



FACULTY OF ENGINEERING TECHNOLOGY  
ELECTRIC & COMPUTER ENGINEERING DEPARTMENT  
Computer Networks (ENCS336)  
Report Project: Socket Programming,

---

**Partners:** Wasfia Awwd

**Sec:** 2

**ID Number:** 1211039

**Instructor:** Ibrahim Nemer

**Date:** 10/8/2024

# Outline

---

- 1. Title Page**
  - Course and Project Details
  - Partner and Instructor Information
  - Submission Date
- 2. Abstract**
  - Overview of networking commands, socket programming, and web server implementation
- 3. Task 1: Network Commands & Wireshark**
  - Definitions and usage of ping, tracert, nslookup, telnet
  - Wireshark DNS analysis
- 4. Task 2: Socket Programming**
  - TCP Client-Server implementation
  - UDP Client-Server implementation
- 5. Task 3: Web Server**
  - Web server setup and testing
  - Handling HTTP requests and logging
- 6. Challenges**
  - Telnet command troubleshooting
  - Web server bind exception resolution
- 7. Conclusion**
  - Summary of tasks and skills gained
- 8. References**
  - Cited tools and sources

## Abstract

---

This project explains the application of core networking principles and socket programming techniques through a series of practical exercises. It begins with the exploration of essential networking commands such as ping, tracert, nslookup, and telnet, which are utilized to diagnose and understand network behavior. Wireshark is employed to capture and analyze DNS traffic, providing insights into packet-level communication. The project advances into the development of TCP and UDP client-server applications, where custom protocols are implemented to achieve specific tasks like message exchange and data manipulation. Finally, the project involves the creation of a simple yet functional web server using socket programming, capable of handling HTTP requests and serving web content. Throughout the project, I have gained experience in networking, enhanced my understanding of socket programming, and developed a working knowledge of web server implementation, all of which are crucial skills for a career in computer science and network engineering.

## Task1: Commands & Wireshark

---

In this task, I will explore some key network commands and use Wireshark to capture and analyze DNS messages. Before starting, I'll list the definitions of each concept to ensure a clear understanding before applying them practically.

### 1. Definitions of ping, tracert, nslookup, and telnet:

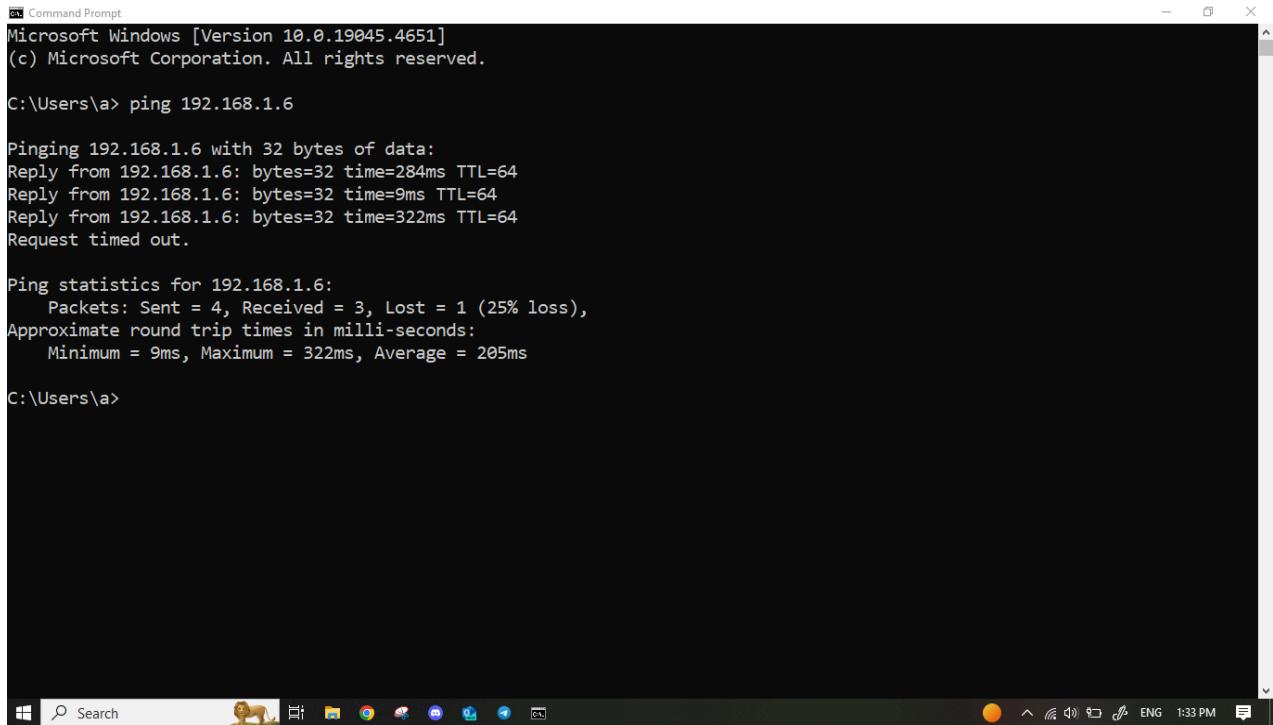
- a) **Ping** is the time it takes for a small data set to be transmitted from your device to a server on the Internet and back to your device again and is measured in milliseconds. For example, when playing an online game, a higher ping indicates that it takes longer for data packets to travel between your computer and the game server (or other players' computers). This increased delay can result in lag, leading to poor gameplay performance.
- b) **Tracert** is a network tool that tracks the path data takes from your computer to a destination, like a website. It shows each step (hop) the data takes through network devices (like routers) and how long each step takes. It's used to diagnose network problems and understand the route data travels across the internet.
- c) **nslookup** is a network tool used to find the IP address associated with a domain name (like "something.com") or vice versa. It queries DNS (Domain Name System) servers to get this information, helping you troubleshoot domain-related issues or understand where a domain is hosted.
- d) **Telnet** is a network protocol and tool used to connect to remote servers or devices over a network. It allows you to interact with the server through a command-line interface, often used for testing and managing services on remote systems.

After defining these concepts, I'll ensure my computer is connected to the internet and then process to run these commands. I'll start by pinging a device on my local

network (I chose my mobile phone) and a public website to observe the differences in response times and connectivity.

## 2. Running commands on cmd:

### a) Ping for device in the same network (Mobile phone):



```
Command Prompt
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\a> ping 192.168.1.6

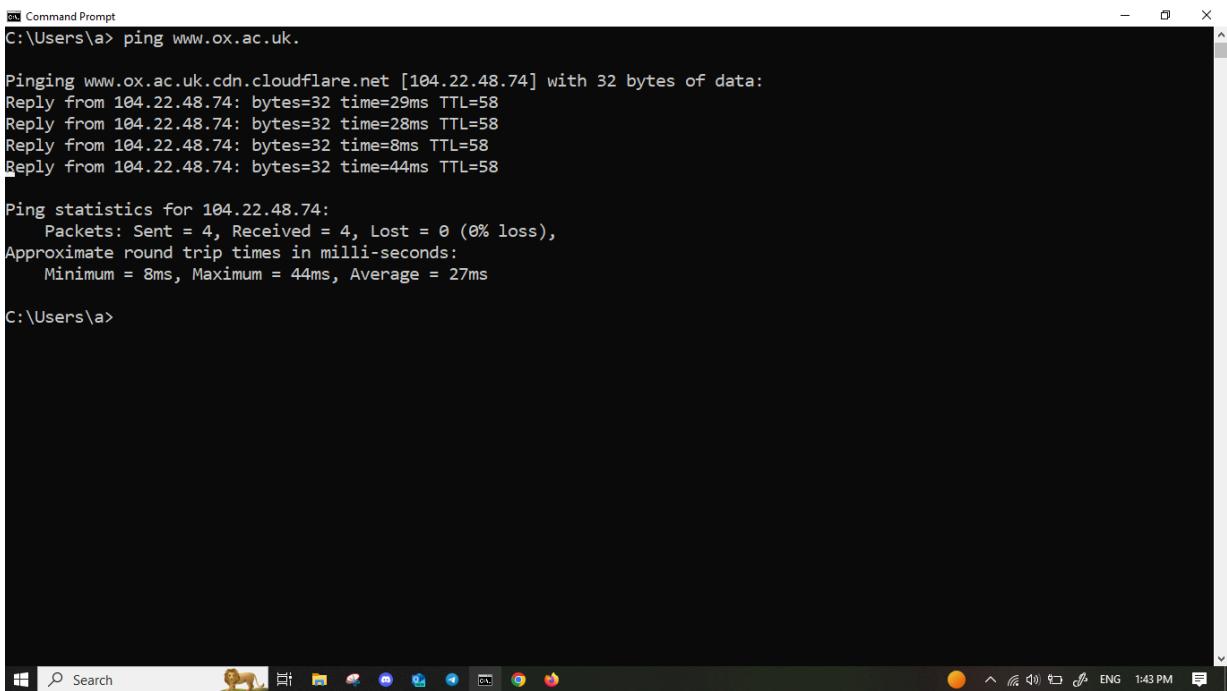
Pinging 192.168.1.6 with 32 bytes of data:
Reply from 192.168.1.6: bytes=32 time=284ms TTL=64
Reply from 192.168.1.6: bytes=32 time=9ms TTL=64
Reply from 192.168.1.6: bytes=32 time=322ms TTL=64
Request timed out.

Ping statistics for 192.168.1.6:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 322ms, Average = 205ms

C:\Users\a>
```

The screenshot shows the result of a ping command to the IP address of my phone 192.168.1.6. Out of four packets sent, three were successfully received, and one was lost, resulting in a 25% packet loss. The round trip times for the successful pings varied significantly, ranging from 9 ms to 322 ms, with an average of 205 ms. This indicates some instability or network issues affecting the connection to that IP address.

### b) Ping for the url [www.ox.ac.uk](http://www.ox.ac.uk) :



```
Command Prompt
C:\Users\a> ping www.ox.ac.uk.

Pinging www.ox.ac.uk.cdn.cloudflare.net [104.22.48.74] with 32 bytes of data:
Reply from 104.22.48.74: bytes=32 time=29ms TTL=58
Reply from 104.22.48.74: bytes=32 time=28ms TTL=58
Reply from 104.22.48.74: bytes=32 time=8ms TTL=58
Reply from 104.22.48.74: bytes=32 time=44ms TTL=58

Ping statistics for 104.22.48.74:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 44ms, Average = 27ms

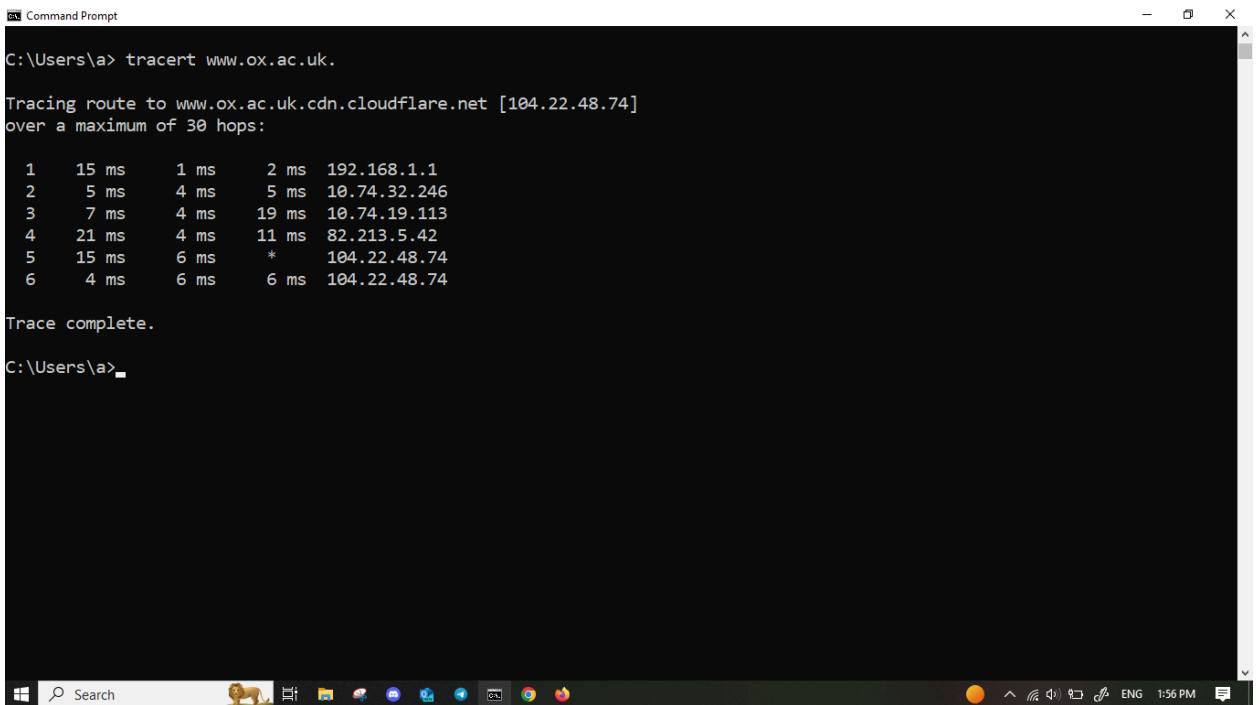
C:\Users\a>
```

The screenshot shows the result of a ping command to the domain [www.ox.ac.uk](http://www.ox.ac.uk), resolving to the IP address 104.22.48.74. All four packets sent were successfully received, resulting in 0% packet loss. The round trip times for these pings were consistent and low, with a minimum of 8 ms, a maximum of 44 ms, and an average of 27 ms. This indicates a stable and reliable connection to the domain.

- c) specifying the location of the server from where i got the response:

The IP address 104.22.48.74 is the identifier of the server that responded to my ping. The server is part of Cloudflare's Content Delivery Network (CDN), which means that the physical server I applied the ping command on could be in a data center near my location, since Cloudflare has data centers all over the world, so while the IP address belongs to Cloudflare, the actual physical server could be in a different country or region.

- d) Tracert for [www.ox.ac.uk](http://www.ox.ac.uk) :



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command entered is "tracert www.ox.ac.uk.". The output shows the tracing route to the domain, which resolved to the IP address 104.22.48.74 over a maximum of 30 hops. The trace completed in 6 hops, with times ranging from 1 ms to 21 ms. The first hop is to the local gateway (192.168.1.1), followed by several internal network hops before reaching external IP addresses, including one at 82.213.5.42 and the final destination at 104.22.48.74. There is one instance of a missing response (indicated by the asterisk). The taskbar at the bottom shows various icons and the system tray indicates the date and time as 1:56 PM.

```
C:\Users\...\a> tracert www.ox.ac.uk.

Tracing route to www.ox.ac.uk.cdn.cloudflare.net [104.22.48.74]
over a maximum of 30 hops:

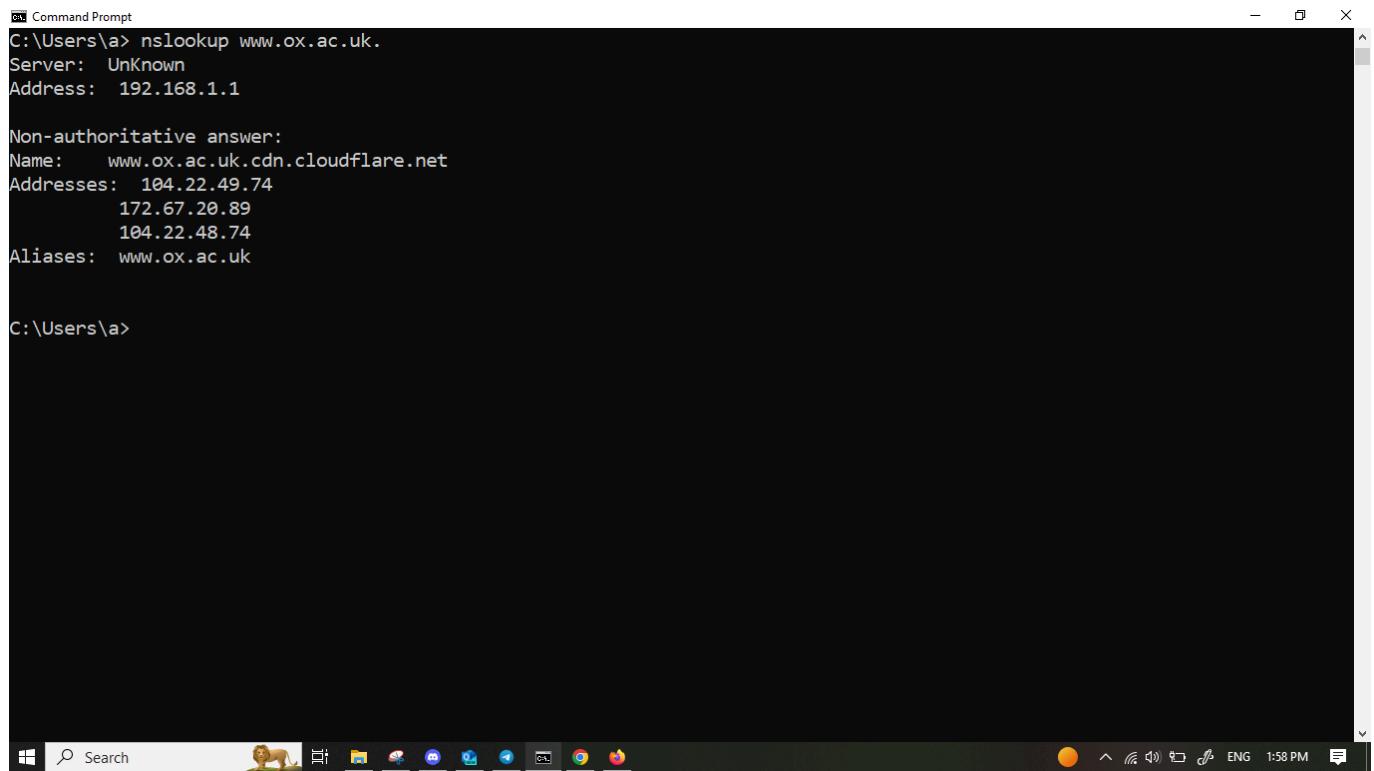
 1  15 ms    1 ms    2 ms  192.168.1.1
 2  5 ms     4 ms    5 ms  10.74.32.246
 3  7 ms     4 ms   19 ms  10.74.19.113
 4  21 ms    4 ms   11 ms  82.213.5.42
 5  15 ms    6 ms    *    104.22.48.74
 6  4 ms     6 ms    6 ms  104.22.48.74

Trace complete.

C:\Users\...\a>
```

