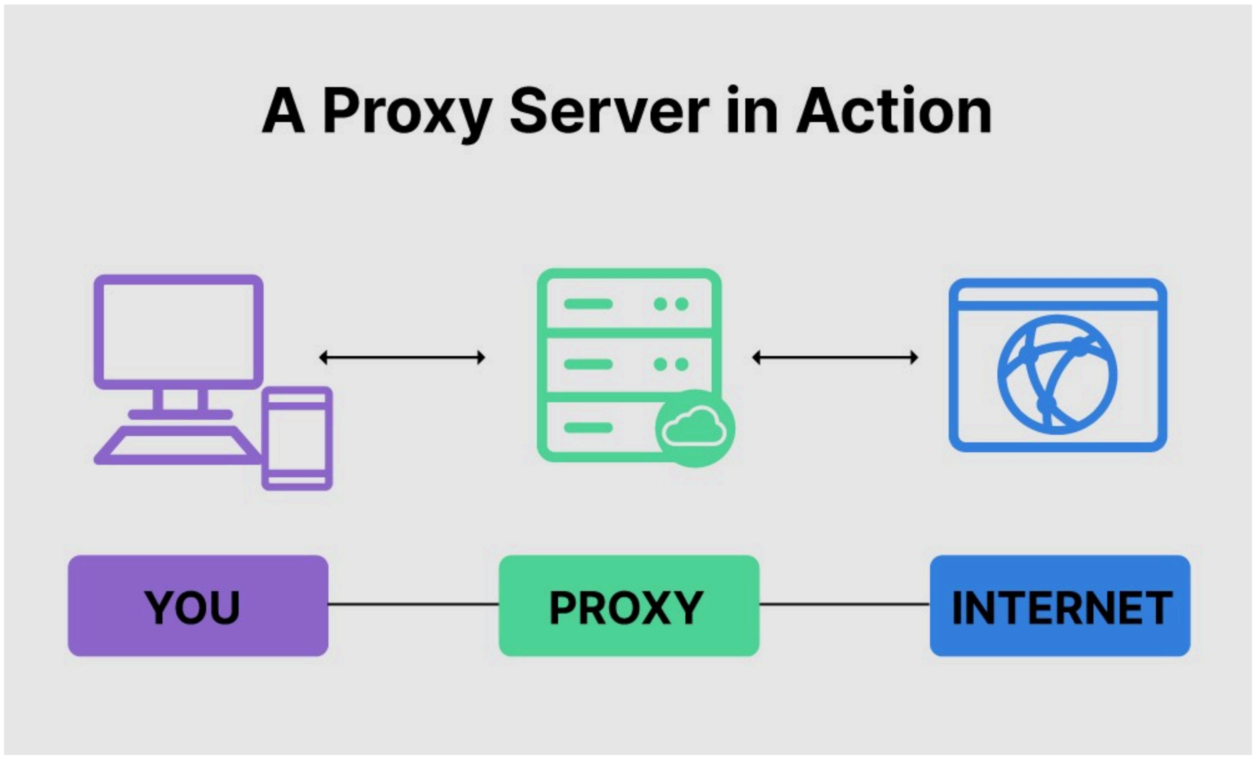
CSL2090 Principles of Computer System-II
Project Report on Proxy Server

Group Members

Shruti Chaudhary	B22AI037
Gouri Patidar	B22AI020

Proxy Server

A proxy server acts as an intermediary between clients such as web browsers or other applications and other servers such as web servers or FTP servers. It receives requests from clients seeking resources or services and forwards those requests to the appropriate servers. When the servers respond, the proxy server then sends the response back to the clients.



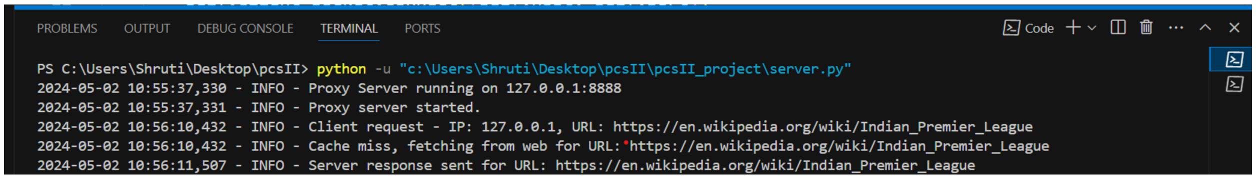
Key Functions of proxy servers:

- Proxy servers can mask the IP addresses of clients, providing anonymity and privacy by hiding the client's original IP address from the server they are communicating with.
- Proxy servers can cache frequently accessed web pages, images, and other content locally. When clients request cached content, the proxy server can serve it directly without needing to fetch it from the original server.

