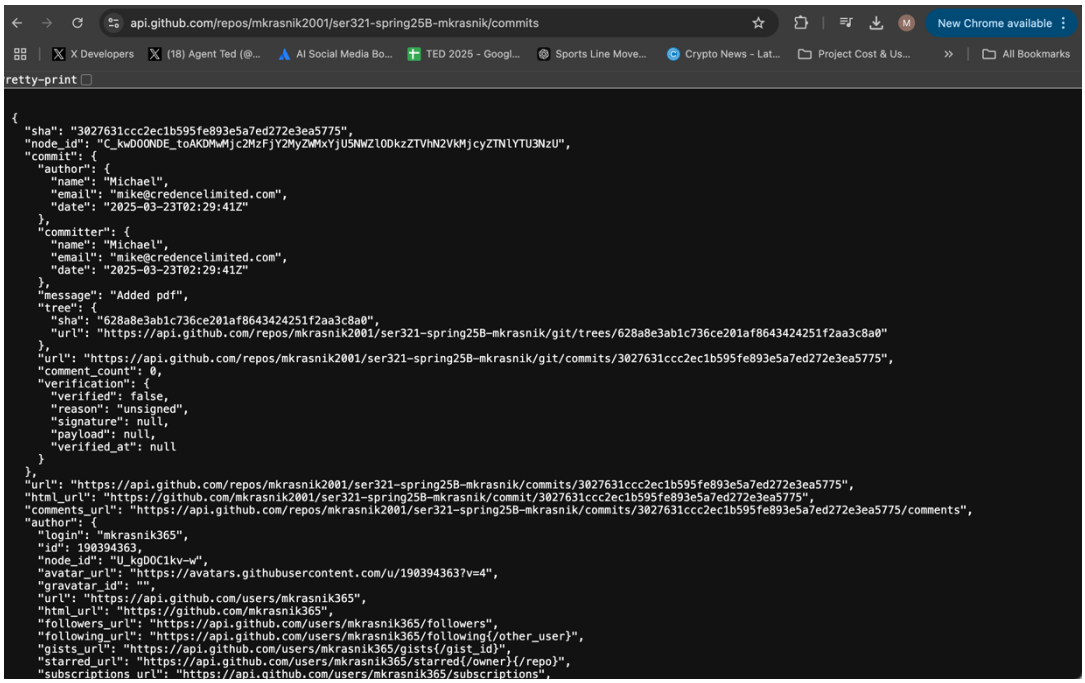
Assignment 2 SER 321 – Michael Krasnik {mkrasnik}

