Distributed Programming Assignment

Venkata Sai Srikar Jilla
W1652687

Overview:

This program implements a simple HTTP server using the Go 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.

Features and Functionalities:

Request Handling:

- 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 "/private" 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.

Dispatcher Mechanism:

The server continuously listens for incoming client connections using the ln.Accept() method. When a client connects to the server, it establishes a new connection represented by the conn object.

For each accepted connection, the server doesn't process the request in the main execution thread. Instead, it spawns a new lightweight thread, known as a "goroutine", to handle the client's request. This is done using the go keyword followed by the processClientRequest(conn) function. The use of goroutines allows the server to handle multiple requests simultaneously without waiting for one request to complete before moving to the next.

Content Type Handling: The server can serve various file types, such as .html, .txt, .jpg, and .gif.

Custom Port and Document Root: Through the use of command-line flags, the server can be started on a custom port (-port) and can serve files from a specified directory (-document_root).

Client Request Handling: When a client connects to the server and sends an HTTP request, the processClientRequest function reads the request, extracts the necessary headers, and delegates further handling to the handleRequest function.

Timezone Handling: The program is set to recognize the Pacific timezone (America/Los Angeles) for logging purposes.

Logging: Every response from the server is logged to the console with a timestamp in the Pacific time zone, the HTTP status code, and the request URI.

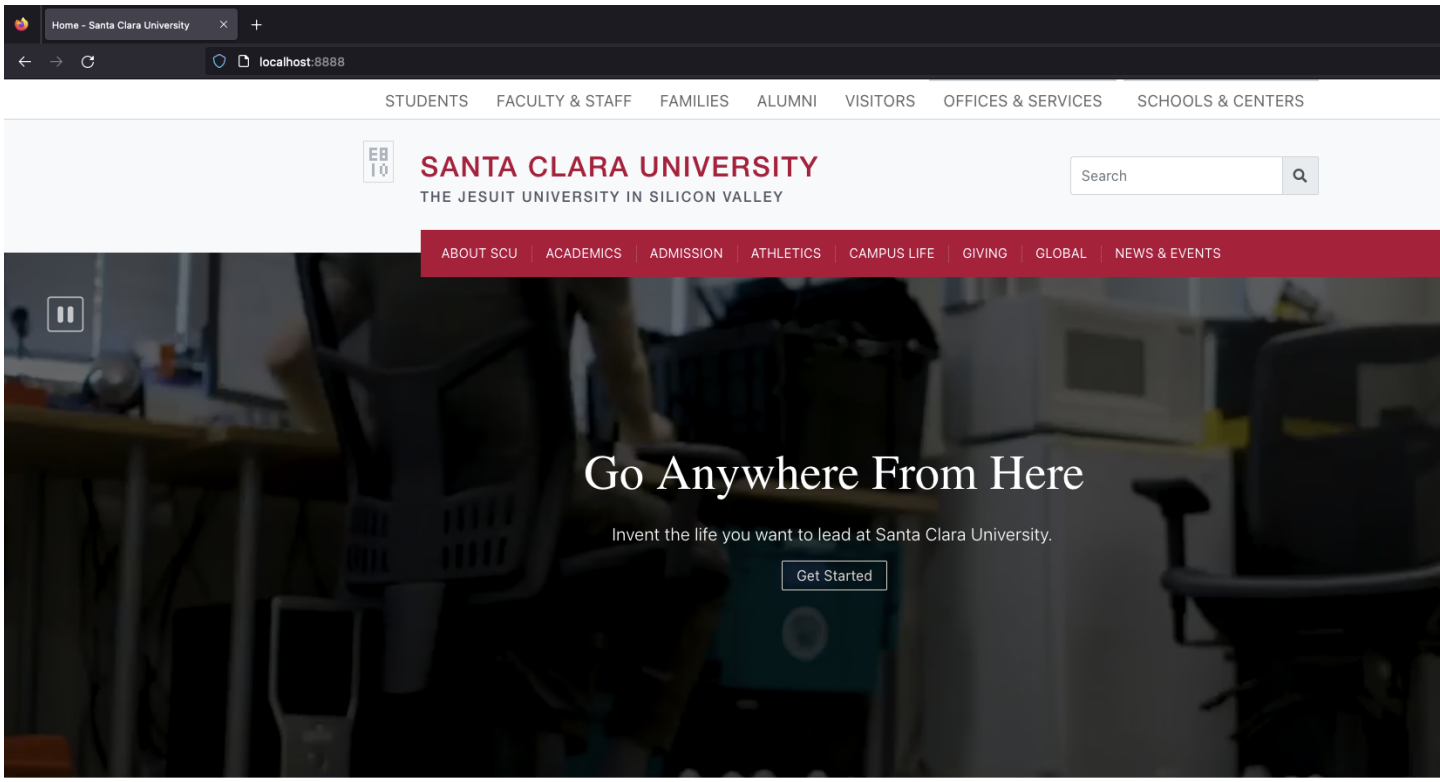
Usage:

go run server.go -port=8888 -document_root=./webserver_files

This would start the server on port 8888 and serve files from the "webserver_files" directory.

Screenshots:

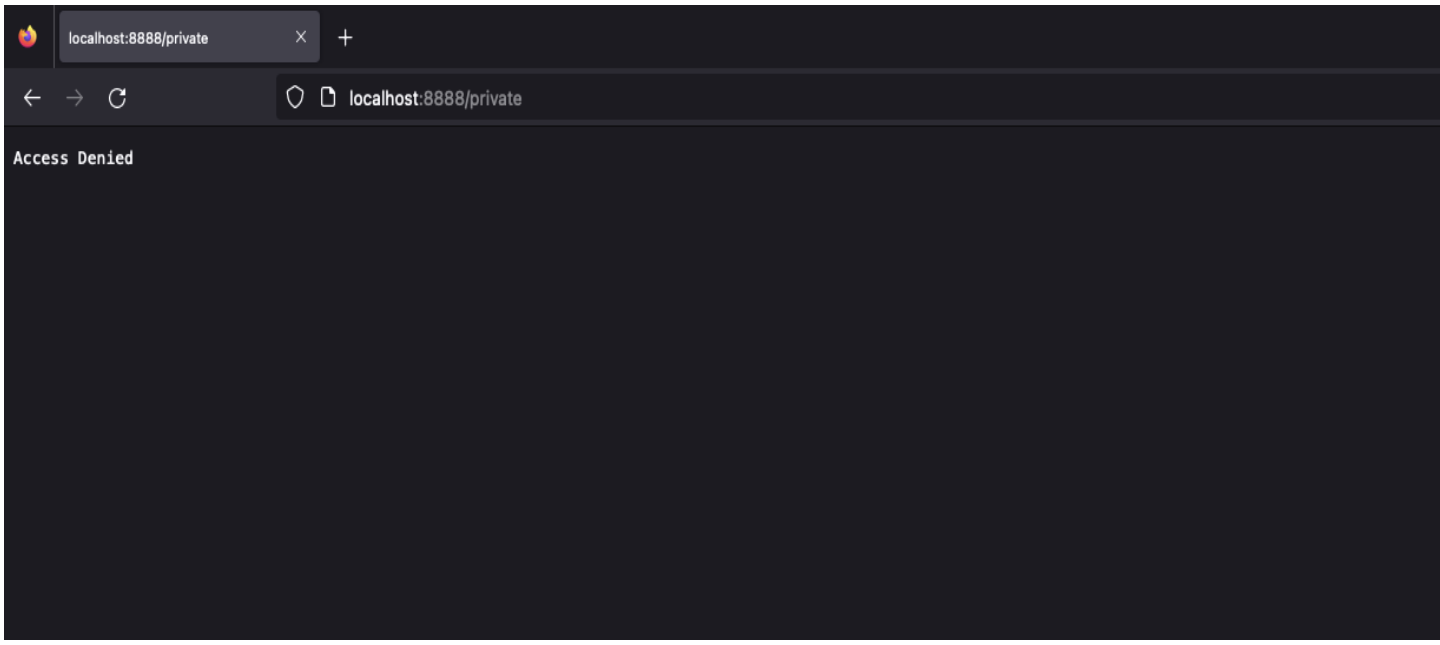
1. Url: http://localhost:8888/ SCU Home Page; Status Code: 200



We're students, teachers, and mentors headquartered in the most innovative place on earth: Silicon Valley.

