

**MARMARA UNIVERSITY
FACULTY OF ENGINEERING**

**DEPARTMENT OF
COMPUTER ENGINEERING**



CSE4074 Computer Networks
Programming Assignment Report

**SOCKET PROGRAMMING
HTTP - based Room Reservation**

150119825 - Merve Rana Kızıl
150517059 - Özge Saltan

Aim:

Our aim in this project is to reach room server, activity server, and reservation server using socket programming and try to do the desired actions. While room server and activity server can work on their own, reservation server can work by getting information from room server and activity server.

Socket Programming:

To open a socket in a networked environment, we use a socket library in Python.

```
import socket

# create a socket object

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name

host = socket.gethostname()

port = 8080

# connection to hostname on the port.

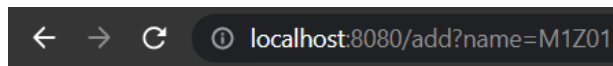
s.connect((host, port))
```

This code creates a socket and connects to the specified host and port. The socket will then be able to send and receive data over the network.

Room Server:

There are four operations for the room server. These are adding a room, removing the room, reserving the room, and checking the availability of the room. A JSON file named rooms.json is used to maintain a simple database for the operations of rooms. This server uses port number 8080. Supported GET requests are listed below.

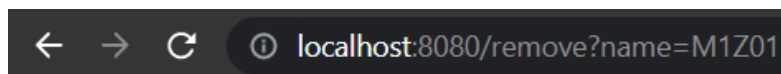
/add?name=roomname: This request adds a new room to the database if it does not exist and it returns the message “Room added” on the browser. Screenshots of the JSON file and browser after the room addition operation are given below. Room M2Z08 was added to the JSON file previously.



Room added

```
{ rooms.json > ...
1  {
2    "rooms": [
3      {
4        "name": "M2Z08",
5        "reservation_info": []
6      },
7      {
8        "name": "M1Z01",
9        "reservation_info": []
10     }
11   ]
12 }
```

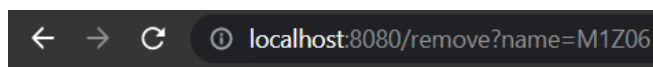
/remove?name=roomname: This request removes the new room from the database if it exists and it returns the message “Room removed” on the browser. Screenshots of the JSON file and browser after the room removal operation are given below.



Room removed

```
{ } rooms.json > ...
1 {
2   "rooms": [
3     {
4       "name": "M2Z08",
5       "reservation_info": []
6     }
7   ]
8 }
```

If the room does not exist in the database, it returns the message “Room does not exist.” on the browser. A screenshot is given below.

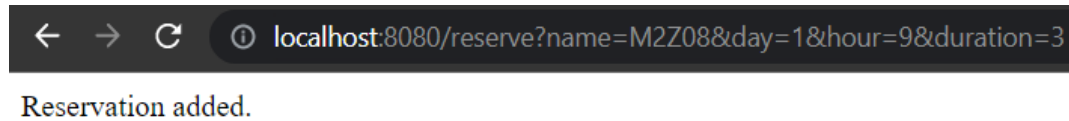


← → ↻ ⓘ localhost:8080/remove?name=M1Z06

Room does not exist.

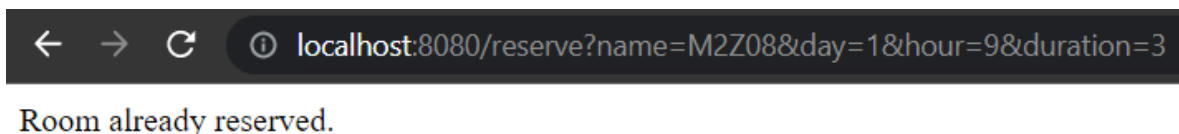
```
{ } rooms.json > ...
1 {
2   "rooms": [
3     {
4       "name": "M2Z08",
5       "reservation_info": []
6     }
7   ]
8 }
```

/reserve?name=roomname&day=x&hour=y&duration=z: This request reserves the room by using the variables provided and adds the reservation information to the database. It returns the message “Reservation added.” on the browser if the room is available. Screenshots of the JSON file and browser after the reservation addition operation are given below.

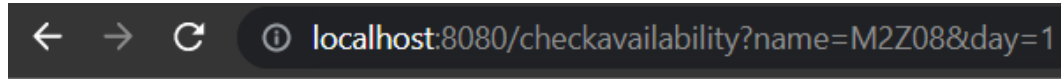


```
{
  "rooms": [
    {
      "name": "M2Z08",
      "reservation_info": [
        {
          "day": 1,
          "hour": [
            9,
            10,
            11
          ],
          "duration": 3
        }
      ]
    }
  ]
}
```

If the room is not available, it returns the message “Room already reserved.” on the browser. A screenshot for the unavailable room is given below.

