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)

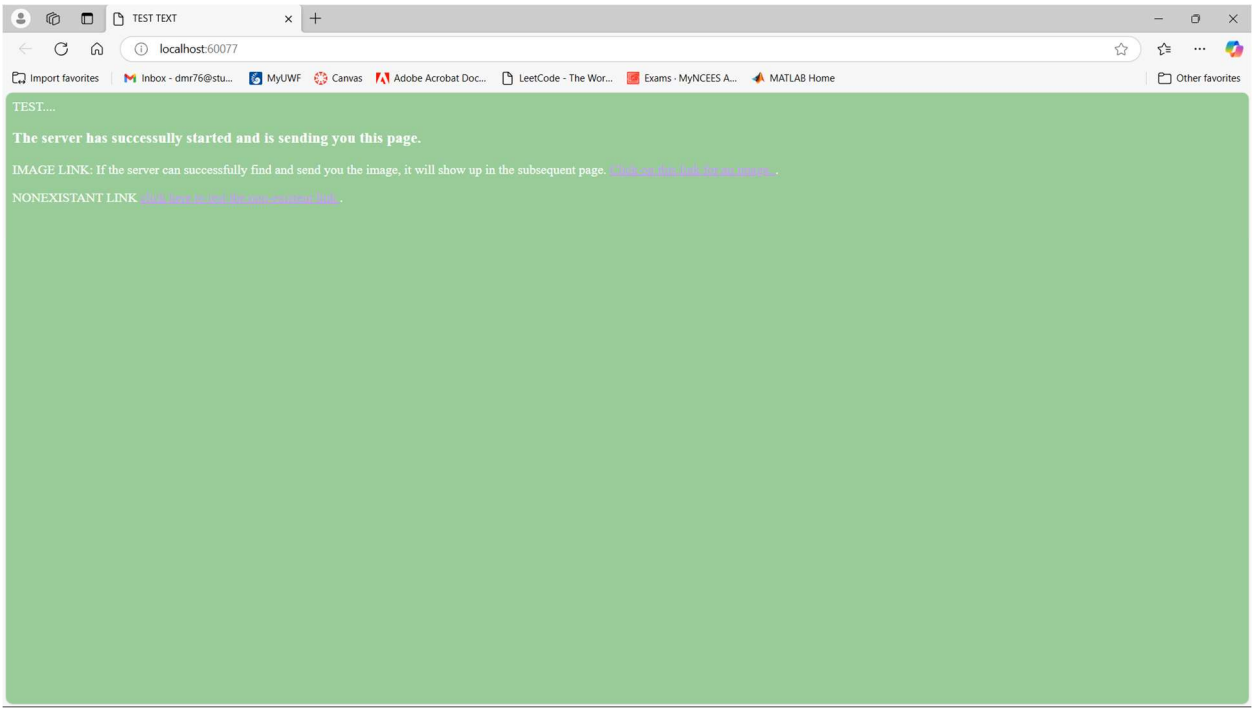

Figure 1. This is the initial webpage that the server link sends you

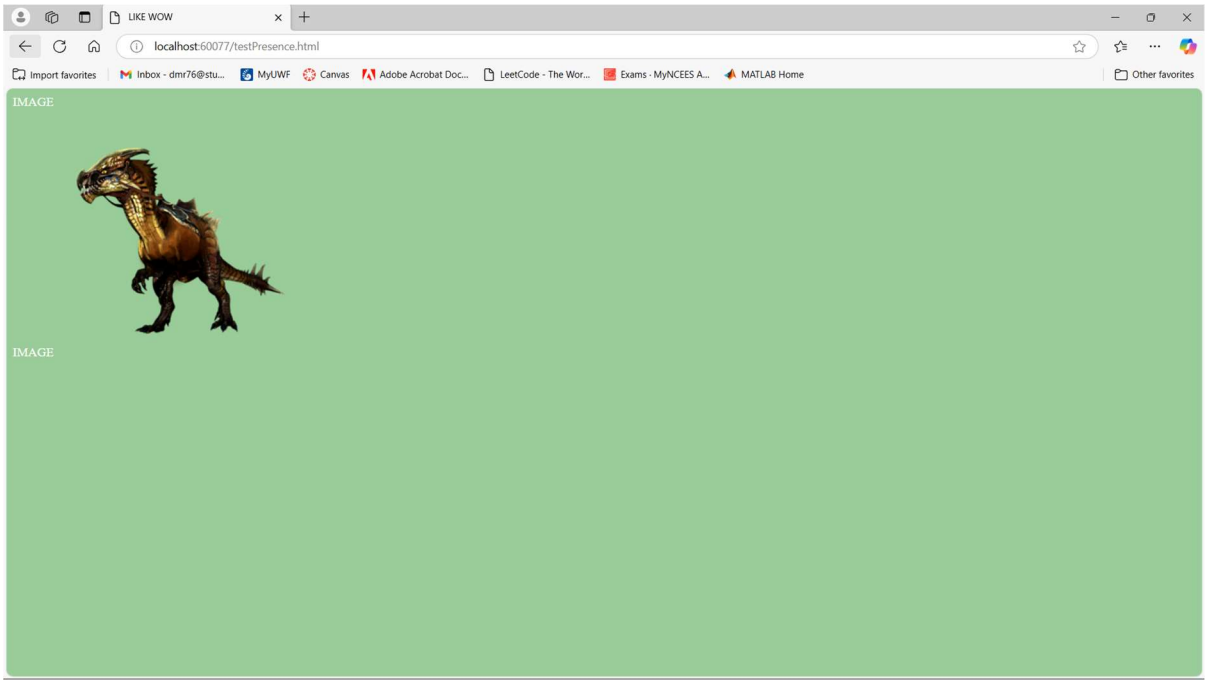

Figure 2. This is the IMAGE LINK and the subsequent page

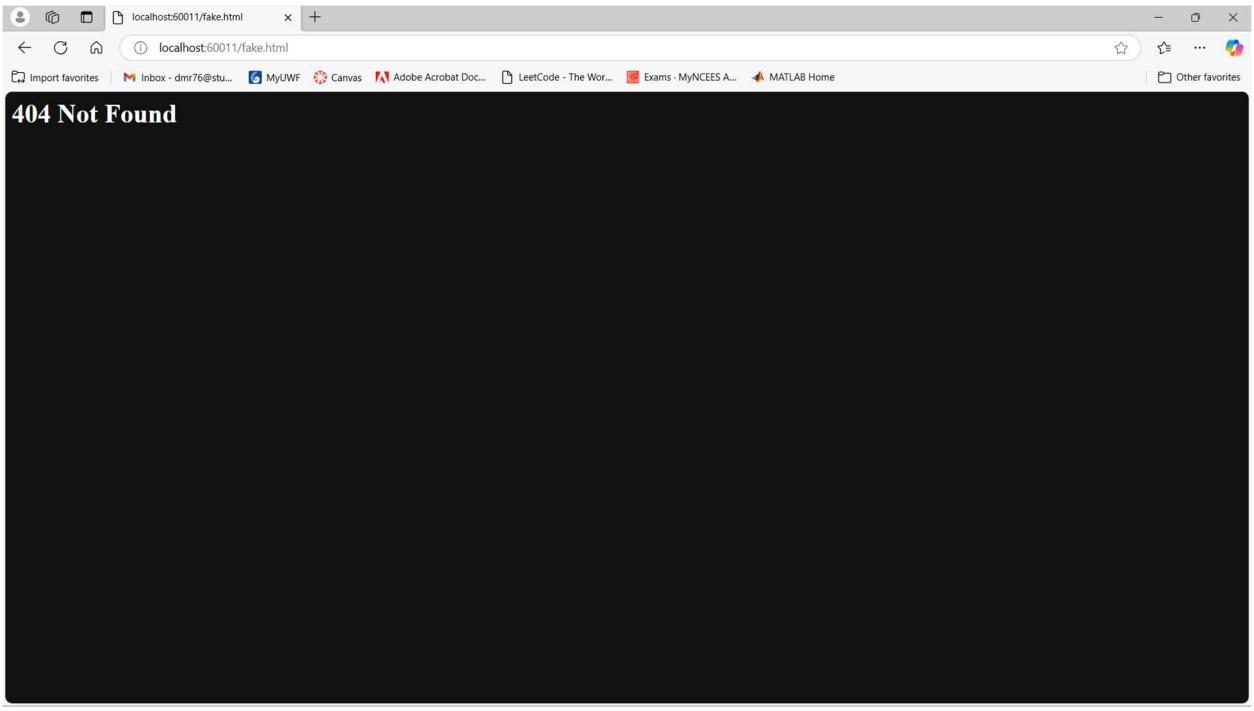