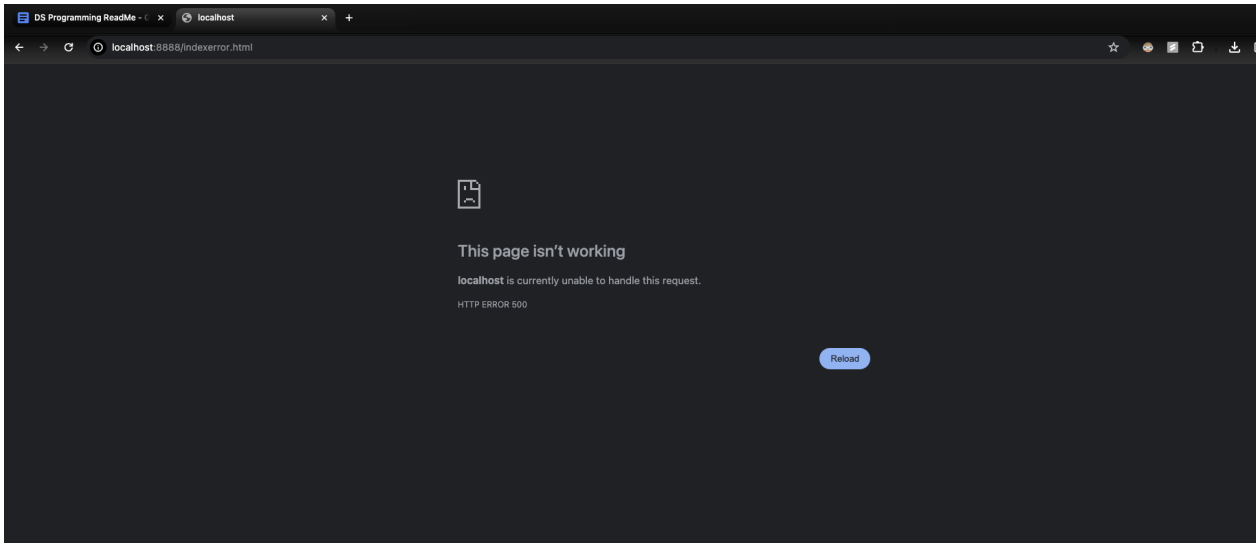
Server Log:

2. Url: http://localhost:8888/private; Private Url for which access is denied; Status Code: 403



Server log:

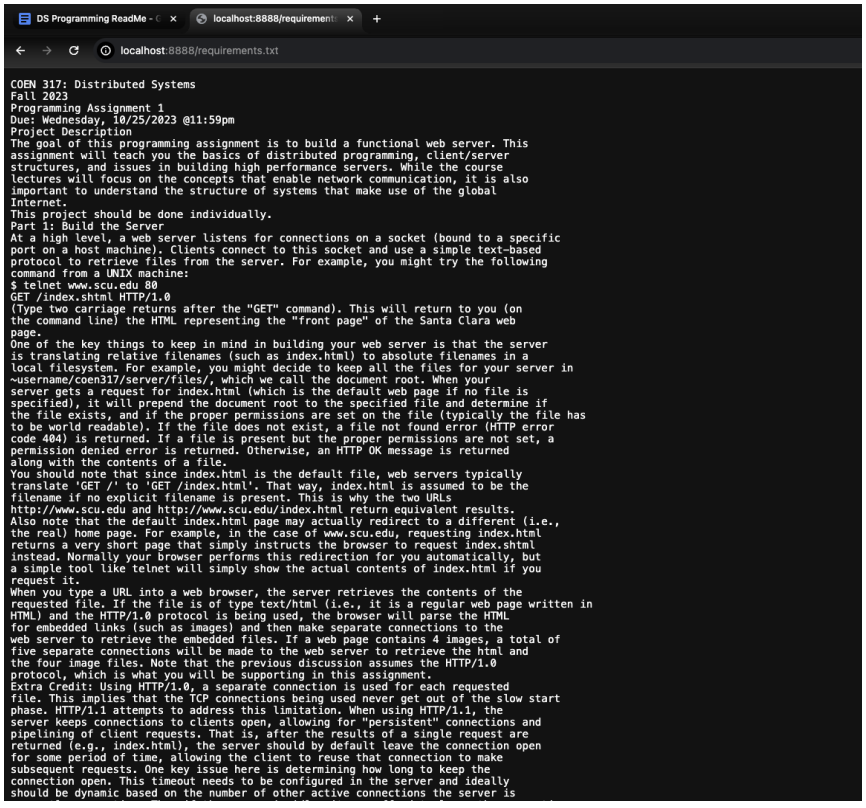
3. Url: http://localhost:8888/indexerror.html; url exists server unable to serve. Status Code: 500