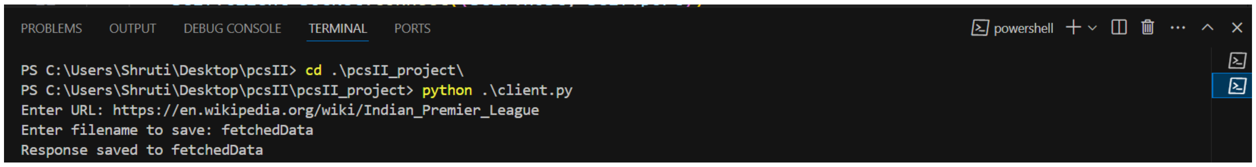
- Proxy servers can distribute incoming client requests across multiple backend servers, balancing the load to ensure optimal resource utilization and prevent individual servers from being overwhelmed.
- Proxy servers can serve as a barrier between internal networks and external networks,they can enforce security policies, inspect and filter traffic for threats, and provide an additional layer of defense for network security.

How does it work?

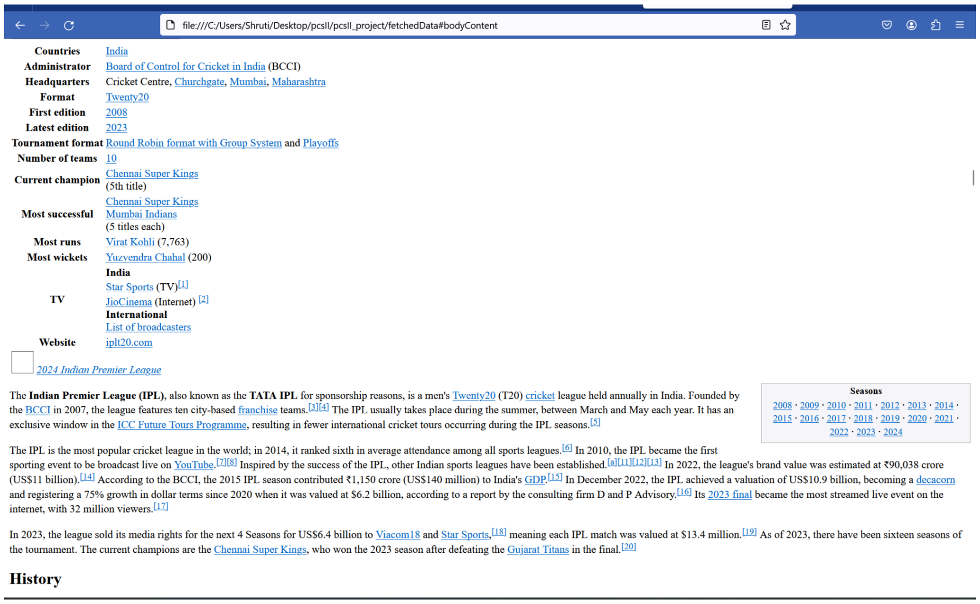
- Run the proxy_server.py script to start the proxy server on a specified host and port.
- Run the proxy_client.py script and provide a URL as input.
- The client sends an HTTP GET request to the proxy server for the specified URL.
- The proxy server fetches the requested URL from the web (or cache) and sends the response back to the client.
- The client receives the response, saves it to a local HTML file, and opens it in the default web browser for viewing.



```
PS C:\Users\Shruti\Desktop\pcsII> python -u "c:\Users\Shruti\Desktop\pcsII\pcsII_project\server.py"
2024-05-02 10:55:37,330 - INFO - Proxy Server running on 127.0.0.1:8888
2024-05-02 10:55:37,331 - INFO - Proxy server started.
2024-05-02 10:56:10,432 - INFO - Client request - IP: 127.0.0.1, URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 10:56:10,432 - INFO - Cache miss, fetching from web for URL:"https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 10:56:11,507 - INFO - Server response sent for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
```



```
PS C:\Users\Shruti\Desktop\pcsII> cd .\pcsII_project\
PS C:\Users\Shruti\Desktop\pcsII\pcsII_project> python .\client.py
Enter URL: https://en.wikipedia.org/wiki/Indian_Premier_League
Enter filename to save: fetchedData
Response saved to fetchedData
```



