Advanced Computer Networks Lab

Dr Sudipta Saha
Associate Professor
Dept of Computer Science & Engineering
Indian Institute of Technology Bhubaneswar

Lab Assignment-1

Teaching Assistants: Subham and Lamia

DSSRG: Decentralized Smart Systems Research Group

https://sites.google.com/iitbbs.ac.in/dssrg
Or Google dssrg iitbbs

About

- Code to interact with the standards
- Customized client programs to interact with existing web servers

- Customized web servers to provide a specific type of service and can be interacted with normal browsers

Types of servers [Example…]

1. Web Server
2. Database Server
3. Email Server
4. DNS Server
5. FTP Server
6. DHCP Server
7. Cloud Server
8. Print Server
9. NTP Server

…

Know the language a server speaks:

HTTP/HTTPS

Assignment 1.1

100 Marks

Problem Statement

The Hypertext Transfer Protocol (HTTP) is an application-layer protocol that operates over the Transmission Control Protocol (TCP). Modern web browsers act as HTTP clients and communicate with web servers by sending HTTP requests and receiving HTTP responses over TCP connections.

In this experiment, you are required to implement a simple HTTP server using TCP/IP socket programming in C, which can communicate with a standard web browser without using any external libraries or frameworks.

Assignment 1.2

100 Marks

Problem Statement

Implement a secure HTTPS client in C that connects to the search interface of DuckDuckGo. The client program should ask the user for a search query and then it should retrieve the corresponding search results as an HTML response.

Assignment 1.2

Expected Output

- Successful establishment of a TLS connection.
- Display of the negotiated TLS cipher suite.

```
Subham@systems-HP-ProOne-440-AIO:~/ACN/ACN Test Programs$ gcc duckduckgo_https.c -o duckduckgo_https -lssl -lcrypto
Subham@systems-HP-ProOne-440-AIO:~/ACN/ACN Test Programs$ ./duckduckgo_https
Enter search query: IIT BBS

[+] TLS established using TLS_AES_256_GCM_SHA384
```

- Receipt of an HTTP `200 OK` response.

```
Subham@systems-HP-ProOne-440-AIO:~/ACN/ACN Test Programs$ gcc duckduckgo_https.c -o duckduckgo_https -lssl -lcrypto
Subham@systems-HP-ProOne-440-AIO:~/ACN/ACN Test Programs$ ./duckduckgo_https
Enter search query: IIT BBS

[+] TLS established using TLS_AES_256_GCM_SHA384

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 09 Jan 2026 07:03:40 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Vary: Accept-Encoding
Server-Timing: total;dur=1824;desc="Backend Total [n]"
Vary: Origin
```

Assignment 1.2

Expected Output

- HTML content containing real search results returned by the search engine.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!--[if IE 6]><html class="ie6" xmlns="http://www.w3.org/1999/xhtml"><![endif]-->
<!--[if IE 7]><html class="lt-ie8 lt-ie9" xmlns="http://www.w3.org/1999/xhtml"><![endif]-->
<!--[if IE 8]><html class="lt-ie9" xmlns="http://www.w3.org/1999/xhtml"><![endif]-->
<!--[if gt IE 8]><!--><html xmlns="http://www.w3.org/1999/xhtml"><!--<![endif]-->
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=3.0, user-scalable=1" />
  <meta name="referrer" content="origin" />
  <meta name="HandheldFriendly" content="true" />
  <meta name="robots" content="noindex, nofollow" />
  <title>IIT BBS at DuckDuckGo</title>
  <link title="DuckDuckGo (HTML)" type="application/opensearchdescription+xml" rel="search" href="//duckduckgo.com/opensearch_html_v2.xml" />
  <link href="//duckduckgo.com/favicon.ico" rel="shortcut icon" />
  <link rel="icon" href="//duckduckgo.com/favicon.ico" type="image/x-icon" />
  <link id="icon60" rel="apple-touch-icon" href="//duckduckgo.com/assets/icons/meta/DDG-iOS-icon_60x60.png?v=2"/>
  <link id="icon76" rel="apple-touch-icon" sizes="76x76" href="//duckduckgo.com/assets/icons/meta/DDG-iOS-icon_76x76.png?v=2"/>
  <link id="icon120" rel="apple-touch-icon" sizes="120x120" href="//duckduckgo.com/assets/icons/meta/DDG-iOS-icon_120x120.png?v=2"/>
  <link id="icon152" rel="apple-touch-icon" sizes="152x152" href="//duckduckgo.com/assets/icons/meta/DDG-iOS-icon_152x152.png?v=2"/>
  <link rel="image_src" href="//duckduckgo.com/assets/icons/meta/DDG-icon_256x256.png">
  <link rel="stylesheet" media="handheld, all" href="//duckduckgo.com/dist/h.048d08a46e4d9eef6e45.css" type="text/css"/>
</head>

<body class="body--html">
  <a name="top" id="top"></a>

  <form action="/html/" method="post">
    <input type="text" name="state_hidden" id="state_hidden" />
  </form>

