Distributed Systems
Programming Assignment

Name: Samuel Karumanchi

Description:

This program implements a simple HTTP server using Python programming language. The server listens on a specified port and serves files from a specified directory. There are additional functionalities, like handling requests for specific routes or files, logging requests, and handling errors.

The below are the basic features and functionalities of my project;

● If the request doesn't follow the basic structure of an HTTP request, a 400 (Bad Request) response is sent.

● Requests to the root ("/") are automatically redirected to "/index.html".

● Any request starting with "/restricted" is treated as a forbidden resource, and a 403 (Forbidden) response is sent.

● If the requested file exists in the specified directory, it is served to the client. Otherwise, a 404 (Not Found) response is sent.

● For other errors, a 500 (Internal Server Error) is sent.

The web server is designed to handle incoming connections from clients over the network. It operates on a specified port, allowing clients to establish connections and retrieve files using the HTTP/1.0 and HTTP/1.1 protocols. When a client sends a request, the server parses the request to extract the requested filename and determines the appropriate file path within the server's document root directory. Additionally, the server implements access control mechanisms to ensure that only authorized users can access certain files. For instance, files like 'restricted' from the 'scu.edu' domain are restricted to users with appropriate permissions. This access control feature enhances security and prevents unauthorized access to sensitive information.

In summary, my web server provides a robust and functional platform for serving web content to clients over the network. It supports standard HTTP protocols, handles file requests efficiently, and enforces access control measures to maintain security and integrity.

Submitted Files:

- ➔ PA Report
- ➔ server.py
- ➔ README.md
- ➔ webserver_files (Folder)

Instructions:

The command ‘python3 server.py -root './server_files' -port 9768’ is used to run the file.

Open any web browser and try the following searches in the address bar:

- localhost: 9768/
- localhost: 9768/index.html
- localhost: 9768/scu.jpg
- localhost: 9768/scu.gif
- localhost: 9768/restricted

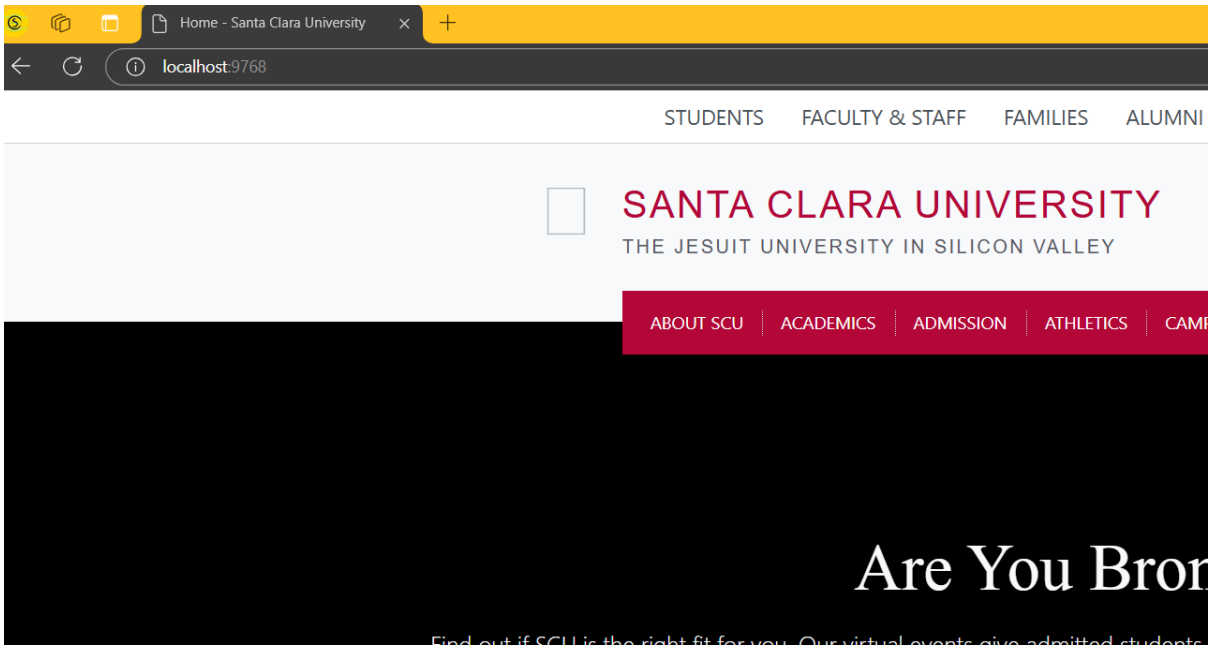
Try other types as well to see error detection. Port number can be changed using the command line and logs can also be viewed in the console.