file:///C:/Users/Shruti/Desktop/pcsII/pcsII_project/fetchedData#bodyContent

Countries	India
Administrator	Board of Control for Cricket in India (BCCI)
Headquarters	Cricket Centre, Churchgate, Mumbai, Maharashtra
Format	Twenty20
First edition	2008
Latest edition	2023
Tournament format	Round Robin format with Group System and Playoffs
Number of teams	10
Current champion	Chennai Super Kings (5th title)
Most successful	Chennai Super Kings Mumbai Indians (5 titles each)
Most runs	Virat Kohli (7,763)
Most wickets	Yuzvendra Chahal (200)
TV	**India** Star Sports (TV)[1] JioCinema (Internet) [2] **International** List of broadcasters
Website	iplt20.com

2024 Indian Premier League

Seasons
2008 · 2009 · 2010 · 2011 · 2012 · 2013 · 2014 · 2015 · 2016 · 2017 · 2018 · 2019 · 2020 · 2021 · 2022 · 2023 · 2024

The **Indian Premier League (IPL)**, also known as the **TATA IPL** for sponsorship reasons, is a men's Twenty20 (T20) cricket league held annually in India. Founded by the BCCI in 2007, the league features ten city-based franchise teams.[3][4] The IPL usually takes place during the summer, between March and May each year. It has an exclusive window in the ICC Future Tours Programme, resulting in fewer international cricket tours occurring during the IPL seasons.[5]

The IPL is the most popular cricket league in the world; in 2014, it ranked sixth in average attendance among all sports leagues.[6] In 2010, the IPL became the first sporting event to be broadcast live on YouTube.[7][8] Inspired by the success of the IPL, other Indian sports leagues have been established.[a][11][12][13] In 2022, the league's brand value was estimated at ₹90,038 crore (US$11 billion).[14] According to the BCCI, the 2015 IPL season contributed ₹1,150 crore (US$140 million) to India's GDP.[15] In December 2022, the IPL achieved a valuation of US$10.9 billion, becoming a decacorn and registering a 75% growth in dollar terms since 2020 when it was valued at $6.2 billion, according to a report by the consulting firm D and P Advisory.[16] Its 2023 final became the most streamed live event on the internet, with 32 million viewers.[17]

In 2023, the league sold its media rights for the next 4 Seasons for US$6.4 billion to Viacom18 and Star Sports,[18] meaning each IPL match was valued at $13.4 million.[19] As of 2023, there have been sixteen seasons of the tournament. The current champions are the Chennai Super Kings, who won the 2023 season after defeating the Gujarat Titans in the final.[20]

History

Server-Side Features of the Proxy Server:

Here is the more detailed description of each feature that we have incorporated in our proxy server.

1. Caching mechanism

Overview:

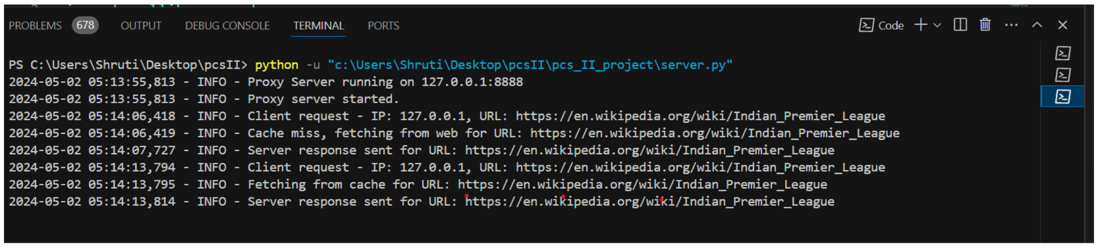
- The server implements a caching mechanism to store responses from remote servers temporarily.

