View the HTTP traffic for this webpage in the Network tool. Does the browser download the components of the webpage (images etc) in a sequential or pipelined or parallel manner? [find out and show a demo to the TAs]



The screenshot shows a web browser displaying a joke page titled "Networking-related jokes". The page features a large image of a moray eel with the text "SO THE ROUTER SAYS TO THE DOCTOR" above it and "IT HURTS WHEN I DO" below it. To the left of the image is a sidebar with links to "Github", "Website", and "Coding". Below the image is a "Change Image" button.

On the right side of the browser, the developer tools Network tab is open, showing a timeline and a list of requests. The requests listed are:

- 10.196.13.94 (200) document Other
- image1.jpg (200) text/html (index)
- app.js (200) script (index)

The timeline shows the duration for each request, with "Waterfall" details for each. A detailed breakdown of the request for "app.js" is shown on the right, including Resource Scheduling, Queuing, Connection Start, Stalled, DNS Lookup, Initial connection, Request/Response (Request sent, Waiting for server response, Content Download), and an Explanation of 31.88 ms. The Server Timing section indicates a total time of 31.88 ms.

HERE, WE CAN SEE THAT parallel manner

- Observe the HTTP requests in Wireshark. Look out for the TCP handshaking (SYN/ SYN-ACK/ACK sequence). Is there one persistent TCP connection per webpage?

The screenshot shows Wireshark capturing network traffic for a session named "http". The packet list pane shows several HTTP requests and responses. Key packets include:

- Packet 44: 536 GET / HTTP/1.1 (1914 bytes on wire)
- Packet 48: 68 HTTP/1.1 200 OK (text/html)
- Packet 54: 435 GET /image1.jpg HTTP/1.1
- Packet 62: 370 GET /app.js HTTP/1.1
- Packet 66: 68 HTTP/1.1 200 OK (text/html)
- Packet 68: 19144 HTTP/1.1 200 OK (text/html)
- Packet 89: 536 GET / HTTP/1.1
- Packet 103: 68 HTTP/1.1 200 OK (text/html)
- Packet 109: 435 GET /image1.jpg HTTP/1.1
- Packet 118: 19144 HTTP/1.1 200 OK (text/html)
- Packet 119: 370 GET /app.js HTTP/1.1
- Packet 125: 68 HTTP/1.1 200 OK (text/html)

The details pane shows the structure of the captured packets, including headers like "Source Port: 9899" and "Destination Port: 59022". The bytes pane shows the raw binary data of the captured frames.

On the right side of the screen, a browser window displays the "Welcome to Team Code Conqueror" homepage. The page includes a "Home" link, a "About Us" section, and a "Check out some of our latest projects:" section with links to "Github", "Website", and "Coding". Below the browser is another instance of the joke page with the moray eel image.

37 ●

Code Conqueror - Home x Code Conqueror - Home x +

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Info
44	14.805773827	10.196.13.94	10.196.13.94	HTTP	516	GET / HTTP/1.1
48	14.806056616	10.196.13.94	10.196.13.94	HTTP	68	HTTP/1.1 200 OK (te
54	14.835854335	10.196.13.94	10.196.13.94	HTTP	435	GET /image1.jpg HTTP
62	14.837132555	10.196.13.94	10.196.13.94	HTTP	370	GET /app.js HTTP/1.1
66	14.837384186	10.196.13.94	10.196.13.94	HTTP	68	HTTP/1.1 200 OK (te
68	14.842461971	10.196.13.94	10.196.13.94	HTTP	19144	HTTP/1.1 200 OK (te
99	15.555622503	10.196.13.94	10.196.13.94	HTTP	516	GET / HTTP/1.1
103	15.556771108	10.196.13.94	10.196.13.94	HTTP	68	HTTP/1.1 200 OK (te
109	15.587446396	10.196.13.94	10.196.13.94	HTTP	435	GET /image1.jpg HTTP
118	15.591643615	10.196.13.94	10.196.13.94	HTTP	19144	HTTP/1.1 200 OK (te
119	15.591836225	10.196.13.94	10.196.13.94	HTTP	370	GET /app.js HTTP/1.1
125	15.592145043	10.196.13.94	10.196.13.94	HTTP	68	HTTP/1.1 200 OK (te
671	119.890915957	10.196.13.94	10.196.13.94	HTTP	516	GET / HTTP/1.1
675	119.891143783	10.196.13.94	10.196.13.94	HTTP	68	HTTP/1.1 200 OK (te
681	119.891220507	10.196.13.94	10.196.13.94	HTTP	435	GET /image1.jpg HTTP