The screenshot shows the result of a tracert (traceroute) command to the domain [www.ox.ac.uk](http://www.ox.ac.uk), resolving to the IP address 104.22.48.74. The trace completed in 6 hops, with the times for each hop ranging from 1 ms to 21 ms. The first hop is to the local gateway (192.168.1.1), followed by a few internal network hops before reaching external IP addresses, including one at 82.213.5.42 and the final destination at 104.22.48.74. There is one instance of a missing response (indicated by the asterisk), which is common and not necessarily indicative of a problem. Overall, the trace shows a relatively quick and direct route to the destination.

e) nslookup for [www.ox.ac.uk](http://www.ox.ac.uk) :



```
Command Prompt
C:\Users\a> nslookup www.ox.ac.uk.
Server: UnKnown
Address: 192.168.1.1

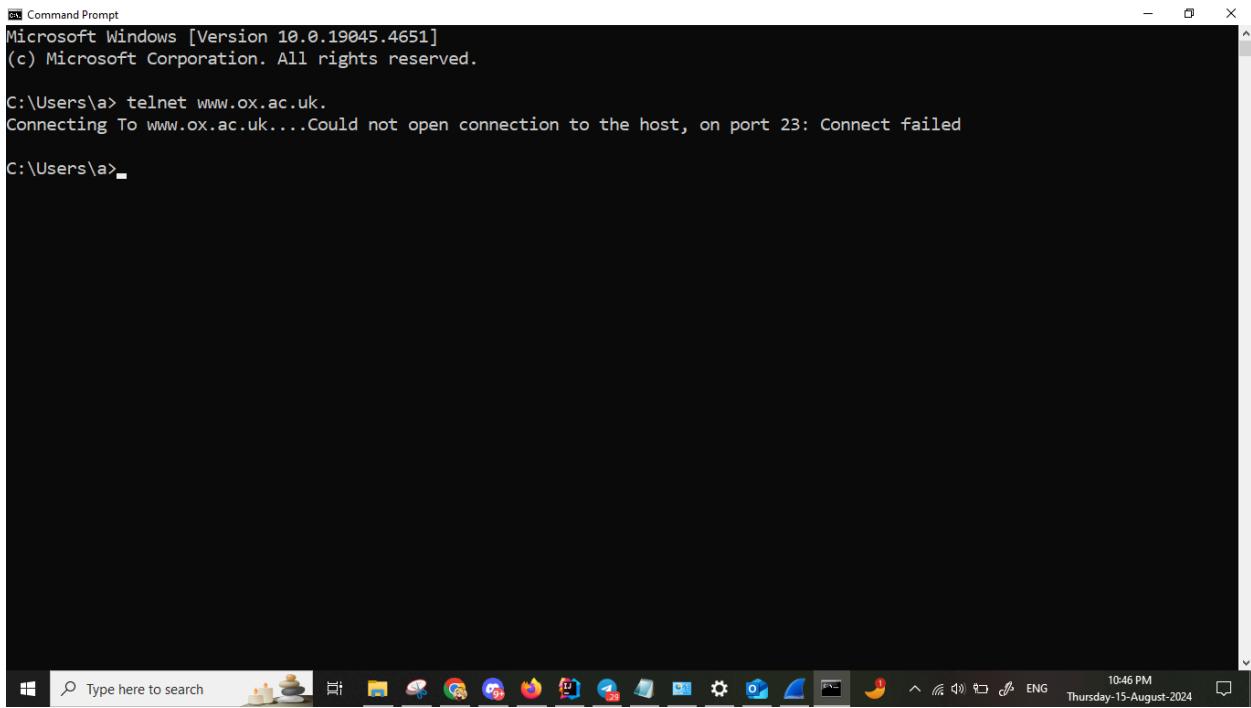
Non-authoritative answer:
Name: www.ox.ac.uk.cdn.cloudflare.net
Addresses: 104.22.49.74
          172.67.20.89
          104.22.48.74
Aliases: www.ox.ac.uk

C:\Users\a>
```

The screenshot shows a Windows Command Prompt window titled "Command Prompt". The user has run the command "nslookup www.ox.ac.uk.". The output indicates that the server is "UnKnown" and the local address is "192.168.1.1". The non-authoritative answer shows that the domain "www.ox.ac.uk" is associated with the IP addresses 104.22.49.74, 172.67.20.89, and 104.22.48.74, and has an alias "www.ox.ac.uk". The taskbar at the bottom of the screen shows various icons for common applications like File Explorer, Task View, and the Start button.

The screenshot shows the result of promoting an nslookup query for the domain "[www.ox.ac.uk](http://www.ox.ac.uk)." The results indicate that the DNS lookup returned IP addresses associated with Cloudflare's content delivery network (CDN), suggesting that the Oxford University website is utilizing Cloudflare for DNS services and possibly for enhanced security and performance. The local DNS server's address is shown as "192.168.1.1," but it is marked as "Unknown," which may indicate a local network configuration detail.

f) telnet [www.ox.ac.uk](http://www.ox.ac.uk).



```
Command Prompt
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aa> telnet www.ox.ac.uk.
Connecting To www.ox.ac.uk....Could not open connection to the host, on port 23: Connect failed

C:\Users\aa>
```

Connection to port 23 failed, this problem is mentioned in the [Alternatives Solutions, Issues and Limitations](#) part of the report.

### 3. Some details about autonomous system (AS) number, number of IPs, prefixes, peers, name of Tier1-ISP of [www.ox.ac.uk](#). :

#### [ox.ac.uk](#)

[DNS Info](#) [Website Info](#) [IP Info](#)

---

[151.101.2.216](#) > [151.101.0.0/22](#) > [AS54113](#) > Fastly, Inc.  
[151.101.2.216](#) > [151.101.0.0/16](#) > [AS54113](#) > Fastly, Inc.  
[151.101.130.216](#) > [151.101.128.0/22](#) > [AS54113](#) > Fastly, Inc.  
[151.101.130.216](#) > [151.101.0.0/16](#) > [AS54113](#) > Fastly, Inc.  
[151.101.66.216](#) > [151.101.64.0/22](#) > [AS54113](#) > Fastly, Inc.  
[151.101.66.216](#) > [151.101.0.0/16](#) > [AS54113](#) > Fastly, Inc.  
[151.101.194.216](#) > [151.101.192.0/22](#) > [AS54113](#) > Fastly, Inc.  
[151.101.194.216](#) > [151.101.0.0/16](#) > [AS54113](#) > Fastly, Inc.

This screenshot maps specific IP addresses to the larger IP blocks and the autonomous system (Fastly's AS54113) that manages them. This is crucial for understanding how internet traffic is routed through Fastly's network.

Breakdown of the Information:

**IP Addresses:**

- **151.101.2.216**
- **151.101.130.216**
- **151.101.66.216**
- **151.101.194.216**

These are individual IP addresses that belong to the same larger IP block managed by the same organization, which is Fastly, Inc.

**IP Prefixes:**

- **151.101.0.0/22**
- **151.101.0.0/16**
- **151.101.128.0/22**
- **151.101.64.0/22**
- **151.101.192.0/22**

These represent the IP address ranges (prefixes) that include the specific IP addresses listed. The /22 and /16 are CIDR (Classless Inter-Domain Routing) notations that define the size of the IP address block. A /22 prefix contains 1,024 IP addresses, while a /16 prefix contains 65,536 IP addresses.

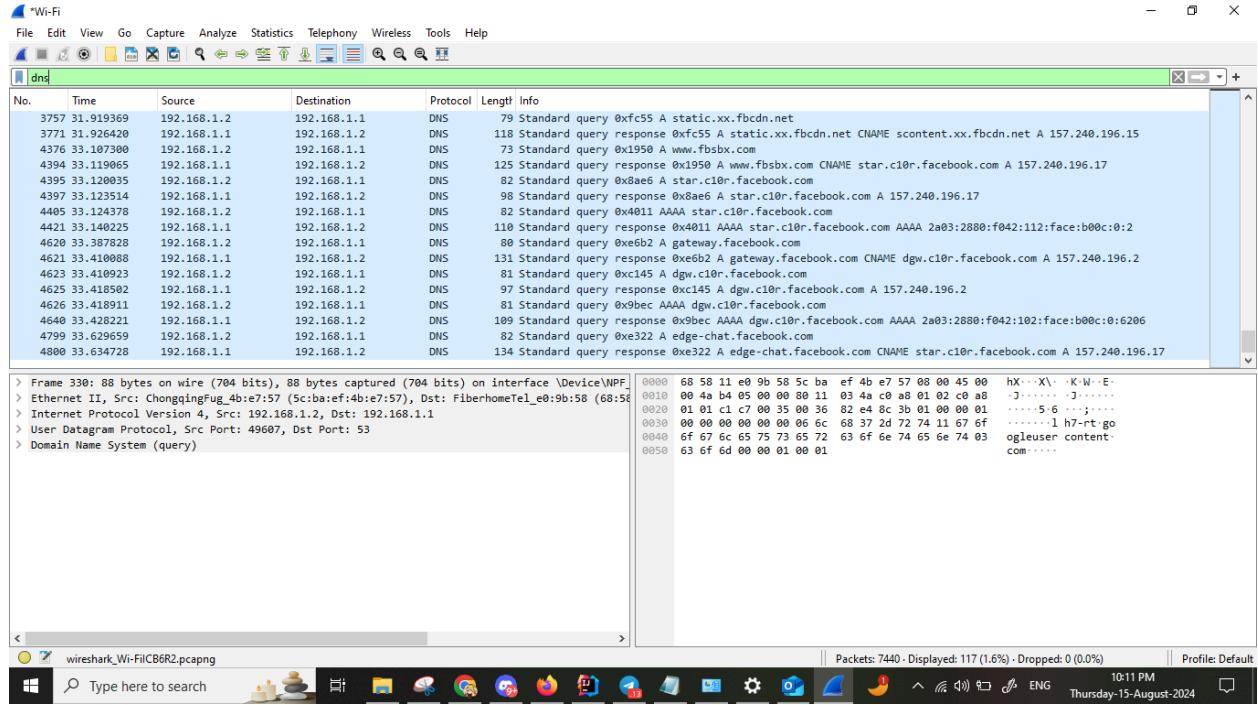
**Autonomous System (AS) Number:**

- **AS54113**

This is the Autonomous System Number associated with these IP address ranges. AS54113 is assigned to Fastly, Inc.

Oxford is not a Tier 1 ISP, meaning it does not have settlement-free peering relationships with every other network on the Internet. Instead, it relies on JANET and other Tier 2 ISPs for transit services.

#### 4. Capturing some DNS messages from Wireshark:



The screenshot shows a Wireshark capture focused on DNS traffic between a local machine (192.168.1.2) and a router (192.168.1.1). The DNS queries are related to Facebook domains, such as static.xx.fbcdn.net, star.c10r.facebook.com, and gateway.facebook.com, with responses indicating successful resolution to both IPv4 and IPv6 addresses. The capture highlights multiple standard DNS queries and their corresponding responses, illustrating typical DNS resolution for accessing Facebook services.

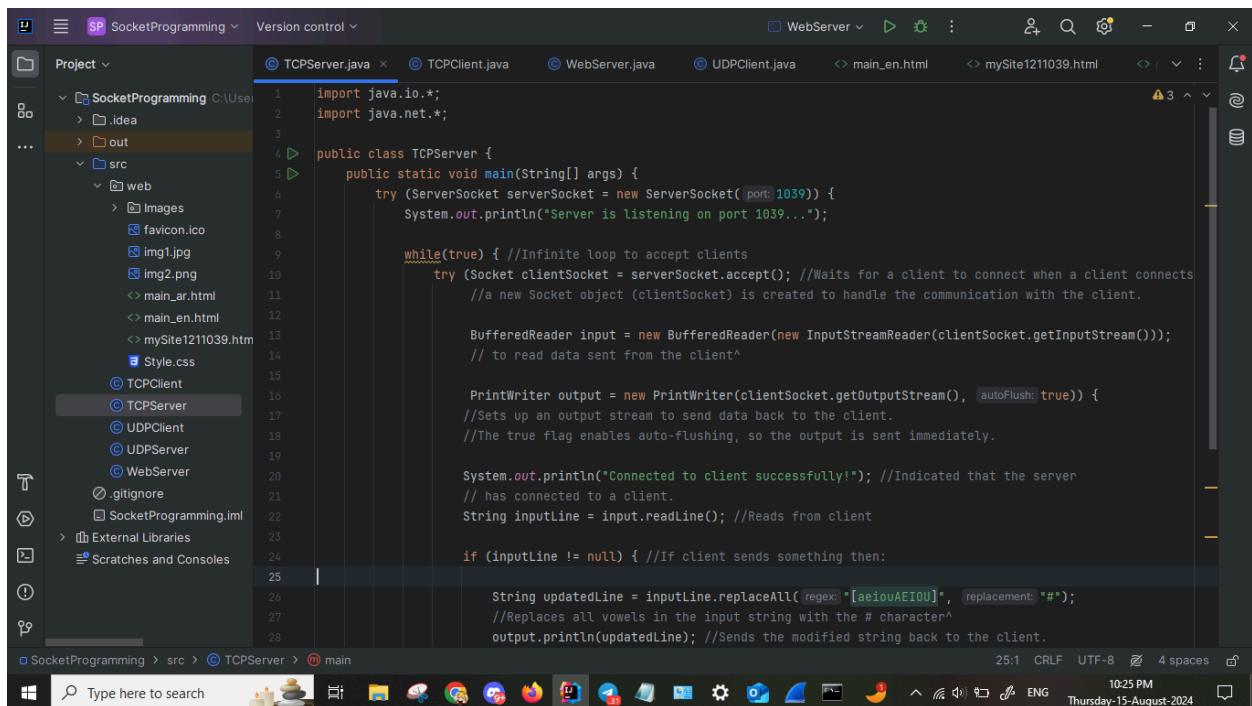
## Task2: Socket Programming (TCP and UDP)

In this task, I implemented socket programming for both TCP and UDP communications. For the TCP part, I developed a client-server application in Java where the client sends a string to the server. The server processes the string by replacing all vowels with '#' and returns the modified string to the client for display. The communication uses a port number 1039 derived from my ID. For the UDP part, I created client and server applications where the server listens on the same port. Clients and the server exchange messages, with the server displaying messages received from all peers. Each client only shows messages exchanged with the server. This setup allows for testing and demonstrating peer-to-peer communication and message handling.

### TCP part

#### Codes:

- TCP Server code:



The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The left sidebar displays the project structure with files like TCPClient.java, WebServer.java, UDPClient.java, main\_en.html, and mySite1211039.html. The right pane shows the code editor for TCPServer.java. The code implements a TCP server that listens on port 1039. It reads input from a client, replaces vowels with '#', and sends the modified string back. The code uses BufferedReader for reading and PrintWriter for writing.

```
import java.io.*;
import java.net.*;

public class TCPServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(port: 1039)) {
            System.out.println("Server is listening on port 1039...");

            while(true) { //Infinite loop to accept clients
                try (Socket clientSocket = serverSocket.accept()); //Waits for a client to connect when a client connects
                //a new Socket object (clientSocket) is created to handle the communication with the client.

                BufferedReader input = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
                // to read data sent from the client^

                PrintWriter output = new PrintWriter(clientSocket.getOutputStream(), autoFlush: true));
                //Sets up an output stream to send data back to the client.
                //The true flag enables auto-flushing, so the output is sent immediately.

                System.out.println("Connected to client successfully!"); //Indicated that the server
                // has connected to a client.
                String inputLine = input.readLine(); //Reads from client

                if (inputLine != null) { //If client sends something then:

                    String updatedLine = inputLine.replaceAll(regex: "[aeiouAEIOU]", replacement: "#");
                    //Replaces all vowels in the input string with the # character^
                    output.println(updatedLine); //Sends the modified string back to the client.
                }
            }
        }
    }
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The left sidebar displays the project structure, including a 'src' folder containing 'TCPClient.java', 'TCPServer.java', and 'WebServer.java'. The right pane shows the content of 'TCPServer.java'. The code implements a TCP server that reads input from a client, replaces vowels with '#', and sends the modified string back. It also handles exceptions and includes imports for BufferedReader, PrintWriter, and various Java net classes.

```
BufferedReader input = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
// to read data sent from the client

PrintWriter output = new PrintWriter(clientSocket.getOutputStream(), true);
// Sets up an output stream to send data back to the client.
// The true flag enables auto-flushing, so the output is sent immediately.