- Each cached response is stored along with a timestamp indicating when it was cached.
- When a request is made for a URL, the server checks if the response is available in the cache and if it's still within the cache expiry time. If yes, it serves the cached response; otherwise, it fetches a fresh response from the web.

<u>Implementation Details:</u>

- The proxy server stores cached responses in a dictionary (self.cache), where the key is the URL of the requested resource, and the value is an instance of the CachedResponse class.
- Each CachedResponse object contains the response data (data) and a timestamp (timestamp) indicating when the response was cached.
- Cached responses have an expiry time set to 60 seconds. If the cached response exceeds this time limit, it is considered expired and will not be served to clients.
- When handling a client request, the server checks if a cached response exists for the requested URL and if its timestamp is within the expiry time.
- If a cached response exists for the requested URL and its timestamp is within the expiry time, it is considered a cache hit. The server fetches the response data from the cache and serves it to the client.
- If there is no cached response for the requested URL or if the cached response has expired, it is considered a cache miss. In this case, the server fetches the response data from the web server and caches it for future use.
- When a fresh response is fetched from the web server, the server updates the cache by storing the response data along with the current timestamp.
- This ensures that subsequent requests for the same URL within the cache expiry time will be served from the cache, reducing the need to fetch the same content repeatedly from the web server.

Given below is the message logs of the proxy server showing cache hits and misses:



```
PS C:\Users\Shruti\Desktop\pcsII> python -u "c:\Users\Shruti\Desktop\pcsII\pcs_II_project\server.py"
2024-05-02 05:13:55,813 - INFO - Proxy Server running on 127.0.0.1:8888
2024-05-02 05:13:55,813 - INFO - Proxy server started.
2024-05-02 05:14:06,418 - INFO - Client request - IP: 127.0.0.1, URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:14:06,419 - INFO - Cache miss, fetching from web for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:14:07,727 - INFO - Server response sent for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:14:13,794 - INFO - Client request - IP: 127.0.0.1, URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:14:13,795 - INFO - Fetching from cache for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:14:13,814 - INFO - Server response sent for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
```

<u>**2. Rate Limiting:**</u>

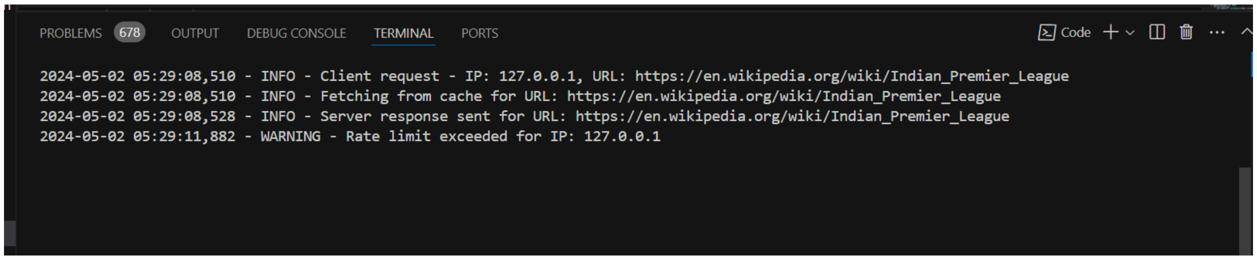
<u>Overview:</u>

- The server limits the number of requests allowed per IP address within a specified time interval (e.g., 10 requests per IP per minute).
- It tracks the number of requests made by each IP address and resets the count after the specified interval.
- If a client exceeds the allowed number of requests within the interval, the server closes the connection and logs a warning

Implementation Details:

- The proxy server enforces rate limiting based on the maximum number of requests allowed per IP address within a specified time interval,in our case it is 5 requests per minute.
- The server tracks the number of requests made by each client IP address using a dictionary, where the key is the IP address, and the value is the count of requests made.
- This dictionary is initialized and updated within the handle_client() method, which is responsible for processing client requests
- The server resets the request count for each IP address if the specified time interval has passed since the last request was processed.
- If a client exceeds the maximum number of requests allowed within the time interval, the server logs a warning message indicating that the rate limit has been exceeded for the client's IP address.
- Subsequently, the server closes the connection with the client, preventing further requests from being processed until the next time interval begins.

