1 Understanding HTTP (20 points)

For this section, you will only need your web browser and Wireshark. (Wireshark only for taking a look at things yourself.) To understand GET HTTP requests a little better, we want to do a couple of GET requests through an API. (For this assignment, it is for the GitHub API.) The API works by nesting topics - where you will need to do a base call to get some data, use that data to perform the next call, and so forth.

To begin, go to the GitHub API documentation webpage. This will give you some information about the API. It might be overwhelming to read at first, but that is okay. What's interesting, however, is that you can make requests directly on your browser and not just from things like Java, Javascript, etc.

For instance, we use the List repositories for a user API call to get all public repos from a specific user. In your browser, use the following URL:

https://api.github.com/users/amehlhase316/repos

This should show you a JSON file for all of my public repos. (You can also use a different username if you like, maybe you should and check what you come up with)

Next, we want to look at one specific repo. To do this, we will use the Getarepository API call. For that, go to:

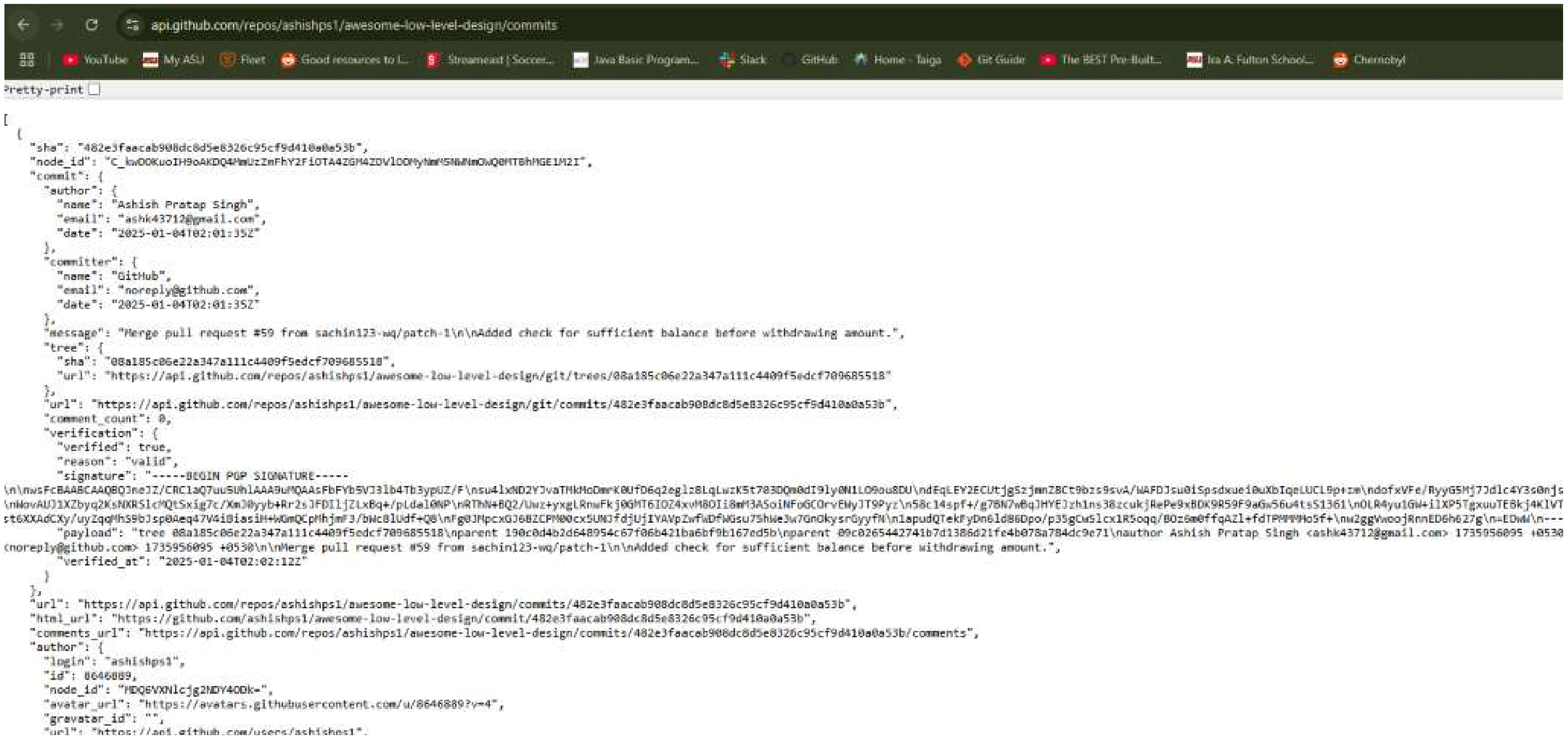
https://api.github.com/repos/amehlhase316/memoranda

This will give us JSON on the Memoranda project where I am the owner. You can use any other username and project you like (make sure it is public though).

Now, find and run a new call that gets all the commits on the default branch for a repository of your choosing. Any public repo that has some branches and commits on them is fine.

Deliverable (5 points): Paste the URL you used into your document, take a screenshot, and add it to your document. (The screenshot should show the call you made and its resulting JSON. If the JSON is too big, partially showing it is okay as well.)