System.out.println("Connected to client successfully!"); // Indicated that the server
// has connected to a client.
String inputLine = input.readLine(); // Reads from client

if (inputLine != null) { // If client sends something then:

    String updatedLine = inputLine.replaceAll("[aeiouAEIOU]", "#");
    // Replaces all vowels in the input string with the # character
    output.println(updatedLine); // Sends the modified string back to the client.
    System.out.println(updatedLine);

} catch (IOException e) {
    e.printStackTrace();
}

} catch (IOException e) {
    e.printStackTrace();
}
}
```

## ● TCP Client code:

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The left sidebar displays the project structure, including a 'src' folder containing 'TCPClient.java', 'TCPServer.java', and 'UDPClient.java'. The right pane shows the content of 'UDPClient.java'. The code implements a UDP client that sends messages to a server at port 1039 and receives responses. It uses Scanner to read input from the user and DatagramPacket to handle network communication.

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class UDPClient {
    public static void main(String[] args) {
        final int serverPort = 1039; // Fixed port where server is listening
        Scanner scanner = new Scanner(System.in);

        try {
            InetAddress serverAddress = InetAddress.getByName("localhost");
            DatagramSocket clientSocket = new DatagramSocket();
            int clientNum = 1;

            while (true) {
                // Input message to send
                System.out.print("Enter your message client" + clientNum + ": ");
                String message = scanner.nextLine(); // Scan the message
                String fullMessage = "Client" + clientNum + ":" + message; // Get the full message

                // Send the message to the server
                byte[] buffer = fullMessage.getBytes(); // Place the message in the buffer
                DatagramPacket sendPacket = new DatagramPacket(buffer, buffer.length, serverAddress, serverPort);
                clientSocket.send(sendPacket); // Send the packet to the server

                // Receive the response from the server
                DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
                clientSocket.receive(receivePacket);
                String response = new String(receivePacket.getData());
                System.out.println("Response from server: " + response);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'src' directory contains files: TCPClient.java, WebServer.java, UDPClient.java (selected), main\_en.html, and mySite1211039.htm. The UDPClient.java code is displayed in the editor:

```
DatagramSocket clientSocket = new DatagramSocket();
int clientNum = 1;

while (true) {
    // Input message to send
    System.out.print("Enter your message client" + clientNum + ": ");
    String message = scanner.nextLine(); // Scan the message
    String fullMessage = "Client" + clientNum + ":" + message; // Get the full message

    // Send the message to the server
    byte[] buffer = fullMessage.getBytes(); // Place the message in the buffer
    DatagramPacket sendPacket = new DatagramPacket(buffer, buffer.length, serverAddress, serverPort);
    clientSocket.send(sendPacket); // Send the packet to the server

    // Receive the response from the server
    DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
    clientSocket.receive(receivePacket);
    String responseMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());
    System.out.println("Server Response: " + responseMessage);

    clientNum++;
} catch (Exception e) {
    e.printStackTrace();
}
```

The status bar at the bottom shows: 23:37 CRLF UTF-8 4 spaces. The taskbar includes icons for File Explorer, Task View, Start, Taskbar settings, and various pinned applications.

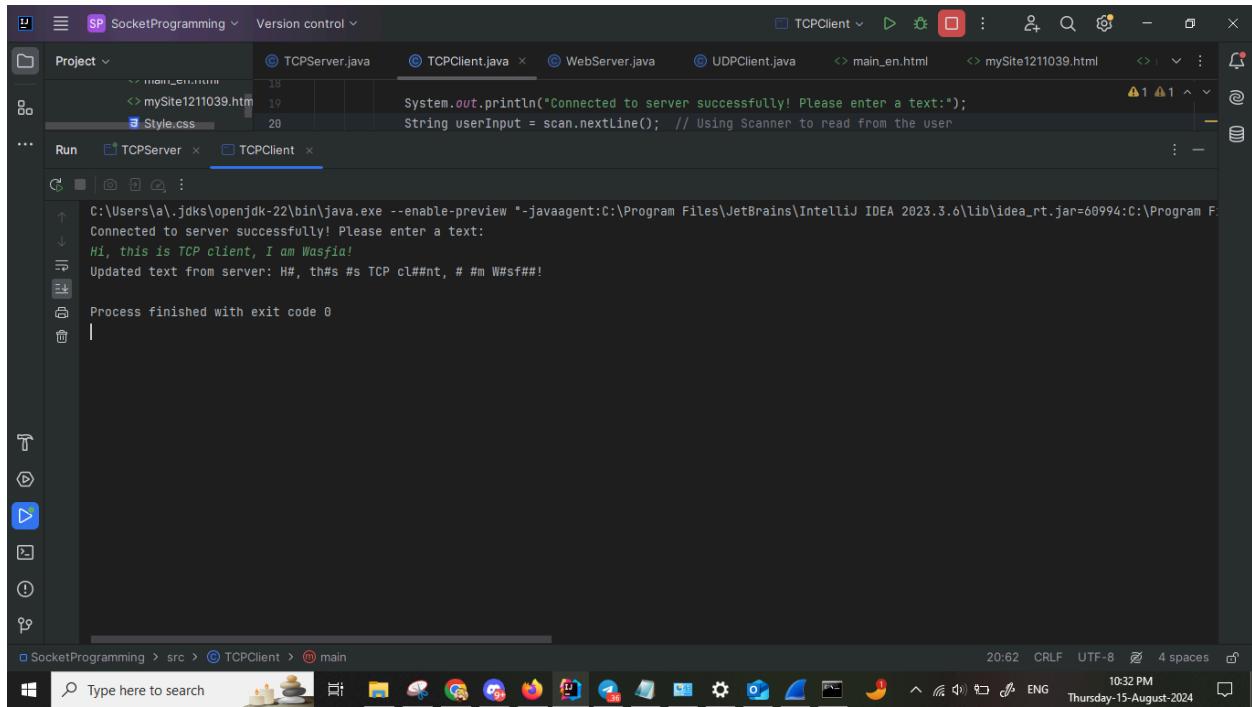
## Runs:

Beginning with running the TCP server:

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'Run' tab is selected, showing the 'WebServer' configuration. The output window displays the message: 'C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea\_rt.jar=60931:C:\Program F. Server is listening on port 1039...'. The status bar at the bottom shows: 25:1 CRLF UTF-8 4 spaces. The taskbar includes icons for File Explorer, Task View, Start, Taskbar settings, and various pinned applications.

This message on the terminal indicates that the server has successfully started and is now listening for incoming client connections on port 1039.

Secondly, we run the TCPClient:



The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'TCPClient.java' file is the active editor, displaying Java code for a TCP client. The code includes importing java.io.\* and java.net.\* packages, defining a class named TCPClient, and implementing a main method that prints a connection message and reads user input from the console. The 'Run' tool bar at the top has 'TCPServer' selected. The bottom status bar shows the path 'C:\Users\al\jdks\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea\_rt.jar=60994:C:\Program' and the time '20:02'. The terminal window shows the client connecting to the server and sending a message, which is then processed by the server and returned to the client.

```
System.out.println("Connected to server successfully! Please enter a text:");
String userInput = scan.nextLine(); // Using Scanner to read from the user
```

```
C:\Users\al\jdks\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea_rt.jar=60994:C:\Program F
Connected to server successfully! Please enter a text:
Hi, this is TCP client, I am Wasfia!
Updated text from server: H#, th#s #s TCP cl#nt, # #m W#sf#!
Process finished with exit code 0
```

This screenshot shows what happens after running the TCP client, we can see that the client successfully connects to the server, and the server prompts the user to enter text, displaying .The client inputs "Hi, this is TCP client, I am Wasfia!" which is sent to the server. The server receives this input, processes it by replacing all vowels with the “#” character, and sends back the modified string. The TCPClient program then prints the received modified string and the program terminates with "Process finished with exit code 0," indicating successful execution and communication between the client and server.

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The code editor displays `TCPClient.java` with the following content:

```
System.out.println("Connected to server successfully! Please enter a text:");
String userInput = scan.nextLine(); // Using Scanner to read from the user
```

The run output window shows the following logs:

```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=60991:C:\Program F
Server is listening on port 1039...
Connected to client successfully!
H#, th# s TCP cL##nt, # ## W#sf##!
```

And this screenshot from the TCPServer shows the modified message.

## UDP Part

Codes:

- **UDPServer code**

The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The code editor displays `UDPServer.java` with the following content:

```
public class UDPServer {
    public static void main(String[] args) {
        final int port = 1039; // Fixed port for server

        try (DatagramSocket serverSocket = new DatagramSocket(port)) {
            byte[] buffer = new byte[1024];
            System.out.println("Server is listening on port: " + port);

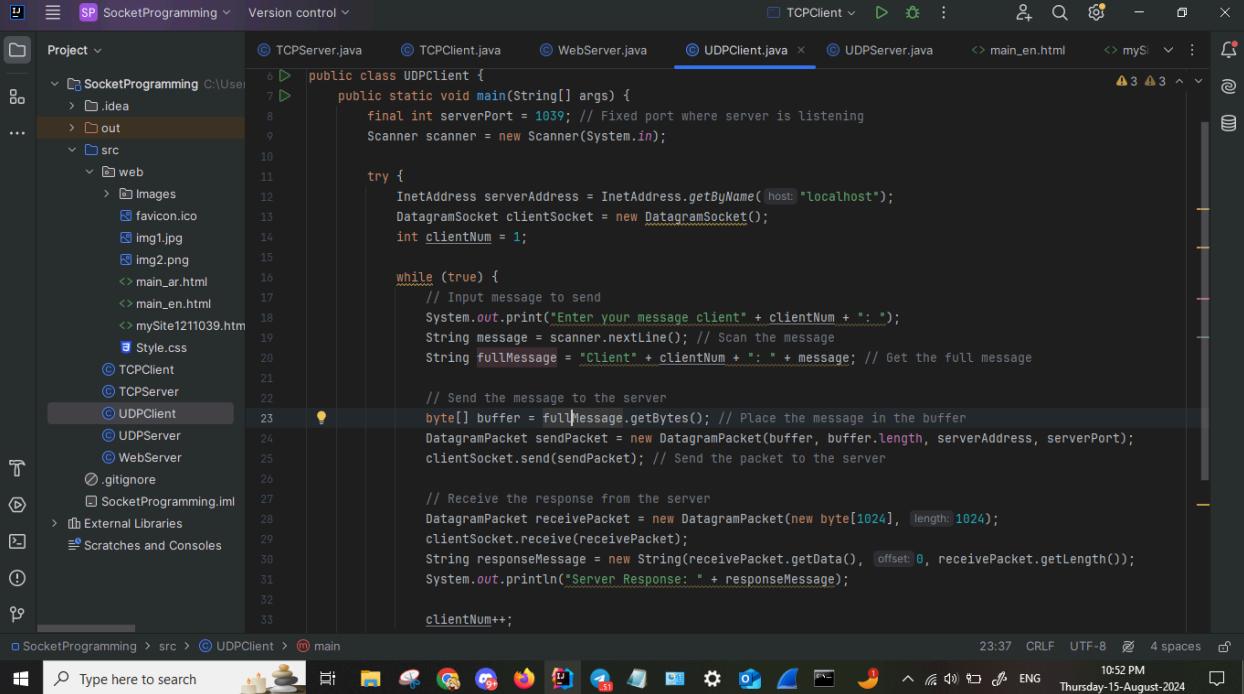
            while (true) {
                // Receive a packet
                DatagramPacket receivedPacket = new DatagramPacket(buffer, buffer.length);
                serverSocket.receive(receivedPacket);
                String receivedMessage = new String(receivedPacket.getData(), offset: 0, receivedPacket.getLength());
                System.out.println("Message from " + receivedMessage);

                // Get input from the console to send back to the client
                Scanner scan = new Scanner(System.in);
                System.out.print("Enter your message to send to client: ");
                String fullMessage = scan.nextLine();

                // Prepare the message to be sent back to the client
                byte[] bufferServer = fullMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(bufferServer, bufferServer.length,
                    receivedPacket.getAddress(), receivedPacket.getPort());

                // Send the packet to the client
                serverSocket.send(sendPacket);
            }
        }
    }
}
```

- UDPClient Code:



The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'src' directory contains several Java files: TCPClient.java, UDPClient.java (which is currently selected), UDPServer.java, and WebServer.java. The UDPClient.java code is displayed in the editor:

```

public class UDPClient {
    public static void main(String[] args) {
        final int serverPort = 1039; // Fixed port where server is listening
        Scanner scanner = new Scanner(System.in);

        try {
            InetAddress serverAddress = InetAddress.getByName("localhost");
            DatagramSocket clientSocket = new DatagramSocket();
            int clientNum = 1;

            while (true) {
                // Input message to send
                System.out.print("Enter your message client" + clientNum + ":" );
                String message = scanner.nextLine(); // Scan the message
                String fullMessage = "Client" + clientNum + ":" + message; // Get the full message

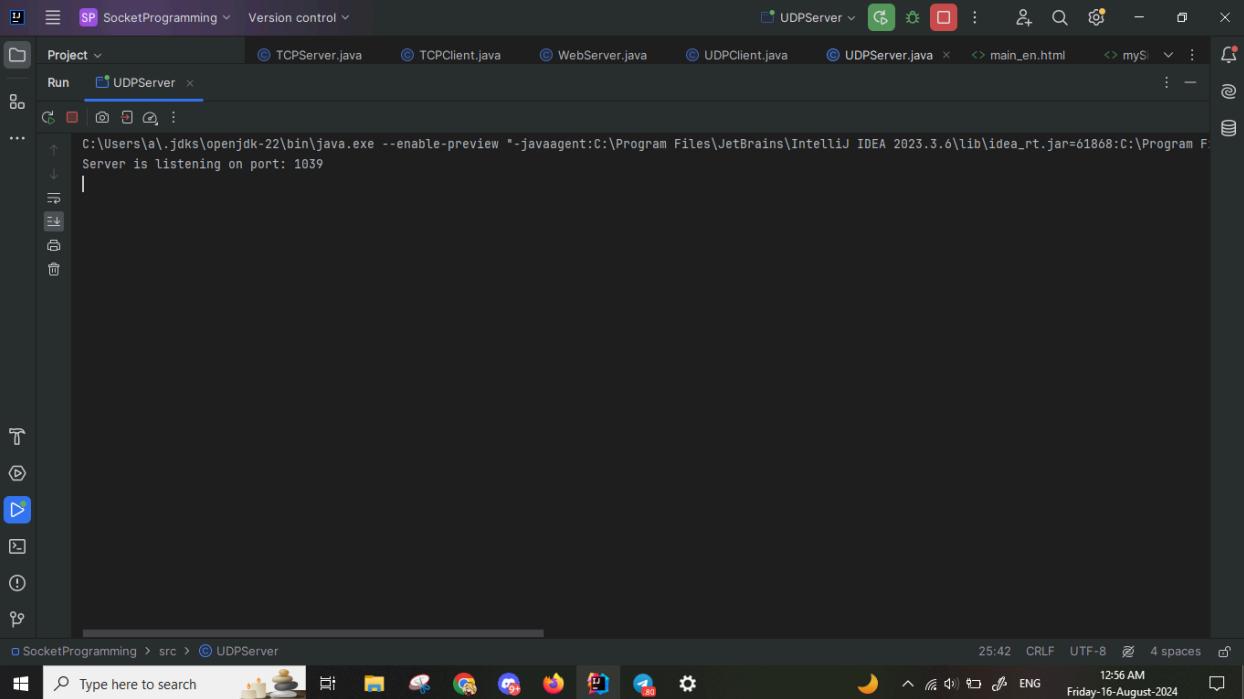
                // Send the message to the server
                byte[] buffer = fullMessage.getBytes(); // Place the message in the buffer
                DatagramPacket sendPacket = new DatagramPacket(buffer, buffer.length, serverAddress, serverPort);
                clientSocket.send(sendPacket); // Send the packet to the server

                // Receive the response from the server
                DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
                clientSocket.receive(receivePacket);
                String responseMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());
                System.out.println("Server Response: " + responseMessage);

                clientNum++;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

## Runs:



The screenshot shows the IntelliJ IDEA interface with the 'Run' tool window open. The 'Run' tab is selected, and the 'UDPClient' run configuration is active. The output pane displays the following text:

```

C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea_rt.jar=61868:C:\Program F
Server is listening on port: 1039

```

This screenshot shows the result of running UDPServer. This message on the terminal indicates that the server has successfully started and is now listening for incoming client connections on port 1039.

The screenshot displays the IntelliJ IDEA interface. The left sidebar shows a project structure with files like main\_ar.html, main\_en.html, mySite1211039.htm, Style.css, TCPClient, TCPServer, UDPClient (selected), UDPServer, WebServer, .gitignore, and SocketProgramming.iml. The main editor window shows the content of UDPClient.java. The terminal window at the bottom shows the output of the Java application, where a client has typed "Hi from client 1".

```
24 // Prepare the message to be sent back to the client
25 byte[] bufferServer = fullMessage.getBytes();
26 DatagramPacket sendPacket = new DatagramPacket(bufferServer, bufferServer.length,
27         receivedPacket.getAddress(), receivedPacket.getPort());
28
29         // Send the packet to the client
30         serverSocket.send(sendPacket);
31     } catch (Exception e) {
32         e.printStackTrace();
33     }
34 }
```

C:\Users\...\jdks\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea\_rt.jar=61875:C:\Program F  
Enter your message client1: Hi from client 1

This screenshot shows the result of running UDPClient. Client number 1 was successfully connected to the server and ready to type a message, and typed “Hi from client 1” as shown in the screenshot.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project:** SP SocketProgramming
- Files:** TCPClient.java, WebServer.java, UDPClient.java, UDPServer.java, main\_ar.html, main\_en.html, mySite1211039.htm, Style.css
- Code Editor:** The UDPServer.java file is open, showing Java code for a UDP server. Lines 25-31 are highlighted:

```
// Prepare the message to be sent back to the client
byte[] bufferServer = fullMessage.getBytes();
DatagramPacket sendPacket = new DatagramPacket(bufferServer, bufferServer.length,
        receivedPacket.getAddress(), receivedPacket.getPort());
// Send the packet to the client
serverSocket.send(sendPacket);
```
- Terminal:** The terminal window shows the following interaction:

```
C:\Users\Ajdks\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=61868:C:\Program F
Server is listening on port: 1039
Message from Client1: Hi from client 1
Enter your message to send to client: hi client 1 im UDPServer!
```
- Status Bar:** Shows the current time (25:42), encoding (CRLF), file encoding (UTF-8), and file save status (4 spaces).

This screenshot shows the UDPServer terminal after receiving client 1 message, server can reply back to this message, it replied back with “hi client 1 I’m UDPServer” as shown in the screenshot.

This screenshot shows the UDPClient terminal window in IntelliJ IDEA. The terminal output shows the server responding to client 1 and then client 2 sending a message.

```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=61875:C:\Program F
Enter your message client1: Hi from client 1
Server Response: hi client 1 im UDPServer!
Enter your message client2: Hi from client 2
```

This screenshot shows the UDPClient terminal, client 1 received the server's message successfully, now client 2 can go ahead and message the server, as shown in the screenshot client 2 sends a message saying "Hi from client 2".

This screenshot shows the UDPClient terminal window in IntelliJ IDEA. The terminal output shows the server listening on port 1039 and clients 1 and 2 sending messages to the server.

```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=61868:C:\Program F
Server is listening on port: 1039
Message from Client1: Hi from client 1
Enter your message to send to client: hi client 1 im UDPServer!
Message from Client2: Hi from client 2
Enter your message to send to client: hi client 2 im UDPServer!
```

This screenshot shows the UDPServer terminal, the server has successfully received the message from client 2, and can reply back to client 2, it replied back with “hi client 2 im UDPServer!”.

