ACN Programming Assignment-2: WWW

PART-1: A Simple Web Client

Explanation

The Client code is present in Client.py file. The python script is designed to download and render web content, including HTML, stylesheets, scripts, and images from a specified web server. It does this by employing a series of key components and functionalities:

Firstly, we import of various import statements, including modules such as `sys`, `os`, `webbrowser`, `requests`, and `BeautifulSoup`, to interact with the operating system, handle command-line arguments, open web pages in a browser, make HTTP requests, and parse HTML documents.

Global variables like `files` and `prefix` are employed to maintain a record of downloaded resources and provide a URL prefix for constructing full URLs when necessary.

The `parseArguments` function extracts essential information from the command-line arguments, including the server address, port number, and the target file path. The function also assigns arguments according to the user input as when the length of argument is two or three it makes a request directly to the server and for for or five it make request through the proxy server .The `prefix` variable is utilized to create complete URLs when additional host and port arguments are provided.It return address,port and filePath.

The `parseHTML` function plays the main role in saving and parsing the HTML content of the requested web page. It also handles the downloading of linked resources such as stylesheets, scripts, and images. We are using beautifulsoup to parse the html file . We are first creating a beautifulsoup object and then parsing the images, scripts and stylesheets url and then making a request to get the file and then download the file in the local system. This function takes two arguments, one is file path and the other is html response.

The script processes command-line arguments to identify the server address, port, and file path, and then proceeds to send an HTTP GET request to the specified URL, combining the server address, port, and file path.

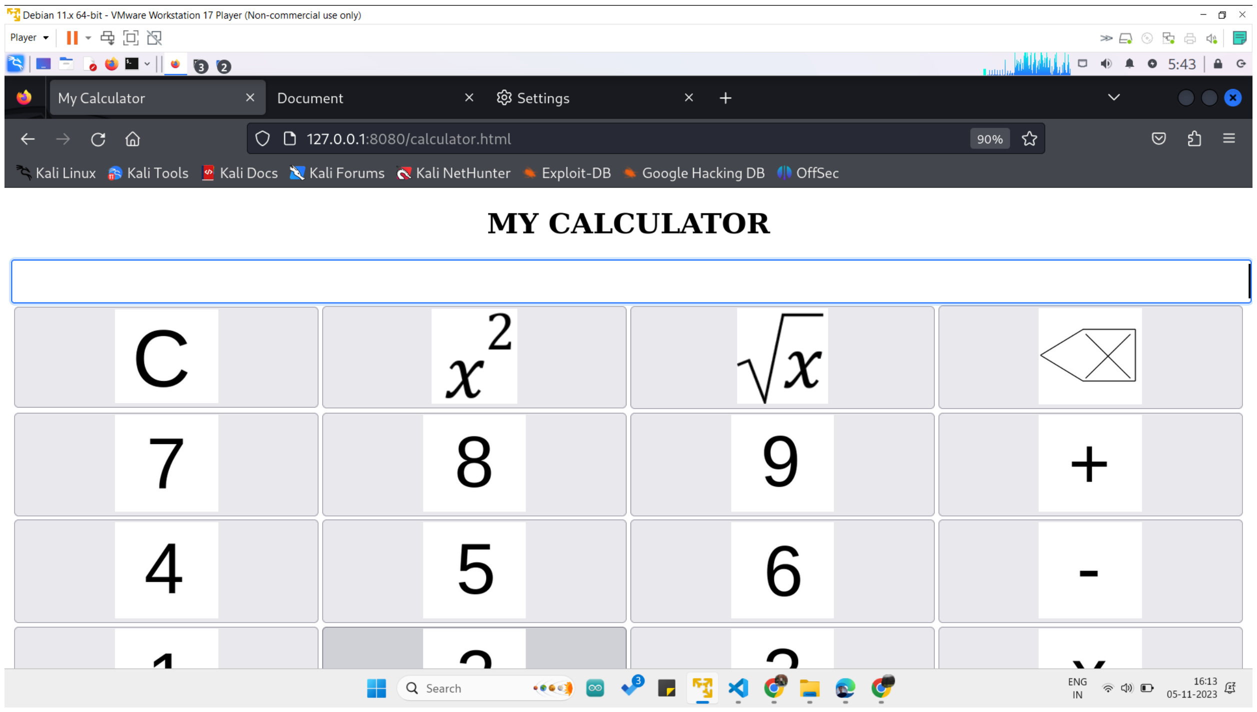
If the HTTP response status code is not 200 (OK), the script generates an "error.html" file and captures the response content as an error message.