  <div>
    <div class="site-wrapper-border"></div>

    <div id="header" class="header cw header--html">
      <a title="DuckDuckGo" href="/html/" class="header__logo-wrap"></a>
```

Assignment 1.2

Constraints:

- The program must use raw TCP sockets and OpenSSL; high-level HTTP libraries are not permitted.
- The program should not use JavaScript, browser automation tools, or external APIs.
- The program is expected to retrieve HTML content.

Assignment 1.3

100 Marks

Problem Statement

In the World Wide Web architecture, web pages are stored and managed on the server side, while clients (such as browsers or custom programs) interact with the server using the HTTP protocol.

A fundamental principle of web-based systems is that:

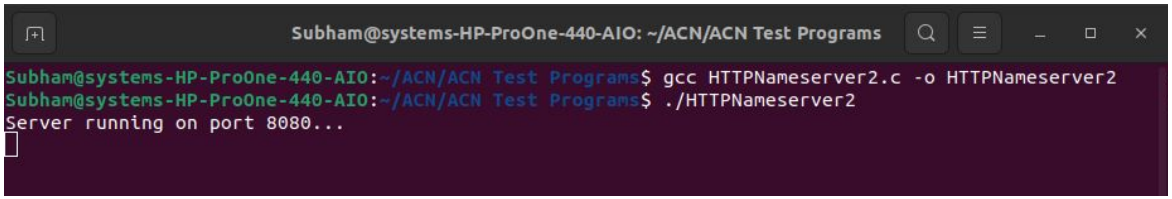
> "A client cannot directly modify files stored on a server.
> The client can only send a request, and the server decides whether and how to update its resources."

In this experiment, you will explore this principle by designing a TCP/IP-based HTTP system where a client program sends data to a server, and the server dynamically updates an HTML page based on the client's request.

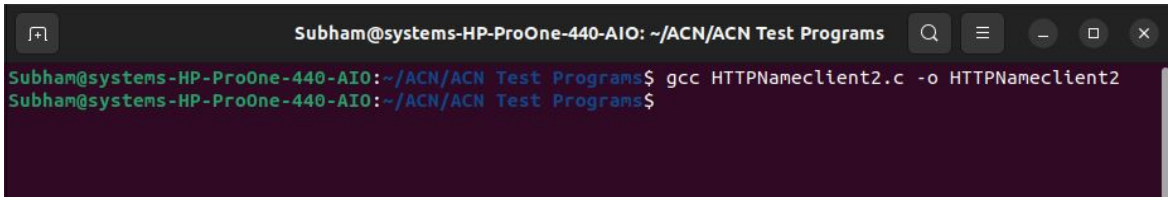
Assignment 1.3

Expected Output:

Server side Terminal:



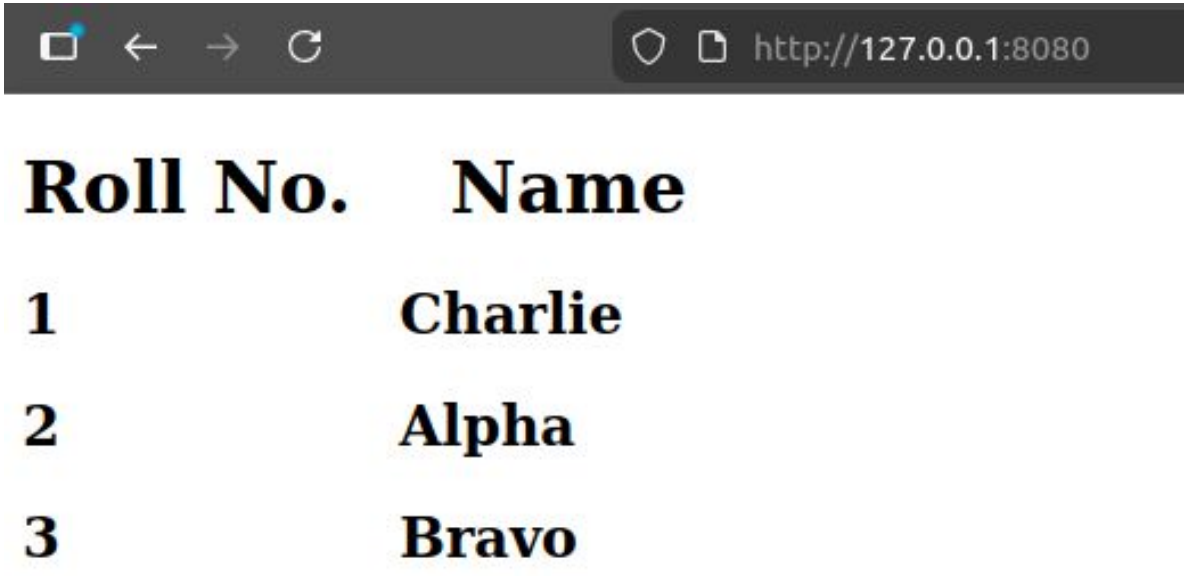
Client side Terminal:



Assignment 1.3

Expected Output:

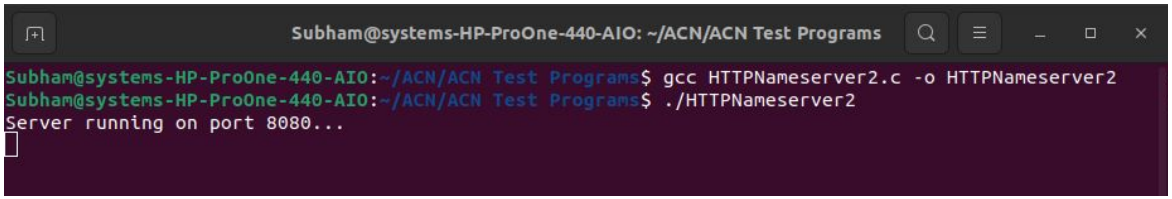
When the server program is executed, the names of all your classmates should be displayed in the sorted order of their roll numbers.



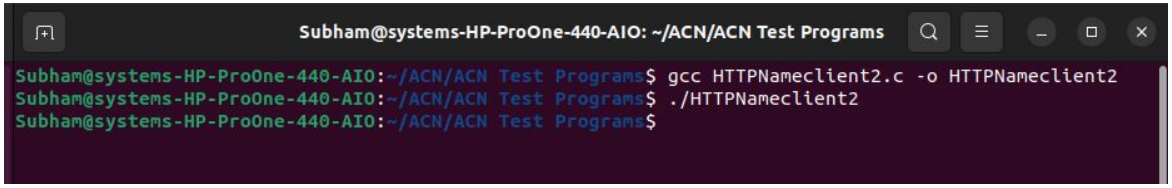
Assignment 1.3

Expected Output:

Server side Terminal:



Client side Terminal:



Assignment 1.3

Expected Output:

When the client program is executed, the names of all your classmates should be displayed in the sorted order of their first names.

Assignment 1.3

Constraints:

- Do not use external web frameworks or libraries.
- Use only:
  -Socket system calls
  -Standard C file I/O
  -Basic string processing
- HTTPS is not required.

Assignment 1.4

100 Marks

Problem Statement

Design and implement a TCP/IP-based HTTP server in C that supports the following custom services using a standard web browser as the client.

The server maintains a hard-coded list of names of all your classmate's first name stored in memory.

The server must provide the following services:

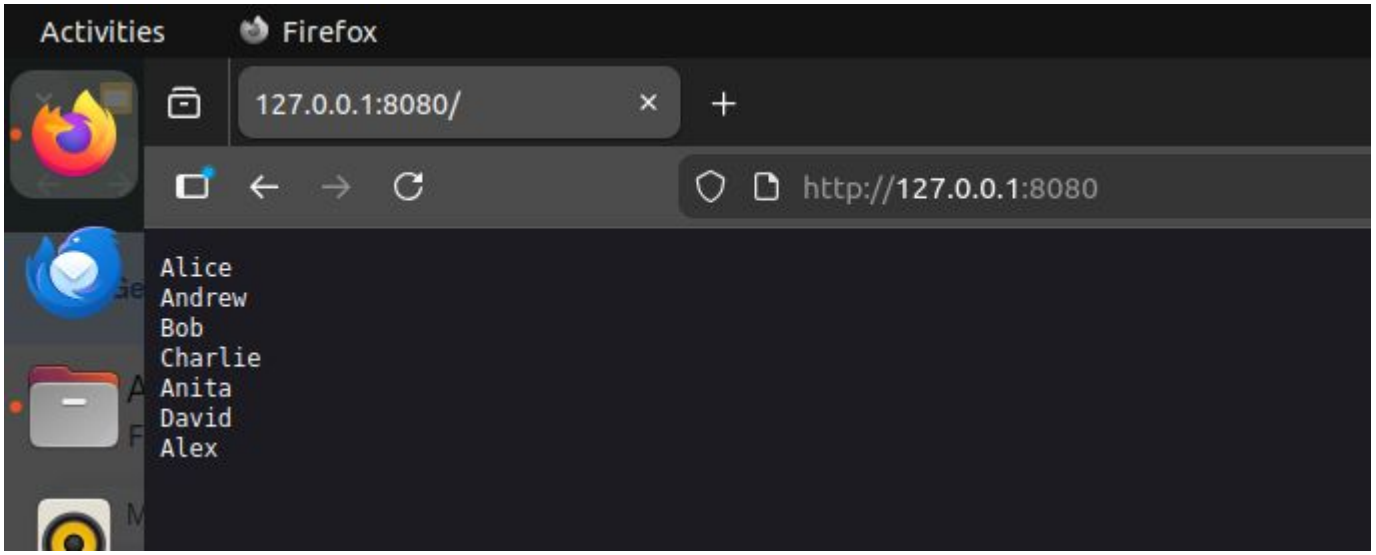
1. **COUNT:** Returns the total number of students stored on the server.
