1DV701 Assignment 2 - Web Server

In this assignment,I did (hr222rs) 60% of the work Umar (um222db) did 40%, We developed a multi-threaded web server in Java, designed to serve static content using the HTTP protocol while efficiently managing multiple client connections through thread pooling. The server listens on port 2800 and serves HTML and PNG files from the /public directory. It supports both GET and POST requests and implements robust security measures to prevent unauthorized access and directory traversal attacks.

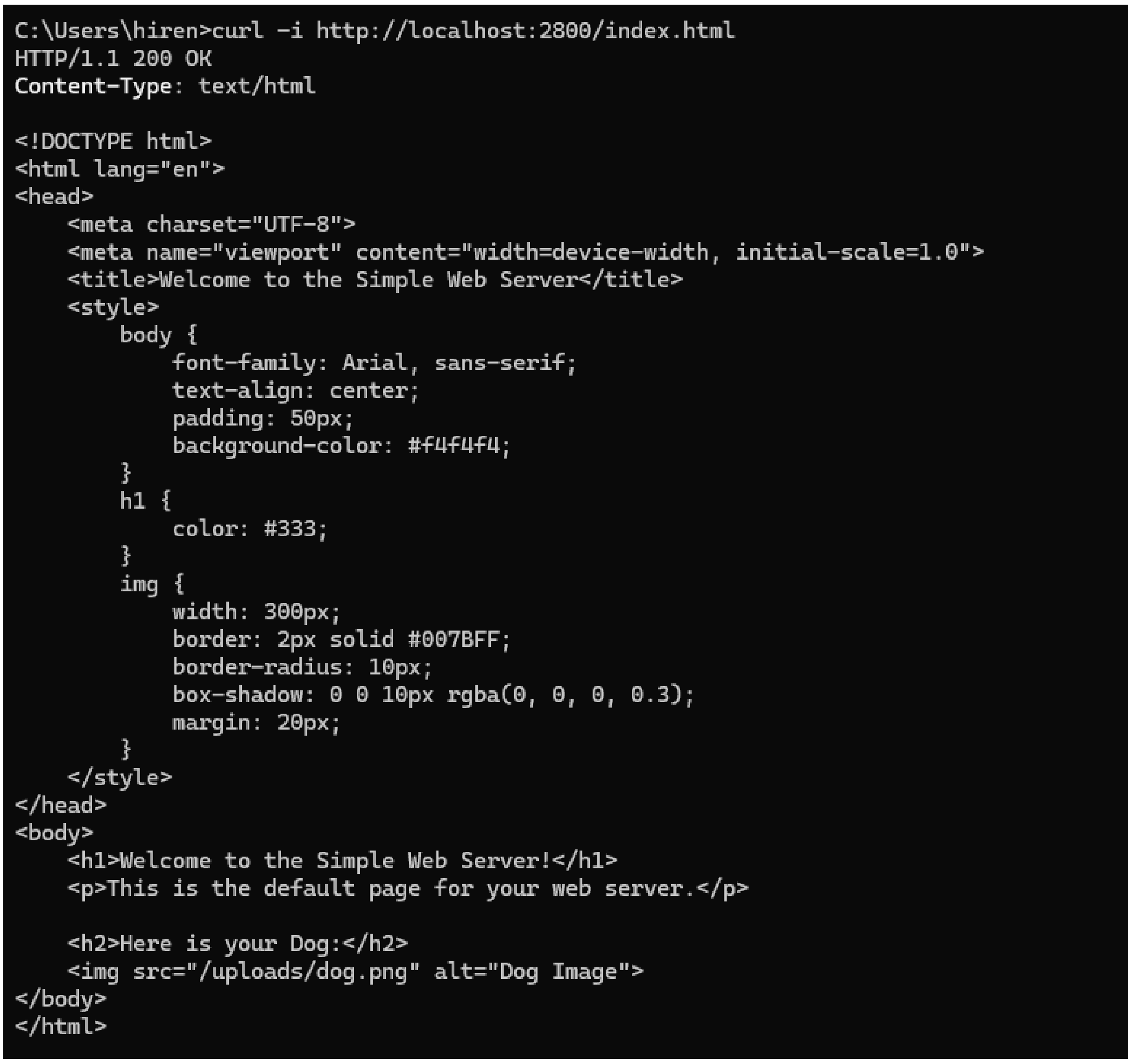
The server ensures that users cannot access files outside the designated directory by sanitizing file paths and validating requested resources. It correctly handles various HTTP response codes, including 200 OK for successful requests, 302 Found for redirections, 400 Bad Request for malformed client requests, and 404 Not Found for missing resources. The implementation includes proper exception handling to maintain stability and responsiveness even under heavy load or in the presence of erroneous requests.

