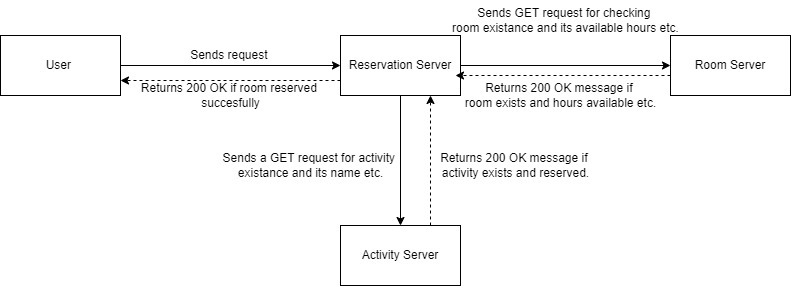
**CSE4074 Computer Networks**

**Programming Assignment**

**Group Members:** Mahmut SALMAN 150118506

Hakan SANDIKÇI 150119809

Hakan ADAKLI 150116069

**Design Document : **

**Implementation Details:**

1. **Room Server:** When the server receives an HTTP request, it processes the request and sends back a response. The main method creates a ServerSocket to listen for incoming connections on port 8081. It then reads in a list of rooms from a file called rooms.txt and stores them in an ArrayList called rooms. The server then enters a loop and waits for incoming connections. When a connection is accepted, a new ClientHandler thread is created to handle the request.

The ClientHandler class implements the Runnable interface, so it can be executed in a separate thread. Its run method reads the incoming request, parses the query string, and then sends a response back to the client.

It's worth noting that the server only handles a single request at a time, so it's not well-suited for handling a high volume of requests concurrently.

When we make these request in the web browser the server gets multiple requests so we needed to filter the requets with favicon.

We handle get requests based on whether they consists variables like day, duration, add etc.

1. **Activity Server :** This class has arraylist variable called activitynames. In the main function of this class we read activityNames.txt . We add these activities into this arraylist. After that the activity server waits for incoming requests.

Also in this class we use ClientHandler2 which implements runnable class. This server accepts add, remove, check requests as mentioned in the project document. We also filter the requests consists “favicon” in their request string.

After we handle incoming request using Socket we close Socket and we update activityNames.txt file.

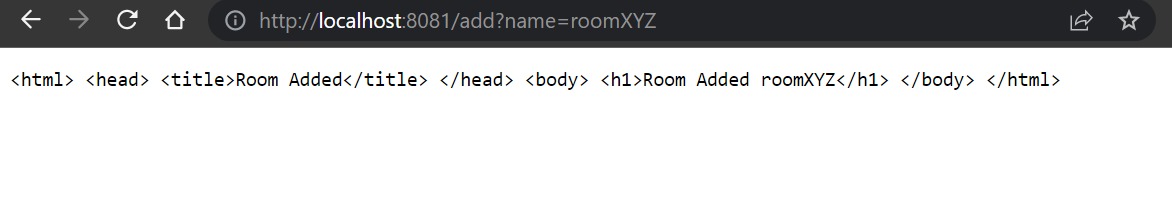
1. **Reservation Server:** This server uses port number 8080. In the beginning of this server’s main function we read reservations.txt file. And we create reservation info objects and put these objects into the reservationMap Hashmap. After this, this server waits for incoming requests.

After we filter the request with favicon we check these requests whether they consists pre-defined variables ( ie. For handling this request : /reserve?room=roomname&activity=activityname&day=x&hour=y&duration=z) we check if the incoming request has reserve, room, activity.

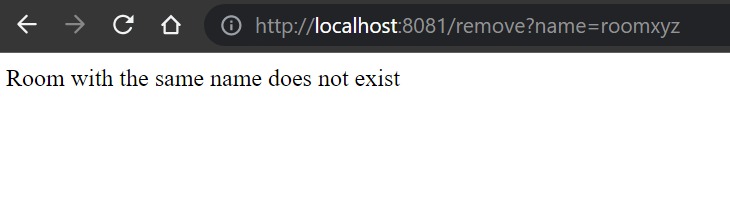
After extracting parameters from the request we first check if and activiitiy exists with name “activityname” . To do this we send a request to activity server and based on response we continue to next step or 403 Forbidden response. If that activity exists then we send another request tor room server to reserve the room with the variables day, hour and duration.

