SSN COLLEGE OF ENGINEERING
KALAVAKKAM-603110

Department of Computer Science and Engineering

UCS2501 – Computer Networks - CSE-B-23

III Year CSE - (V Semester)

Establishing a Home PC Server: Enabling Remote File Retrieval via FTP

Academic Year 2023-24

Batch: 2021- 2025

Faculty Incharge: Dr Jansi Rani S. V.

Students:

Pooja Premnath	3122 21 5001 066
Rakshith Subramanian	3122 21 5001 078
Tejas V	3122 21 5001 116
Yashasvee V	3122 21 5001 124

INDEX

Problem Definition:

This project deals with converting a personal computer into a remote file server so that users anywhere on the Internet can access files from it. It also focuses on addressing challenges related to firewalls and port forwarding. The key focus is on the setup of the server, which faced considerable obstacles arising from restrictions imposed by the Internet Service Provider (ISP) on port forwarding, a critical component for external service accessibility. To circumvent this limitation, Ngrok, a tool proficient in creating secure tunnels to localhost, was strategically employed.

The implementation involves the development of a custom Python server, to facilitate HTTP-based file access. The server ensures compatibility across various devices and networks, and its architecture integrates basic authentication measures for access control, with username and password verification enhancing overall security.

The core implementation strategy revolves around a Python script utilizing the socket server modules. The server, configured to listen on port 1515, effectively manages incoming requests and serves files from a designated path on the host machine.

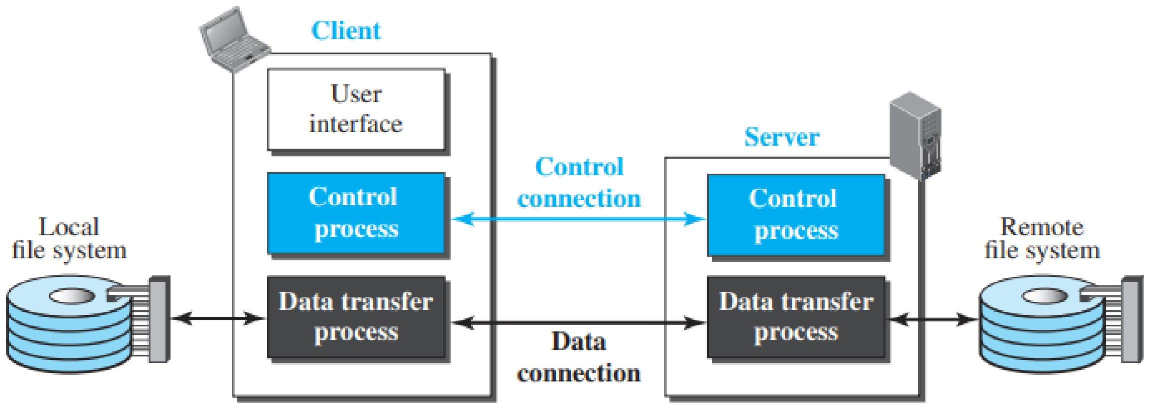
User authentication is executed through a username and password combination/ Access is contingent upon the supplied credentials, directing users to designated folders.

This report explores the establishment of a home server, offering secure and convenient remote file access. It encompasses a comprehensive overview of the methodology used, the challenges faced, the deployment of Ngrok to overcome ISP limitations, and the implementation details of the custom Python server for seamless file management.

Protocol/ Method Explanation

Introduction

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another. In FTP, the client has three components: the user interface, the client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes. Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.



Two Connections

The two connections in FTP have different lifetimes. The control connection remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transfer activity. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP session, the control connection opens.

While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred. FTP uses two well-known TCP ports: port 21 is used for the control connection, and port 20 is used for the data connection.

<u>Control Connection</u>

Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each line is terminated with a two-character (carriage return and line feed) end-of-line token. During this control connection, commands are sent from the client to the server and responses are sent from the server to the client. Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument.

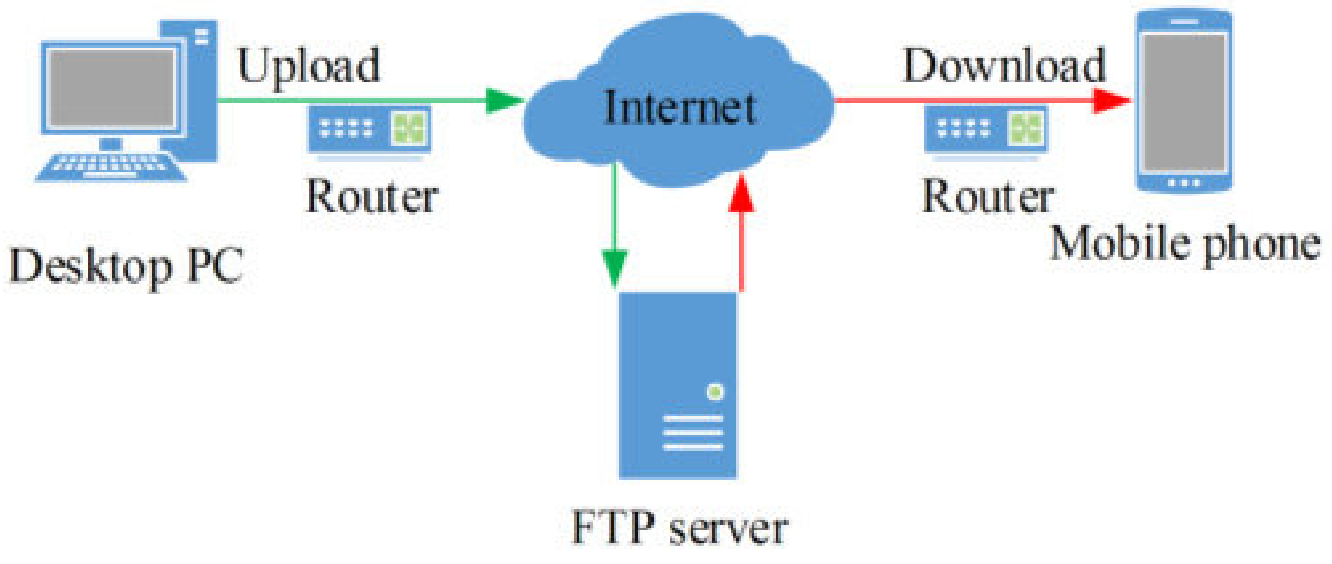
<u>Data Connection</u>

The data connection uses the well-known port 20 at the server site. However, the creation of a data connection is different from the control connection. The following shows the steps: 1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files. 2. Using the PORT command the client sends this port number to the server. 3. The server receives the port number and issues an active open using the well-known port 20 and the received ephemeral port number.

Communication over Data Connection

The purpose and implementation of the data connection are different from those of the control connection. We want to transfer files through the data connection. The client must define the type of file to be transferred, the structure of the data, and the transmission mode. Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode.

Topology



Challenges

1. Network Address Translation (NAT) Restrictions: The initial challenge arose due to the local machine being positioned behind a Network Address Translation (NAT) configuration. This common router setup inherently limits the direct accessibility of services from external locations, posing a fundamental obstacle to establishing a connection with the server.

2. ISP Restrictions on Port Forwarding (Jio Fiber): Jio Fiber, the Internet Service Provider (ISP), imposed a significant constraint by disabling port forwarding capabilities. Port forwarding, a crucial mechanism for routing external requests to specific services on the local machine, was rendered unavailable. This restriction hindered the intended goal of enabling external access to the server.

3. Firewall Configuration Challenges: The firewall introduced another layer of complexity, demanding careful configuration for successful implementation. While essential for system security, the firewall required adjustment to strike a balance between allowing external access and maintaining protection.

Usage of Ngrok

Ngrok is a versatile tunneling service that facilitates secure and seamless connections to local servers from external networks. It creates a public URL for a locally hosted service, effectively bypassing challenges like NAT configurations and restrictions imposed by ISPs on port forwarding. In the context of creating an FTP server on a home PC, Ngrok acts as a bridge between the local machine and the internet. By establishing a secure tunnel, Ngrok allows external devices to access the FTP server on the home PC without the need for complicated network configurations or exposing the server directly to the internet.

Overcoming Restrictions with Ngrok

1. Overcoming Network Address Translation (NAT) Restrictions: Ngrok proves instrumental in bypassing NAT restrictions by establishing a secure tunnel between the local machine and the external network. By creating a public URL linked to the Ngrok tunnel, external systems can connect to the server without being impeded by NAT configurations. Ngrok essentially acts as an intermediary, facilitating communication between the server and external entities despite the challenges posed by NAT.