A key feature of the server is its login authentication system. User credentials are stored in a separate file, and when a POST request is sent to the /login endpoint, the server validates the submitted username and password. If the credentials are correct, the server responds with 302 Found and redirects the user to the upload page. If the authentication fails, it returns 401 Unauthorized, ensuring that only authorized users gain access to specific features.

Additionally, the server supports file uploads using multipart/form-data forms, allowing users to upload PNG images. The server extracts file data from the request body and securely stores uploaded files in a designated uploads directory within the public folder. To prevent security risks, uploaded filenames are sanitized by removing potentially harmful characters, ensuring safe storage. I had troubles with converting them into bits and then into image, but later on i figured it out.

Extensive testing was conducted using web browsers, curl, and REST clients to verify the server's functionality and resilience. The command curl -i http://localhost:2800/index.html was used to fetch the homepage while displaying HTTP headers, confirming the server's correct response with HTTP/1.1 200 OK and Content-Type: text/html. Authentication tests included sending a valid POST request to /login with correct credentials, resulting in a 302 Found response and redirection, while an incomplete request lacking a password triggered 400 Bad Request. Furthermore, a GET request to a non-existent file returned 404 Not Found, confirming that the server correctly handles missing resources.

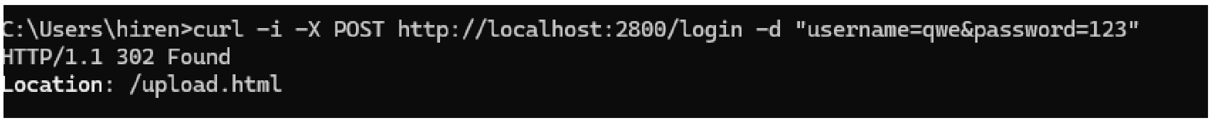
This project provided hands-on experience with Java networking APIs, TCP sockets, and HTTP protocol implementation. It reinforced the importance of security considerations in web servers, such as preventing directory traversal attacks and handling authentication securely. Additionally, the assignment deepened my understanding of multi-threading, enabling the server to efficiently handle concurrent client requests without performance degradation. Overall, this implementation showcases a functional and secure Java-based web server that adheres to core web development principles and best practices.



```
C:\Users\hiren>curl -i http://localhost:2800/index.html
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to the Simple Web Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
            background-color: #f4f4f4;
        }
        h1 {
            color: #333;
        }
        img {
            width: 300px;
            border: 2px solid #007BFF;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
            margin: 20px;
        }
    </style>
</head>
<body>
    <h1>Welcome to the Simple Web Server!</h1>
    <p>This is the default page for your web server.</p>

    <h2>Here is your Dog:</h2>
    <img src="/uploads/dog.png" alt="Dog Image">
</body>
</html>
```

curl -i: Fetches the content of the specified URL (http://localhost:2800/index.html) and includes HTTP headers in the output.

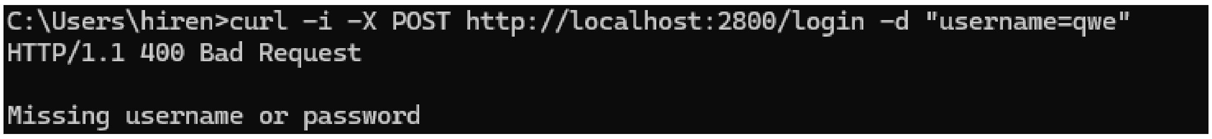
HTTP/1.1 200 OK: Indicates that the request was successful.
Content-Type: text/html: Specifies that the response is an HTML document.



```
:\Users\hiren>curl -i -X POST http://localhost:2800/login -d "username=qwe&password=123"
TTP/1.1 302 Found
ocation: /upload.html
```

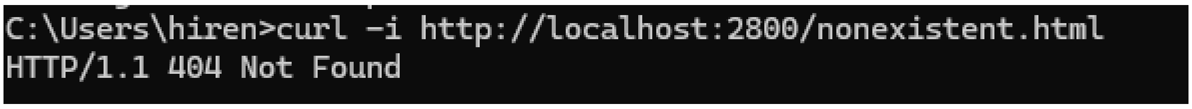
This command sends a POST request to http://localhost:2800/Login with form data (username=qwe&password=123).
The server responds with HTTP/1.1 302 Found, indicating a successful login.

The Location: /upload.html header suggests that the server is redirecting the user to upload.html, likely the upload page.