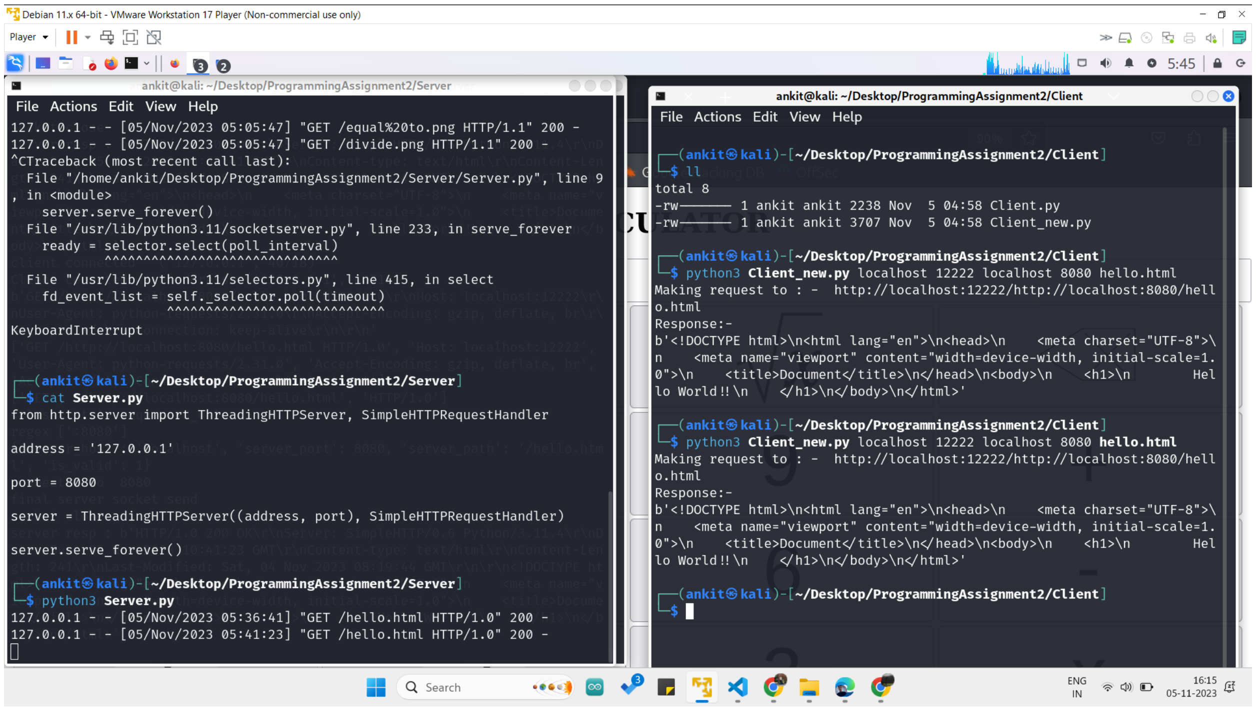
Once the web content is retrieved and processed, the script employs the `webbrowser` module to open the downloaded file in the default web browser.

Lastly, the script is designed to perform cleanup tasks by removing all downloaded files stored in the `files` list after the content has been displayed in the web browser.

In summary, this Python script works as a http client which makes requests to the server and shows the output in the browser to the user.