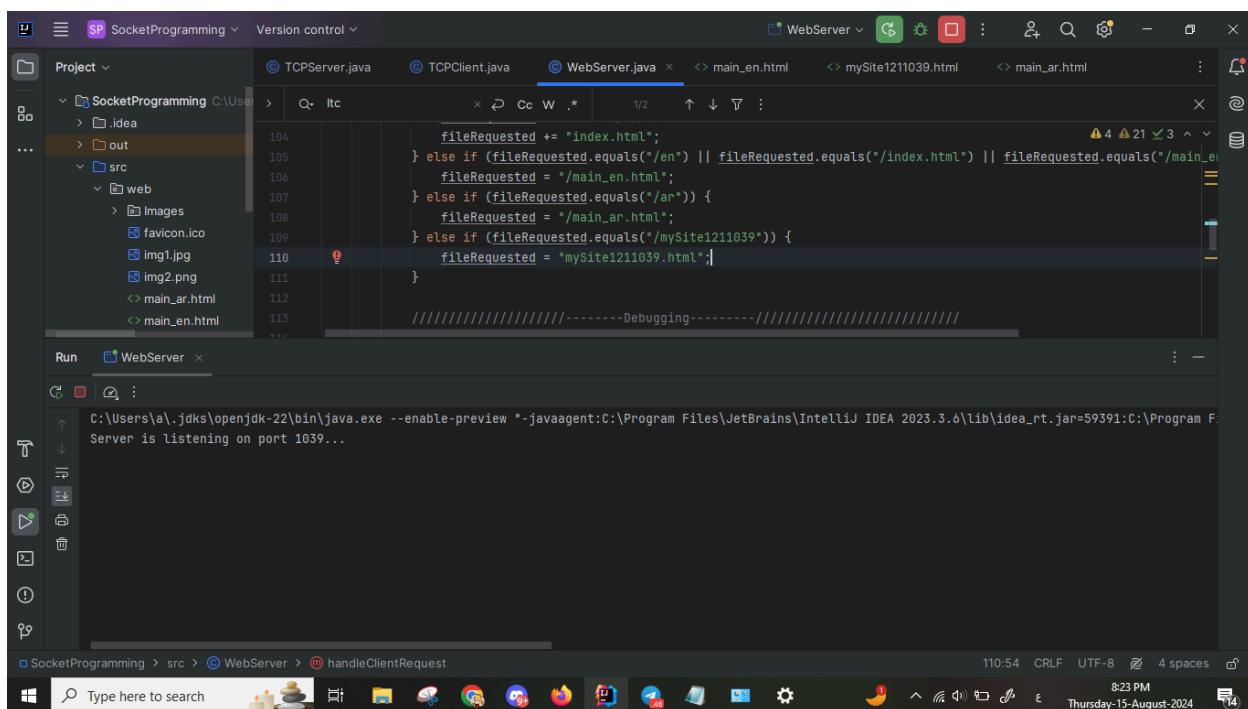
The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'Project' tool window on the left lists files like main\_ar.html, main\_en.html, mySite1211039.htm, Style.css, TCPClient, TCPServer, and UDPClient. The 'Run' tool window at the bottom shows the terminal output for the UDPClient run:

```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=61875:C:\Program F
Enter your message client1: Hi from client 1
Server Response: hi client 1 im UDPServer!
Enter your message client2: Hi from client 2
Server Response: hi client 2 im UDPServer!
Enter your message client3: |
```

This screenshot displays the terminal for UDPClient. Client 2 has successfully received a message from the server, and Client 3 is now ready to send its message to the server.

## Task3: Web Server

As part of my assignment, I implemented a simple web server using Java socket programming, listening on port 1039. The server handles requests for various paths, serving HTML and CSS files, as well as images in .jpg and .png formats. For specific requests, it serves main\_en.html for English content and main\_ar.html for Arabic content. Additionally, the server provides information about me, including my bio, showcases my art projects, and allows clients to search for images within a folder containing my drawings. It also includes redirections and custom 404 error handling, and it logs all HTTP requests in the terminal.



I ran the webServer in this screenshot

Next i'm attaching screenshots for opening the required URLs from my computer and the WebServer terminal output when opening each URL:

- <http://localhost:1039/index.html>



The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'WebServer.java' file is the active editor, displaying Java code for handling client requests. The code includes logic for returning 'index.html', 'main\_en.html', 'main\_ar.html', and 'mySite1211039.html' based on the requested file name. Below the code editor is the 'Run' tab, which shows the server's log output. The log indicates two client connections: one for 'img1.jpg' and another for 'img2.png'. Both files were found in the 'web' directory, and their full paths are listed in the log. The bottom status bar shows the current time as 8:26 PM on Thursday, 15 August 2024.

```

104     fileRequested += "index.html";
105 } else if (fileRequested.equals("/en") || fileRequested.equals("/index.html") || fileRequested.equals("/main_en.html")){
106     fileRequested = "/main_en.html";
107 } else if (fileRequested.equals("/ar")) {
108     fileRequested = "/main_ar.html";
109 } else if (fileRequested.equals("mySite1211039")) {
110     fileRequested = "mySite1211039.html";
111 }
112
113 //-----Debugging-----

```

## Clearing the output:

### Client Connection:

New client connected: 0:0:0:0:0:0:1: This means the server accepted a connection from your browser. The IP address 0:0:0:0:0:0:1 is the IPv6 loopback address, equivalent to 127.0.0.1 in IPv4.

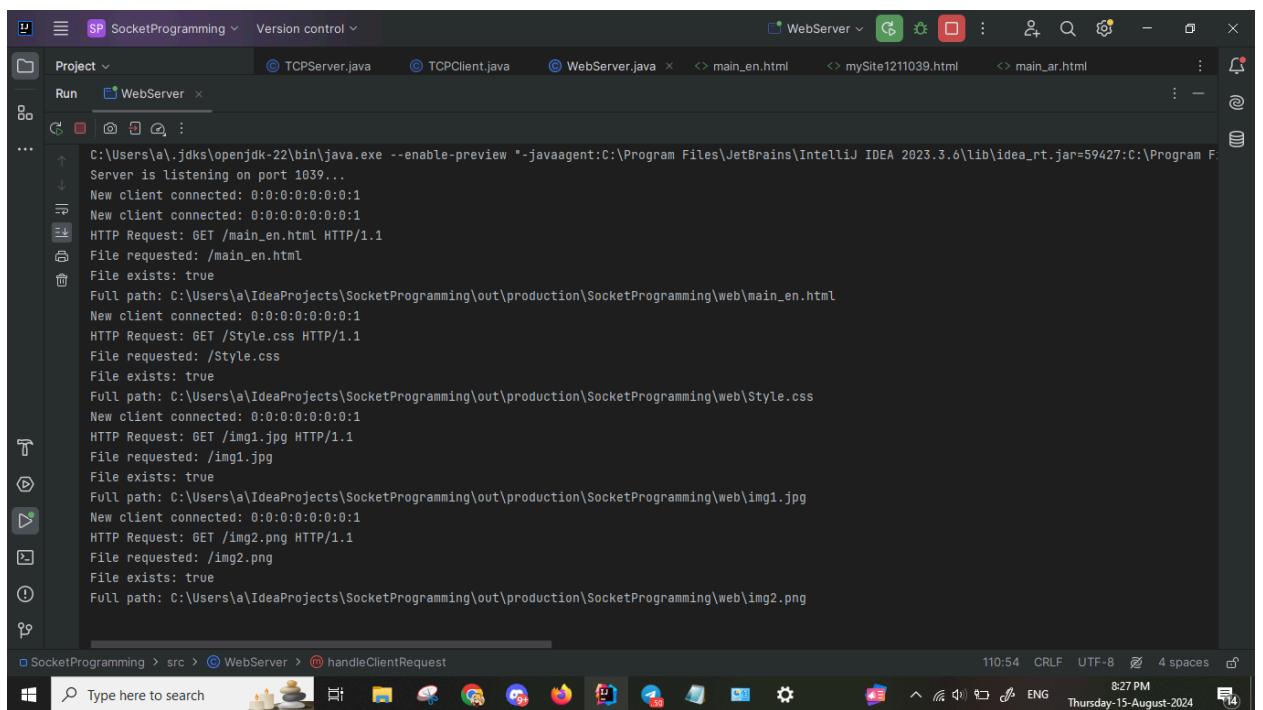
### HTTP Request Handling:

The server received a request for img1.jpg and img2.png, both of which were found in the ROOT\_IMG directory (File exists: true).

### File Serving:

The server successfully served the images (img1.jpg and img2.png) from the ROOT\_IMG directory. It logged the full paths to these images.

- [http://localhost:1039/main\\_en.html](http://localhost:1039/main_en.html)



## **Clearing the output:**

### **Server Start:**

The server successfully starts and begins listening on port 1039.

### **Client Connections:**

New client connected: 0:0:0:0:0:0:0:1: This indicates that a new client has connected to the server. The IP address shown (0:0:0:0:0:0:1) is the IPv6 loopback address, equivalent to 127.0.0.1 in IPv4, meaning the client is on the same machine as the server.

### **HTTP Requests:**

HTTP Request: GET /main\_en.html HTTP/1.1: The client is asking for the main\_en.html file.

File exists: true: The server found the requested file and is ready to serve it.

Full path: ...\\main\_en.html: The complete file path on the server's machine where the file is located.

### **Serving Files:**

The server logs show that it successfully served several files, including main\_en.html, style.css, img1.jpg, and img2.png.

- <http://localhost:1039/en>



Project: SocketProgramming

Run: WebServer

```

C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea_rt.jar=59442:C:\Program
Server is listening on port 1039...
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /en HTTP/1.1
File requested: /main_en.html
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\main_en.html
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /Style.css HTTP/1.1
File requested: /Style.css
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /img1.jpg HTTP/1.1
File requested: /img1.jpg
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /img2.png HTTP/1.1
File requested: /img2.png
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img2.png

```

110:54 CRLF UTF-8 8:28 PM Thursday-15-August-2024

Clearing the output:

### **Server Initialization:**

The server is up and running on port 1039, just like before.

### **Client Request (for /en):**

HTTP Request: GET /en HTTP/1.1

The server received a request for the /en endpoint. It appears that the server mapped this to main\_en.html.

### **File Handling for main\_en.html:**

The file main\_en.html is found in the root directory and is served to the client (your browser).

### **Full path:**

C:\Users\alberto\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\main\_en.html

The file was successfully found and served.

### **Additional Requests:**

After serving main\_en.html, the browser made additional requests for resources like style.css, img1.jpg, and img2.png. These resources were likely linked within main\_en.html:

- <http://localhost:1039/ar>



```

C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=59461:C:\Program F...
Server is listening on port 1039...
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /ar HTTP/1.1
File requested: /main_ar.html
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\main_ar.html
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /Style.css HTTP/1.1
File requested: /Style.css
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /img1.jpg HTTP/1.1
File requested: /img1.jpg
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /img2.png HTTP/1.1
File requested: /img2.png
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img2.png

```

Clearing the output:

## Server Initialization:

The server is listening on port 1039, which is the same as before.

## **First Client Request (for /ar):**

HTTP Request: GET /ar HTTP/1.1

The server received a request for the /ar endpoint. The server likely interprets this request as a file or directory.

The server checks if /ar exists. Since it doesn't seem to be logged in the output directly, it likely maps /ar to main\_ar.html (as seen in the output).

## **File Handling for main\_ar.html:**

The file main\_ar.html is found in the root directory, so the server sends this file back to the client (your browser).

### **Full path:**

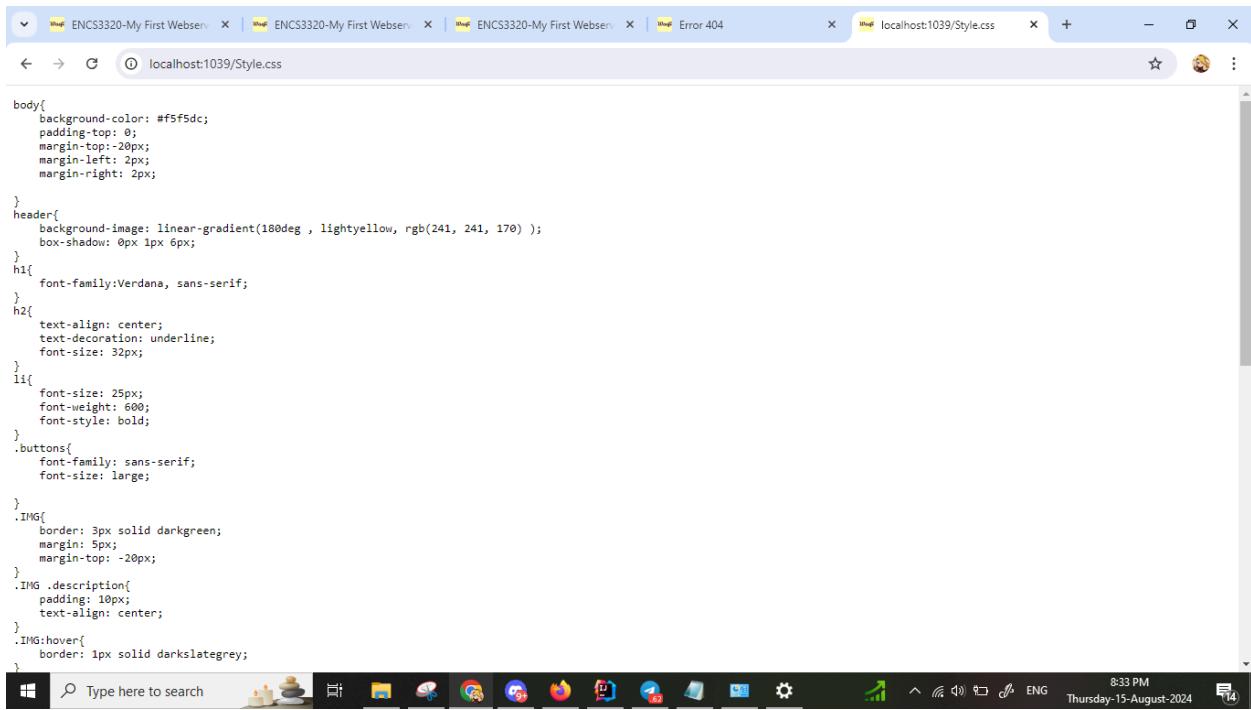
C:\Users\albert\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\main\_ar.html

The file was successfully found and served.

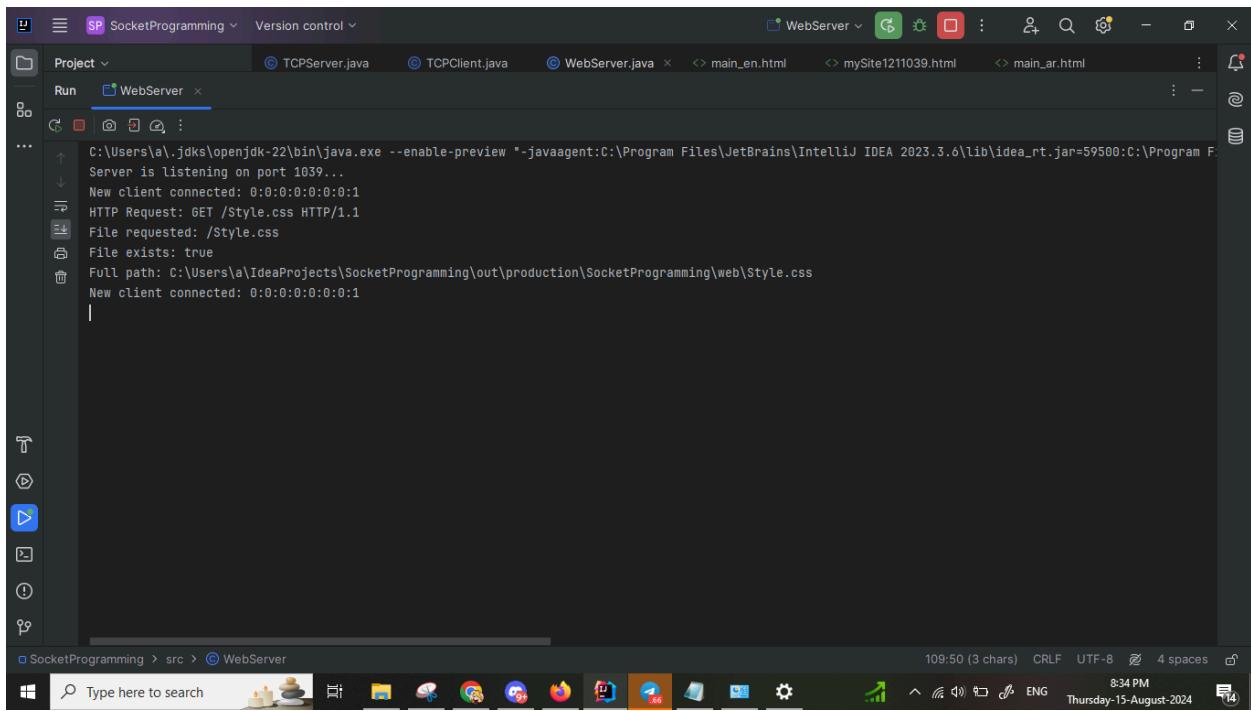
## **Additional Requests:**

After serving main\_ar.html, the browser made additional requests for resources like style.css, img1.jpg, and img2.png. These are likely referenced within main\_ar.html:

- <http://localhost:1039/Style.css>



```
body{
    background-color: #f5f5dc;
    padding-top: 0;
    margin-top: -20px;
    margin-left: 2px;
    margin-right: 2px;
}
header{
    background-image: linear-gradient(180deg , lightyellow, rgb(241, 241, 170 ) );
    box-shadow: 0px 1px 6px;
}
h1{
    font-family:Verdana, sans-serif;
}
h2{
    text-align: center;
    text-decoration: underline;
    font-size: 32px;
}
li{
    font-size: 25px;
    font-weight: 600;
    font-style: bold;
}
.buttons{
    font-family: sans-serif;
    font-size: large;
}
.IMG{
    border: 3px solid darkgreen;
    margin: 5px;
    margin-top: -20px;
}
.IMG.description{
    padding: 10px;
    text-align: center;
}
.IMG:hover{
    border: 1px solid darkslategrey;
}
```



```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea_rt.jar=59500:C:\Program F...
Server is listening on port 1039...
New client connected: 0:0:0:0:0:0:1
HTTP Request: GET /Style.css HTTP/1.1
File requested: /Style.css
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css
New client connected: 0:0:0:0:0:0:1
```

## Clearing the Output:

## **Server Initialization:**

The server is listening on port 1039, indicating that it has successfully started and is ready to handle incoming requests.

## **First Client Request (for /index.html):**

**HTTP Request:** GET /Style.css HTTP/1.1

The server received a request for the /Style.css file. This request likely came as part of the process of rendering index.html, where the browser requests associated CSS files to style the HTML content.

## **File Handling for Style.css:**

**File Requested:** /style.css

The server identifies that the requested file is /style.css.

## **Full Path:**

C:\Users\alberto\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css

The full path of the requested file is displayed, indicating where the server found the file on the filesystem.

## **Additional Client Connections:**

**New Client Connected:** 0:0:0:0:0:0:1

Another client connection was established, due to the browser making additional requests for resources needed to render the page, such as images, scripts, or other assets.

- <http://localhost:1039/img1.jpg>



```

<!DOCTYPE html>
<html>
<head>

```

C:\Users\al\jdks\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.6\lib\idea\_rt.jar=59527:C:\Program F...  
Server is listening on port 1039...  
New client connected: 0:0:0:0:0:0:1  
HTTP Request: GET /img1.jpg HTTP/1.1  
File requested: /img1.jpg  
File exists: true  
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg  
New client connected: 0:0:0:0:0:0:1  
HTTP Request: GET /favicon.ico HTTP/1.1  
File requested: /favicon.ico  
File exists: true  
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\favicon.ico

## Clearing the Output:

### Server Initialization:

- The server is listening on port 1839, which indicates it has started successfully.