2. **COUNT with Regular Expression:** Returns the number of students whose names match a given regular expression.
3. **PRINT:** Prints the names of all students.
4. **PRINT with Regular Expression:** Prints only those student names that match a given regular expression.

All services must be accessed using HTTP GET requests sent by a web browser.

Assignment 1.4

Expected Output

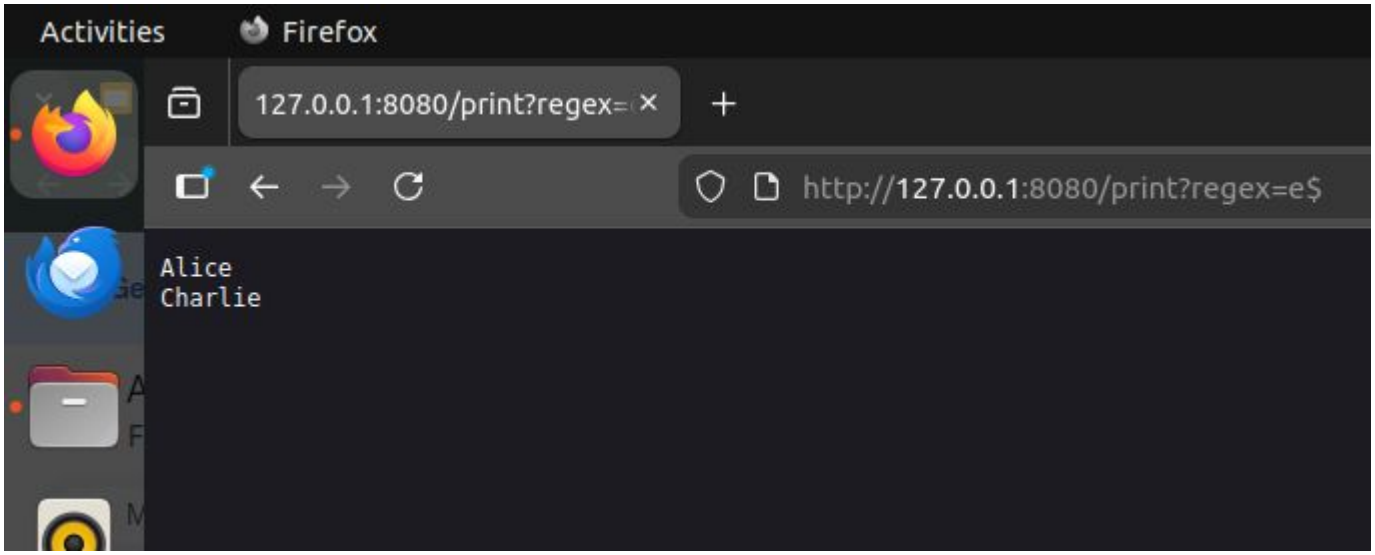
- When you'll visit the address http://127.0.0.1:8080/ or http://127.0.0.1:8080/print in your browser, you should be able to see the names of all your classmates.



Assignment 1.4

Expected Output

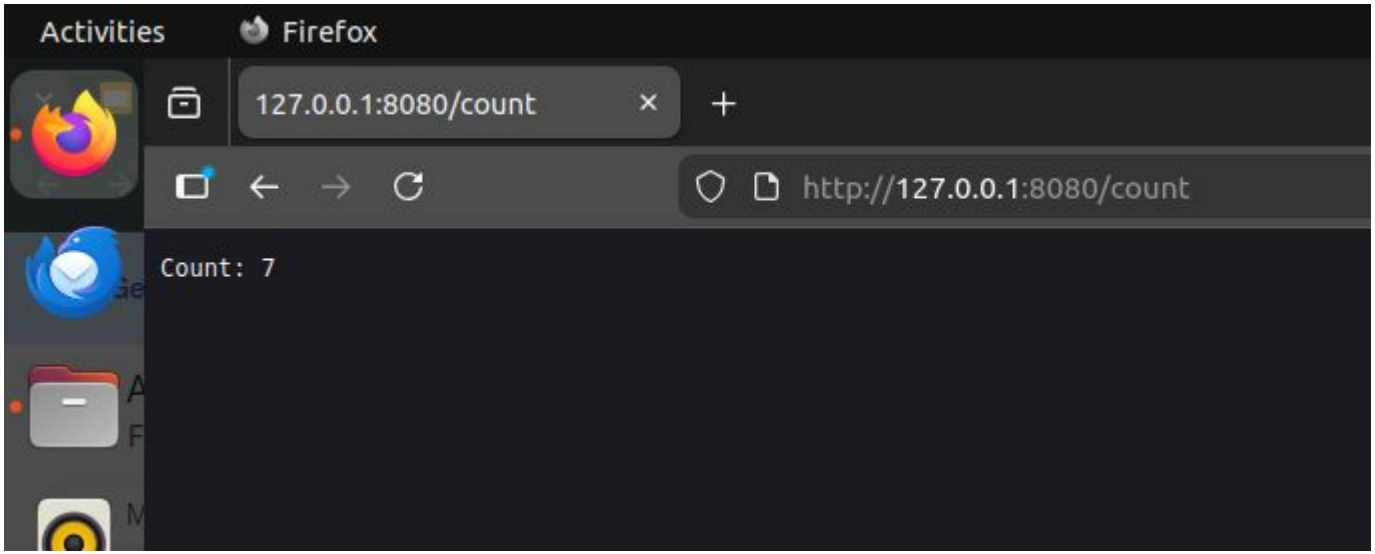
- When you'll visit the address http://127.0.0.1:8080/print?regex=e$ in your browser, you should be able to see the names of your classmates whose names end with A.



Assignment 1.4

Expected Output

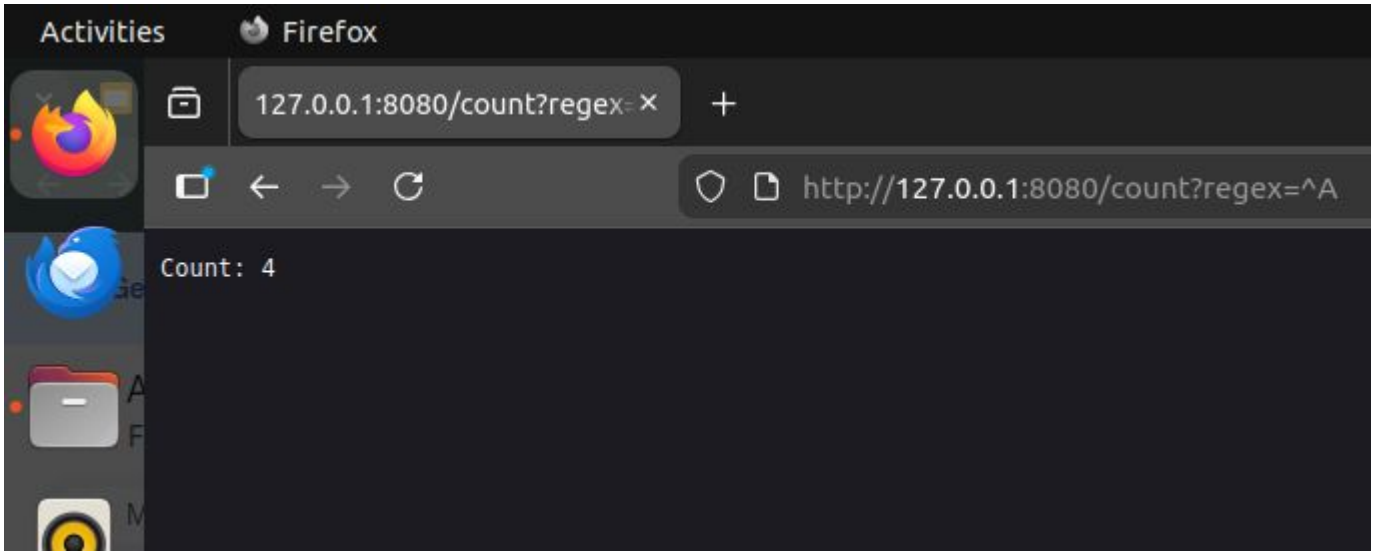
- When you'll visit the address http://127.0.0.1:8080/count in your browser, you should be able to see the total count of your classmates.



Assignment 1.4

Expected Output

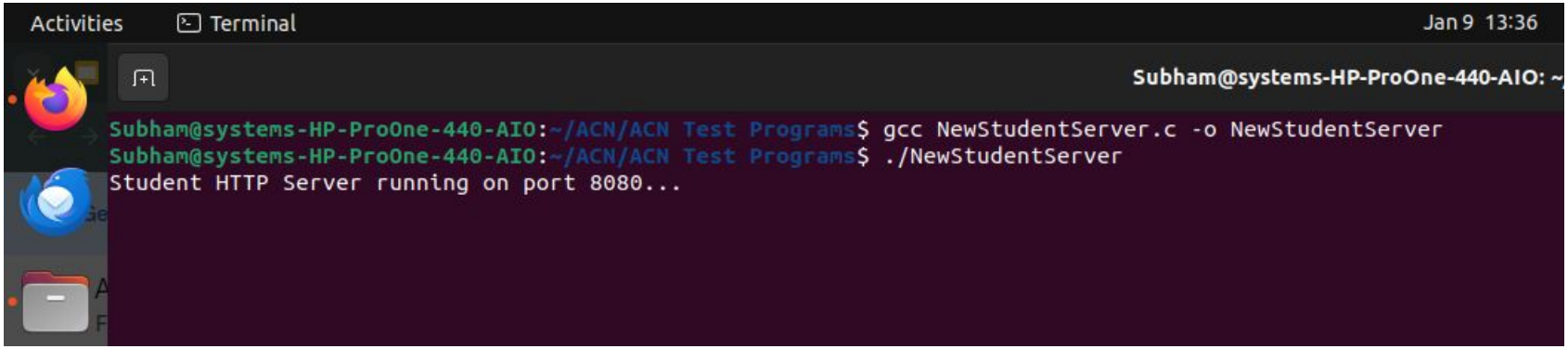
- When you'll visit the address http://127.0.0.1:8080/count?regex=^A in your browser, you should be able to see the total count of your classmates whose names start with A.



