Homework 4 – Programming Assignment

CS5700

Implementing Using Python Client/Server Libraries

Points Possible: 20

Problem Statement:

The purpose of this assignment is for you to learn how to implement a client/server Python program using built-in client/server Python libraries. It is best practice to utilize out-of-the-box libraries, instead of re-inventing the wheel and attempting to implement from scratch. Although, it is a good learning experience to do the latter, as you saw in the past homework.

In this assignment, you will have a template client-hw4.py and server-hw4.py file already provided to you on Canvas. These files have some code already within them. Your job is to add in code to the files so that the server will be able to only serve HTML and JSON files to the client. In other words, the only supported content-types will be **text/html** and **application/json**.

You would also need to implement error handling for files that cannot be found and unsupported content-types.

This program will only utilize HTTP GET requests (client will send an HTTP GET request and the server will receive it).

Set the server to listen on port 8070.

The server should not terminate between requests, it should remain active and listening.

Two sample files will also be provided to you on Canvas: data.json and index.html.

The client-hw4.py, server-hw4.py, data.json, and index.html should all be located in the same folder on your computer.

To summarize, there are four scenarios to which your program must support:

1. Server must return the requested HTML file back to the client if it exists.
2. Server must return the requested JSON file back to the client if it exists.
3. If the JSON or HTML file doesn't exist then a 404 File Not Found error is sent from the server to the client.
4. If the file is an unsupported type (i.e. not text/html or application/json), then a 404 Unsupported Route error is sent from the server to the client.

See the "Examples" section for further requirements and how the output is expected to look like. Do not deviate from the format as specified in the "Examples" sections.

In addition, only write your code where you see "Insert your code here" as a comment. Do not modify any pre-existing code or add any code outside of where the "Insert your code here" area.

Failure to follow these instructions will get you a lot of points deducted, up to and including receiving zero credit.

Submission Requirements:

1. Write your name as a comment at the top of your code.
2. You should submit two files: client-hw4.py and server-hw4.py
    a. Using any other file name will get you points deducted.
    b. As mentioned above, you need to update the already provided client-hw4.py and server-hw4.py files with your code. See the above instructions for further details.
3. You cannot modify the pre-existing code in the server-hw4.py and client-hw4.py.
    a. In addition, you are not allowed to add any other library than what is already provided in the code.
    b. You can only add code where it says “Insert your code here” in the client-hw4.py and server-hw4.py files.
    c. Failure to follow these instructions will result in massive point deductions, up to and including receiving zero credit.
4. Your program must be able to be executed via the command prompt in Windows or the Terminal on the Mac. See the examples below.
5. You can work with other students or individually, up to you. However, you must submit your assignment individually on Canvas.
6. You must submit your file(s) on Canvas. Any other submission method (such as email) will be rejected and you will receive zero credit.
7. This assignment must be done using the Python version as mentioned in the syllabus.
8. The format of the output in the console of the client and server, while server-hw4.py is running and after you execute each client-hw4.py request, should match exactly to the format that you see in the examples below. Anything different or any other result that’s outputted via our test cases will result in point deductions. An obvious exception is the value of the timestamp in the server console.
9. Canvas will automatically add a hyphen and a number to the file name if you upload the same file more than once. That’s fine and don’t worry about it, I will not deduct points for that.
10. Do not hard-code “data.json” or “index.html” anywhere in your code! Doing so will result in a lot of points deductions. We may test your code with different filenames as well.
11. If any requirements are unclear, ambiguous, you feel are contradictory, or if you have any questions, you would need to reach out to the professor and/or TAs as soon as possible. Do not assume anything! It is your responsibility to clear up any confusion and confirm any assumptions. Otherwise, you risk getting a lot of points deducted and your regrade request will be denied.

Examples:

The below examples are run from the command prompt. This is how your code will be graded. Your program absolutely needs to be able to be run from the command prompt or Terminal, otherwise you will get zero credit. No exceptions. The output of your code should display in the command prompt or Terminal console.

The format of the output in the console of the client and server, while server-hw4.py is running and after you execute each client-hw4.py request, should match exactly to the format that you see in the

examples below. Anything different or any other result that's outputted via our test cases will result in point deductions. An obvious exception is the value of the timestamp in the server console.

Open two command prompt or Terminal windows: one for the server and the other for the client.