https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik

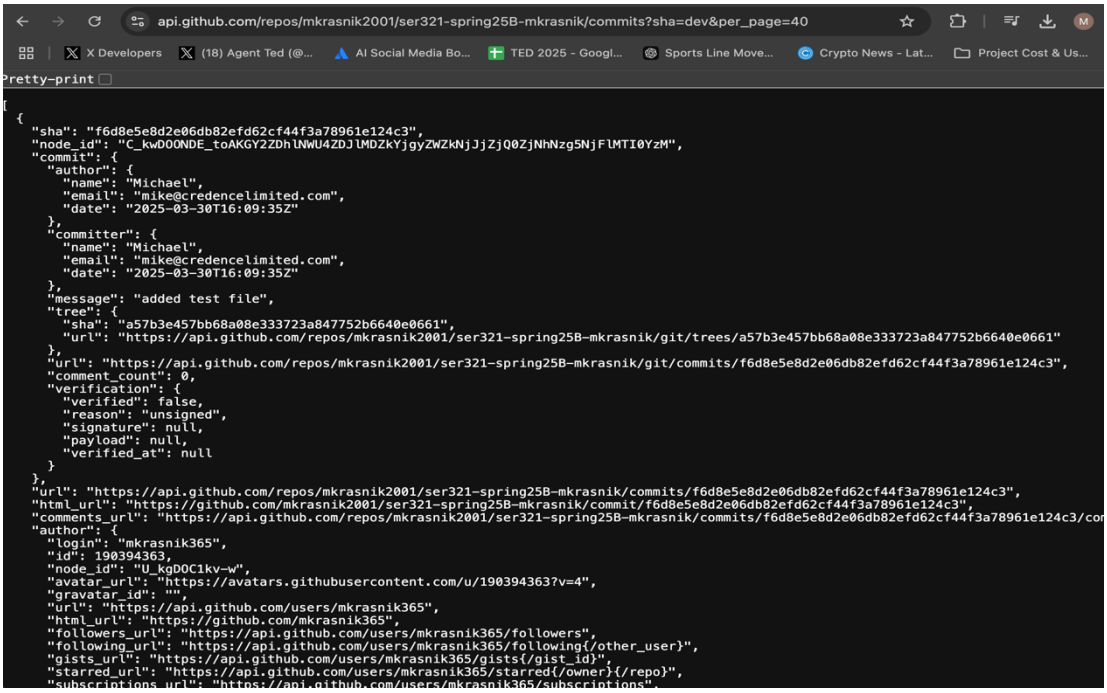
Task 1

https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits




```
{
  "sha": "3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "node_id": "C_kwDOONDE_toAKDMwMjc2MzFjY2MyZWMxYjU5NWZlODkzZTVhN2VkMjcyZTNlYTU3NzU",
  "commit": {
    "author": {
      "name": "Michael",
      "email": "mike@credencelimited.com",
      "date": "2025-03-23T02:29:41Z"
    },
    "committer": {
      "name": "Michael",
      "email": "mike@credencelimited.com",
      "date": "2025-03-23T02:29:41Z"
    },
    "message": "Added pdf",
    "tree": {
      "sha": "628a8e3ab1c736ce201af8643424251f2aa3c8a0",
      "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/trees/628a8e3ab1c736ce201af8643424251f2aa3c8a0"
    },
    "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
    "comment_count": 0,
    "verification": {
      "verified": false,
      "reason": "unsigned",
      "signature": null,
      "payload": null,
      "verified_at": null
    }
  },
  "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/commit/3027631ccc2ec1b595fe893e5a7ed272e3ea5775",
  "comments_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/3027631ccc2ec1b595fe893e5a7ed272e3ea5775/comments",
  "author": {
    "login": "mkrasnik365",
    "id": 190394363,
    "node_id": "U_kgDOC1kv-w",
    "avatar_url": "https://avatars.githubusercontent.com/u/190394363?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mkrasnik365",
    "html_url": "https://github.com/mkrasnik365",
    "followers_url": "https://api.github.com/users/mkrasnik365/followers",
    "following_url": "https://api.github.com/users/mkrasnik365/following{/other_user}",
    "gists_url": "https://api.github.com/users/mkrasnik365/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mkrasnik365/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/mkrasnik365/subscriptions",
```


https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits?sha=dev&per_page=40




```
[
  {
    "sha": "f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "node_id": "C_kwDOONDE_toAKGY2ZDhlNWU4ZDJlMDZkYjgyZWZkNjJjZjQ0ZjNhNzg5NjFlMTI0YzM",
    "commit": {
      "author": {
        "name": "Michael",
        "email": "mike@credencelimited.com",
        "date": "2025-03-30T16:09:35Z"
      },
      "committer": {
        "name": "Michael",
        "email": "mike@credencelimited.com",
        "date": "2025-03-30T16:09:35Z"
      },
      "message": "added test file",
      "tree": {
        "sha": "a57b3e457bb68a08e333723a847752b6640e0661",
        "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/trees/a57b3e457bb68a08e333723a847752b6640e0661"
      },
      "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/git/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
      "comment_count": 0,
      "verification": {
        "verified": false,
        "reason": "unsigned",
        "signature": null,
        "payload": null,
        "verified_at": null
      }
    },
    "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/commit/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3",
    "comments_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/commits/f6d8e5e8d2e06db82efd62cf44f3a78961e124c3/com
    "author": {
      "login": "mkrasnik365",
      "id": 190394363,
      "node_id": "U_kgDOC1kv-w",
      "avatar_url": "https://avatars.githubusercontent.com/u/190394363?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/mkrasnik365",
      "html_url": "https://github.com/mkrasnik365",
      "followers_url": "https://api.github.com/users/mkrasnik365/followers",
      "following_url": "https://api.github.com/users/mkrasnik365/following{/other_user}",
      "gists_url": "https://api.github.com/users/mkrasnik365/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/mkrasnik365/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/mkrasnik365/subscriptions",
```