Given below is the message logs when rate limit exceeded:



```
2024-05-02 05:29:08,510 - INFO - Client request - IP: 127.0.0.1, URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:29:08,510 - INFO - Fetching from cache for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:29:08,528 - INFO - Server response sent for URL: https://en.wikipedia.org/wiki/Indian_Premier_League
2024-05-02 05:29:11,882 - WARNING - Rate limit exceeded for IP: 127.0.0.1
```

3. Thread Pooling

Overview

Thread pooling is introduced to improve concurrency management and resource utilization. The thread pool limits the number of active threads, preventing resource exhaustion and reducing the overhead of creating and destroying threads. This change enhances server stability and scalability.

Implementation Details

- **Thread Pool Size:** A ThreadPoolExecutor is used to create a pool of threads with a maximum size of 5. This means that no more than 5 tasks can be executed concurrently. Tasks submitted to the thread pool are either executed immediately if a thread is available or queued if the pool is at full capacity.
- **Concurrency Control:** The proxy server submits client handling tasks to the thread pool, allowing it to manage multiple client requests concurrently. This approach reduces the risk of resource exhaustion and improves efficiency by reusing threads.
- **Handling Excessive Requests:** If the thread pool reaches its capacity, additional tasks are queued until a thread becomes available. This helps maintain stability by preventing the server from creating too many threads.

Benefits

- Resource Management: Thread pooling reduces the overhead associated with creating and destroying threads, leading to better resource management.
- Scalability: The server can scale to handle a large number of concurrent connections without exceeding system limits or resource constraints.

4. Stress Test for Concurrency and Rate Limiting

Overview

A stress test was designed to evaluate the server's performance under high load and concurrent client requests. The test simulates multiple clients making requests to the proxy server to check concurrency control, rate limiting, and thread pooling behavior.

