



ભારતીય સોંકેટિક વિજ્ઞાન સંસ્કૃત પ્રેરણાબાદ
ભારતીય પ્રૌદ્યોગિકી સંસ્થાન હૈદરાબાદ
Indian Institute of Technology Hyderabad

ACN PROGRAMMING ASSIGNMENT 2
WWW

SUBMITTED BY:

1. Ashish B Emmanuel: CS23MTECH11004
2. Ashutosh Rajput: CS23MTECH11005
3. Keshvi Dharaiya: SM23MTECH14002

GROUP ID AND EXTENSION:

- GROUP-3
5. Speedy Web Client

COURSE:

Advanced Computer Networking

FACULTY:

Prof. Bheemarjuna Reddy Tamma

Index

<u>Sr.no</u>	<u>Content</u>
1.	Simple Web Client directly to the Web server
2.	Simple Web Client through the web proxy
3.	Mozilla Firefox and Web Proxy
4.	Simple Web Client – Web proxy – Web server
5.	Simple Web Client and Web Server
6.	Browser to Web Server through Proxy
7.	Extended Web Client directly to the Web server
8.	References
9.	Anti-plagiarism statement

Introduction

In this assignment, we have to make the Web clients, Web servers, Web proxy and one extension. It is divided into four parts like:

Part 1 is the implementation of a basic web client

Part 2 is the implementation of the basic web server

Part 3 is the implementation of a basic web proxy

Part 4 for us it is Extension 5 which is Speedy Web Client

In this assignment, Python language is used for the code and have executed all the Parts of the assignment. In the fourth part, we have selected Extension 5 which is Speedy Web Client. Please note that:

PORT number of Simple Web Server: 12001

PORT number of Proxy Server: 12002

The results and explanation of all the parts are as follows.

Simple Web Client directly to the Web Server

The Simple client code is already there in the folder provided with the name “Client.py”

In the coding part of the Simple client, the client is requesting for the Images, Icons, Scripts and links and also downloading them in the Folder “Downloads”. For this, we have created different functions and if for example image is not found on the web server- in this type of case it prints the statement “No Images Found!” like that for all.

The client is getting the different images, icons, scripts, HTML files, and links from the cse.iith.ac.in the website and it is downloading all files in the Downloads folder as it is mentioned in the code. Here is the screenshot of the same. We can see that it is making one folder with the name “Downloads” and saving all the files in it.

See in the output client is getting status codes 200 and 302 here. 200 is - Request succeeded and the information is returned in the response. 302 indicates that the Request source has moved but found! This status code client is getting for the scripts. In a similar way, we have handled some of the status codes like 400,401,404,301,304 etc.

```
ashish@ashish-Lenovo-Legion-5-15ARH05:~/Downloads/Prg-Asg$ python3 Client.py cse.iith.ac.in 443 /
Total Number of Arguments entered! : <built-in function len>

Connection through normal webserver

Url is : GET http://cse.iith.ac.in/ HTTP/1.0
Host: cse.iith.ac.in:443

Connection Made with server
The Status Code is : 200
The received file is a base HTML file

The Title of the HTML file is : Home | Department of CSE, IIT Hyderabad
Parsing images

Image References list

1. ./assets/images/CSELogo.png

2. ./assets/images/gallery/LYD01709.jpg

3. ./assets/images/gallery/LYD01882.jpg

4. ./assets/images/gallery/LYD01721.jpg

5. ./assets/images/gallery/LYD01843.jpg

Making connection for getting image : 1

Connection Made with server
Closing connection for image : 1

Making connection for getting image : 2

Connection Made with server
Closing connection for image : 2

Making connection for getting image : 3
```

```
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg

Connection Made with server
Closing connection for Script : 5

Making Connection for getting the script : 6

Connection Made with server
Error in getting Script, Status Code : 302, Request source has moved but found!

Closing connection for Script : 6

Making Connection for getting the script : 7

Connection Made with server
Error in getting Script, Status Code : 302, Request source has moved but found!

Closing connection for Script : 7

Making Connection for getting the script : 8

Connection Made with server
Closing connection for Script : 8

Making Connection for getting the script : 9

Connection Made with server
Closing connection for Script : 9

Making Connection for getting the script : 10

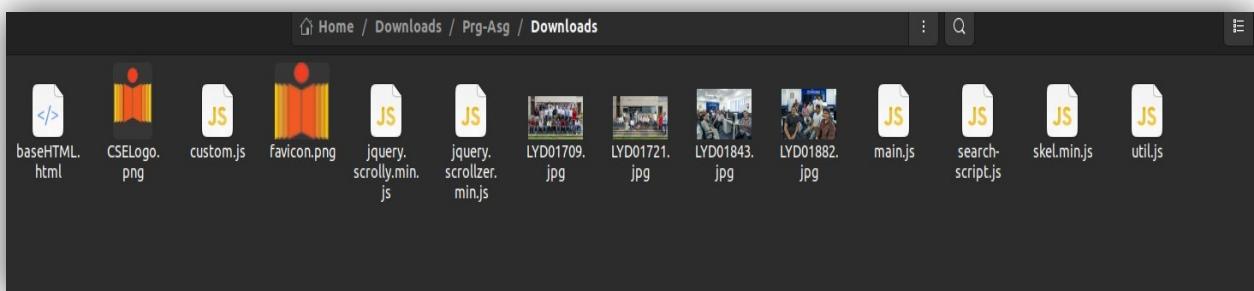
Connection Made with server
Closing connection for Script : 10

Making Connection for getting the script : 11

Connection Made with server
Closing connection for Script : 11

The Total time taken is : 37.46131634712219 seconds
ashish@ashish-Lenovo-Legion-5-15ARH05:~/Downloads/Prg-Asg$
```

We have also calculated the time that how much time our simple web client is taking to get all the information. Here we can see that it is taking 37.46 sec.