https://api.github.com/repos/ashishps1/awesome-low-level-design/commits



```
[
  {
    "sha": "482e3faacab908dc8d5e8326c95cf9d410a0a53b",
    "node_id": "C_kwDOKuoIH9oAKDQ4MmUzZmFhY2FiOTA4ZGM4ZDVlODMyNmM5NWNmOWQ0MTBhMGE1M2I",
    "commit": {
      "author": {
        "name": "Ashish Pratap Singh",
        "email": "ashk43712@gmail.com",
        "date": "2025-01-04T02:01:35Z"
      },
      "committer": {
        "name": "GitHub",
        "email": "noreply@github.com",
        "date": "2025-01-04T02:01:35Z"
      },
      "message": "Merge pull request #59 from sachin123-wq/patch-1\n\nAdded check for sufficient balance before withdrawing amount.",
      "tree": {
        "sha": "08a185c06e22a347a111c4409f5edcf709685518",
        "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/git/trees/08a185c06e22a347a111c4409f5edcf709685518"
      },
      "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/git/commits/482e3faacab908dc8d5e8326c95cf9d410a0a53b",
      "comment_count": 0,
      "verification": {
        "verified": true,
        "reason": "valid",
        "signature": "-----BEGIN PGP SIGNATURE-----
\n\nwsFcBAABCAAQBQJneJZ/CRC1aQ7uu9UhlAAA9uMQAAsFbFYb5VJ3lb4Tb3ypUZ/F\nsu4lxND2YJvaTMkMoDmrK0UfD6q2eglz8LqLwzK5t783DQm0dI9ly0N1LO9ou8DU\ndEqLEY2ECUtjg5zjmnZ8Ct9bzs9svA/WAFDJsu0iSpsdxue10uXbIqeLUCL5p+zm\ndofxVFe/RyyG5Mj7Jdlc4Y3s0njs
\nWovAUJ1XZbyq2KsNXR5lcMQtSxig7c/XmJ0yyb+Rr2sJFDIljZLxBq+/pLdal0NP\nRThN+BQ2/Uwz+yxgLRnwFkj0GMT6IOZ4xvM8OIi8mM3A5oiNFoGCOrvEWyJT9Pyz\n58c14spf+/g7BN7wBqJHYEJzh1ns38zcukjRePe9xBDK9R59F9aGw56u4tsS1361\nOLR4yu1GW+i1XP5TgxuuTE8kj4KlVT
st6XXAdCXy/uyZqqMh59bJsp0Aeq47V4i8iasiH+WGmQCpMhjmF3/bWc8lUdf+Q8\nFg0JMpcxGJ68ZCPM00cx5UNJfdjUjIYAVpZwfWDfWGsu75hWe3w7GnOkysrGyyfN\n1apudQTekFyDn6ld86Dpo/p35gCw5lcx1R5oqq/BOz6m0ffqAZl+fdTPMMMMo5f+\nw2ggVwoojRnnED6h627g\n=EDwW\n---
        "payload": "tree 08a185c06e22a347a111c4409f5edcf709685518\nparent 190c0d4b2d648954c67f06b421ba6bf9b167ed5b\nparent 09c0265442741b7d1386d21fe4b078a784dc9e71\nauthor Ashish Pratap Singh <ashk43712@gmail.com> 1735956095 +0530
<noreply@github.com> 1735956095 +0530\n\nMerge pull request #59 from sachin123-wq/patch-1\n\nAdded check for sufficient balance before withdrawing amount.",
        "verified_at": "2025-01-04T02:02:12Z"
      }
    },
    "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/commits/482e3faacab908dc8d5e8326c95cf9d410a0a53b",
    "html_url": "https://github.com/ashishps1/awesome-low-level-design/commit/482e3faacab908dc8d5e8326c95cf9d410a0a53b",
    "comments_url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/commits/482e3faacab908dc8d5e8326c95cf9d410a0a53b/comments",
    "author": {
      "login": "ashishps1",
      "id": 8646889,
      "node_id": "MDQ6VXNlcjg2NDY4ODk=",
      "avatar_url": "https://avatars.githubusercontent.com/u/8646889?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/ashishps1",
```

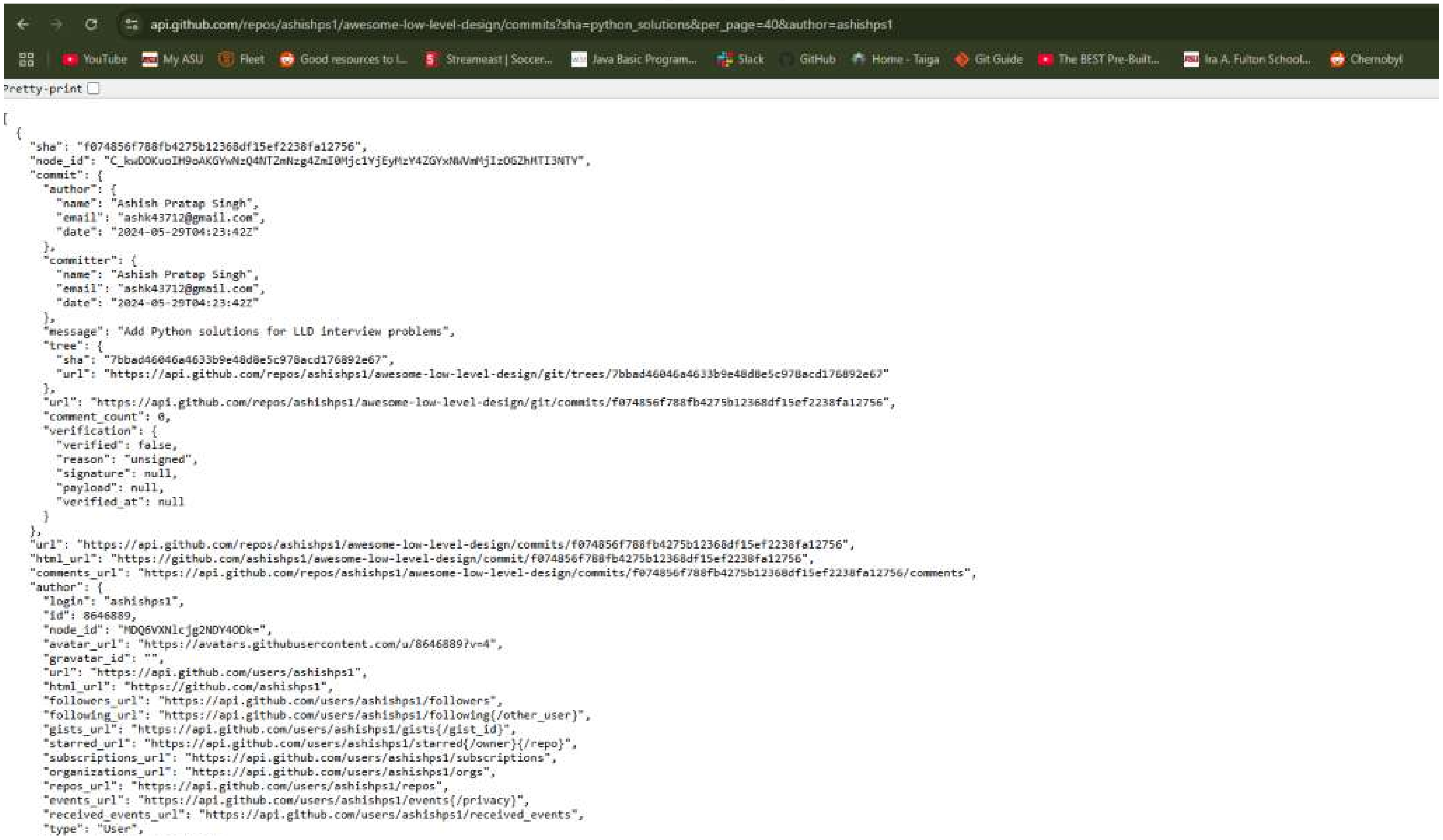
```
    "html_url": "https://github.com/ashishps1",
    "followers_url": "https://api.github.com/users/ashishps1/followers",
    "following_url": "https://api.github.com/users/ashishps1/following{/other_user}",
    "gists_url": "https://api.github.com/users/ashishps1/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/ashishps1/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/ashishps1/subscriptions",
    "organizations_url": "https://api.github.com/users/ashishps1/orgs",
    "repos_url": "https://api.github.com/users/ashishps1/repos",
    "events_url": "https://api.github.com/users/ashishps1/events{/privacy}",
    "received_events_url": "https://api.github.com/users/ashishps1/received_events",
    "type": "User",
    "user_view_type": "public",
    "site_admin": false
  },
  "committer": {
```

