Project ReportNetwork Applications Project Report

short line

Syed Aalyan Raza Kazmi

22i 0833

CS - 5A  
4th December, 2024

# Objectives**:**

The primary objective of this project is to develop a collection of network applications to explore and demonstrate fundamental concepts of network programming, including ICMP ping, traceroute, multi-threaded traceroute, a web server, and a proxy server. The project leverages socket programming to provide efficient and scalable network solutions.

---

# Technologies Used:

* - **Programming Language**:
  + Python 3
* - **Libraries:**
  + - “*socket”* for low-level networking
  + - “*argparse”* for command-line argument parsing
  + - “threading” for multi-threaded implementation
  + - “*os*” and “*sys*” for system-level operations
  + - “*struct*” for handling binary data
  + - “*time”* for performance measurement
* **Tools:**
  + - Command-line Interface
  + - Basic text editor/IDE

---

# Implementation Details:

## Design and Approach

The project is designed as a command-line application with modular functionality. Each feature (ping, traceroute, web server, proxy server) is implemented as a class that inherits from a base class, ensuring extensibility and maintainability. The `argparse` library handles sub-command arguments to provide a user-friendly interface.

## Cisco/Socket Programming Concepts

**Raw Sockets:** Used for ICMP ping and traceroute functionalities.

**UDP and TCP Sockets:** Applied in traceroute, web server, and proxy server implementations.

**Multi-threading:** Enhances performance and responsiveness, especially for multi-threaded traceroute and handling concurrent web/proxy server requests.

## Key Code Snippets

**Example: ICMP Ping Implementation**

Send ICMP Echo Request

packet = struct.pack('!BBHHH', ICMP\_ECHO\_REQUEST, 0, checksum, packet\_id, sequence\_number)

sock.sendto(packet, (destination, 1))

Receive ICMP Echo Reply

response, addr = sock.recvfrom(65535)

**Example: Web Server Request Handling**

with open(filename[1:], 'r') as f:

content = f.read()

response = 'HTTP/1.1 200 OK\r\n\r\n' + content

connection\_socket.send(response.encode())