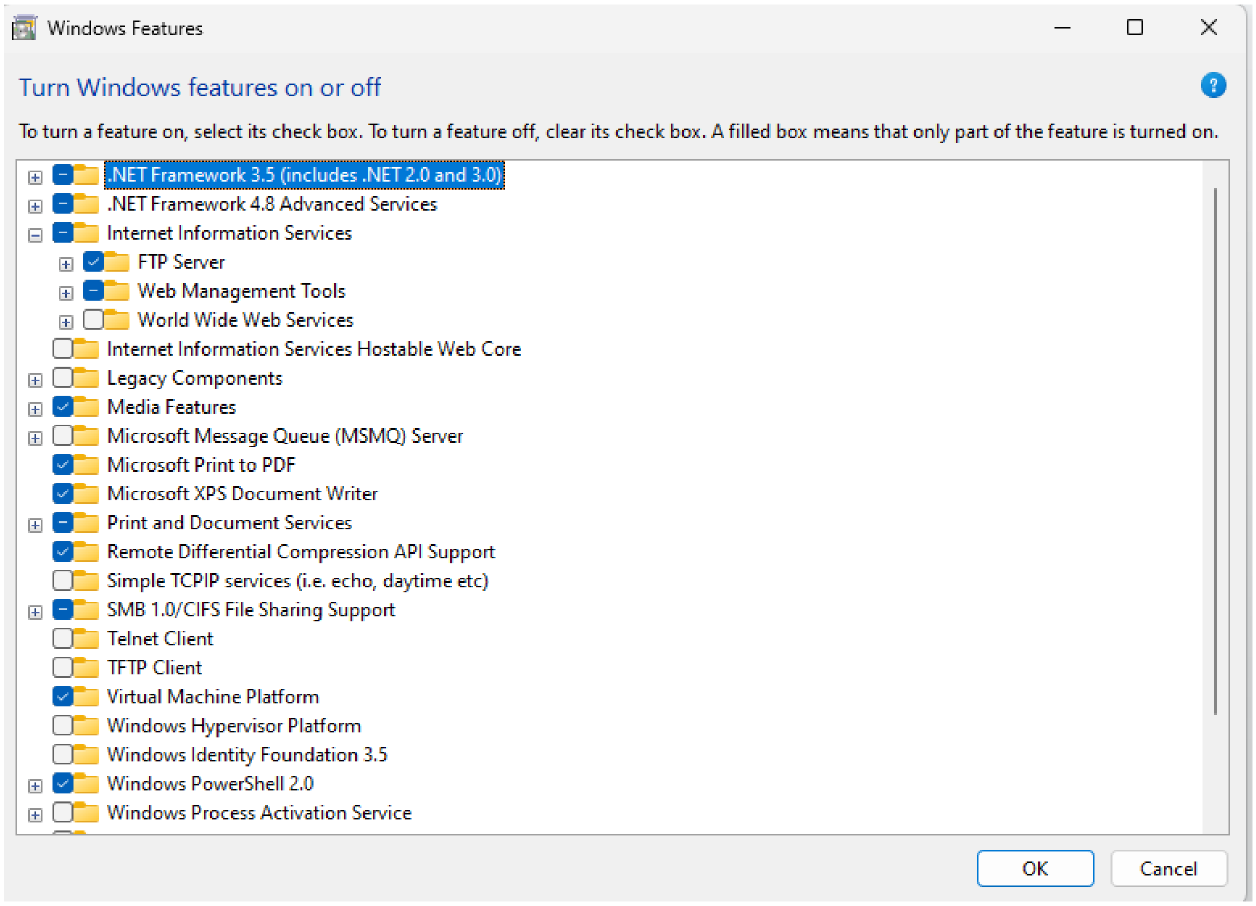
2. Overcoming ISP Restrictions on Port Forwarding (Jio Fiber): In the context of port forwarding being disabled by Jio Fiber, Ngrok circumvents this limitation by creating a secure tunnel that automatically handles the intricacies of port forwarding. Ngrok assigns a public URL to the local server, eliminating the need for manual port forwarding configurations. This allows external requests to seamlessly reach the server even in situations where the ISP restricts traditional port forwarding capabilities.

3. Overcoming Firewall Configuration Challenges: Ngrok's versatility extends to addressing firewall challenges. Since Ngrok operates by establishing outbound connections, it can traverse firewalls without requiring explicit firewall rule adjustments. This simplifies the process of enabling external access to the server while maintaining a secure firewall configuration.

System Configuration Requirements

Enabling FTP

To enable FTP on a Windows system, one must do the following: First, navigate to the "Control Panel" and select "Programs and Features." From there, choose "Turn Windows features on or off" and locate "Internet Information Services." Expand the "FTP Server" option, check "FTP Extensibility" and "FTP Service," and proceed with the installation. Once installed, configure FTP settings within IIS, specifying authentication methods and access permissions for enhanced security. This integration allows for the seamless utilization of FTP protocols, transforming the Windows machine into an FTP server.



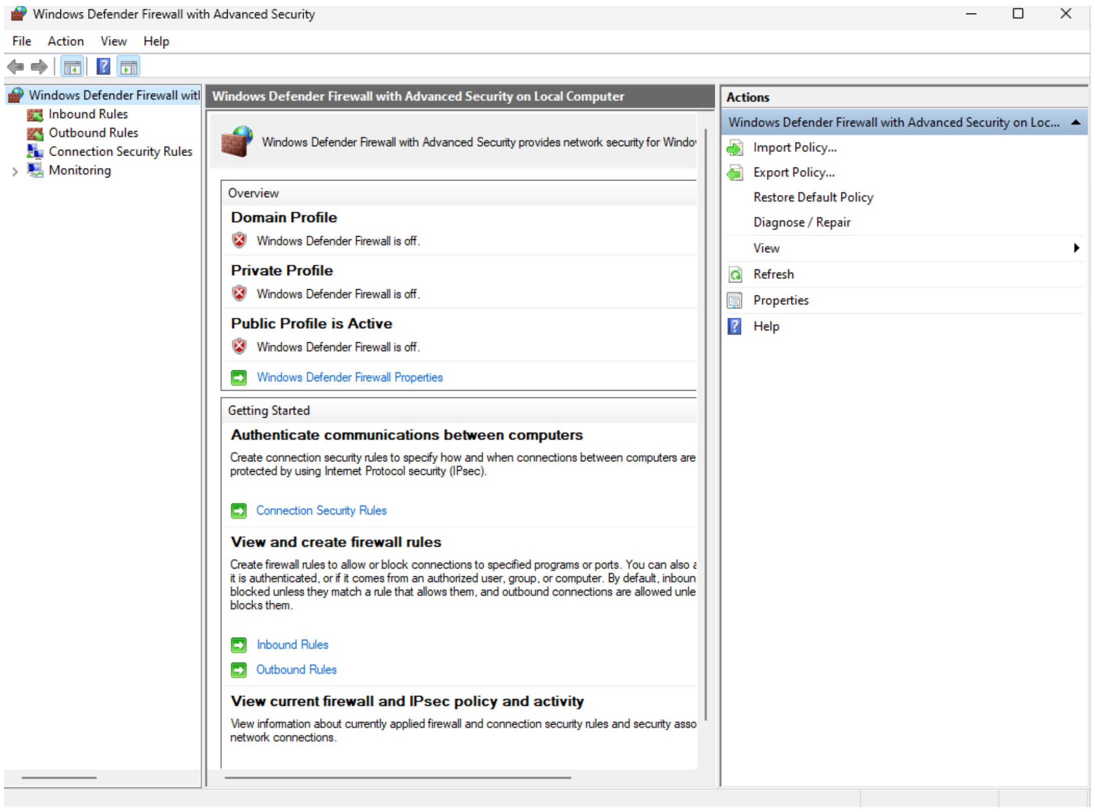
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ■ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.8 Advanced Services
- ■ Internet Information Services
  - ✓ FTP Server
  - ■ Web Management Tools
  - ☐ World Wide Web Services
- ☐ Internet Information Services Hostable Web Core
- ☐ Legacy Components
- ✓ Media Features
- ☐ Microsoft Message Queue (MSMQ) Server
- ✓ Microsoft Print to PDF
- ✓ Microsoft XPS Document Writer
- ■ Print and Document Services
- ✓ Remote Differential Compression API Support
- ☐ Simple TCPIP services (i.e. echo, daytime etc)
- ■ SMB 1.0/CIFS File Sharing Support
- ☐ Telnet Client
- ☐ TFTP Client
- ✓ Virtual Machine Platform
- ☐ Windows Hypervisor Platform
- ☐ Windows Identity Foundation 3.5
- ✓ Windows PowerShell 2.0
- ☐ Windows Process Activation Service

Circumventing Firewalls

When dealing with rigid firewall settings, it might be necessary to temporarily disable or adjust the firewall to facilitate smooth communication. In Windows, access the "Control Panel" and navigate to "System and Security" and then "Windows Defender Firewall." Here, one can either create specific inbound rules to permit FTP traffic on the designated port or, in situations where FTP access is impeded, temporarily disable the firewall for testing purposes.