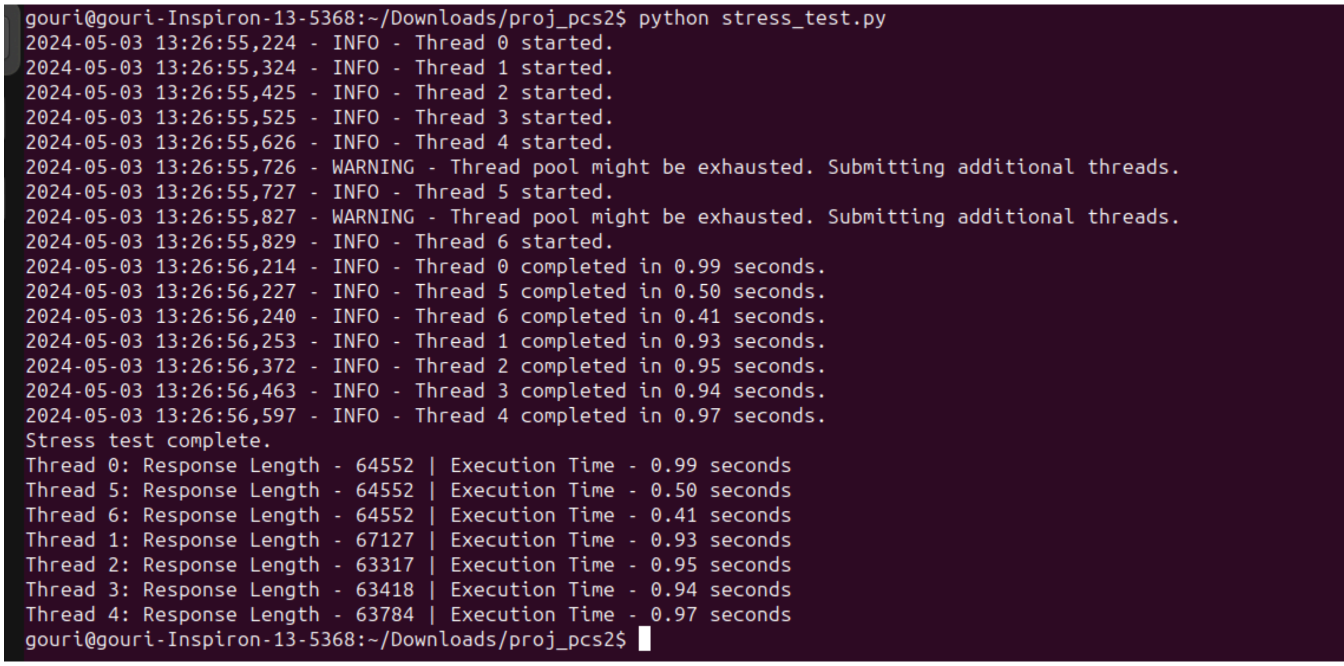
Implementation Details

- **Number of Threads**: The stress test creates 7 concurrent threads to simulate client requests to the server.
- **Concurrency Control**: The test checks if the thread pool manages concurrency efficiently and doesn't exceed its capacity.
- **Rate Limiting**: The test verifies if the server correctly enforces rate limiting, closing connections when a client exceeds the allowed number of requests.
- **Monitoring Thread Pool**: The test logs when the thread pool is at capacity and monitors the execution time for each thread.

Results

- **Concurrency Handling**: The server successfully handled multiple concurrent client requests, indicating that thread pooling is effective.

- **Rate Limiting Enforcement**: The server correctly enforced rate limiting, closing connections when the limit was exceeded.
- **Thread Pool Stability**: The thread pool maintained stability, with no resource exhaustion or unexpected shutdowns.



```
gouri@gouri-Inspiron-13-5368:~/Downloads/proj_pcs2$ python stress_test.py
2024-05-03 13:26:55,224 - INFO - Thread 0 started.
2024-05-03 13:26:55,324 - INFO - Thread 1 started.
2024-05-03 13:26:55,425 - INFO - Thread 2 started.
2024-05-03 13:26:55,525 - INFO - Thread 3 started.
2024-05-03 13:26:55,626 - INFO - Thread 4 started.
2024-05-03 13:26:55,726 - WARNING - Thread pool might be exhausted. Submitting additional threads.
2024-05-03 13:26:55,727 - INFO - Thread 5 started.
2024-05-03 13:26:55,827 - WARNING - Thread pool might be exhausted. Submitting additional threads.
2024-05-03 13:26:55,829 - INFO - Thread 6 started.
2024-05-03 13:26:56,214 - INFO - Thread 0 completed in 0.99 seconds.
2024-05-03 13:26:56,227 - INFO - Thread 5 completed in 0.50 seconds.
2024-05-03 13:26:56,240 - INFO - Thread 6 completed in 0.41 seconds.
2024-05-03 13:26:56,253 - INFO - Thread 1 completed in 0.93 seconds.
2024-05-03 13:26:56,372 - INFO - Thread 2 completed in 0.95 seconds.
2024-05-03 13:26:56,463 - INFO - Thread 3 completed in 0.94 seconds.
2024-05-03 13:26:56,597 - INFO - Thread 4 completed in 0.97 seconds.
Stress test complete.
Thread 0: Response Length - 64552 | Execution Time - 0.99 seconds
Thread 5: Response Length - 64552 | Execution Time - 0.50 seconds
Thread 6: Response Length - 64552 | Execution Time - 0.41 seconds
Thread 1: Response Length - 67127 | Execution Time - 0.93 seconds
Thread 2: Response Length - 63317 | Execution Time - 0.95 seconds
Thread 3: Response Length - 63418 | Execution Time - 0.94 seconds
Thread 4: Response Length - 63784 | Execution Time - 0.97 seconds
gouri@gouri-Inspiron-13-5368:~/Downloads/proj_pcs2$
```

5.Gzip Compression:

Overview:

- Before sending a response to the client, the server compresses the content using gzip compression.
- Gzip compression reduces the size of the response data, resulting in faster transmission over the network and reduced bandwidth usage.
- Compression is applied to both cached responses and fresh responses fetched from the web.
- Smaller file sizes also benefit clients by reducing storage requirements and improving loading times for web pages and other resources.

