Python Proxy Server

Session: 2022-2026

Submitted by:

Ifra Fazal 2022-CS-194
Burhan Ahmad 2022-CS-191

Submitted to:

Sir Tehseen Ul Hassan Shah

Department of Computer Science

University Of Engineering And Technology,

Lahore, Pakistan

Contents

Introduction

This project is a Python-based proxy server with a graphical user interface (GUI) built using PyQt5. The proxy server is designed to manage web requests, providing features such as caching and URL blocking. It aims to enhance security, improve browsing performance, and offer an intuitive interface for managing proxy settings.

Project Description

The Python Proxy VPN project is a robust and user-friendly proxy server equipped with a graphical user interface (GUI) developed using PyQt5. This proxy server is designed to control and enhance web traffic management with features including caching, URL blocking, rate limiting. The project aims to provide a secure, efficient, and accessible solution for individuals or organizations looking to manage their internet usage more effectively.

Methodology Used

1. Requirements Analysis

The first step involves gathering and analyzing the requirements for the proxy server. This includes understanding the core functionalities needed, such as caching, URL blocking, rate limiting. The requirements are then documented and prioritized to guide the development process.

2. System Design

2.1 Architecture Design

The system architecture is designed to ensure modularity, scalability, and maintainability. The main components include:

- **Proxy Server:** Handles incoming requests and responses.
- **Caching Module:** Manages cached responses to improve performance.
- **Blocking Module:** Manages the list of blocked URLs/IPs.
- **Rate Limiting Module:** Controls the frequency of requests from clients.
- **Graphical User Interface (GUI):** Provides a user-friendly interface for managing the proxy server.

2.2 Data Flow Design

The data flow between components is designed to ensure efficient processing of requests and responses. This includes defining how data is passed between the proxy server, caching module, blocking module, rate limiting module, and GUI.

3. Implementation

3.1 Proxy Server

Implemented using the http.server and socketserver modules. The server handles HTTP requests, forwards them to the target URLs, and returns the responses to the clients.

3.2 Caching Module

Uses the hashlib module to create unique cache keys based on URLs. Responses are stored in a cache directory and retrieved for repeated requests to improve performance.

3.3 Blocking Module

Maintains a list of blocked URLs/IPs. Incoming requests are checked against this list, and access is denied for any matches.

3.4 Rate Limiting Module
Monitors the frequency of requests from each client IP and enforces rate limits to prevent abuse.

3.5 Graphical User Interface (GUI)
Developed using PyQt5 to provide an intuitive interface for managing the proxy server. The GUI includes components for starting/stopping the server, viewing logs, and managing blocked URLs

Key Features and Functionality

Key Features

1. **Caching:**
   - **Functionality:** Stores responses of frequently requested URLs in a cache directory.
   - **Benefit:** Reduces bandwidth usage and improves response times for subsequent requests to the same URLs.
2. **URL Blocking:**
   - **Functionality:** Maintains a list of blocked URLs and IP addresses.
   - **Benefit:** Enhances security by preventing access to malicious or unwanted websites.
3. **Rate Limiting:**
   - **Functionality:** Controls the number of requests a client can make within a specified time period.
   - **Benefit:** Prevents abuse and ensures fair usage among clients.
4. **Graphical User Interface (GUI):**
   - **Functionality:** Provides an intuitive and user-friendly interface for managing the proxy server.
   - **Benefit:** Allows users to easily start/stop the server, view logs, and manage blocked URLs.

Detailed Functionality

1. **Proxy Server:**
   - Receives client requests and forwards them to the target URLs.
   - Returns responses to the clients after processing.
2. **Cache Management:**
   - Generates unique cache keys based on URLs using hashlib.
   - Checks if a response is cached and serves it if available.
   - Fetches responses from the source if not cached and stores them in the cache.
3. **URL Blocking:**
   - Checks incoming requests against the list of blocked URLs/IPs.
   - Denies access to any blocked URLs and sends a "403 Forbidden" response.
4. **Rate Limiting:**