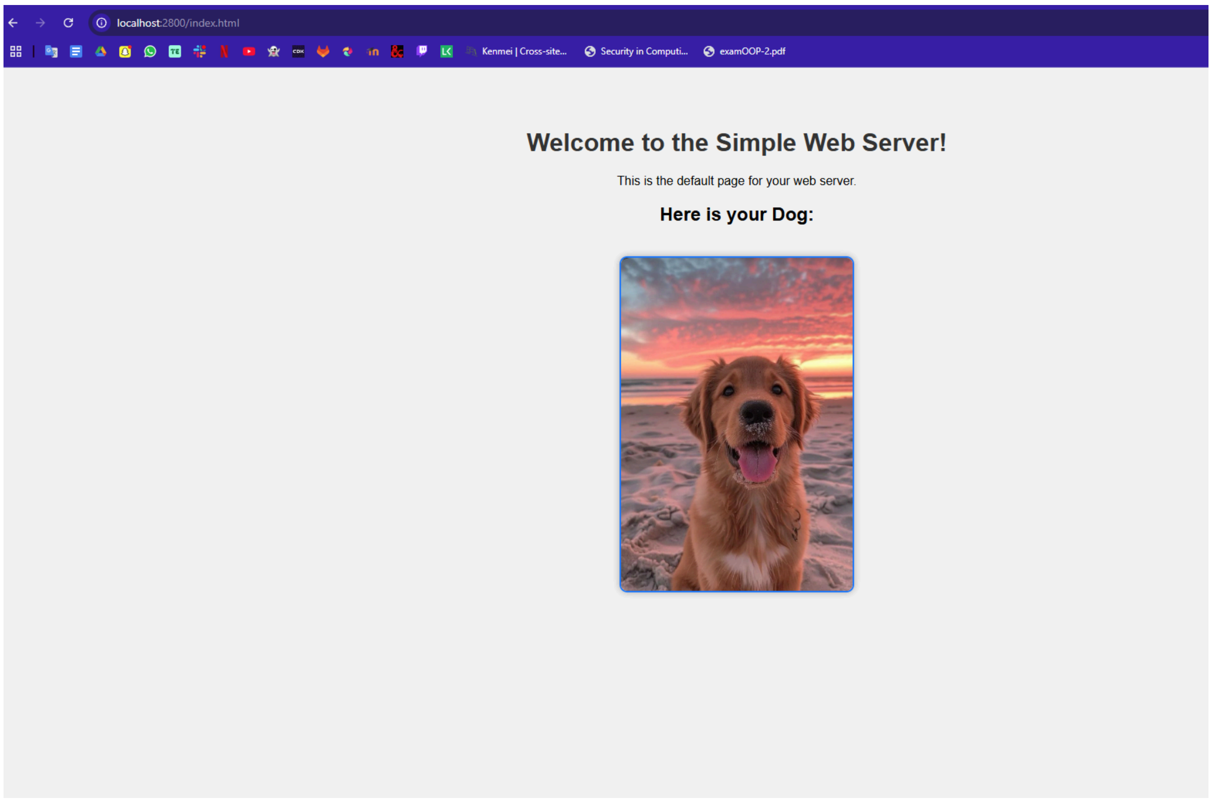
```
C:\Users\hiren>curl -i -X POST http://localhost:2800/login -d "username=qwe"
HTTP/1.1 400 Bad Request

Missing username or password
```

This is another POST request, but it only includes a username and omits the password.
The server returns HTTP/1.1 400 Bad Request, meaning the request was malformed.
The message Missing username or password suggests that both fields are required for authentication.



```
C:\Users\hiren>curl -i http://localhost:2800/nonexistent.html
HTTP/1.1 404 Not Found
```

This is a GET request to http://localhost:2800/nonexistent.html.
The server responds with 404 Not Found, meaning the requested file does not exist on the server.