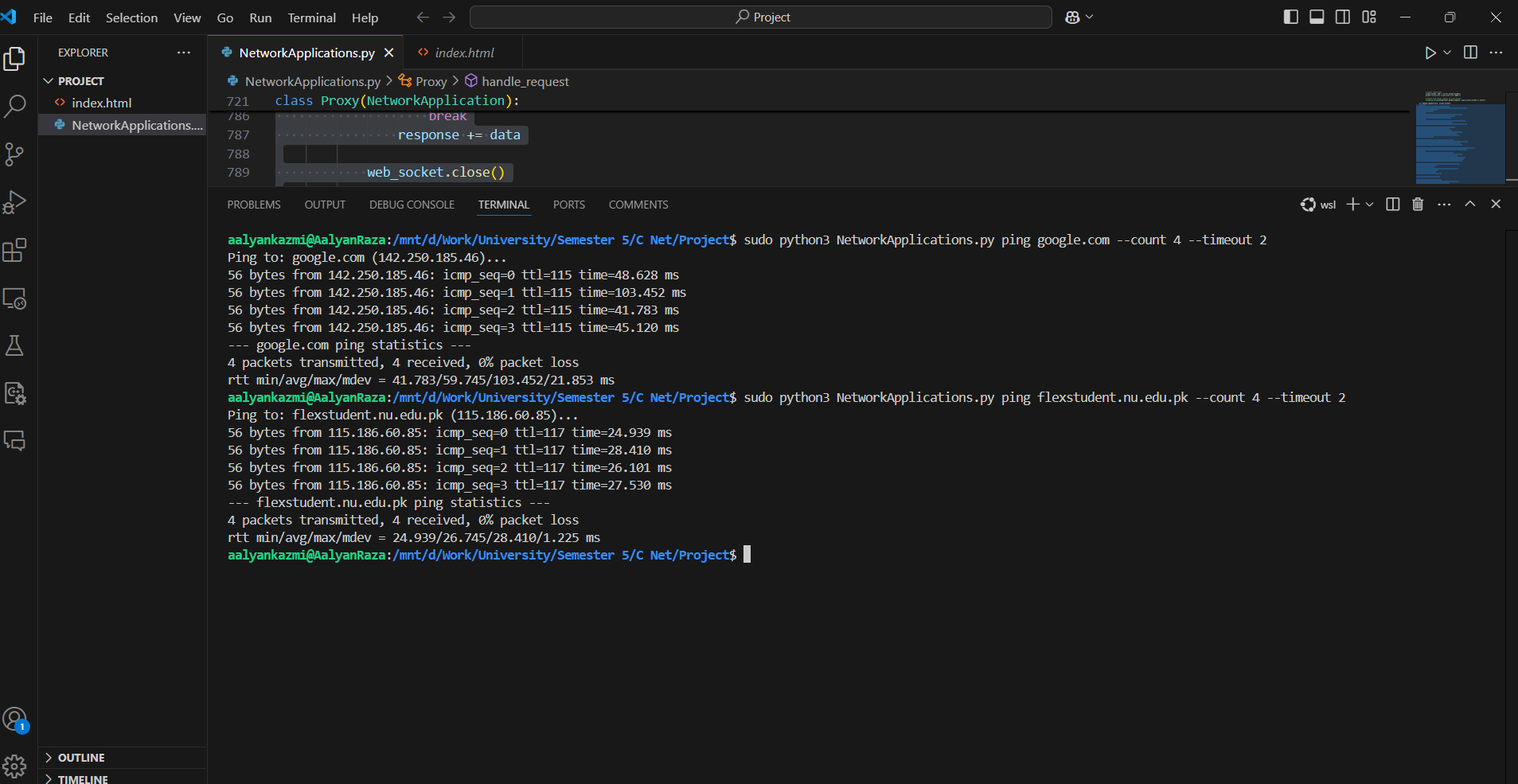
```

---

# Results and Testing

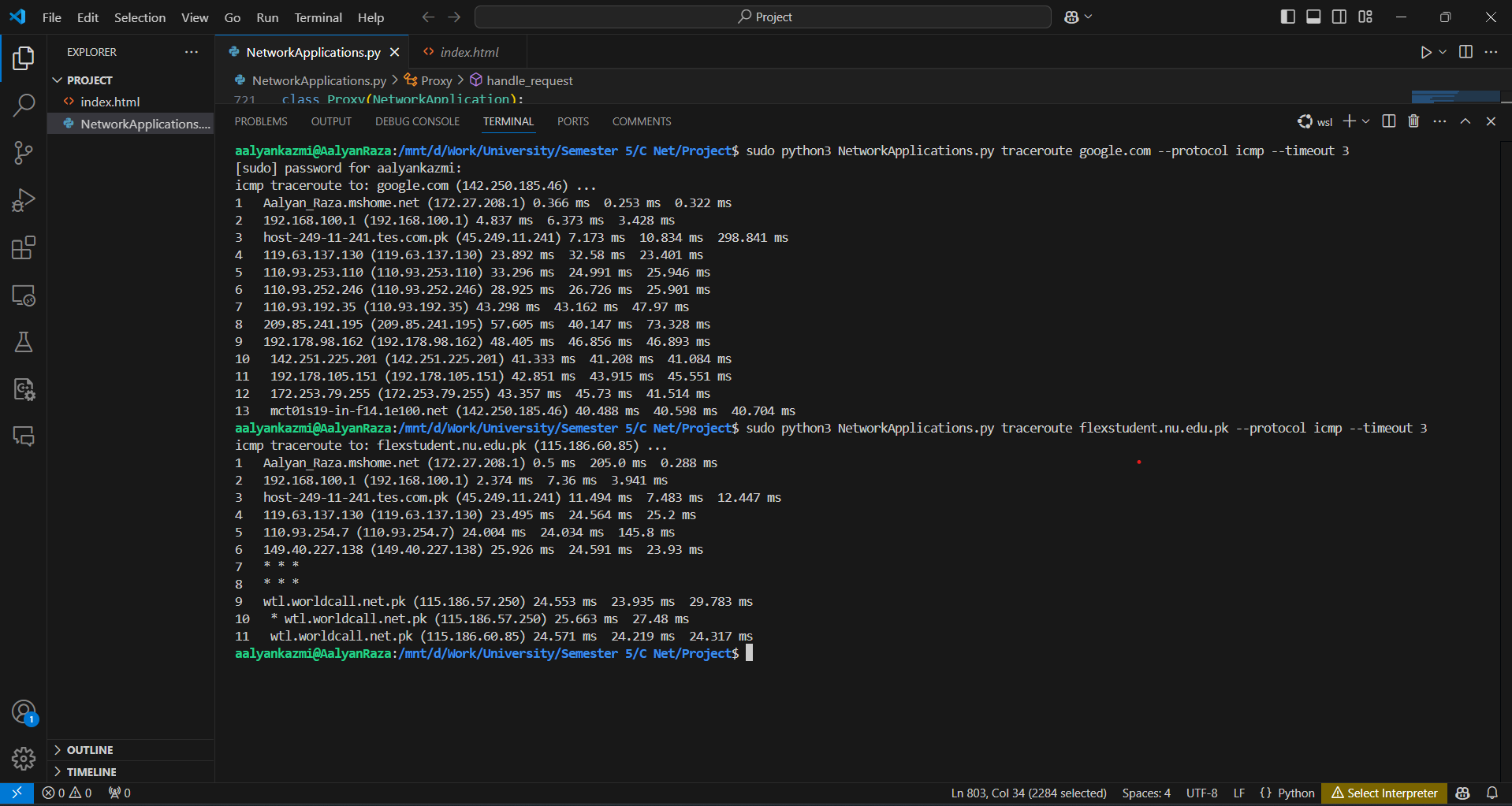
## Ping Command:

Successfully sends ICMP echo requests and displays round-trip times.



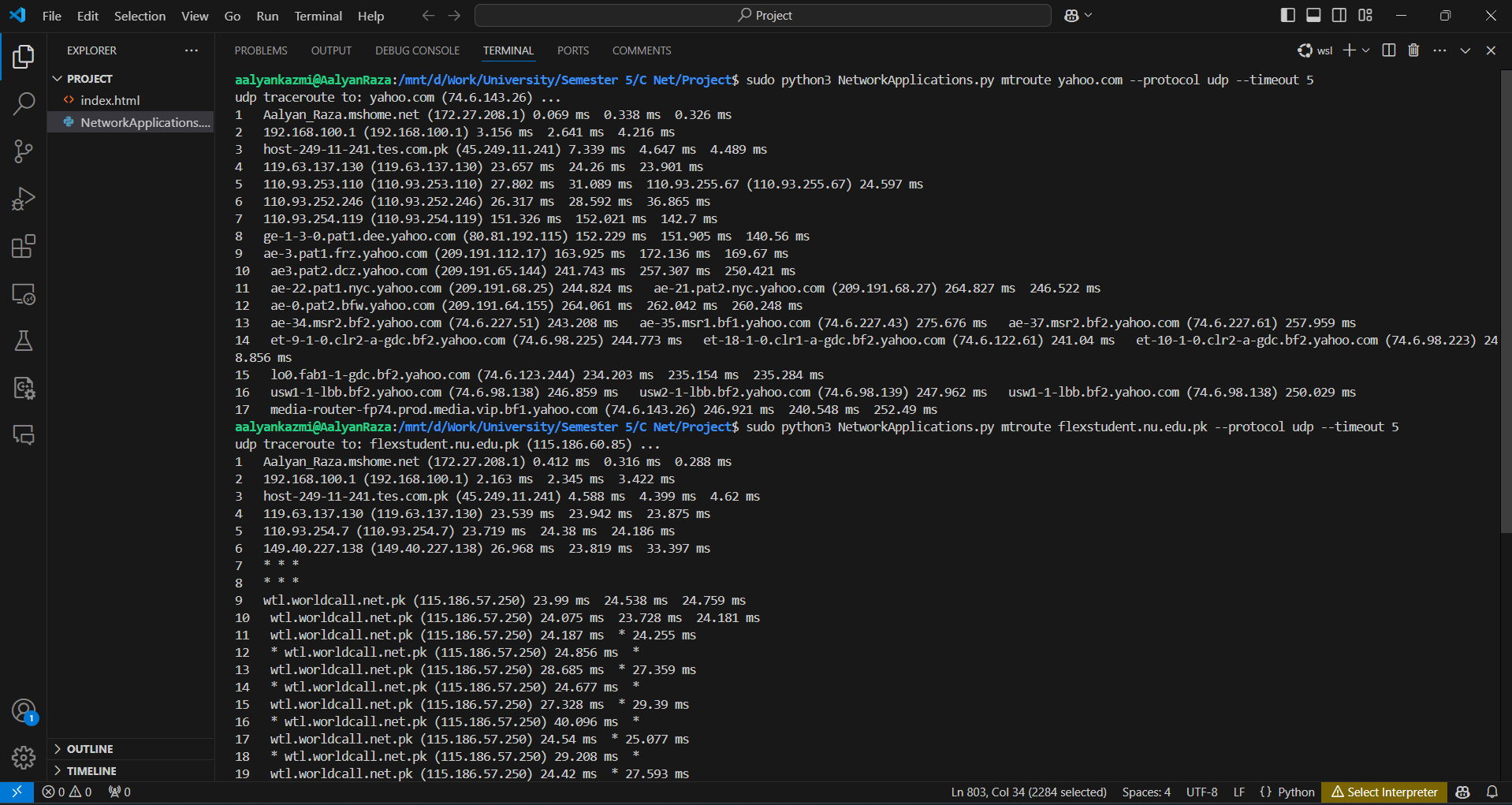
## Traceroute Command:

Maps the path packets take to the destination.