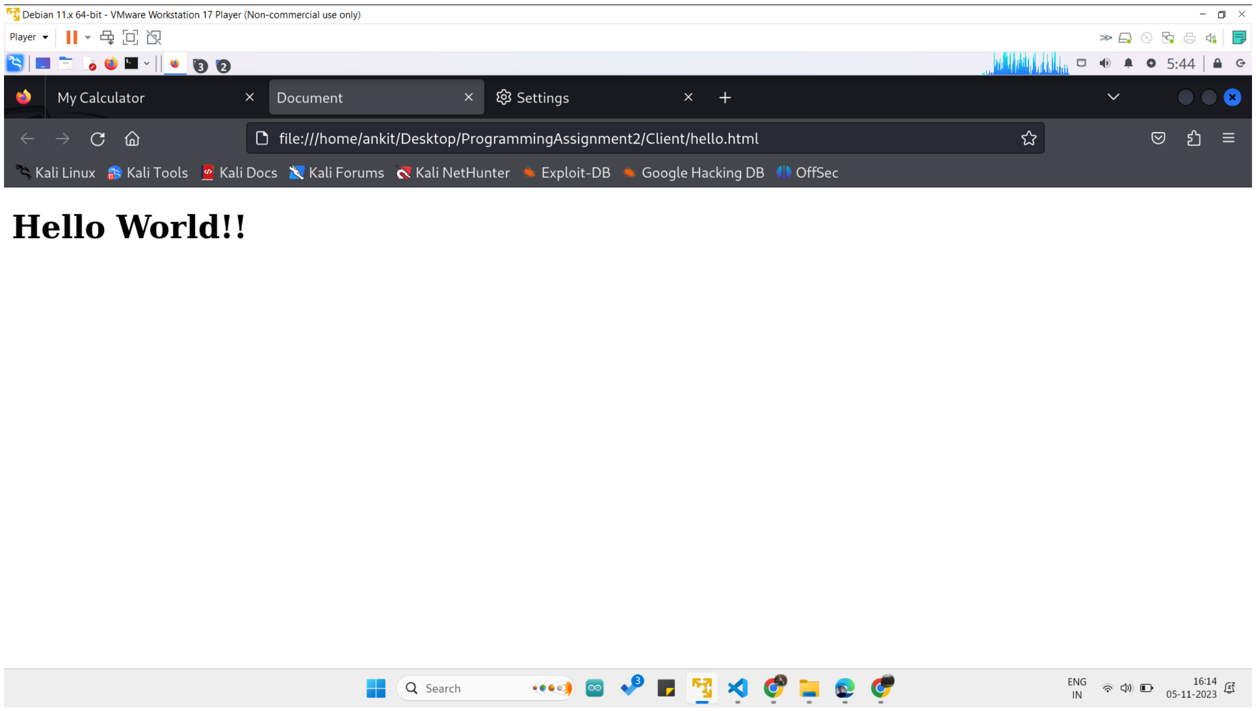
<u>Screenshot</u>




Debian 11.x 64-bit - VMware Workstation 17 Player (Non-commercial use only)
ankit@kali: ~/Desktop/ProgrammingAssignment2/Server
ankit@kali: ~/Desktop/ProgrammingAssignment2/Client
File Actions Edit View Help

```
127.0.0.1 - - [05/Nov/2023 05:05:47] "GET /equal%20to.png HTTP/1.1" 200 -
127.0.0.1 - - [05/Nov/2023 05:05:47] "GET /divide.png HTTP/1.1" 200 -
^CTraceback (most recent call last):
  File "/home/ankit/Desktop/ProgrammingAssignment2/Server/Server.py", line 9
, in <module>
    server.serve_forever()
  File "/usr/lib/python3.11/socketserver.py", line 233, in serve_forever
    ready = selector.select(poll_interval)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/selectors.py", line 415, in select
    fd_event_list = self._selector.poll(timeout)
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt

┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Server]
└─$ cat Server.py
from http.server import ThreadingHTTPServer, SimpleHTTPRequestHandler

address = '127.0.0.1'

port = 8080

server = ThreadingHTTPServer((address, port), SimpleHTTPRequestHandler)

server.serve_forever()

┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Server]
└─$ python3 Server.py
127.0.0.1 - - [05/Nov/2023 05:36:41] "GET /hello.html HTTP/1.0" 200 -
127.0.0.1 - - [05/Nov/2023 05:41:23] "GET /hello.html HTTP/1.0" 200 -
```

```
┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Client]
└─$ ll
total 8
-rw——————— 1 ankit ankit 2238 Nov  5 04:58 Client.py
-rw——————— 1 ankit ankit 3707 Nov  5 04:58 Client_new.py

┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Client]
└─$ python3 Client_new.py localhost 12222 localhost 8080 hello.html
Making request to : -  http://localhost:12222/http://localhost:8080/hell
o.html
Response:-
b'<!DOCTYPE html>\n<html lang="en">\n<head>\n    <meta charset="UTF-8">\
n    <meta name="viewport" content="width=device-width, initial-scale=1.
0">\n    <title>Document</title>\n</head>\n<body>\n    <h1>\n        Hel
lo World!!\n    </h1>\n</body>\n</html>'

┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Client]
└─$ python3 Client_new.py localhost 12222 localhost 8080 hello.html
Making request to : -  http://localhost:12222/http://localhost:8080/hell
o.html
Response:-
b'<!DOCTYPE html>\n<html lang="en">\n<head>\n    <meta charset="UTF-8">\
n    <meta name="viewport" content="width=device-width, initial-scale=1.
0">\n    <title>Document</title>\n</head>\n<body>\n    <h1>\n        Hel
lo World!!\n    </h1>\n</body>\n</html>'

┌──(ankit㉿kali)-[~/Desktop/ProgrammingAssignment2/Client]
└─$ 
```


Debian 11.x 64-bit - VMware Workstation 17 Player (Non-commercial use only)
My Calculator
Document
Settings
file:///home/ankit/Desktop/ProgrammingAssignment2/Client/hello.html
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

Hello World!!

PART-2: A Simple Web Proxy Server

Explanation

The provided code is a Python program that implements a simple HTTP proxy server. The proxy server listens on a specified IP address (127.0.0.1) and port (8081). It acts as an intermediary between a client and a web server, forwarding HTTP requests from clients to the target web server and then forwarding the web server's responses back to the client.

