Name of Student: Pushkar Sane	
Roll Number: 45	Lab Assignment Number: 6
Title of Lab Assignment: 1) Create a HTTP Server and serve HTML,CSV,JSON and PDF Files. 2) Create a HTTP Server and stream a video file using piping. 3) Develop 3 HTML Web Pages for college home page (write about college & dept), about me, contact. Create a server and render these pages using Routing.	
DOP: 11-10-2023	DOS: 13-10-2023

CO Mapped: CO1	PO Mapped: PO3, PO5, PSO1, PSO2	Signature:

Practical No. 6

Aim:

1. Create a HTTP Server and serve HTML,CSV,JSON and PDF Files.

2. Create a HTTP Server and stream a video file using piping.

3. Develop 3 HTML Web Pages for college home page (write about college & dept), about me, contact. Create a server and render these pages using Routing.

Description:

1. Create an HTTP Server and Serve Various File Types (HTML, CSV, JSON, PDF)

- **HTTP Server:**

  An HTTP (Hypertext Transfer Protocol) server is a software application that handles incoming requests from clients, such as web browsers, and responds with the appropriate content. It listens on a specific port (commonly port 80 for HTTP) and communicates using the HTTP protocol.

- **Content Types and `Content-Type` Header:**

  The `Content-Type` header in an HTTP response specifies the type of data being sent. It helps the client (browser) understand how to interpret the content. For example, HTML files have a `Content-Type` of 'text/html', CSV files are 'text/csv', JSON files are 'application/json', and PDF files are 'application/pdf'.

- **Serving Files:**

  To serve files, the server reads the file's content and sends it as the response. The `ContentType` header is set according to the file's type, and the browser renders the content appropriately.

- **File Streaming:**

  When serving large files, like PDFs, it's more efficient to use streaming. Instead of loading the entire file into memory before sending it, you send chunks of data as they are read. This is memory-efficient and allows users to start viewing or downloading the content more quickly.

- **HTTP Server:**

  Component: The Node.js `http` module is used to create an HTTP server. It listens for incoming HTTP requests and manages the communication.

- **File System Interaction:**

  Component: The `fs` (File System) module in Node.js is used to read files from the server's file system.

- **Content-Type Header:**

  Component: The `Content-Type` header in the HTTP response is set to specify the type of the content being served.

- **Routing:**

  Component: In this basic example, routing is done using conditional logic based on the URL path requested.

- **File Streaming:**

  Component: In the case of large files like PDFs, streaming is achieved using Node.js's stream APIs. This efficiently sends file data in chunks from the server to the client.

2. **Create an HTTP Server and Stream a Video File Using Piping**

   - **HTTP Streaming:**

     HTTP streaming allows media content to be sent over an HTTP connection progressively. It's commonly used for videos and audio files. The server sends the content as it's available, and the client starts rendering or playing it without waiting for the entire file to download.

   - **Content-Disposition Header:**

     The `Content-Disposition` header can be used to suggest how the browser should handle the file. Setting it to 'inline' suggests that the file should be displayed in the browser, while 'attachment' suggests that the file should be downloaded.

- **Piping:**

  In Node.js, piping is a technique to efficiently transfer data from a source to a destination. It's especially useful when dealing with large files. A readable stream (source) is piped into a writable stream (destination), and data flows between them in chunks. This is memory-efficient and avoids loading the entire file into memory.

- **HTTP Streaming:**

  Component: HTTP streaming is the main concept, where data is sent over an HTTP connection progressively.

- **Content-Disposition Header:**

  Component: The `Content-Disposition` header can be used to suggest how the browser should handle the file. In this scenario, it might be set to 'inline' to prompt in-browser rendering.

- **Piping:**

  Component: The `pipe` method from Node.js stream APIs is used to efficiently transfer data from a source (file) to a destination (response). This technique is memory-efficient and suitable for streaming large files.

3. **Develop 3 HTML Web Pages and Serve Them Using Routing:**

   - **HTML Pages:**

     HTML (Hypertext Markup Language) is the standard language for creating web pages. It provides the structure and content of web documents. Browsers interpret HTML to render web pages.

   - **Routing:**

     Routing in web development involves mapping URLs to specific actions or responses on the server. It's essential for handling multiple pages or different functionalities within a web application. Routes determine which code should execute based on the URL requested by the client.

- **Server-Side Rendering:**

  Server-side rendering (SSR) is a technique where web pages are generated dynamically on the server and sent as fully rendered HTML to the client. This is in contrast to client-side rendering (CSR), where most rendering occurs in the browser using JavaScript. SSR is used for SEO optimization and better performance in some cases.

- **Express.js:**

  Express.js is a popular Node.js web application framework that simplifies building web applications and APIs. It provides tools for routing, middleware, and serving static files. In the example, Express is used to define routes and serve HTML pages based on the URL requested.