Windows Defender Firewall with Advanced Security
File Action View Help
Windows Defender Firewall witl
Inbound Rules
Outbound Rules
Connection Security Rules
Monitoring
Actions
Windows Defender Firewall with Advanced Security on Loc...
Import Policy...
Export Policy...
Restore Default Policy
Diagnose / Repair
View
Refresh
Properties
Help

Windows Defender Firewall with Advanced Security on Local Computer

Windows Defender Firewall with Advanced Security provides network security for Windo

Overview

Domain Profile
Windows Defender Firewall is off.

Private Profile
Windows Defender Firewall is off.

Public Profile is Active
Windows Defender Firewall is off.

Windows Defender Firewall Properties

Getting Started

Authenticate communications between computers
Create connection security rules to specify how and when connections between computers are protected by using Internet Protocol security (IPsec).

Connection Security Rules

View and create firewall rules
Create firewall rules to allow or block connections to specified programs or ports. You can also a it is authenticated, or if it comes from an authorized user, group, or computer. By default, inboun blocked unless they match a rule that allows them, and outbound connections are allowed unle blocks them.

Inbound Rules
Outbound Rules

View current firewall and IPsec policy and activity
View information about currently applied firewall and connection security rules and security asso network connections.

Method of Implementation

The implemented solution comprises server-side Python code, Ngrok for secure tunneling, and a frontend HTML form. The Python code utilizes the socketserver modules to instantiate a rudimentary server with FTP functionality. This server incorporates user authentication, file serving based on credentials, and logging of incoming requests. The server runs concurrently within a separate thread for optimal performance.

To address challenges associated with network configurations such as NAT and ISP restrictions on port forwarding, Ngrok is employed. Ngrok establishes a secure tunnel to localhost, facilitating external access without necessitating manual port forwarding. This approach effectively mitigates complexities arising from network constraints.

The HTML form functions as the client-side interface for user authentication. Users are prompted for a username and password, and upon submission, this information is transmitted to the Python server. The server-side script subsequently validates the credentials, granting access to the designated file path in the event of successful authentication.

Modules

http.server Module:

The http.server module provides basic HTTP server classes, and in this code, specifically, the SimpleHTTPRequestHandler class is extended to handle HTTP GET requests. It is utilized for serving static files and customizing server behavior.

socketserver Module:

The socketserver module provides a framework for network servers. In this code, the TCPServer class is employed to create a basic TCP server. It simplifies the process of creating custom server classes.

os Module:

The os module provides a way to interact with the operating system. It is used in this code to manipulate the current working directory (os.chdir) based on the server's script path.

threading Module:

The threading module provides a way to run multiple threads concurrently. In this code, it is used to spawn a separate thread (cthread) for running the HTTP server concurrently with the main thread.

Algorithm

1. **Initialization:**

- Open a log file ("logs.txt") for appending incoming requests.
- Initialize global variables uname and passwd to store username and password.

2. **Define Server Handler Class:**

- Create a class GetHandler extending http.server. SimpleHTTPRequestHandler.
- Override the do_GET method to intercept incoming GET requests.

3. **Authorization Function:**

- Define the authorize function, which accepts the requested path (st), server path (PATH), and anonymous path (ANONYMOUS_PATH) as parameters.
- Converts the path to a list for processing.
- Checks for valid username and password in the provided path.
- If valid, extracts the requested path, and provides access to it.

- If invalid or incomplete credentials, access is denied.

4. **Server Start Function:**

- Define the StartServer function: Sets the server's host to "127.0.0.1" and port to 1515.
- Sets the handler to GetHandler.
- Changes the current working directory to the script path (SCRIPT_PATH).
- Creates a socketserver.TCPServer instance and starts serving indefinitely.

5. **Server Execution:**

- Start the server, and handle potential interruptions, such as a KeyboardInterrupt.
- On interruption, shutdown the server gracefully.

6. **Main Thread Execution:**

- Initialize variables, and print a message indicating server startup.
- If executed as the main script:
- Initialize the necessary variables.
- Spawn a separate thread (cthread) to start the server concurrently.

Front-End Access

1. **HTML Structure:**

- Declare the HTML5 document type.
- Specify the character set and viewport configuration for responsiveness.
- Include the Tailwind CSS framework for styling.
- Import the jQuery library for simplified JavaScript handling.

2. **Form Setup:**
- Begin the HTML body with a form element identified as "loginform."

User Interface Section:

- Introduce a styled section for the user interface, utilizing Tailwind CSS classes.
- Establish a container for content with specified padding and margin.

Header Content:

- Create a flexible column for text-centered content.
- Include a heading conveying the server's purpose and authorship.

User Input Section:

- Design a flexible row for user input, adjusting styles based on screen size.
- Define input fields for username and password, with appropriate labels.

JavaScript Interaction:

- Incorporate a JavaScript script within the HTML body.
- Capture user input for both username and password on button click.
- Assemble the input data for subsequent transmission to the server.
- Execute the runPyScript function to send a POST request to "/ftpserver.py" with the user credentials.

Server Communication Function:

- Define the runPyScript function in JavaScript.

- Utilize jQuery's $.ajax to handle asynchronous server communication.
- Specify the type of request as POST, the target URL as "/ftpserver.py," and data as the assembled input.
- Synchronize the request by setting async to false.
- Capture the result for further processing.

<u>Code</u>

<u>Python Code for FTP Server</u>

```python
# Import necessary modules
import http.server
import socketserver
import os
import threading

# Define constants for username, password, and file paths
USERNAME = "pooja"
PASSWORD = "pooja"
PATH = r"C:\Users\pooja\Pooja_FTP_Server_Folder"
SCRIPT_PATH =
r"C:\Users\pooja\Simple_FTP_Server\Simple_FTP_Server"
ANONYMOUS_PATH = os.path.join(SCRIPT_PATH, "Error")

# Open a log file in append mode
f = open("logs.txt", "a")

# Initialize global variables for username and password
uname = ""
passwd = ""
```

```python
# Function to initialize variables
def initvars():
    global uname
    global passwd
    uname = ""
    passwd = ""

# Print the server's file path
print(f"PATH : {PATH}")

# Custom handler class for handling GET requests
class GetHandler(http.server.SimpleHTTPRequestHandler):
    def do_GET(self):
        # Print the incoming request path
        print("printing request : ", self.path)
        # Log the request in the file
        st = str(self.path)
        f.write(st + "\n")
        # Authorize and process the request
        authorize(st, PATH, ANONYMOUS_PATH)
        http.server.SimpleHTTPRequestHandler.do_GET(self)

# Function to authorize and process the request
def authorize(st, PATH, ANONYMOUS_PATH):
    li = list(st)
    global uname
    global passwd

    print("uname is ", uname)
    print("passwd is ", passwd)
    # Check if provided username and password match the
constants
    if uname == USERNAME and passwd == PASSWORD:
        ind1 = li.index("/")
        add_path = st[ind1:]
        new_path = os.path.join(PATH, add_path)
        print("Providing access to : ", new_path)

    else:
        # If username and password not provided in the
path, extract and validate
        if "username=" in st and "passwd=" in st:
            ind1 = li.index("=")
            ind2 = li.index("&")
            uname = st[ind1 + 1:ind2]
            print("Received username : ", uname)

            li = [li[i] for i in range(ind2, len(st))]
            ind1 = li.index("=")

            # Extract and concatenate characters to form
the password
            for i in range(ind1 + 1, len(li)):
                passwd = passwd + li[i]

            print("Received password : ", passwd)

            # Change the current working directory based
on the authentication
            if uname == USERNAME and passwd == PASSWORD:
                os.chdir(PATH)
            else:
                os.chdir(ANONYMOUS_PATH)
        else:
            # Access is prohibited if username and
password are not provided
            print("Access prohibited")
            os.chdir(SCRIPT_PATH)

# Function to start the server
def StartServer():
    print("Server is up")
    # Set the server's host and port
    host = "127.0.0.1"
    port = 1515
    handler = GetHandler
    # Change the current working directory to the script
path
    os.chdir(SCRIPT_PATH)

    # Create an instance of the server
    http_server = socketserver.TCPServer((host, port),
handler)

    try:
        # Start serving indefinitely
        http_server.serve_forever()
    except KeyboardInterrupt:
        # Shutdown the server gracefully on
KeyboardInterrupt
        print("Server is shutting down")
        http_server.shutdown()
        http_server.server_close()
        print("Server has stopped")
```

```
# Print a message indicating server startup
print("Server starting")

# Initialize variables
initvars()

# If executed as the main script, spawn a separate thread
to start the server concurrently
if __name__ == '__main__':
    cthread = threading.Thread(target=StartServer)
    cthread.daemon = True
    cthread.start()

# Handle potential keyboard interruptions in the main
thread
try:
    cthread.join()
except KeyboardInterrupt:
    print("Main thread received KeyboardInterrupt.
Exiting.")
```