255 >

- Frame 68: 19144 bytes on wire (153152 bits)
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 10.196.13.94, Dest: 10.196.13.94
- Transmission Control Protocol, Src Port: 9090, Dest Port: 59022
- Source Port: 9090
- Destination Port: 59022
- [Stream index: 3]
- [Conversation completeness: Complete, WI
- [TCP Segment Len: 19076]
- Sequence Number: 65537 (relative sequence number)
- Sequence Number (raw): 3076132626
- [Next Sequence Number: 84614 (relative acknowledgement number)]
- Acknowledgment Number: 368 (relative acknowledgement number)
- Acknowledgment number (raw): 4222192193
- 1000 = Header Length: 32 bytes (8)
- Flags: 0x019 (FIN, PSH, ACK)
- Window: 512
- [Calculated window size: 65536]
- [Window size scaling factor: 128]
- Checksum: 0x7aae [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- Options: (12 bytes), No-Operation (NOP),
- [Timestamps]
- [SEQ/ACK analysis]
- [3 Reassembled TCP Segments (84612 bytes):]
- Hypertext Transfer Protocol
- Line-based text data: text/html (570 lines)

Hex	Text
0030	80 19 02 00 7a ee 00 00 01 01 08 0a e7 9f c5 7
0040	e7 9f c5 71 5c 64 73 91 53 ec 42 32 36 b4 f8 e
0050	d3 ed b6 46 46 3b 71 5a 52 a4 29 48 c4 d5 f4 f
0060	9b c7 69 59 88 cf 1c d6 8d 3b e8 16 47 2f a9 d
0070	45 a7 9c 1b a1 1b 13 c8 24 67 ff 00 d5 53 28 2
0080	6e 26 d8 e8 2d e4 36 a2 78 e5 0e 8e 33 b8 56 1
0090	a5 cc db 1c a1 a9 c6 fc 69 82 da 0f 83 1e 27 0
00a0	33 bd b0 c1 6e bf eb 16 94 1f 35 58 fa af cc 0
00b0	ec b3 c3 6f 1f ed 0c 20 82 57 8a f6 b4 2
00c0	08 b3 8a ad 5b 68 72 b7 4b 33 6a 76 76 9a 7d a
00d0	b7 9a a5 e4 e9 0d b4 31 ae e7 96 46 21 51 5
00e0	96 23 03 dc 57 a1 2a 96 d0 c2 14 b9 9d cd c6 f
00f0	6b f1 a7 79 64 f8 6f e2 24 fa 69 d2 7f 85 64 e
0100	d9 af b0 89 2a f8 0f e3 4d b4 6f 24 df 0e 3c 4
0110	e5 a8 2c c4 e9 f2 00 00 ea 7a 54 b2 bd 9a 46 6
0120	85 2d 3e 20 f8 ba da 5b 8f 08 f8 4f 53 d5 a1 b
0130	70 93 49 69 6c d2 84 62 32 01 c0 e0 e2 9d c9 f
0140	28 dc 97 c1 7f 1d 64 8f cb 6f 87 3e 20 db ff 0
0150	60 d9 2b 2f 66 af 73 54 ac 67 4d f0 cf e3 5c a
0160	ac 7e 1d f8 90 15 39 18 d3 a4 ff 00 0a dd 4d a
0170	63 37 46 2c 9a 6f 87 7f 1b a5 55 0d f0 e7 c4 5
0180	2f 42 34 d9 33 fc aa 16 8e e6 89 58 7b f8 13 e
0190	5c 50 b3 cd f0 eb c4 22 34 5c b3 7f 67 c8 30 9
01a0	72 71 45 95 ee 2e 53 33 47 d4 bf b4 d3 71 1
01b0	15 91 7e f0 f7 a5 24 af 72 22 9f 35 99 ad f6 e
01c0	97 4d 37 97 4d b4 42 32 7f 0a 53 5a 84 36 2c e
01d0	76 7f 17 75 ed 3a 0d 5f c3 1e 03 d6 ae f4 cb 8
01e0	4d bd cc 36 6e e9 20 04 82 41 03 07 90 47 e1 4
01f0	ab ee 5c 63 62 c3 78 6b e3 d3 8c 49 f0 df 5e 6
0200	ff 00 60 d9 33 50 a9 a5 b1 a7 31 8d aa f8 87 0
0210	1a 06 a4 be 1e f1 0f 85 ef 6c 75 97 d9 e5 da 0
0220	11 49 1b 7f 0b f2 9e 79 ed 52 e9 6a 0a 56 35 6
0230	f0 af c7 47 63 b3 a1 af 88 11 7b 2f f6 74 87 1