### First Client Request (for /img1.jpg):

- HTTP Request:** GET /img1.jpg HTTP/1.1

The server received a request for the image file /img1.jpg.

- File Handling for img1.jpg:**

**File Requested:** /img1.jpg

The server identifies the requested file as /img1.jpg.

**File Exists:** true

The server confirms that the file /img1.jpg exists.

**Full Path:**

C:\Users\a\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg

The full path shows where the server found the file on the filesystem.

**Additional Client Request (for /favicon.ico):**

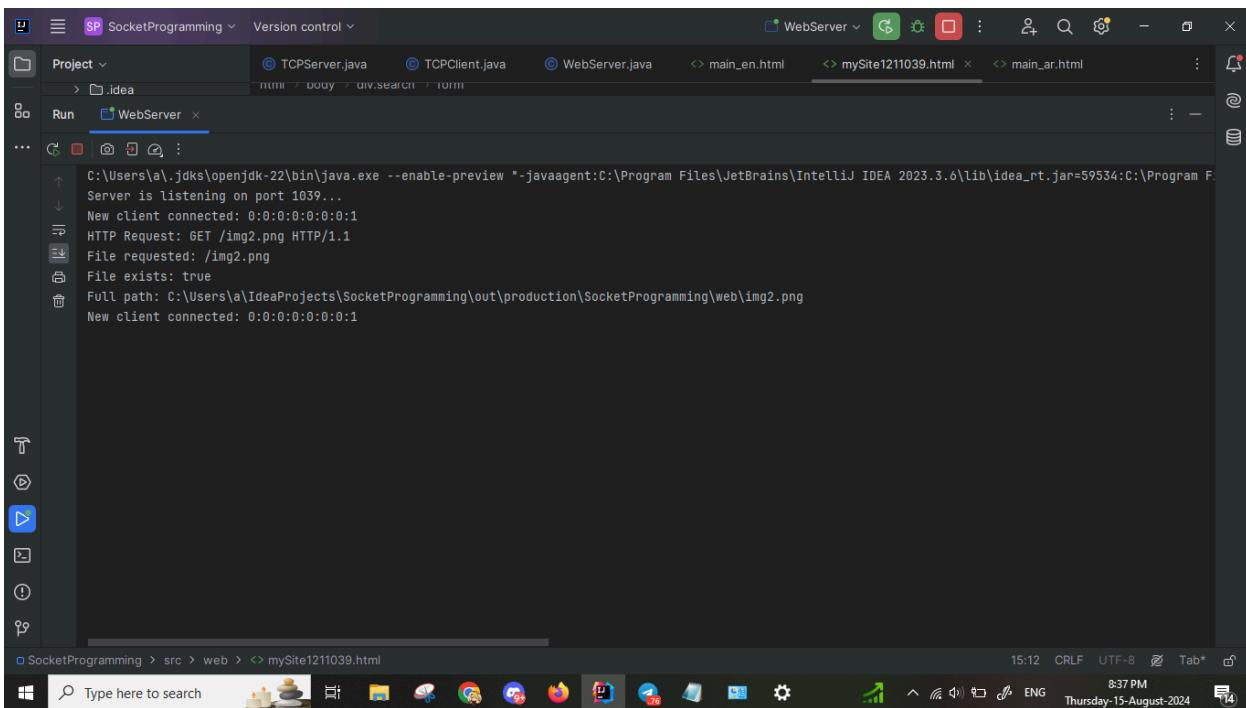
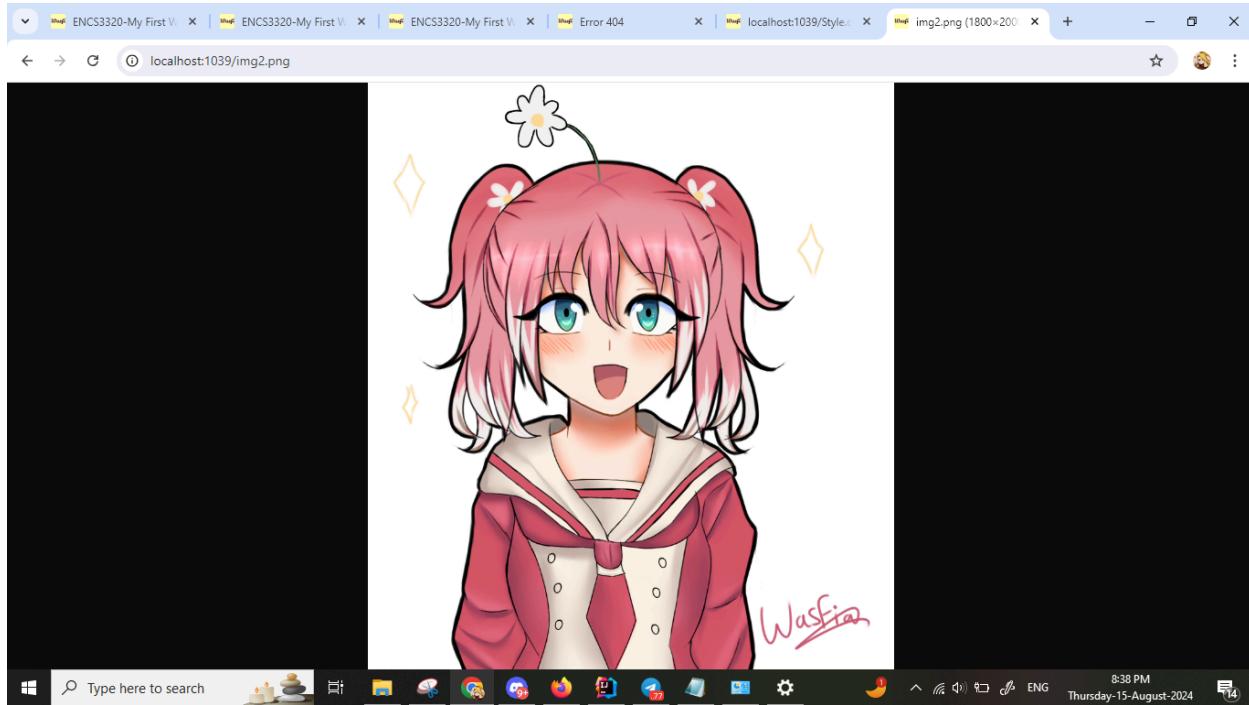
**New Client Connected:** 0:0:0:0:0:0:1

A new client connection was established, likely the browser automatically requesting the favicon.

**HTTP Request:** GET /favicon.ico HTTP/1.1

The server received a request for /favicon.ico, which is the default small icon that browsers often request.

- <http://localhost:1039/img2.png>



## Server Initialization:

The server is listening on port 1039, which indicates it has started successfully.

### **First Client Request (for /img2.png):**

**HTTP Request:** GET /img2.png HTTP/1.1

The server received a request for the image file /img2.png.

### **File Handling for img2.png:**

**File Requested:** /img2.png

The server identifies the requested file as /img2.png.

**File Exists:** true

The server confirms that the file /img2.png exists.

**Full Path:**

C:\Users\a\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img2.png

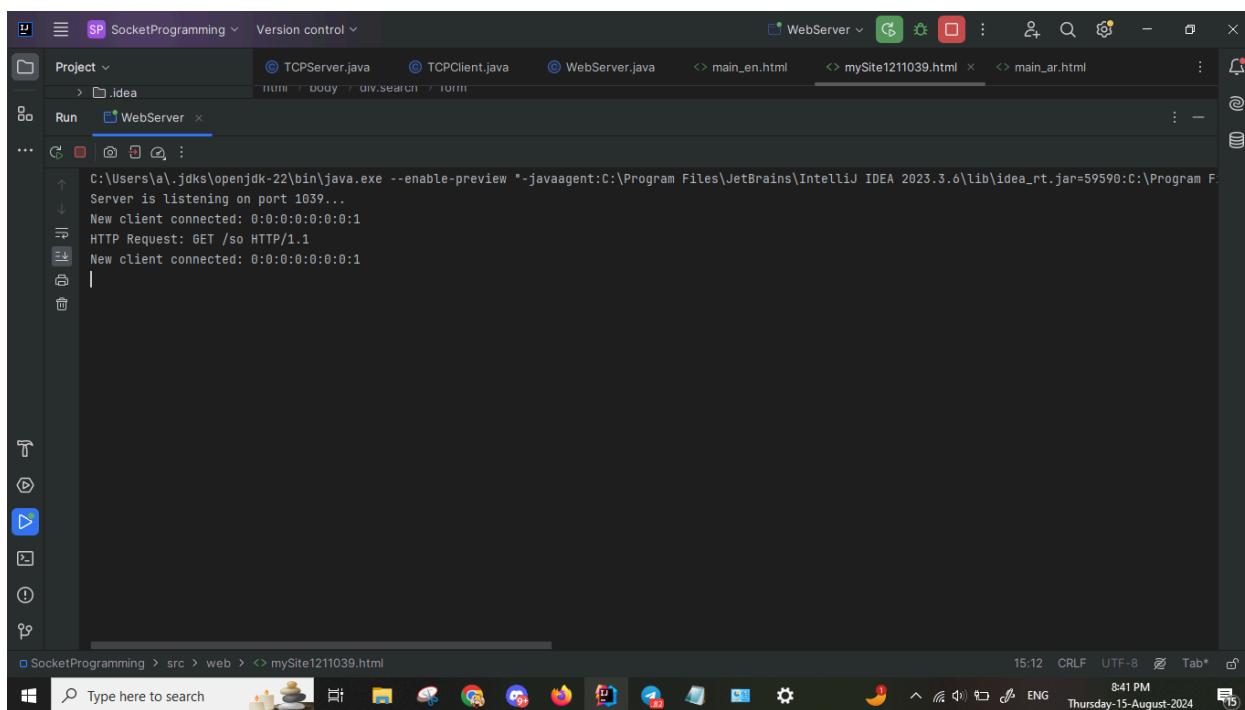
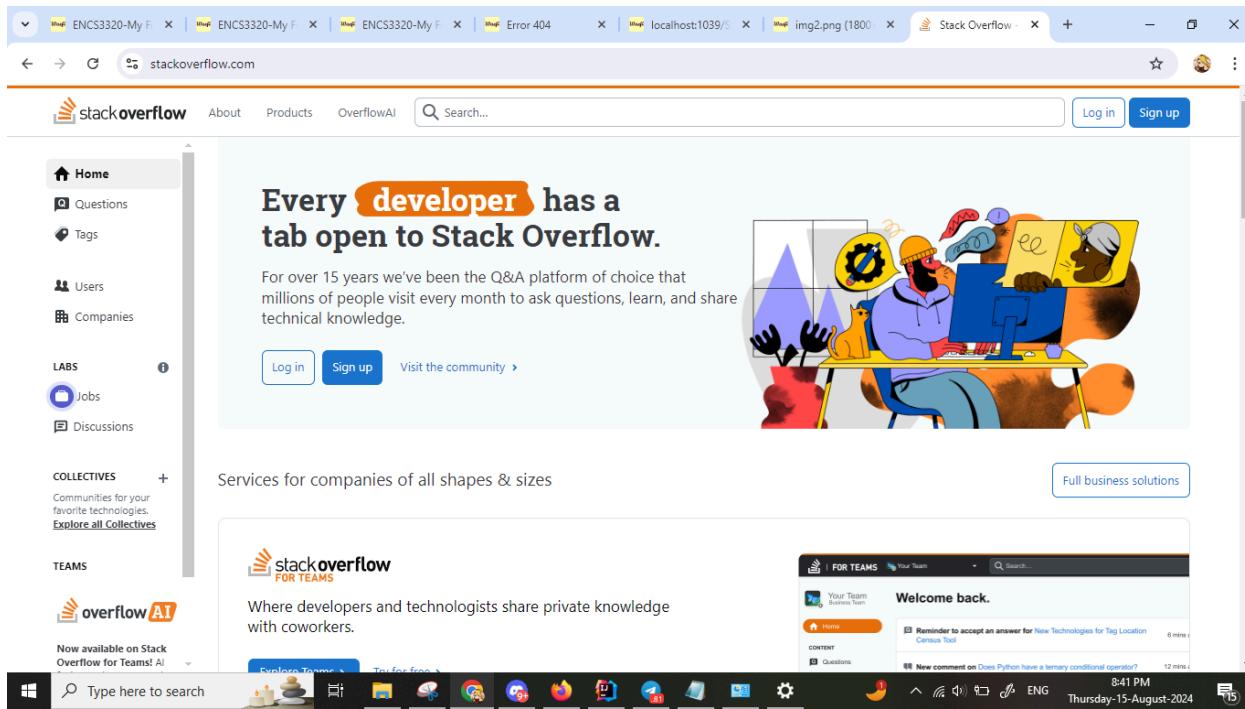
The full path shows where the server found the file on the filesystem.

### **Additional Client Connection:**

**New Client Connected:** 0:0:0:0:0:0:1

Another client connection was established, which might be due to the browser making additional requests or refreshing the connection.

- <http://localhost:1039/so>



## Clearing the Output:

## **Server Initialization:**

The server is listening on port 1039, indicating that it has started successfully.

## **First Client Request (for /so):**

**HTTP Request:** GET /so HTTP/1.1

The server received a request for the /so endpoint. This request seems to be for a file or directory that might not exist on the server.

## **No File Handling Logged:**

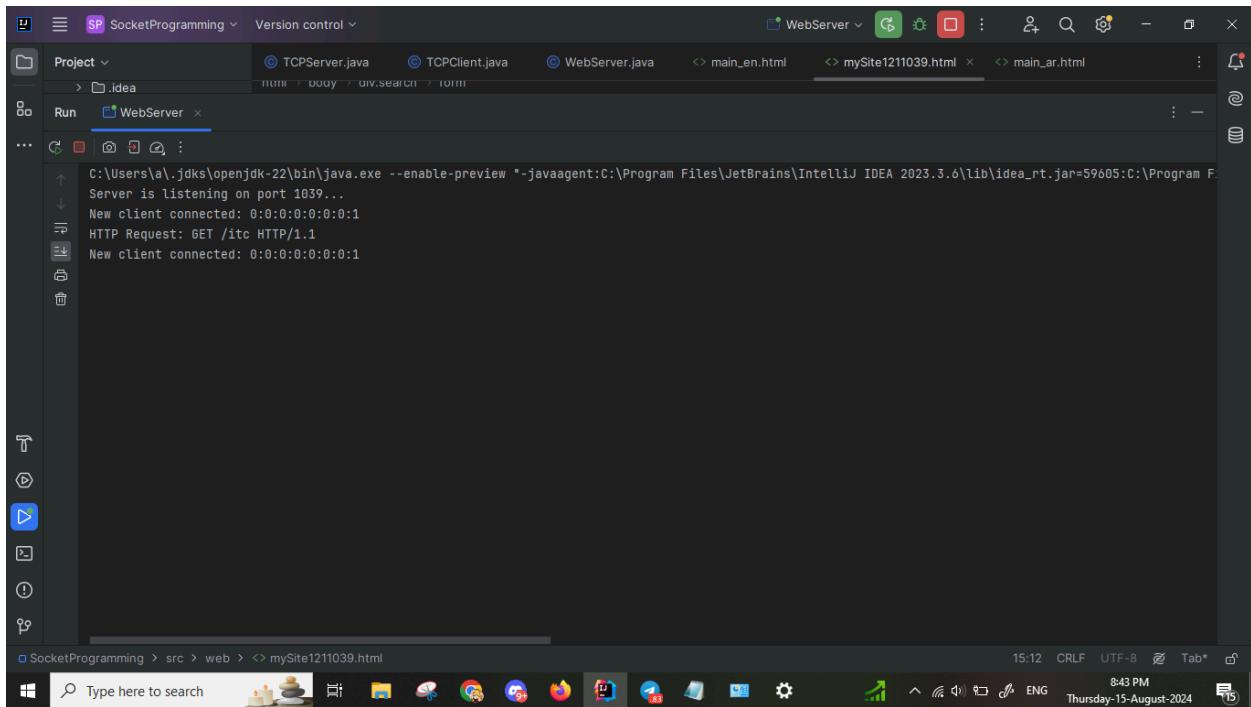
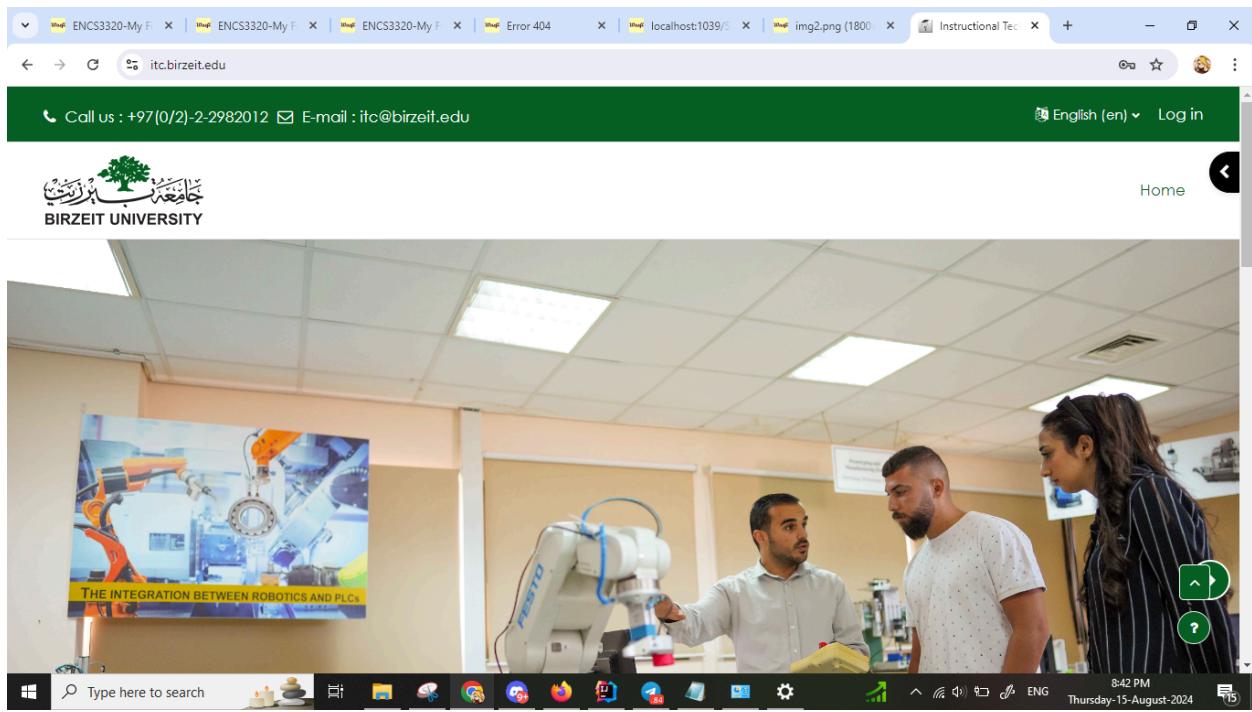
There is no log indicating whether the file /so exists or not. This could suggest that the server either did not find the file or did not log the result.