Screenshots:

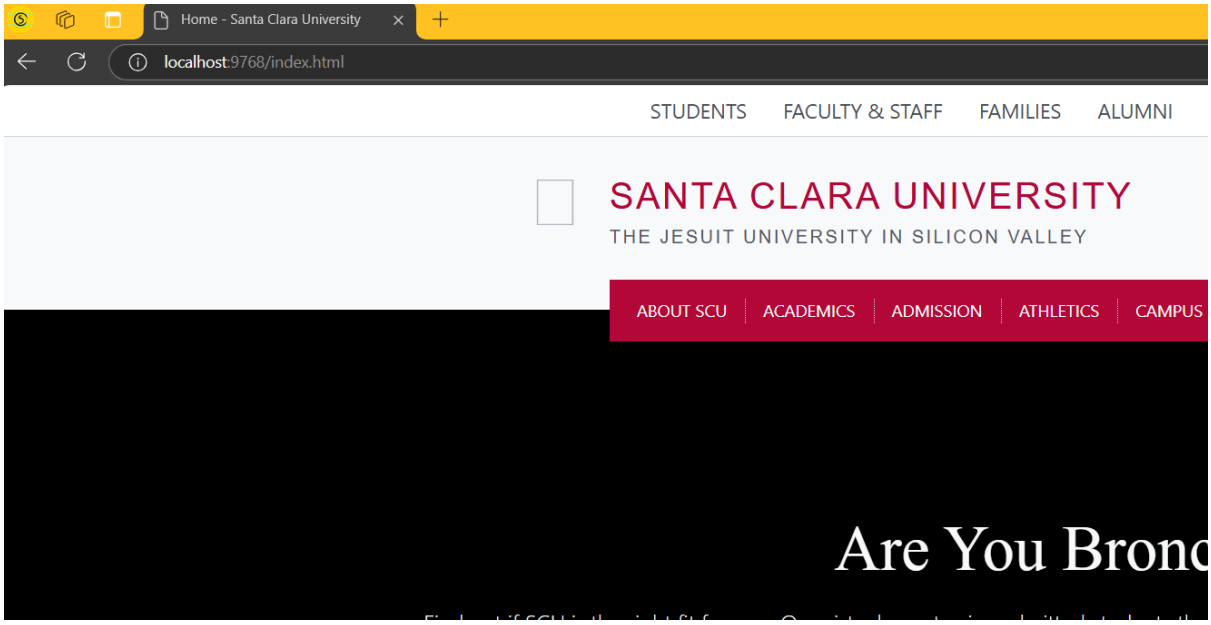
Listening for incoming requests.

```
Restarting kernel...
runfile('C:/Users/ksben/Documents/santa clara uni docs/Academics/Assisngments/DS/PA- Samuel
Karumanchi/server.py', wdir='C:/Users/ksben/Documents/santa clara uni docs/Academics/
Assisngments/DS/PA- Samuel Karumanchi')
Server is listening on port 9768...
```

