Ain Shams University

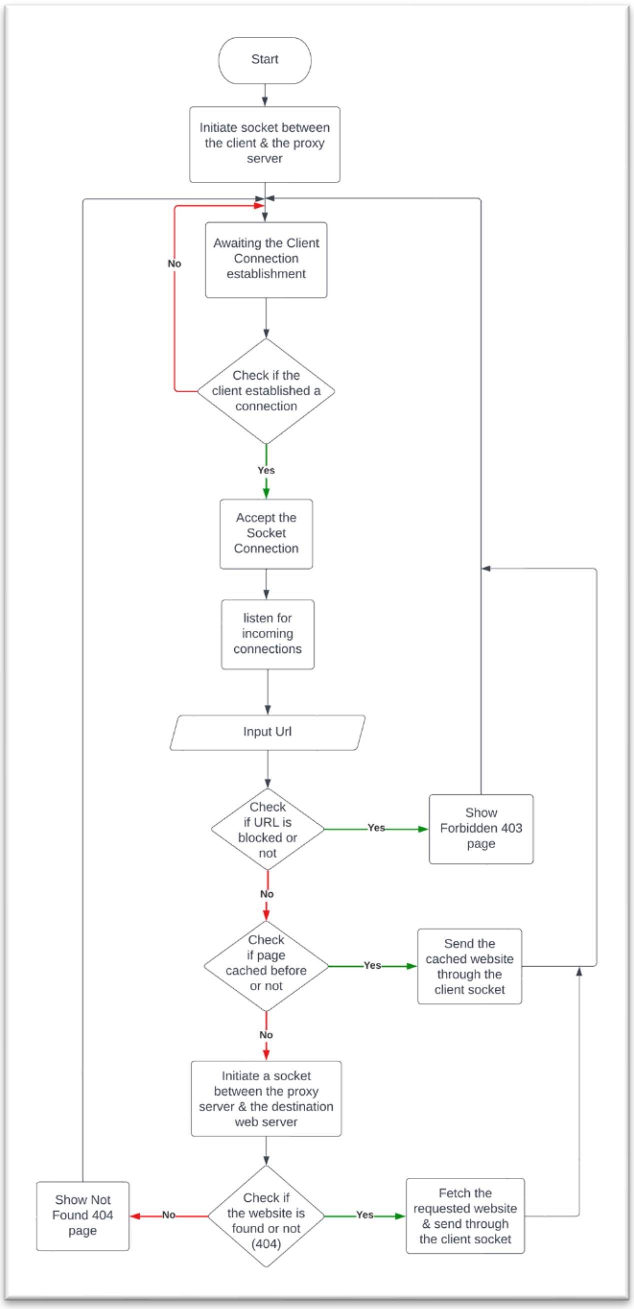
Faculty of Engineering

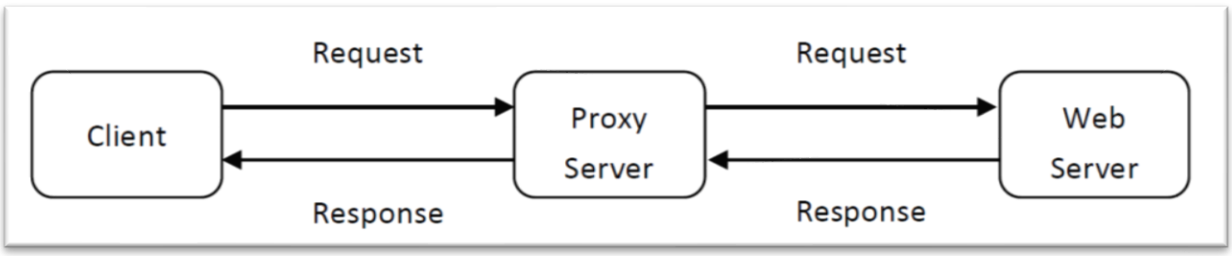
Computer Engineering and Software Systems Program

Ahmed Sameh Mohamed Mourad	19P5861
Elsaeed Ahmed Elsaeed Ali	19P1087
Osama Ali Mohamed Mohamed	19P6937