## **Additional Client Connection:**

**New Client Connected:** 0:0:0:0:0:0:0:1

Another client connection was established, possibly due to the browser retrying the request or making another request after receiving an incomplete response.

- <http://localhost:1039/itc>



## **Clearing the Output:**

## **Server Initialization:**

The server is listening on port 1039, indicating that it has started successfully.

## **First Client Request (for /itc):**

**HTTP Request:** GET /itc HTTP/1.1

The server received a request for the /itc endpoint. This appears to be a request for a file or directory that may not exist on the server.

## **No File Handling Logged:**

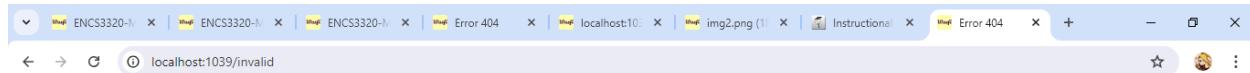
Similar to the previous request for /so, there is no log indicating whether the file /itc exists or not. This suggests that the server did not find the file or did not log the result.

## **Additional Client Connection:**

**New Client Connected:** 0:0:0:0:0:0:1

Another client connection was established, likely due to the browser retrying the request or making another request after the initial one.

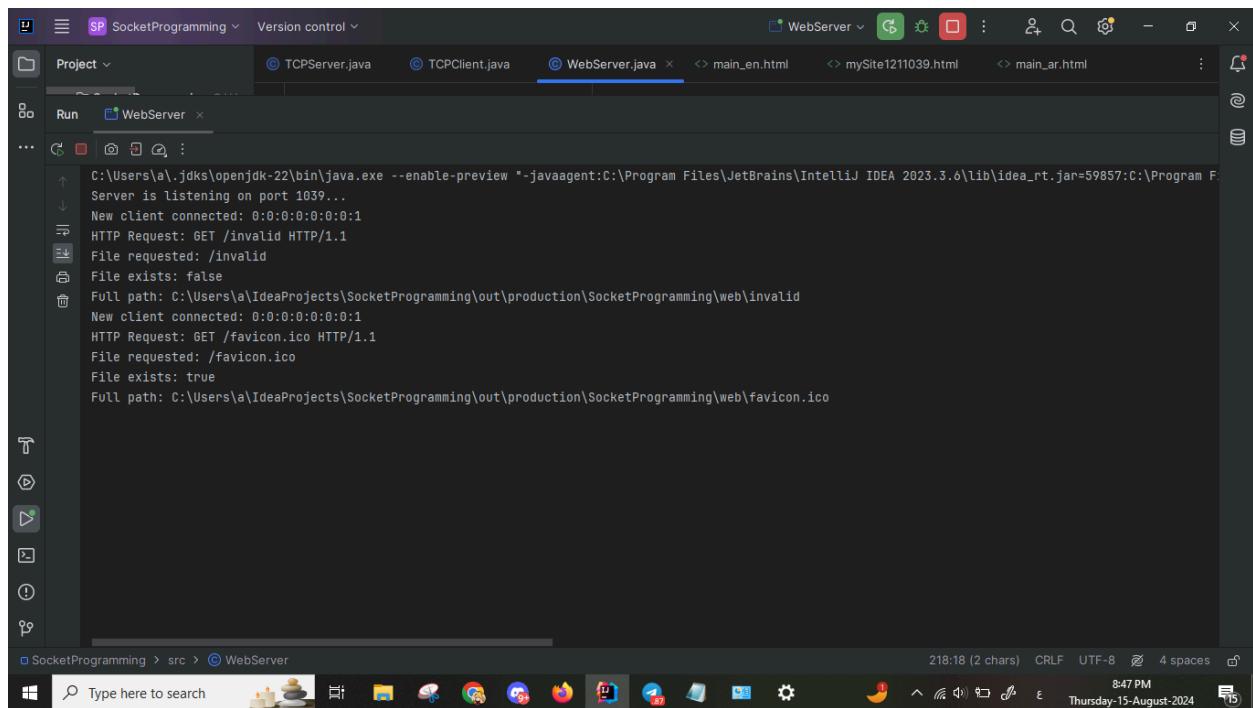
- **Invalid Requests:**



## The file is not found

Name: Wasfia Awwad  
ID: 1211039

Client IP: 0:0:0:0:0:0:1  
Port: 1039



## **Clearing the Output:**

## **Server Initialization:**

The server is listening on port 1039, indicating that it has started successfully.

## **First Client Request (for /invalid):**

HTTP Request: GET /invalid HTTP/1.1

The server received a request for the /invalid endpoint.

## **File Handling for invalid:**

File Requested: /invalid

The server identifies the requested file as /invalid.

## **File Exists: false**

The server confirms that the file /invalid does not exist in the specified directory.

## **Full Path:**

C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\invalid

The full path shows where the server attempted to find the file on the filesystem, but it was not found.

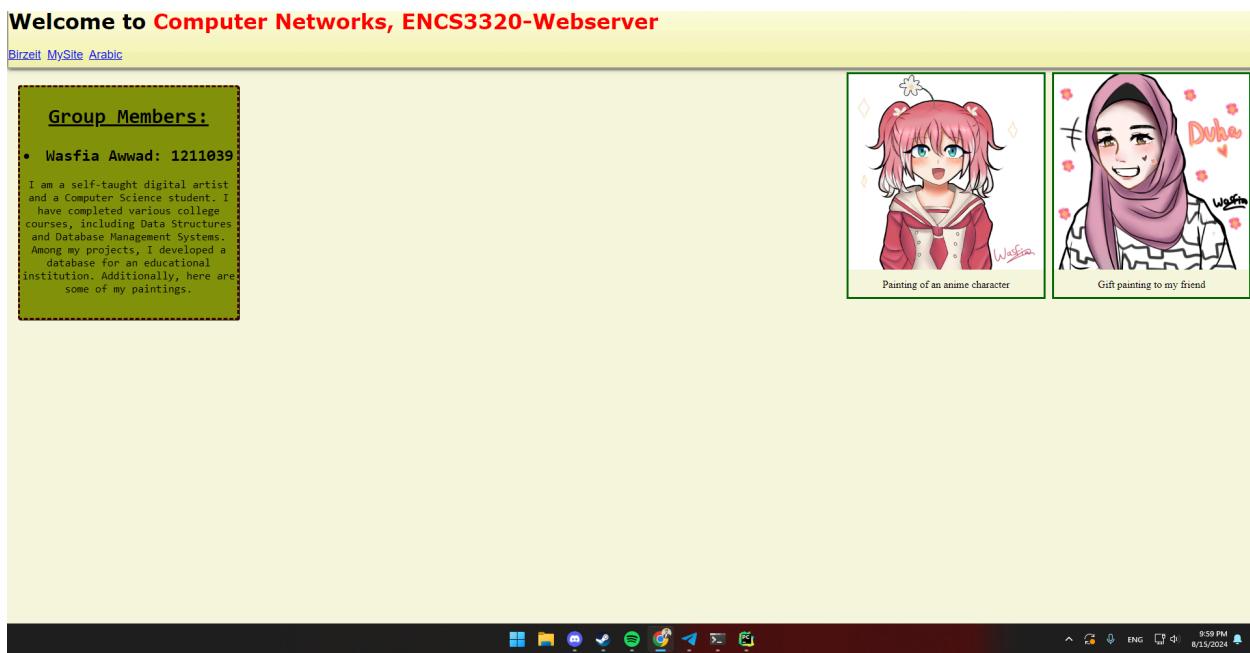
## **Additional Client Request (for /favicon.ico):**

**New Client Connected: 0:0:0:0:0:0:1**

Another client connection was established, likely the browser automatically requesting the favicon after the initial request.

## **Accessing web server from another device:**

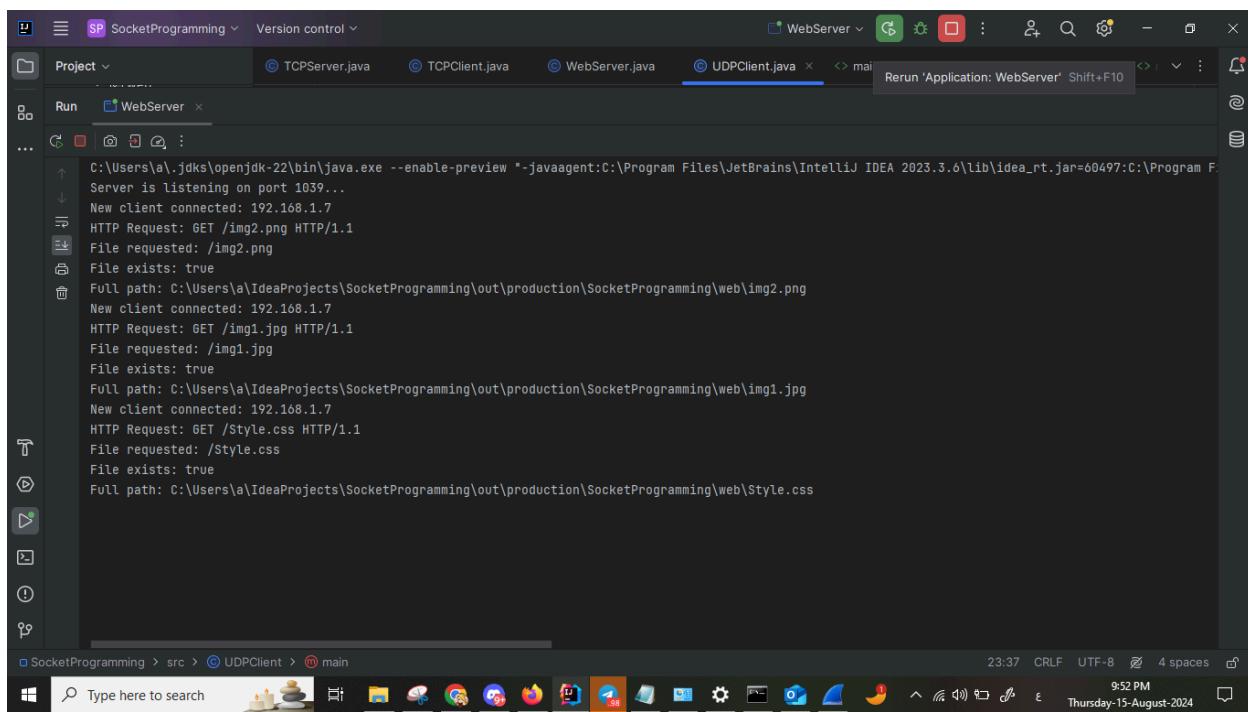
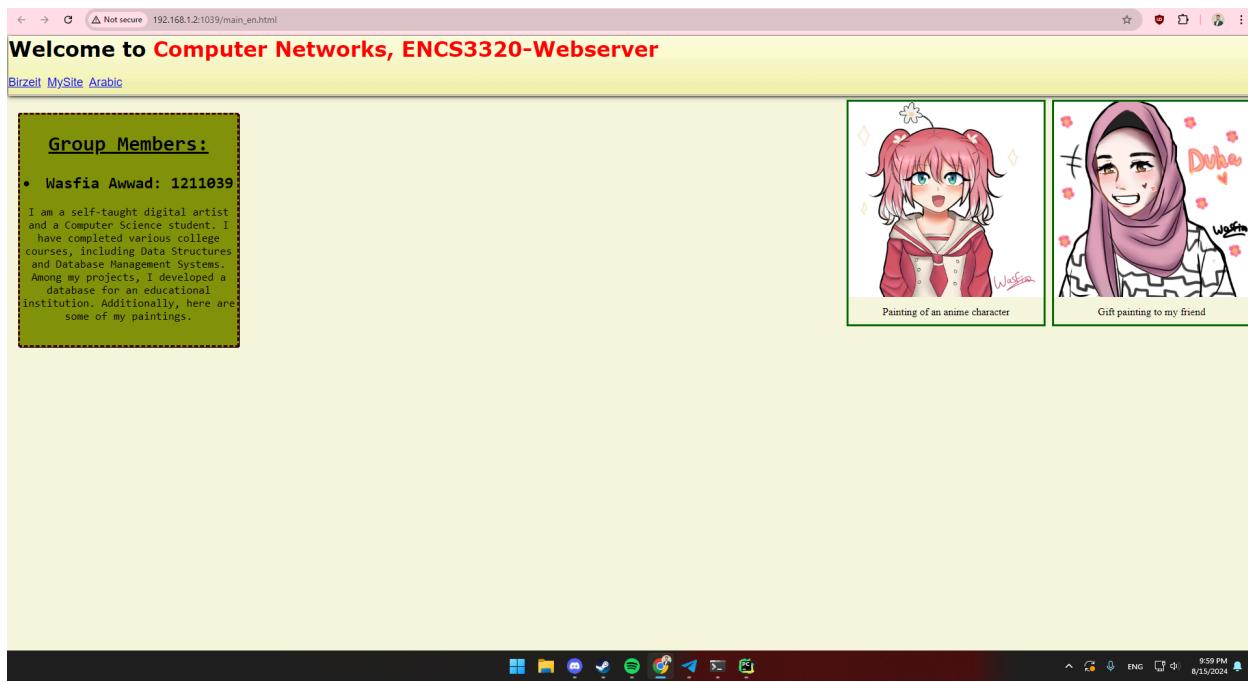
- <http://localhost:1039/index.html>



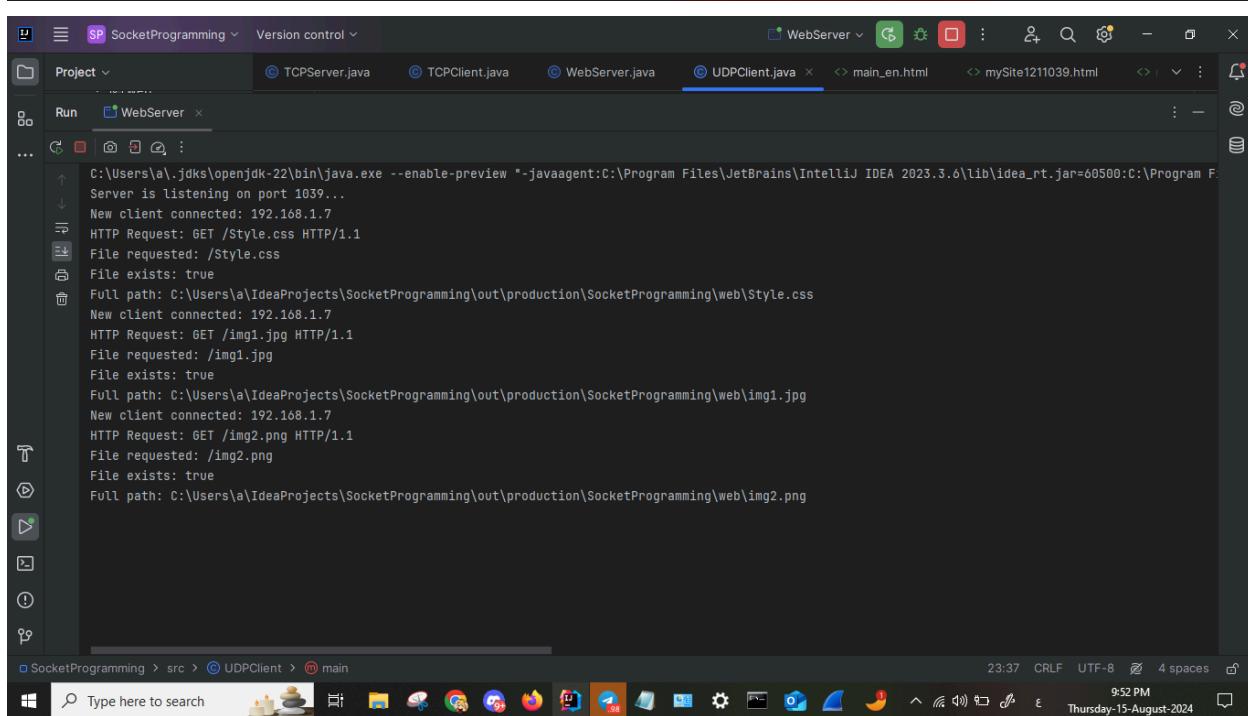
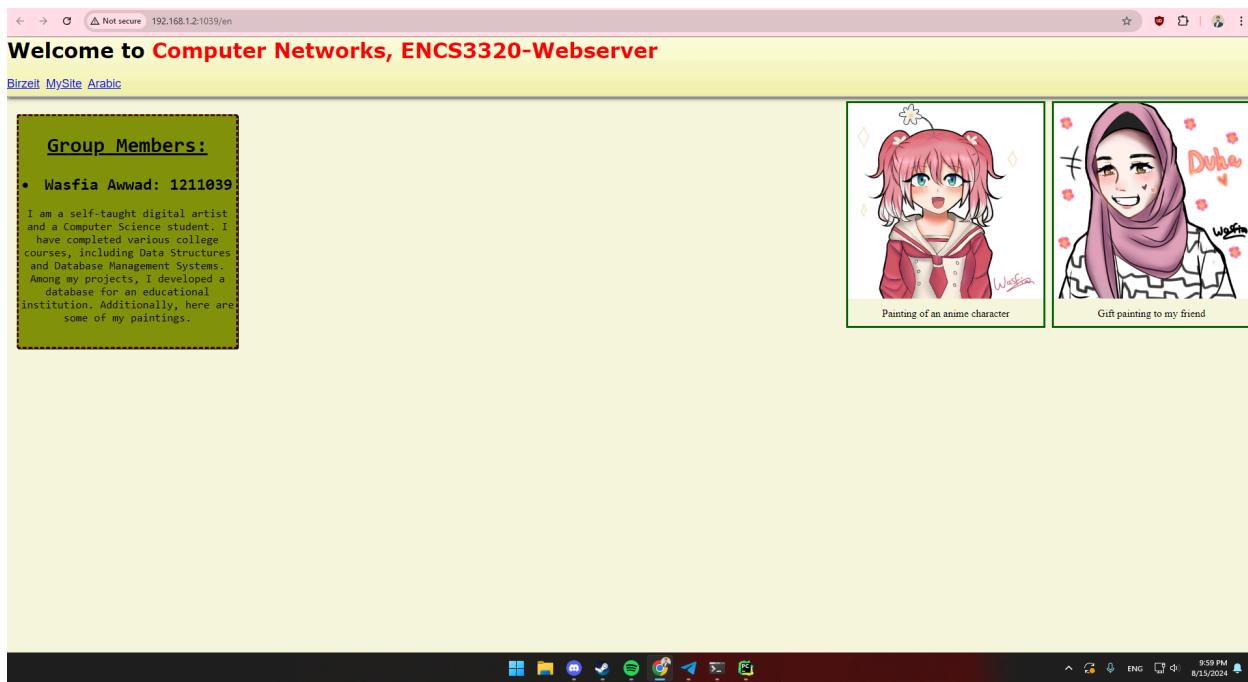
The screenshot shows the IntelliJ IDEA interface with the project 'SocketProgramming' open. The 'Run' tool window is active, showing logs for a 'WebServer' run. The logs indicate the server is listening on port 1039 and handling requests for 'Style.css', 'img1.jpg', and 'img2.png'. The IDE's status bar shows the current time as 23:37 and date as Thursday-15-August-2024.

```
C:\Users\al\jdks\openjdk-22\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=60492:C:\Program F...
Server is listening on port 1039...
New client connected: 192.168.1.7
HTTP Request: GET /Style.css HTTP/1.1
File requested: /Style.css
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css
New client connected: 192.168.1.7
HTTP Request: GET /img1.jpg HTTP/1.1
File requested: /img1.jpg
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg
New client connected: 192.168.1.7
HTTP Request: GET /img2.png HTTP/1.1
File requested: /img2.png
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img2.png
```