From here, perform another call. In this call, add GET parameters/query arguments (to the call from above) that specifies a specific branch (not the default), and sets the perpage limit to 40 (so 50 commits are shown instead of just 30). Usually, APIs limit their response size and return one page. If you need more data you would then call the next page or increase the page limit (as we do here for simplicity - usually there is a limit to this "page limit").
If you are stuck on this, please go to the documentation and try to understand how this API call works.

Look for one more API call you can make on a public repo, any call that uses some query arguments is fine.

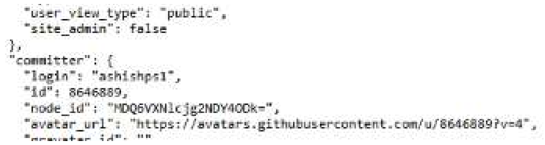
Deliverable (5 points): Paste the URLs you used into your document, take a screenshot of your result, and add it to the document (do this for all API calls you made).

https://api.github.com/repos/ashishps1/awesome-low-level-design/commits?sha=python_solutions&per_page=40&author=ashishps1



api.github.com/repos/ashishps1/awesome-low-level-design/commits?sha=python_solutions&per_page=40&author=ashishps1

```
[
  {
    "sha": "f074856f788fb4275b12368df15ef2238fa12756",
    "node_id": "C_kwDOKuoIH9oAKGYwNzQ4NTZmNzg4ZmI0Mjc1YjEyMzY4ZGYxNWVmMjIzOGZhMTI3NTY",
    "commit": {
      "author": {
        "name": "Ashish Pratap Singh",
        "email": "ashk43712@gmail.com",
        "date": "2024-05-29T04:23:42Z"
      },
      "committer": {
        "name": "Ashish Pratap Singh",
        "email": "ashk43712@gmail.com",
        "date": "2024-05-29T04:23:42Z"
      },
      "message": "Add Python solutions for LLD interview problems",
      "tree": {
        "sha": "7bbad46046a4633b9e48d8e5c978acd176892e67",
        "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/git/trees/7bbad46046a4633b9e48d8e5c978acd176892e67"
      },
      "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/git/commits/f074856f788fb4275b12368df15ef2238fa12756",
      "comment_count": 0,
      "verification": {
        "verified": false,
        "reason": "unsigned",
        "signature": null,
        "payload": null,
        "verified_at": null
      }
    },
    "url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/commits/f074856f788fb4275b12368df15ef2238fa12756",
    "html_url": "https://github.com/ashishps1/awesome-low-level-design/commit/f074856f788fb4275b12368df15ef2238fa12756",
    "comments_url": "https://api.github.com/repos/ashishps1/awesome-low-level-design/commits/f074856f788fb4275b12368df15ef2238fa12756/comments",
    "author": {
      "login": "ashishps1",
      "id": 8646889,
      "node_id": "MDQ6VXNlcjg2NDY4ODk=",
      "avatar_url": "https://avatars.githubusercontent.com/u/8646889?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/ashishps1",
      "html_url": "https://github.com/ashishps1",
      "followers_url": "https://api.github.com/users/ashishps1/followers",
      "following_url": "https://api.github.com/users/ashishps1/following{/other_user}",
      "gists_url": "https://api.github.com/users/ashishps1/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/ashishps1/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/ashishps1/subscriptions",
      "organizations_url": "https://api.github.com/users/ashishps1/orgs",
      "repos_url": "https://api.github.com/users/ashishps1/repos",
      "events_url": "https://api.github.com/users/ashishps1/events{/privacy}",
      "received_events_url": "https://api.github.com/users/ashishps1/received_events",
      "type": "User",
```



```
    "user_view_type": "public",
    "site_admin": false
  },
  "committer": {
    "login": "ashishps1",
    "id": 8646889,
    "node_id": "MDQ6VXNlcjg2NDY4ODk=",
    "avatar_url": "https://avatars.githubusercontent.com/u/8646889?v=4",
    "gravatar_id": ""
```

Deliverable: Answer the following in your document (10 points):

1. Explain the specific API calls you used, include the information you needed to provide and include the link to the API documentation for that call.
   - https://api.github.com/repos/ashishps1/awesome-low-level-design/commits?sha=python_solutions&per_page=40&author=ashishps1
   - Here I used a combination of 3 API calls.
     - Sha, this call is query a specific branch within the repository
     - Per_pag, this call is to now limit the amount of commits displayed per page
     - Author, this will now query only commits by the listed author
2. Explain the difference between stateless and stateful communication.
   - Stateless is an instance of communication to a server that doesn't hold memory of the interactions once the communication line has been severed. Returning a communication link could put you through to any available version of a server available for what you are doing. The servers don't have memory of any previous interaction because not only is it a new server but it is designed to fully function with no prior context.
   - Stateful is the opposite, it maintains a saved state of the previous instances of communication.