<u>HTML, CSS, Javascript (with JQuery and Ajax)</u>

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
```

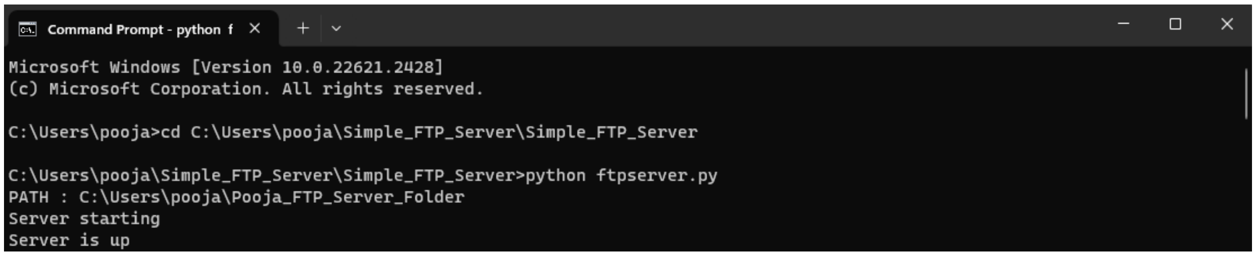
```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <link
href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.
css" rel="stylesheet">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/j
query.min.js"></script>
  <title>Pooja's FTP Server</title>
</head>

<body>


  <form id="loginform">


    <section class="text-gray-600 body-font">
      <div class="container px-5 py-24 mx-auto">
        <div class="flex flex-col text-center w-full
mb-12">
          <h1 class="sm:text-3xl text-2xl font-medium
title-font mb-4 text-gray-900">A Simple FTP Server By
Team 2
          </h1>

        </div>
        <div
          class="flex lg:w-2/3 w-full sm:flex-row
flex-col mx-auto px-8 sm:space-x-4 sm:space-y-0 space-y-4
sm:px-0 items-end">
```

```
          <div class="relative flex-grow w-full">
            <label for="full-name" class="leading-7
text-sm text-gray-600">User Name</label>




            <input id="username" type="text"
name="username" hint="user"
              class="w-full bg-gray-100 bg-opacity-50
rounded border border-gray-300 focus:border-green-500
focus:bg-transparent focus:ring-2 focus:ring-green-200
text-base outline-none text-gray-700 py-1 px-3 leading-8
transition-colors duration-200 ease-in-out">

          </div>
          <div class="relative flex-grow w-full">
            <label for="email" class="leading-7 text-sm
text-gray-600">Password</label>

            <input id="password" type="password"
name="passwd" hint="****"
              class="w-full bg-gray-100 bg-opacity-50
rounded border border-gray-300 focus:border-green-500
focus:bg-transparent focus:ring-2 focus:ring-green-200
text-base outline-none text-gray-700 py-1 px-3 leading-8
transition-colors duration-200 ease-in-out">

          </div>


          <input
```

```
            class="text-white bg-green-500 border-0 py-2 
px-8 focus:outline-none hover:bg-green-600 rounded 
text-lg"
            id="submit" type="submit" value="Login">


        </div>
      </div>
    </section>
  </form>

</body>

<script>

  var user = $('#username').val();
  var passwd = $('#password').val();
  $('#submit').click(function () {
    //alert("button clicked"+user);
    datatosend = user + ";" + passwd;
    result = runPyScript(datatosend);
    //console.log('Got back ' + result);
  });




  function runPyScript(input) {
    var jqXHR = $.ajax({
      type: "POST",
      url: "/ftpserver.py",
      async: false,
      data: { mydata: input }
    });
  }


</script>

</html>
```

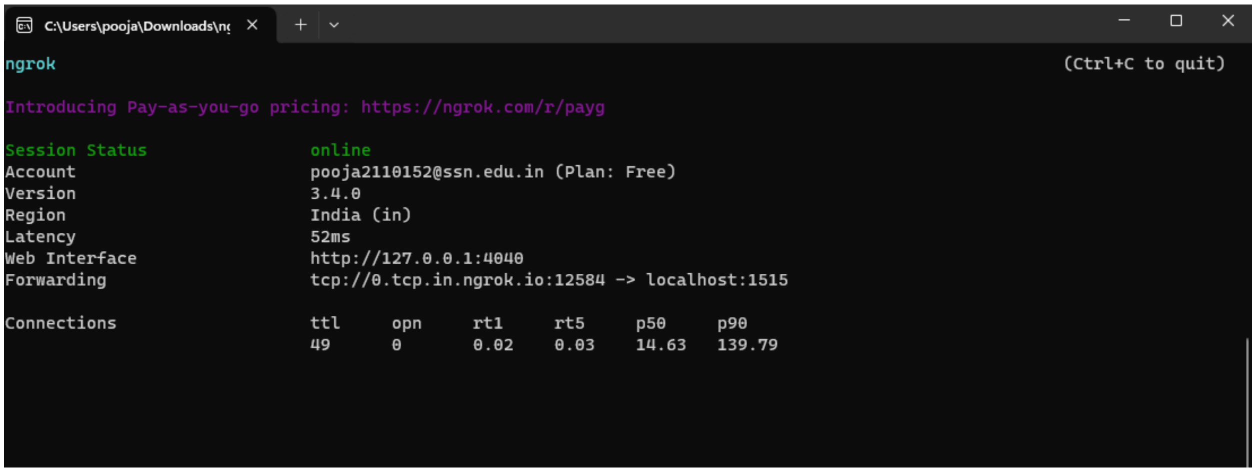
Output

Run Python script to start the server



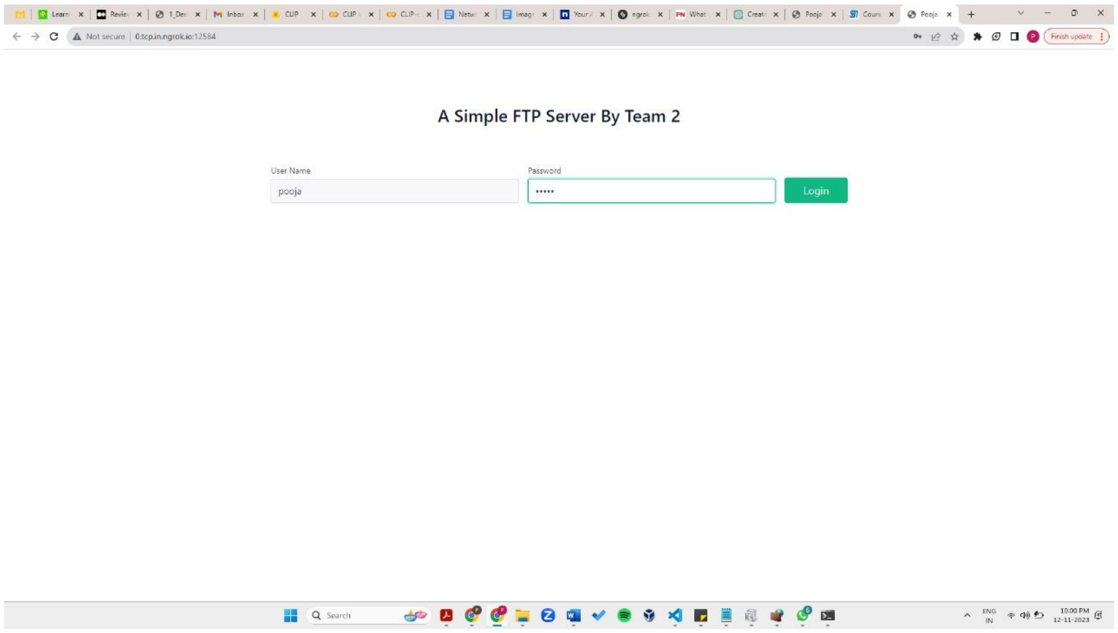
```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pooja>cd C:\Users\pooja\Simple_FTP_Server\Simple_FTP_Server

C:\Users\pooja\Simple_FTP_Server\Simple_FTP_Server>python ftpserver.py
PATH : C:\Users\pooja\Pooja_FTP_Server_Folder
Server starting
Server is up
```

Initialize and begin an ngrok session



```
ngrok                                                          (Ctrl+C to quit)

Introducing Pay-as-you-go pricing: https://ngrok.com/r/payg

Session Status                online
Account                       pooja2110152@ssn.edu.in (Plan: Free)
Version                       3.4.0
Region                        India (in)
Latency                       52ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    tcp://0.tcp.in.ngrok.io:12584 -> localhost:1515

