HTTP Server

****

**Session: 2022-2026**

**Submitted by:**

Muhammad Taha Saleem 2022-CS-139 Mohammad Abdullah 2022-CS-155

**Submitted to:**

Sir Tehseen Ul Hassan Shah

Department of Computer Science

**University Of Engineering And Technology, Lahore, Pakistan**

Contents

[**Introduction** 3](#_Toc186051513)

[**Project Description** 3](#_Toc186051514)

[**Methodology Used** 3](#_Toc186051515)

[**Key Features and Functionality** 4](#_Toc186051516)

[**Technologies and Tools Used** 5](#_Toc186051517)

[**Flow Chart Diagram** 6](#_Toc186051518)

[**Tool Screenshots** 7](#_Toc186051519)

[**Results and Analysis** 8](#_Toc186051520)

[**Conclusion** 9](#_Toc186051521)

[**Github Link** 9](#_Toc186051522)

[**References** 9](#_Toc186051523)

# **Introduction**

The rapid growth of internet and web-based applications have emphasized the importance of Hyper Text Transfer Protocol (HTTP) Server. It is the backbone of online communication. HTTP is a protocol for the application layer in the Open System Interconnection (OSI) Model [1]. HTTP servers play an important role for data communication to the client. Client requests to the server and server responds to the client appropriately [2].

The primary objective of the project is to develop a Hyper Text Transfer Protocol (HTTP) Server with a usable graphical user interface (GUI) for making its operations easier for users and strengthen understanding of the HTTP and its relevance to the computer networking. The project is going to address the server management issues like handling client request, response handling and monitoring server requests while making it easier for user to understand the HTTP server.

# **Project Description**

The primary goal of the project is to create a modern HTTP server with am graphical user interface, making it easier to start, stop and monitor the working of the server. It provides a functional platform for understanding how HTTP server’s intended functionalities and how they work. It is designed to be user-friendly for beginners to learn about HTTP protocol.

The project is built with key features to enhance its usability and functionality. The server is managed through a visually appealing GUI. It allows user to input the configurations for the server. It also handles the key server operations like starting the server, stopping the server, monitoring important information about the server like clients connected, handling client requests and information of headers at each layer for every request received from the client and every response sent to the client. It supports essential HTTP methods like GET, POST, PUT and DELETE [3]. The server is able to capture and analyze the packets related to the server port in real-time. The server is also featured with file handling to ensure that the clients can upload or download their files through the server. Multi-threading has been incorporated into the HTTP server to enable multiple clients at a time. However, the project has some exclusions of advanced features like security features of encryption and decryption of data. The project was developed to strengthen our understanding of the HTTP protocol.

# **Methodology Used**

A structured process was followed for the development of this project combining both the theoretical understanding of the HTTP servers and the OSI Model of computer networking with the practical implementation using Python.

We firstly understood the requirements and core functionalities of the HTTP server. Essential functionalities were outlined such as handling client requests, handling HTTP methods like GET, POST, PUT, DELETE, generating responses to the client and providing a user-friendly GUI for simplified interaction.

Then, a GUI was designed using the Tkinter library in Python to enhance the usability of the HTTP server. The interface was designed to simplify server operations. The backend of the project is designed to incorporate multiple clients simultaneously by multi-threading of client requests.

After the requirement gathering and design phase, implementation was done on the previous phases. The GUI was developed to enable user interaction with the server. The server was coded to support essential methods and the intended functional requirements decided in the first phase.

The testing and debugging were performed to validate that the server performs its intended functionalities under different circumstances. Common issues like incorrect request handling and other bugs were debugged and resolved. A visual representation of the used methodology is depicted in the Figure 1.

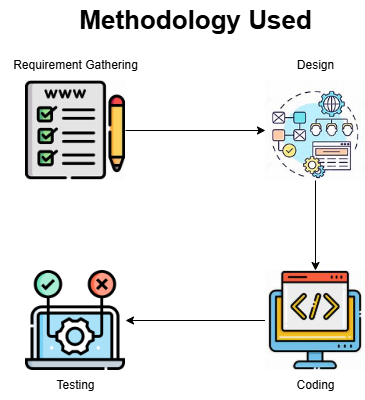


Figure . Methodology Used

# **Key Features and Functionality**

The HTTP server incorporates a variety of features to enhance usability, functionality, and user experience. Its key features are described below.