Table of Contents

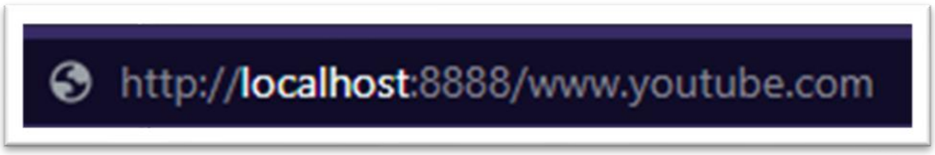
1.0 Design of the System:


Start
Initiate socket between the client & the proxy server
Awaiting the Client Connection establishment
No
Check if the client established a connection
Yes
Accept the Socket Connection
listen for incoming connections
Input Url
Check if URL is blocked or not
Yes
Show Forbidden 403 page
No
Check if page cached before or not
Yes
Send the cached website through the client socket
No
Initiate a socket between the proxy server & the destination web server
Check if the website is found or not (404)
No
Show Not Found 404 page
Yes
Fetch the requested website & send through the client socket



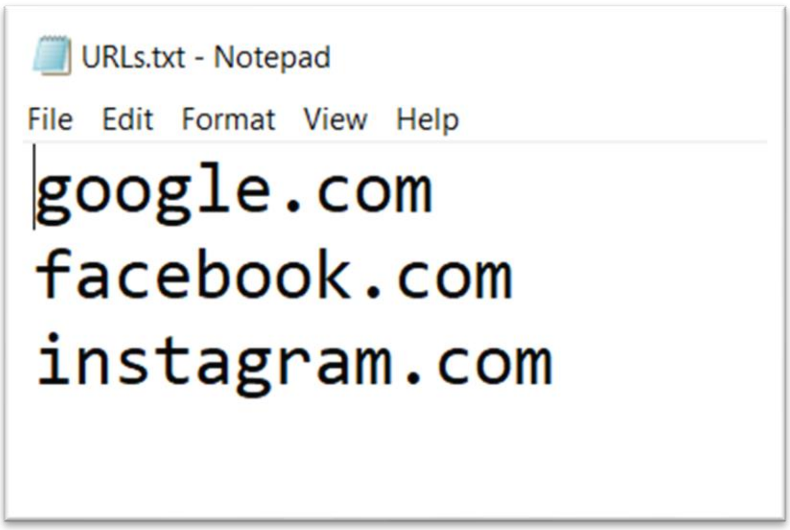
1.1 Client Part:

The client sends a request with a URL to Proxy Server.