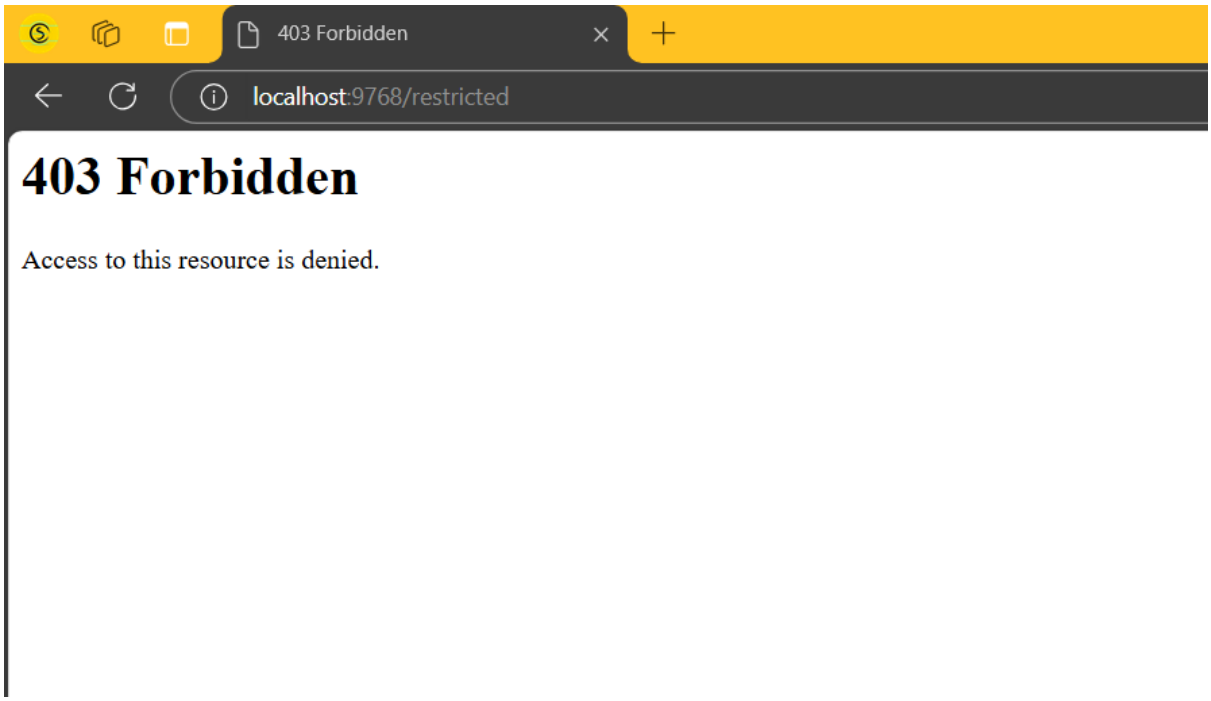
for localhost:9000 (same as localhost:9000/index.html)