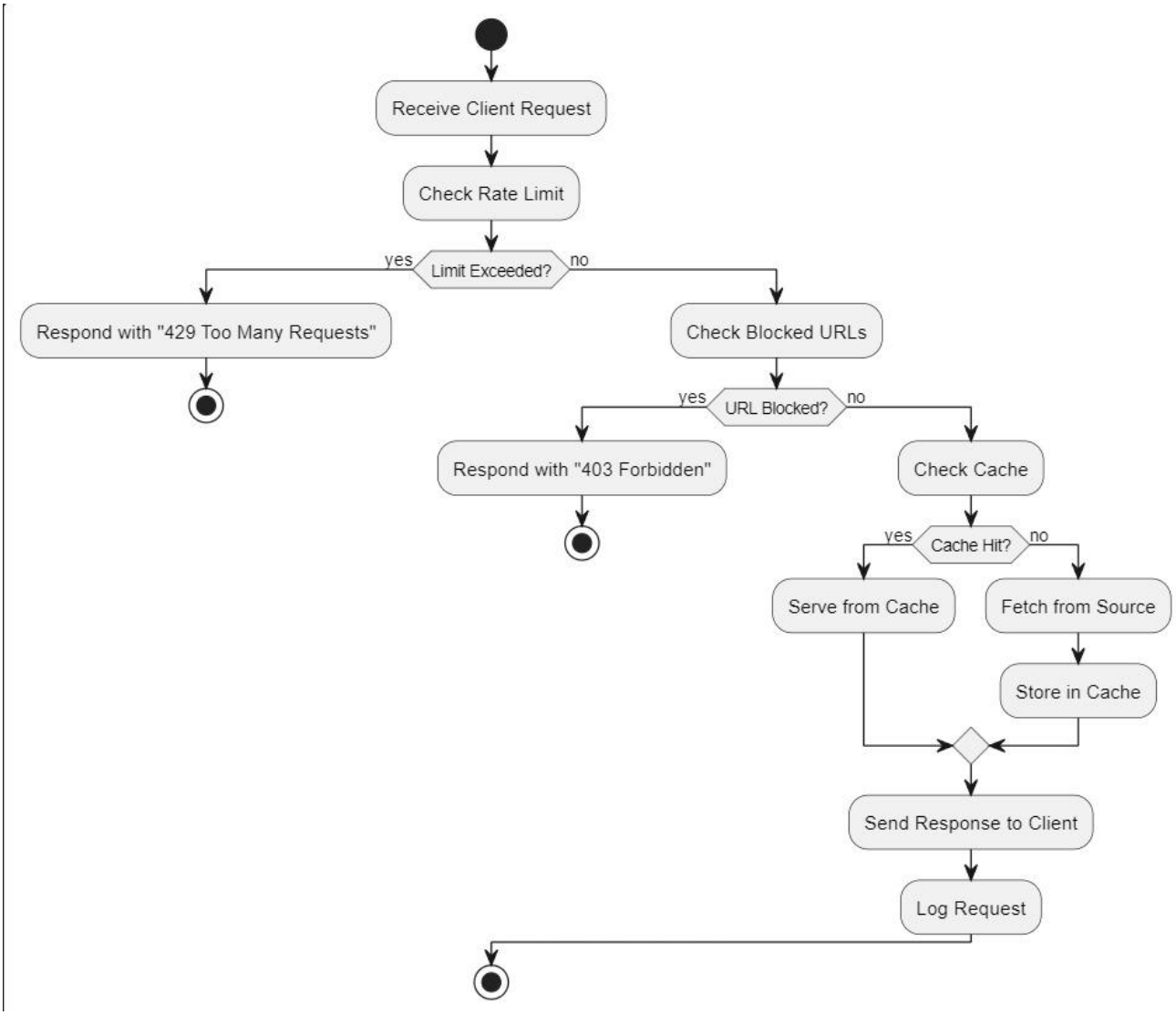
- Monitors the frequency of requests from each client IP.
- Enforces rate limits and sends a "429 Too Many Requests" response if limits are exceeded.

5. **GUI Components:**
   - Status Indicator: Displays the current status of the proxy server (connected or disconnected).
   - Control Buttons: Circular buttons for starting and stopping the proxy server.
   - Logs Section: Displays real-time logs of requests, responses, and events.
   - Blocked URLs Management: Allows users to add or remove blocked URLs.