Assignment 1.4

Expected Output

- Meanwhile the server should be running on a terminal.



Assignment 1.4

Constraints:

- Do not use external HTTP libraries or frameworks
- No need to use HTTPS
- Use only:
  - Socket system calls
  - Standard C libraries
  - POSIX regular expressions
- You can use the header file `<regex.h>`

Assignment 1.5

100 Marks

Problem Statement

Design and implement an extension to the previously developed custom HTTP server to support a calculator service.

The server must accept arithmetic requests from a standard web browser and return the computed result as an HTTP response.

The calculator service must support basic arithmetic operations such as:

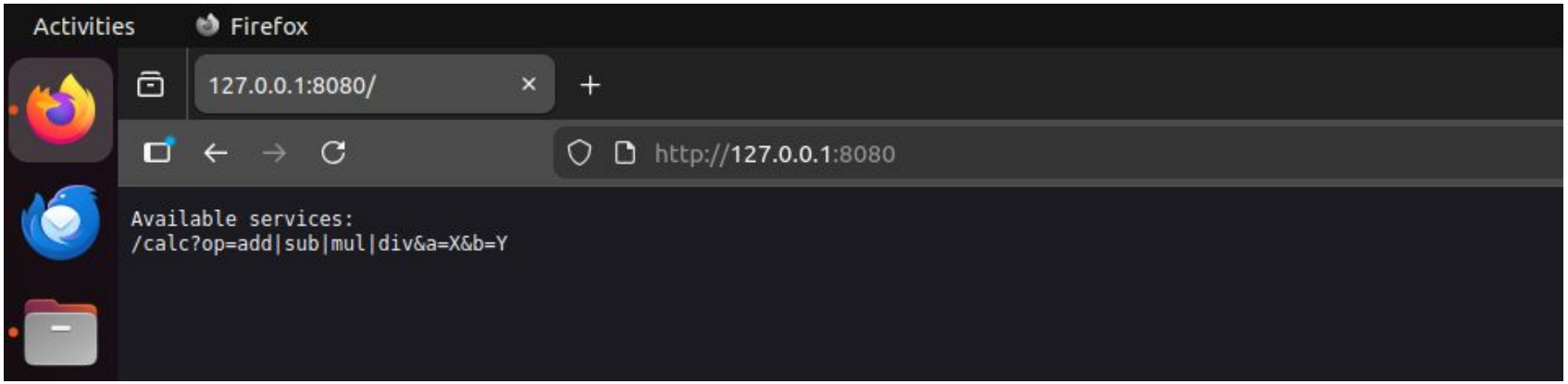
- Addition
- Subtraction
- Multiplication
- Division

The client must not be a custom program; only a browser should be used to interact with the service.

Assignment 1.5

Expected Output

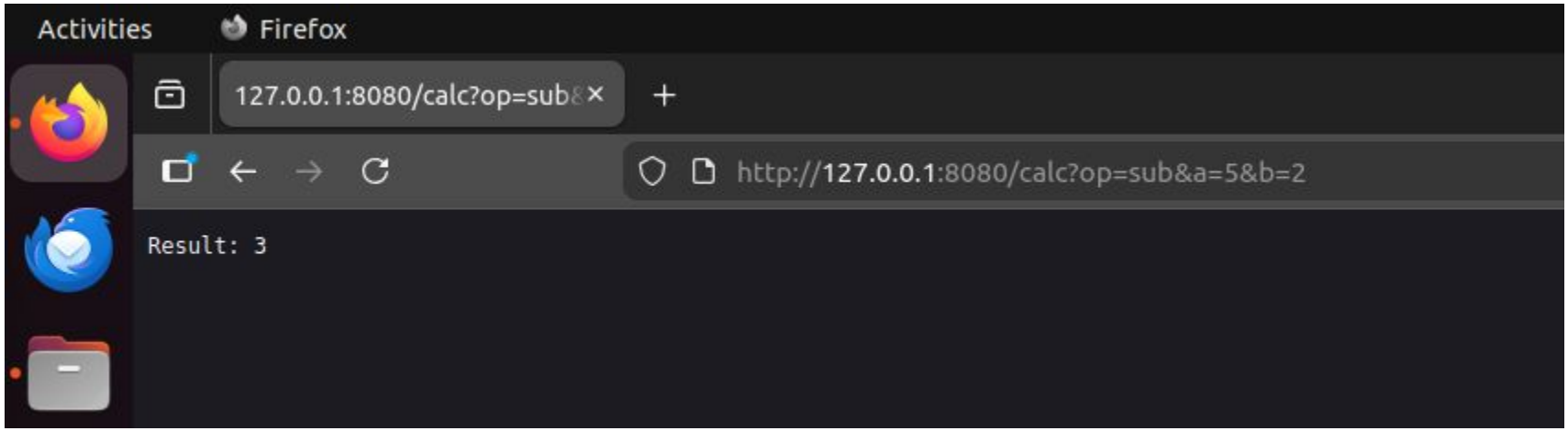
- When you'll visit the address http://127.0.0.1:8080/ in your browser, you should be able to see the list of all the functionalities (or any meaningful message).



Assignment 1.5

Expected Output

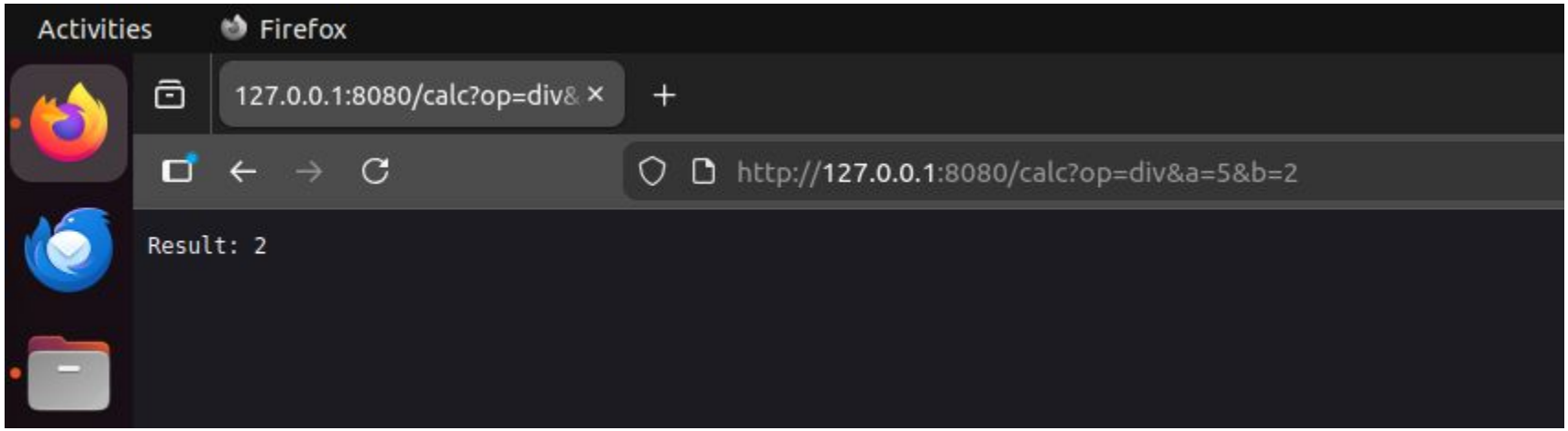