Server Log:

```
2023/10/25 21:41:45 [2023-10-25T21:41:45-07:00] 500: /indexerror.html
```

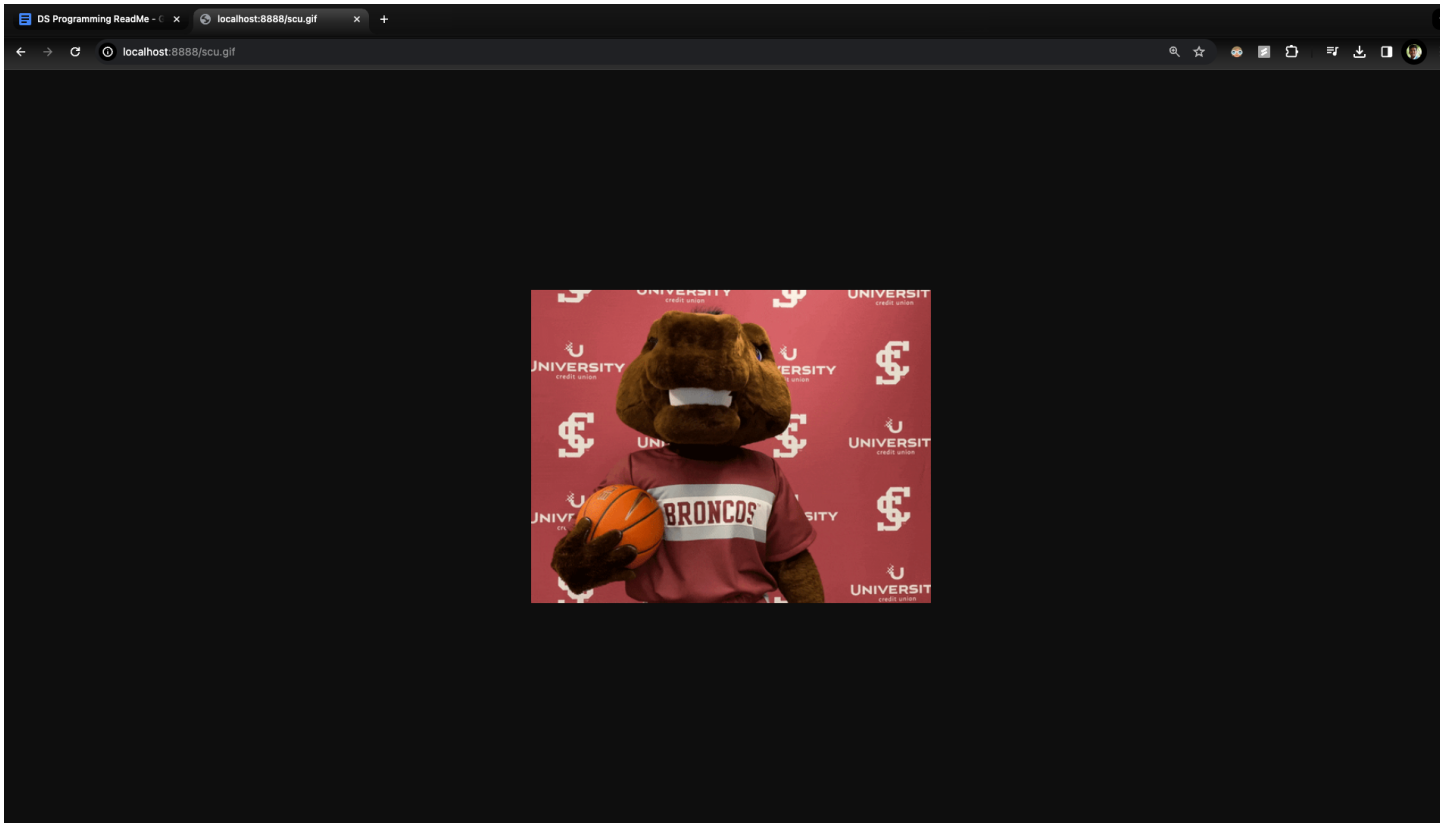
4. Supporting txt; requirements.txt http://localhost:8888/requirements.txt; Status Code: 200