Question 5-

You can use the dig command to send DNS queries to any specified DNS server.

The general format of the command: \$ dig @<DNS server> <name to be looked up> <type> Example: \$ dig @a.root-servers.net com NS

◦ Go through this short tutorial for the dig command

<https://www.hostinger.in/tutorials/how-to-use-the-dig-command-in-linux/>

◦ Find out the appropriate command options (for dig) to send a DNS query to the root server “a.root-servers.net”, requesting the address of all name-servers (NS) for the “com” top-level domains.

The screenshot shows two terminal windows side-by-side. Both windows are running on a Linux system with the command 'dig' being used to query the root servers for the 'com' domain. The left window shows the raw DNS query and response, including the authority section which lists various root servers like 'a.gtld-servers.net' and 'b.gtld-servers.net'. The right window shows the same query with additional tracing information, including the path of the query through multiple root servers and the final response from a specific server.

```
; <>> Dig 9.18.8.1-Debian <>> @a.root-servers.net com NS
; (2 servers found)
; global options: +cmd
; Got answer:
;=>>> HEADER<<- opcode: QUERY, status: NOERROR, id: 41179
; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27
; WARNING: recursion requested but not available
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; QUESTION SECTION:
; <com. IN NS>
; AUTHORITY SECTION:
; com. 172800 IN NS e.gtld-servers.net.
; com. 172800 IN NS b.gtld-servers.net.
; com. 172800 IN NS j.gtld-servers.net.
; com. 172800 IN NS m.gtld-servers.net.
; com. 172800 IN NS i.gtld-servers.net.
; com. 172800 IN NS f.gtld-servers.net.
; com. 172800 IN NS a.gtld-servers.net.
; com. 172800 IN NS g.gtld-servers.net.
; com. 172800 IN NS h.gtld-servers.net.
; com. 172800 IN NS l.gtld-servers.net.
; com. 172800 IN NS k.gtld-servers.net.
; com. 172800 IN NS c.gtld-servers.net.
; com. 172800 IN NS d.gtld-servers.net.

; ADDITIONAL SECTION:
; e.gtld-servers.net. 172800 IN A 192.12.94.30
; e.gtld-servers.net. 172800 IN AAAA 2001:502:1ca1::30
; b.gtld-servers.net. 172800 IN A 192.33.14.30
; b.gtld-servers.net. 172800 IN AAAA 2001:503:231d::2:30
; j.gtld-servers.net. 172800 IN A 192.48.79.30
; j.gtld-servers.net. 172800 IN AAAA 2001:502:7094::30
; m.gtld-servers.net. 172800 IN A 192.55.83.30
; m.gtld-servers.net. 172800 IN AAAA 2001:501:b1f9::30
; i.gtld-servers.net. 172800 IN A 192.43.172.30
; i.gtld-servers.net. 172800 IN AAAA 2001:503:39c1::30
; f.gtld-servers.net. 172800 IN A 192.35.51.30
; f.gtld-servers.net. 172800 IN AAAA 2001:503:d414::30
; a.gtld-servers.net. 172800 IN A 192.5.6.30
; a.gtld-servers.net. 172800 IN AAAA 2001:503:a83e::2:30
; g.gtld-servers.net. 172800 IN A 192.42.93.30
; g.gtld-servers.net. 172800 IN AAAA 2001:503:eea3::30
; h.gtld-servers.net. 172800 IN A 192.54.112.30
; h.gtld-servers.net. 172800 IN AAAA 2001:502:8cc::30
; l.gtld-servers.net. 172800 IN A 192.41.162.30
; l.gtld-servers.net. 172800 IN AAAA 2001:500:d937::30
; k.gtld-servers.net. 172800 IN A 192.52.178.30
; k.gtld-servers.net. 172800 IN AAAA 2001:503:d2d::30
; c.gtld-servers.net. 172800 IN A 192.26.92.30
; c.gtld-servers.net. 172800 IN AAAA 2001:503:83eb::30
; d.gtld-servers.net. 172800 IN AAAA 2001:500:856e::30

; Query time: 192 msec
; SERVER: 198.41.0.4453(a.root-servers.net) (UDP)
; WHEN: Wed Apr 05 23:45:56 IST 2023
; MSG SIZE rcvd: 828
```