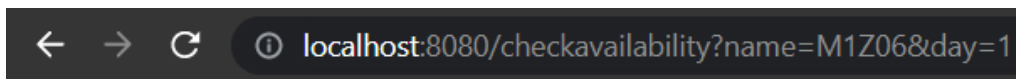


/checkavailability?name=roomname&day=x: This request checks the available hours for the provided room name and returns the available hours on the browser. A screenshot of the browser is given below.



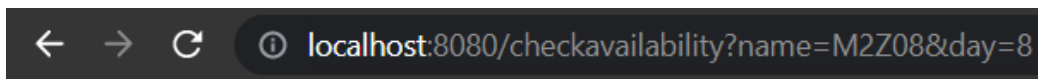
Reservation available. Available hours: 12, 13, 14, 15, 16

If the room does not exist in the database, it returns the message “Room does not exist.” on the browser. A screenshot is given below.



Room not found.

If the day is not a valid input, it returns the message “Day is not valid.” on the browser. A screenshot is given below.

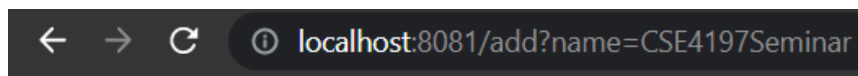


Day is not valid.

Activity Server:

There are three operations for the activity server. These are adding a new activity, removing the activity, and checking the activity. A JSON file named activities.json is used to maintain a simple database for the operations of rooms. This server uses port number 8081. Supported GET requests are listed below.

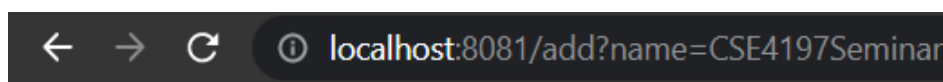
/add?name=activityname: This request adds a new activity if it does not exist in the database. A screenshot of the browser is given below. Activity “Presentation” was added to the database previously.



Activity added

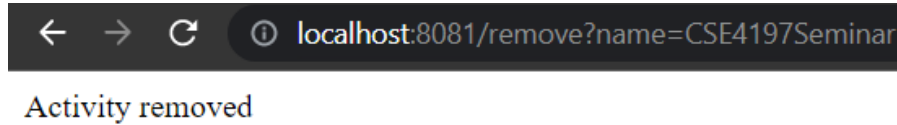
```
{ } activities.json > ...
1  {
2      "activities": [
3          {
4              "name": "CSE4197Seminar"
5          },
6          {
7              "name": "Presentation"
8          }
9      ]
10 }
```

If the activity already exists in the database the message “Activity already exists.” is returned on the browser. A screenshot is given below for the addition of the previously added activity.



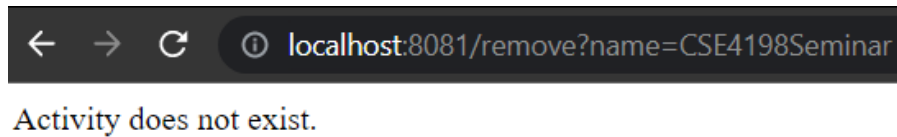
Activity already exists.

/remove?name=activityname: This request removes the activity with the provided name if it exists in the database. A screenshot of the browser is given below. Activity “Presentation” was added to the database previously.

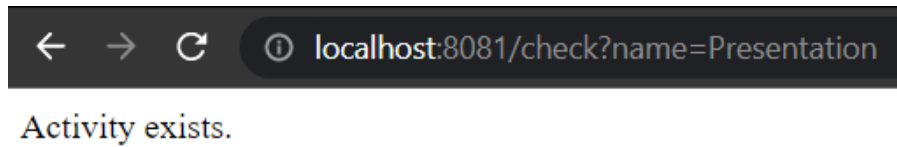


```
{ } activities.json > ...
1  {
2    "activities": [
3      {
4        "name": "Presentation"
5      }
6    ]
7  }
```

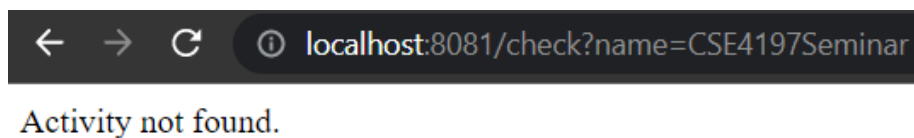
If the activity does not exist in the database the message “Activity does not exist.” is returned on the browser.



/check?name=activityname: This request checks whether there exists an activity with the provided name. A screenshot of the browser is given below.