https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/pulls?state=open


```
[
 {
   "url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/pulls/1",
   "id": 2427622128,
   "node_id": "PR_kwDOONDE_s6QspLw",
   "html_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1",
   "diff_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1.diff",
   "patch_url": "https://github.com/mkrasnik2001/ser321-spring25B-mkrasnik/pull/1.patch",
   "issue_url": "https://api.github.com/repos/mkrasnik2001/ser321-spring25B-mkrasnik/issues/1",
   "number": 1,
   "state": "open",
   "locked": false,
   "title": "Added Test Txt File (WIP PR)",
   "user": {
     "login": "mkrasnik2001",
     "id": 158517273,
     "node_id": "U_kgDOCXLIGQ",
     "avatar_url": "https://avatars.githubusercontent.com/u/158517273?v=4",
     "gravatar_id": "",
     "url": "https://api.github.com/users/mkrasnik2001",
     "html_url": "https://github.com/mkrasnik2001",
     "followers_url": "https://api.github.com/users/mkrasnik2001/followers",
     "following_url": "https://api.github.com/users/mkrasnik2001/following{/other_user}",
     "gists_url": "https://api.github.com/users/mkrasnik2001/gists{/gist_id}",
     "starred_url": "https://api.github.com/users/mkrasnik2001/starred{/owner}{/repo}",
     "subscriptions_url": "https://api.github.com/users/mkrasnik2001/subscriptions",
     "organizations_url": "https://api.github.com/users/mkrasnik2001/orgs",
     "repos_url": "https://api.github.com/users/mkrasnik2001/repos",
     "events_url": "https://api.github.com/users/mkrasnik2001/events{/privacy}",
     "received_events_url": "https://api.github.com/users/mkrasnik2001/received_events",
     "type": "User",
     "user_view_type": "public",
     "site_admin": false
   },
   "body": null,
   "created_at": "2025-03-30T16:13:52Z",
   "updated_at": "2025-03-30T16:13:52Z",
   "closed_at": null,
   "merged_at": null,
   "merge_commit_sha": "8a4790d6acd6960d187092b3ed1071333158b604",
   "assignee": null,
   "assignees": [

   ],
   "requested_reviewers": [
```