Connections                   ttl     opn     rt1     rt5     p50     p90
                              49      0       0.02    0.03    14.63   139.79
```

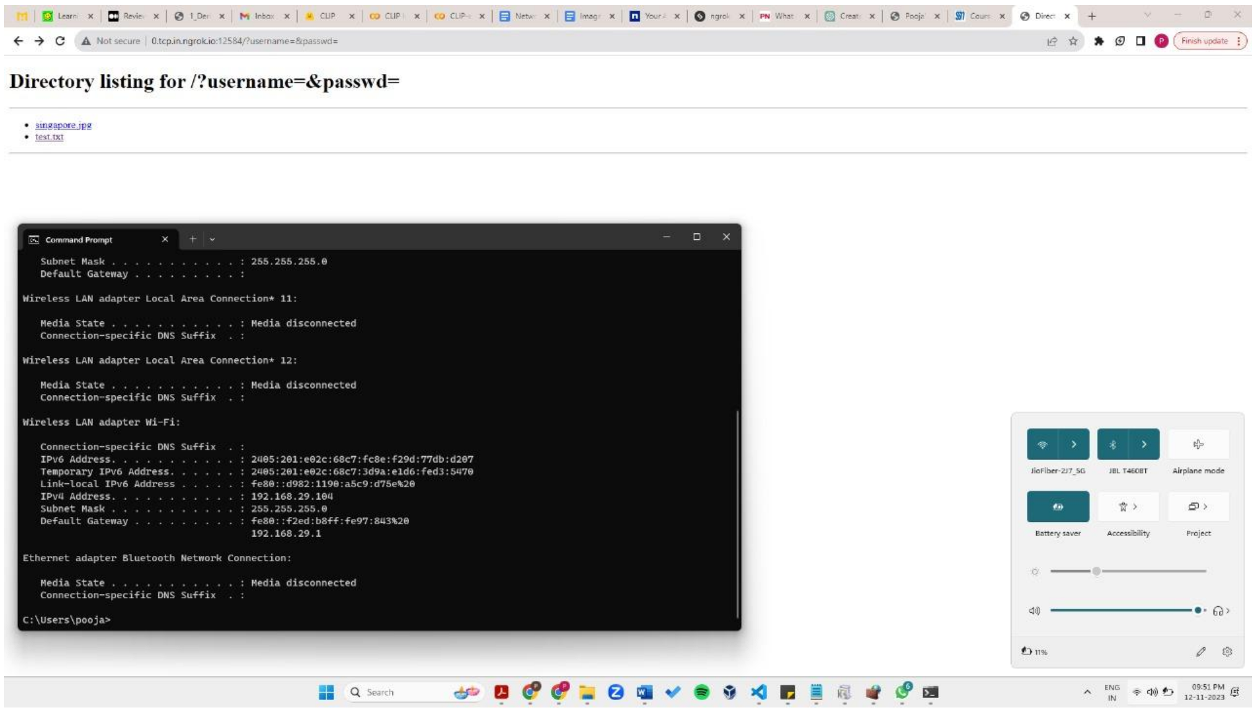
Server administrator login


A Simple FTP Server By Team 2

User Name: pooja

Password: •••••

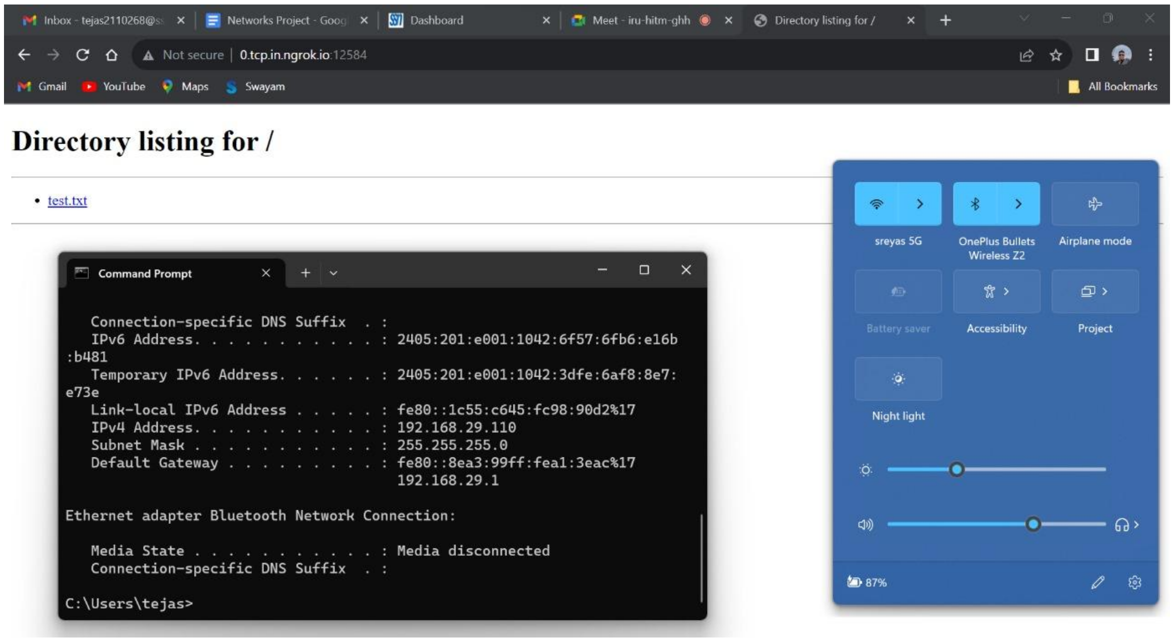
Logged in screen on the server administrator’s side


Directory listing for /?username=&passwd=

- singapore.jpg
- test.txt

```
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:201:e02c:68c7:fc8e:f29d:77db:d207
   Temporary IPv6 Address. . . . . . : 2405:201:e02c:68c7:3d9a:e1d6:fed3:5470
   Link-local IPv6 Address . . . . . : fe80::d982:1190:a5c9:d75e%20
   IPv4 Address. . . . . . . . . . . : 192.168.29.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::f2ed:b8ff:fe97:843%20
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\pooja>
```

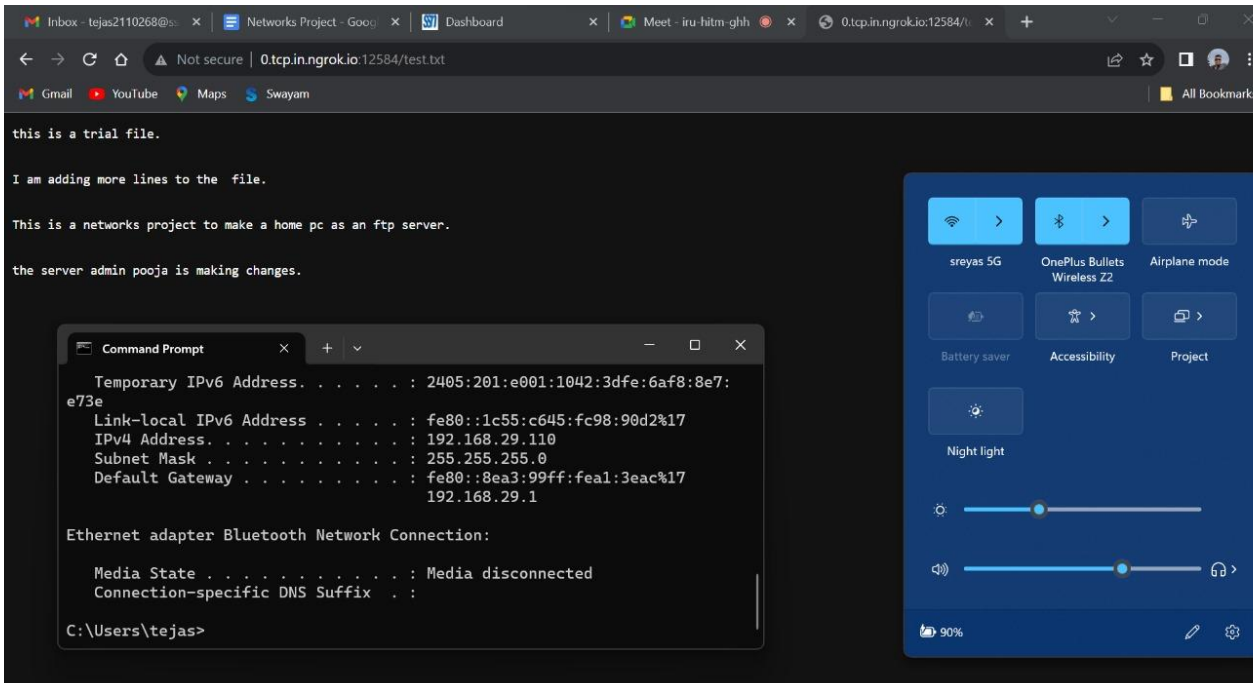
Access files from outside the network


Directory listing for /

- test.txt