* **User-Friendly Graphical User Interface (GUI):** The server is built with an intuitive GUI to make it accessible for beginners. Users can input configurations for the server. They can start / stop the server and two panels are built to show output and backend logs in real-time to server handler.
* **Support for Essential HTTP Methods:** The server supports essential HTTP methods like GET, POST, PUT, DELETE. Users can request for files stored on the server using GET method. POST method can be used to upload files to the server. Files uploaded on the server can be updated or deleted through PUT and DELETE methods.
* **Dynamic File Handling:** The server has been incorporated with dynamic file handling to serve static files on the server to client. For example, user can request for any file in any directory on the server. If the file exists, it will be served to the client. Otherwise, it will respond with a 404 error and Not Found html.
* **Real-Time Packet Sniffing:** The server has been integrated with real-time packet sniffing of the server’s requests and responses. These packets are logged in the server panel on the GUI. Information of the packet headers are logged in the panel, helping users understand network communication at each level. This feature make project valuable for learning networking.
* **Multi-Threaded Request Handling:** The server is employed with multi-threading for handling client requests simultaneously. A thread is created for every client request and its response is generated in this thread. This multi-threaded architecture ensures smooth operations for the server under high load [4] [5].
* **Real-Time Logging:** The server logs are provided in two panels. The server output panel display high-level operations for the general public while packet analysis is shown in the other panel. The user can toggle the output of both these panels.
* **Authentication and Authorization:** The server has been incorporated with basic authentication and authorization mechanisms for security. It has been implemented for POST, PUT and DELETE methods.

This project is incorporated with a GUI, unlike traditional HTTP servers which are based on the command line interface. Features such as detailed packet analysis of the server make the project a valuable tool for learning computer network protocols and HTTP operations.

# **Technologies and Tools Used**

The following technologies and tools were used in the development of the HTTP server.

1. **Python**

Python is a versatile and beginner friendly language with a rich ecosystem of libraries making it ideal for network programming.

1. **Tkinter**

Tkinter is a library for development of graphical user interface (GUI) in Python. It was selected for its simplicity and ability to create a responsive interface. It allowed us to develop a user-friendly GUI for the server.

1. **Scapy**

Scapy is a powerful python libray for network packet manipulation and analysis. It was used because of its ability to sniff and analyze network packets in real-time. The project sniffes packets of the server requests and responses. This helped in creating an educational value for the project in learning networking concepts.

1. **Socket Programming**

The socket module in Python provided a straightforward way to implement client-server communication. It was essential for handling client connections, processing requests, and sending responses as per the HTTP protocol [5].

1. **Threading Module**

The threading module of python enabled server to perform simultaneous processing of multiple client request. This was crucial for ensuring smooth server operations under load. [4]

1. **MIME Types Module**

The mimetypes module of python was used to determine the content type of files being served. It helped the server handle different file formats appropriately, ensuring accurate responses to client requests.

1. **OS Module**

The os module helped in managing file handling operations, such as if a file exists, accessing file and paths. It was essential for implementing file upload or download functionalities and serving static files to the client.

1. **Urllib**

This module was used to parse URLs and query parameters, making it easier to process client requests and serve dynamic content.

# 

# **Flow Chart Diagram**

The server is initialized by the user and it will start waiting for the requests from the clients. Simultaneously, it will start packet sniffing. When a request is received, it will process request and respond accordingly. If invalid method is used, it will send an error response.

