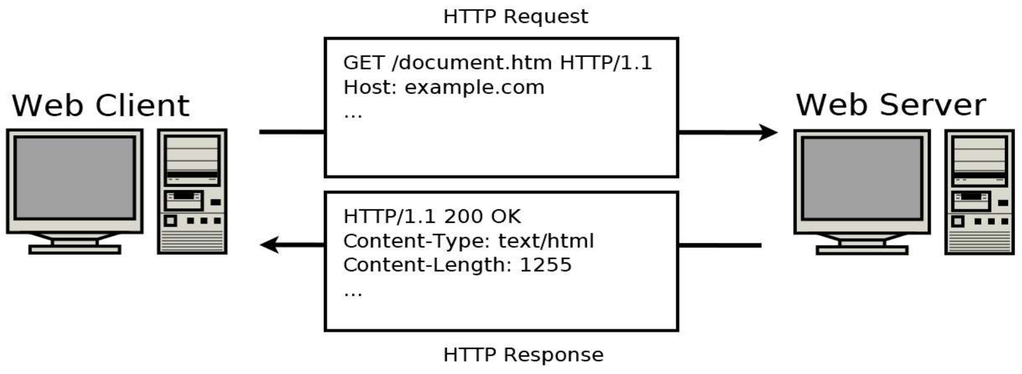
1.

- **Name:** Anoop Shivayogi (W1648523)
- **Assignment**: Programming assignment 1
- **Description**:
    - Simple HTTP web server implementation using Java. The web server responds to client requests, which will be parsed according to the valid HTTP request format rules and a corresponding response will be produced (in this case, SCU Homepage).
    - When the client makes a request to the server and the connection is accepted. The application is multithreaded, it will spawn a new thread to handle the request. the web server reads and parses the request. If no specific file names are mentioned in the request, the server will load the file index.html.
    - The web server currently supports images of type jpeg/jpg, gif, ico, png, and files of type txt/html is supported.
    - Web Server handles HTTP response codes 200, 400, 403 and 404. If the file is not found, a 404 error is returned. If the file is present and accessible, the HTTP OK message is returned (HTTP code 200) with the contents of the file.
    - Supported Response Headers: Response code, Content-Type, Content-Length, Date.



(Image Source: www.google.com - Webserver and web client)

- **List of files submitted:**
    - Readme.pdf
    - Server.java
    - index_files (Folder containing images and supporting .css/.js files of SCU homepage)
    - index.html (HTML file containing the SCU homepage)
    - Makefile to build and run the web server (Java Runtime environment must be preinstalled. to install on ubuntu/Debian Linux: Sudo apt install openjdk-17-jdk-headlessess)
    - Screenshots (A folder containing all the screenshots used in this pdf)

- **Instructions to run the program:**

METHOD 1 (Makefile):

- Open the terminal, switch the directory to the Webserver_Anoop file directory > run the Makefile in the terminal using the following two commands:
  - make build // Compiles the java code
  - make start // Starts the webserver at port 9000