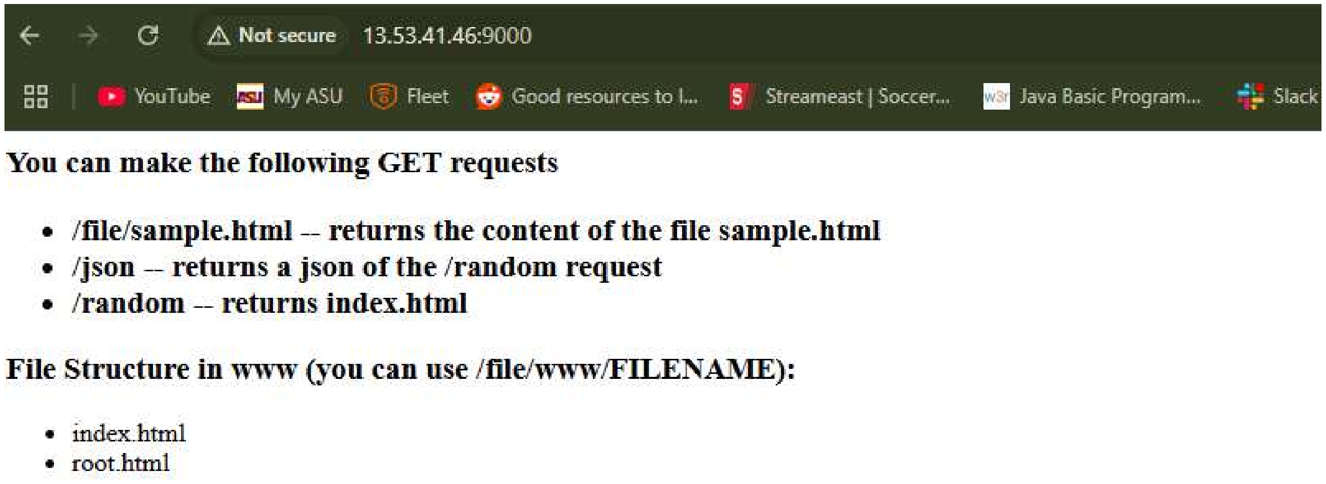
You should take a look at Wireshark and check the communication that was going on with your calls. You do not have to document anything for me, but I advise you to take a detailed look and to do your best in understanding the traffic generated.

2 Set up your second system and run servers on it (70 points)

2.1 Getting sample code onto your systems (should be done already)

- ***Done***

2.2 Running a Simple Java WebServer (10 points)

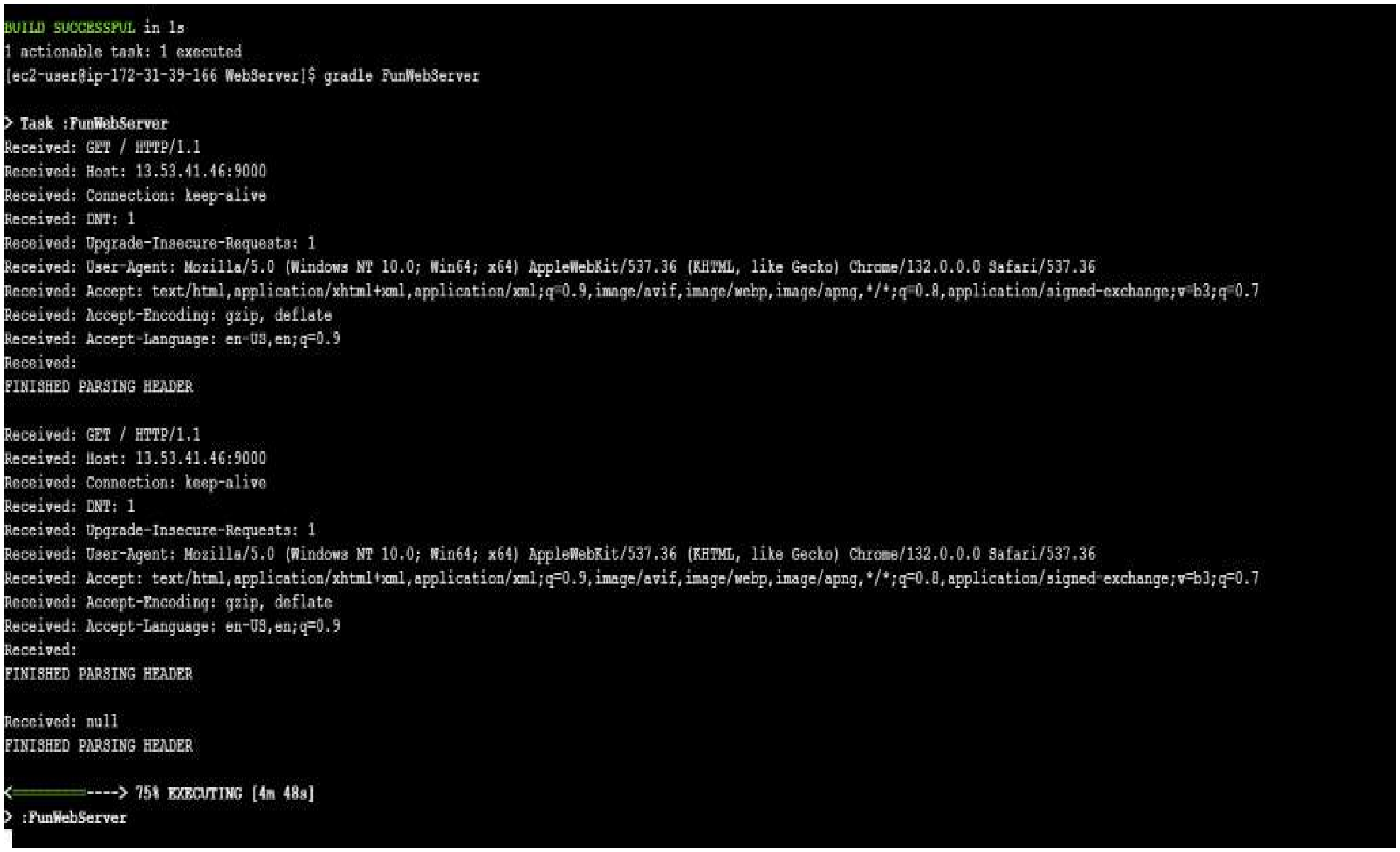


You can make the following GET requests

- **/file/sample.html -- returns the content of the file sample.html**
- **/json -- returns a json of the /random request**
- **/random -- returns index.html**