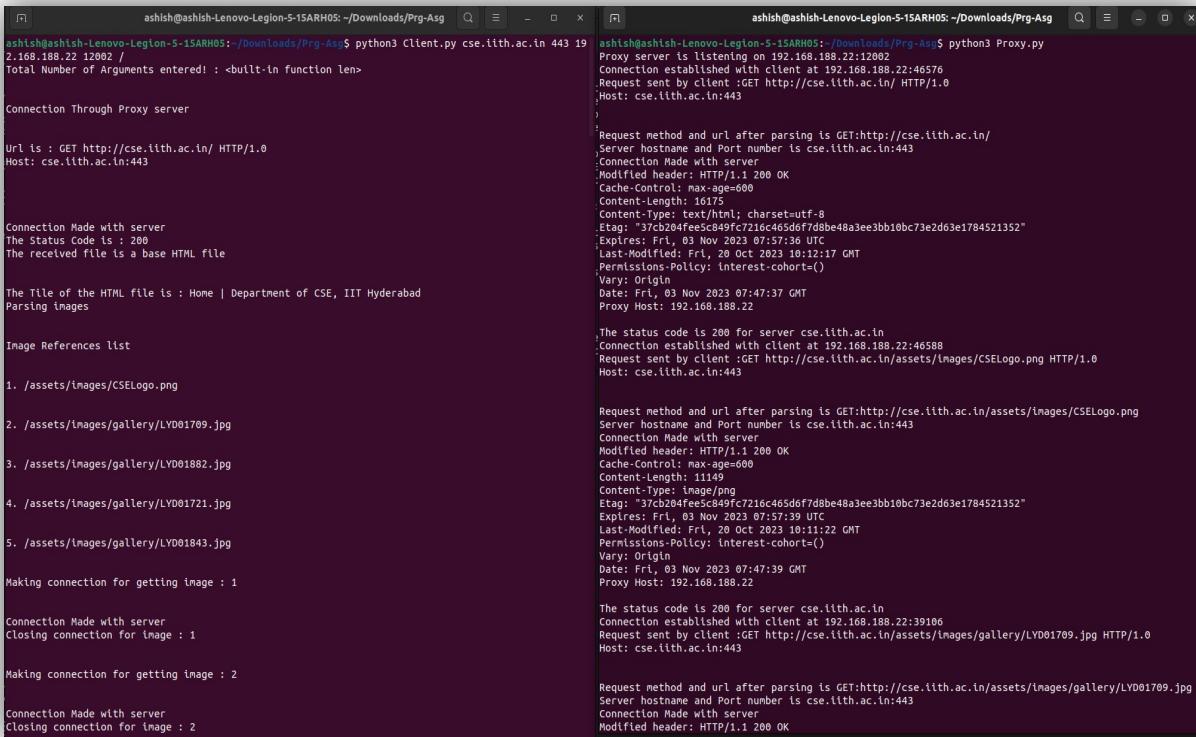


Simple Web Client through the Web Proxy

Here in this part again the client has requested the cse.iith.ac.in the website only but by using our Web proxy. Here on the left side Client is running and connecting to the web proxy and on the right side our web proxy is running and establishing the connection with the client and then with the cse web server.

Here first the client makes a connection with the Proxy then the web proxy requests to the cse.iith.ac.in the website and makes a connection with that server. After this server will send the HTML file, images, scripts etc. to the web proxy and the web proxy will send it to the client. Here in the results, we can see that as like the above case, the client is getting an error message for scripts that is 302 status code - Request source has moved but found! through the web proxy.

Here also we have calculated the time and that is 38.17 sec.



The screenshot shows two terminal windows side-by-side. The left terminal window (Client) shows the command `python3 Client.py cse.iith.ac.in 443 19 2.168.188.22 12002 /` being run, followed by the output: "Total Number of Arguments entered! : <built-in function len>". It then displays the process of connecting through the proxy server, showing the URL as `http://cse.iith.ac.in/` and the host as `cse.iith.ac.in:443`. It also shows the received HTML file content, which includes the CSE logo image. The right terminal window (Proxy) shows the command `python3 Proxy.py` being run, followed by detailed logs of the proxy's actions. These logs include the proxy server listening on port 12002, establishing a connection with the client at 192.168.188.22:46576, sending a GET request to the cse.iith.ac.in server, and receiving the HTML response. The logs also show the proxy sending the HTML back to the client and handling image requests for the CSE logo and other gallery images. The logs are timestamped with dates like Nov 03, 2023, and times like 07:57:39 UTC.

```

ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg
```

Connection Made with server
Closing connection for Script : 5

Making Connection for getting the script : 6

Connection Made with server
Error in getting Script, Status Code : 302, Request source has moved but found!

Closing connection for Script : 6

Making Connection for getting the script : 7

Connection Made with server
Error in getting Script, Status Code : 302, Request source has moved but found!

Closing connection for Script : 7

Making Connection for getting the script : 8

Connection Made with server
Closing connection for Script : 8

Making Connection for getting the script : 9

Connection Made with server
Closing connection for Script : 9

Making Connection for getting the script : 10

Connection Made with server
Closing connection for Script : 10

Making Connection for getting the script : 11

Connection Made with server
Closing connection for Script : 11

The Total time taken is : 38.169841051101685 seconds

```

ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg
```

Content-Type: application/javascript
Etag: "37cb20afe5c849fc7216c465d6f7d8be48a3ee3bb10bc73e2d63e1784521352"
Expires: Fri, 03 Nov 2023 07:58:09 UTC
Last-Modified: Fri, 20 Oct 2023 10:11:22 GMT
Permissions-Policy: interest-cohort=()
Vary: Origin
Date: Fri, 03 Nov 2023 07:48:09 GMT
Proxy Host: 192.168.188.22

The status code is 200 for server cse.lith.ac.in
Connection established with client at 192.168.188.22:36084
Request sent by client :GET http://cse.lith.ac.in/assets/js/main.js HTTP/1.0
Host: cse.lith.ac.in:443

Request method and url after parsing is GET: http://cse.lith.ac.in/assets/js/main.js
Server hostname and Port number is cse.lith.ac.in:443
Connection Made with server
Modified header: HTTP/1.1 200 OK
Cache-Control: max-age=600
Content-Length: 395
Content-Type: application/javascript
Etag: "37cb20afe5c849fc7216c465d6f7d8be48a3ee3bb10bc73e2d63e1784521352"
Expires: Fri, 03 Nov 2023 07:58:11 UTC
Last-Modified: Fri, 20 Oct 2023 10:11:22 GMT
Permissions-Policy: interest-cohort=()
Vary: Origin
Date: Fri, 03 Nov 2023 07:48:11 GMT
Proxy Host: 192.168.188.22

