نام و نام خانوادگی: محمد کربلایی شعبانی
شماره دانشجویی: ۹۹۲۲۲۰۸۵

شماره تمرین:پایانی
تاریخ تحویل: ۱۴ تیر ۱۴۰۳

عنوان تمرین: Proxy Server

چکیده

در این پروژه یک proxy server پیاده سازی شده است. Proxy server ها با عمل کردن به عنوان واسطی بین کلاینت و سرور مزایایی از جمله امنیت و سرعت بالاتر در پاسخدهی فراهم میکنند. امنیت و حریم خصوصی از طریق پنهان کردن هویت کلاینت ها انجام میشود چرا که درخواست ها به طور مستقیم به سرور ارسال نمیشوند بلکه از طریق proxy server ارسال و پاسخ دریافت شده از سرور به طور غیر مستقیم به دست کلاینت میرسد.

سرعت پاسخدهی ازین جهت در استفاده از proxy server ها بالاتر است که با cache کردن داده از دریافت دوباره اطلاعات از سرور جلوگیری میکند و چون معمولا proxy server ها فاصله کمتری تا کلاینت ها دارند با تاخیر کمتری و بدون محاسبات چند باره پاسخوگی کلاینت ها هستند.

راه اندازی سرور

به طور کلی اعمال عادی مربوط به راه اندازی سرور در تابع start_server انجام میشوند. به این صورت که سوکت سرور ساخته شده و با localhost و پورت ۸۸۸۸ تنظیم میشود و منتظر اتصال سوکت کلاینت ها میماند. به محض اتصال یک کلاینت یک ریسمان ساخته شده و تابع handle_client در آن thread اجرا میشود.

```
def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('', PORT))
    server_socket.listen(5)
    print(f"Proxy server is running on port {PORT}")

    while True:
        client_socket, addr = server_socket.accept()
        client_handler = threading.Thread(target=handle_client, args=(client_socket,))
        client_handler.start()
```

تصویر ۱ – کد مربوط به راه اندازی اولیه سرور

درخواست GET

در تابع handle_client ابتدا اطلاعات درخواست کلاینت استخراج میشود و به تابع fetch_from_server پاس داده میشود تا درخواست به سرور ارسال شود. مکانیزم های متفاوت دیگری هم در این تابع پیاده سازی شده است که در بخش های بعدی مورد برسی قرار میگرند.

نام و نام خانوادگی: محمد کربلایی شعبانی

شماره دانشجویی: ۹۹۲۲۲۰۸۵

شماره تمرین:پایانی

تاریخ تحویل: ۱۴ تیر ۱۴۰۳

عنوان تمرین: Proxy Server

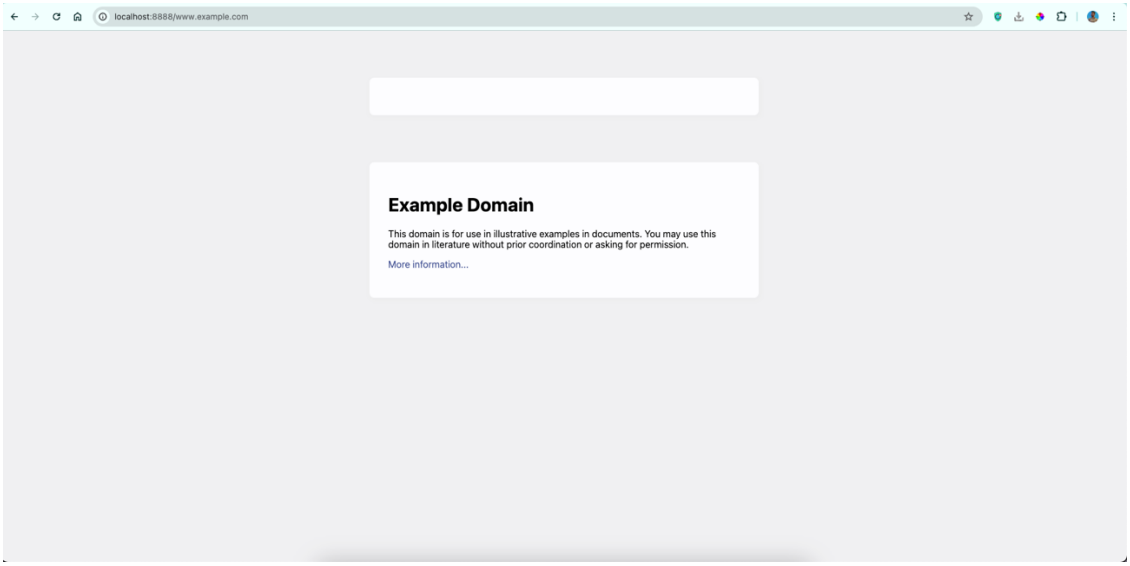
مکانیزم لاگ برای proxy server پیاده سازی شده است که اطلاعات هر درخواست از جمله زمان شروع و پایان و اختلاف این دو زمان از جمله url درخواست شده در یک فایل csv به اسم request_logs ذخیره میشود. وظیفه لاگ کردن این اطلاعات بر عهده تابع log_data هست