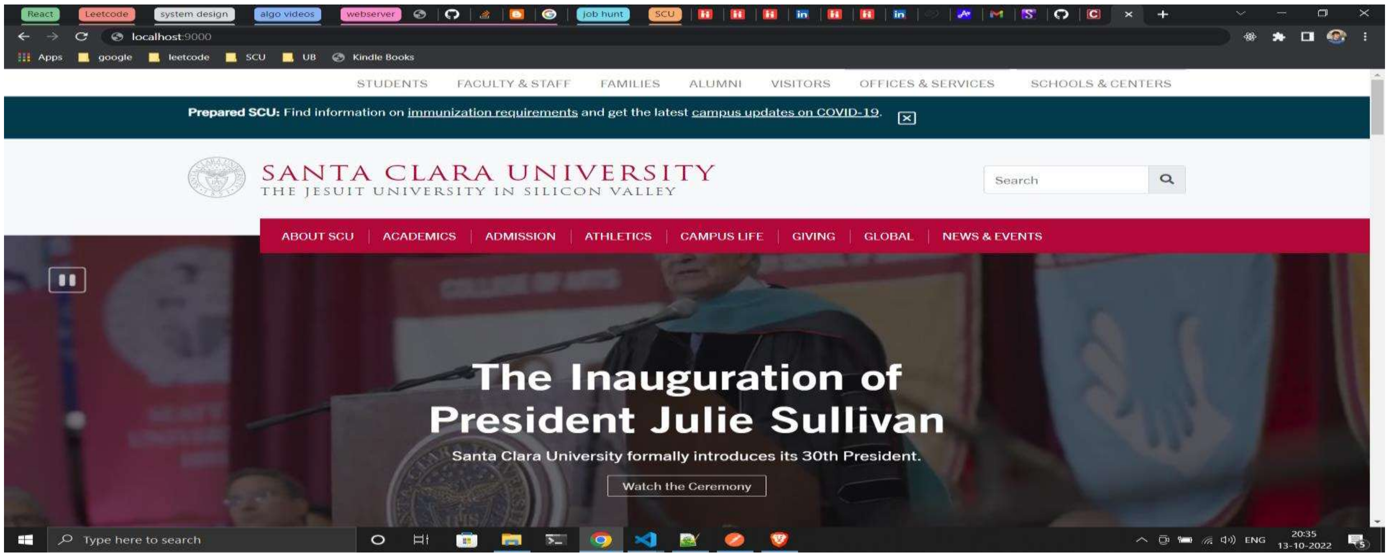
NOTE: To change the port, use method 2

METHOD 2 (Manual):

- Open the terminal, switch the directory to the Webserver_Anoop file directory.
- For compilation, run the command: javac Server.java
- To run the program, run the command: java Server -document_root / -port 9000
  - **Document_root <path/to/document_root> can be provided along with -port <number>.**
- The webpage can be requested via the browser by typing in http://localhost:9000 or the port number specified.
- **The same webpage can be obtained via HTTP GET request calls made via Client software like postman.**

- **Logs and Screenshots:**

1. Client requesting address localhost:9000 via chrome browser and server by default built a response and sent back index.html since no specific file was requested.