Firstly, we import various modules, including **'socket'** for socket communication, **'_thread'** for handling multiple client connections, **'ssl'** for secure connections, **'urllib.parse'** for URL parsing, and **'re'** for regular expressions.

This script establishes the `SERVER_IP` as '127.0.0.1' and `SERVER_PORT` as 8081, defining the IP address and port on which the proxy server will accept incoming connections.

`ProxyServer` class is defined to create and manage the proxy server. The constructor sets up the proxy server socket, binds it to the specified IP and port, and listens for incoming connections.

`forwardRequest` method handles forwarding HTTP requests to the target web server. It creates a new socket to connect to the web server, sends the client's request to the server, and then receives the server's response. It can handle secure (HTTPS) connections using SSL.

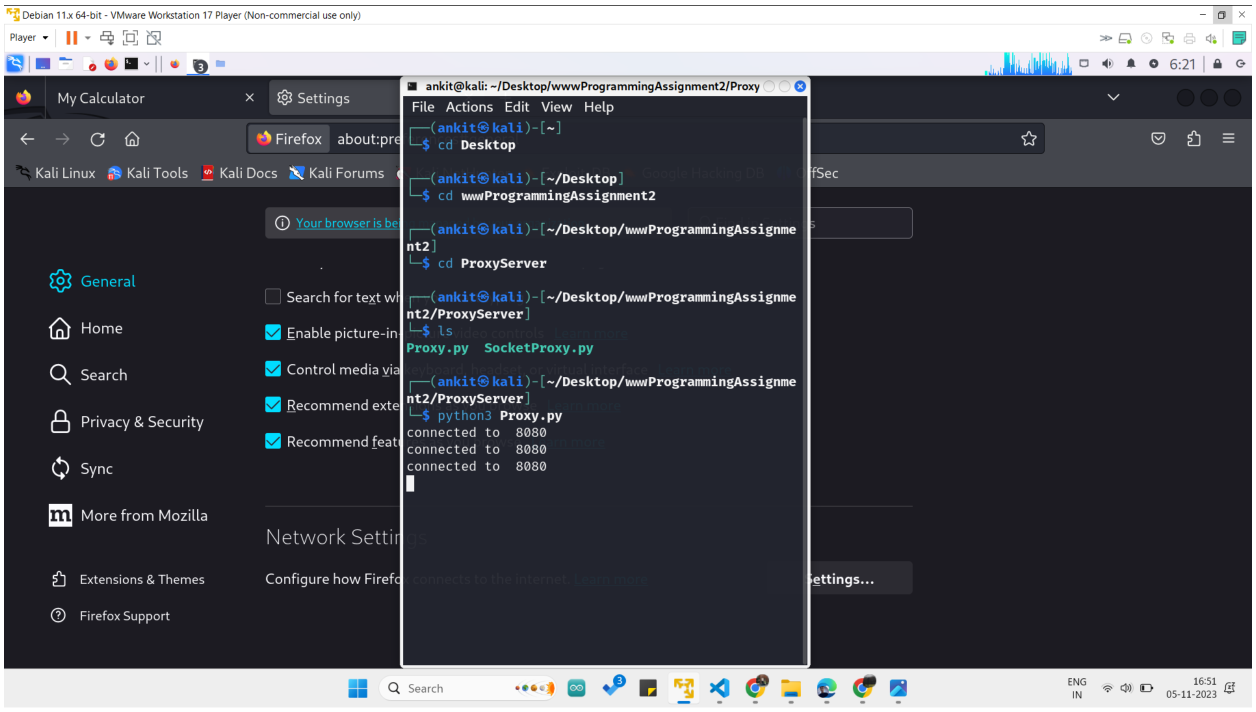
`parseRequest` method parses the client's HTTP request to extract the target server's hostname, port, and path. It uses the 'urllib.parse' module to break down the URL and determine whether it's an HTTP or HTTPS request. It also extracts the port number,hostname,path if specified in the URL.

`handleClientThead` method is used to handle individual client connections in separate threads. It receives the client's request, parses it, forwards it to the target server, and then sends the server's response back to the client.

`listenClient` method continuously listens for incoming client connections. When a client connects to the proxy server, a new thread is started to handle that client's request. This allows the proxy server to handle multiple clients concurrently.

The proxy server primarily works with HTTP requests and responses, forwarding data between the client and the target web server.