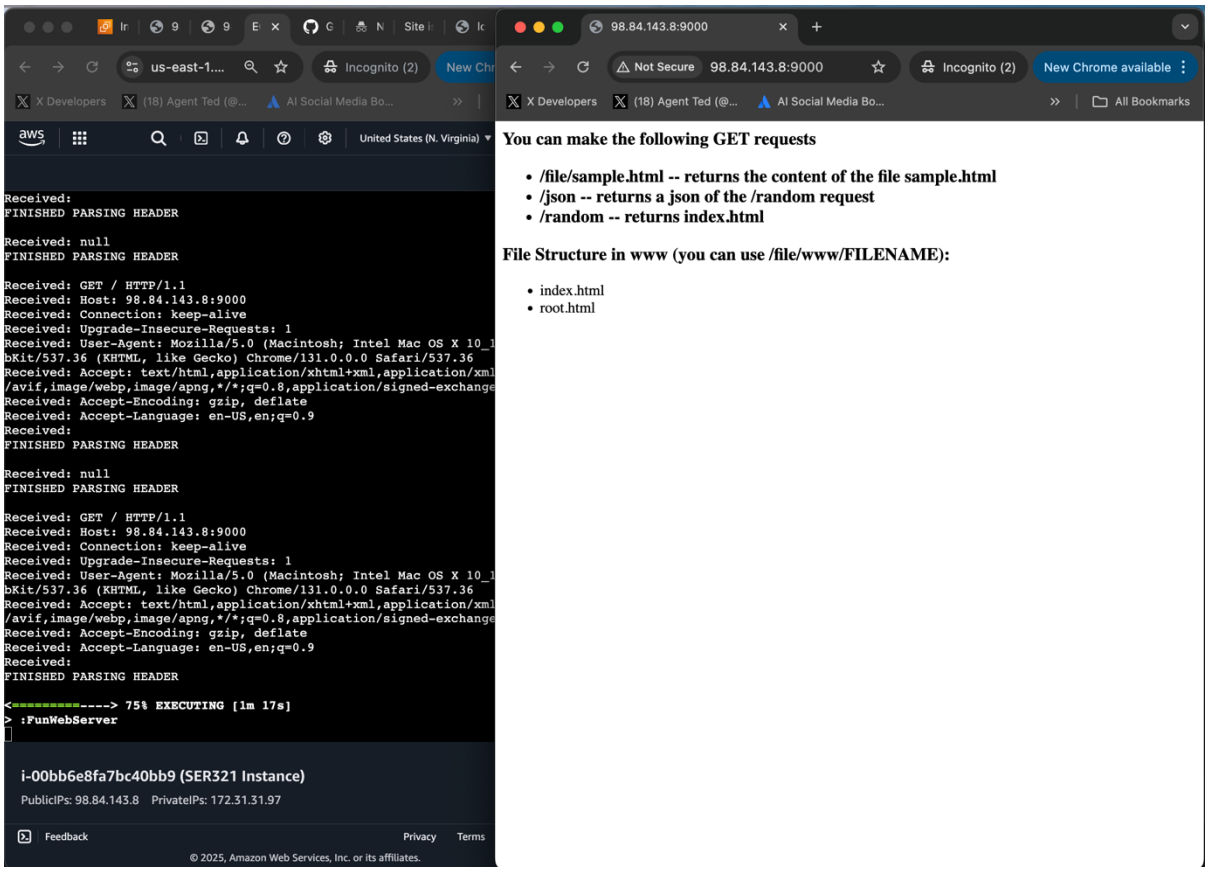

1. The first API call used was a GET request to view all the commits of a repo on the default branch. There are no query parameters used in this request, just the /commit endpoint was used. Doc link: https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28 . The second API call was also a GET request to view the commits of a specific branch while limiting the amount of commits per page to 40. The query parameters used was specifying the branch as well as the page limit. Doc link: https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28 . The third API call used was another GET request to the pull requests endpoint which returns a list of PRs that are still open. The query parameter used was specifying the state of the PR (open in this case). Doc link: https://docs.github.com/en/rest/pulls/pulls?apiVersion=2022-11-28 .