http://localhost:8888/www.youtube.com

1.2 Proxy Server Part:

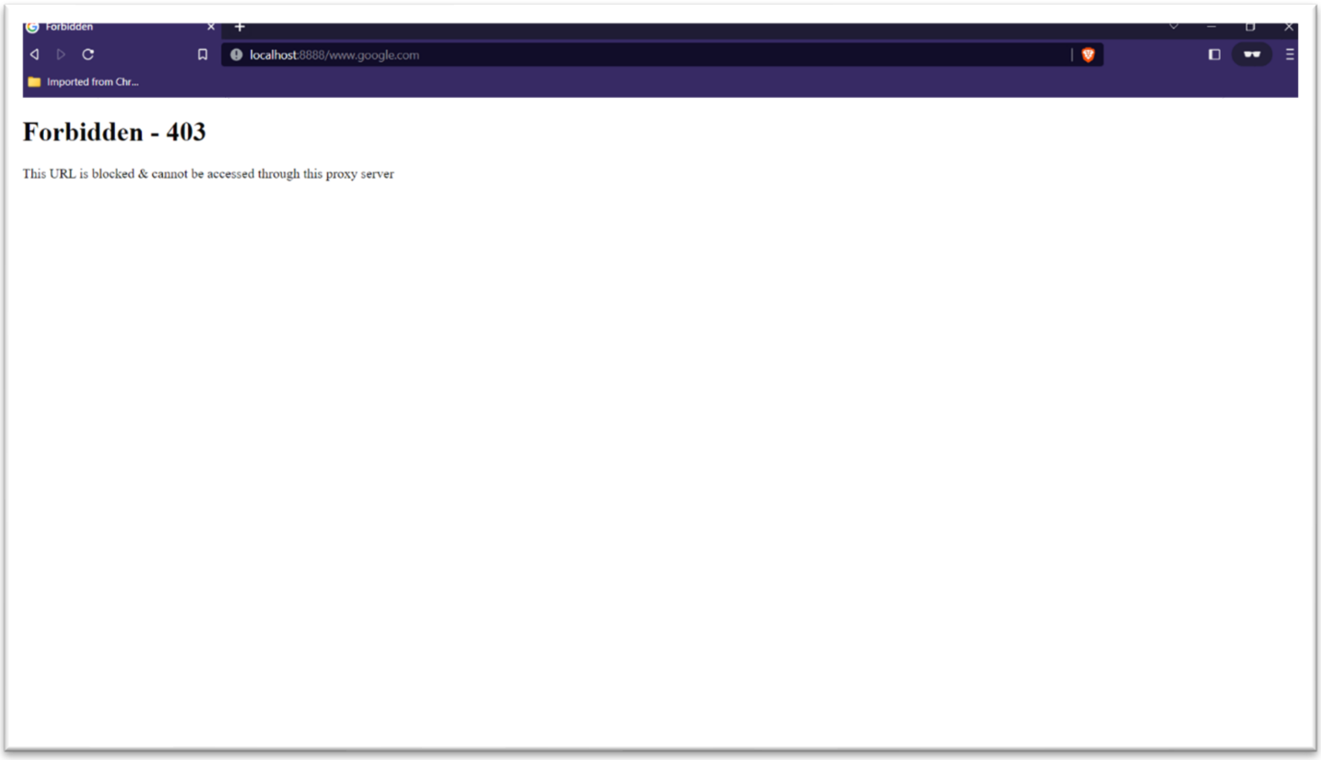
Firstly, the server checks if the URL is blocked or not. By comparing the requested URL with the blocked URLs that are stored in the “URLs.txt” file and seeing if it matches any URL in the URLs file.



```
google.com
facebook.com
instagram.com
```

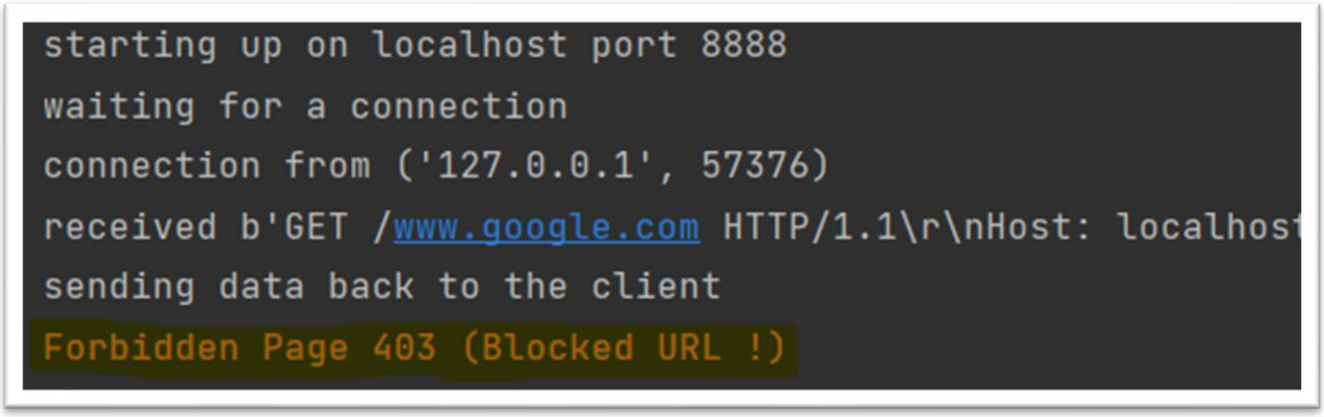
1.2.1 If it matches:

Then it displays a forbidden 403 page saying that the URL is blocked on the browser & displays the same message in the debug console.