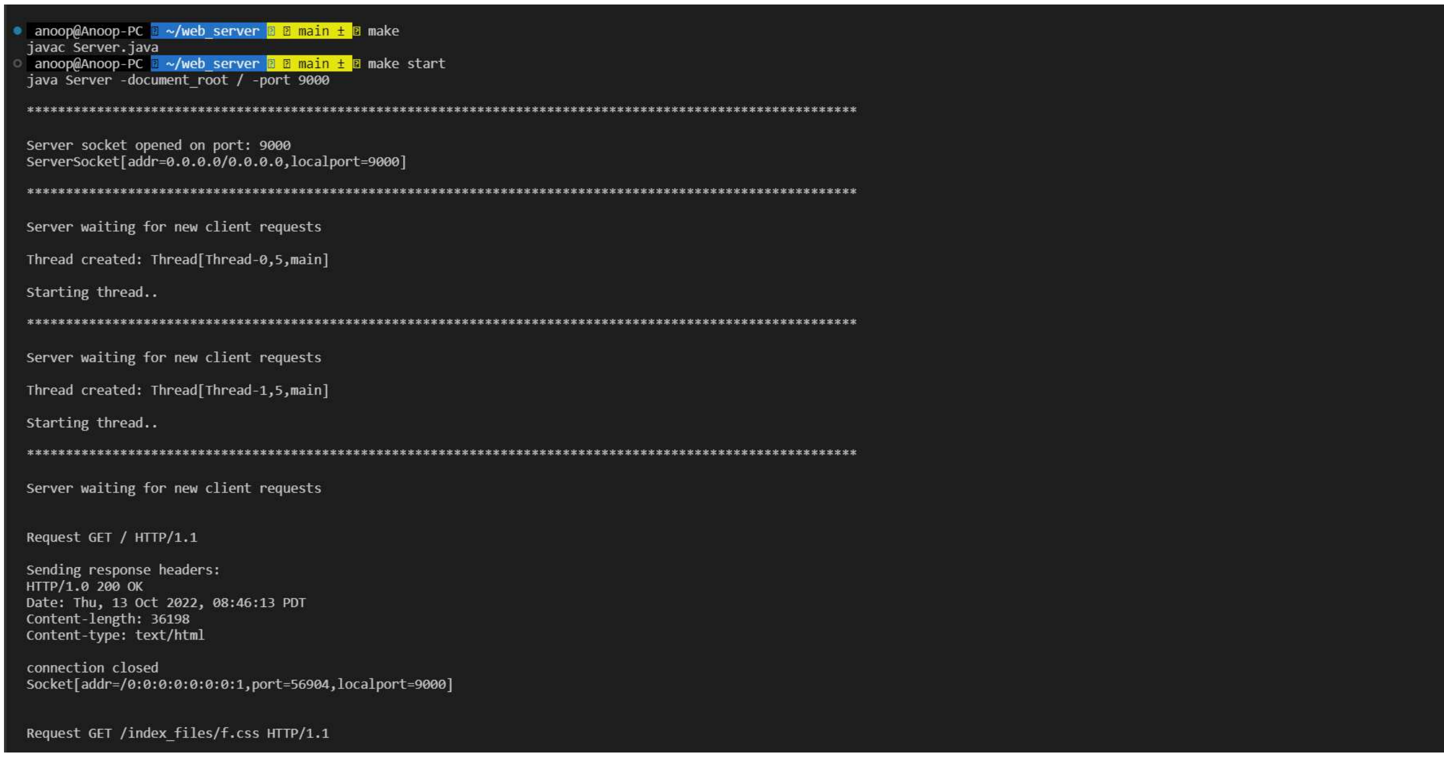


Prepared SCU: Find information on immunization requirements and get the latest campus updates on COVID-19.

The Inauguration of President Julie Sullivan

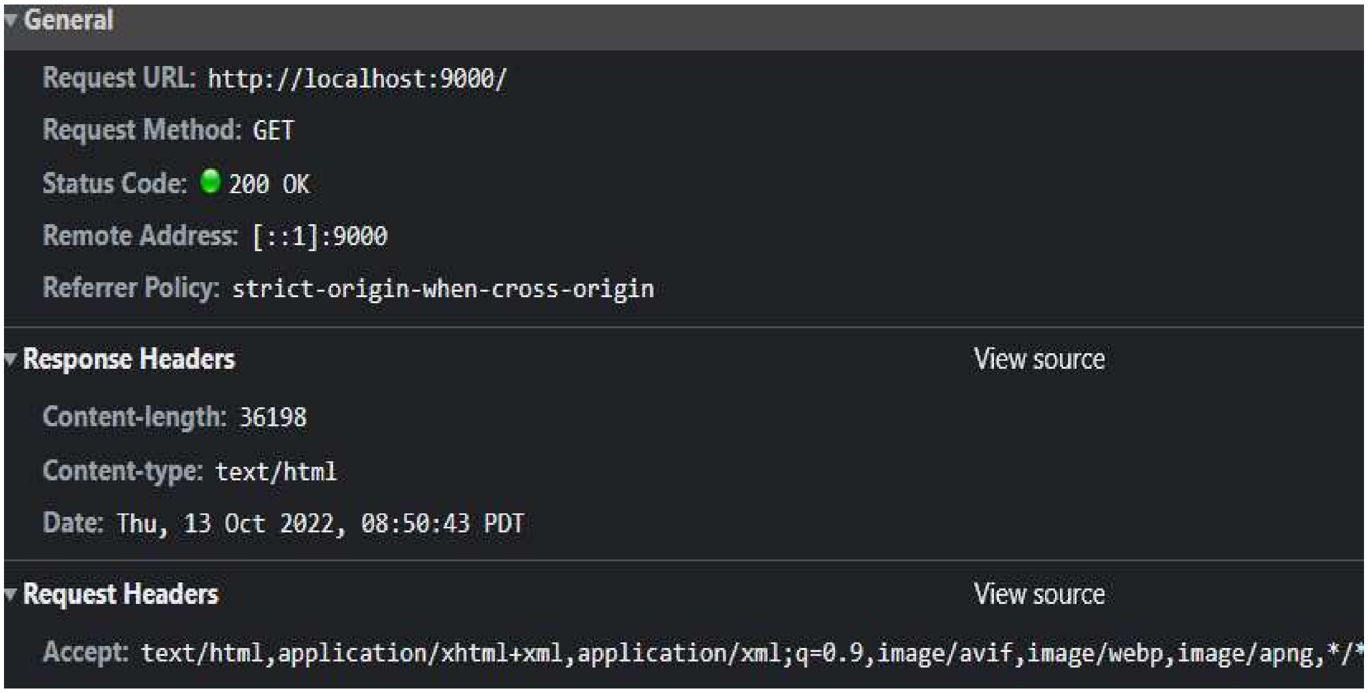
Santa Clara University formally introduces its 30th President.