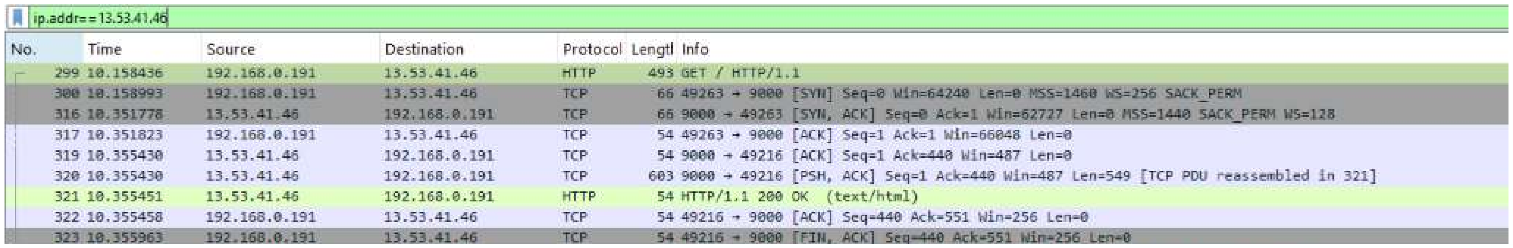
File Structure in www (you can use /file/www/FILENAME):

- index.html
- root.html



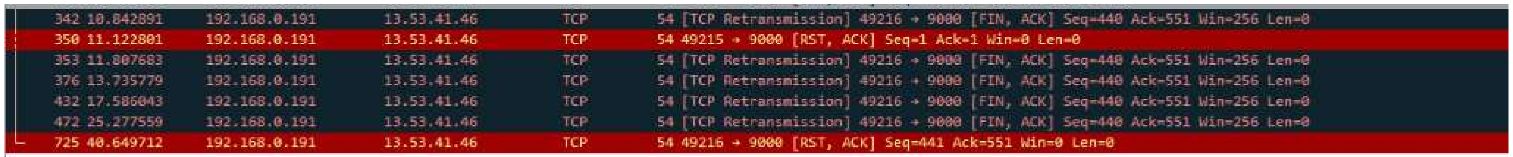
```
BUILD SUCCESSFUL in 1s
1 actionable task: 1 executed
[ec2-user@ip-172-31-39-166 WebServer]$ gradle FunWebServer

> Task :FunWebServer
Received: GET / HTTP/1.1
Received: Host: 13.53.41.46:9000
Received: Connection: keep-alive
Received: DNT: 1
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: GET / HTTP/1.1
Received: Host: 13.53.41.46:9000
Received: Connection: keep-alive
Received: DNT: 1
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: null
FINISHED PARSING HEADER

<=========----> 75% EXECUTING [4m 48s]
> :FunWebServer
```

2.3 Analyze what happens (10 points)



ip.addr==13.53.41.46

No.	Time	Source	Destination	Protocol	Lengtl	Info
299	10.158436	192.168.0.191	13.53.41.46	HTTP	493	GET / HTTP/1.1
300	10.158993	192.168.0.191	13.53.41.46	TCP	66	49263 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
316	10.351778	13.53.41.46	192.168.0.191	TCP	66	9000 → 49263 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1440 SACK_PERM WS=128
317	10.351823	192.168.0.191	13.53.41.46	TCP	54	49263 → 9000 [ACK] Seq=1 Ack=1 Win=66048 Len=0
319	10.355430	13.53.41.46	192.168.0.191	TCP	54	9000 → 49216 [ACK] Seq=1 Ack=440 Win=487 Len=0
320	10.355430	13.53.41.46	192.168.0.191	TCP	603	9000 → 49216 [PSH, ACK] Seq=1 Ack=440 Win=487 Len=549 [TCP PDU reassembled in 321]
321	10.355451	13.53.41.46	192.168.0.191	HTTP	54	HTTP/1.1 200 OK (text/html)
322	10.355458	192.168.0.191	13.53.41.46	TCP	54	49216 → 9000 [ACK] Seq=440 Ack=551 Win=256 Len=0
323	10.355963	192.168.0.191	13.53.41.46	TCP	54	49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0



```
342 10.842891   192.168.0.191   13.53.41.46   TCP   54 [TCP Retransmission] 49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0
350 11.122801   192.168.0.191   13.53.41.46   TCP   54 49215 → 9000 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
353 11.807683   192.168.0.191   13.53.41.46   TCP   54 [TCP Retransmission] 49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0
376 13.735779   192.168.0.191   13.53.41.46   TCP   54 [TCP Retransmission] 49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0
432 17.586043   192.168.0.191   13.53.41.46   TCP   54 [TCP Retransmission] 49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0
472 25.277559   192.168.0.191   13.53.41.46   TCP   54 [TCP Retransmission] 49216 → 9000 [FIN, ACK] Seq=440 Ack=551 Win=256 Len=0
725 40.649712   192.168.0.191   13.53.41.46   TCP   54 49216 → 9000 [RST, ACK] Seq=441 Ack=551 Win=0 Len=0
```

1. What filter did you use? Explain why you chose that filter. *something with port 9000 probably*
    - I used the ip.addr filter to find the communication with my specifc AWS IP which I assumed would only have the traffic related to what I am doing.