If the activity does not exist in the database the message “Activity not found.” is returned on the browser.



Reservation Server:

There are four operations here. These are reserving a room, list availability of the room using roomname and day and only roomname, and display id. A JSON file named reservations.json is used to maintain a simple database for the operations of rooms. This server uses port number 8082. Supported GET requests are listed below.

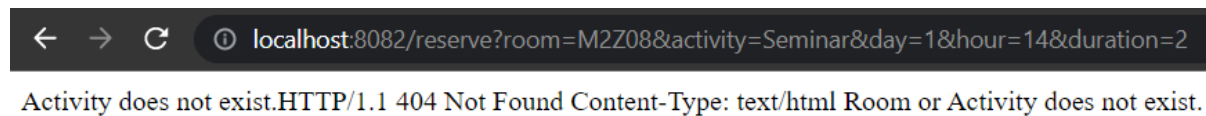
/reserve?room=roomname&activity=activityname&day=x&hour=y&duration=z: When the request comes from the server, it communicates with the Activity Server and checks the presence of the entered activityname. If available, it communicates with the Room Server and an attempt is made to reserve a room. If the room is reserved, a reservation_id is generated.

When the appropriate activity name and room name are entered and the day, hour and duration are correct, the room name can be reserved. It is written to the JSON file. You can see the screenshot below.

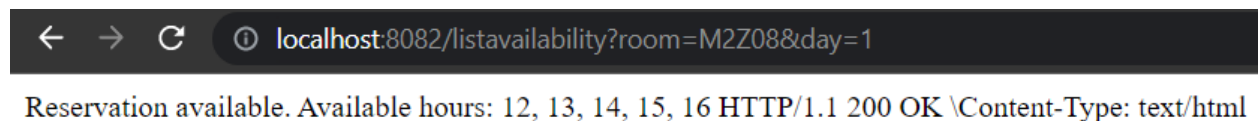


```
{ reservations.json > ...
1  {
2    "reservation_ids": [
3      1
4    ],
5    "reservations": [
6      {
7        "room": "M2Z08",
8        "reservation_info": [
9          {
10           "activity": "Presentation",
11           "day": 1,
12           "hour": [
13             14,
14             15
15           ],
16           "duration": 2,
17           "reservation_id": 1
18         }
19       ]
20     }
21   ]
22 }
```

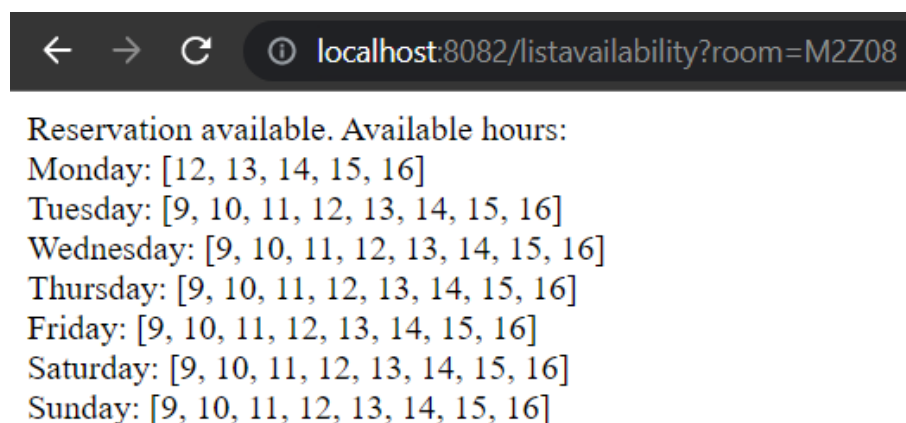
If there is no activity or room name, it informs that it does not exist. You can see the screenshot below.



/listavailability?room=roomname&day=x: Lists the entered roomname and all available hours for the day after contacting Room Server. When an existing room and day are entered, it shows the available hours of the room. You can see the screenshot below.