2. Sample logs screenshot, multiple threads are being spawned for each request from the client side -



```
 anoop@Anoop-PC  ~/web_server   main ±  make
 javac Server.java
 anoop@Anoop-PC  ~/web_server   main ±  make start
 java Server -document_root / -port 9000

 ************************************************************************************************************

 Server socket opened on port: 9000
 ServerSocket[addr=0.0.0.0/0.0.0.0,localport=9000]

 ************************************************************************************************************

 Server waiting for new client requests

 Thread created: Thread[Thread-0,5,main]

 Starting thread..

 ************************************************************************************************************

 Server waiting for new client requests

 Thread created: Thread[Thread-1,5,main]

 Starting thread..

 ************************************************************************************************************

 Server waiting for new client requests


 Request GET / HTTP/1.1

 Sending response headers:
 HTTP/1.0 200 OK
 Date: Thu, 13 Oct 2022, 08:46:13 PDT
 Content-length: 36198
 Content-type: text/html

 connection closed
 Socket[addr=/0:0:0:0:0:0:0:1,port=56904,localport=9000]


 Request GET /index_files/f.css HTTP/1.1
```

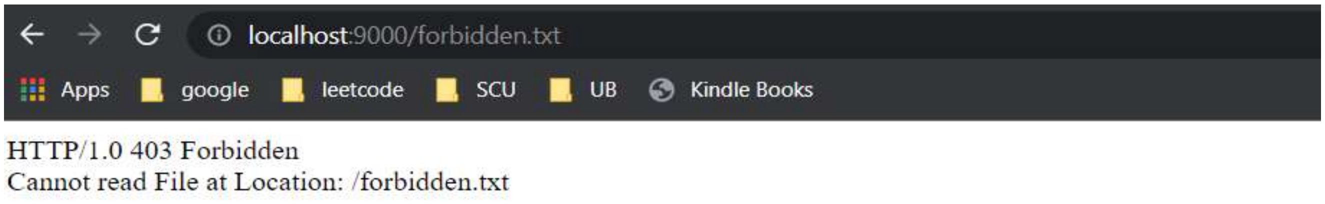
3. **<u>Supported Response Headers:</u>**



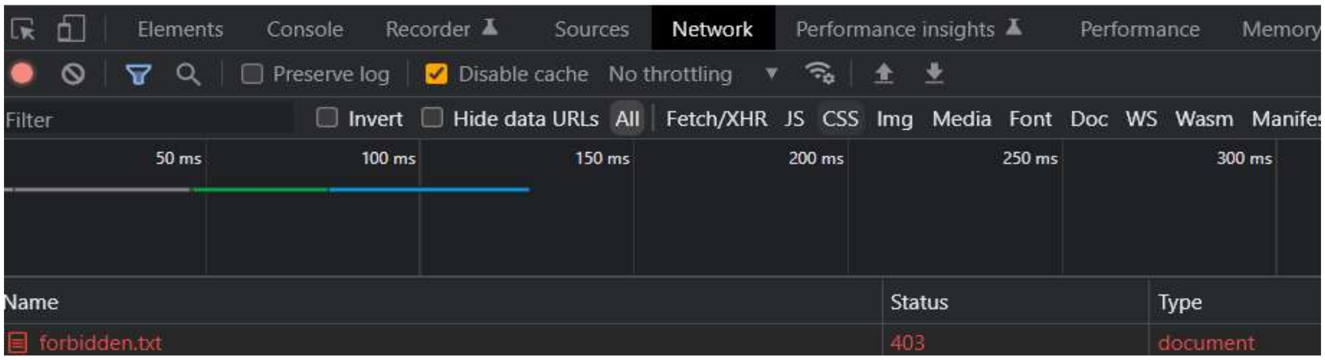
```
▾ General
    Request URL: http://localhost:9000/
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: [::1]:9000
    Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers                                    View source
    Content-length: 36198
    Content-type: text/html
    Date: Thu, 13 Oct 2022, 08:50:43 PDT
▾ Request Headers                                     View source
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
```