- **Express.js:**

  Component: Express.js is a Node.js web framework used to simplify routing and handling HTTP requests. It provides routing, middleware, and various tools for building web applications.

- **Routing:**

  Component: Express.js is used to define routes, which map URLs to specific functions or templates (HTML pages) to be served in response to different URL paths.

- **HTML Pages:**

  Component: HTML pages are the actual content that is served to the client. These pages are stored in the 'public' directory and sent to the client based on the requested route.
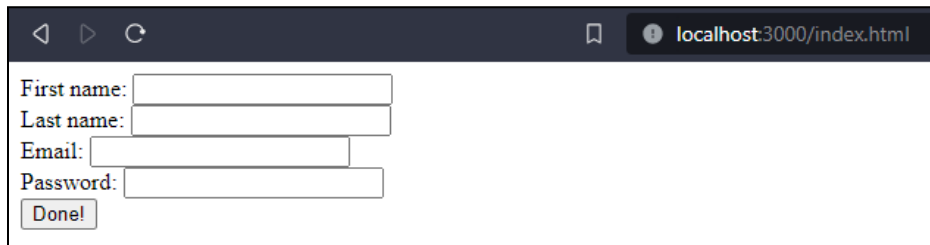
- **Server-Side Rendering:**

  Component: In the Express.js-based server, server-side rendering is the primary technique. It involves dynamically generating web pages on the server and sending fully rendered HTML pages to the client.

1. Create a HTTP Server and serve HTML, CSV, JSON and PDF Files.

server.js

```javascript
const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

   const { url } = req;

   const filePath = path.join(__dirname, 'public', url);

   fs.readFile(filePath, (err, data) => {

      if (err) {

         res.writeHead(404, {'Content-Type': 'text/plain'});

         res.end('File not found');

      } else {

         let contentType = 'text/plain';

         if (filePath.endsWith('.html')) {

            contentType = 'text/html';

         } else if (filePath.endsWith('.csv')) {

            contentType = 'text/csv';

         } else if (filePath.endsWith('.json')) {

            contentType = 'application/json';

         } else if (filePath.endsWith('.pdf')) {

            contentType = 'application/pdf';

         }

         res.writeHead(200, {'Content-Type': contentType });

         res.end(data);

      }

   });

});

const port = 3000;

server.listen(port, () => {

   console.log(`Server is running on http://localhost:${port}`);

});
```

Output:


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS

[Running] node "f:\Pushkar\MCA\Sem-1\WAT\server.js"
Server is running on http://localhost:3000
```
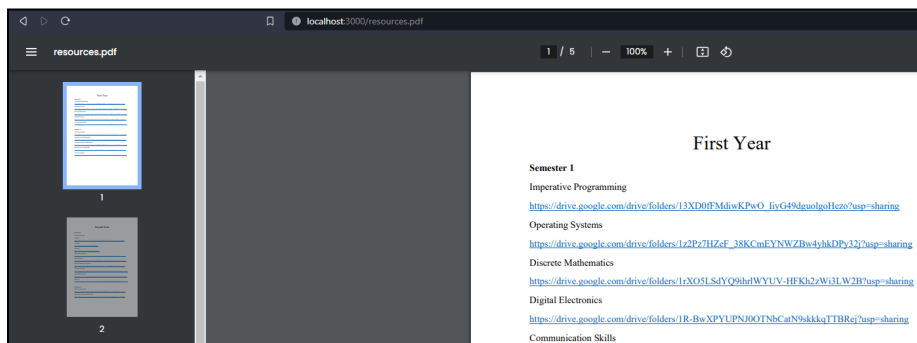
index.html (Serving html file)
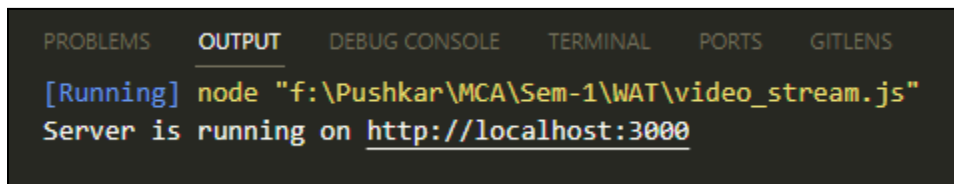
resources.pdf (Serving pdf file)

data.json (Serving as json file)

2. **Create a HTTP Server and stream a video file using piping.**

   videosteam.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const server = http.createServer((req, res) => {
    const videoPath = path.join(__dirname, 'public', 'video.mp4');
    const stat = fs.statSync(videoPath);
    res.writeHead(200, {
        'Content-Type': 'video/mp4',
        'Content-Length': stat.size,
    });
    const videoStream = fs.createReadStream(videoPath);
    videoStream.pipe(res);
});
const port = 3000;
server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

   Output:

video.mp4 (Streaming video using piping).

3. **Develop 3 HTML Web Pages for college home page(write about college & dept), about me, contact. Create a server and render these pages using Routing.**
   index.js