- When you'll visit the address http://127.0.0.1:8080/calc?op=sub&a=5&b=2 in your browser, you should be able to see the result of (5 - 2) = 3.



Assignment 1.5

Expected Output

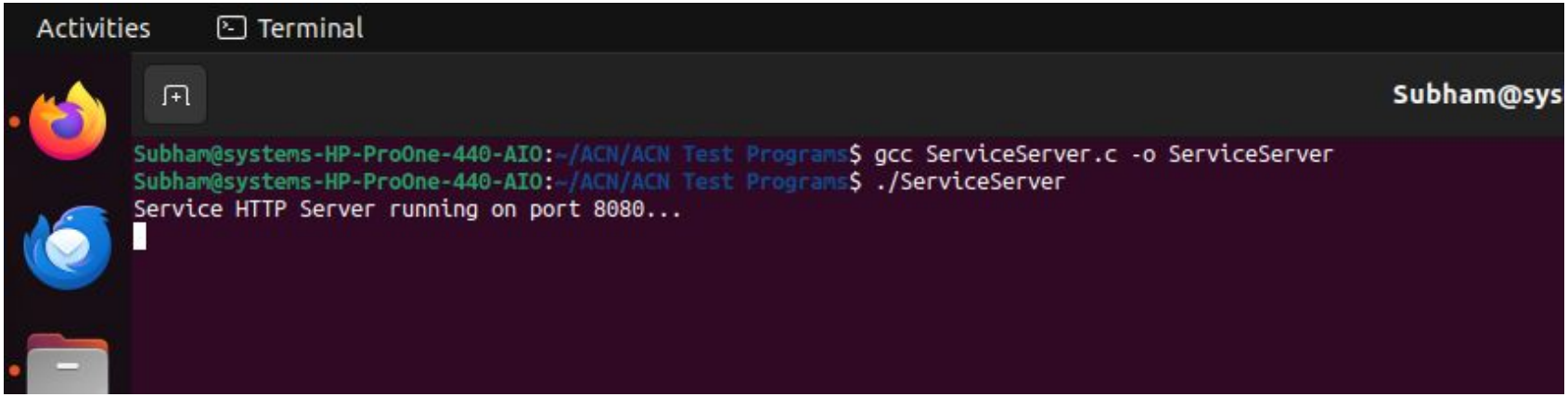
- When you'll visit the address http://127.0.0.1:8080/calc?op=div&a=5&b=2 in your browser, you should be able to see the result of (5 / 2) = 2.



Assignment 1.5

Expected Output

- Meanwhile the server should be running on a terminal.



Assignment 1.5

Constraints

- Use only:
  - TCP socket programming
  - Basic string manipulation
  - Standard C libraries
- Do not use external HTTP libraries or frameworks
- HTTPS is not required
- The server should handle one request at a time (concurrency is not needed)

Assignment 1.6

100 Marks

Problem Statement

To design and implement a WebSocket-based server in C that enables a web browser to communicate with the server without using HTTP request methods such as GET, POST, PUT, DELETE, etc., and to provide specialized student-related services using custom application-level commands.

Functional Requirements

1. Implement a WebSocket server in C that:

- Listens on a specified TCP port
- Correctly performs the WebSocket handshake
- Upgrades the connection from HTTP to WebSocket
- Communicates with the browser without using HTTP request methods after the upgrade