Screenshot



```
ankit@kali: ~/Desktop/wwwProgrammingAssignment2/Proxy
File  Actions  Edit  View  Help
┌──(ankit㉿kali)-[~]
└─$ cd Desktop

┌──(ankit㉿kali)-[~/Desktop]
└─$ cd wwwProgrammingAssignment2

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignme
nt2]
└─$ cd ProxyServer

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignme
nt2/ProxyServer]
└─$ ls
Proxy.py  SocketProxy.py

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignme
nt2/ProxyServer]
└─$ python3 Proxy.py
connected to  8080
connected to  8080
connected to  8080
```

PART-3: A Simple Web Server

Explanation

This Python script is a simple yet effective tool for creating a basic HTTP server using the `http.server` module. The server is configured to listen on a specific IP address, '127.0.0.1,' and port number 8080. Its primary function is to serve files from the local file system to clients who connect to it.

We begins with necessary import statements, bringing in key classes from the `**http.server**` module, including `**ThreadingHTTPServer**` and `**SimpleHTTPRequestHandler**`. The **ThreadingHTTPServer** is used to set up the HTTP server, while the **SimpleHTTPRequestHandler** serves as the default request handler for managing file serving.

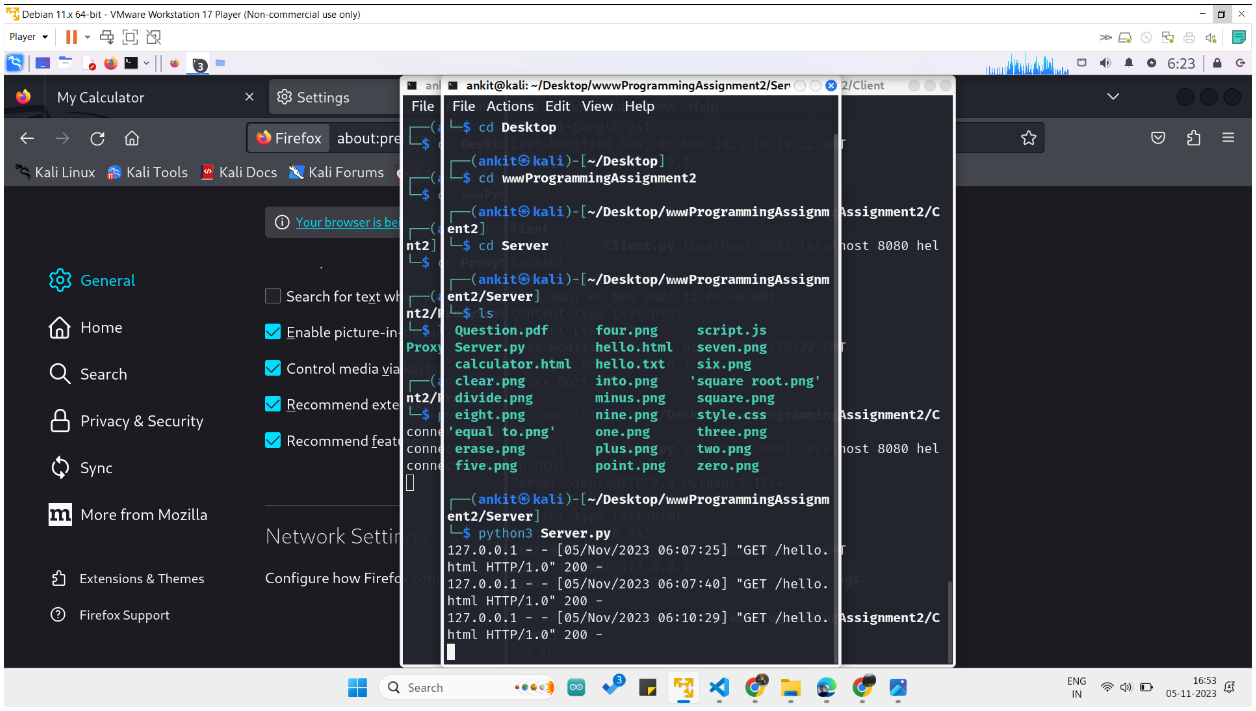
The configuration section specifies the IP address as '127.0.0.1' and the port as 8080. This designates that the server will exclusively accept connections from the local machine.

The script proceeds to create an instance of the `ThreadingHTTPServer` class, where the listening address and port are defined. Furthermore, the `SimpleHTTPRequestHandler` class is designated as the request handler.

The server functionality is initiated by calling the `server.serve_forever()` method. This step ensures the server runs continuously, persistently awaiting incoming HTTP requests from clients.

The core of the server's functionality is the handling of incoming HTTP GET requests by the `SimpleHTTPRequestHandler` class. When a client sends such a request, the handler endeavors to serve the requested file from the local file system, providing the file's content alongside essential HTTP headers.

