**Assignment 8**

**WRC078BEI048**

1. **In the Programming language of you choice to write the web app that allows you to upload files.**

**Ans:**

App.py

from flask import Flask, request, redirect, url\_for, render\_template

from werkzeug.utils import secure\_filename

import os

app = Flask(\_\_name\_\_)

# Configurations

UPLOAD\_FOLDER = 'uploads'

ALLOWED\_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

if not os.path.exists(UPLOAD\_FOLDER):

    os.makedirs(UPLOAD\_FOLDER)

def allowed\_file(filename):

    return '.' in filename and \

           filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

@app.route('/')

def upload\_form():

    return render\_template('upload.html')

@app.route('/upload', methods=['POST'])

def upload\_file():

    if 'file' not in request.files:

        return redirect(request.url)

    file = request.files['file']

    if file.filename == '':

        return redirect(request.url)

    if file and allowed\_file(file.filename):

        filename = secure\_filename(file.filename)

        file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

        return 'File successfully uploaded'

    else:

        return 'Allowed file types are txt, pdf, png, jpg, jpeg, gif'

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

upload.html

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>Upload File</title>

  </head>

  <body>

    <div class="container">

      <h2>Upload File</h2>

      <form method="POST" action="/upload" enctype="multipart/form-data">

        <div>

          <input type="file" name="file">

        </div>

        <div>

          <input type="submit" value="Upload">

        </div>

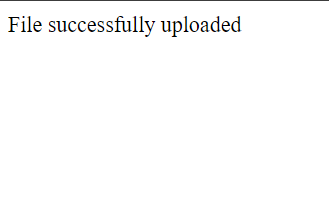
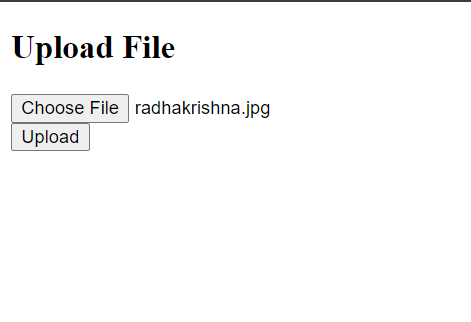
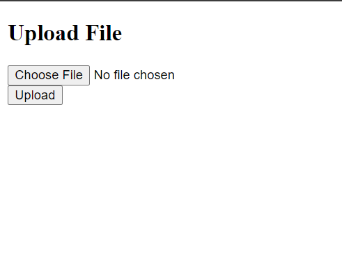
      </form>