در تصویر شماره دو یک نمونه از محتویات این فایل را مشاهده میکنید.

	Request ID	URL	Start Time	End Time	Elapsed Time
	0	www.example.	1720107056.3	1720107058.7	2.48
	2	www.example.	1720107060.8	1720107060.8	0
	4	www.google.c	1720107064.4	1720107066.8	2.33
	14	www.example.	1720107074.1	1720107074.1	0

تصویر ۲ – نمونه ای از لاگ سرور

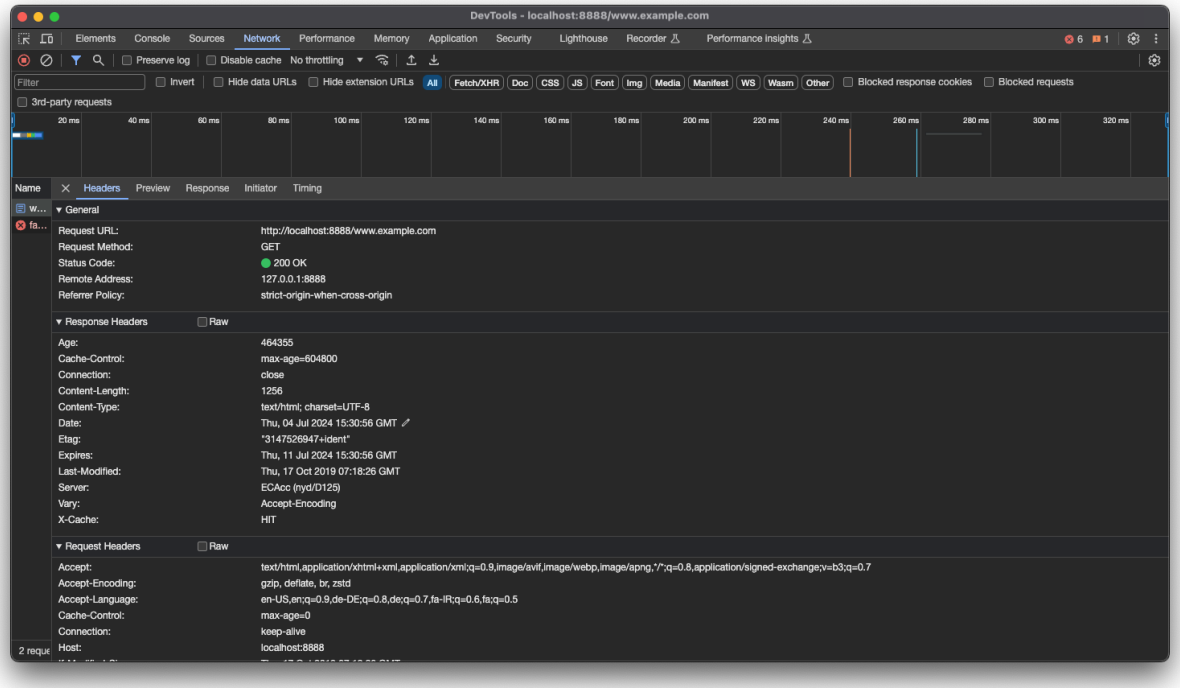
در تصویر شماره دو مشاهده میکنید که ۴ درخواست به سرور ارسال شده است که درخواست اول برای دریافت www.example.com بوده که ۲.۴۸ ثانیه به طول انجامیده است. در تصویر شماره سه تصویر محتویات وبسایت که توسط proxy server به دست کلاینت (در اینجا مرورگر) رسیده است را مشاهده میکنید.