Screenshot



```
ankit@kali: ~/Desktop/wwwProgrammingAssignment2/Ser
└─$ cd Desktop

┌──(ankit㉿kali)-[~/Desktop]
└─$ cd wwwProgrammingAssignment2

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignm
ent2]
└─$ cd Server

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignm
ent2/Server]
└─$ ls
 Question.pdf     four.png     script.js
 Server.py        hello.html   seven.png
 calculator.html  hello.txt    six.png
 clear.png        into.png    'square root.png'
 divide.png       minus.png    square.png
 eight.png        nine.png     style.css
'equal to.png'    one.png      three.png
 erase.png        plus.png     two.png
 five.png         point.png    zero.png

┌──(ankit㉿kali)-[~/Desktop/wwwProgrammingAssignm
ent2/Server]
└─$ python3 Server.py
127.0.0.1 - - [05/Nov/2023 06:07:25] "GET /hello.
html HTTP/1.0" 200 -
127.0.0.1 - - [05/Nov/2023 06:07:40] "GET /hello.
html HTTP/1.0" 200 -
127.0.0.1 - - [05/Nov/2023 06:10:29] "GET /hello.
html HTTP/1.0" 200 -
```

PART-4: Extensions to Simple Proxy Server or web client/web server

Explanation

This Python script represents a fundamental implementation of an HTTP proxy server. The proxy server is configured to listen on a specified IP address ('127.0.0.1') and port (8081). Its primary function is to receive incoming HTTP requests from clients, relay these requests to the specified target server indicated in the client's request, and subsequently forward the target server's response back to the original client. The script aims to provide a straightforward HTTP proxy server for managing web requests.

The code begins by importing essential modules for network operations. These modules include socket-related functionality (`**socket**`, `**AF_INET**`, `**SOCK_STREAM**`, `**SOL_SOCKET**`, `**SO_REUSEADDR**`), the `**start_new_thread**` module for handling multiple client connections concurrently, `ssl` for secure communication, `urllib.parse` for URL parsing and manipulation, `englisttohindi` for English to Hindi translation, and `BeautifulSoup` for HTML parsing,datetime of date,matplotlib for graph creation.

The script establishes the `SERVER_IP` as '127.0.0.1' and `SERVER_PORT` as 8081, defining the IP address and port on which the proxy server will accept incoming connections.

An instance of the `ProxyServer` class is instantiated, initializing the proxy server to listen on the specified IP address and port

forwardRequest method manages the forwarding of the client's request to the target server specified in the request. It establishes a connection to the target server and sends the client's request. For secure communication (HTTPS, port 443), it ensures secure socket wrapping using SSL. The method retrieves and returns the response from the target server for transmission back to the client.

saveUserData is used to save the data of the user in the json file which is used to create the graph

plotGraph is used to create a graph using matplotlib and saving it in the local folder. It takes list inputs for the x and y axis and name of the file to be saved.

createGraphData is used to create data from the json file and use the data for making graph. The data created are for no of request made today, in a week, in a month, this method also call plotGraph method to create the graph

Translate method is used to change the english text to hindi text using the englisttohindi module and BeautifulSoup to parse the HTML content . This method loops over all tags and replace the english text with hindi text

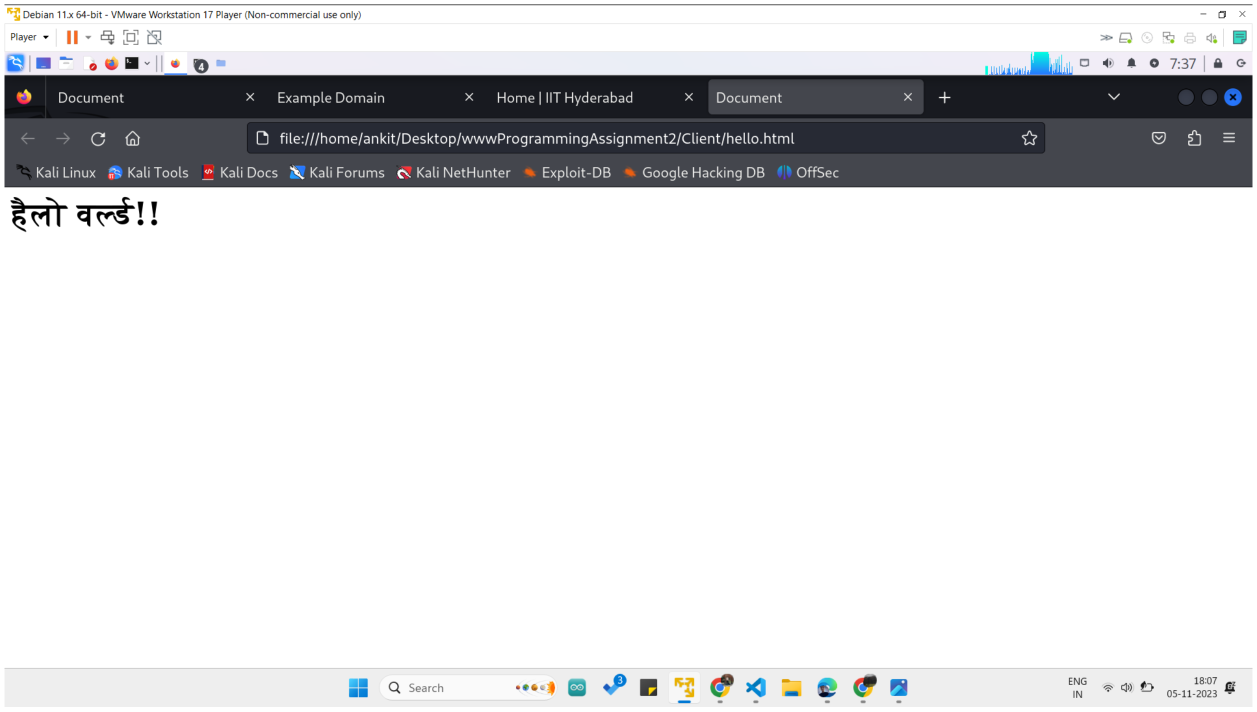
handleClient method is invoked in a separate thread for each client connection. It takes charge of receiving the client's request, parsing it, and forwarding it to the target server through the `makeRequestToFinalServer` method. After receiving the response from the target server, it is sent back to the client.