Forbidden - 403

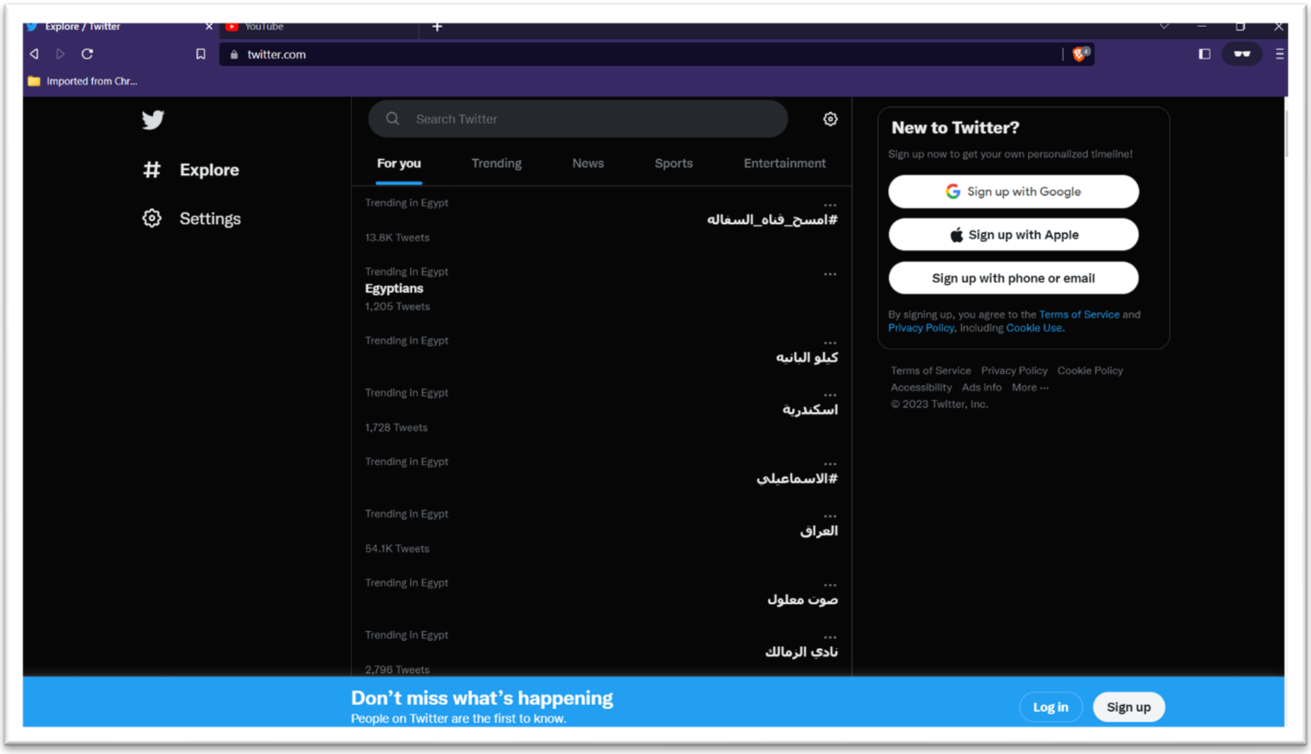
This URL is blocked & cannot be accessed through this proxy server