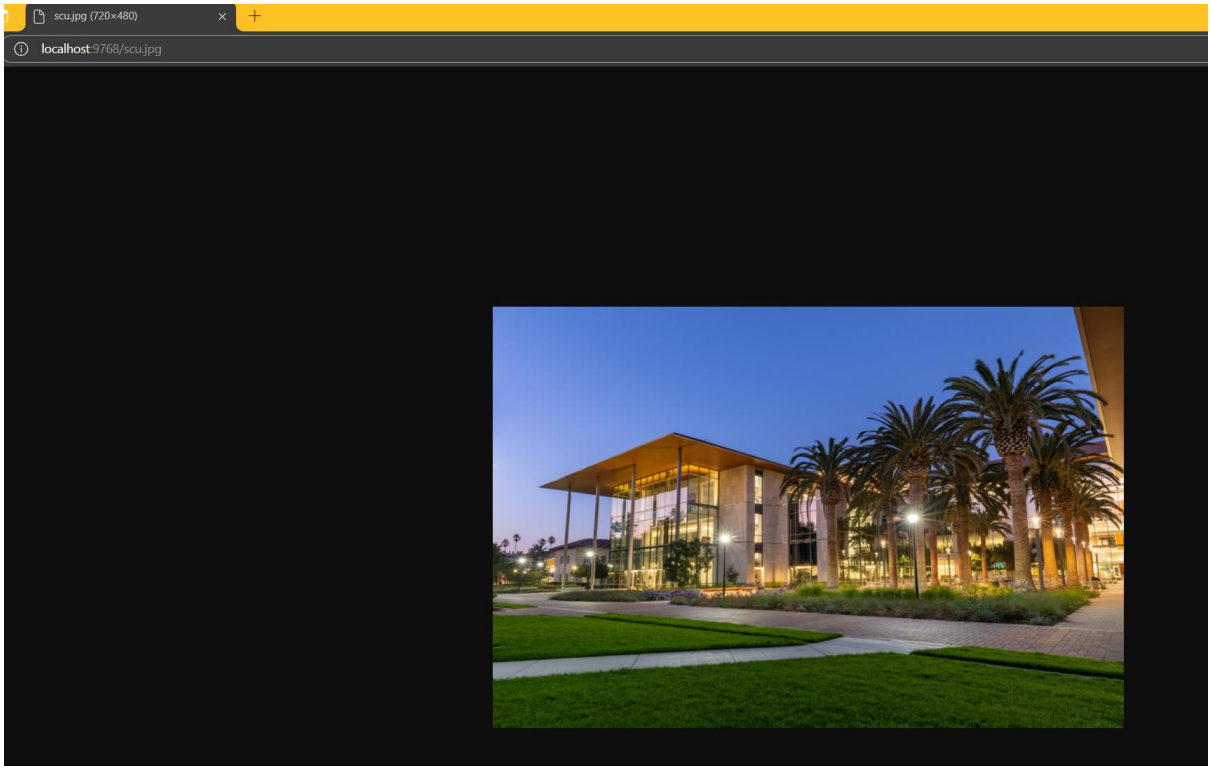
for localhost:9000/index.html



for permission denied



For scu.jpg



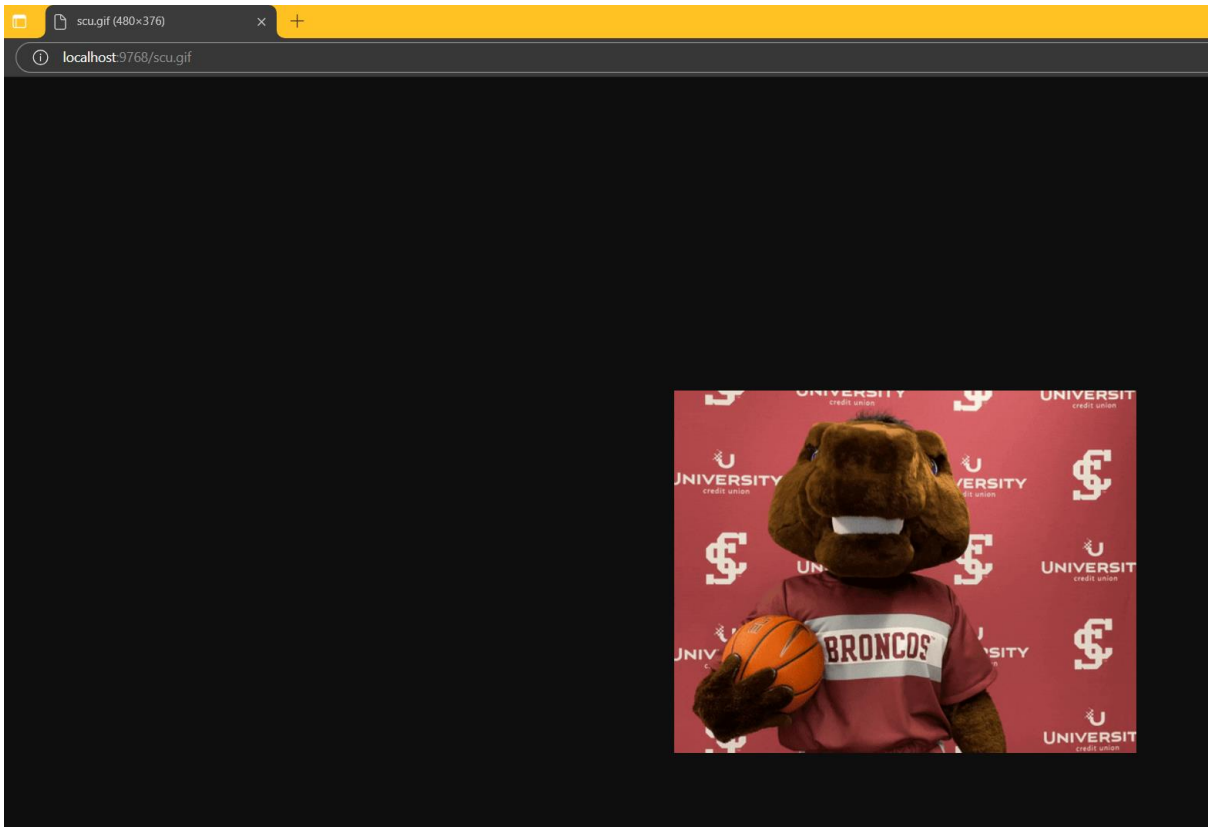
For Aboutproject.txt

localhost:9768/Aboutproject.txt

localhost:9768/Aboutproject.txt

```
COEN 317: Distributed Systems
Fall 2024
Programming Assignment 1
Due: Wednesday, 10/30/2024 @11:59pm
Project Description
The goal of this programming assignment is to build a functional web server. This
assignment will teach you the basics of distributed programming, client/server
structures, and issues in building high performance servers. While the course
lectures will focus on the concepts that enable network communication, it is also
important to understand the structure of systems that make use of the global
Internet.
This project should be done individually.
Part 1: Build the Server
At a high level, a web server listens for connections on a socket (bound to a specific
port on a host machine). Clients connect to this socket and use a simple text-based
protocol to retrieve files from the server. For example, you might try the following
command from a UNIX machine:
$ telnet www.scu.edu 80
GET /index.shtml HTTP/1.0
(Type two carriage returns after the "GET" command). This will return to you (on
the command line) the HTML representing the "front page" of the Santa Clara web
page.
One of the key things to keep in mind in building your web server is that the server
is translating relative filenames (such as index.html) to absolute filenames in a
local filesystem. For example, you might decide to keep all the files for your server in
~username/coen317/server/files/, which we call the document root. When your
server gets a request for index.html (which is the default web page if no file is
specified), it will prepend the document root to the specified file and determine if
the file exists, and if the proper permissions are set on the file (typically the file has
to be world readable). If the file does not exist, a file not found error (HTTP error
code 404) is returned. If a file is present but the proper permissions are not set, a