The status code is 200 for server cse.lith.ac.in
Connection established with client at 192.168.188.22:36100
Request sent by client :GET http://cse.lith.ac.in/assets/js/custom.js HTTP/1.0
Host: cse.lith.ac.in:443

Request method and url after parsing is GET: http://cse.lith.ac.in/assets/js/custom.js
Server hostname and Port number is cse.lith.ac.in:443
Connection Made with server
Modified header: HTTP/1.1 200 OK
Cache-Control: max-age=600
Content-Length: 2963
Content-Type: application/javascript
Etag: "37cb20afe5c849fc7216c465d6f7d8be48a3ee3bb10bc73e2d63e1784521352"
Expires: Fri, 03 Nov 2023 07:58:13 UTC
Last-Modified: Fri, 20 Oct 2023 10:11:22 GMT
Permissions-Policy: interest-cohort=()
Vary: Origin
Date: Fri, 03 Nov 2023 07:48:13 GMT
Proxy Host: 192.168.188.22

The status code is 200 for server cse.lith.ac.in

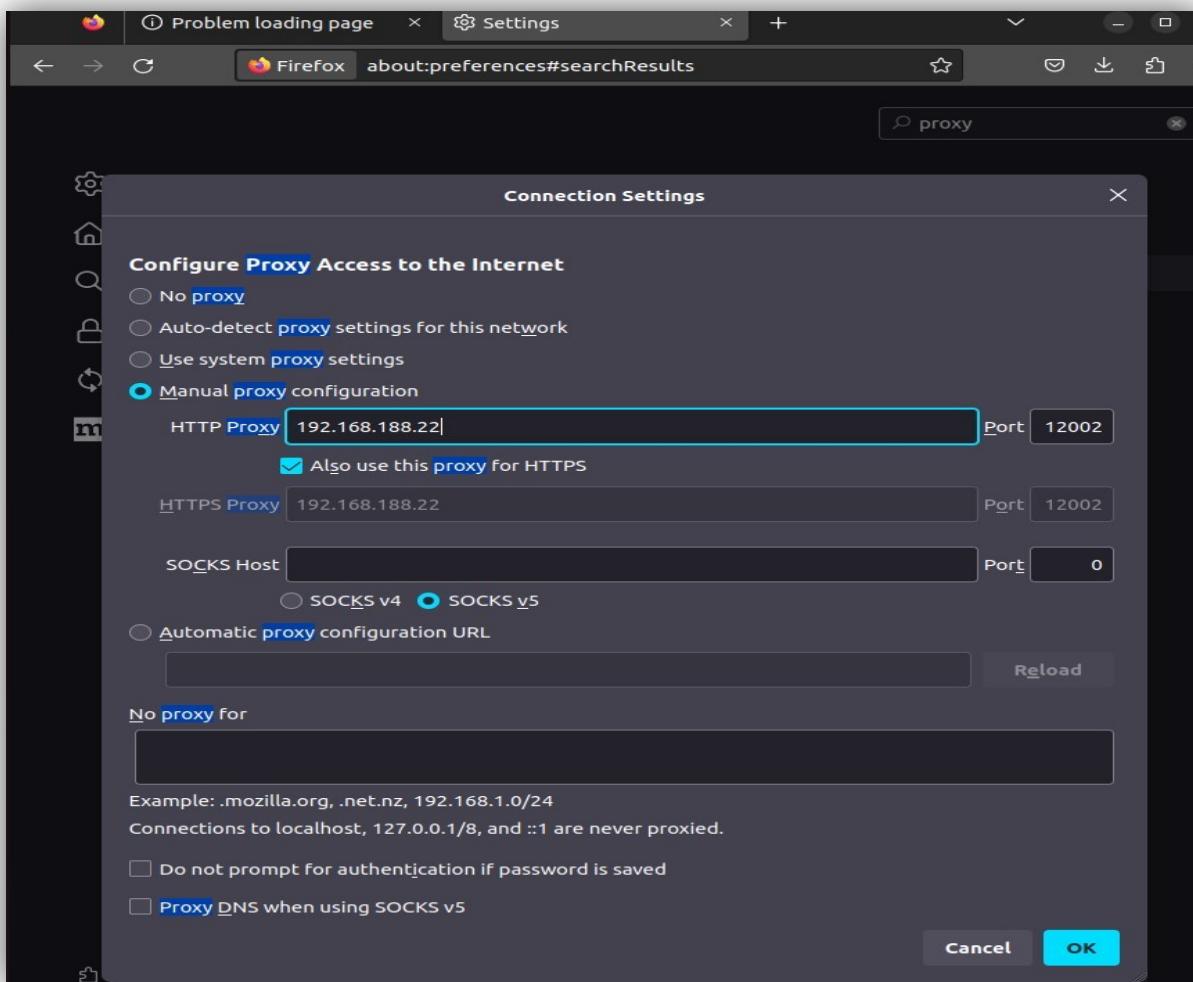
Here in the above screenshot, we can see that the client is making a connection again and again for scripts 1,2,3... like that for images also it is making a connection again and again because it is a non – persistence TCP connection and not the persistent one.

In this assignment, we are using non-persistence so for all the results it is showing like that.

Mozilla Firefox and Web Proxy

In this part, we are sending a request from Mozilla Firefox to our web proxy and getting the following results. For this, we have changed the connection settings of the Mozilla Firefox and selected the Manual proxy configuration and HTTP Proxy. Here our proxy is connecting with the web browser and had a request sent by the client for “httpbin.org”.

Here we observed that we are getting the same results by using a web proxy for this website.