<u>Status codes implemented</u> – 400, 403, 404, 200

<u>403:</u> Accessing a forbidden text file – Do not have required permission on linux to read the file when the file exists.

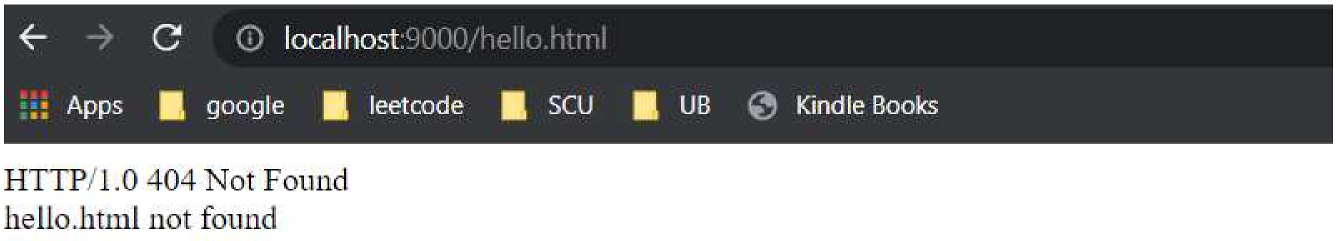

localhost:9000/forbidden.txt

```
HTTP/1.0 403 Forbidden
Cannot read File at Location: /forbidden.txt
```

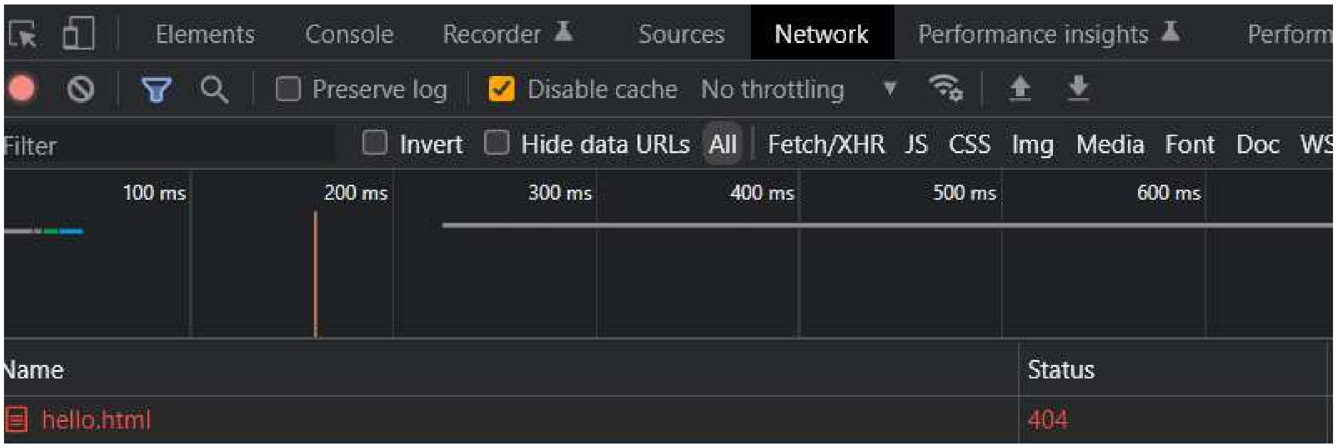


Name	Status	Type
forbidden.txt	403	document

<u>404:</u> When the file that the client is trying to access does not exist.