Server Log:

```
2023/10/25 21:53:53 [2023-10-25T21:53:53-07:00] 403: /private
```

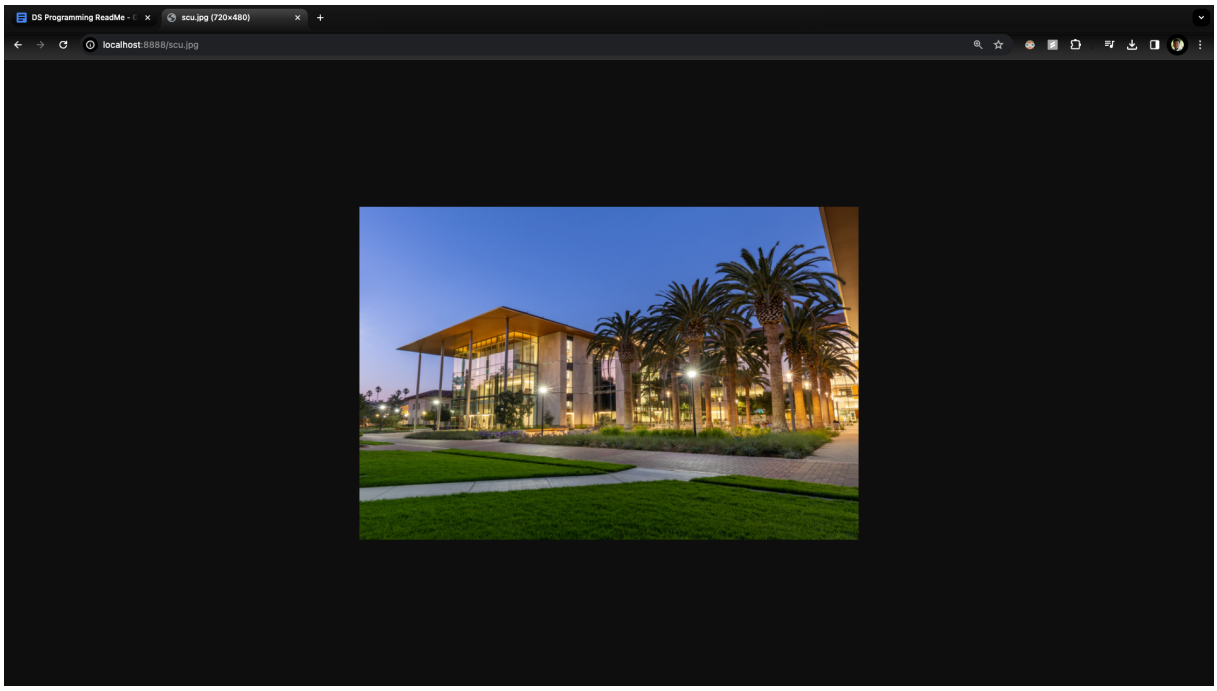
5. SCU GIF; url: http://localhost:8888/scu.gif; Status Code: 200



Server Log:

```
2023/10/25 21:57:02 [2023-10-25T21:57:02-07:00] 200: /scu.gif
```

6. SCU JPG: http://localhost:8888/scu.jpg; Status Code: 200



Server Log:

```
2023/10/25 21:57:48 [2023-10-25T21:57:48-07:00] 200: /scu.jpg
```

7. Status Code: 400 Using Terminal:

```
Terminal   Local   Local (2)   Local (3)
saisrikar@sais-MacBook-Pro DSProgrammingAssignment % echo -ne "GET" | nc localhost 8888
HTTP/1.1 400 Bad Request
Content-Type: text/plain

Bad Request%
saisrikar@sais-MacBook-Pro DSProgrammingAssignment %
```