Figure 3. This is the nonexistent link and the subsequent error page

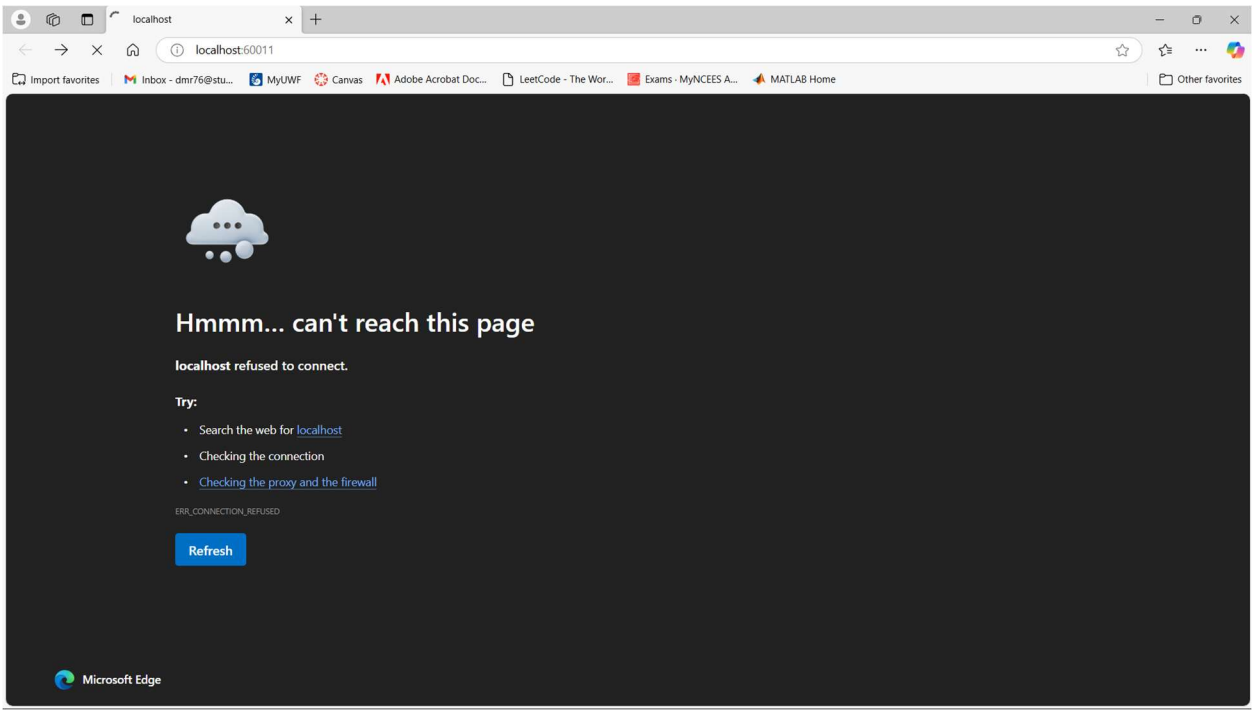

Figure 3. This is the nonexistent link and the subsequent error page

```
Server running on port: 60011
Input in URL: http://localhost:60011
Note: press Ctrl + C to stop the server
Waiting for incoming connections...
Client connected: 127.0.0.1
Client connected: 127.0.0.1
Client connected: 127.0.0.1
Client connected: 127.0.0.1
Client connected: 127.0.0.1
Client connected: 127.0.0.1
terminate called after throwing an instance of 'std::out_of_range'
  what():  basic_string::substr: __pos (which is 1) > this->size() (which is 0)
Aborted (core dumped)
```