تصویر ۴ – فراخوانی وبسایت www.example.com در مرورگر

نام و نام خانوادگی: محمد کربلایی شعبانی
شماره دانشجویی: ۹۹۲۲۲۰۸۵
شماره تمرین:پایانی
تاریخ تحویل: ۱۴ تیر ۱۴۰۳
عنوان تمرین: Proxy Server

لازم به ذکر است که درخواستی که برای دریافت این وبسایت به proxy server ارسال شده از نوع GET است که در تصویر شماره ۴ مشاهده میکنید. یک HTTP request توسط مرورگر فرستاده شده و پاسخ ۲۰۰ OK دریافت شده است.



تصویر ۴ – جزییات درخواست GET ارسال شده توسط مرورگر

```python
def handle_client(client_socket):
    global request_counter

    with counter_lock:
        request_id = request_counter
        request_counter += 1

    start_time = time.time()

    try:
        request = client_socket.recv(BUFFER_SIZE).decode()
        headers = request.split('\r\n')
        if len(headers) < 1:
            client_socket.close()
            return

        first_line = headers[0].split()
        if len(first_line) < 2:
            client_socket.close()
            return

        method = first_line[0]
        url = first_line[1][1:]

        if not url.startswith("www."):
            client_socket.close()
            return

        cache_key = get_cache_key(url)
        cache_file_path = os.path.join(CACHE_DIR, cache_key)

        if os.path.exists(cache_file_path):
            with open(cache_file_path, 'rb') as cache_file:
                response = cache_file.read()
                client_socket.sendall(response)
            log_data(request_id, url, start_time, time.time())
            client_socket.close()
            return

        fetch_from_server(client_socket, method, url, headers, cache_file_path, request_id, start_time)

    except Exception as e:
        handle_error(client_socket, str(e))

    client_socket.close()
    log_data(request_id, url, start_time, time.time())
```

تصویر ۵ – کد تابع **handle_client**

نام و نام خانوادگی: محمد کربلایی شعبانی
شماره دانشجویی: ۹۹۲۲۲۰۸۵
شماره تمرین:پایانی
تاریخ تحویل: ۱۴ تیر ۱۴۰۳
عنوان تمرین: Proxy Server

