CSE 4074 – Programming Assignment

Due 30.12.2023 Friday, 11:59 PM

Socket Programming – HTTP-based Room Reservation

In this project, you are required to implement an HTTP server that acts as a room reservation server. There should be two other HTTP servers for tracking rooms and activities as well. All the three server programs should take single argument which specifies the port number. After starting a server, you can connect to it using any web browser or `curl` on the same machine with http://localhost:‹port›/ (or from another machine, by replacing localhost with the IP address). You may use any programming language, providing that you apply socket programming.

Room Server:
The server starts out with empty set of rooms, and it also maintains a simple database. It must support several kinds of GET requests:
/add?name=roomname: Adds a new room. If the room is already added, this server should return an HTTP 403 Forbidden message indicating that the room already exists in the database. Otherwise, it adds the room to the database, and sends back a 200 OK message including an HTML file in the body that informs that the room is added.
/remove?name=roomname: Removes the room with name=roomname. If the room exists, it removes the room from the database and sends back HTTP 200 OK message (including relevant info in the body as an HTML object). If the room doesn't exist, then it sends back an HTTP 403 Forbidden message.
/reserve?name=roomname&day=x&hour=y&duration=z: Reserves the room with name=roomname for the day x of the week at y:00 for z hours.
For example, /reserve?name=M2Z08&day=1&hour=9&duration=3 reserves M2Z08 room for Monday, 9:00 am for 3 hours. Note that reservations may only be done weekly. Day value can be an integer between 1 and 7 (1: Monday … 7: Sunday). Hour value can be an integer between 9 and 17 (reservations can be done for full-hour. Half hour reservations are not possible.) If any of these inputs are not valid, it sends back an HTTP 400 Bad Request message. If the room is already reserved then HTTP 403 Forbidden message will be sent.
/checkavailability?name=roomname&day=x: Checks the available hours for the room with name=roomname and returns back all the available hours (for the specified date) in the body of HTTP 200 OK message. If no such room exists it sends back an HTTP 404 Not Found message, or if x is not a valid input then it sends back an HTTP 400 Bad Request message.

Activity Server:
The server starts out with empty set of activities, and it also maintains a simple database. It must support several kinds of GET requests:
/add?name=activityname: Adds a new activity. If the activity is already added, this server should return an HTTP 403 Forbidden message indicating that the activity already exists in the database. Otherwise, it adds the activity to the database, and sends back a 200 OK message including an HTML file in the body that informs that the activity is added.
/remove?name=activityname: Removes the activity with name=activityname. If the activity exists, it removes the activity from the database and sends back HTTP 200 OK message (including relevant info in the body as an HTML object). If the activity doesn't exist, then it sends back an HTTP 403 Forbidden message.
/check?name=activityname: Checks whether there exists an activity with name=activityname.

If the activity exists, it sends back an HTTP 200 OK message, otherwise it sends HTTP 404 Not Found message.

Reservation Server:

This server is for reserving rooms for activities, and it also maintains a simple database. End users will only access this server. It must support several kinds of GET requests:

/reserve?room=roomname&activity=activityname&day=x&hour=y&duration=z: When this request is received, the server first checks whether there exists an activity with name activityname by contacting the Activity Server. If no such activity exists it sends back an HTTP 404 Not Found message. If exists, then it contacts with the Room Server and tries to reserve the room for the specified date/hour/duration. If any of the input is invalid, it sends back an HTTP 400 Bad Request message. If all the inputs are valid, then it either reserves the room and sends back an HTTP 200 OK message, or it sends back an HTTP 403 Forbidden message indicating that the room is not available. If the room is reserved, a reservation_id is generated (which can be an integer), and an entry is stored for the reservation_id.

/listavailability?room=roomname&day=x: Lists all the available hours for the specified day (after contacting the Room Server). (HTTP 200 OK is returned in success. In case of error relevant error messages will be sent as described above).

/listavailability?room=roomname: Lists all the available hours for all days of the week (after contacting the Room Server probably several times). (HTTP 200 OK is returned in success. In case of error relevant error messages will be sent as described above).

/display?id=reservation_id: Returns the details of the reservation with the specified id. If the reservation_id does not exist, it returns back an HTTP 404 Not Found message.

For every success or error response message, it is desired to send a relevant info in the message body as an HTML object (for all the servers).

Your server should print out information about every message received and every message sent. For storing data, you may use a database management system if you prefer. However, using a simple txt file as a database is also ok.

Example Run:

Now, let's say that you specify the port numbers of the servers as follows:
Reservation Server: 8080; Room Server: 8081; Activity Server: 8082

Then, the following is an example run:

(Note that here we displayed `curl` output, but in the demo we will use a browser. Whole HTTP messages are not shown, as the status and the header lines are not visible in the browser or `curl` output, but they should be printed on the screen by the server. Also note that you may modify the contents of HTML messages as you wish, providing that it gives the relevant information.)

```
$ curl http://localhost:8081/add?name=M2Z08

<HTML>
<HEAD>
<TITLE>Room Added</TITLE>
</HEAD>
<BODY> Room with name M2Z08 is successfully added.</BODY>
</HTML>

$ curl http://localhost:8081/remove?name=M1Z01

<HTML>
<HEAD>
<TITLE>Error</TITLE>
</HEAD>
<BODY> Room with name M1Z01 is not found. </BODY>
</HTML>
```

```
$ curl http://localhost:8082/add?name=CSE4197Seminar
<HTML>
<HEAD>
<TITLE>Activity Added</TITLE>
</HEAD>
<BODY> Activity with name CSE4197Seminar is successfully
added.</BODY> </HTML>


$ curl
http://localhost:8080/reserve?room=M2Z08&activity=CSE4197&day=5&hour=10&duration=2
<HTML>
<HEAD>
<TITLE>Reservation Successful</TITLE>
</HEAD>
<BODY> Room M2Z08 is reserved for activity CSE4197Seminar on Friday 10:00-12:00.
Your Reservation ID is 1.</BODY>
</HTML>


$ curl http://localhost:8080/listavailability?room=M2Z08&day=5
<HTML>
<HEAD>
<TITLE>Available Hours</TITLE>
</HEAD>
<BODY> On Friday, Room M2Z08 is available for the following hours: 9 13 14 15 16
17</BODY>
</HTML>


$ curl http://localhost:8080/display?id=1
<HTML>
<HEAD>
<TITLE>Reservation Info</TITLE>
</HEAD>
<BODY> Reservation ID:1 <BR>
Room: M2Z08 <BR>
Activity: CSE4197Seminar <BR>
When: Friday 10:00-12:00. </BODY>
</HTML>
```

Some **bonus points** will be given if you also support POST method together with the GET method. Also, projects with multi-threaded implementation considering concurrency issues among multiple users may be awarded if the idea is well documented. Any genius ideas or extra important features may also be awarded by bonus points. You may get up to 20% bonus from all these.
You can do your project in groups of **two** or **three**. But you have to give the names of your group members before December 16, Friday by sending email to **kubra.uludag@marmara.edu.tr**. After December 16, no new groups will be allowed.

What to submit? - You should submit your projects in a zip file which contains your well COMMENTED source code and DETAILED report via **google classroom**.

Name of the zip file should be: name1surname1_name2surname2_name3surname3.zip. No need to write your IDs on the file names. But in your report, clearly indicate student IDs and names of group members. Detailed report should include design document, implementation details and screenshots from sample outputs.