The screenshot shows a dual-pane interface. On the left is a web browser displaying the [httpbin.org](http://www.httpbin.org) homepage. The page features a navigation bar with links for 'Settings', 'HTTP Methods', 'Auth', 'Status codes', 'Request inspection', 'Response inspection', 'Response formats', 'Dynamic data', 'Cookies', 'Images', and 'Redirects'. Below the navigation is a sidebar titled 'Schemes' with a dropdown menu set to 'HTTP'. The main content area displays various sections of the httpbin documentation. On the right is a terminal window with a dark background and white text. It shows a command-line session where a Python proxy script is running on port 52880. The terminal output includes several log entries from different clients making requests to the httpbin service. One entry shows a client connecting via CONNECT to push.services.mozilla.com:443, while another shows a client connecting to fonts.googleapis.com:443.

```
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg$ python3 Proxy.py
Proxy server is listening on 192.168.188.22:52880
Connection established with client at 192.168.188.22:52880
Request sent by client :GET http://www.httpbin.org/ HTTP/1.1
Host: www.httpbin.org
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Request method and url after parsing is GET:http://www.httpbin.org/
Server hostname and Port number is www.httpbin.org:80
Connection established with client at 192.168.188.22:52886
Request sent by client :CONNECT push.services.mozilla.com:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: push.services.mozilla.com:443

Request method and url after parsing is CONNECT:push.services.mozilla.com:443
Server hostname and Port number is push.services.mozilla.com:443
Connection Made with server
Modified header: HTTP/1.1 200 OK
Date: Fri, 03 Nov 2023 08:04:03 GMT
Content-type: text/html; charset=utf-8
Content-length: 593
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Proxy Host: 192.168.188.22

The status code is 200 for server www.httpbin.org
Connection established with client at 192.168.188.22:52900
Request sent by client :CONNECT fonts.googleapis.com:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: fonts.googleapis.com:443

Request method and url after parsing is CONNECT:fonts.googleapis.com:443
Server hostname and Port number is fonts.googleapis.com:443
Connection established with client at 192.168.188.22:52912
Request sent by client :GET http://www.httpbin.org/flasgger_static/swagger-ui-bundle.js HTTP/1.1
Host: www.httpbin.org
```

Simple Web Client – Web proxy – Web server

In this part, the web client -web proxy and web server- are all made by us and establishing the connection & getting the HTML file and following the results.

Here on the left side client is running, in the middle web proxy is running and on the right side web server is running. The client is first connected with the proxy server. The screenshot shows the message on the client and proxy side that the connection is established. Here in the results, we can see that the web server is handling 3 clients.

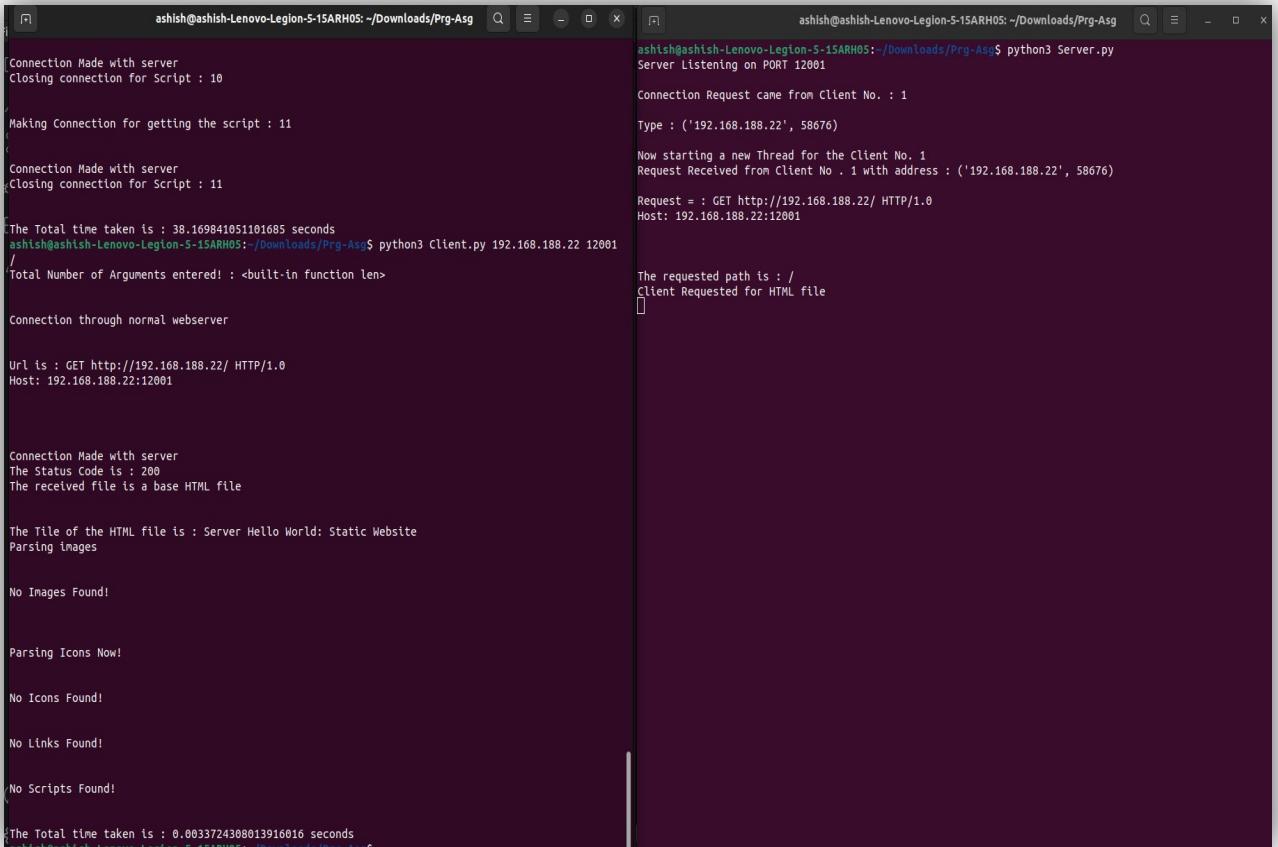
The server has three HTML files with title “HelloWorld”, “TestHtml1” and “TestHtml2” so the client gets the file that was requested. It doesn't have images, scripts, links etc. so it displays the message- “NO Images Found!”, “No Scripts Found!” etc.

The screenshot displays three terminal windows side-by-side, illustrating the flow of requests from a client to a proxy and then to a server.