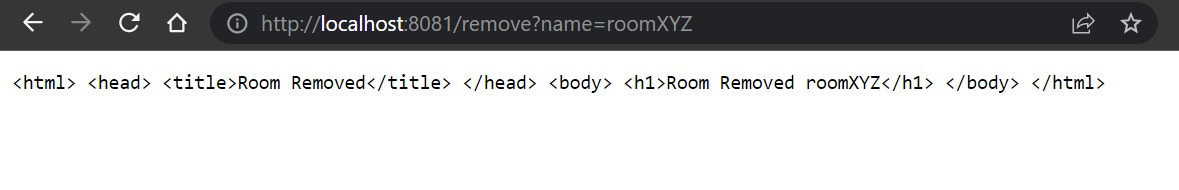
If the response is 200 OK it means the user has succesfully reserved this room.

**Screenshots:**

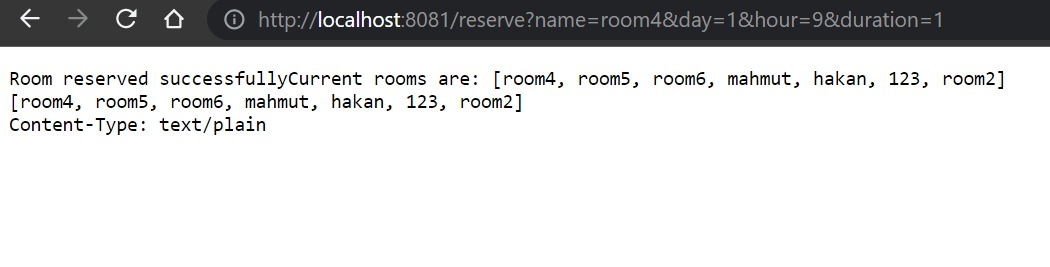
**AddRoom:**

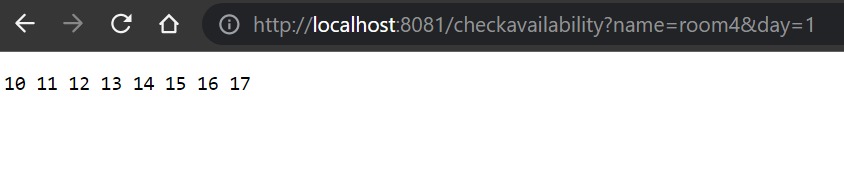
**Trying to add name with same name:**

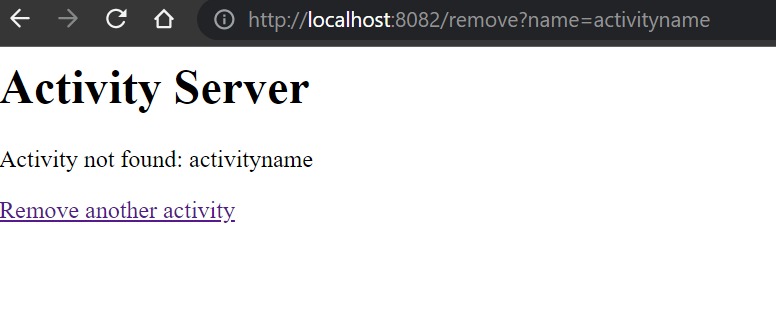
****

**Remove Name: **

**Room Reserve:**

****

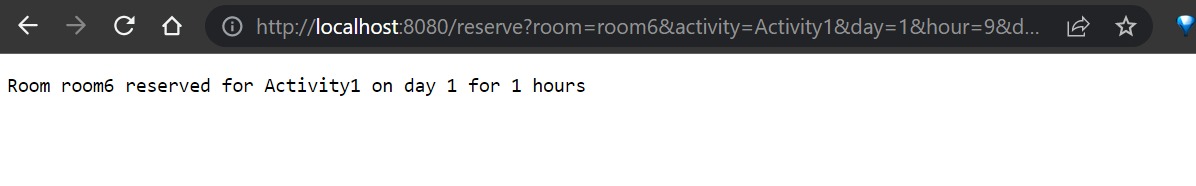
**Checking availability for given room:**

Checking Activity name for existance

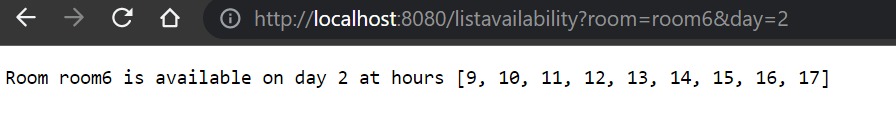
Removing Activity



Reserving Room



List availability



Listing availability for room in a week



Displaying reservation by ID