parseRequest method parses the client's HTTP request to extract the target server's hostname, port, and path. It uses the 'urllib.parse' module to break down the URL and determine whether it's an HTTP or HTTPS request. It also extracts the port number,hostname,path if specified in the URL.

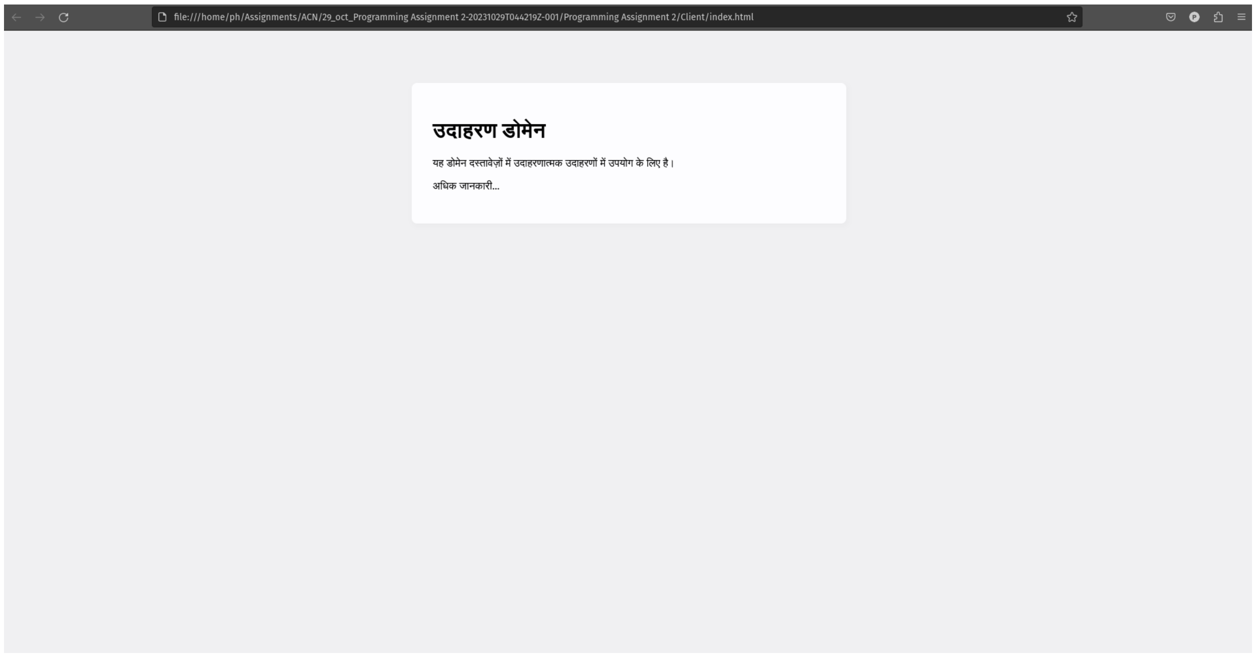
Responsible for parsing the client's HTTP request, this method extracts vital details like the target server's host, port, and path. It also validates the request and determines if it's an HTTPS request. The parsed information is encapsulated in a dictionary.

listenClient method listens for incoming client connections and spawns a new thread for each client connection. It is the entry point for client interaction.