2. Stateless communication refers to communication between the client and server where each request is independent. The server does not maintain any knowledge of past requests by the client and whenever a client sends a request, it needs to provide all the required info within the same request for it to be fulfilled. In

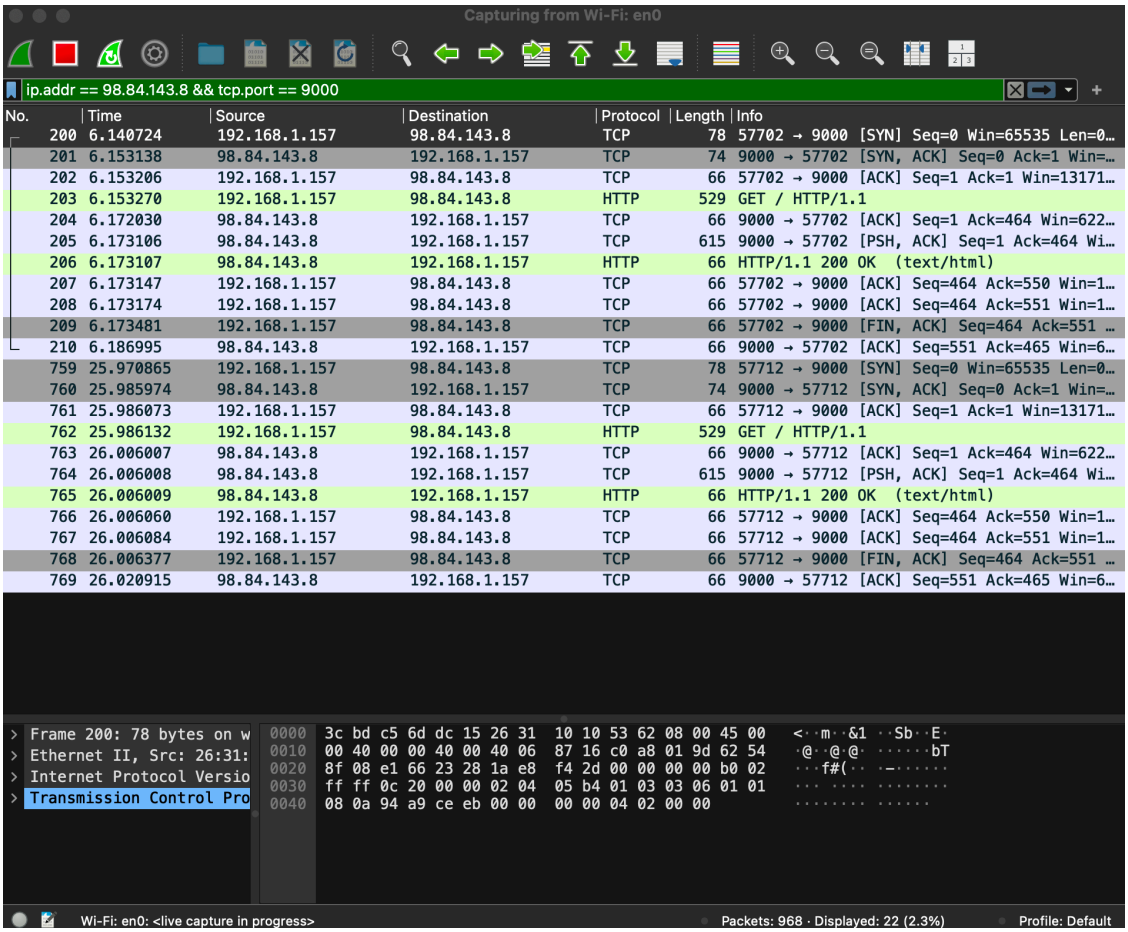
other words, there is no “state” or context maintained of that logical connection between the client and the server. A stateless communication protocol example is HTTP. Stateful communication is the opposite, this is when the server does store context and states of the conversation with the client and during requests the client doesn’t need to provide info that was provided earlier (like authentication). An example protocol could be SSH.