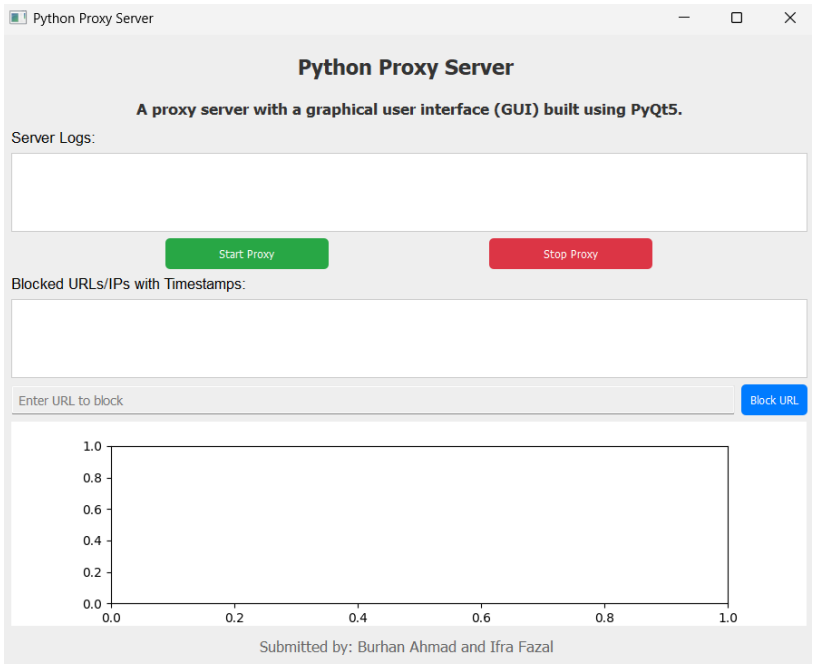
Technologies and Tools Used

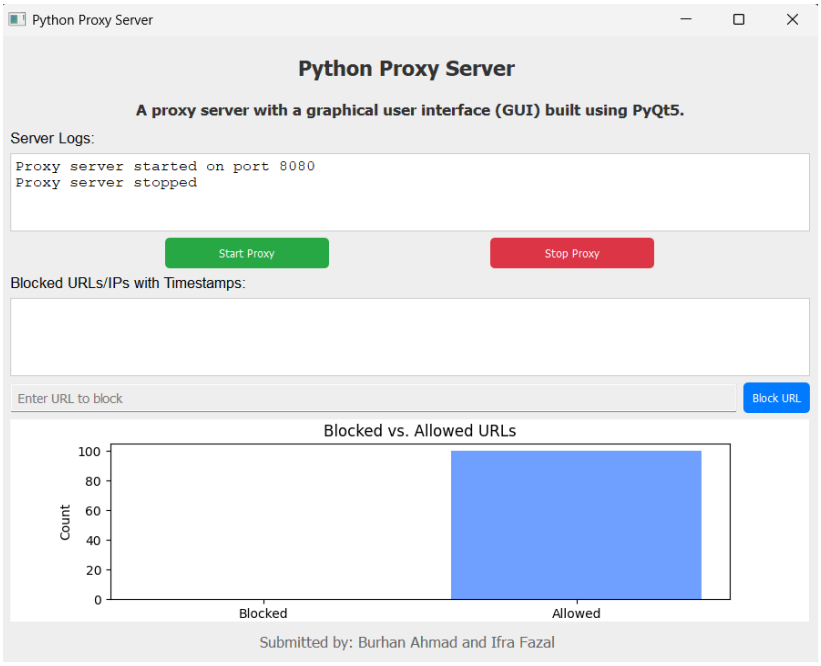
- Python
- http.server
- socketserver
- urllib.request
- hashlib
- threading
- datetime
- PyQt5
- Integrated Development Environment (IDE)
- Version Control System (Git)
- Rate Limiting
- Matplotlib

Flow Chart Diagram



Tool Screenshots





Results and Analysis

Results

- **Caching:** Successfully implemented caching mechanism, reducing load times and bandwidth usage for frequently requested URLs.
- **URL Blocking:** Effectively blocked specified URLs and IP addresses, enhancing security by preventing access to unwanted sites.
- **Rate Limiting:** Rate limiting functionality successfully restricted the number of requests a client can make within a specified period, ensuring fair usage.
- **Graphical User Interface (GUI):** The GUI was developed using PyQt5, offering an intuitive interface for managing the proxy server, including starting/stopping the server, viewing logs, and managing blocked URLs.

Analysis

- **Performance Improvement:** The caching mechanism significantly reduced load times for frequently accessed URLs, leading to a better user experience and more efficient bandwidth usage.
- **Enhanced Security:** The URL blocking feature effectively prevented access to malicious or unwanted websites, contributing to a safer browsing environment.
- **Compatibility and Tracking:** The addition of custom headers improved the compatibility of HTTP requests with different web servers and allowed for better user tracking.
- **Comprehensive Monitoring:** The logging functionality provided detailed insights into the proxy server's operations, facilitating easier monitoring, debugging, and analysis of web traffic patterns.
- **User-Friendly Management:** The PyQt5-based GUI provided an accessible and intuitive interface for managing the proxy server, making it easy for users to control the server, view logs, and add or remove blocked URLs.

Conclusion

The Python Proxy VPN project successfully combines essential web traffic management features, including caching, URL blocking, rate limiting, custom headers, and logging, into a single cohesive solution. By providing an intuitive and user-friendly PyQt5-based GUI, the project ensures easy administration and monitoring of the proxy server. This results in a robust, efficient, and secure web browsing environment that meets the needs of both individual users and organizations. The project's success lies in its effective implementation of these features, making it a valuable tool for enhancing web security, optimizing performance, and ensuring controlled access.

Github Link

https://github.com/ifra71/proxy-server.git

References

- Python Documentation: http.server
- Python Documentation: socketserver
- PyQt5 Documentation: PyQt5
- Python Documentation: hashlib
- Python Package Index (PyPI): PyQt5
- Stack Overflow: Various discussions on implementing caching, URL blocking, rate limiting, and authentication in Python.