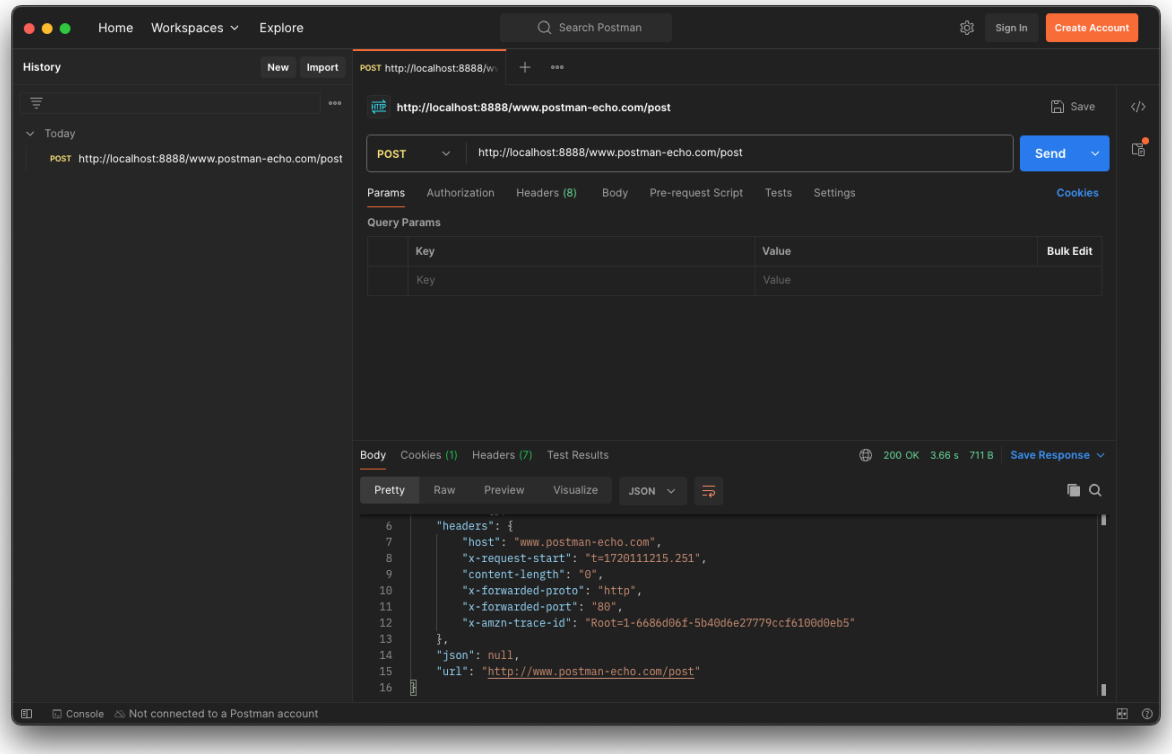
درخواست POST

همانطور که قبلا ذکر شد این پیاده سازی قابلیت مدیریت کردن HTTP POST را هم دارد به این صورت که در تابع fetch_from_server در تصویر شماره ۶ میبینید.

```
def fetch_from_server(client_socket, method, url, headers, cache_file_path, request_id, start_time):
    try:
        web_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        web_server_socket.connect((url.split('/')[0], 80))

        if method == 'GET':
            web_server_socket.sendall(f"GET /{'/'.join(url.split('/')[1:])} HTTP/1.0\r\nHost: {url.sp
        elif method == 'POST':
            content_length = 0
            for header in headers:
                if header.lower().startswith('content-length:'):
                    content_length = int(header.split()[1])
                    break
            post_data = client_socket.recv(content_length).decode()
            web_server_socket.sendall(f"POST /{'/'.join(url.split('/')[1:])} HTTP/1.0\r\nHost: {url.s

        response = b""
        while True:
            data = web_server_socket.recv(BUFFER_SIZE)
            if not data:
                break
            response += data

        web_server_socket.close()
```

تصویر ۶ – ارسال درخواست های GET و POST

با توجه به اینکه ارسال درخواست post با استفاده از مرورگر به راحتی امکان پذیر نیست از postman برای ارسال یک درخواست post استفاده کردیم که در تصویر شماره هفت مشاهده میکنید که به درستی ارسال شده و پاسخ مناسب و درست هم دریافت شده است.



تصویر ۷ – نمونه POST request در نرم افزار postman

Caching

ویژگی caching از ویژگی های اساسی proxy server هاست که در این پیاده سازی هم لحاظ شده است به این صورت که هر درخواستی به سرور ارسال میشود ابتدا چک میشود که آیا فایلی با اسم مشابه در پوشه cache ذخیره سازی شده است یا خیر. در صورتی که فایل از قبل cache شده باشد هیچ درخواستی به سرور ارسال نمیشود و مستقیما دیتای cache شده به کلاینت ارسال میشود که به این شکل سرعت پاسخدهی به کلاینت به شدت بالا میرود.

برای مثال در مورد تاثیر مکانیزم cache پیاده سازی شده به تصویر ۲ دقت کنید. در این تصویر مشهود است که در درخواست اول به سرور www.example.com حدودا دوثانیه وقت صرف شده است در صورتی که در درخواست دوباره این وبسایت زمانی کمتر از یک ثانیه صرف شده است که تقریبا قابل چشم پوشی است. این نشان میدهد که caching کاملا موثر بوده و زمان پاسخدهی برای داده تکراری را به شدت کاهش داده است.