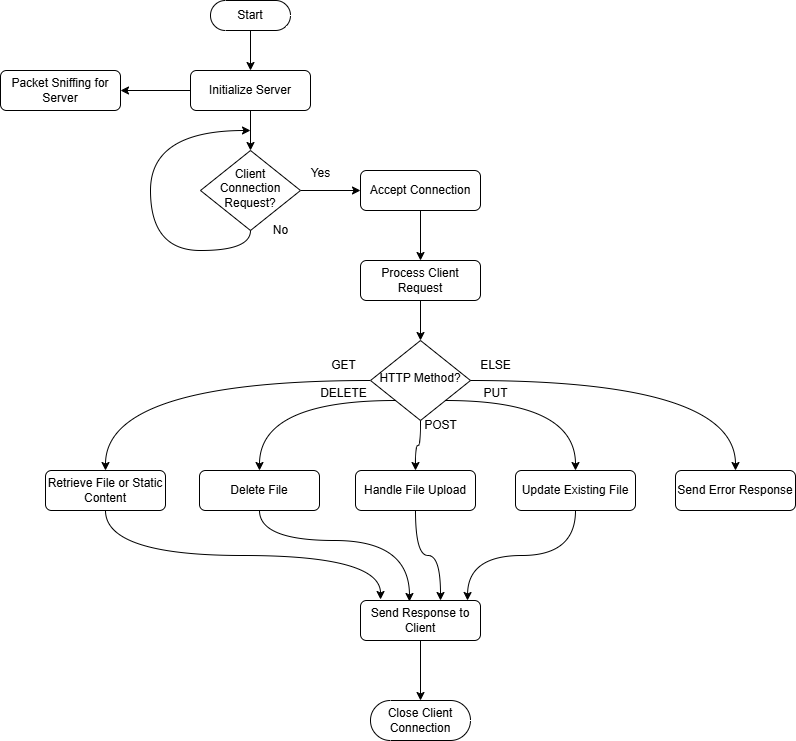


Figure . System Workflow

# **Tool Screenshots**

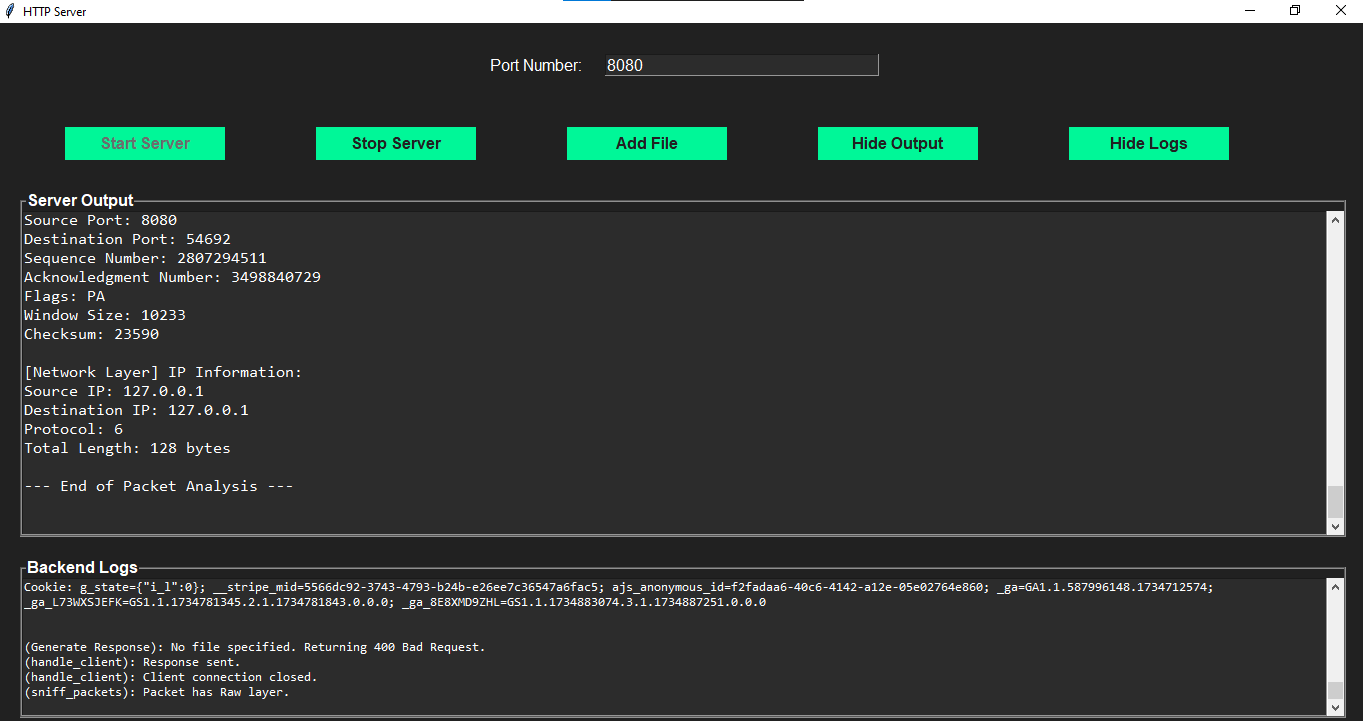


Figure . Server Page

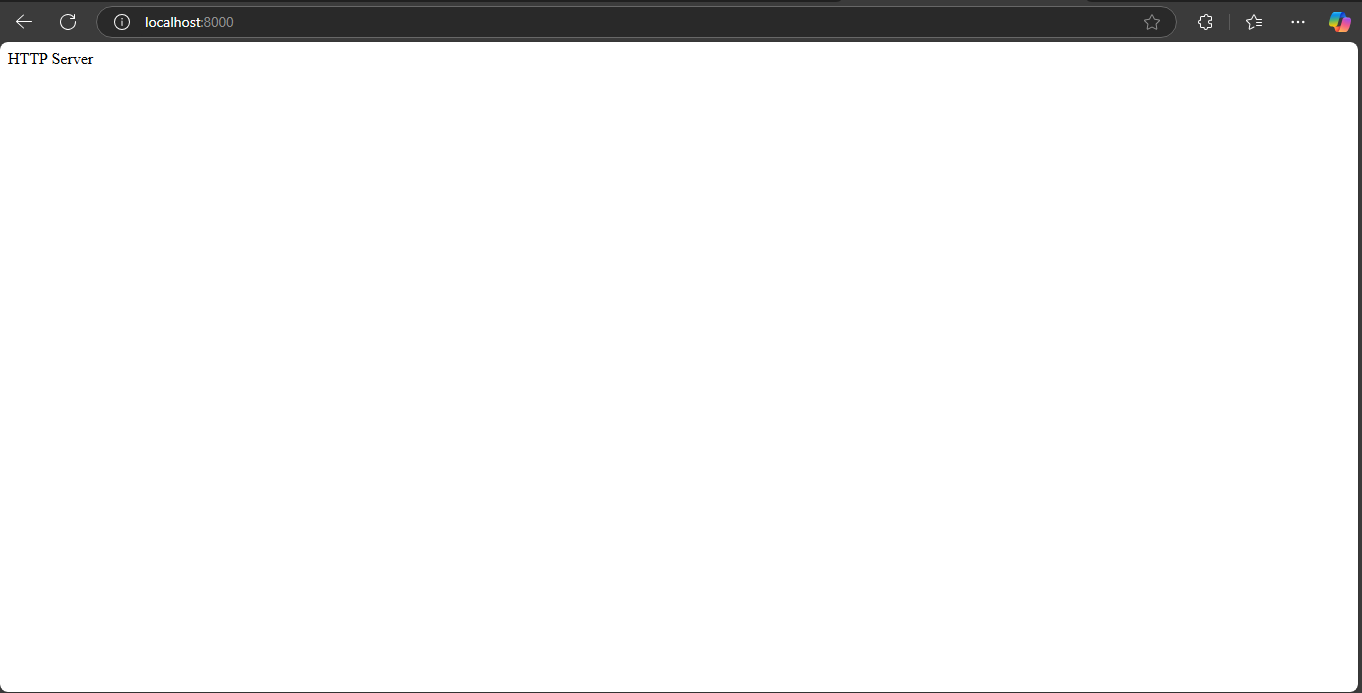


Figure . Client requesting server

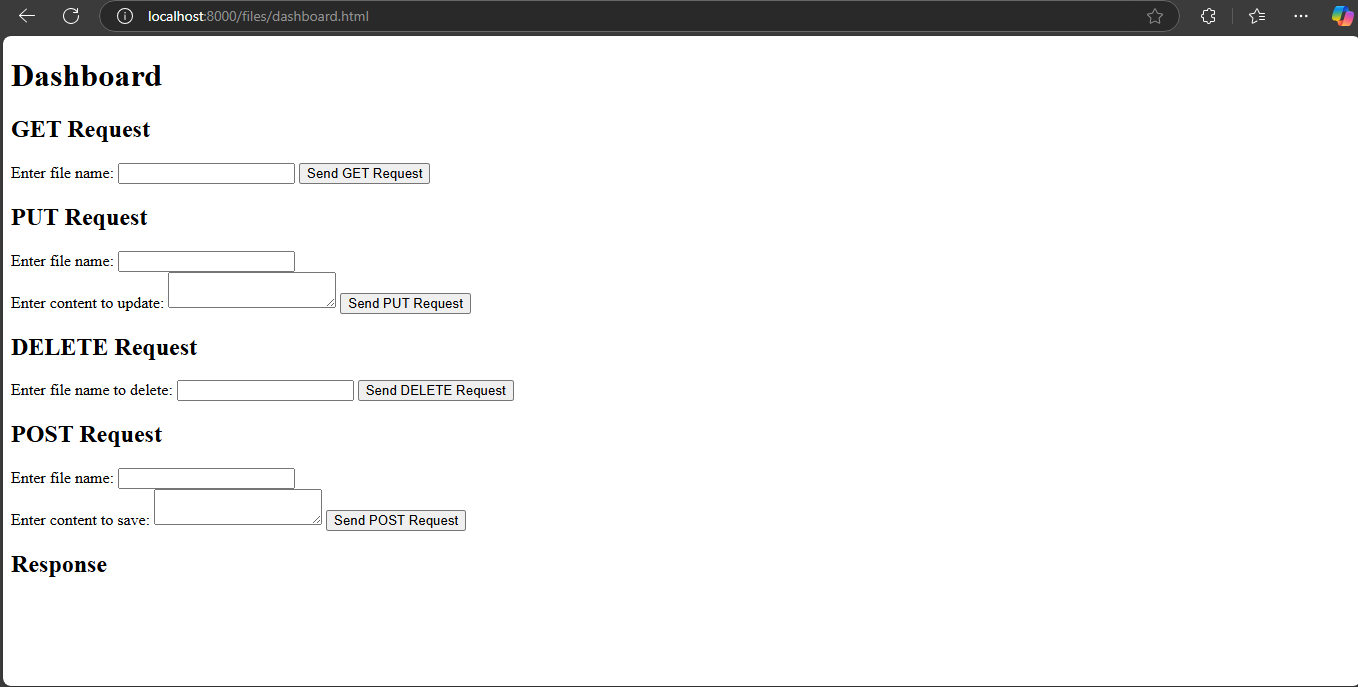


Figure . Sending Requests to server

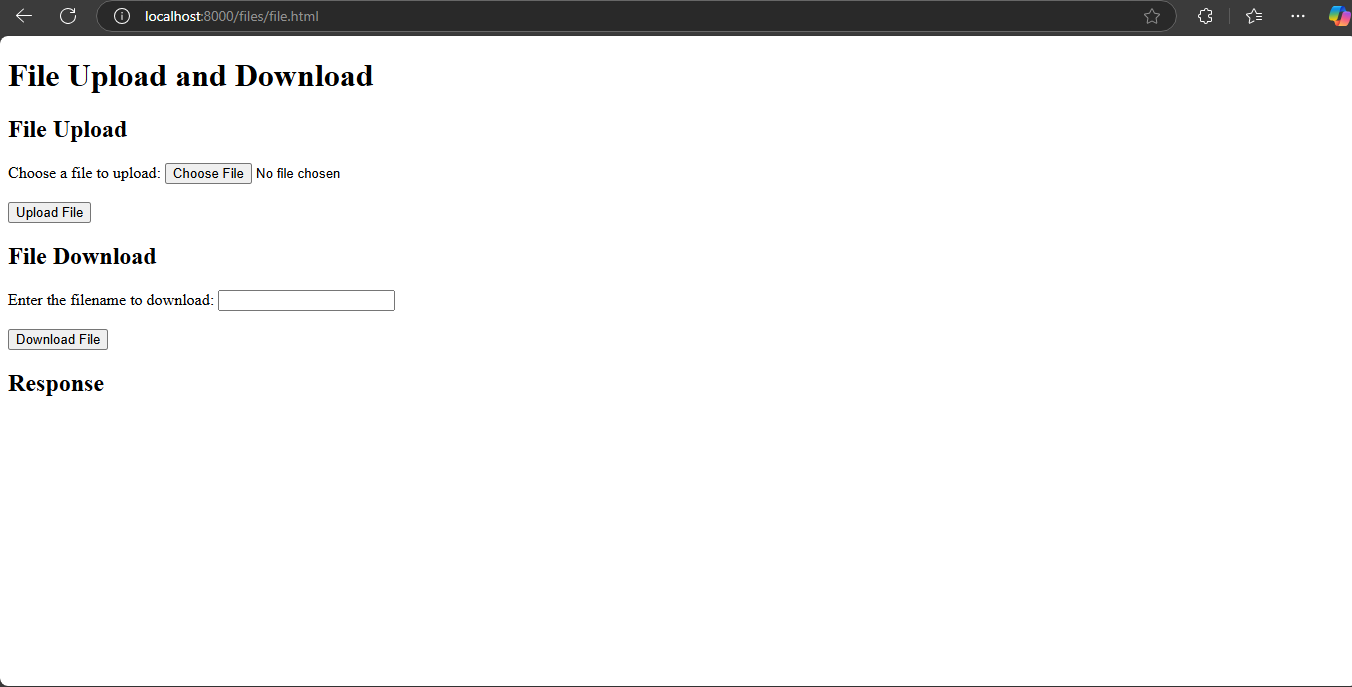


Figure . File Handling for the server

# 

# **Results and Analysis**

The project successfully met its objective and performed its intended functionalities. It is built with an operational GUI-based HTTP server with support for HTTP methods and real-time packet sniffing for the server’s requests and responses. The project is perfectly aligned with the objectives