- Left Terminal (Client):** Shows the execution of `Client.py` with arguments `192.168.188.22 12001 192.168.188.22 12002`. It outputs "Connection Through Proxy server", the URL `http://192.168.188.22/ HTTP/1.0`, and the host `192.168.188.22:12001`. It also indicates "Connection Made with server", "The Status Code is : 200", and "The received file is a base HTML file". It then lists the content of the HTML file: "The Title of the HTML file is : Server Hello World: Static Website", "Parsing images", "No Images Found!", "Parsing Icons Now!", "No Icons Found!", "No Links Found!", and "No Scripts Found!". Finally, it shows the total time taken: "The Total time taken is : 0.0036318302154541016 seconds".
- Middle Terminal (Proxy):** Shows the execution of `Proxy.py`. It logs the proxy server listening on port 12002, a connection established from client 192.168.188.22:54932, and a request sent by the client for `http://192.168.188.22/ HTTP/1.0`. It also shows the status code 200 for the server 192.168.188.22.
- Right Terminal (Server):** Shows the execution of `Server.py`. It logs a connection request from client 192.168.188.22:45346, starting a new thread for client 1. It receives a request for `/` from client 192.168.188.22:45346. It then handles three more clients (No. 2, No. 3) for requests `/`, `/192.168.188.190`, and `/192.168.188.190` respectively. The server logs detailed headers for each request, including User-Agent, Accept, DNT, Upgrade-Insecure-Requests, and Accept-Language.

Simple Web Client and Web Server

This part is almost the same as above, only the change is the client is directly requesting to the web server without a web proxy and getting the same results as we are getting above, like the same HTML file and Icons and showing there are no images, no scripts, no links etc.



The image shows two terminal windows side-by-side. The left window represents the client and the right window represents the server.

Client Terminal (Left):

```
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg
Connection Made with server
Closing connection for Script : 10

Making Connection for getting the script : 11

Connection Made with server
Closing connection for Script : 11

The Total time taken is : 38.169841051101685 seconds
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg$ python3 Client.py 192.168.188.22 12001
/
'Total Number of Arguments entered! : <built-in function len>

Connection through normal webserver

Url is : GET http://192.168.188.22/ HTTP/1.0
Host: 192.168.188.22:12001

Connection Made with server
The Status Code is : 200
The received file is a base HTML file

The Title of the HTML file is : Server Hello World: Static Website
Parsing Images

No Images Found!

Parsing Icons Now!

No Icons Found!

No Links Found!

No Scripts Found!

The Total time taken is : 0.0033724308013916816 seconds
```

Server Terminal (Right):

```
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg
Server Listening on PORT 12001

Connection Request came from Client No. : 1
Type : ('192.168.188.22', 58676)

Now starting a new Thread for the Client No. 1
Request Received from Client No . 1 with address : ('192.168.188.22', 58676)
Request = : GET http://192.168.188.22/ HTTP/1.0
Host: 192.168.188.22:12001

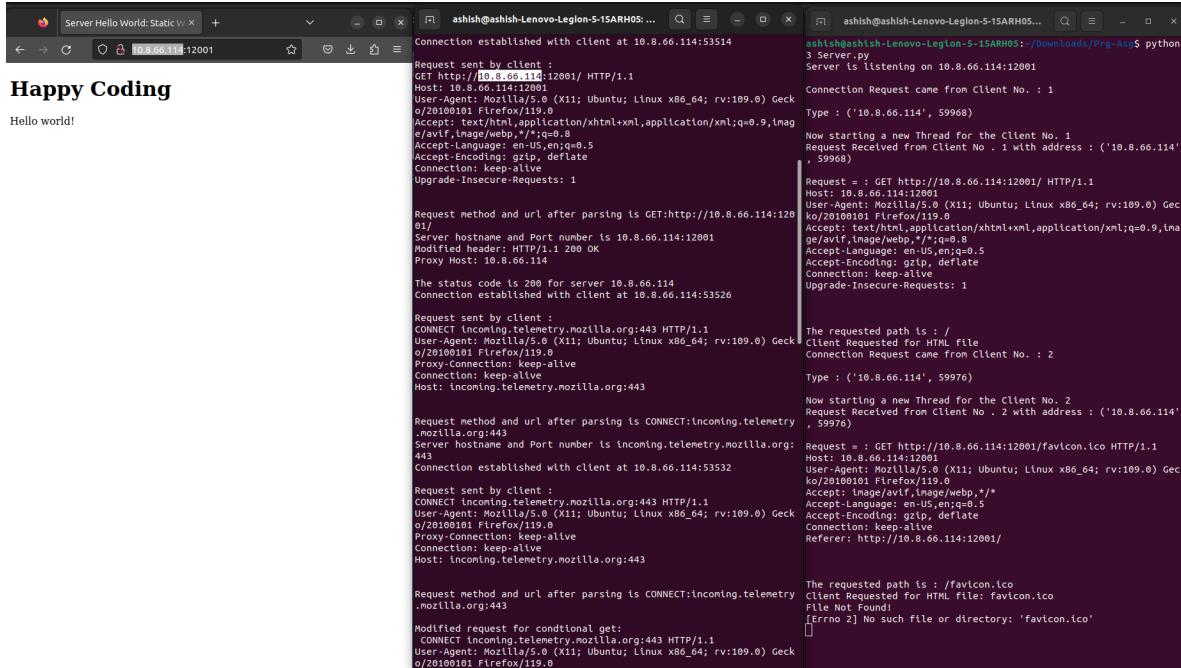
The requested path is : /
Client Requested for HTML file
```

Now if the client requests another file or something else that is not on our web server it will show the message “File not Found!” and terminate the connection.

Browser to Web Server through Proxy

In this part we are using web browser to send requests to web server through our proxy. We have to configure our web browser to use our proxy to send all requests.

In the Images below image the browser (on the left) is fetching HTML files from the server (running on the right terminal) through proxy (running on the middle terminal)