Task 2

2.2 Running a Simple Java WebServer (10 points)



2.3 Analyze what happens (10 points)

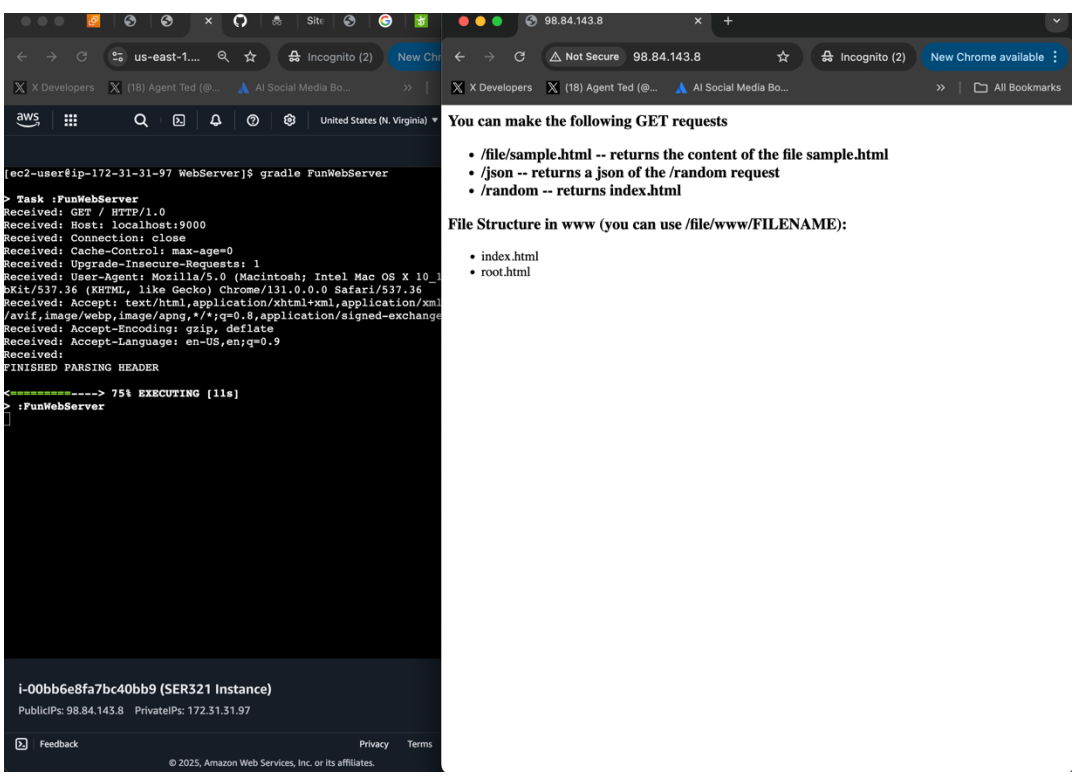


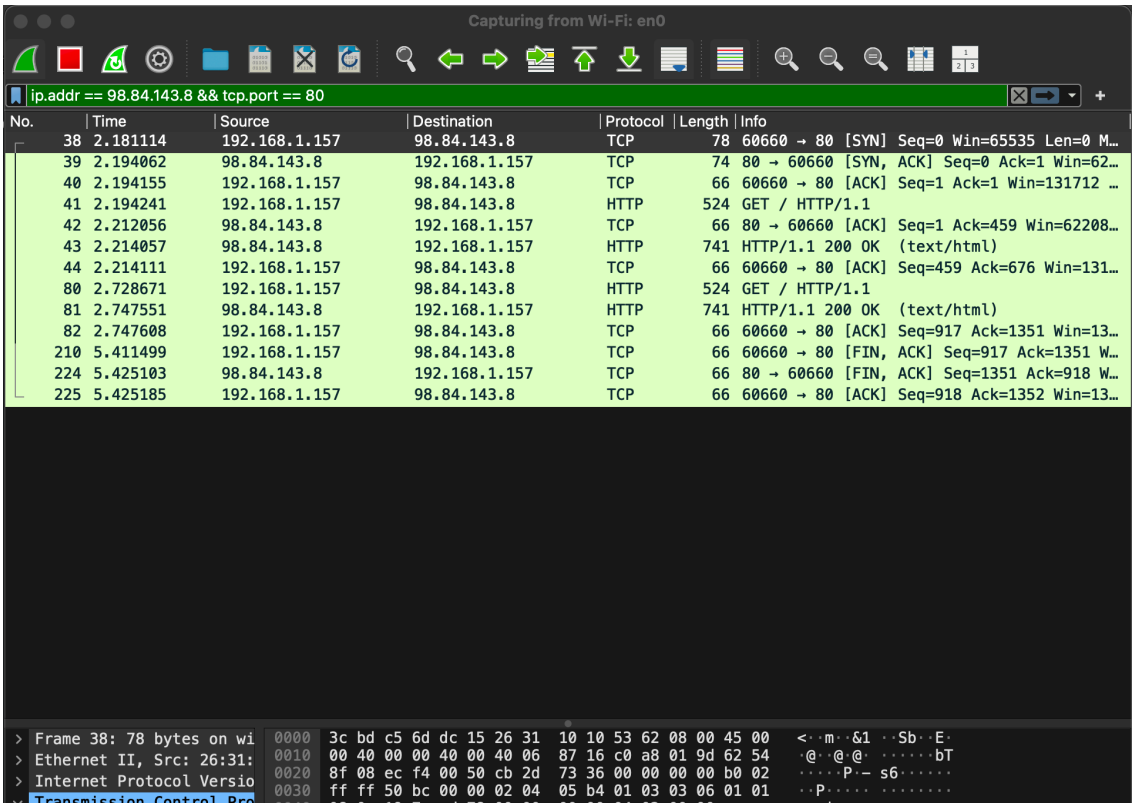
1. On Wireshark I used the following filter: ip.addr == 98.84.143.8 && tcp.port == 9000. The ip address filter isolates traffic to and from my EC2 instance and the port filter isolates only TCP traffic to 9000 which is the port that the FunWebServer app was listening on. HTTP uses TCP as it's transfer layer protocol.
2. When we click the "Random" button on the /random page the client sends a new HTTP GET request to the server that is read, processed and shows a new image, however if we refresh the browser, the are more frames coming through because the client first sends a request to load the /random page and then calls the random button to load an image effectively doing the same thing as pressing the button but with an extra step at the beginning when loading the page.
3. The most common types of response codes are 200 which means the request was processed successfully, 400 level codes which means the client did something wrong with the request and 500 level codes where the error is on the server side.
4. The response codes I see at the moment are 200, which means the request was processed successfully and 400 which is a "Bad Request" and means the server didn't understand

the request. When I went to the /json url I got a 400 response code which shows that the server didn't know what to do with this request.