2. Maintain a hardcoded list of student names within the server.

Assignment 1.6

Problem Statement

To design and implement a WebSocket-based server in C that enables a web browser to communicate with the server without using HTTP request methods such as GET, POST, PUT, DELETE, etc., and to provide specialized student-related services using custom application-level commands.

Functional Requirements

3. Design and implement the following custom command-based services (sent as plain text messages over WebSocket):

- `COUNT`
  → Display the total number of students
- `COUNT <RegEx>`
  → Display the number of students whose names match the given regular expression
- `PRINT`
  → Display all student names
- `PRINT <RegEx>`
  → Display student names that match the given regular expression

Assignment 1.6

Problem Statement

To design and implement a WebSocket-based server in C that enables a web browser to communicate with the server without using HTTP request methods such as GET, POST, PUT, DELETE, etc., and to provide specialized student-related services using custom application-level commands.

Functional Requirements

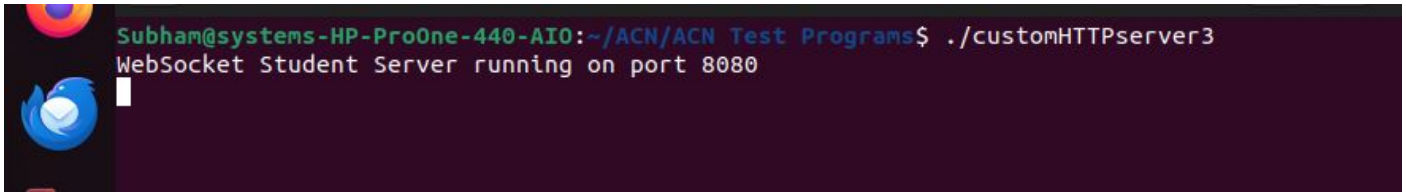
4. The server must interpret commands sent as WebSocket messages, not as HTTP requests.