```
   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:201:e001:1042:6f57:6fb6:e16b
:b481
   Temporary IPv6 Address. . . . . . : 2405:201:e001:1042:3dfe:6af8:8e7:
e73e
   Link-local IPv6 Address . . . . . : fe80::1c55:c645:fc98:90d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::8ea3:99ff:fea1:3eac%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\tejas>
```

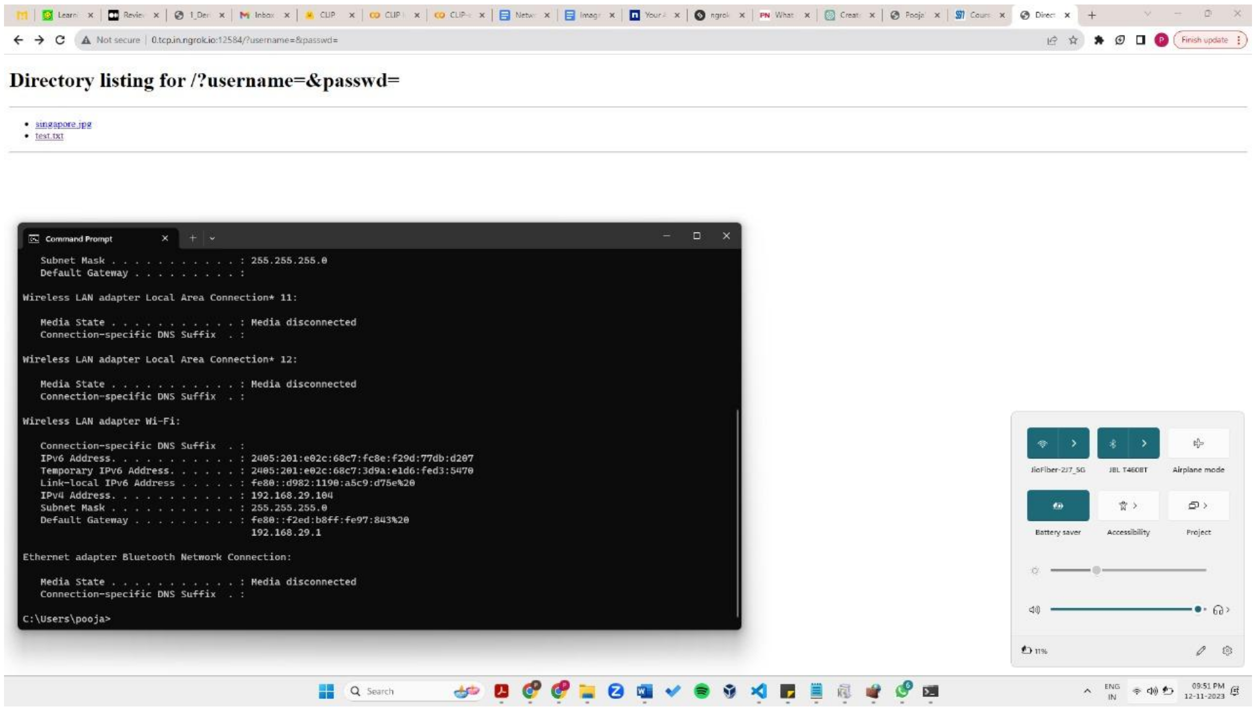
Contents of the text file from the client side



```
this is a trial file.

I am adding more lines to the  file.

This is a networks project to make a home pc as an ftp server.

