****

Department of Computer Science & Mathematics

CSC 430 Computer Networks  
Fall 2024-2025  
DESIGN PROJECT

Gaelle Tomko 202206978

Rafqa El Haddad 202206513

Mahmoud sidani 202205770

*This page was intentionally left blank*

# Table Of Contents

Contents

[Table Of Contents 3](#_Toc184606256)

[I. Overview: 4](#_Toc184606257)

II. Code and functionalities:……………………………………………………………….4

III. Admin interface:………………………………………………………………………15

IV. Testing:………………………………………………………………………………….17

V. Problems we faced:……………………………………………………………………18

# 

# Overview:

This project includes creating a **Proxy Server** that forwards client Requests to a server and relays the response back. It uses multithreading and socket programming to handle many http requests and cache responses. The server includes logging functionalities, blacklist/whitelist managements and an admin interface for monitoring and managing logs , cache ,and much more we will delve deeper into the server below.

1. Code and functionalities:  
     
   1. Imports:  
    A screen shot of a computer

   Description automatically generated

A computer code with black text

Description automatically generated2. Flask and MySql:

Above is the setup for the setup for the flask app for the admin interface and the my SQL config(proxy\_server)

3.Variables :  
 A screenshot of a computer code

Description automatically generated

Here we created the Global variables necessary for our code to function.

* The response\_cache is used to store cached.
* Blacklist and whitelist used to allow and block domains.
* The log file hold the request and responses.

4.Log function :  
 A screenshot of a computer program

Description automatically generated

This function logs a message with a timestamp. It first gives the time now and then writes the logs into app.log and it is then saved into the database. If any error occurs it is caught and printed.

5.Cache Retrieval: A computer screen shot of a program

Description automatically generated

* This function checks if the request is found in the cache then checks if the req is still valid using the expires\_at that compares the timestamp with our current time
* If the cache response is valid it returns the cached response(HIT)
* If not valid then that means it is expired hence it is deleted
* If the request was not found ( Cache miss)

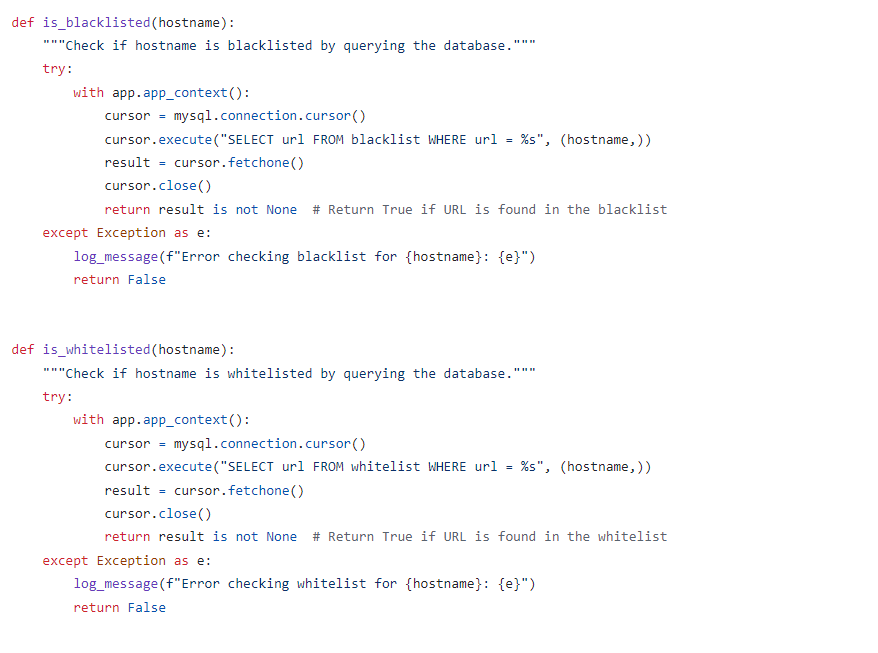
6.Cache Insert:

A screenshot of a computer program

Description automatically generated

* Add to cache : it stores responses along with their exp time and saves the req and size and expiration into the cache in the SQL table

7.Blacklist/Whitelist:



* When a client sends a request the handle client checks the target hostname if it is in the blacklist it calls Is\_blacklisted().
* If it was in the whitelist it calls is\_whitlisted().
* Both functions query the database to check the hostname (url) if no result was found then it is whitelisted.

8.Proxy Server:

A computer screen shot of a computer code

Description automatically generated

* This function initializes a TCP Socket to listen for client connections it is created using the AF\_INET and SOCK\_STREAM
* It uses multithreading: For each connection the server creates a new thread to handle the client request.

A screenshot of a computer program

Description automatically generated9.Client Request Handle:

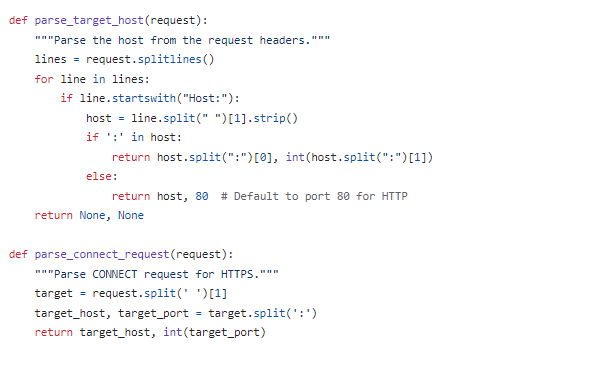
A screenshot of a computer program

Description automatically generated

This function handles communication between the server and the client :

* It receives the clients request processes it and forwards it to the server if the target is HTTPS it uses the https\_tunnel to secure a connection to the server .
* It checks if the response is cached it send it back to the client and if it was not cached it forwards it to the HTTP request to the server.
* It reuses the blacklist to check if it if the host is blacklisted or not.
* It logs all actions.