5. The output of each command must be displayed in the web browser.

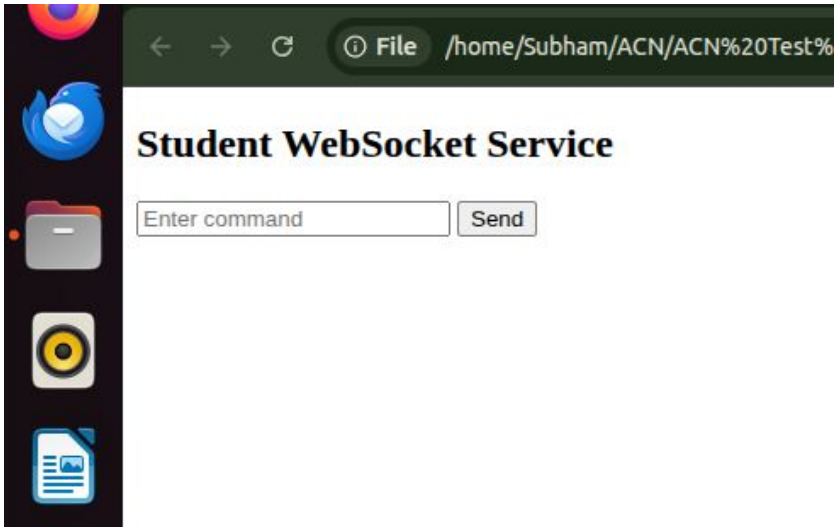
Assignment 1.6

Expected Output

- When server program is run on terminal…



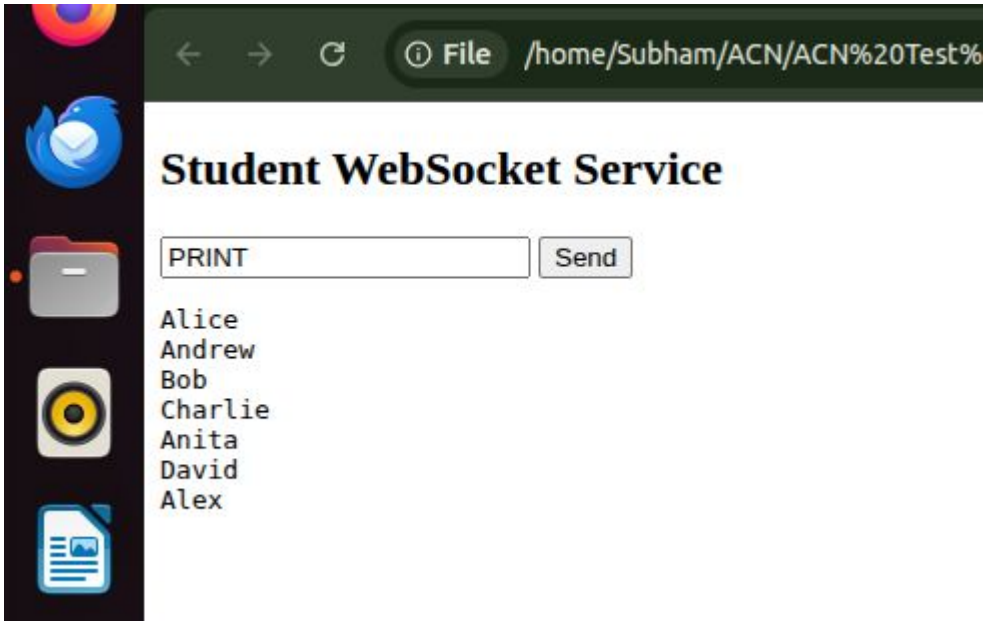
- When you open the HTML file…



Assignment 1.6

Expected Output

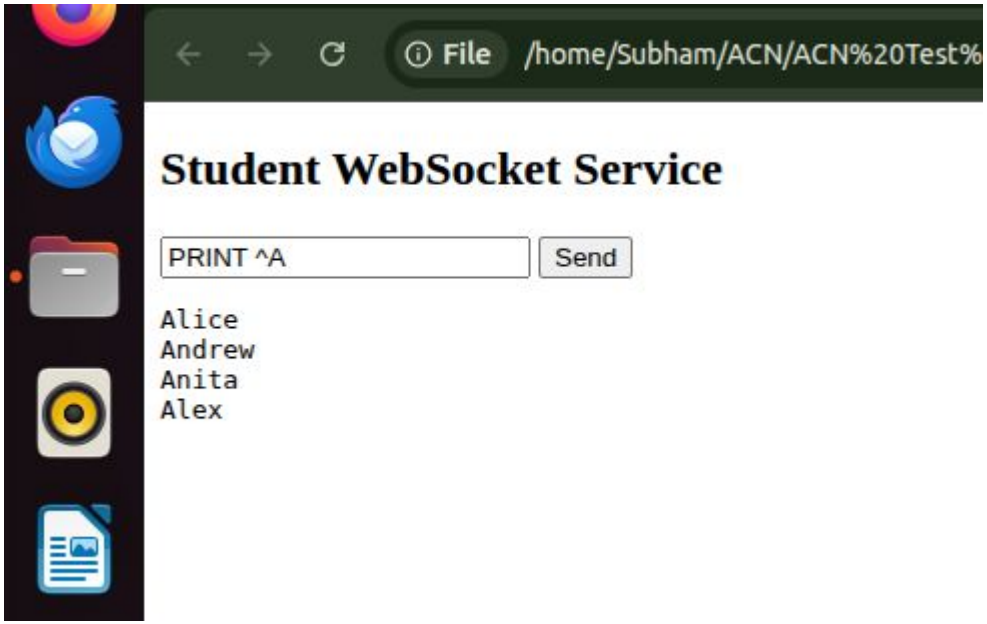
- When you type “PRINT” in the textbox and click send, you should get this output.



Assignment 1.6

Expected Output

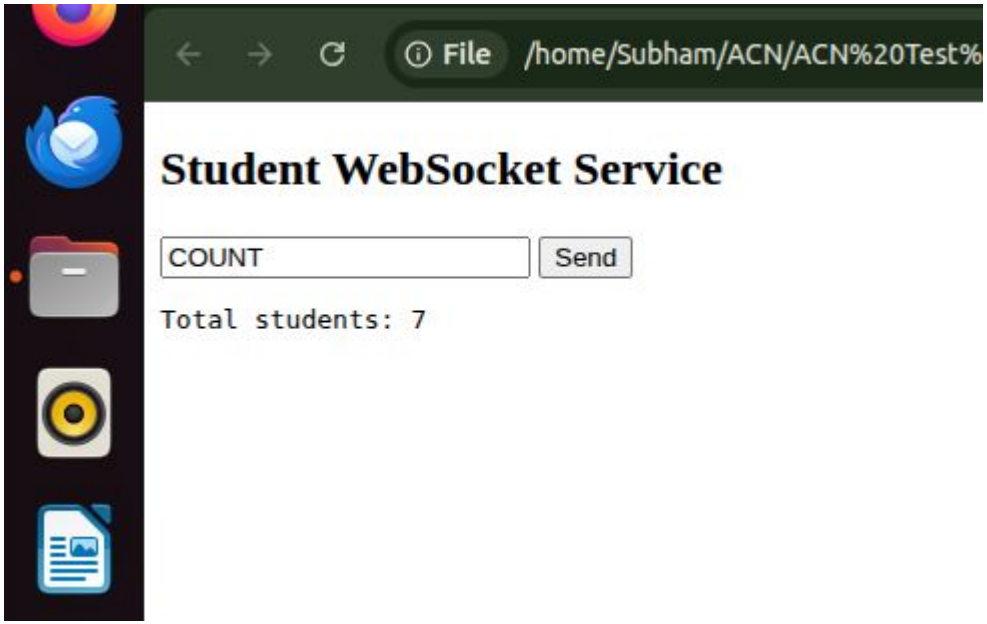
- When you type “PRINT ^A” in the textbox and click send, you should get this output.



Assignment 1.6

Expected Output

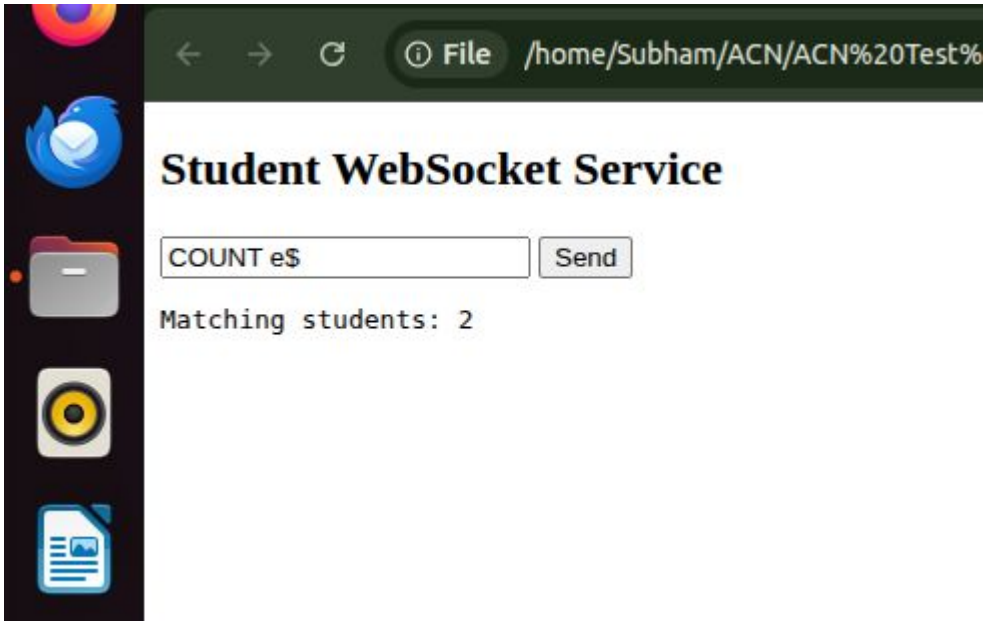
- When you type “COUNT in the textbox and click send, you should get this output.



Assignment 1.6

Expected Output

- When you type “COUNT e$” in the textbox and click send, you should get this output.



Assignment 1.6

Constraints

- HTTP methods such as GET, POST, PUT, DELETE, etc. must not be used for client–server communication after the WebSocket connection is established.
- All interaction must occur through WebSocket message frames.
- No REST APIs or form-based HTTP requests should be used.
- You can use the header file `<regex.h>`
- You can use OpenSSL library