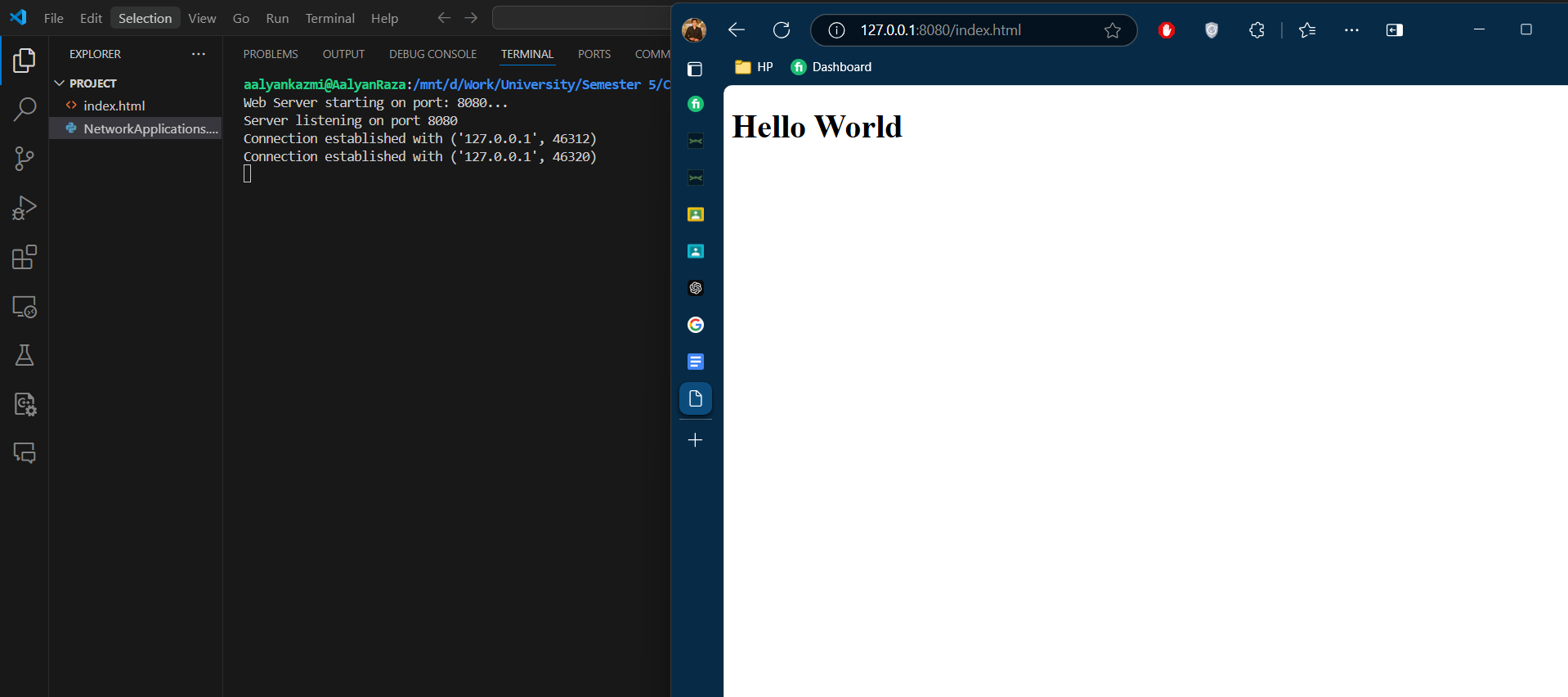
## Multi-threaded Traceroute:

Improves performance by concurrently sending and receiving packets.



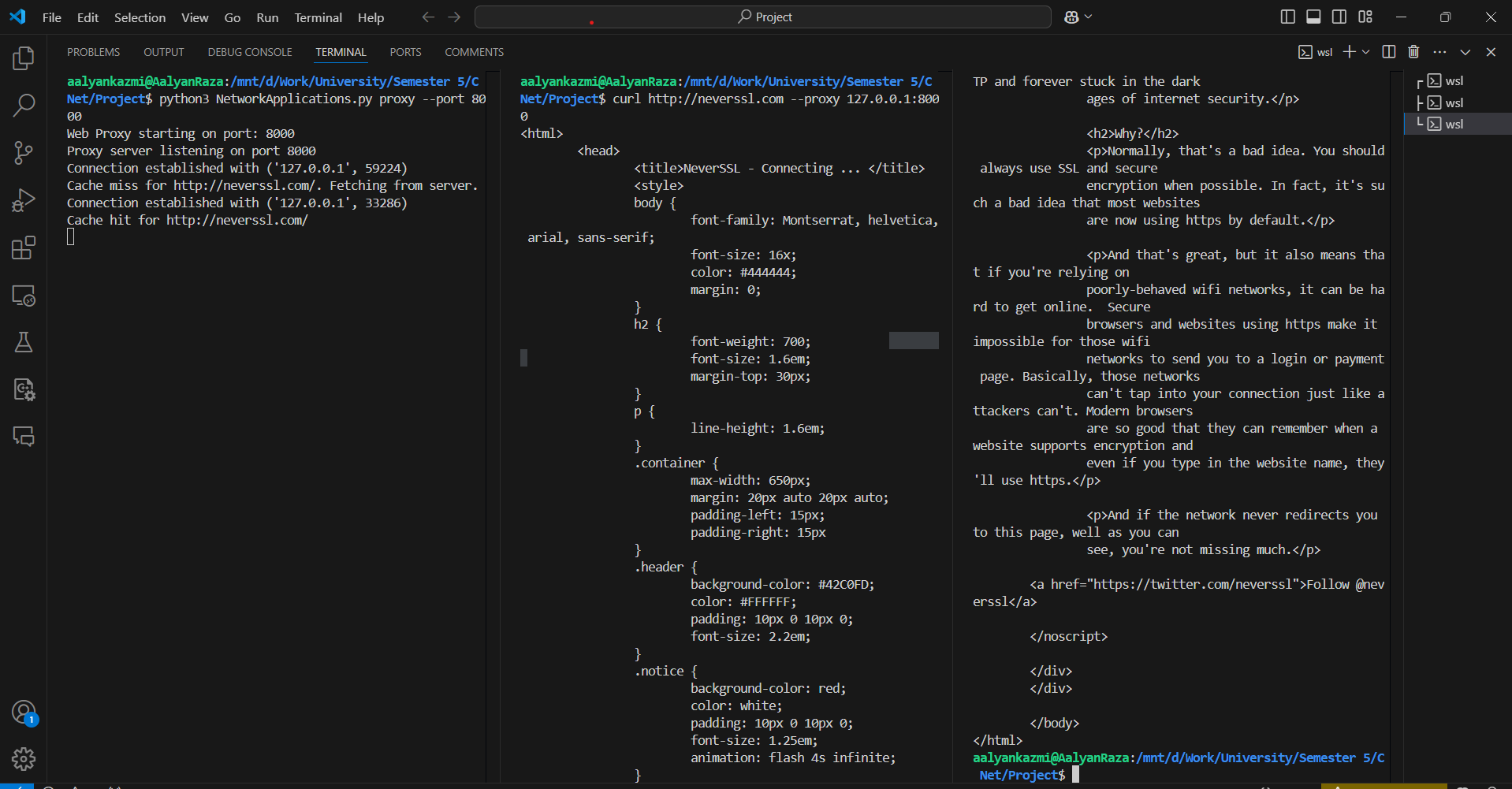
## Web Server:

Serves static HTML files.



## Proxy Server:

Fetches and caches web content.



# Challenges and Learnings:

1. **Raw Socket Permissions:** Overcame administrative permission issues for creating raw sockets.
2. **Timeout Handling:** Implemented robust error handling for timeouts.
3. **Concurrency Bugs:** Debugged thread synchronization issues in multi-threaded traceroute.

---

# Conclusion

This project successfully implements essential network applications using Python. The modular design ensures easy scalability, and the use of socket programming provides an in-depth understanding of networking concepts. Future improvements could include:

* Support for IPv6
* Enhanced logging and reporting features
* Improved user interface with additional customization options

---