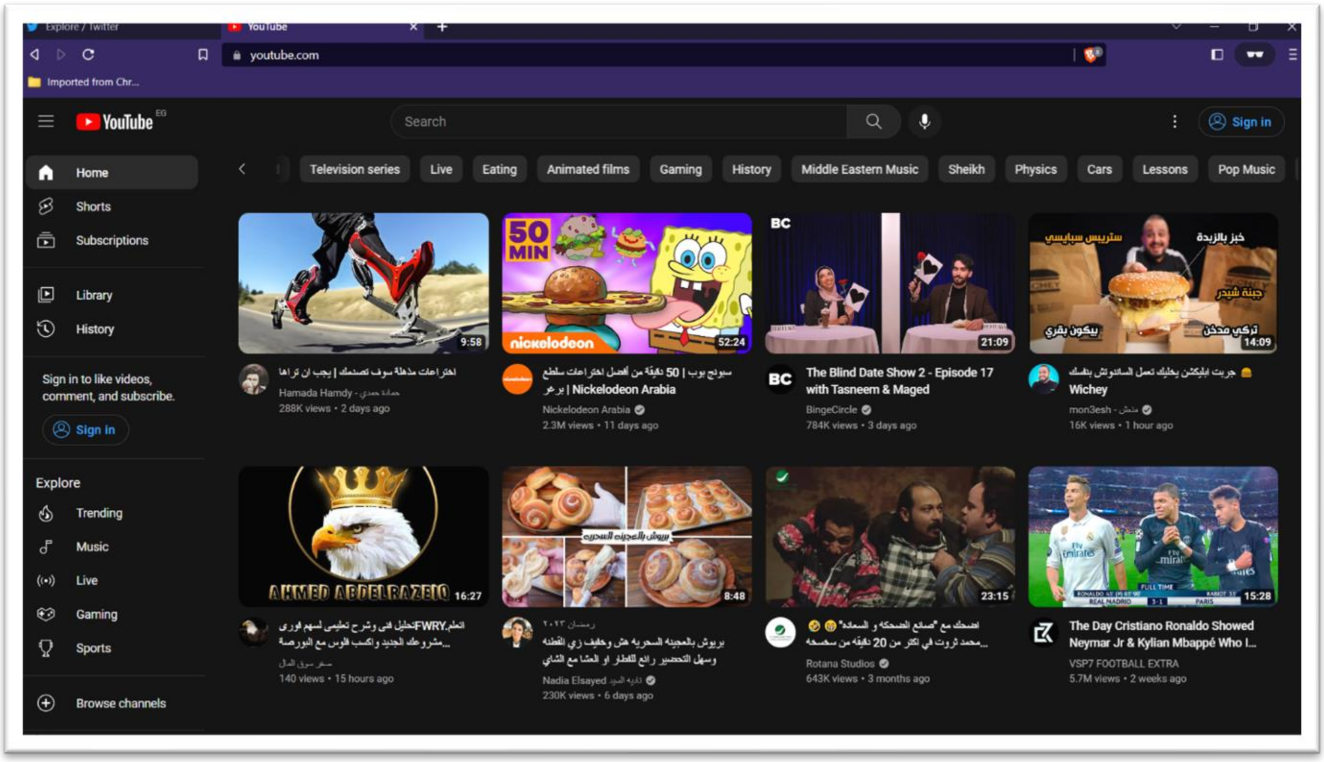


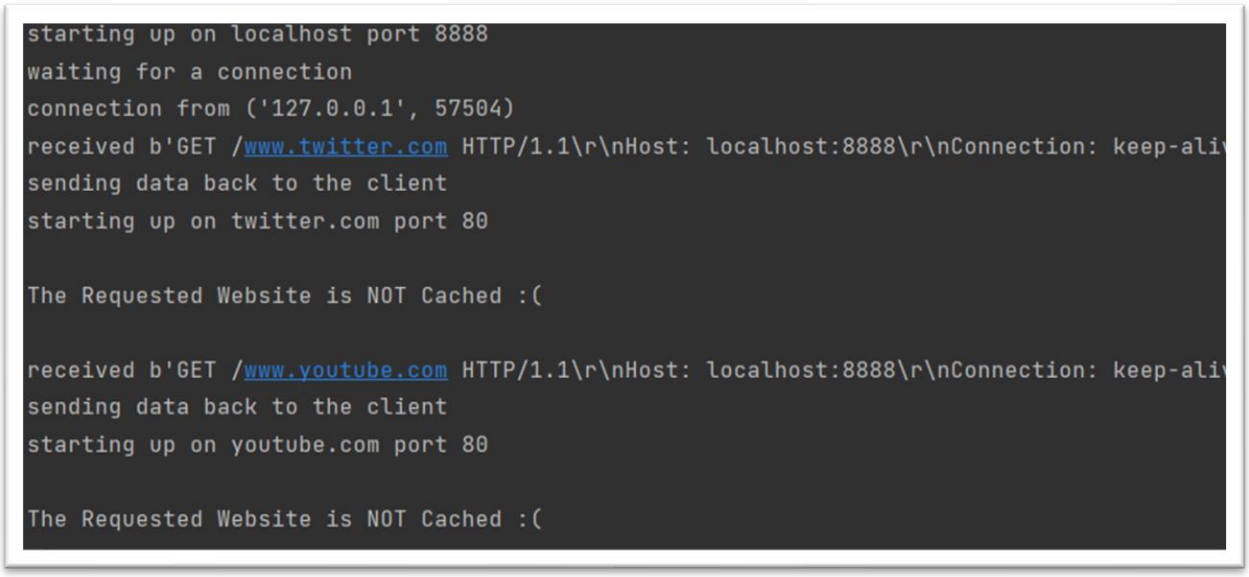
```
starting up on localhost port 8888
waiting for a connection
connection from ('127.0.0.1', 57376)
received b'GET /www.google.com HTTP/1.1\r\nHost: localhos
sending data back to the client
Forbidden Page 403 (Blocked URL !)
```

1.2.2 And if it doesn't match

Then it reaches the requested website on the browser.