Screenshot



हैलो वर्ल्ड!!

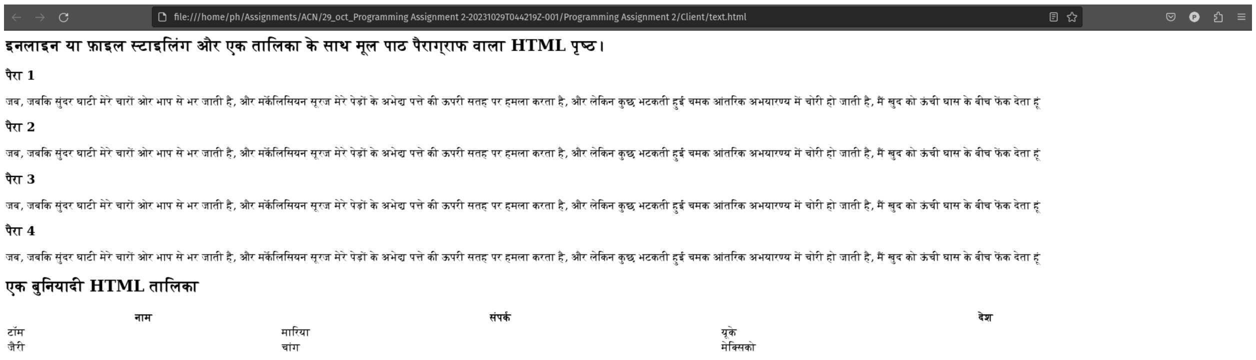


file:///home/ph/Assignments/ACN/29_oct_Programming Assignment 2-20231029T044219Z-001/Programming Assignment 2/Client/index.html

उदाहरण डोमेन

यह डोमेन दस्तावेज़ों में उदाहरणात्मक उदाहरणों में उपयोग के लिए है।

अधिक जानकारी...



file:///home/ph/Assignments/ACN/29_oct_Programming Assignment 2-20231029T044219Z-001/Programming Assignment 2/Client/text.html

इनलाइन या फ़ाइल स्टाइलिंग और एक तालिका के साथ मूल पाठ पैराग्राफ वाला HTML पृष्ठ।

पैरा 1

जब, जबकि सुंदर घाटी मेरे चारों ओर भाप से भर जाती है, और मेर्कलिसियन सूरज मेरे पेड़ों के अभेद्य पत्ते की ऊपरी सतह पर हमला करता है, और लेकिन कुछ भटकती हुई चमक आंतरिक अभयारण्य में चोरी हो जाती है, मैं खुद को ऊंची घास के बीच फेंक देता हूं

पैरा 2

जब, जबकि सुंदर घाटी मेरे चारों ओर भाप से भर जाती है, और मेर्कलिसियन सूरज मेरे पेड़ों के अभेद्य पत्ते की ऊपरी सतह पर हमला करता है, और लेकिन कुछ भटकती हुई चमक आंतरिक अभयारण्य में चोरी हो जाती है, मैं खुद को ऊंची घास के बीच फेंक देता हूं

पैरा 3

जब, जबकि सुंदर घाटी मेरे चारों ओर भाप से भर जाती है, और मेर्कलिसियन सूरज मेरे पेड़ों के अभेद्य पत्ते की ऊपरी सतह पर हमला करता है, और लेकिन कुछ भटकती हुई चमक आंतरिक अभयारण्य में चोरी हो जाती है, मैं खुद को ऊंची घास के बीच फेंक देता हूं

पैरा 4

जब, जबकि सुंदर घाटी मेरे चारों ओर भाप से भर जाती है, और मेर्कलिसियन सूरज मेरे पेड़ों के अभेद्य पत्ते की ऊपरी सतह पर हमला करता है, और लेकिन कुछ भटकती हुई चमक आंतरिक अभयारण्य में चोरी हो जाती है, मैं खुद को ऊंची घास के बीच फेंक देता हूं

एक बुनियादी HTML तालिका

नाम	संपर्क	देश
टॉम	मारिया	यूके
जैरी	चांग	मेक्सिको

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 : Ankit Kumar,Pramod Hembrom,Raghavendra Kulkarni
Date: 05/11/2023
Signatures: Ankit Kumar,Pramod Hembrom,Raghavendra Kulkarni

References:
https://www.w3schools.com/python/matplotlib_bars.asp
https://pypi.org/project/englisttohindi/
https://www.geeksforgeeks.org/python-find-text-using-beautifulsoup-then-replace-in-original-soup-variable/
https://docs.python-requests.org/en/v1.1.0/
https://www.internalpointers.com/post/making-http-requests-sockets-python
https://docs.python.org/3/library/http.server.html
https://www.geeksforgeeks.org/socket-programming-multi-threading-python/