the server admin pooja is making changes.
```

```
   Temporary IPv6 Address. . . . . . : 2405:201:e001:1042:3dfe:6af8:8e7:
e73e
   Link-local IPv6 Address . . . . . : fe80::1c55:c645:fc98:90d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::8ea3:99ff:fea1:3eac%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\tejas>
```

Addition of new files from the server side

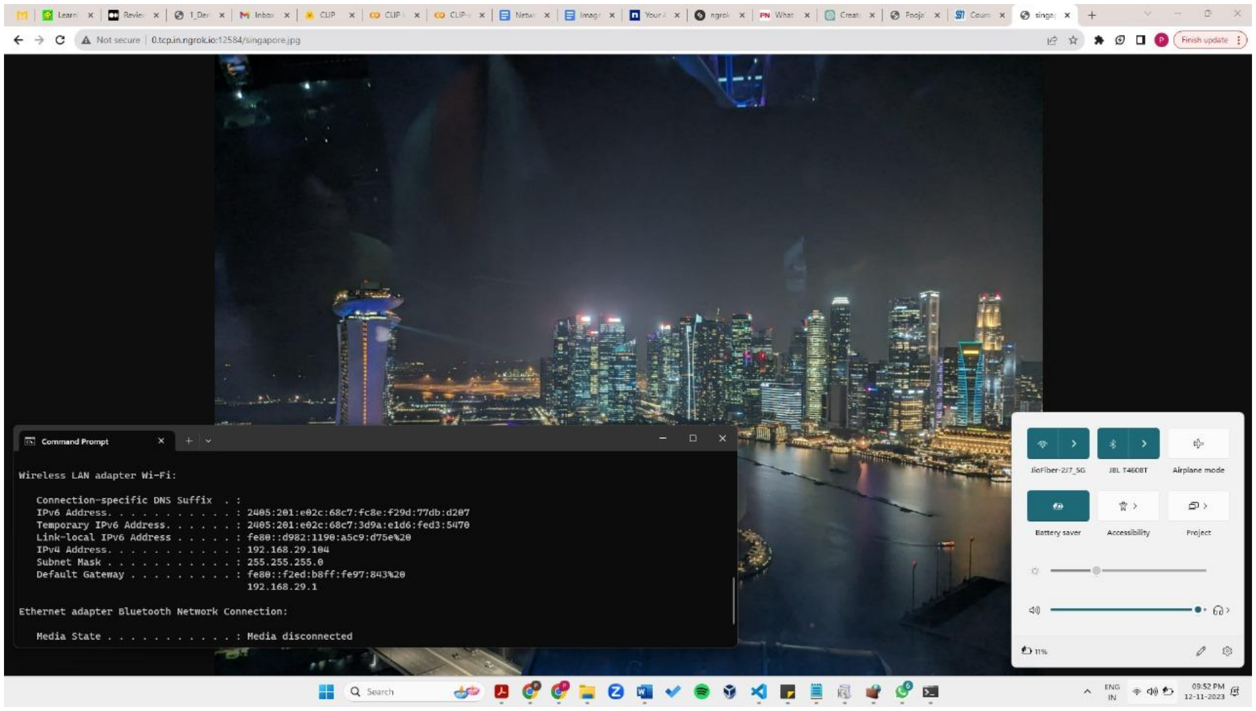


0.tcp.in.ngrok.io:12584/?username=&passwd=

Directory listing for /?username=&passwd=

- singapore.jpg
- test.txt

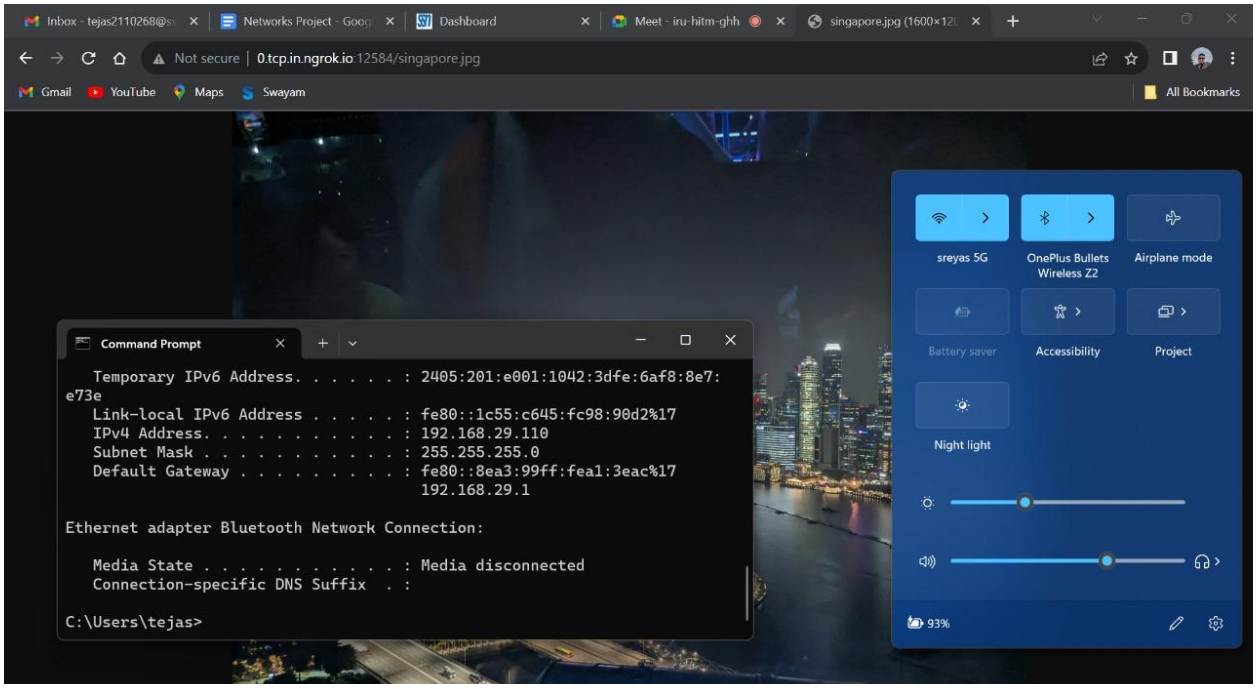
```
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:201:e02c:68c7:fc8e:f29d:77db:d207
   Temporary IPv6 Address. . . . . . : 2405:201:e02c:68c7:3d9a:e1d6:fed3:5470
   Link-local IPv6 Address . . . . . : fe80::d982:1190:a5c9:d75e%20
   IPv4 Address. . . . . . . . . . . : 192.168.29.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::f2ed:b8ff:fe97:843%20
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\pooja>
```



0.tcp.in.ngrok.io:12584/singapore.jpg

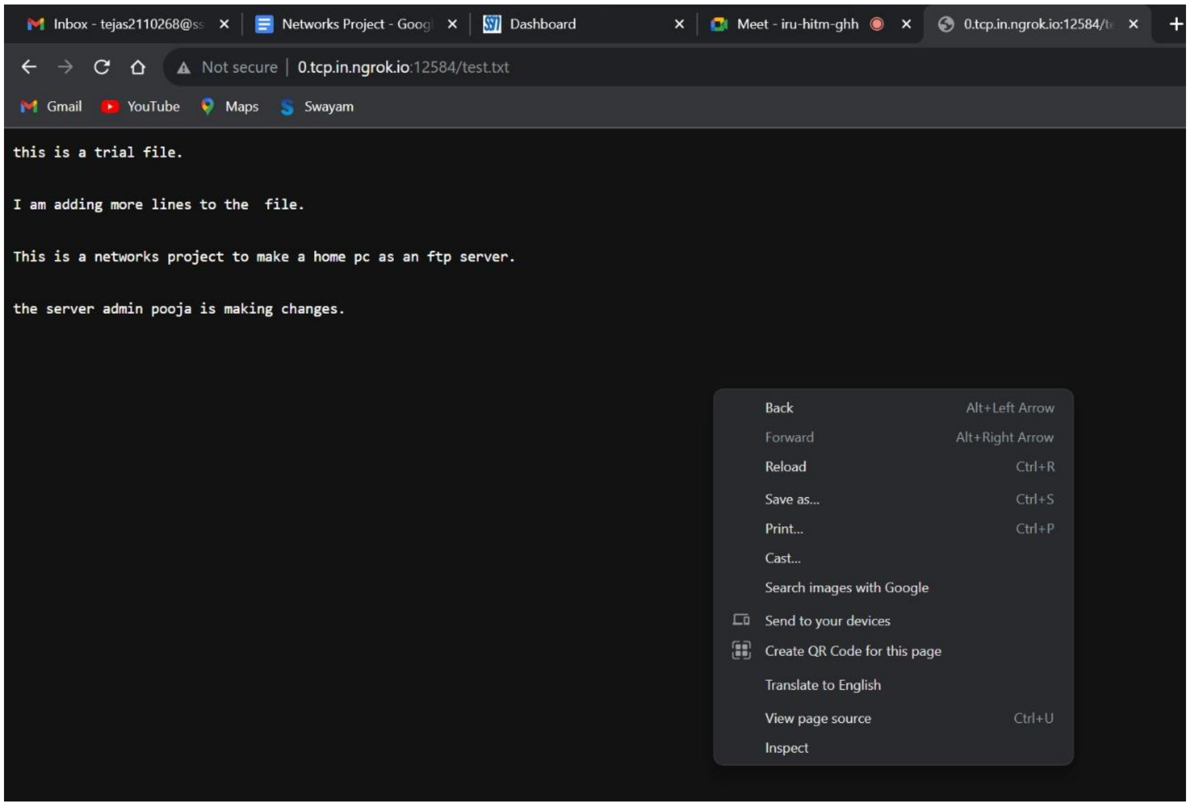
```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:201:e02c:68c7:fc8e:f29d:77db:d207
   Temporary IPv6 Address. . . . . . : 2405:201:e02c:68c7:3d9a:e1d6:fed3:5470
   Link-local IPv6 Address . . . . . : fe80::d982:1190:a5c9:d75e%20
   IPv4 Address. . . . . . . . . . . : 192.168.29.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::f2ed:b8ff:fe97:843%20
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
```

Accessing the new files from the client side

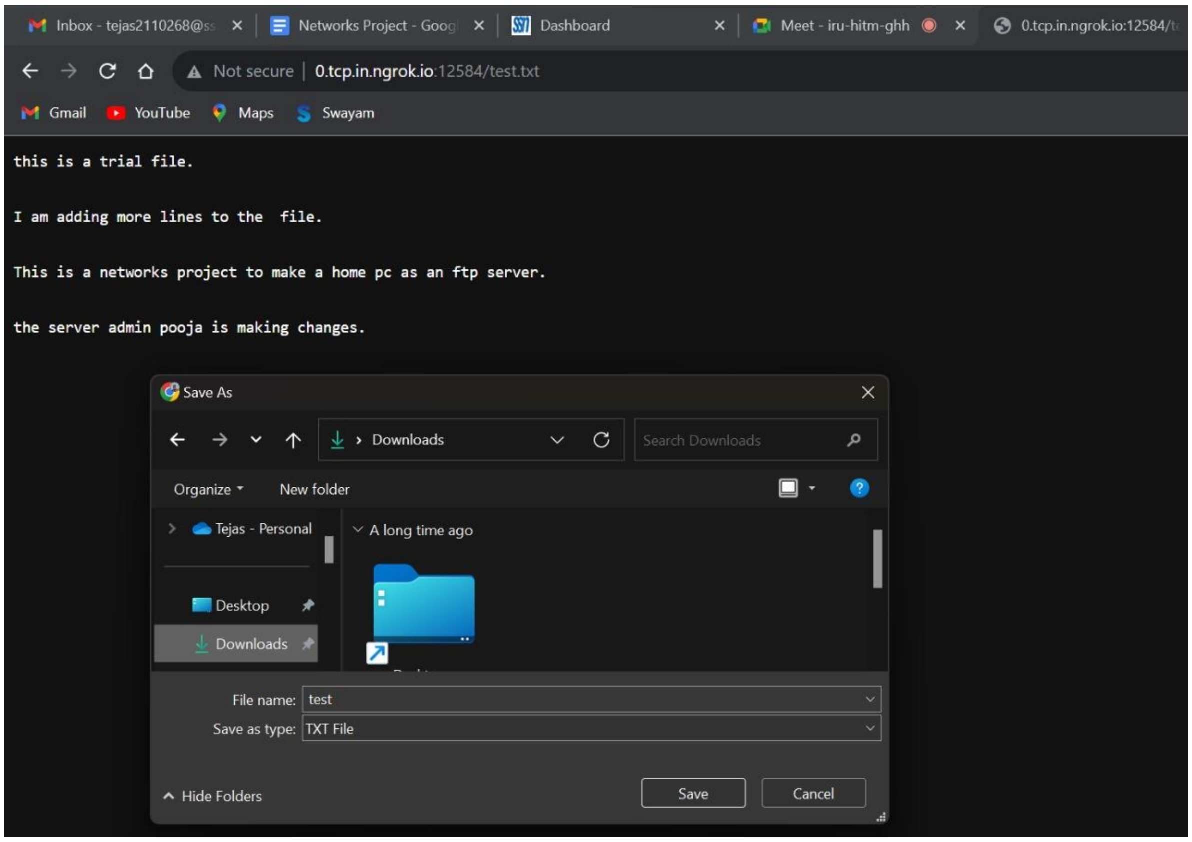