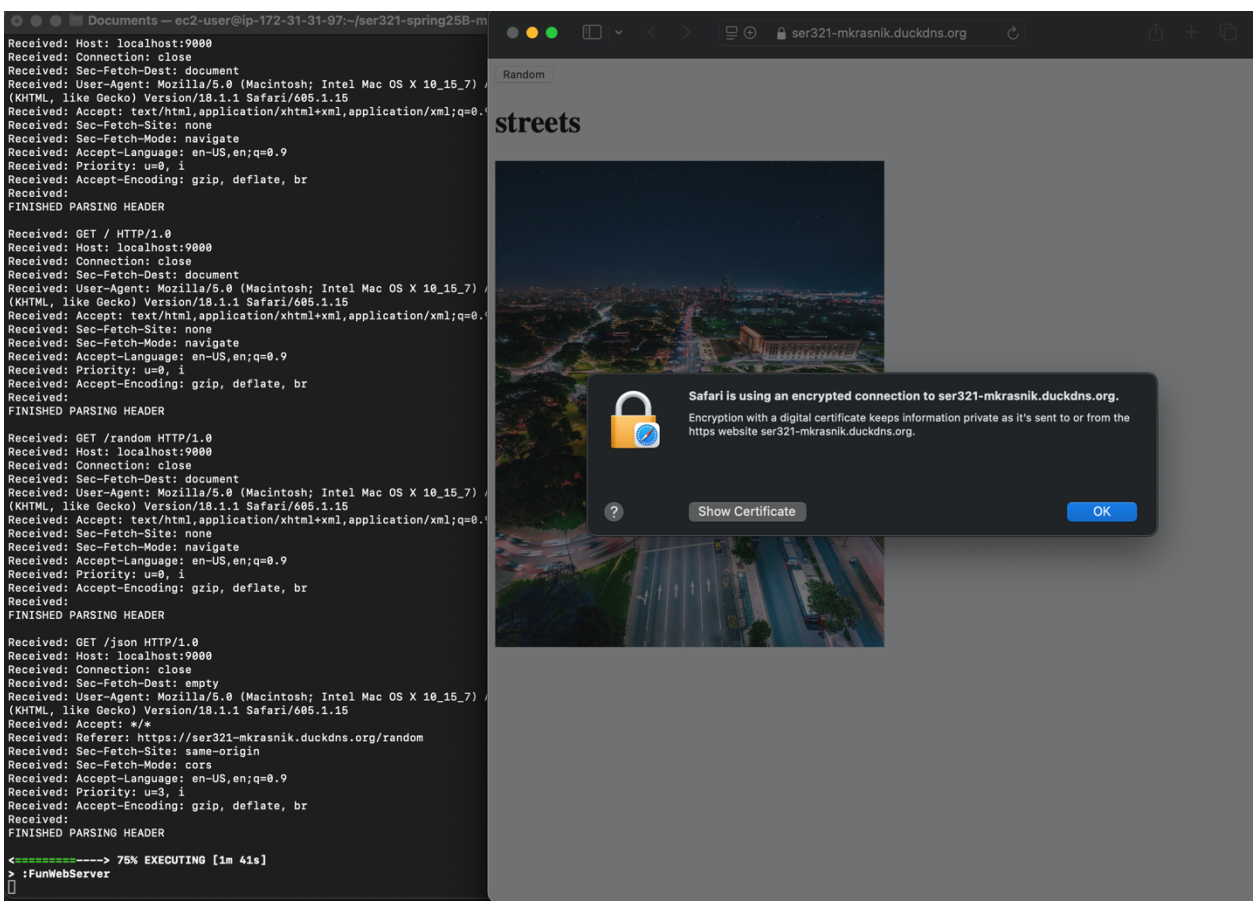
5. Yes I can see the data that's sent back. When you click on the info section you can see the JSON returned with the data (in the /random example it's links to jpg files).
6. HTTPS is more common, because you won't be able to "read" the data if you capture the packets coming in. This is because HTTPS is encrypted and if a malicious node intercepts the packets they still won't be able to understand the data because it'll be encrypted.
7. Our server by default listens to 9000, this is not a common HTTP port. The common HTTP port is usually port 80.
8. The OS assigns temporary ports to use when sending outgoing requests and change all the time. In our case I see ports 59180, 59230 and 59179 used on the local side for outgoing requests.

