# Project 1 HTTP Server Report

## Devan Rivera and Ashley Villegas

## 2/2/2025

## Project Description

This project implements a multi-threaded HTTP server in C++. The server dynamically assigns a port between 60001-60099, handles HTTP GET requests, and serves HTML and image files.

It correctly returns:

- 200 OK for valid files.

- 404 Not Found for missing files.

- 400 Bad Request for unsupported HTTP methods.

The server runs indefinitely until manually stopped and supports multiple clients simultaneously using multithreading.

## System Protocol’s

Protocol’s Utilized:

* DNS (Domain Name System)
  + Application Layer
* TCP (Transmission Control Protocol)
  + Transport Layer
* HTTP/1.1(HyperText Transfer Protocol)
  + Application Layer

These are the necessary protocols that allow a browser client to communicate and send messages to an HTTP server. DNS is used to convert domain names into IP addresses for our HTTP server to locate. TCP is used to provide reliable, in order, and error-checked data between the HTTP server and the client communicating with it. TCP utilizes a three-way handshake to establish a connection between client and server. HTTP/1.1 is used to define the structure of requests and responses for the client and server communication. By utilizing this protocol structure, an HTTP server can be made to communicate data reliably, securely, and efficiently.

## Testing of HTTP Server (WSLa – UBUNTU)

A screenshot of a computer

Description automatically generated

Figure 1. This is the initial webpage that the server link sends you

A computer screen shot of a bird

Description automatically generated

Figure 2. This is the IMAGE LINK and the subsequent page

A computer screen shot of a black screen

Description automatically generated

Figure 3. This is the nonexistent link and the subsequent error page

A screenshot of a computer

Description automatically generated

Figure 3. This is the nonexistent link and the subsequent error page

A screen shot of a computer

Description automatically generated

Figure 4. The command prompt output once the server is offline

A screenshot of a computer

Description automatically generated

Figure 5. The webpage with the server fully disconnected

## Testing of HTTP Server (Putty)



Figure 6. The command to test the httpServer through the terminal

A screenshot of a computer program

Description automatically generated

Figure 7. The successful output once the command has been entered.

A computer screen shot of a black screen

Description automatically generated

Figure 8. The expected failure once the command has been entered

A computer screen shot of a black screen

Description automatically generated

Figure 9. The expected failure once the command has been entered.