The screenshot shows a Firefox browser window on the left and a terminal window on the right. The browser window displays the text "Happy Coding". The terminal window shows the proxy log for the first request:

```
ashish@ashish-Lenovo-Legion-5-15ARH05:~$ python3 Server.py
Server is listening on 10.8.66.114:12001
Connection Request came from Client No. : 1
Type : ('10.8.66.114', 59968)
Now starting a new Thread for the Client No. 1
Request Received From Client No. 1 with address : ('10.8.66.114', 59968)
Request = : GET http://10.8.66.114:12001/ HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Request method and url after parsing is GET:http://10.8.66.114:12001/
Server hostname and Port number is 10.8.66.114:12001
Modified header: HTTP/1.1 200 OK
Proxy Host: 10.8.66.114

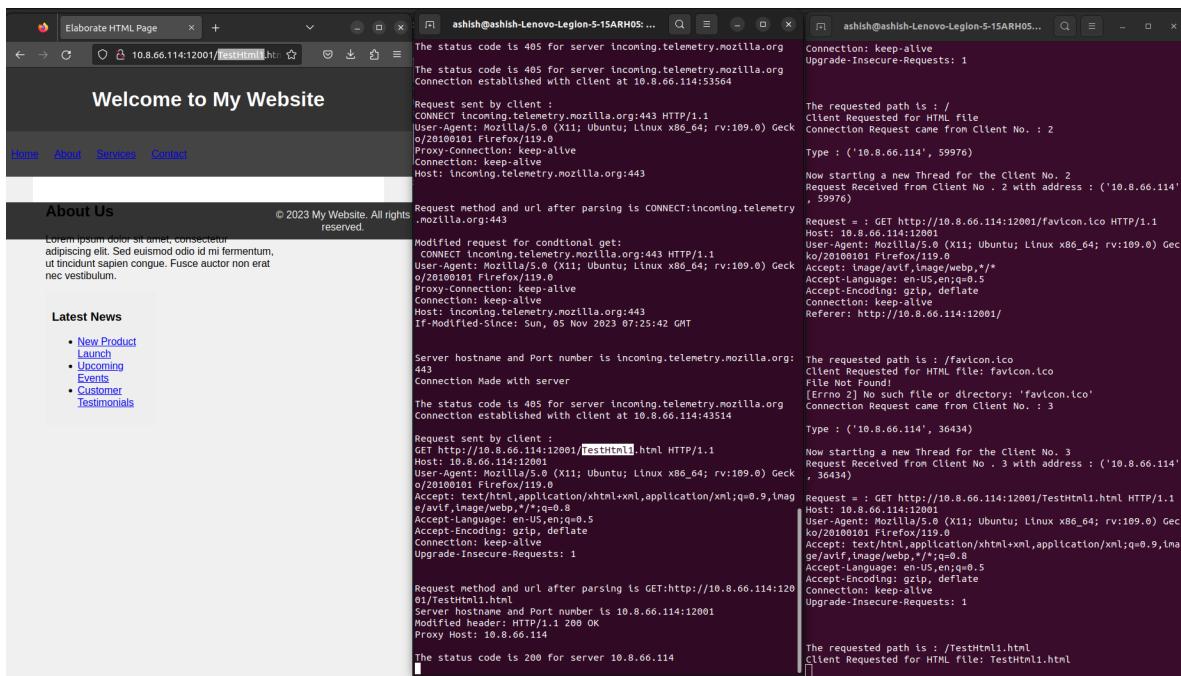
The status code is 200 for server 10.8.66.114
Connection established with client at 10.8.66.114:53526

Request sent by client :
CONNECT incoming.telemetry.mozilla.org:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: incoming.telemetry.mozilla.org:443

Request method and url after parsing is CONNECT:incoming.telemetry.mozilla.org:443
Server hostname and Port number is incoming.telemetry.mozilla.org:443
Connection established with client at 10.8.66.114:53526

Request sent by client :
CONNECT incoming.telemetry.mozilla.org:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: incoming.telemetry.mozilla.org:443

Request method and url after parsing is CONNECT:incoming.telemetry.mozilla.org:443
Modified request for conditional get:
CONNECT incoming.telemetry.mozilla.org:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
```



The screenshot shows a Firefox browser window on the left and a terminal window on the right. The browser window displays a website titled "Welcome to My Website" with a menu bar (Home, About, Services, Contact) and a sidebar with sections "About Us" and "Latest News". The terminal window shows the proxy log for the second request:

```
ashish@ashish-Lenovo-Legion-5-15ARH05:~$ python3 Server.py
Connection: keep-alive
Upgrade-Insecure-Requests: 1

The requested path is : /
Client Requested for HTML file
Connection Request came from Client No. : 2
Type : ('10.8.66.114', 59976)
Now starting a new Thread for the Client No. 2
Request Received From Client No. 2 with address : ('10.8.66.114', 59976)
Request = : GET http://10.8.66.114:12001/favicon.ico HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://10.8.66.114:12001

The requested path is : /favicon.ico
Client Requested for HTML file: favicon.ico
File Not Found!
[Errno 2] No such file or directory: 'favicon.ico'

ashish@ashish-Lenovo-Legion-5-15ARH05:~$ python3 Server.py
Connection: keep-alive
Upgrade-Insecure-Requests: 1

The requested path is : /
Client Requested for HTML file
Connection Request came from Client No. : 2
Type : ('10.8.66.114', 59976)
Now starting a new Thread for the Client No. 2
Request Received From Client No. 2 with address : ('10.8.66.114', 59976)
Request = : GET http://10.8.66.114:12001/favicon.ico HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://10.8.66.114:12001

The requested path is : /favicon.ico
Client Requested for HTML file: favicon.ico
File Not Found!
[Errno 2] No such file or directory: 'favicon.ico'
Connection Request came from Client No. : 3
Type : ('10.8.66.114', 36434)
Now starting a new Thread for the Client No. 3
Request Received From Client No. 3 with address : ('10.8.66.114', 36434)
Request = : GET http://10.8.66.114:12001/TestHtml1.html HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Request method and url after parsing is GET:http://10.8.66.114:12001/TestHtml1.html
Server hostname and Port number is 10.8.66.114:12001
Modified header: HTTP/1.1 200 OK
Proxy Host: 10.8.66.114

The status code is 200 for server 10.8.66.114
```

ashish@ashish-Lenovo-Legion-5-15ARH05: ~

```
Modified request for conditional get:
CONNECT incoming.telemetry.mozilla.org:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko
/20100101 Firefox/119.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: incoming.telemetry.mozilla.org:443
If-Modified-Since: Sun, 05 Nov 2023 07:25:42 GMT

Server hostname and Port number is incoming.telemetry.mozilla.org:443
Connection Made with server

The status code is 405 for server incoming.telemetry.mozilla.org
Connection established with client at 10.8.66.114:43514

Request sent by client :
GET http://10.8.66.114:12001/TestHtml1.html HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko
/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

The requested path is : /favicon.ico
Client Requested for HTML file: favicon.ico
File Not Found!
[Errno 2] No such file or directory: 'favicon.ico'
Connection Request came from Client No. : 3
Type : ('10.8.66.114', 36434)

Now starting a new Thread for the Client No. 3
Request Received from Client No . 3 with address : ('10.8.66.114', 36434)

Request = : GET http://10.8.66.114:12001/TestHtml1.html HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko
/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

The requested path is : /TestHtml1.html
Client Requested for HTML file: TestHtml1.html
Connection Request came from Client No. : 4
Type : ('10.8.66.114', 35952)

Now starting a new Thread for the Client No. 4
Request Received from Client No . 4 with address : ('10.8.66.114', 35952)

Request = : GET http://10.8.66.114:12001/TestHtml2.html HTTP/1.1
Host: 10.8.66.114:12001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko
/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

The requested path is : /TestHtml2.html
Client Requested for HTML file: TestHtml2.html
```