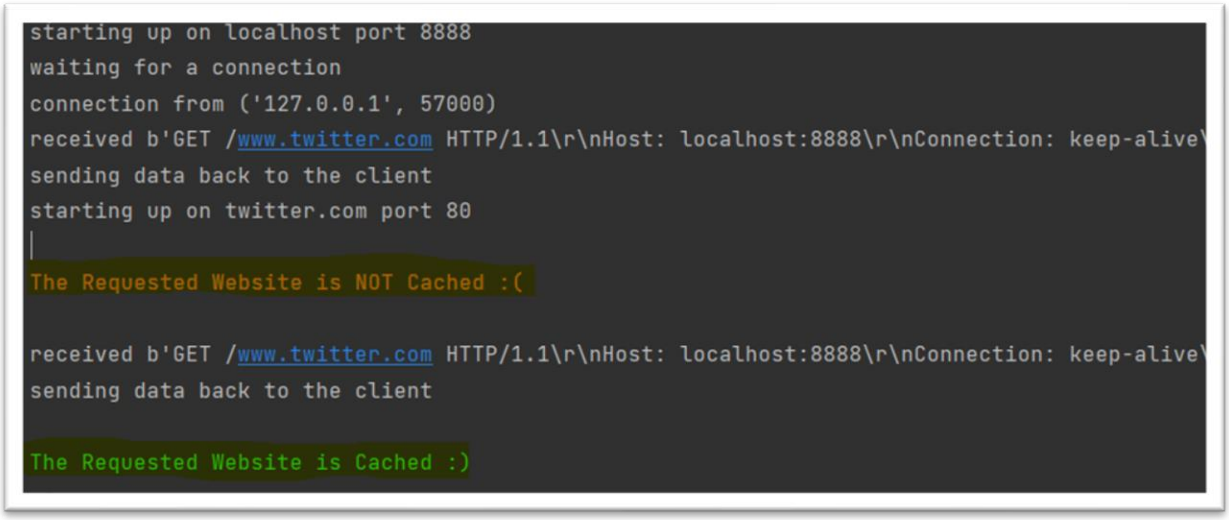
```
starting up on localhost port 8888
waiting for a connection
connection from ('127.0.0.1', 57504)
received b'GET /www.twitter.com HTTP/1.1\r\nHost: localhost:8888\r\nConnection: keep-aliv
sending data back to the client
starting up on twitter.com port 80

The Requested Website is NOT Cached :(

received b'GET /www.youtube.com HTTP/1.1\r\nHost: localhost:8888\r\nConnection: keep-aliv
sending data back to the client
starting up on youtube.com port 80

The Requested Website is NOT Cached :(
```

1.3 Caching:

Secondly, if the website wasn't blocked, it checks if the website is cached or not, then it sends the data back to the client through the connection socket.

Test Case for caching the valid twitter (www.twitter.com):