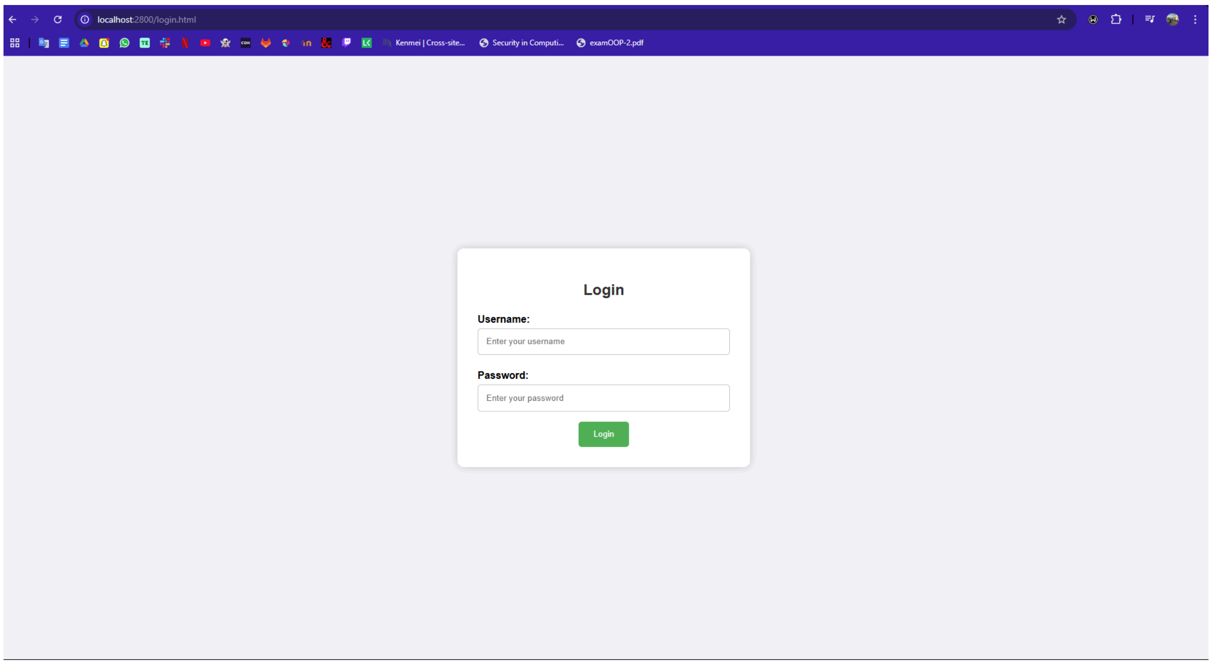


Welcome to the Simple Web Server!

This is the default page for your web server.

Here is your Dog:

Index.html


localhost:2800/login.html

Kenmei | Cross-site... Security in Computi... examOOP-2.pdf

Login

Username:

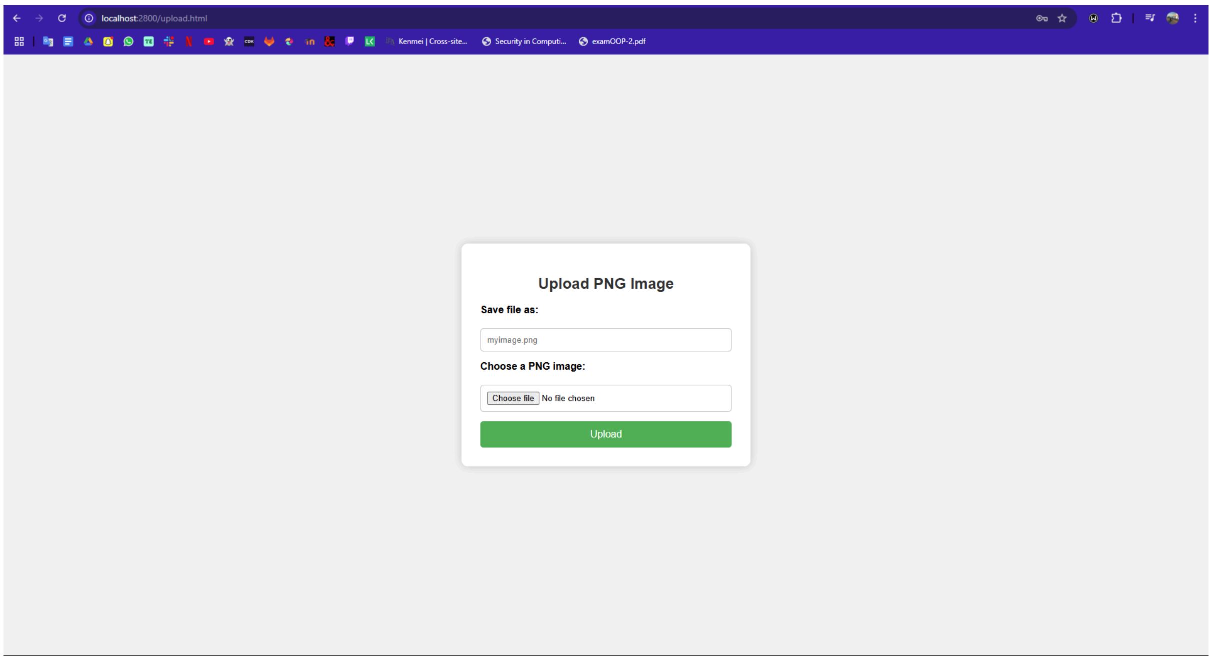
Enter your username

Password:

Enter your password

Login

login.html


localhost:2800/upload.html

Kenmei | Cross-site... Security in Computi... examOOP-2.pdf

Upload PNG Image

Save file as:

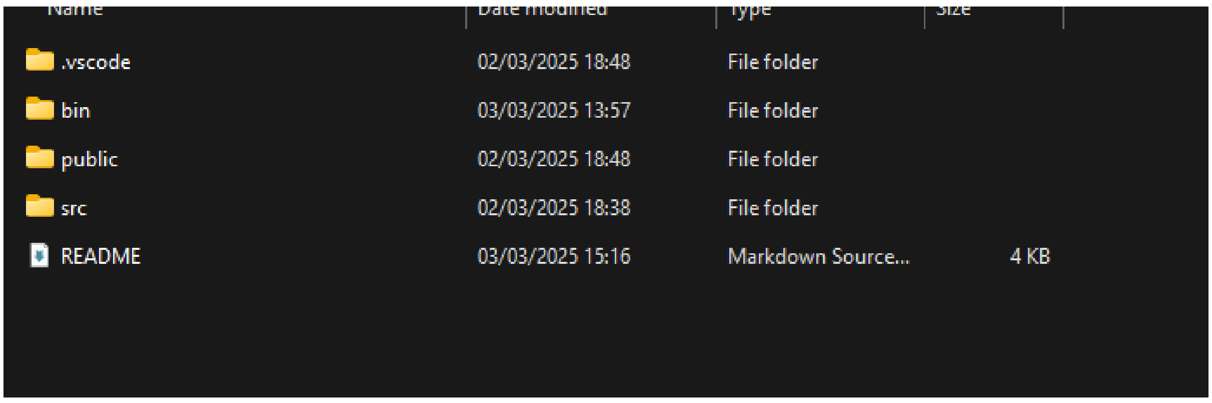
myimage.png

Choose a PNG image:

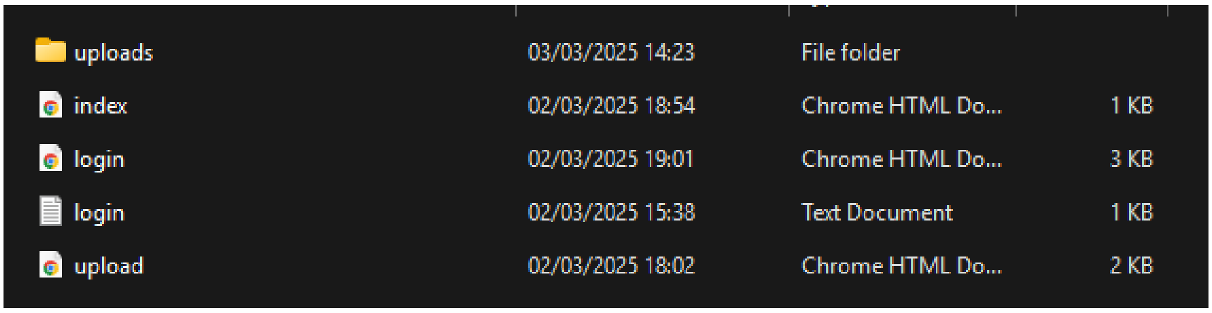
Choose file No file chosen

Upload

upload.html



Name	Date modified	Type	Size
.vscode	02/03/2025 18:48	File folder	
bin	03/03/2025 13:57	File folder	
public	02/03/2025 18:48	File folder	
src	02/03/2025 18:38	File folder	
README	03/03/2025 15:16	Markdown Source...	4 KB



uploads	03/03/2025 14:23	File folder	
index	02/03/2025 18:54	Chrome HTML Do...	1 KB
login	02/03/2025 19:01	Chrome HTML Do...	3 KB
login	02/03/2025 15:38	Text Document	1 KB
upload	02/03/2025 18:02	Chrome HTML Do...	2 KB

```
assignment2/
├── bin/                # Compiled classes will go here
├── public/              # Directory for static files
│   └──login.txt
│   └── index.html          # Your index.html file
│   └──login.html
│   └──upload.html
│   └──upload/
│       └── dog.png
├── src/
│   └── main/
│       └── java/
│           └── webserver/
│               ├── SimpleWebServer.java
│               └── Main.java
├── .vscode/
│   ├── settings.json
│   └── launch.json
└── README.md
```

This are the html file from the bowser and the structure of the project

upload/ is where all the image are being saved