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

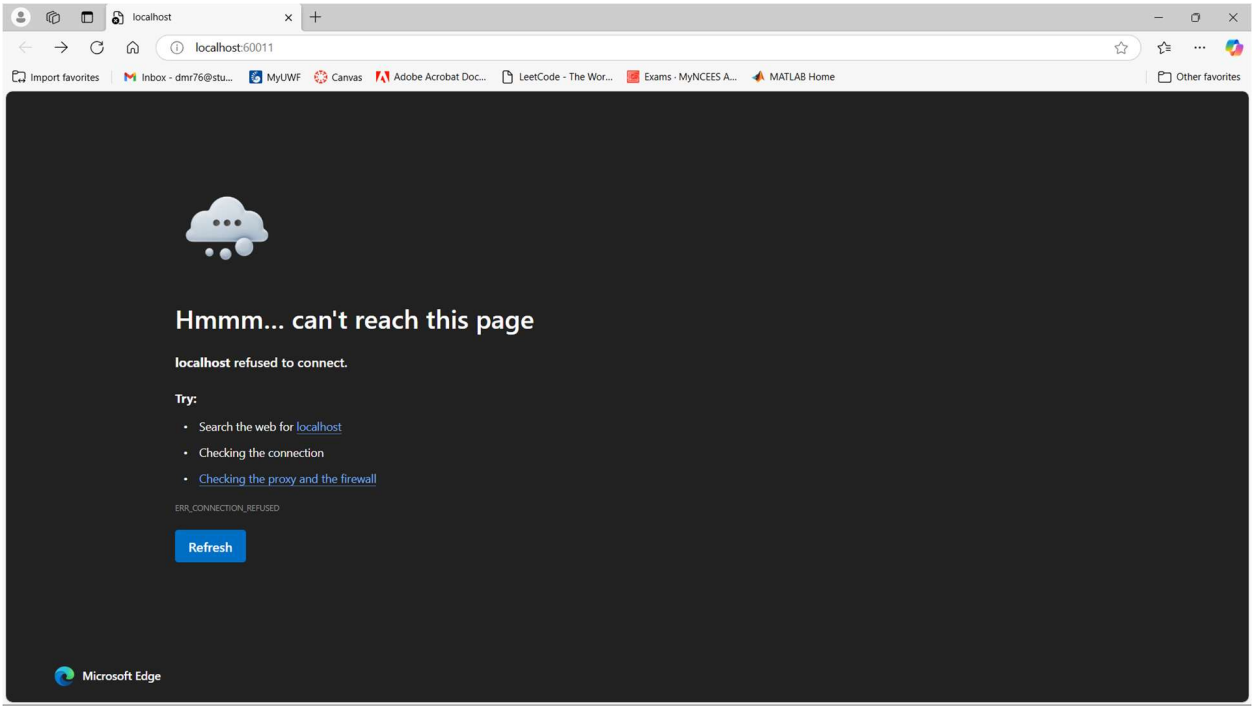

Figure 5. The webpage with the server fully disconnected

Testing of HTTP Server (Putty)

```
[dmr76@cs-ssh Sys2Proj1]$ curl -v http://localhost:60054
```

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

```
> Host: localhost:60007
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content Type: text/html
< Content Length: 22
< Connection: close
<
* Closing connection 0
<h1>404 Not Found</h1>[dmr76@cs-ssh Sys2Proj1]$ curl -v http://localhost:60054
* Rebuilt URL to: http://localhost:60054/
*   Trying ::1...
* TCP_NODELAY set
* connect to ::1 port 60054 failed: Connection refused
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 60054 (#0)
> GET / HTTP/1.1
> Host: localhost:60054
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content Type: text/html
< Content Length: 503
< Connection: close
<
<HTML>
<HEAD><TITLE>TEST TEXT</TITLE></HEAD>
<BODY BGCOLOR="#99cc99" TEXT="#000000" LINK="#2020ff" VLINK="#4040cc">


<P>
TEST....
<P>

<H3> The server has successully started and is sending you this page. </H3>

<P>
IMAGE LINK: If the server can successfully find and send you the image, it will
show up in the subsequent page. <A HREF="testPresence.html">Click on this link f
or an image. </A>.

<P>
NONEXISTANT LINK <A HREF="fake.html">click here to test the non-existent link </
A>.

</BODY>
</HTML>
* Closing connection 0
[dmr76@cs-ssh Sys2Proj1]$
```

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

```
[dmr76@cs-ssh Sys2Proj1]$ curl -v http://localhost:60054/fake.html
*   Trying ::1...
* TCP_NODELAY set
* connect to ::1 port 60054 failed: Connection refused
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 60054 (#0)
> GET /fake.html HTTP/1.1
> Host: localhost:60054
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content Type: text/html
< Content Length: 22
< Connection: close
<
* Closing connection 0
```

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

```
[dmr76@cs-ssh Sys2Proj1]$ curl -v http://localhost:60054/fake.html
*   Trying ::1...
* TCP_NODELAY set
* connect to ::1 port 60054 failed: Connection refused
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 60054 (#0)
> GET /fake.html HTTP/1.1
> Host: localhost:60054
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content Type: text/html
< Content Length: 22
< Connection: close
<
* Closing connection 0
<h1>404 Not Found</h1>[dmr76@cs-ssh Sys2Proj1]$ curl -X POST -v  http://localhost:60054/index.html
*   Trying ::1...
* TCP_NODELAY set
* connect to ::1 port 60054 failed: Connection refused
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 60054 (#0)
> POST /index.html HTTP/1.1
> Host: localhost:60054
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Content Type: text/html
< Content Length: 24
< Connection: close
<
* Closing connection 0
```

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