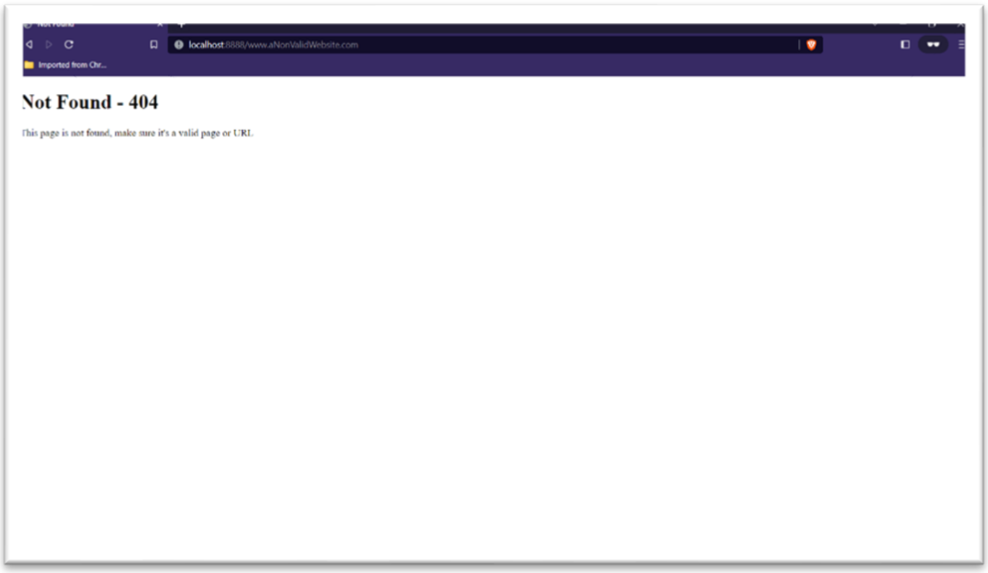
Upon opening the website for the first time, it's clear that it's not cached and gets loaded from the destination web server & sent to the client through the client socket. But the second time, it gets loaded from the cache of the proxy server.



```
starting up on localhost port 8888
waiting for a connection
connection from ('127.0.0.1', 57000)
received b'GET /www.twitter.com HTTP/1.1\r\nHost: localhost:8888\r\nConnection: keep-alive\
sending data back to the client
starting up on twitter.com port 80
|
The Requested Website is NOT Cached :(

received b'GET /www.twitter.com HTTP/1.1\r\nHost: localhost:8888\r\nConnection: keep-alive\
sending data back to the client

The Requested Website is Cached :)
```

1.4 validity of the website

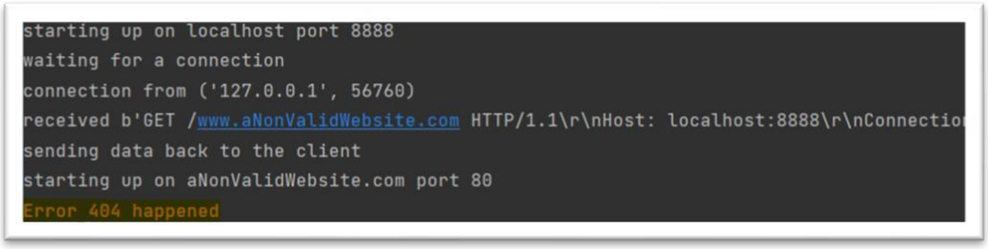
But before it checks if the website is cached or not, it first checks if this website is valid or not, if it's valid, it will send it to the client through the client socket & if it's not valid, then it will show a Not Found 404 page on the browser through the client socket.



localhost:8888/www.aNonValidWebsite.com

Not Found - 404

This page is not found, make sure it's a valid page or URL



```
starting up on localhost port 8888
waiting for a connection
connection from ('127.0.0.1', 56760)
received b'GET /www.aNonValidWebsite.com HTTP/1.1\r\nHost: localhost:8888\r\nConnectio
sending data back to the client
starting up on aNonValidWebsite.com port 80
Error 404 happened
```

2.0 Code:

2.1 Cache

```python
from website import Website


class Cache:
    cached_websites = dict()

    def add_website(self, website_url: str, response):
        self.cached_websites[website_url] = response

    def get_website(self, website_url: str):
        return self.cached_websites[website_url]

    def is_cached(self, website_url: str) -> Website:
        return Website(is_cached=True,
response=self.get_website(website_url=website_url)) if website_url in
self.cached_websites else Website(is_cached=False, response=None)
```

2.2 Error_handler

```python
import socket
import error_pages


class ErrorHandler:

    @staticmethod
    def forbidden(connection: socket):
        forbidden_url = bytes('HTTP/1.1 403 Forbidden\r\n' '\n' +
error_pages.forbidden_page + '\r\n', 'utf-8')

        # Send the forbidden page to the client via his socket
        connection.sendall(forbidden_url)

    @staticmethod
    def not_found(connection: socket):
        not_found_url = bytes('HTTP/1.1 404 Not Found\r\n' '\n' +
error_pages.not_found_page + '\r\n', 'utf-8')

        # Send the not found page to the client via his socket
        connection.sendall(not_found_url)
```