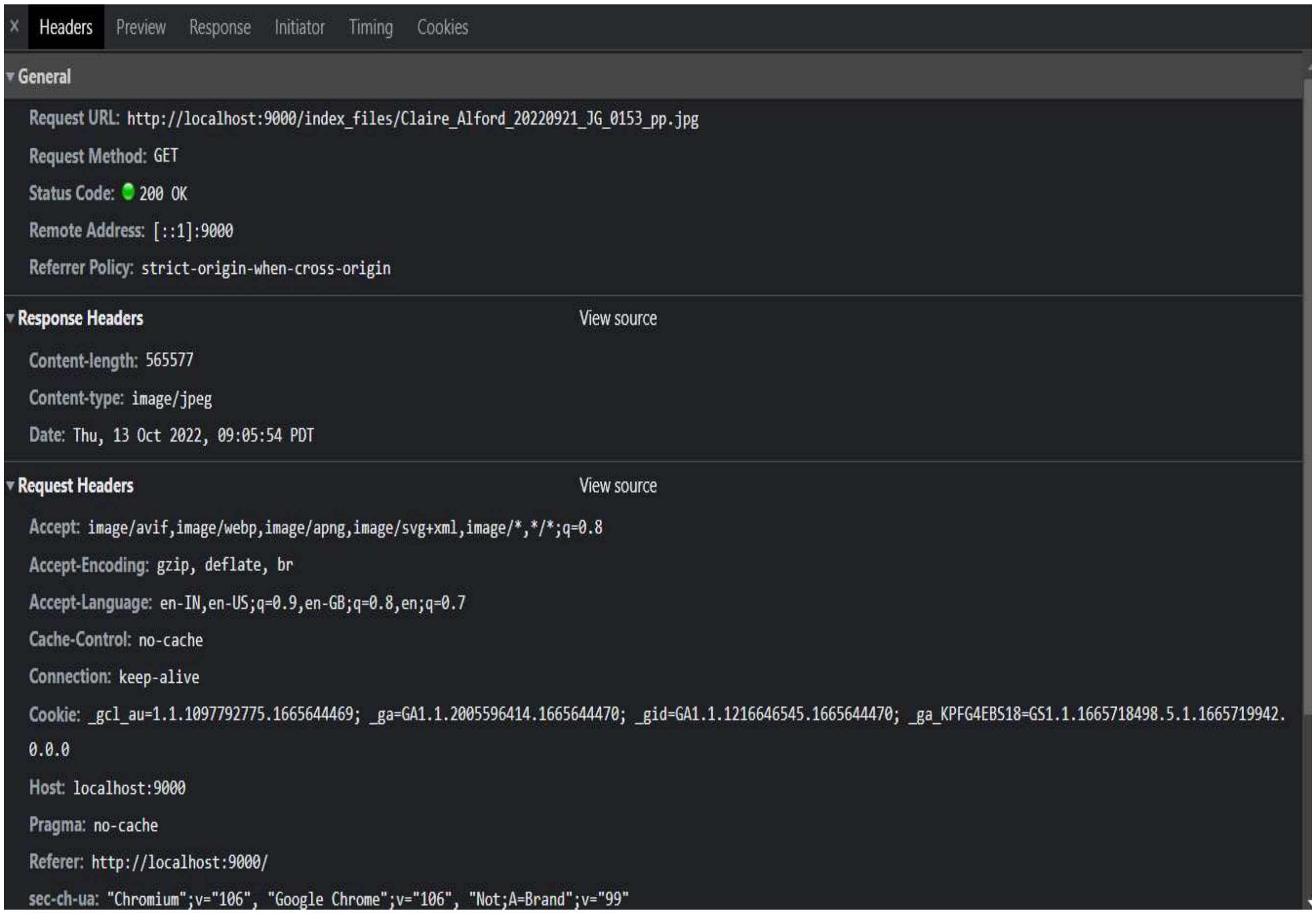

localhost:9000/hello.html

```
HTTP/1.0 404 Not Found
hello.html not found
```



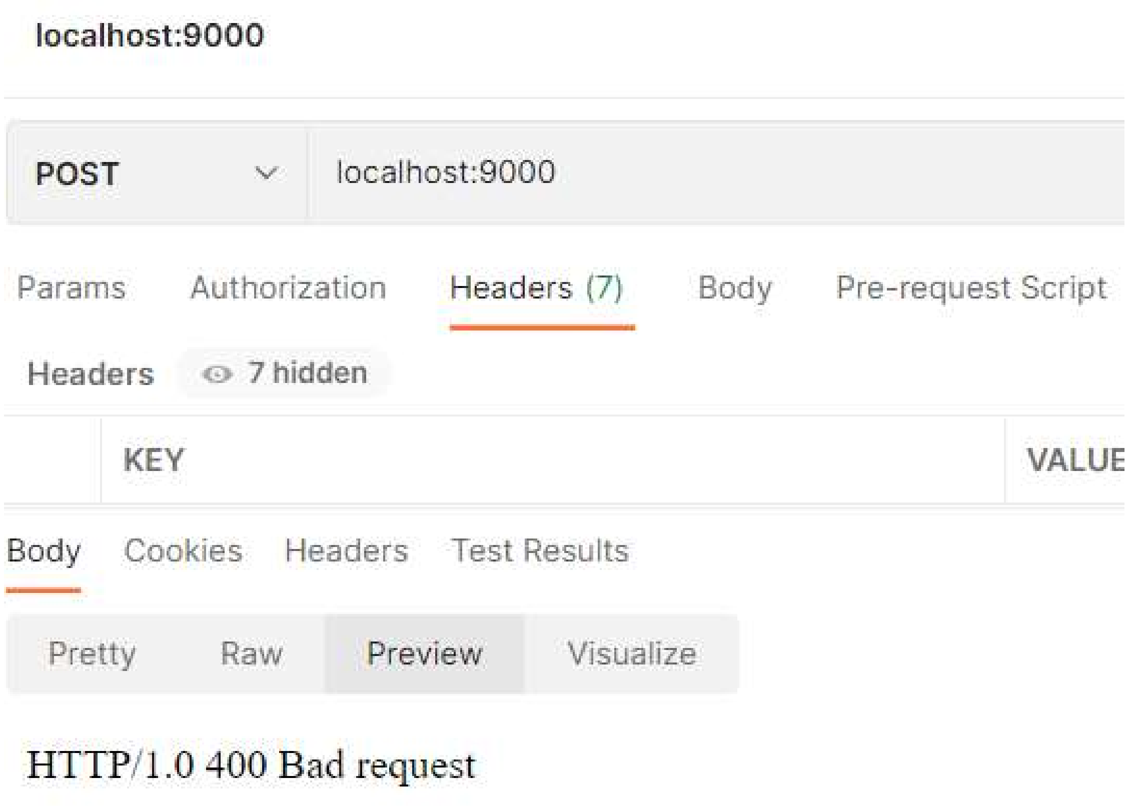
Name	Status
hello.html	404

<u>200:</u> When the request has been successful. Status code 200 OK is sent back to the client.



```
▾ General
  Request URL: http://localhost:9000/index_files/Claire_Alford_20220921_JG_0153_pp.jpg
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: [::1]:9000
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers                                   View source
  Content-length: 565577
  Content-type: image/jpeg
  Date: Thu, 13 Oct 2022, 09:05:54 PDT
▾ Request Headers                                    View source
  Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
  Accept-Encoding: gzip, deflate, br
  Accept-Language: en-IN,en-US;q=0.9,en-GB;q=0.8,en;q=0.7
  Cache-Control: no-cache
  Connection: keep-alive
  Cookie: _gcl_au=1.1.1097792775.1665644469; _ga=GA1.1.2005596414.1665644470; _gid=GA1.1.1216646545.1665644470; _ga_KPFG4EBS18=GS1.1.1665718498.5.1.1665719942.
  0.0.0
  Host: localhost:9000
  Pragma: no-cache
  Referer: http://localhost:9000/
  sec-ch-ua: "Chromium";v="106", "Google Chrome";v="106", "Not;A=Brand";v="99"
```

<u>400:</u> Unsupported HTTP method is requested, for example: POST request from the client.



```
HTTP/1.0 400 Bad request
```

3. References:

- HTTP Wikipedia http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- w3c HTTP page: https://www.w3.org/Protocols/
- Java threads: https://www.w3schools.com/java/java_threads.asp
- Java socket programming: https://www.javatpoint.com/socket-programming