The screenshot shows a single terminal window displaying the output of the 'dig' command for the 'stackoverflow.com' domain. The output includes the question, authority section, and additional section. The authority section lists several IP addresses for the 'com' root servers. The additional section lists various IP addresses for different sub-domains within the 'com' domain, such as 'a.stackoverflow.com', 'b.stackoverflow.com', etc. The terminal window also shows some decorative text at the bottom related to networking and jokes.

```
; <>>> Dig 9.18.8.1-Debian <>> @a.root-servers.net stackoverflow.com NS
; (2 servers found)
; global options: +cmd
; Got answer:
;=>>> HEADER<<- opcode: QUERY, status: NOERROR, id: 4453
; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27
; WARNING: recursion requested but not available
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; QUESTION SECTION:
; <stackoverflow.com. IN NS>
; AUTHORITY SECTION:
; com. 172800 IN NS a.gtld-servers.net.
; com. 172800 IN NS g.gtld-servers.net.
; com. 172800 IN NS h.gtld-servers.net.
; com. 172800 IN NS l.gtld-servers.net.
; com. 172800 IN NS k.gtld-servers.net.
; com. 172800 IN NS c.gtld-servers.net.
; com. 172800 IN NS d.gtld-servers.net.

; ADDITIONAL SECTION:
; a.stackoverflow.com. 172800 IN A 192.12.94.30
; a.stackoverflow.com. 172800 IN AAAA 2001:502:1ca1::30
; b.stackoverflow.com. 172800 IN A 192.33.14.30
; b.stackoverflow.com. 172800 IN AAAA 2001:503:231d::2:30
; j.stackoverflow.com. 172800 IN A 192.48.79.30
; j.stackoverflow.com. 172800 IN AAAA 2001:502:7094::30
; m.stackoverflow.com. 172800 IN A 192.55.83.30
; m.stackoverflow.com. 172800 IN AAAA 2001:501:b1f9::30
; i.stackoverflow.com. 172800 IN A 192.43.172.30
; i.stackoverflow.com. 172800 IN AAAA 2001:503:39c1::30
; f.stackoverflow.com. 172800 IN A 192.35.51.30
; f.stackoverflow.com. 172800 IN AAAA 2001:503:d414::30
; a.stackoverflow.com. 172800 IN A 192.5.6.30
; a.stackoverflow.com. 172800 IN AAAA 2001:503:a83e::2:30
; g.stackoverflow.com. 172800 IN A 192.42.93.30
; g.stackoverflow.com. 172800 IN AAAA 2001:503:eea3::30
; h.stackoverflow.com. 172800 IN A 192.54.112.30
; h.stackoverflow.com. 172800 IN AAAA 2001:502:8cc::30
; l.stackoverflow.com. 172800 IN A 192.41.162.30
; l.stackoverflow.com. 172800 IN AAAA 2001:500:d937::30
; k.stackoverflow.com. 172800 IN A 192.52.178.30
; k.stackoverflow.com. 172800 IN AAAA 2001:503:d2d::30
; c.stackoverflow.com. 172800 IN A 192.26.92.30
; c.stackoverflow.com. 172800 IN AAAA 2001:503:83eb::30
; d.stackoverflow.com. 172800 IN AAAA 2001:500:856e::30

; Query time: 192 msec
; SERVER: 198.41.0.4453(a.root-servers.net) (UDP)
; WHEN: Wed Apr 05 23:45:56 IST 2023
; MSG SIZE rcvd: 828
```