2. What happens when you are on the /random page and click the "Random" button? Compare this to refreshing your browser. (You can also use the command line output that the WebServer generates to answer this.) *one goes right to /json the other one first to /random and then /json 2 points*
    - I see that refreshing the page re-calls /random, however, using the "Random" button does not; it instead makes a JSON call to receive a random file.

3. What types of response codes are you able to receive through different requests to your server? *200, 400, 404 should be the ones*
    - 200: normal valid server requests
    - 400: misspelling things like ranom or JSON
    - 404: not sure how to get this. Could this be from me filtering with the IP instead of the port?

4. Explain the response codes you receive and why you get them. *explained the response codes well 2 points*
    - 200: this is the ok response which is telling us that everything was submitted and returned correctly.
    - 400: this came from me misspelling words associated with the API requests.

5. When you do a :9000, take a look at what Wireshark generates as a server response. Are you able to find the data that the server sends back to you? (This should be the "Data" section of your response.) *Should be able to see the data as plain text what is in the HTML file*
    - Yes, it is the HTML file for the display we see on the screen listing the few options we can use.

6. Based on the previous question, explain why HTTPS is now more common than HTTP. *HTTPs is more secure*
    - I would assume HTTPS is more popular because it prevents you from being able to easily see the communication back from the server. You wouldn't be able to just view the HTML file for google as it responded back. This is due to its encryption.

7. In our case - what port does the server listen to for HTTP requests, and is that the most common port for HTTP? *9000 but most common for HTTP is 80*
    - The server is listening for 9000, which we specified. The most common for HTTP is 80 and 443 for HTTPS.

8. Which local port is used when sending different requests to the WebServer? *would be some large port number eg. 13119, 45000 or similar*
    - The local port sending requests is 52377

2.4 Setting up a "real" WebServer (10 points)

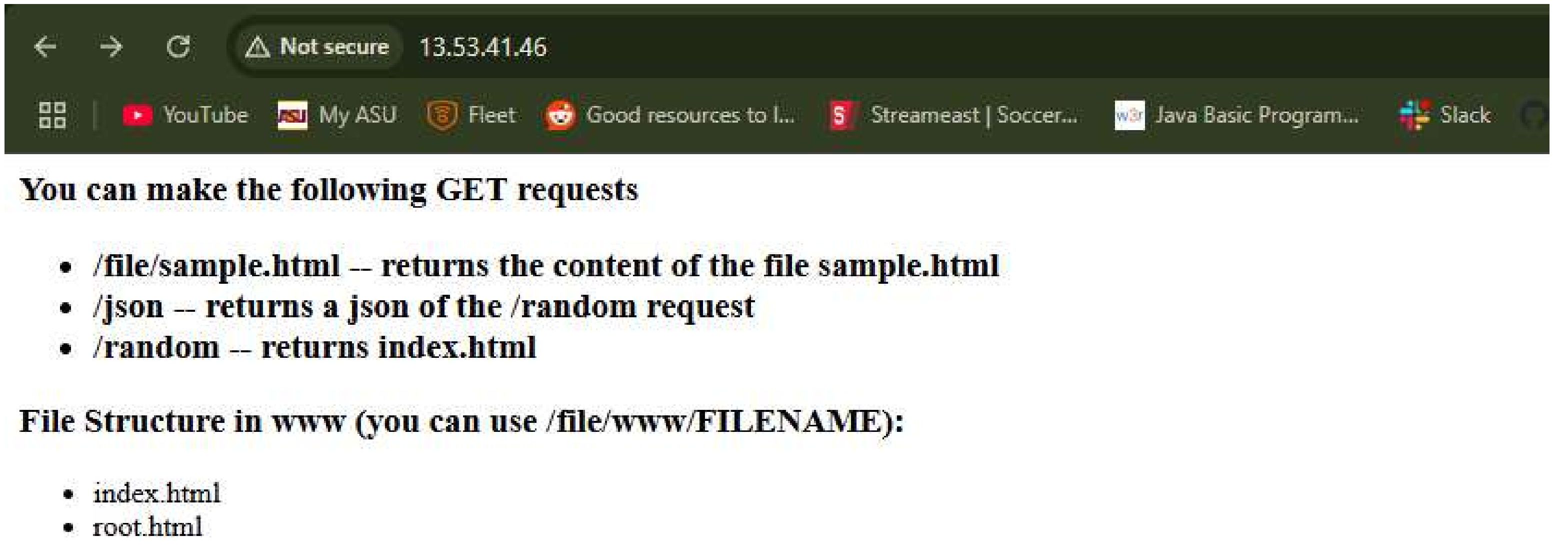
1. What is the URL that you can now use to reach the main page?
    - http://13.53.41.46/

2. Check the traffic to your WebServer. What port is the traffic going to now? Is it the same as before, or is it (and should it) be different? Should be 80 now
    - Everything is mostly to 80, however, there are a couple instances of 9000.

3. Is it still using HTTP, or is it now using HTTPS? Why? still HTTP
    - It is still using HTTP.
    - We did not reconfigure our server to run as HTTPS.

4. Could you change your security settings on AWS now? yes port 9000 could be removed now
   - I don't fully understand the question here. Yes, you could change the security settings on AWS. Things like changing or adding port numbers are possible.

5. Take a screenshot of your web browser, your second machine, and the port number on Wireshark. This should be similar to the screenshot you took before (but also with Wireshark), and add it to your document for this task. Note: If we are unable to see that you reached the WebServer with the "different URL", we will not know if you actually set up the server correctly. Therefore, please make sure that it shows up if you want points for this task.