Implementation Details:

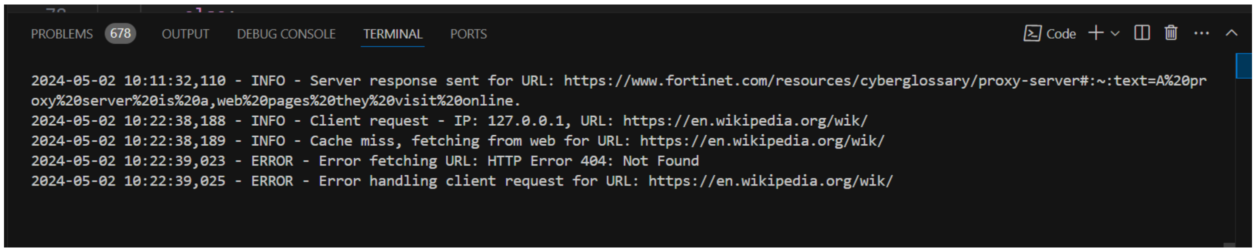
- The proxy server compresses the content of responses before sending them to clients using gzip compression. This is achieved through the compress_content() method.
- The gzip compression level is set to the highest value of 9 in our code. This indicates maximum compression, which may result in smaller compressed files.

- The compress_content() method includes error handling to catch any exceptions that may occur during the compression process. If an error occurs, the method logs an error message and returns None, indicating that compression failed.

6.Error Handling:

- Exceptions that occur during content compression, such as errors in creating the gzip compressor object or writing compressed data, are caught using a try-except block.
- If an exception occurs, an error message is logged, and None is returned to indicate failure in compressing the content.
- Exceptions that may arise during URL fetching, such as network errors or invalid URLs, are caught using a try-except block.
- If an exception occurs, an error message indicating the failure reason is sent back to the client socket, and None is returned to indicate failure in fetching the URL
- The logging module is used to log error messages when exceptions occur.

Given below is the message logs when wrong url entered:



```
2024-05-02 10:11:32,110 - INFO - Server response sent for URL: https://www.fortinet.com/resources/cyberglossary/proxy-server#:~:text=A%20pr
oxy%20server%20is%20a,web%20pages%20they%20visit%20online.
2024-05-02 10:22:38,188 - INFO - Client request - IP: 127.0.0.1, URL: https://en.wikipedia.org/wik/
2024-05-02 10:22:38,189 - INFO - Cache miss, fetching from web for URL: https://en.wikipedia.org/wik/
2024-05-02 10:22:39,023 - ERROR - Error fetching URL: HTTP Error 404: Not Found
2024-05-02 10:22:39,025 - ERROR - Error handling client request for URL: https://en.wikipedia.org/wik/
```

Client-Side Features of the Proxy Client

1. HTTP Request Formatting:

- The client constructs HTTP GET requests with the requested URL and sends them to the proxy server.
- Each request includes the necessary headers, such as the Host header indicating the origin server.

2. Response Handling:

- The client receives response data from the proxy server in chunks of 4096 bytes.
- It aggregates these chunks until the entire response is received, handling the data efficiently to avoid memory issues.

3. Saving Response to File:

- After receiving the response from the proxy server, the client saves the received response content to an HTML file.

- It checks if the response is gzip-compressed by inspecting the first few bytes of the response.
- If the response is gzip-compressed, it decompresses the content using the gzip module and writes the decompressed content to the specified file.
- If the response is not compressed, it writes the response directly to the file.

4. Open in Browser

- The open_in_browser method opens the saved HTML file in the default web browser.
- It utilizes the webbrowser module to open the file in a new tab or window, depending on the browser's configuration.

General Features:

1. Logging:

- Both the server and client incorporate logging functionality using Python's logging module.
- Logs provide detailed information about the server's behavior, including client requests, cache hits/misses, errors, and warnings.
- Logging helps in monitoring the server's performance, debugging issues, and auditing user activity.

2. Configuration Parameters:

- The server allows customization of various parameters, such as the server host, port, cache expiry time, maximum requests per IP, and request interval.
- These parameters are set via constructor arguments when initializing the ProxyServer instance.

Contribution of group members:

Shruti-Implemented caching mechanism,threading,rate limiting feature of proxy server,client side of proxy client,helped in making project report and presentation.
Gouri-Implemented thread pooling and concurrency control, gzip compression,stress test code for testing,error handling,helped in making the project report and presentation.