- Try running dig with the “+trace” option to see all the steps performed in iteratively querying the hierarchy of DNS servers. Try to make sense of the output. Example: dig +trace @8.8.8.8 stackoverflow.com
 - Open Wireshark and observe the DNS traffic on your computer using the filter “DNS”

We try with leetcode

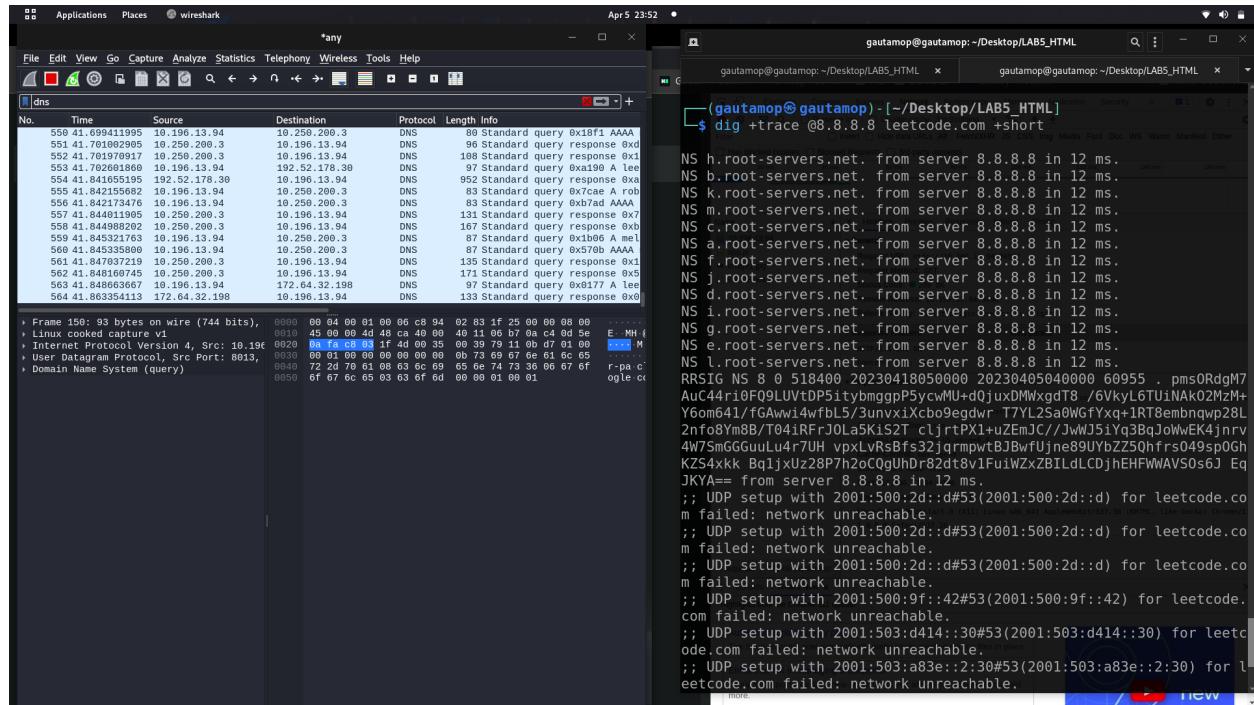
```
[gautamop@gautamop] ~[Desktop/LAB5_HTML]
$ dig +trace @8.8.8.8 leetcode.com

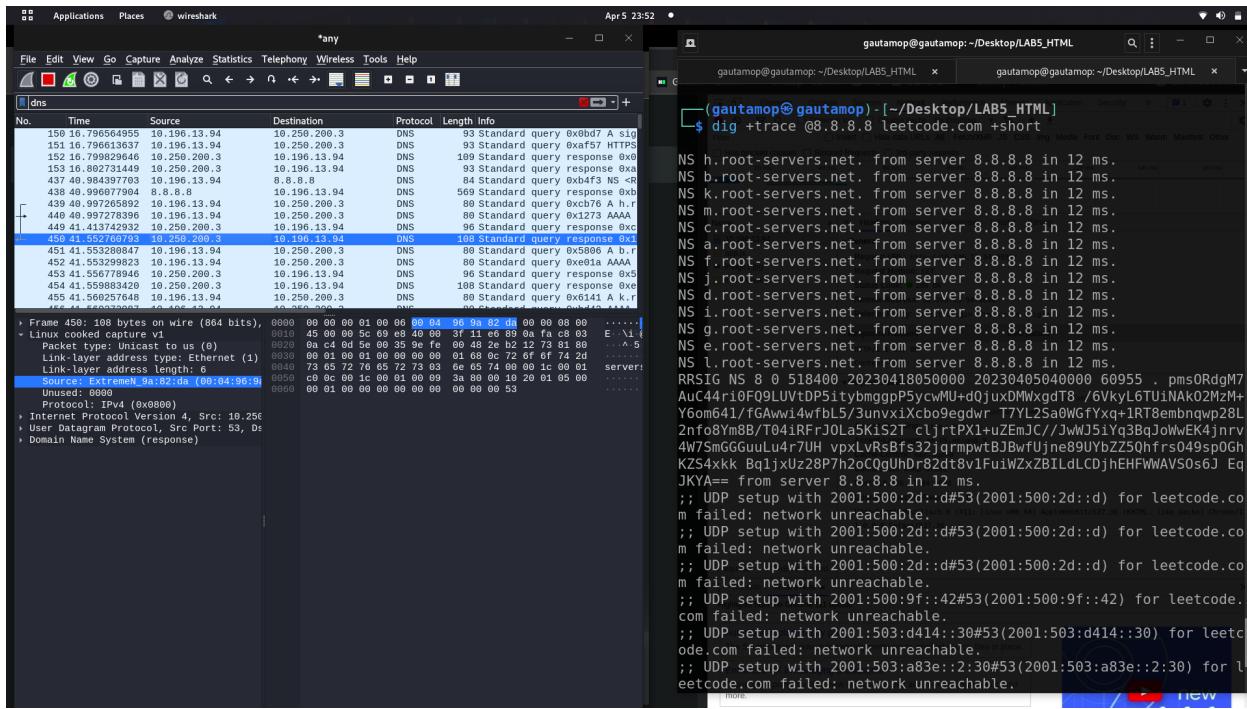
; <>> DiG 9.18.1-Debian <>> +trace @8.8.8.8 leetcode.com
; (1 server found)
;; global options: +cmd
;; +-----+
; 75333 IN A 192.168.1.13 NS h.root-servers.net.
; 75333 IN A 192.168.1.13 NS b.root-servers.net.
; 75333 IN A 192.168.1.13 NS k.root-servers.net.
; 75333 IN A 192.168.1.13 NS m.root-servers.net.
; 75333 IN A 192.168.1.13 NS c.root-servers.net.
; 75333 IN A 192.168.1.13 NS a.root-servers.net.
; 75333 IN A 192.168.1.13 NS f.root-servers.net.
; 75333 IN A 192.168.1.13 NS j.root-servers.net.
; 75333 IN A 192.168.1.13 NS d.root-servers.net.
; 75333 IN A 192.168.1.13 NS i.root-servers.net.
; 75333 IN A 192.168.1.13 NS g.root-servers.net.
; 75333 IN A 192.168.1.13 NS e.root-servers.net.
; 75333 IN A 192.168.1.13 NS l.root-servers.net.
; 75333 IN RRSIG NS 8 0 518406 20230418050000 20230405040000 60955 . pms0RdgM7Au44ri0F09LUvTP5itybmogpP5ycwMu+0qjuxDMwXqdT8 /6VkyL6TuNak02M