Extended Web Client directly to the Webserver

It's part 4 of the assignment which is a speedy web client. Here also our speedy web client is requesting the cse.iith.ac.in the web server and getting a quick response in 4.83 seconds. We observed that in part 1 we were getting responses in 37.46 seconds and here as it is a speedy client, we are getting it in only 4.83 seconds. Because Here client is making a parallel connection with the web server and saving all the files parallelly. To make the parallel TCP connections we are using the "ThreadPoolExecutor" library which is available in Python.

Here also it is handling different status codes like 200, 302...etc. and here also it is downloading all images, icons, html files etc. in the folder named "Parallel Downloads".

```
ashish@ashish-Lenovo-Legion-5-15ARH05:~/Downloads/Prg-Asg$ python3 ExtendedClient.py cse.iith.ac.in 443 /
Total Number of Arguments entered! : 4

Connection through webserver

Url is : GET http://cse.iith.ac.in/ HTTP/1.0
Host: cse.iith.ac.in:443

Making connection with the server
The Status Code is : 200
HTML file received Successfully!
The received file is a base HTML file

The Title of the HTML file is : Home | Department of CSE, IIT Hyderabad
Parsing images

Image References list

1. ./assets/images/CSELogo.png

2. ./assets/images/gallery/LYD01709.jpg

3. ./assets/images/gallery/LYD01882.jpg

4. ./assets/images/gallery/LYD01721.jpg

5. ./assets/images/gallery/LYD01843.jpg

Parsing Icons Now!

1. <link href="/assets/images/favicon.png" rel="shortcut icon" type="image/png"/>

Parsing links

1. https://www.iith.ac.in/

2. /
```

```
ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg$ 
The Status code is : 302, for object : https://www.googletagmanager.com/gtag/js?id=UA-137608318-2

The Status code is : 302, for object : https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js

Error in downloading Object, Error : 302 : Request source has moved but found!
Object : GET http://cse.iith.ac.inhttps://www.googletagmanager.com/gtag/js?id=UA-137608318-2 HTTP/1.0
Host: cse.iith.ac.in:443


The Status code is : 302, for object : https://code.jquery.com/jquery-1.12.4.min.js

Error in downloading Object, Error : 302 : Request source has moved but found!
Object : GET http://cse.iith.ac.inhttps://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js HTTP/1.0
Host: cse.iith.ac.in:443


Error in downloading Object, Error : 302 : Request source has moved but found!
Object : GET http://cse.iith.ac.inhttps://code.jquery.com/jquery-1.12.4.min.js HTTP/1.0
Host: cse.iith.ac.in:443


The Status code is : 200, for object : /assets/js/skel.min.js

The Status code is : 200, for object : /assets/js/jquery.scrollify.min.js

The Status code is : 200, for object : /assets/images/gallery/LYD01882.jpg

The Status code is : 200, for object : /assets/js/custom.js

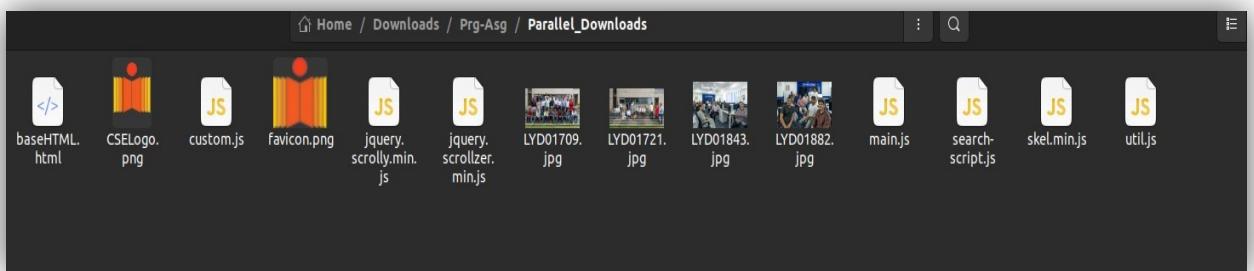
The Status code is : 200, for object : /assets/images/gallery/LYD01843.jpg

The Status code is : 200, for object : /assets/images/gallery/LYD01721.jpg

The Status code is : 200, for object : /assets/images/gallery/LYD01709.jpg

The total time taken is : 4.830164909362793 seconds

ashish@ashish-Lenovo-Legion-5-15ARH05: ~/Downloads/Prg-Asg$ 
```



References

- <https://www.geeksforgeeks.org/creating-a-proxy-webserver-in-python-set-1/>
- **How to make folders in python :**
<https://www.freecodecamp.org/news/creating-a-directory-in-python-how-to-create-a-folder/>
- **How to take Date and time in python :**
<https://www.geeksforgeeks.org/get-current-date-and-time-using-python/>
- **How to take substring in python :**
<https://learnpython.com/blog/substring-of-string-python/>
- **Chatgpt:**

In this assignment, we have used the CHATGPT to learn things properly, to get new ideas and solve the queries. Here are the screenshots:



How to take command line arguments in python



In Python, you can access command line arguments using the `sys.argv` list from the `sys` module. Here's a step-by-step guide on how to do this:



1. Import the `sys` module:

```
python
```

Copy code

```
import sys
```

2. Access the command line arguments using `sys.argv`. `sys.argv` is a list that contains the command-line arguments passed to your Python script. The first element (`sys.argv[0]`) is the script name, and the subsequent elements (`sys.argv[1:]`) are the arguments.