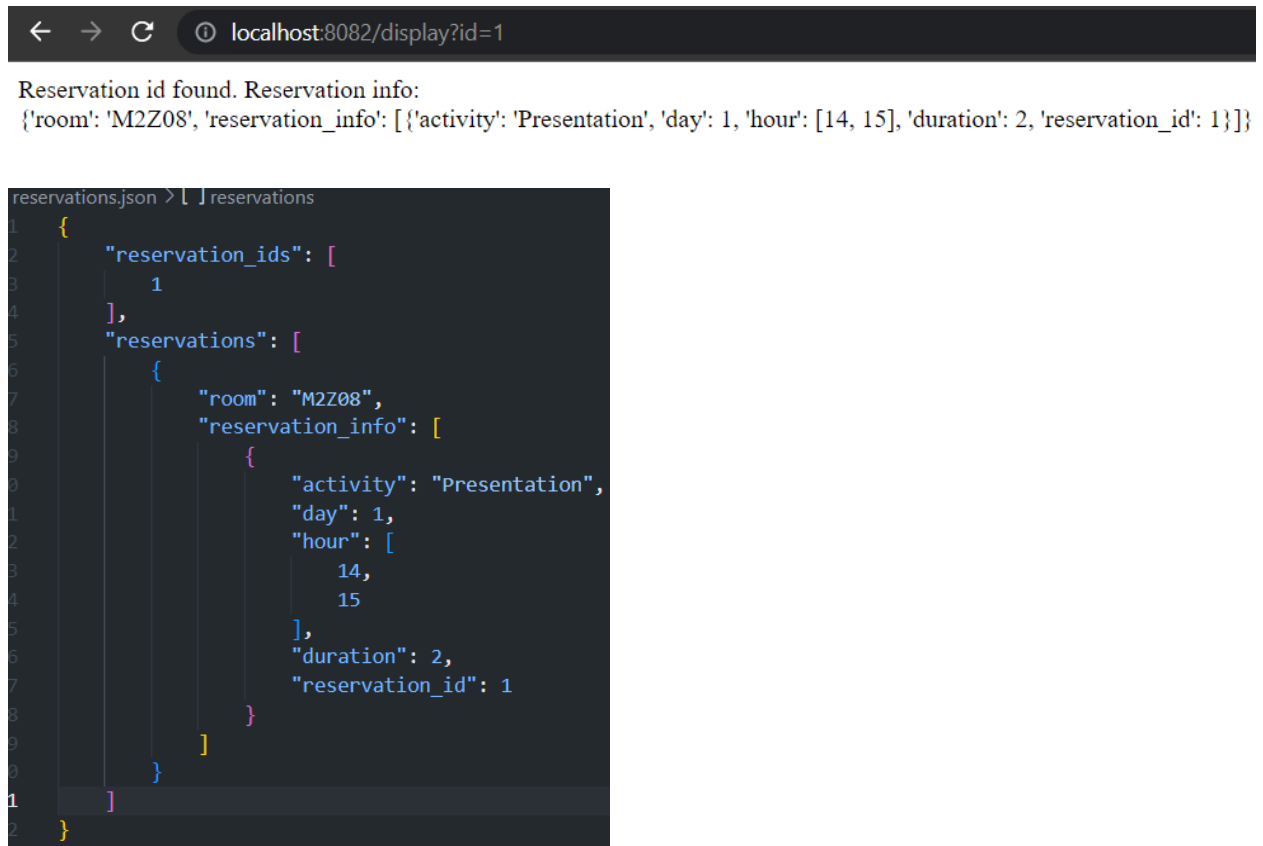


/listavailability?room=roomname: Lists the entered roomname write the all available hours in the all days after contacting Room Server. When an existing room is entered, it shows the available hours in all days of the room. You can see the screenshot below.



/display?id=reservation_id: Returns the details of the reservation that has the ID of the added reservation. If it does not exist, it returns an HTTP 404 Not Found message.

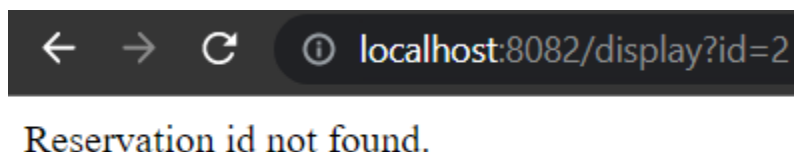
If there is an ID value entered, it informs that it is found and the information is printed. You can see this in the screenshot.



The screenshot shows a web browser at `localhost:8082/display?id=1` displaying the message: "Reservation id found. Reservation info: {'room': 'M2Z08', 'reservation_info': [{'activity': 'Presentation', 'day': 1, 'hour': [14, 15], 'duration': 2, 'reservation_id': 1}]}". Below the browser, a JSON viewer shows the structure of the `reservations.json` file, which contains an array of reservation objects. The first object matches the data shown in the browser.

```
reservations.json > [ ] reservations
{
  "reservation_ids": [
    1
  ],
  "reservations": [
    {
      "room": "M2Z08",
      "reservation_info": [
        {
          "activity": "Presentation",
          "day": 1,
          "hour": [
            14,
            15
          ],
          "duration": 2,
          "reservation_id": 1
        }
      ]
    }
  ]
}
```

If there is an ID value entered, but ID does not exist, give a response Reservation ID not found. You can see this in the screenshot.



The screenshot shows a web browser at `localhost:8082/display?id=2` displaying the message: "Reservation id not found."