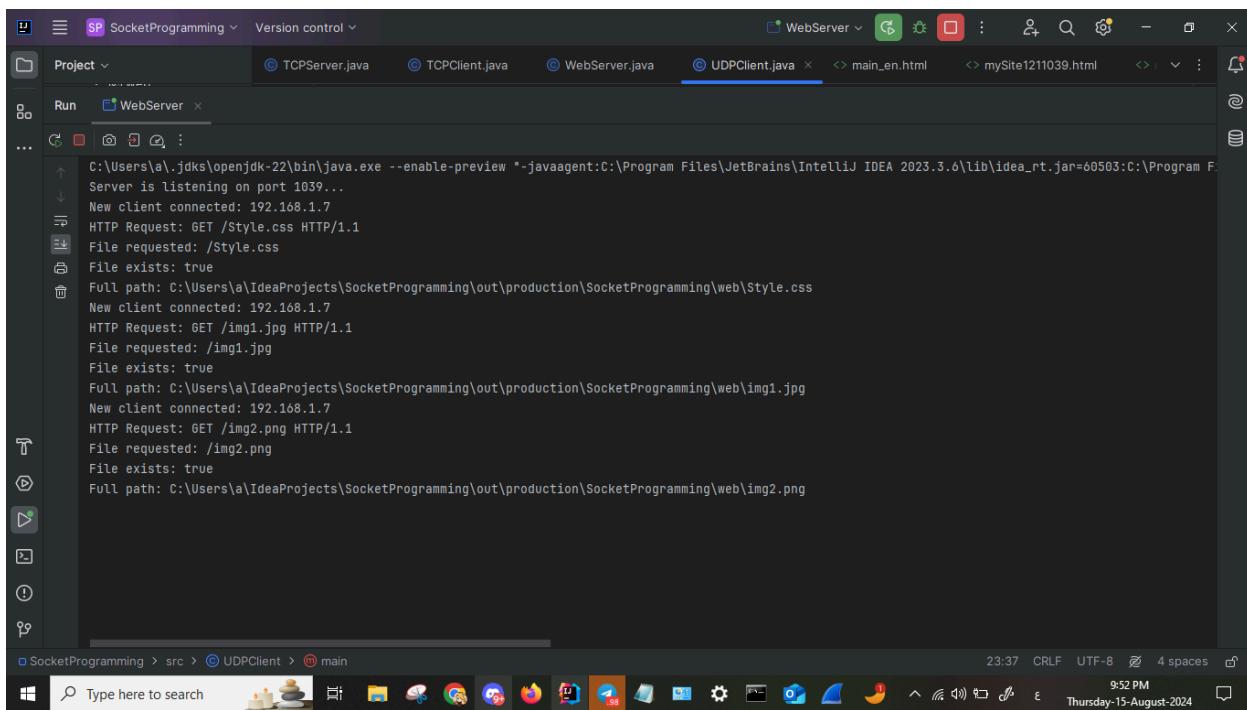
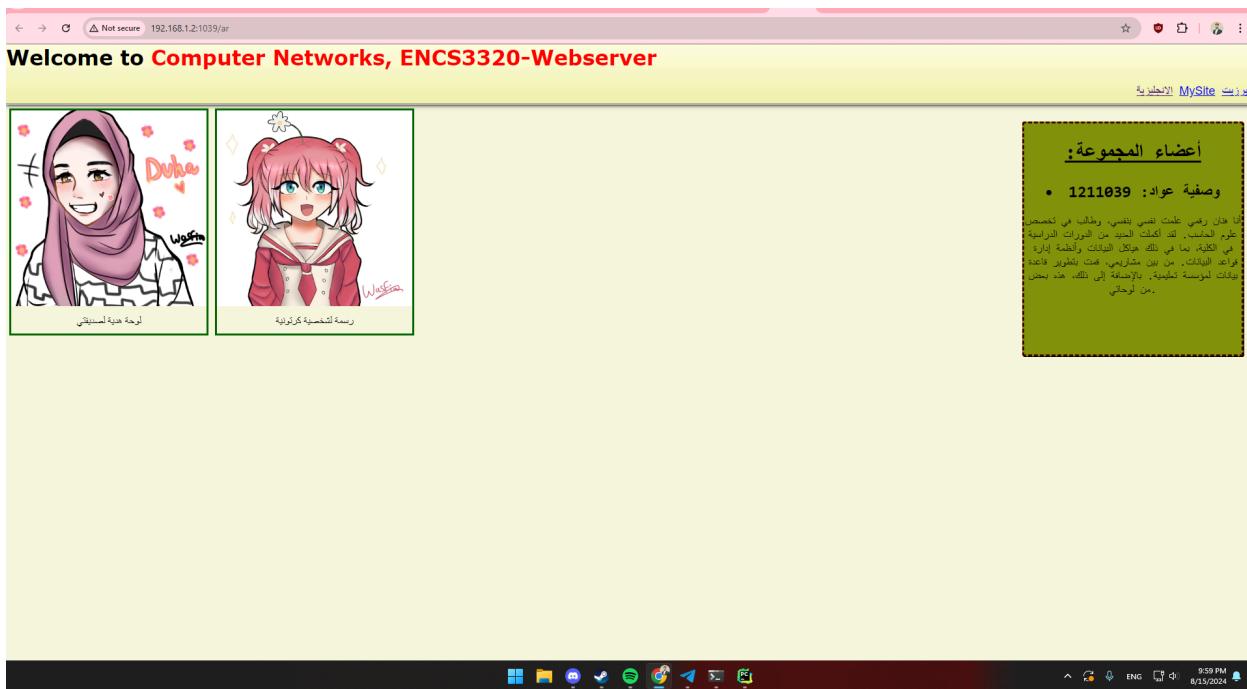
- [http://localhost:1039/main\\_en.html](http://localhost:1039/main_en.html)



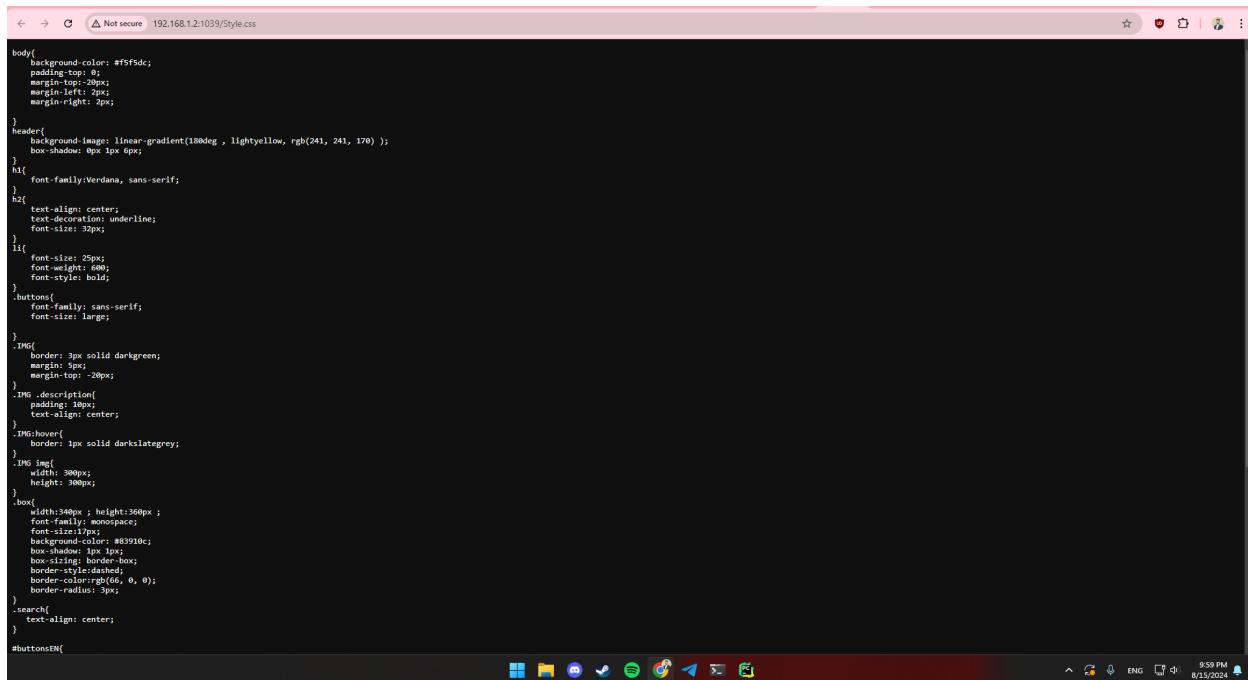
- <http://localhost:1039/en>



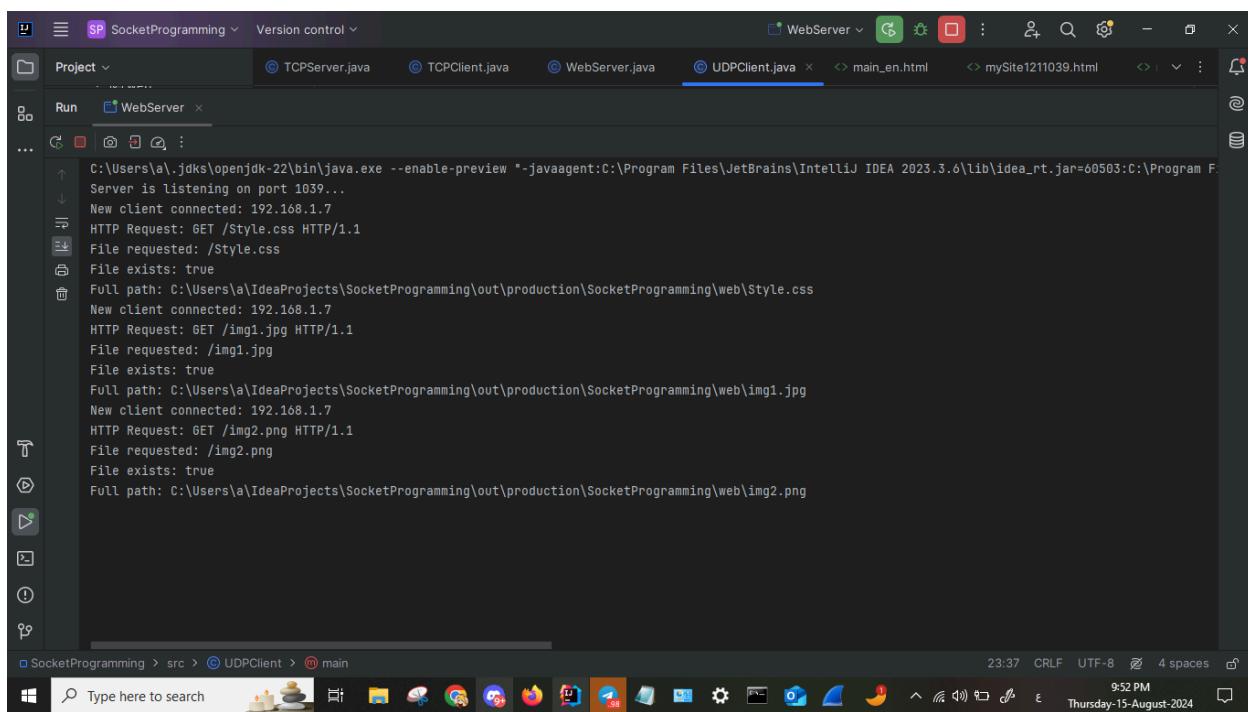
- <http://localhost:1039/ar>



- <http://localhost:1039/Style.css>

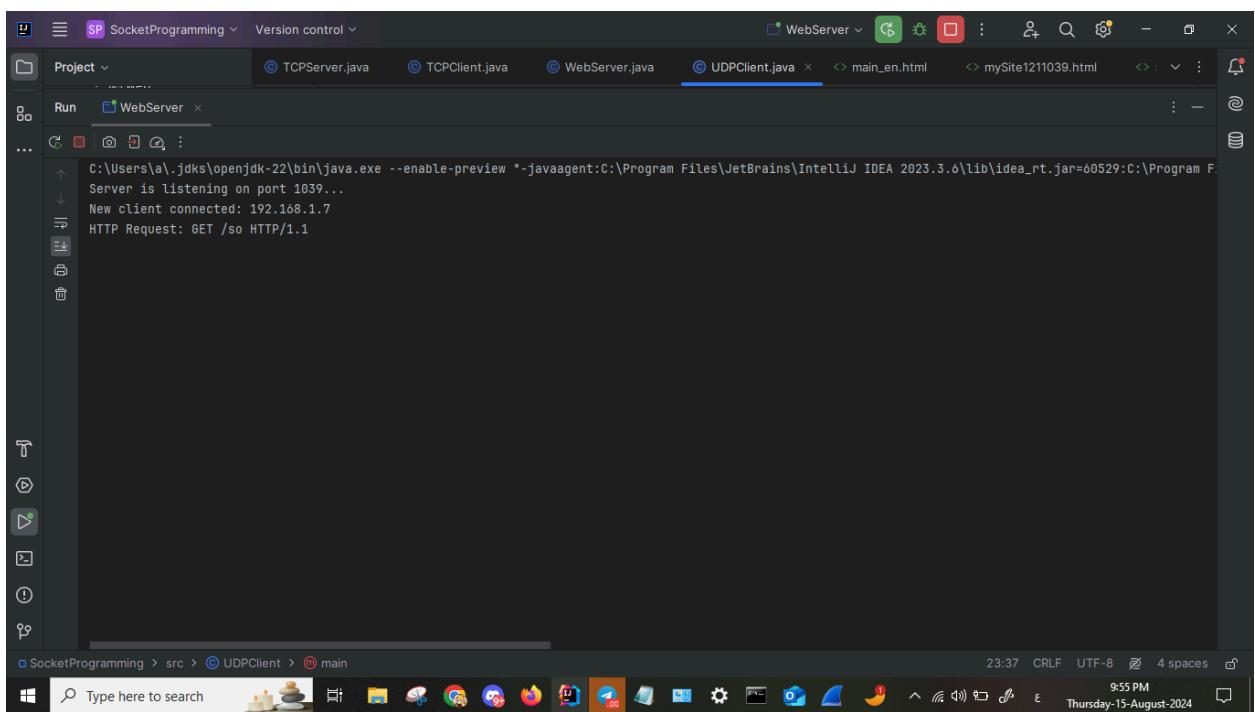
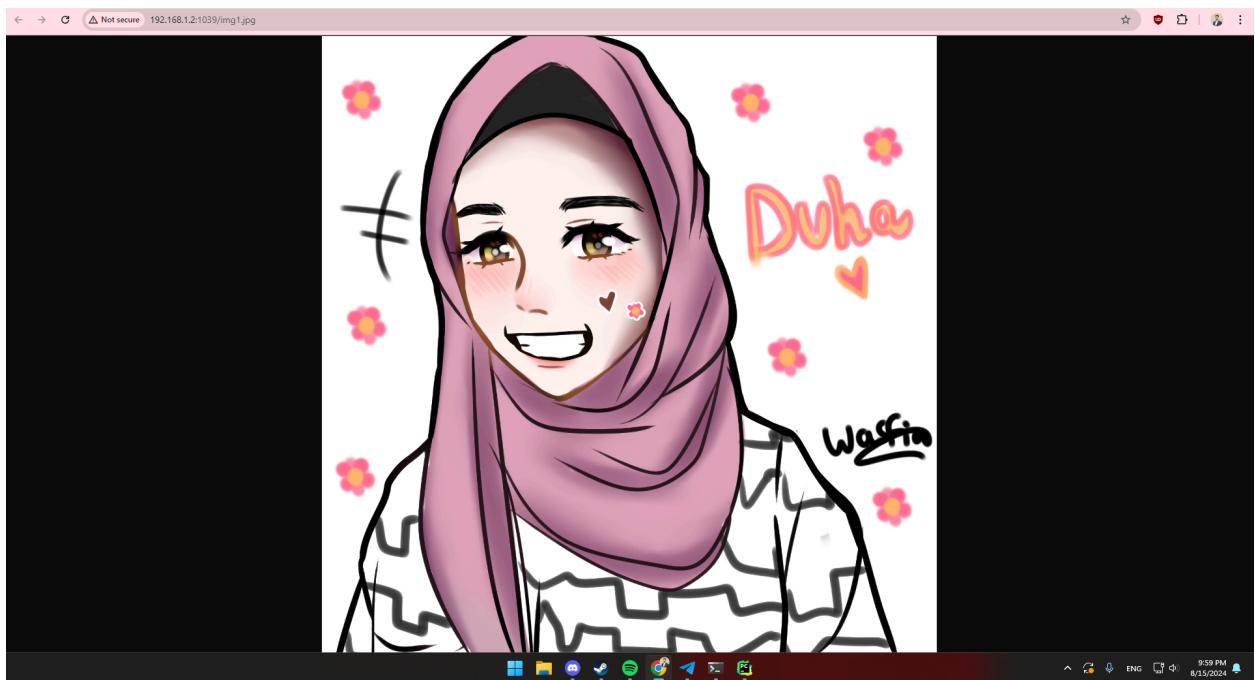


```
body{ background-color: #f5f5dc; padding-top: 8px; margin-top: 2px; margin-left: 2px; margin-right: 2px; } header{ background-image: linear-gradient(180deg , lightyellow, rgb(241, 241, 170) ); box-shadow: 0px 1px 6px; } h1{ font-family:Verdana, sans-serif; } h2{ text-align: center; text-decoration: underline; font-size: 32px; } li{ font-size: 25px; font-weight: 600; font-style: bold; } .buttons{ font-family: sans-serif; font-size: large; } .IMG{ border: 3px solid darkgreen; margin: 5px; margin-top: 20px; } .IMG_desktop{ padding: 10px; text-align: center; } .IMG_hover{ border: 1px solid darkslategray; } .IMG img{ width: 300px; height: 300px; } .box{ width:340px ; height:360px ; font-family: monospace; font-size: 10px; background-color: #83990c; box-shadow: 1px 1px 1px 1px black; border-style:dashed; border-color:rgb(66, 0, 0); border-radius: 5px; } .search{ text-align: center; } #buttonsEN{
```