    </div>

  </body>

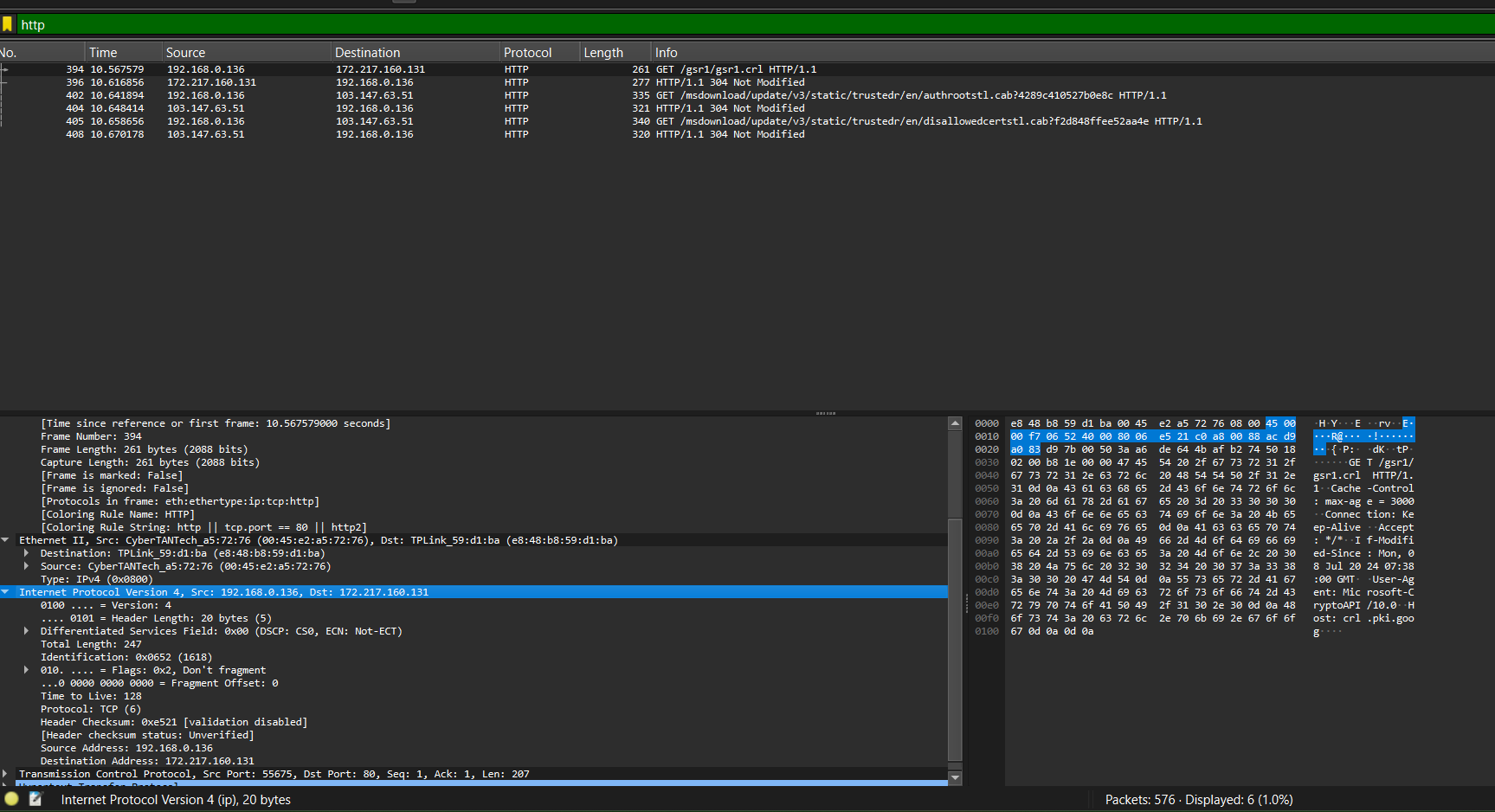
</html>

Using this python and html code I built a web app that allows us to upload a file.

****

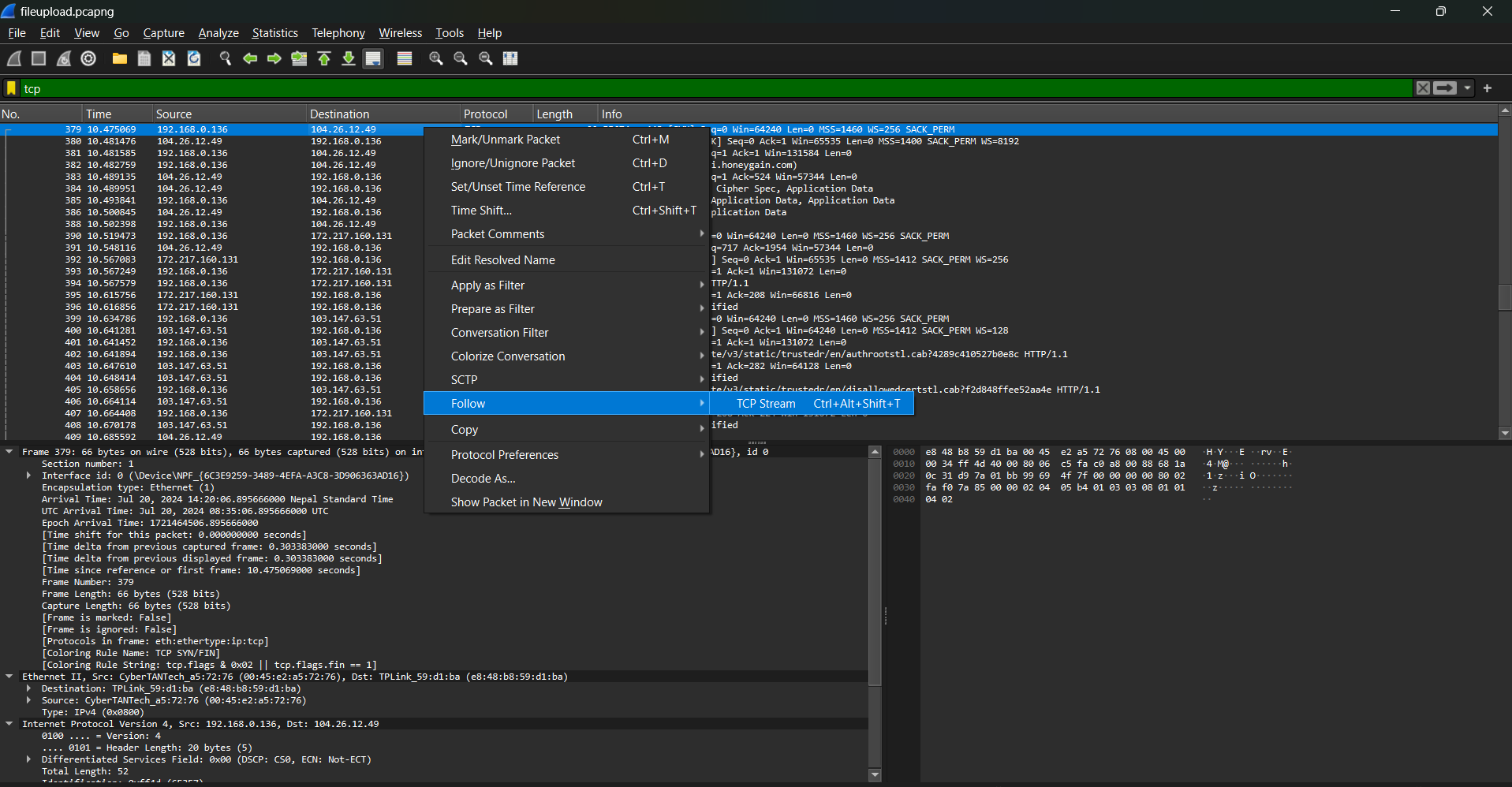
1. **Capture the traffic in wireshark while uploading the file.**