```
   Temporary IPv6 Address. . . . . . : 2405:201:e001:1042:3dfe:6af8:8e7:
e73e
   Link-local IPv6 Address . . . . . : fe80::1c55:c645:fc98:90d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::8ea3:99ff:fea1:3eac%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\tejas>
```

Downloading files from the client side

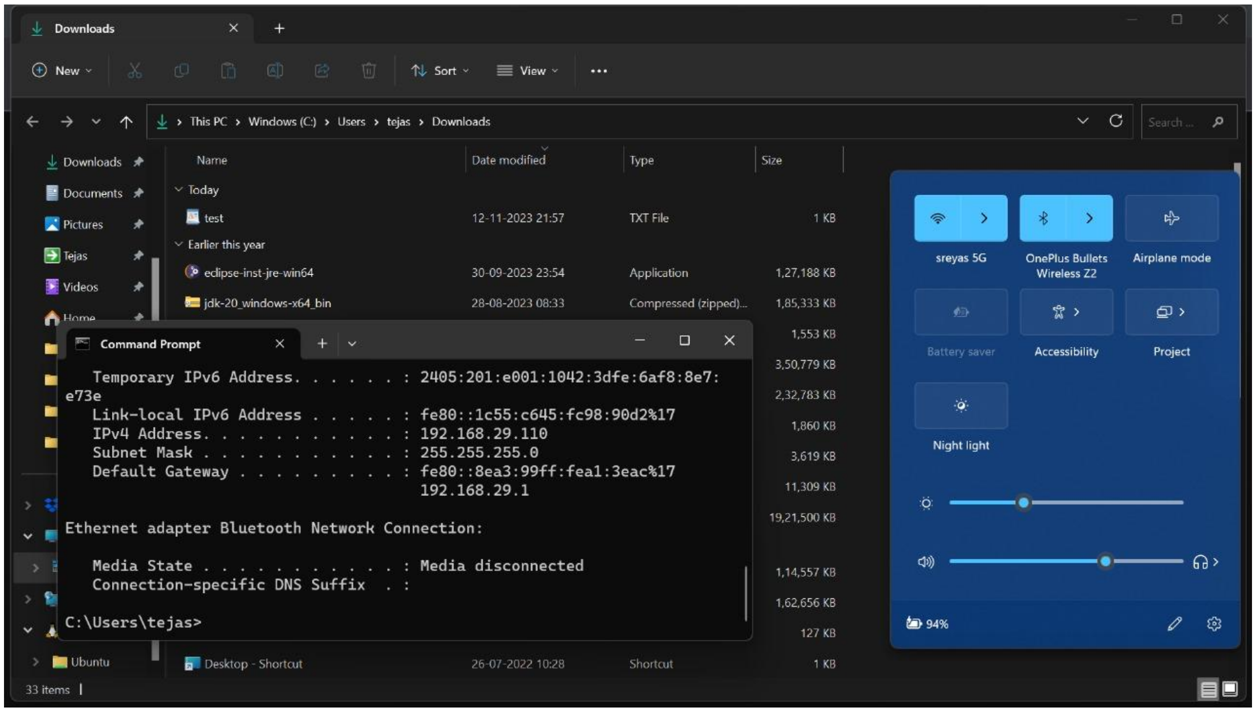


```
this is a trial file.

I am adding more lines to the  file.

This is a networks project to make a home pc as an ftp server.

the server admin pooja is making changes.
```



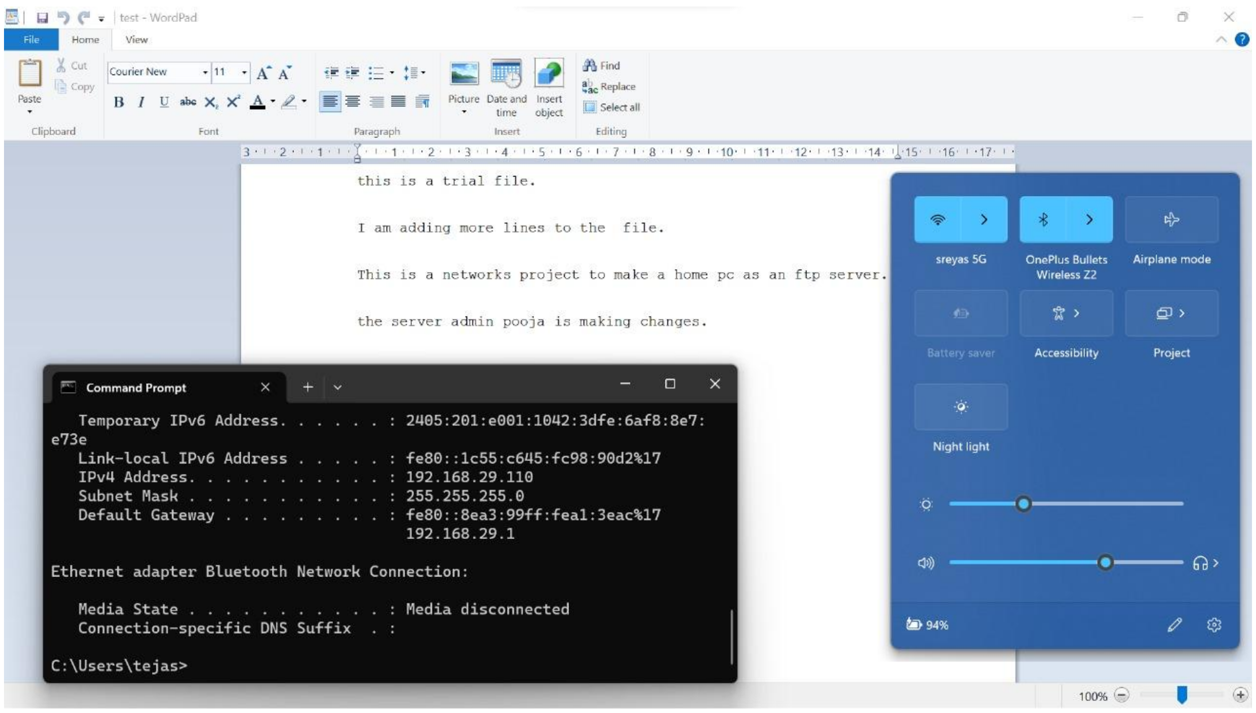
```
this is a trial file.

I am adding more lines to the  file.

This is a networks project to make a home pc as an ftp server.

the server admin pooja is making changes.
```

File name: test
Save as type: TXT File



Name	Date modified	Type	Size
Today			
test	12-11-2023 21:57	TXT File	1 KB
Earlier this year			
eclipse-inst-jre-win64	30-09-2023 23:54	Application	1,27,188 KB
jdk-20_windows-x64_bin	28-08-2023 08:33	Compressed (zipped)...	1,85,333 KB
			1,553 KB
			3,50,779 KB
			2,32,783 KB
			1,860 KB
			3,619 KB
			11,309 KB
			19,21,500 KB
			1,14,557 KB
			1,62,656 KB
			127 KB
Desktop - Shortcut	26-07-2022 10:28	Shortcut	1 KB

```
   Temporary IPv6 Address. . . . . . : 2405:201:e001:1042:3dfe:6af8:8e7:
e73e
   Link-local IPv6 Address . . . . . : fe80::1c55:c645:fc98:90d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::8ea3:99ff:fea1:3eac%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\tejas>
```



```
this is a trial file.

I am adding more lines to the  file.

This is a networks project to make a home pc as an ftp server.

the server admin pooja is making changes.
```

Command Prompt

```
   Temporary IPv6 Address. . . . . . : 2405:201:e001:1042:3dfe:6af8:8e7:
e73e
   Link-local IPv6 Address . . . . . : fe80::1c55:c645:fc98:90d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::8ea3:99ff:fea1:3eac%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\tejas>
```

ReadMe Link

https://github.com/PoojaPremnath2003/NetworksProject_TeamNo2/blob/main/README.md

GitHub Link

https://github.com/PoojaPremnath2003/NetworksProject_TeamNo2

Learning Outcomes

- A comprehensive understanding was gained regarding making one's home PC an FTP server. Challenges included navigating issues associated with Network Address Translation (NAT), coping with limitations imposed by Internet Service Providers (ISPs) on port forwarding, and addressing hindrances posed by stringent firewall configurations.

- In overcoming these challenges, proficiency was developed in utilizing ngrok as a solution. Ngrok's role in creating secure tunnels for external access to locally hosted FTP servers was explored and implemented.
- A Python script for the FTP server was developed, incorporating features such as user authentication and socket programming to establish connectivity. Front-end development involved creating a simple yet functional HTML interface. This interface, powered by JavaScript, jQuery, and AJAX, facilitated user interaction, capturing input, and enabling communication with the Python server script.
- The project shed light on the intricacies of network configurations, emphasizing the impacts of NAT, port forwarding, and firewall settings on networked applications.
- The FTP server on the home PC was able to successfully transfer files to other users, even those outside the home PC’s network.