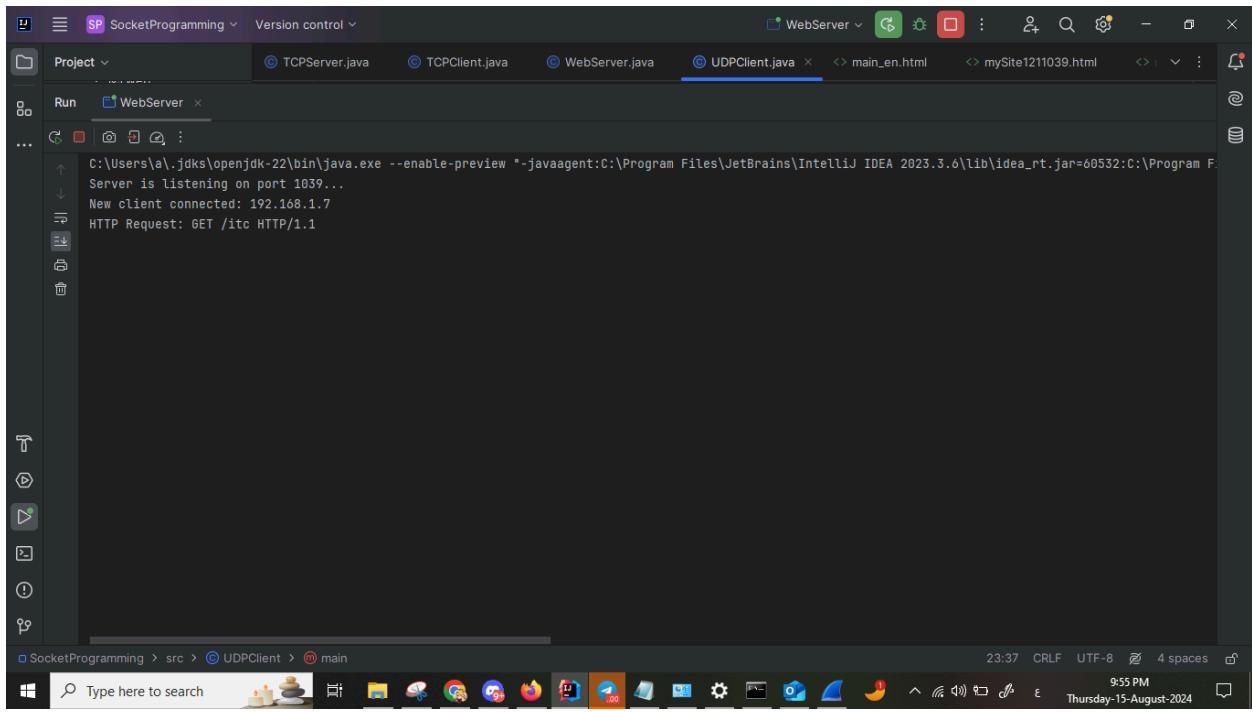
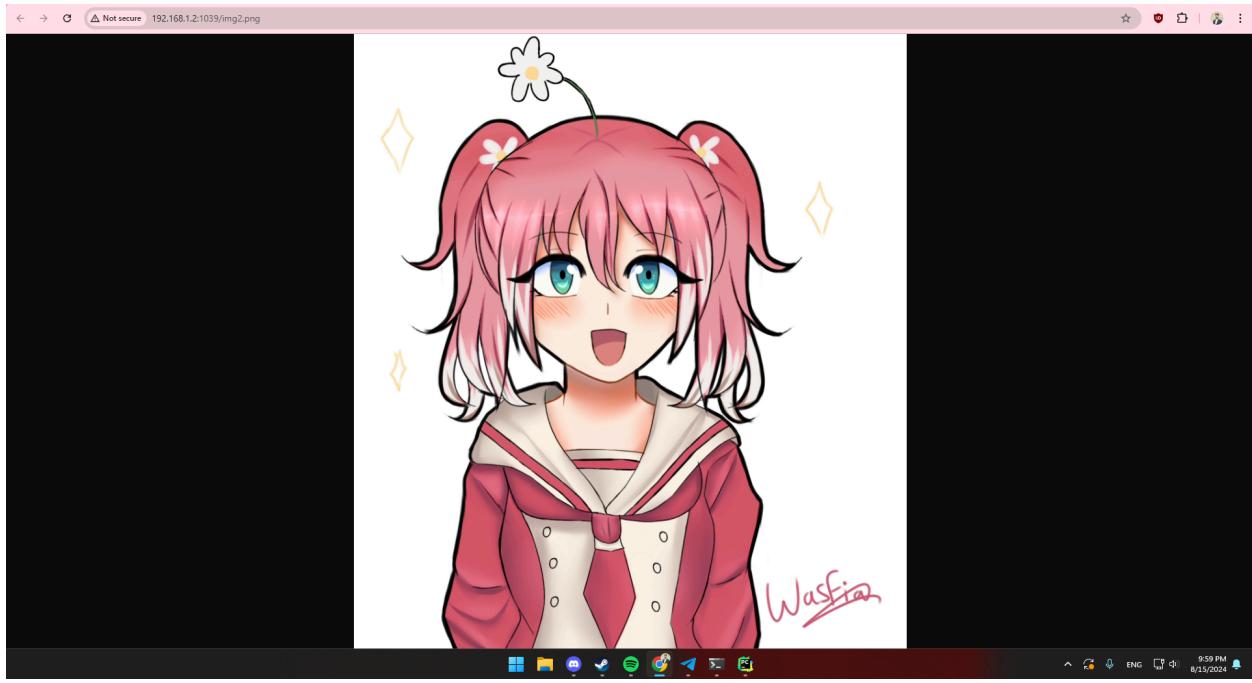


```
C:\Users\al.jdk\openjdk-22\bin\java.exe --enable-preview -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.0\lib\idea_rt.jar=60503:C:\Program F
Server is listening on port 1039...
New client connected: 192.168.1.7
HTTP Request: GET /Style.css HTTP/1.1
File requested: /Style.css
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\Style.css
New client connected: 192.168.1.7
HTTP Request: GET /img1.jpg HTTP/1.1
File requested: /img1.jpg
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg
New client connected: 192.168.1.7
HTTP Request: GET /img1.jpg HTTP/1.1
File requested: /img1.jpg
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img1.jpg
New client connected: 192.168.1.7
HTTP Request: GET /img2.png HTTP/1.1
File requested: /img2.png
File exists: true
Full path: C:\Users\al\IdeaProjects\SocketProgramming\out\production\SocketProgramming\web\img2.png
```

- <http://localhost:1039/img1.jpg>



- <http://localhost:1039/img2.png>

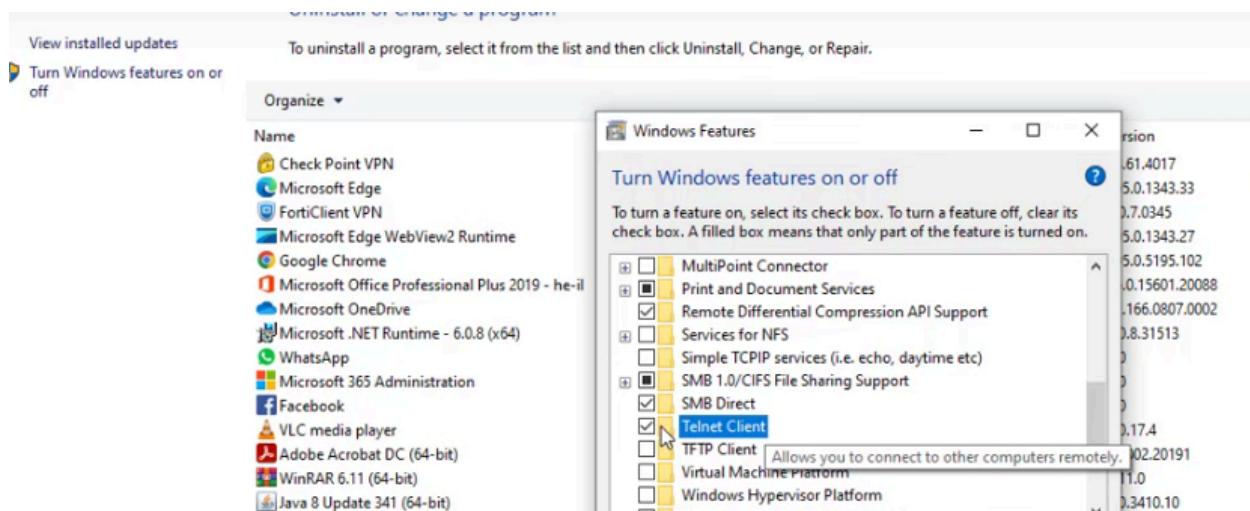


## Alternatives Solutions, Issues and Limitations

**Problem1:** while solving task 1 i faced a problem while applying the telnet [www.ox.ac.uk](http://www.ox.ac.uk) command, i had this output:

```
C:\Users\a> telnet www.ox.ac.uk.  
'telnet' is not recognized as an internal or external command,  
operable program or batch file.
```

The telnet command wasn't recognized, so i went to enable the Telnet client on my Windows system as shown in the screenshot below:



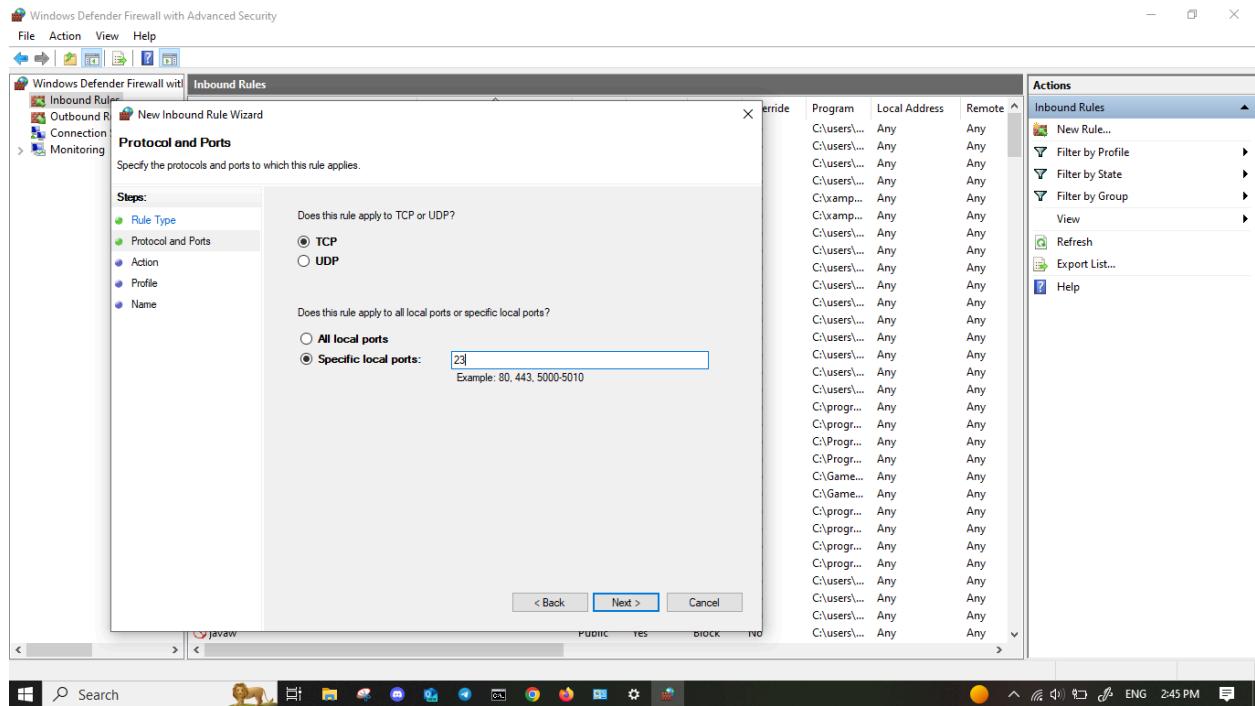
Then i tried running the command again and i had this output

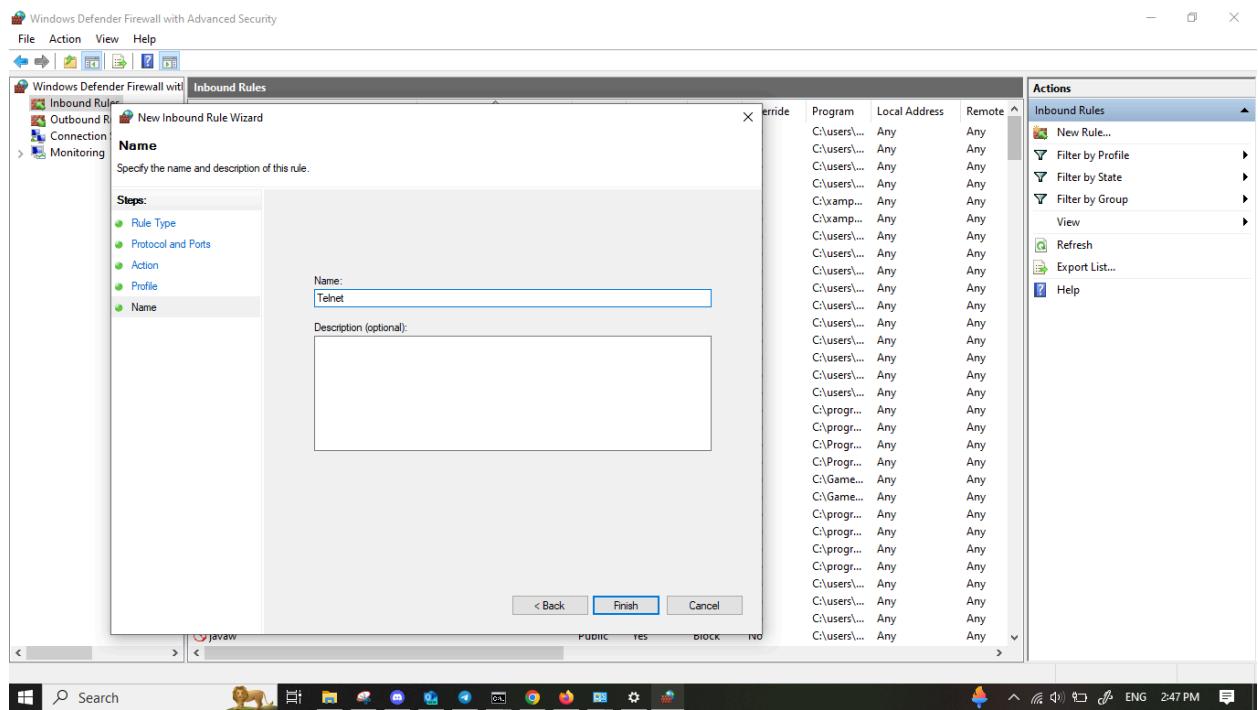
```
C:\Users\a> telnet www.ox.ac.uk.  
Connecting To www.ox.ac.uk....Could not open connection to the host, on port 23: Connect failed
```

So i figured that there's an issue with the port 23, most likely it's closed since the connection failed, so i searched for "port checker" online and entered the port number, and it turned out that it was closed as shown in the screenshot below



## Checked the firewall





Problem still didn't get solved and i had the same error.

**Problem2:** While running my webServer code in task 3 i faced a “bind exception”

A bind expectation signals that an error occurred while attempting to bind a socket to a local address and port. Typically, the port is in use, or the requested local address could not be assigned.

According to [BindException \(Java SE 11 & JDK 11\) - Oracle Help Center](#)

```

        }
    } catch (BindException e) {
        System.err.println("Server exception: Address already in use. Port " + PORT + " might be occupied.");
    } catch (IOException e) {

```

I caught the expectation with a catch block in my code, but had to solve the problem and avoid the exception to run my local web server.

I tried searching for the source that was keeping the port 1039 busy, but since i couldn't find it, i had to terminate the process by force, so i opened the CMD and typed these commands:

```
C:\Users\a>netstat -ano | findstr :1039
  TCP    0.0.0.0:1039          0.0.0.0:0              LISTENING      17256
  TCP    [::]:1039            [::]:0                LISTENING      17256

C:\Users\a>taskkill /PID <PID> /F
The system cannot find the file specified.

C:\Users\a>taskkill /PID 17256 /F
SUCCESS: The process with PID 17256 has been terminated.
```

As shown in the screenshot, The command “netstat -ano | findstr : 1039 “ lists active network connections on port 1039, showing two listening TCP connections with PID 17256. And the command “taskkill /PID 17256 /F “ successfully terminates the process with PID 17256, confirmed by the message "SUCCESS: The process with PID 17256 has been terminated." This action closes the network connections on port 1039 by force.

## Conclusion

---

Through this project, I developed essential skills in networking and web development. I learned how to create TCP and UDP sockets and gained a solid understanding of HTML and web server basics. While I'm proud of what I accomplished on my own, I regret not working in a team. I realize now that collaborating with others could have cost less time and effort. This experience has taught me the importance of teamwork, and I'll make sure to embrace collaboration in future projects.

## References:

---

<https://www.youtube.com/watch?v=-xKgxqG411c>

[Instructor video examples](#)

[BindException \(Java SE 11 & JDK 11 \) - Oracle Help Center](#)

<https://www.youtube.com/watch?v=vmuIYa9DjfI>

<https://www.youtube.com/watch?v=qYh6k-S5xC4&t=175s>