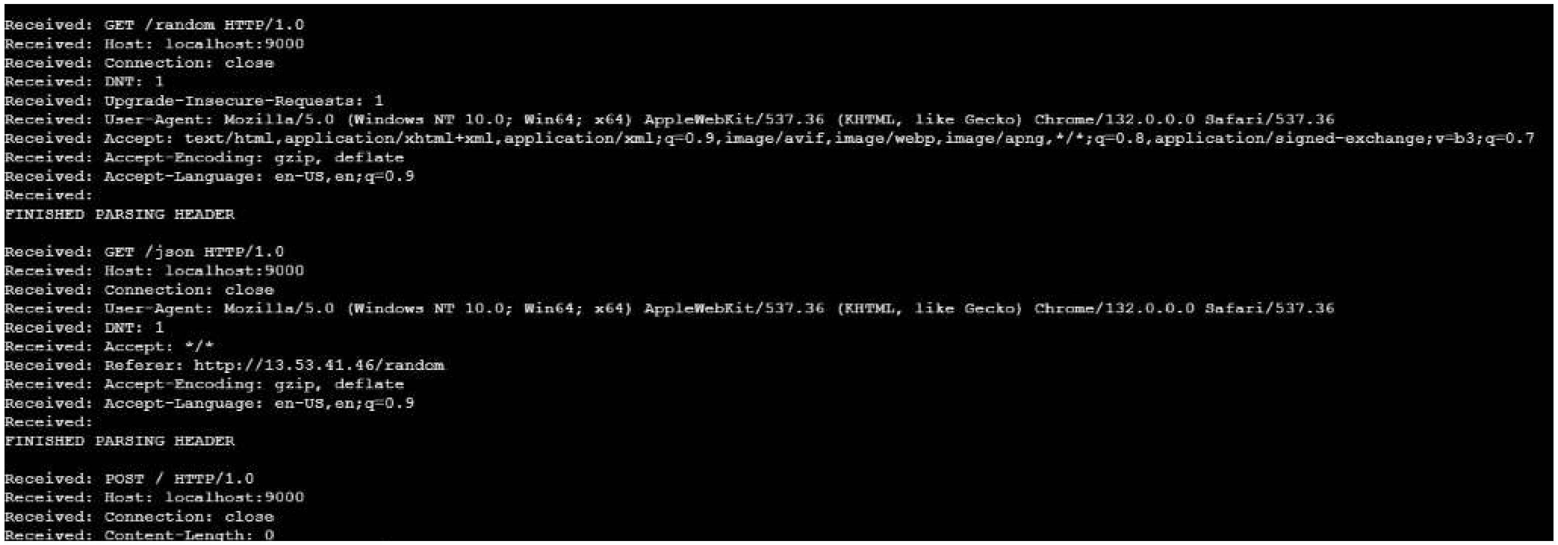


You can make the following GET requests

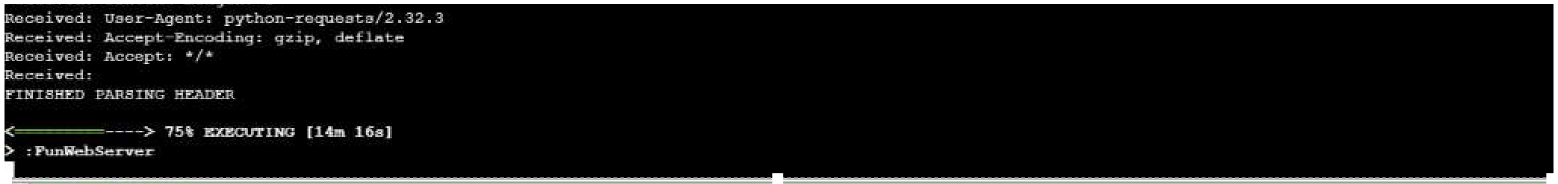
- /file/sample.html -- returns the content of the file sample.html
- /json -- returns a json of the /random request
- /random -- returns index.html

File Structure in www (you can use /file/www/FILENAME):

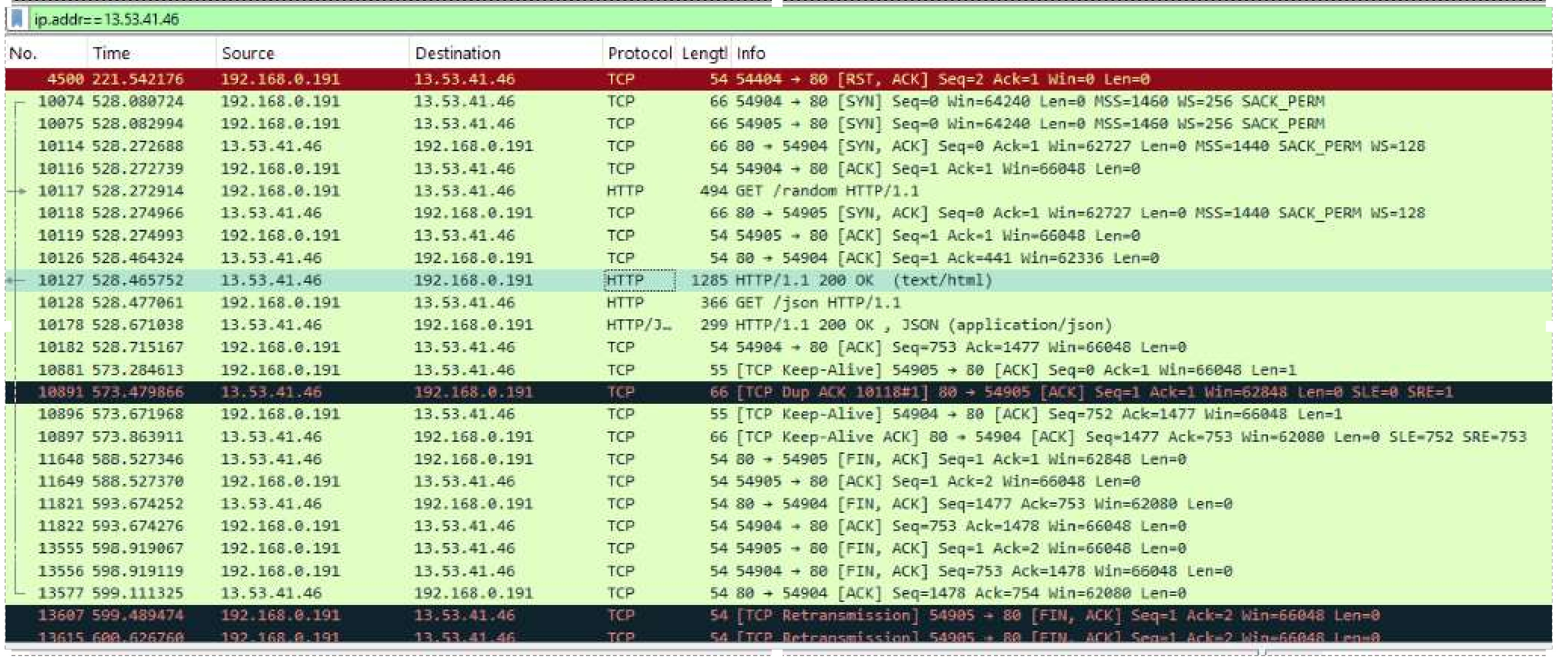
- index.html
- root.html



```
Received: GET /random HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: DNT: 1
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: GET /json HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Received: DNT: 1
Received: Accept: */*
Received: Referer: http://13.53.41.46/random
Received: Accept-Encoding: gzip, deflate
Received: Accept-Language: en-US,en;q=0.9
Received:
FINISHED PARSING HEADER

Received: POST / HTTP/1.0
Received: Host: localhost:9000
Received: Connection: close
Received: Content-Length: 0
```



```
Received: User-Agent: python-requests/2.32.3
Received: Accept-Encoding: gzip, deflate
Received: Accept: */*
Received:
FINISHED PARSING HEADER

<===========---> 75% EXECUTING [14m 16s]
> :FunWebServer
```