zMaY6om641/fcAwI4vfbL5/3unvx1Xcb09gdwT T7Y12s0dWfYxq+IRt8emhbgwZ2L2nf08Yy8Bp/704irFJ0l5K1527 cljrpxTI+zemCJ//JwWJ51Yq3BqJ0wE4jnrw4W75mGGuuLu4r7UH vpxLvrBsBfs32jqmpw
tBJBwfuJne89UYzZ50hfr5049sp0ghkZS4xxKk Bqjlx02z8P/fz0cQ0jdR78d2t8V1fuiWzxB1ldCJhEHFWWAS0s6J EqJQKYA==

Received 525 bytes from 8.8.8.8#53 in 52 ms

;; UDP setup with 2001:500:2d::d#53(2001:500:2d::d) for leetcode.com failed: network unreachable.
```

```
...  
.: Received 1200 bytes from 192.112.36.4#53(g.root-servers.net) in 180 ms  
  
...: UDP setup with 2001:503:d414::30#53(2001:503:d414::30) for leetcode.com failed: network unreachable.  
...: UDP setup with 2001:503:231d::2:30#53(2001:503:231d::2:30) for leetcode.com failed: network unreachable.  
...: UDP setup with 2001:501:b1f9::30#53(2001:501:b1f9::30) for leetcode.com failed: network unreachable.  
leetcode.com. 172800 IN NS rob.ns.cloudflare.com.  
leetcode.com. 172800 IN NS melinda.ns.cloudflare.com.  
CK0P0JMG874L3REJF7EFNB430QVIT88SM.com. 86400 IN NSEC3 1 1 0 -2 CK02D6N1417EQH8NA30NS6104UBL8G5 NS SOA RRSIG DNSKEY NSEC3PARAM  
CK0P0JMG874L3REJF7EFNB430QVIT88SM.com. 86400 IN RRSIG NSEC3 2 8 86400 20230412042392 20230405031302 36739 com. RXHRYUsBj8TdsbwKciKntoYlhP//ZQvbggdKqnm6dyqyD2so2VH8D8 sVuUxXeB0PfFMwQnrlsU1f5H5BFR/jU3lYBm71GlyfWk0ndTl3z2C062wsWklUr1Xqj6hpv0ooxhJml5s u8hRopKGYzIV7qzka5qhl5NwzjQl02km3Vz0n7Tqjv6hSg==  
V45LL01QBSOEFBR206JUE8NBKB02KV1D.com. 86400 IN NSEC3 1 1 0 -2 V45M2PHNUMJ2Q0205R016H1940P8860IQ6 DS RSIG  