**Ans:**



**3. Follow the TCP stream and explain connection initiation, connection maintenance and connection termination**

**Ans:**

****

**1. Connection Initiation**

**Objective:** To identify the TCP handshake that establishes the connection between the client and server.

**Steps:**

1. **Find the SYN Packet**:
   * Look for a packet with the SYN flag set. This is typically the first packet sent by the client to initiate the connection.
   * **Example Packet Details**:
     + **Flags**: SYN
     + **Sequence Number**: X
2. **Find the SYN-ACK Packet**:
   * Find the packet with both SYN and ACK flags set. This is the server’s response to the client’s SYN request.
   * **Example Packet Details**:
     + **Flags**: SYN, ACK
     + **Sequence Number**: Y
     + **Acknowledgment Number**: X + 1 (acknowledges the client’s SYN)
3. **Find the ACK Packet**:
   * Look for the final packet in the handshake with the ACK flag set. This is sent by the client to acknowledge the server’s SYN-ACK.
   * **Example Packet Details**:
     + **Flags**: ACK
     + **Sequence Number**: X + 1
     + **Acknowledgment Number**: Y + 1 (acknowledges the server’s SYN)

**2. Connection Maintenance**

**Objective:** To analyze the data transfer and how TCP maintains the connection.

**Steps:**

1. **Identify Data Packets**:
   * Look at the TCP packets with the PSH (Push) flag set, which indicates data being pushed to the application layer.
   * **Example Packet Details**:
     + **Flags**: PSH, ACK
     + **Sequence Number**: This is the starting byte of the data segment.
     + **Acknowledgment Number**: Indicates the next expected byte from the other side.
2. **Check Flow Control**:
   * Observe the window size in the TCP header to understand how much data the receiver is willing to accept.
   * **Example Field**:
     + **Window Size**: Indicates the amount of buffer space available.
3. **Monitor Congestion Control**:
   * Look for signs of congestion control mechanisms like Slow Start or Congestion Avoidance in TCP headers, though detailed congestion control analysis may require deeper inspection of the TCP behavior over time.
4. **Check Keep-Alive Packets (Optional)**:
   * If configured, check for occasional TCP keep-alive packets sent to ensure the connection remains active.
   * **Example Packet Details**:
     + **Flags**: ACK (with no payload)

**3. Connection Termination**

**Objective:** To identify the sequence of packets that close the TCP connection.

**Steps:**

1. **Find the FIN Packet**:
   * Look for a packet with the FIN flag set. This indicates that one side wants to close the connection.
   * **Example Packet Details**:
     + **Flags**: FIN
     + **Sequence Number**: U
2. **Find the ACK Packet**:
   * Locate the packet that acknowledges the FIN request from the other side.
   * **Example Packet Details**:
     + **Flags**: ACK
     + **Acknowledgment Number**: U + 1 (acknowledges the FIN)
3. **Find the Second FIN Packet**:
   * The side that acknowledged the initial FIN now sends its own FIN packet to close its side of the connection.
   * **Example Packet Details**:
     + **Flags**: FIN
     + **Sequence Number**: W
4. **Find the Final ACK Packet**:
   * Look for the final ACK packet that acknowledges the second FIN request.
   * **Example Packet Details**:
     + **Flags**: ACK
     + **Acknowledgment Number**: W + 1

**How to View This in Wireshark:**

1. **Start Capture**: Begin capturing traffic on the appropriate network interface.
2. **Follow TCP Stream**:
   * Right-click on a TCP packet related to the connection you want to analyze.
   * Select "Follow" > "TCP Stream" to view the entire conversation.
3. **Examine Packets**:
   * Use the filters and packet details to find SYN, SYN-ACK, and ACK packets for initiation.
   * Look for data packets, flow control, and keep-alive packets during maintenance.
   * Identify FIN and ACK packets for termination.