ip.addr==13.53.41.46

No.	Time	Source	Destination	Protocol	Lengtl	Info
4500	221.542176	192.168.0.191	13.53.41.46	TCP	54	54404 → 80 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
10074	528.080724	192.168.0.191	13.53.41.46	TCP	66	54904 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10075	528.082994	192.168.0.191	13.53.41.46	TCP	66	54905 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10114	528.272688	13.53.41.46	192.168.0.191	TCP	66	80 → 54904 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1440 SACK_PERM WS=128
10116	528.272739	192.168.0.191	13.53.41.46	TCP	54	54904 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
10117	528.272914	192.168.0.191	13.53.41.46	HTTP	494	GET /random HTTP/1.1
10118	528.274966	13.53.41.46	192.168.0.191	TCP	66	80 → 54905 [SYN, ACK] Seq=0 Ack=1 Win=62727 Len=0 MSS=1440 SACK_PERM WS=128
10119	528.274993	192.168.0.191	13.53.41.46	TCP	54	54905 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
10126	528.464324	13.53.41.46	192.168.0.191	TCP	54	80 → 54904 [ACK] Seq=1 Ack=441 Win=62336 Len=0
10127	528.465752	13.53.41.46	192.168.0.191	HTTP	1285	HTTP/1.1 200 OK (text/html)
10128	528.477061	192.168.0.191	13.53.41.46	HTTP	366	GET /json HTTP/1.1
10178	528.671038	13.53.41.46	192.168.0.191	HTTP/J…	299	HTTP/1.1 200 OK , JSON (application/json)
10182	528.715167	192.168.0.191	13.53.41.46	TCP	54	54904 → 80 [ACK] Seq=753 Ack=1477 Win=66048 Len=0
10881	573.284613	192.168.0.191	13.53.41.46	TCP	55	[TCP Keep-Alive] 54905 → 80 [ACK] Seq=0 Ack=1 Win=66048 Len=1
10891	573.479866	13.53.41.46	192.168.0.191	TCP	66	[TCP Dup ACK 10118#1] 80 → 54905 [ACK] Seq=1 Ack=1 Win=62848 Len=0 SLE=0 SRE=1
10896	573.671968	192.168.0.191	13.53.41.46	TCP	55	[TCP Keep-Alive] 54904 → 80 [ACK] Seq=752 Ack=1477 Win=66048 Len=1
10897	573.863911	13.53.41.46	192.168.0.191	TCP	66	[TCP Keep-Alive ACK] 80 → 54904 [ACK] Seq=1477 Ack=753 Win=62080 Len=0 SLE=752 SRE=753
11648	588.527346	13.53.41.46	192.168.0.191	TCP	54	80 → 54905 [FIN, ACK] Seq=1 Ack=1 Win=62848 Len=0
11649	588.527370	192.168.0.191	13.53.41.46	TCP	54	54905 → 80 [ACK] Seq=1 Ack=2 Win=66048 Len=0
11821	593.674252	13.53.41.46	192.168.0.191	TCP	54	80 → 54904 [FIN, ACK] Seq=1477 Ack=753 Win=62080 Len=0
11822	593.674276	192.168.0.191	13.53.41.46	TCP	54	54904 → 80 [ACK] Seq=753 Ack=1478 Win=66048 Len=0
13555	598.919067	192.168.0.191	13.53.41.46	TCP	54	54905 → 80 [FIN, ACK] Seq=1 Ack=2 Win=66048 Len=0
13556	598.919119	192.168.0.191	13.53.41.46	TCP	54	54904 → 80 [FIN, ACK] Seq=753 Ack=1478 Win=66048 Len=0
13577	599.111325	13.53.41.46	192.168.0.191	TCP	54	80 → 54904 [ACK] Seq=1478 Ack=754 Win=62080 Len=0
13607	599.489474	192.168.0.191	13.53.41.46	TCP	54	[TCP Retransmission] 54905 → 80 [FIN, ACK] Seq=1 Ack=2 Win=66048 Len=0
13615	600.626760	192.168.0.191	13.53.41.46	TCP	54	[TCP Retransmission] 54905 → 80 [FIN, ACK] Seq=1 Ack=2 Win=66048 Len=0

2.5 Setting up HTTPS (5 points)

- *Not attempted*

2.6.1 Multiply (5 points)

- I decided to put a try block around the user input being parsed to int. That way, if the program is unable to parse the user input into a Integer, then it will trigger a boolean variable to false. From there the code will move to a internal If/else statement that executes dependent on the value of the boolean variable.
- I chose a 406 error code to identify to the user that the input they gave does not match any internal supported media.

2.6.2 GitHub (9 points)