```
<!DOCTYPE html>
<html>
   <head>
      <title>College Home Page</title>
   </head>
   <body>
      <h1>Welcome to Our College</h1>
      <p>We are a leading institution dedicated to providing high-quality education.</p>
   </body>
   <p><a href="about.html">About us</a></p>
   <p><a href="contact.html">Contact us</a></p>
   <p><a href="index.html">Home</a></p>
</html>
```

about.js

```html
<!DOCTYPE html>
<html>
  <head>
    <title>About Me</title>
  </head>
  <body>
    <h1>About Me</h1>
        <p>I am a student at this college, and I love programming and web development.</p>
  </body>
  <p><a href="index.html">Home</a></p>
</html>
```
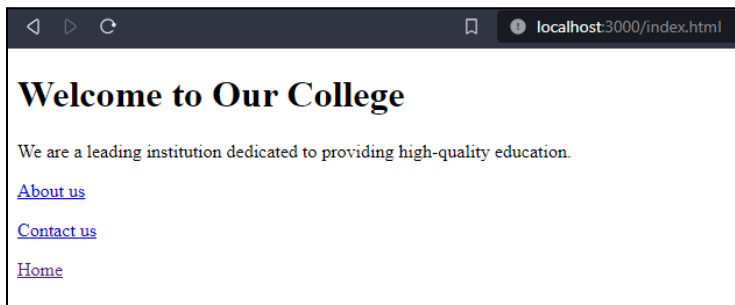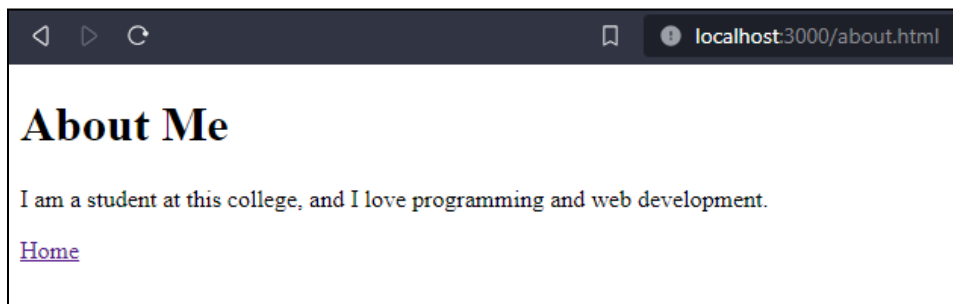
contact.js

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Contact</title>
  </head>
  <body>
    <h1>Contact Us</h1>
    <p>Feel free to get in touch with us by email or phone.</p>
  </body>
  <p><a href="index.html">Home</a></p>
</html>
```

Output:


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS

[Running] node "f:\Pushkar\MCA\Sem-1\WAT\server.js"
Server is running on http://localhost:3000
```

Index.html (Rendering Page 1)

localhost:3000/index.html

Welcome to Our College

We are a leading institution dedicated to providing high-quality education.

About us

Contact us

Home

About.html (Rendering Page 2)

localhost:3000/about.html

About Me

I am a student at this college, and I love programming and web development.

Home

Contact.html (Rendering Page 3)

localhost:3000/contact.html

Contact Us

Feel free to get in touch with us by email or phone.

Home
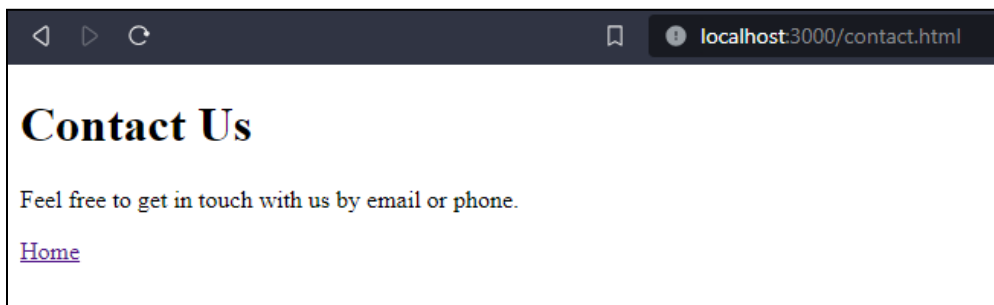
Conclusion:

We learned how to handle various file types (HTML, CSV, JSON, PDF) and ensure that the appropriate Content-Type and Content-Disposition headers are set, allowing for both file downloads and in-browser content rendering, depending on the file type.