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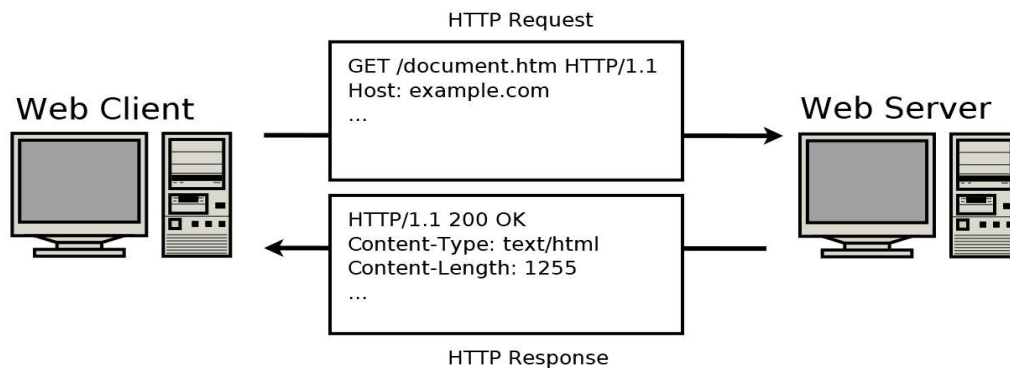
(Image Source: www.google.com - Webserver and web client)

- **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)
  - Forbidden.txt (Permission to be set to chmod 000 - Testing 403 Forbidden)

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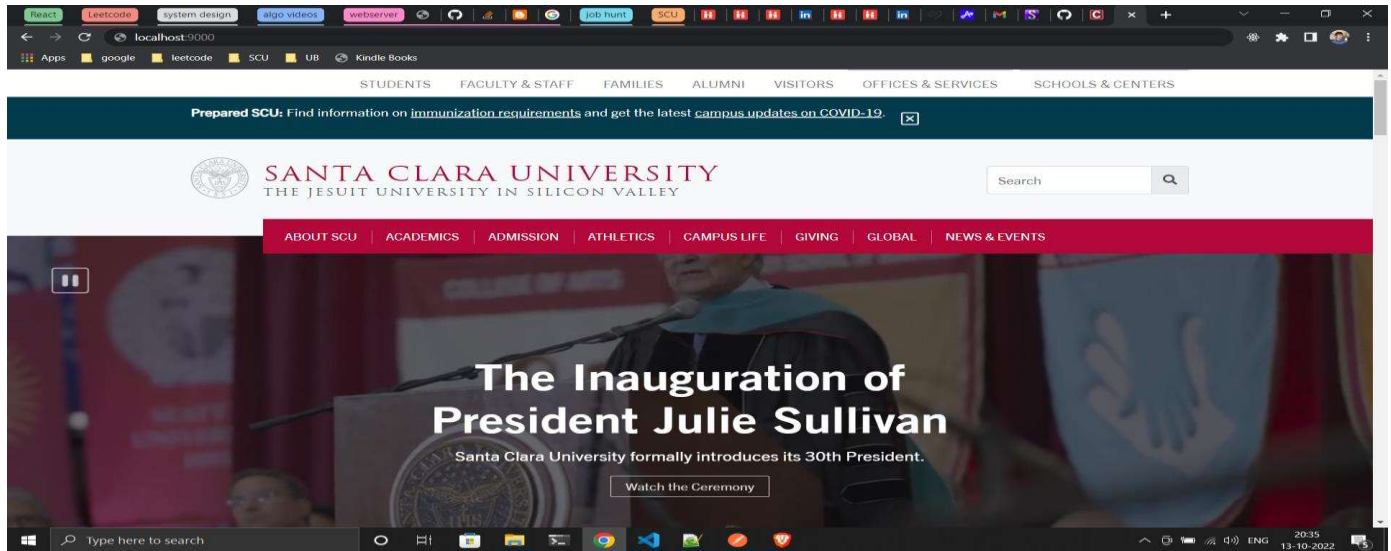
2. Sample logs screenshot, multiple threads are being spawned for each request from the client side -

3. **Supported Response Headers:**

```
▼ 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,*/*
```
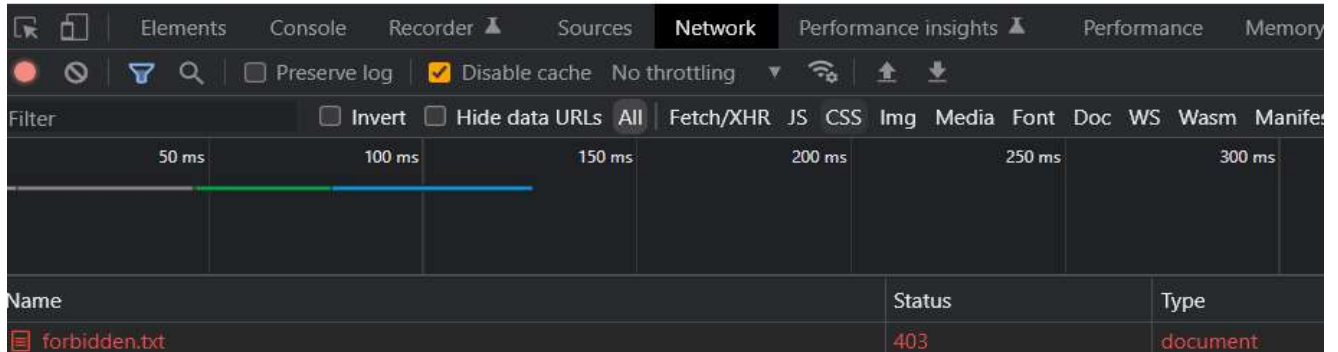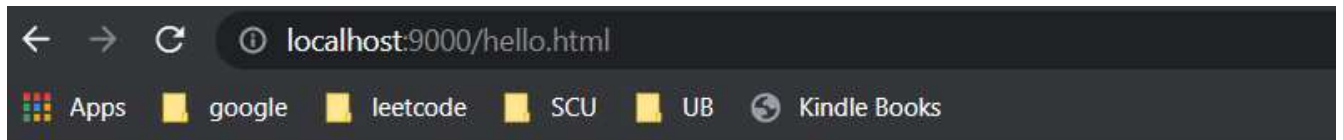
Status codes implemented – 400, 403, 404, 200