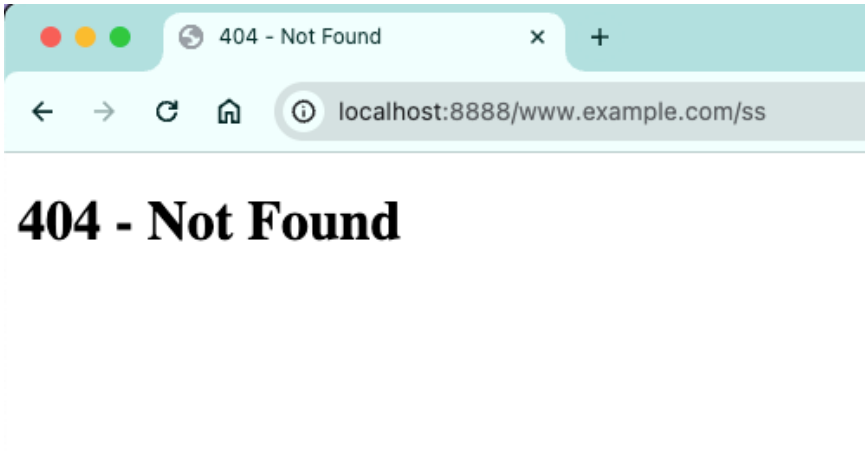
در مورد نام گذاری فایل های cache شده هم از کتابخانه hashlib استفاده میکنیم و url درخواست شده را هش میکنیم و به عنوان اسم فایل استفاده میکنیم.

در صورتی که فایل در سرور ذخیره نشده باشد برای دریافت آن به سرور درخواست داده میشود و پاسخ دریافت شده به کلاینت ارسال میشود همچنین یک نمونه از آن هم در سرور cache میشود تا در درخواست های بعدی مورد استفاده قرار گیرد.

Error Handling

اما ممکن است که وبسایتی درخواستی وجود نداشته باشد یا سرور مقصد چنین مسیری نداشته باشد. در این صورت هم برنامه proxy server ی که پیاده سازه شده توانایی مدیریت این مشکلات را دارد و پاسخ مناسب به کلاینت ارسال میکند.

در این صورت proxy server توانایی مدیریت خطا را دارد و پاسخ دریافتی از سرور را با کد وضعیت مربوطه به کلاینت میفرستد. تصویر ۸ یک نمونه از این وضعیت هست.



تصویر ۸ – نمونه مدیریت error

در تصویر ۹ تابعی که این وظیفه را به عهده دارد مشاهده میکنید.

```
def handle_error(client_socket, error_message, status_code=500):
    error_response = f"HTTP/1.1 {status_code} {error_message}\r\nContent-Length: {len(error_message)}\r\n\r\n{error_messag
    client_socket.sendall(error_response.encode())
```

تصویر ۹ – کد تابع handle_error

نام و نام خانوادگی: محمد کربلایی شعبانی
شماره دانشجویی: ۹۹۲۲۲۰۸۵
شماره تمرین:پایانی **تاریخ تحویل: ۱۴ تیر ۱۴۰۳**
عنوان تمرین: Proxy Server

نتیجه گیری

با پیاده سازی یک proxy server متوجه شدیم که چه مزایایی در استفاده از این موجودیت در شبکه وجود دارد و چرا در معماری شبکه اینترنت امروزی به طور گسترده مورد استفاده قرار میگیرد. Caching به عنوان یکی از مزیت های اصلی proxy server ها هم از هزینه های محاسباتی در سرور میکاهد هم از هزینه هایی زمانی که کلاینت باید متحمل شود. همچنین به طور کلی برای کاهش ترافیک در شبکه بسیار موثر است چرا که با مدیریت و پاسخ دهی به درخواست کلاینت ها در proxy server ها که به نسبت محلی تر هستند نسبت به کلاینت ها ازدحام در شبکه ی اینترنت کاهش می یابد و به طور غیر مستقیم همه کاربران اینترنت را متنعم میسازد.