permission denied error is returned. Otherwise, an HTTP OK message is returned
along with the contents of a file.
You should note that since index.html is the default file, web servers typically
translate 'GET /' to 'GET /index.html'. That way, index.html is assumed to be the
filename if no explicit filename is present. This is why the two URLs
http://www.scu.edu and http://www.scu.edu/index.html return equivalent results.
Also note that the default index.html page may actually redirect to a different (i.e.,
the real) home page. For example, in the case of www.scu.edu, requesting index.html
returns a very short page that simply instructs the browser to request index.shtml
instead. Normally your browser performs this redirection for you automatically, but
a simple tool like telnet will simply show the actual contents of index.html if you
request it.
When you type a URL into a web browser, the server retrieves the contents of the
requested file. If the file is of type text/html (i.e., it is a regular web page written in
HTML) and the HTTP/1.0 protocol is being used, the browser will parse the HTML
for embedded links (such as images) and then make separate connections to the
web server to retrieve the embedded files. If a web page contains 4 images, a total of
five separate connections will be made to the web server to retrieve the html and
the four image files. Note that the previous discussion assumes the HTTP/1.0
protocol, which is what you will be supporting in this assignment.
HTTP/1.0 protocol
When you type a URL into a web browser, the server retrieves the contents of the
requested file. If the file is of type text/html (i.e., it is a regular web page written in
HTML) and the HTTP/1.0 protocol is being used, the browser will parse the HTML
for embedded links (such as images) and then make separate connections to the
web server to retrieve the embedded files. If a web page contains 4 images, a total of
five separate connections will be made to the web server to retrieve the html and
the four image files.
```

For scu.gif



for any other inputs