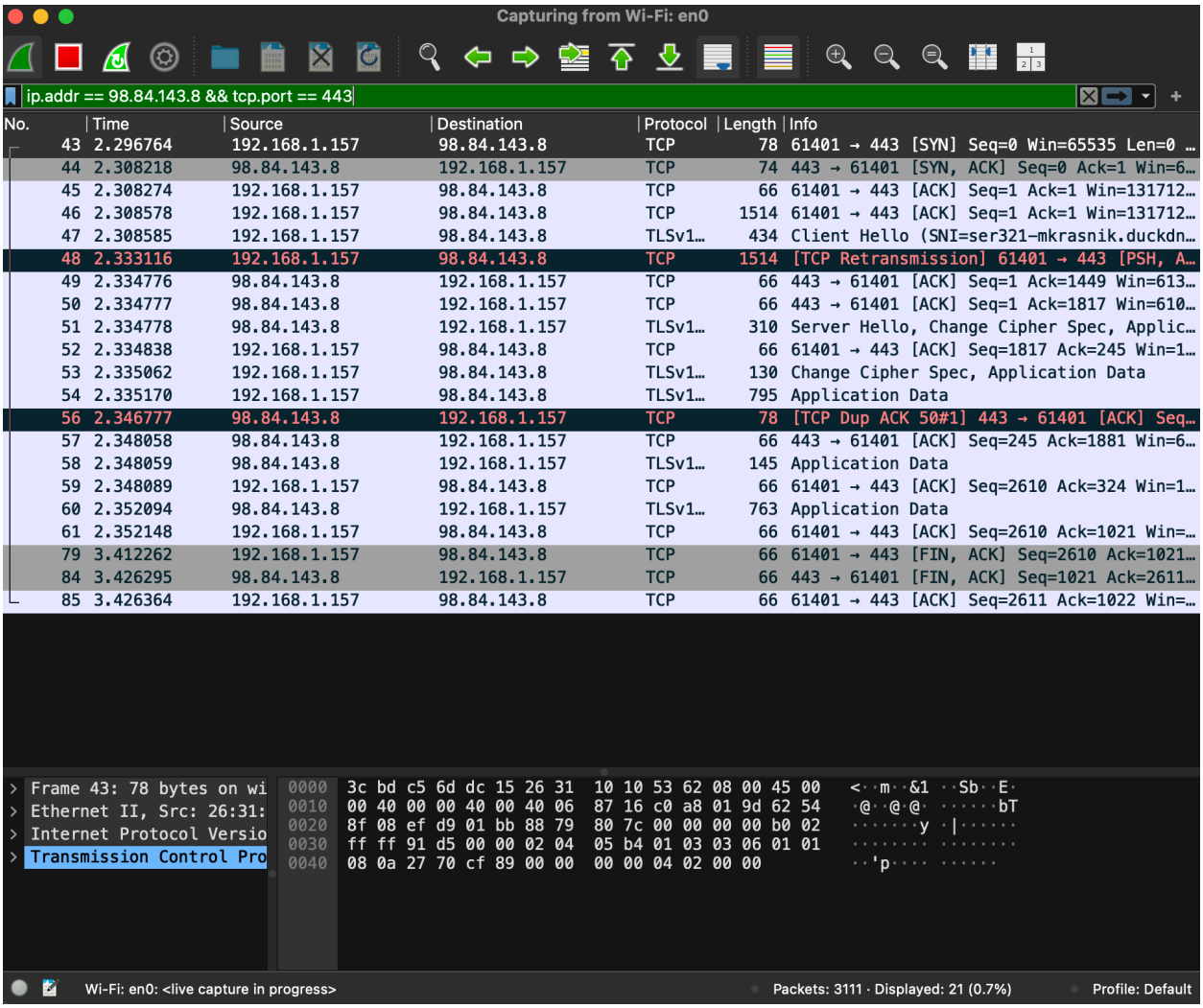
2.4 Setting up a "real" WebServer (10 points)

1. The URL is http://98.84.143.8
2. Traffic is now going to port 80. It is not the same as before (before it was 9000) and yes it should be different.
3. We're still using HTTP because the Transport Layer Security section in the conf file is commented out and therefore nginx is using the normal port 80 and only serving HTTP.
4. Yes you can, because in the ruleset we still have port 80 to allow for inbound TCP traffic, but we don't need anymore the 8000-9999 range rule for inbound TCP traffic. It's a good security practice to update this to avoid exploitation of these ports.
5. 



2.5 Setting up HTTPS (5 points)





1. Traffic is now going through the standard TCP port 443.
2. No all I can see is the TCP payload and the packet data is basically gibberish and encrypted.

2.6.4

Ser321-mkrasnik.duckdns.org

(Posted on #server channel as well)