Here's an example of how to use command line arguments in Python:

```
python

import sys

# Check the number of command line arguments
if len(sys.argv) < 2:
    print("Usage: python script.py arg1 [arg2 ...]")
else:
    # Access individual command line arguments
    script_name = sys.argv[0]
    arguments = sys.argv[1:]

    # Print the script name
    print("Script Name:", script_name)

    # Print the command line arguments
    print("Arguments:", arguments)

    # You can access individual arguments by index
    if len(arguments) >= 1:
        print("First Argument:", arguments[0])

    # You can loop through all the arguments
    print("All Arguments:")
    for arg in arguments:
        print(arg)
```

 Copy code



How to parse image using Beautiful Soup



Beautiful Soup is a Python library primarily used for parsing and navigating HTML or XML documents, not for parsing images. If you want to work with images in Python, you may want to consider using libraries like Pillow (PIL) for image processing or OpenCV for more advanced computer vision tasks. However, if you want to extract information related to images embedded in HTML or XML documents, you can use Beautiful Soup to parse the document and extract image-related data.



Here's a basic example of how to use Beautiful Soup to extract information about images from an HTML document:

1. Install Beautiful Soup if you haven't already:

bash

Copy code

```
pip install beautifulsoup4
```

1. Import the necessary libraries:

python

Copy code

```
from bs4 import BeautifulSoup
import requests
```

1. Fetch the HTML content of a webpage using the `requests` library:

```
python
```

 Copy code

```
url = "https://example.com" # Replace with the URL of the webpage you want
response = requests.get(url)
html_content = response.text
```

1. Parse the HTML content using BeautifulSoup:

```
python
```

 Copy code

```
soup = BeautifulSoup(html_content, 'html.parser')
```

1. Find and extract image tags (e.g., ``) from the parsed HTML:

```
python
```

 Copy code

```
image_tags = soup.find_all('img')

for img_tag in image_tags:
    src = img_tag.get('src') # Get the 'src' attribute of the image tag (URL)
    alt = img_tag.get('alt') # Get the 'alt' attribute (alternative text)
    print(f"Image source: {src}")
    print(f"Image alt text: {alt}")
```



how to form a GET request for an Object



To form a GET request for an object, you need to specify the HTTP method (which is GET in this case), the object's path, and the HTTP version. Additionally, you should include the "Host" header to indicate the host you're requesting the object from. Here's an example of how to form a GET request for an object:

python

Copy code

```
def create_get_request(host, path):
    request = f"GET {path} HTTP/1.1\r\n"
    request += f"Host: {host}\r\n"
    request += "\r\n"

    return request

# Usage
host = "example.com" # Replace with the actual host
path = "/path/to/your/object" # Replace with the path of the object you want
get_request = create_get_request(host, path)
print(get_request)
```

In this code, the `create_get_request` function takes the host (e.g., "example.com") and the object's path (e.g., "/path/to/your/object") as arguments and forms a valid GET request.

Make sure to replace "example.com" with the actual host and "/path/to/your/object" with the desired path of the object you want to retrieve.



Can you make a HTTP response for error code 404



Certainly! Here's how you can create an HTTP response message for a 404 (Not Found) error in Python:

python

 Copy code

```
def create_404_response():
    response = "HTTP/1.1 404 Not Found\r\n"
    response += "\r\n"
    response += "<html><body><h1>404 Not Found</h1></body></html>"

    return response

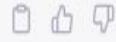
# Usage
error_404_response = create_404_response()
print(error_404_response)
```



How to make a non-persistent TCP connection using Request library in python



The `'requests'` library in Python is primarily designed to work with HTTP, which inherently uses a higher-level protocol on top of TCP. It does not directly handle low-level TCP socket connections. To establish a non-persistent TCP connection, you will need to work with lower-level socket programming using the `'socket'` library in Python.



Here's an example of how to create a non-persistent TCP connection in Python using the `'socket'` library:

Here's an example of how to create a non-persistent TCP connection in Python using the `socket` library:

```
python
import socket

# Define the server's address and port
server_address = ('example.com', 80)

# Create a socket and establish a connection
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(server_address)

# Send data to the server
request = b'GET / HTTP/1.0\r\n\r\n' # HTTP/1.0 with connection close
client_socket.send(request)

# Receive and print the response
response = client_socket.recv(4096)
print(response.decode())

# Close the socket connection
client_socket.close()
```



How to make parallel TCP connections using persistent HTTP and using socket in python



To establish parallel TCP connections using persistent HTTP with sockets in Python, you can create a script that sends multiple HTTP requests to a server and handles the responses concurrently. To do this, you'll need to use Python's `socket` library to manually manage the TCP connections. Here's an example of how you can achieve parallel TCP connections:

Here's a basic template for the modified client code:

```
python Copy code

import requests
from bs4 import BeautifulSoup
from concurrent.futures import ThreadPoolExecutor
import time

def fetch_object(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            # Process and save the object (e.g., image, script, icon)
            # You can measure the latency reduction here
            pass
    except Exception as e:
        print(f"Error fetching {url}: {e}")
```

```
def main():
    url = "http://example.com" # Replace with your target URL
    response = requests.get(url)

    if response.status_code == 200:
        # Parse the HTML content
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find and fetch objects in parallel
        objects = []
        for tag in soup.find_all(["img", "script", "link"]):
            if tag.get("src"):
                objects.append(tag["src"])

        # Start measuring time
        start_time = time.time()

        # Use ThreadPoolExecutor to fetch objects in parallel
        with ThreadPoolExecutor(max_workers=5) as executor:
            for obj in objects:
                executor.submit(fetch_object, obj)

        # Calculate the time taken
        end_time = time.time()
        latency_reduction = end_time - start_time
        print(f"Latency reduction: {latency_reduction} seconds")
    else:
        print("Error fetching HTML content")

if __name__ == "__main__":
    main()
```

Anti-Plagiarism statement

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. Additionally, we acknowledge that we may have used AI tools, such as language models (e.g., ChatGPT, Bard), for assistance in generating and refining my assignment, and we have made all reasonable efforts to ensure that such usage complies with the academic integrity policies set for the course. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, we understand our responsibility to report honour violations by other students if we become aware of it.

Names & Roll No:

1.Ashish B Emmanuel: CS23MTECH11004

2.Ashutosh Rajput: CS23MTECH11005

3.Keshvi Dharaiya: SM23MTECH14002

Date: 4th Nov. 2023

Signatures:

1.A.B.E

2.A.R

3.K.D