403: Accessing a forbidden text file – Do not have required permission on linux to read the file when the file exists.
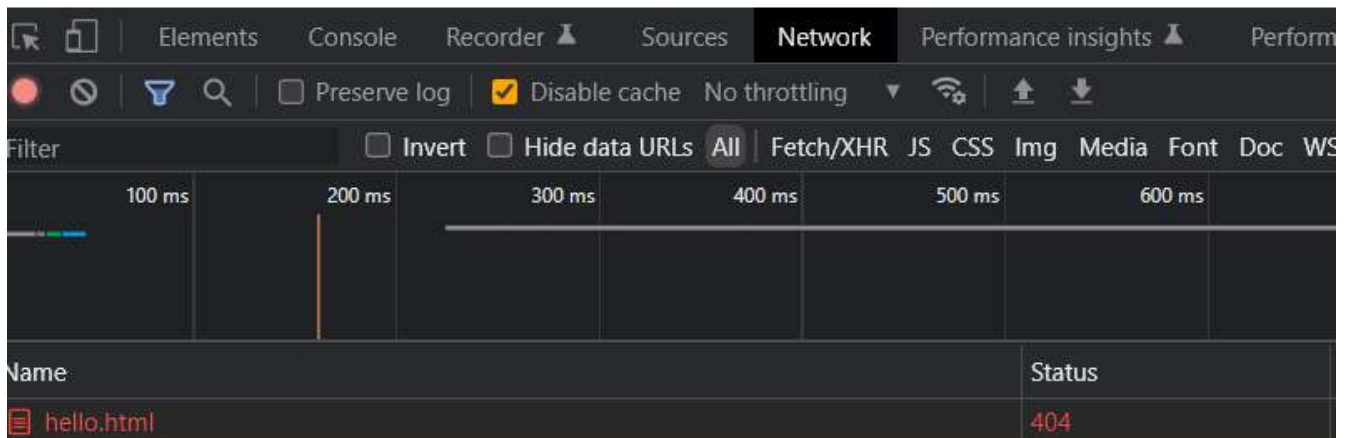
localhost:9000/forbidden.txt

Apps   google   leetcode   SCU   UB   Kindle Books
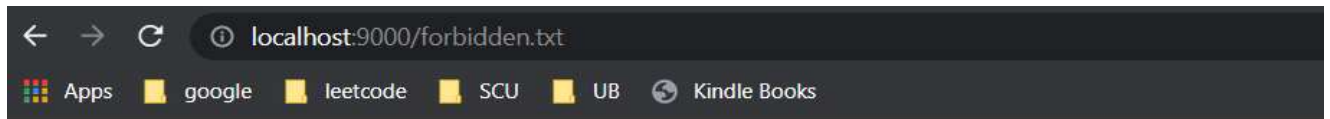
HTTP/1.0 403 Forbidden
Cannot read File at Location: /forbidden.txt

Elements   Console   Recorder   Sources   **Network**   Performance insights   Performance   Memory

Preserve log   ☑ Disable cache   No throttling

Filter   ☐ Invert   ☐ Hide data URLs   All   Fetch/XHR   JS   CSS   Img   Media   Font   Doc   WS   Wasm   Manifes

50 ms   100 ms   150 ms   200 ms   250 ms   300 ms

Name	Status	Type
forbidden.txt	403	document

404: When the file that the client is trying to access does not exist.

localhost:9000/hello.html

Apps   google   leetcode   SCU   UB   Kindle Books

HTTP/1.0 404 Not Found
hello.html not found

Elements   Console   Recorder   Sources   **Network**   Performance insights   Perform

Preserve log   ☑ Disable cache   No throttling

Filter   ☐ Invert   ☐ Hide data URLs   All   Fetch/XHR   JS   CSS   Img   Media   Font   Doc   WS

100 ms   200 ms   300 ms   400 ms   500 ms   600 ms

Name	Status
hello.html	404

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

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

localhost:9000

POST ⌄	localhost:9000

Params   Authorization   Headers (7)   Body   Pre-request Script

Headers   👁 7 hidden

KEY	VALUE

Body   Cookies   Headers   Test Results

Pretty   Raw   Preview   Visualize

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