2.3 error_pages

```
forbidden_page = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Forbidden</title>
</head>
<body>
    <h1>Forbidden - 403</h1>
    <p>This URL is blocked & cannot be accessed through this proxy
server</p>
</body>
</html>
'''

not_found_page = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Not Found</title>
</head>
<body>
    <h1>Not Found - 404</h1>
    <p>This page is not found, make sure it's a valid page or URL</p>
</body>
</html>
'''
```

2.4 filter_result

```
class FilterResult:
    def __init__(self, is_blocked: bool, message: str):
        self.is_blocked = is_blocked
        self.message = message

    def get_result(self):
        return self
```

2.5 main

```python
from socket import *
from cache import Cache
from URLFilter import Filter
from error_handler import ErrorHandler

# Create a TCP/IP socket
sock = socket(AF_INET, SOCK_STREAM)

# Bind the socket to the port number 8888
server_address = ('localhost', 8888)
print('starting up on {} port {}'.format(*server_address))
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

while True:
    # Wait for a connection
    print('waiting for a connection')
    connection, client_address = sock.accept()
    try:
        print('connection from', client_address)

        # Receive the data in small chunks and retransmit it
        while True:
            data = connection.recv(4096)
            print('received {!r}'.format(data))
            cache = Cache()
            filter = Filter()

            if data:
                print('sending data back to the client')

                # Create a TCP/IP socket
                external_socket = socket(AF_INET, SOCK_STREAM)

                url = data.split()[1][5:].decode("utf-8")

                webUrl = filter.is_blocked(url)

                # Check if this website url is blocked or not
                if webUrl.is_blocked:
                    print('Forbidden Page 403 (Blocked URL !)')
                    # Show the forbidden page (403)
                    ErrorHandler.forbidden(connection=connection)
                    continue

                website = cache.is_cached(website_url=url)

                if website.is_cached:
                    print('The Requested Website is Cached :)')

                    # Sending the cached website to the client
                    connection.sendall(website.response)

                else:
                    server_address = (url, 80)
                    print('starting up on {} port
{}'.format(*server_address))

                    try:
                        # Connecting to the destination web server
                        external_socket.connect(server_address)

                        external_socket.sendall(bytes('GET /
HTTP/1.1\r\nHost: ' + url + '\r\nConnection: keep-alive\r\nCache-Control:
max-age=0\r\nsec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108",
"Brave";v="108"\r\nsec-ch-ua-mobile: ?0\r\nsec-ch-ua-platform:
"Windows"\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36\r\nAccept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
,image/apng,*/*;q=0.8\r\nSec-GPC: 1\r\nAccept-Language: en-US,en\r\nSec-
Fetch-Site: none\r\nSec-Fetch-Mode: navigate\r\nSec-Fetch-User: ?1\r\nSec-
Fetch-Dest: document\r\nAccept-Encoding: gzip, deflate, br\r\n\r\n', 'utf-
8'))

                        # Receiving the response from the destination web
server
                        external_response = external_socket.recv(4096)

                        # Forwarding the response to the client
                        connection.sendall(external_response)

                        # Closing the external socket
                        external_socket.close()

                        # Caching the un-cached website
                        cache.add_website(website_url=url,
response=external_response)
                    except gaierror:
                        print('Error 404 happened')

                        # Show the not found page (404)
                        ErrorHandler.not_found(connection=connection)

            else:
                print('no data from', client_address)
                break

    finally:
        # Clean up the connection
        connection.close()
```

2.6 URLFilter

```python
from filter_result import FilterResult


class Filter:

    def is_blocked(self, website_url: str) -> FilterResult:

        url_found_flag = False
        f = open("URLs.txt", "r")

        for x in f:
            x = x.replace("\n", "")
            if x.__eq__(website_url):
                url_found_flag = True
                break

        f.close()

        if url_found_flag:
            data = "This URL is blocked!!!!!"
            return FilterResult(is_blocked=True, message=data)

        else:
            data = "This URL is not blocked."
            return FilterResult(is_blocked=False, message=data)
```

2.7 class website

```python
class Website:
    def __init__(self, is_cached: bool, response):
        self.is_cached = is_cached
        self.response = response

    def get_website(self):
        return self
```

3.0 Important Links:

GitHub Link: https://github.com/OsamaLasheen/Proxy-Server

Online Link to this document: Document Link