V45LL01QBSOEFBR206JUE8NBKB02KV1D.com. 86400 IN RRSIG NSEC3 2 8 86400 20230411045112 20230404034112 36739 com. RLcIkK+jKEYNIy1v0qpg0l+jCbu5D ZtP0P0CjBgn9my4pgBqKMa6dW oedBpb4TQ01-E5pxwNK3tNCawQjbGdn+2PucB4j0Ho3M25QKjx3f9 qyXUEN8Jnhwo01+9C06q5scjqjRbd694a3K3lpXqoUtewoatTEbonh Dv3llCRmx0sUtiQdydMKJdw4SSWTrUsM5Eqx5yGTSam1Eo==  
...: Received 908 bytes from 192.53.83.30#53(g.root-servers.net) in 164 ms  
  
...: UDP setup with 2803:f800::6ca2::0c6a#53(2803:f800::50::6ca2::0c6) for leetcode.com failed: network unreachable.  
...: UDP setup with 2803:f800::6ca2::18c#53(2803:f800::50::6ca2::c18c) for leetcode.com failed: network unreachable.  
leetcode.com. 300 IN A 172.67.72.213  
leetcode.com. 300 IN A 104.26.9.101  
leetcode.com. 300 IN A 104.26.8.101  
...: Received 89 bytes from 108.162.192.198#53(melinda.ns.cloudflare.com) in 60 ms
```





- Find the address of your local DNS server

```
(gautamop@gautamop) - [~/Desktop/LAB5_HTML]
$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 10.250.200.3
```

- Find out the addresses of any 3 root-level DNS servers

```
(gautamop@gautamop) - [~/Desktop/LAB5_HTML]
$ dig +short NS .

c.root-servers.net.
m.root-servers.net.
l.root-servers.net.
b.root-servers.net.
g.root-servers.net.
d.root-servers.net.
a.root-servers.net.
f.root-servers.net.
h.root-servers.net.
i.root-servers.net.
j.root-servers.net.
k.root-servers.net.
e.root-servers.net.
```

Thank You

Thank You