of the project to grasp and strengthen the understanding of computer network protocols and communication. The project is developed to understand the computer network protocols. The packet sniffing features provides an in-depth view of network communications for understanding OSI model and HTTP protocol. The server has been tested and performs its intended functionalities.

# **Conclusion**

The project achieved its primary objective of developing a user-friendly GUI to interact with HTTP server. It simplifies server operations and enhances the understanding of the network protocols and OSI model. There were several challenges faced in the development of the server. Some issues were encountered with the file handling on the server side. The key takeaways from the project are the understanding of the HTTP protocol and networking insights through packet sniffing. Improvements can be performed on this project by adding advanced security features like more advanced authorization, authentication, encryption / decryption of data and CORS.

# **Github Link**

Clone this repository and then navigate to the folder. Then run the server(gui).py file to start the server and then move to browser and request for this process by link localhost:port. Use the port that will be used to start the server.

<https://github.com/MohammadAbdullah5/HTTP-Server>

# 

# **References**

|  |  |  |  |
| --- | --- | --- | --- |
| [1] | "What is OSI Model - Layers of OSI Model," 9 December 2024. [Online]. Available: https://www.geeksforgeeks.org/open-systems-interconnection-model-osi/. |  |  |
| [2] | "What is HTTP?," Geeks for Geeks, 1 April 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-http/. |  |  |
| [3] | "Writing an HTTP server from scratch," 16 November 2017. [Online]. Available: https://bhch.github.io/posts/2017/11/writing-an-http-server-from-scratch/. |  |  |
| [4] | "Multi-threading in Python 3," Java T Point, [Online]. Available: https://www.javatpoint.com/multithreading-in-python-3. |  |  |
| [5] | "Socket Programming in Python (Guide)," Real Python, 7 December 2024. [Online]. Available: https://realpython.com/python-sockets/. |  |  |