Starting the server:

```
>python server-hw4.py
Serving on port 8070
```

Sending an HTML request from the client to the server:

```
>python client-hw4.py "index.html"
Response Status Code: 200
Content Type: text/html
Response Content:
<!DOCTYPE html>
<html>
<head>
   <title>CS5700</title>
</head>
<body>
   <h1>Welcome to the Course: Fundamentals of Computer Networking</h1>
   <p>This is a simple HTML page to test the server.</p>
</body>
</html>
```

After each request, the console on the server side will output a log of each request:

```
127.0.0.1 - - [20/Jan/2024 22:10:55] "GET index.html HTTP/1.1" 200 –
```

Sending a JSON request from the client to the server:

```
>python client-hw4.py "data.json"
Response Status Code: 200
Content Type: application/json
Response Content:
{"library": {"name": "City Central Library", "location": "123 Library Street, Booktown", "books": [{"title":
"The Great Adventure", "author": "Alice Smith", "year": 2018, "genres": ["Adventure", "Fantasy"]}, {"title":
"History of the Ancient World", "author": "John Doe", "year": 2011, "genres": ["History", "Education"]},
{"title": "Cooking with Flavors", "author": "Maria Garcia", "year": 2020, "genres": ["Cooking", "Lifestyle"]},
{"title": "Science and Technology Today", "author": "James Lee", "year": 2019, "genres": ["Science",
"Technology"]}, {"title": "Gardening for Beginners", "author": "Sophie Turner", "year": 2015, "genres":
["Gardening", "Hobby"]}]}}
```

Sending the wrong filename (file not found) request from the client to the server:

```
>python client-hw4.py "FileDoesNotExist.html"
Response Status Code: 404
Content Type: text/html;charset=utf-8
Response Content:
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Error response</title>
  </head>
  <body>
    <h1>Error response</h1>
    <p>Error code: 404</p>
    <p>Message: File Not Found: FileDoesNotExist.html.</p>
    <p>Error code explanation: 404 - Nothing matches the given URI.</p>
  </body>
</html>
```

For file not found errors, two rows should show up in the server's log:

```
127.0.0.1 - - [20/Jan/2024 22:29:44] code 404, message File Not Found: FileDoesNotExist.html
127.0.0.1 - - [20/Jan/2024 22:29:44] "GET FileDoesNotExist.html HTTP/1.1" 404 -
```

Requesting an unsupported content-type from the client to the server:

```
>python client-hw4.py "test.txt"
Response Status Code: 404
Content Type: text/plain
Response Content:
404 Not Found - Unsupported Content Type
```

Based on the examples above, the server should show the following in the console:

```
>python server-hw4.py
Serving on port 8070
127.0.0.1 - - [20/Jan/2024 22:29:05] "GET index.html HTTP/1.1" 200 -
127.0.0.1 - - [20/Jan/2024 22:29:22] "GET data.json HTTP/1.1" 200 -
127.0.0.1 - - [20/Jan/2024 22:29:44] code 404, message File Not Found: FileDoesNotExist.html
127.0.0.1 - - [20/Jan/2024 22:29:44] "GET FileDoesNotExist.html HTTP/1.1" 404 -
127.0.0.1 - - [20/Jan/2024 22:30:42] "GET test.txt HTTP/1.1" 404 -
```

Another example of a file not found request from the client to the server:

```
> python client-hw4.py "FileDoesNotExist.json"
Response Status Code: 404
Content Type: text/html;charset=utf-8
Response Content:
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <meta charset="utf-8">
```

```
    <title>Error response</title>
  </head>
  <body>
    <h1>Error response</h1>
    <p>Error code: 404</p>
    <p>Message: File Not Found: FileDoesNotExist.json.</p>
    <p>Error code explanation: 404 - Nothing matches the given URI.</p>
  </body>
</html>
```

Corresponding server log:

```
127.0.0.1 - - [02/Feb/2024 09:32:36] code 404, message File Not Found: FileDoesNotExist.json
127.0.0.1 - - [02/Feb/2024 09:32:36] "GET FileDoesNotExist.json HTTP/1.1" 404 -
```

What to submit:

- Two files must be submitted on Canvas for this assignment:
  - client-hw4.py
  - server-hw4.py

Grading Guidelines:

- Does the program meet the requested requirements/criteria?
- Are the submission instructions followed?
- Does your code compile and execute?
- Does your code pass my test cases?