10.Parsing:



1. Parse\_target\_host:

* It extracts the host and port from the host header in the http request,if the port is found it splits the host and port.
* If no port was found it returns 80 as a standard HTTP port

1. Parse\_connc\_request:

* It extracts the target host and port and splits connect request into target server and port.

11. HTTPS Tunneling:A screenshot of a computer program

Description automatically generatedA screenshot of a computer code

Description automatically generated

This function sets an HTTPS tunnel to the target server for encrypted communication

* It sends a Http response to the clint saying that a connection has been secured.
* Data is then forwarded without inspecting content.
* Select.select() is then used to wait for data from either the client or the server then forwards the data accordingly.

A screenshot of a computer program

Description automatically generated

* This function is responsible for forwarding HTTP requests from a client to the server, it receives the response and forwards it back to the client.
* Before sending the request to the target server, the request is logged, including the target server and port.
* A new socket (proxy\_socket) is created to connect to the target server and send the client's request.
* Data is received in chunks from the target server and forwarded back to the client.
* The response is split into headers and body.
* After the full response is received, the response gets cached.

2. parse\_request(request)

A computer screen shot of a code

Description automatically generated

This function parses an HTTP request to extract the HTTP method, path, version, and headers from the request.

* The request is split into individual lines by \r\n.
* It extracts the requested path, and the HTTP version and returns them.
* All lines are parsed into key and value pairs stored in a headers (dictionary).

3. parse\_host\_header(headers)

A screenshot of a computer code

Description automatically generated

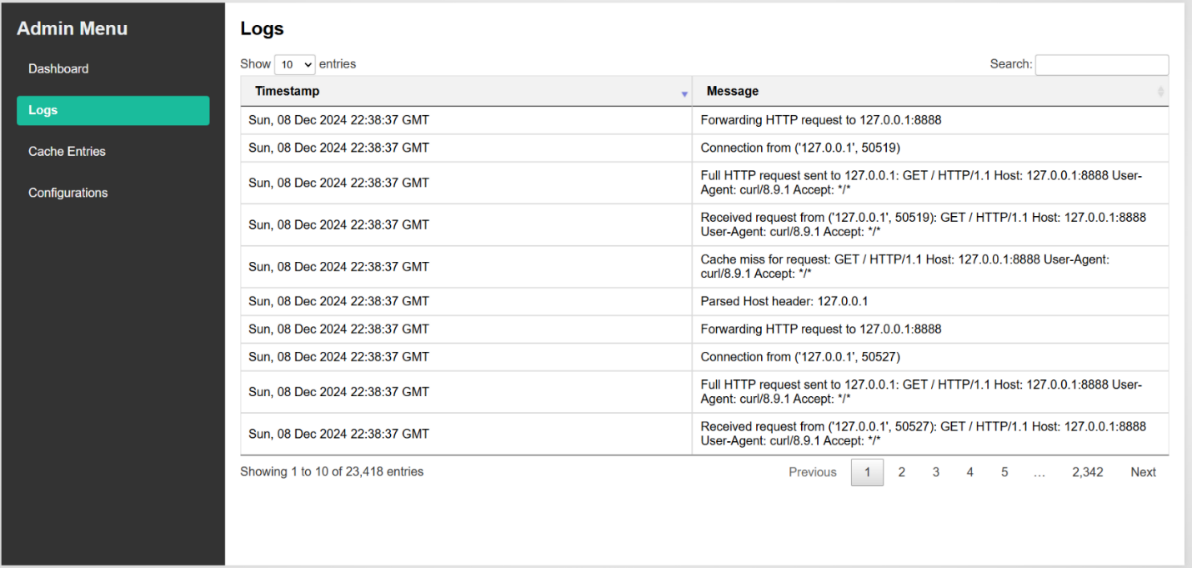
This function extracts the target host and port information from the Host header inside the http request.

* The function looks for the Host header in the request's headers dictionary.
* If a port is found in the Host header (example.com:8080) it splits the header into the server and port.
* If no port is specified, it returns to port 80.

1. Admin interface:

A screenshot of a computer

Description automatically generated

* Those pages were created using HTML CSS and JavaScript for the view, MySQL as Database and flask to create APIs for better communication between components (all APIs can be found in mysqlconnection.py)
* In the admin interface, the Dashboard page displays an input field where you can enter the URL from which you want to fetch a response using the proxy. Once the URL is provided, it will display the HTML page source code.
* The Logs page shows all logs added to the app.log file. These logs are presented in a DataTable with features such as pagination, sorting, and search functionality.

A screenshot of a computer

Description automatically generated

* The Cache Entries page displays all cached data along with their expiration times, also presented in a DataTable.

A screenshot of a computer

Description automatically generated

* The Configurations page allows you to view the blacklist and whitelist. You can add or remove items from these lists, and the proxy will return responses based on these configurations. If a URL is in the blacklist, the proxy will not return any response for it.

1. Testing:

* We used curl to test the functionality of our proxy server a command line interpreter tool for making http Requests. We used it to simulate client requests to the server to verify that it forwarded the requests to the server and returned the responses.
* We used this command: curl -x http://127.0.0.1:8888 <http://example.com>

**What does this command do?**

* The -x specifies the proxy server<http://127.0.0.1:8888> and sends a get request to <http://example.com> through our proxy server.
* It checks that the proxy server correctly logs the request and forwards it to the target
* It ensures that the response from example.com is received by the proxy server and back to curl

**A screenshot of a computer program

Description automatically generated**

1. Problems we faced:
   * 1. We faced a problem with the cache it was taking a bit too much time to return and we couldn’t quite solve the problem.