MARMARA UNIVERSITY

FACULTY OF ENGINEERING DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CSE 4074- COMPUTER NETWORKS
SOCKET PROGRAMMING - HTTP BASED ROOM RESERVATION

STUDENTS	**NUMBERS**
Serkan Koç	150118073
İbrahim Hakkı Candan	150118061
Atila İlhan Yatağan	150118033

Submitted To:
Asst.Prof. Ömer Korçak

Due Date:
03.01.2023

Introduction

In this project we have implemented an HTTP server that acts as a room reservation server. To do this we have created a room server and activity server. Then we created a reservation server which is a proxy server. A user connects to the reservation server only. According to the incoming request, the reservation server sends relative requests to the room server and activity server and responses are sent back to the user via reservation server.

To implement this project we used Python 3.11 and TinyDB package as database for each server. For socket programming there is a package which is socket package. TinyDB is a txt representative database.

Also for simplicity and maintainability we have used microservice architecture.

Detailed System Design

Room Server

room.py

This is a room class where we can create an instance for the room. It contains a constructor which has attributes such as name and available hours. Available hours are set by default as a 2D array which contains hours starting from Monday to Sunday.

room_db.json

This is created by TinyDB which holds the data for created room instances.

main.py

In main we defined socket host and port. Server port is 7005. We created a socket then bind it to the host and port address. After that we started listening to incoming connection requests. When a client connects to the socket then he sends a request. This request is parsed and sent to room_controller.py. Then a response is sent back to main.py which will be sent to the user.

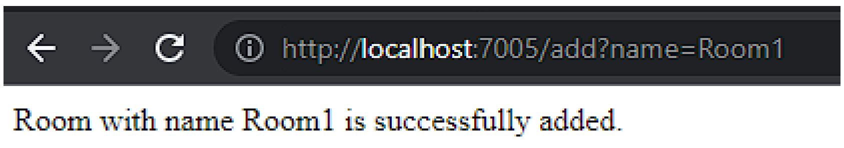
room_controller.py

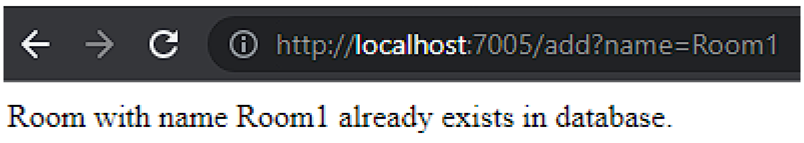
There are four methods in this class. They handle the request coming from main.py and send back an HTML response message. To send an HTML message it calls the methods in room_http_messages.py.

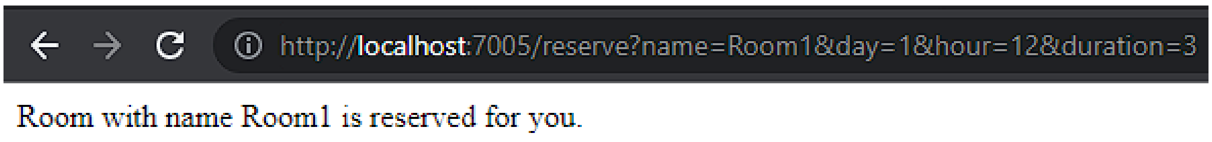
room_http_messages.py

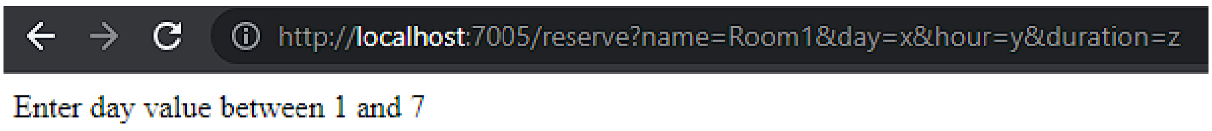
This class contains methods that return HTML strings according to the coming request and status of HTTP codes.

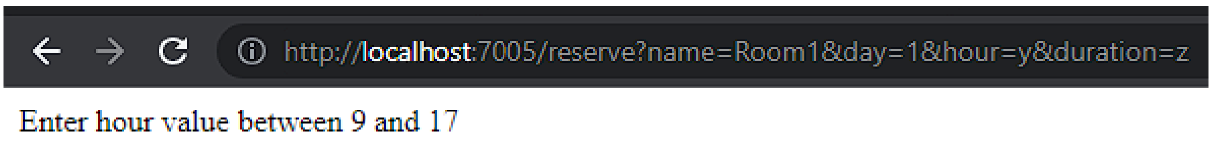
Screenshots of the Requests in Room Server

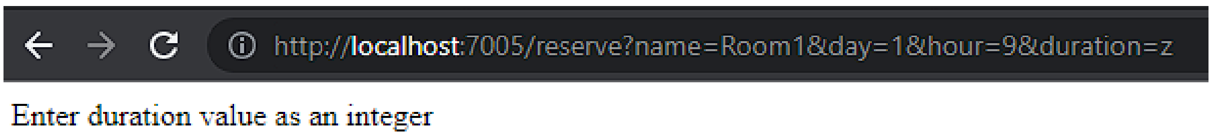

```
http://localhost:7005/add?name=Room1
Room with name Room1 is successfully added.
```

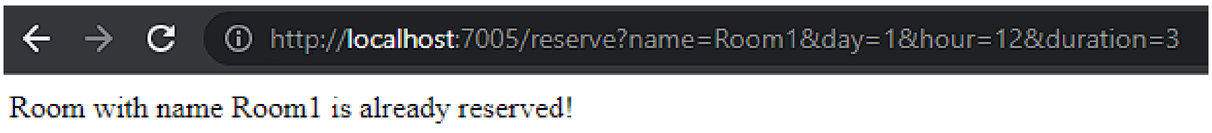

```
http://localhost:7005/add?name=Room1
Room with name Room1 already exists in database.
```

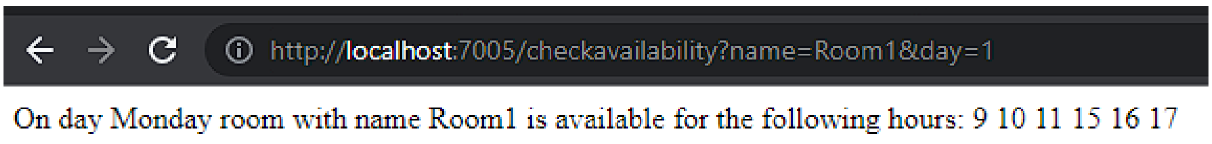

```
http://localhost:7005/reserve?name=Room1&day=1&hour=12&duration=3
Room with name Room1 is reserved for you.
```

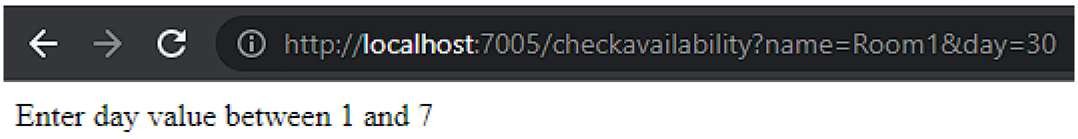

```
http://localhost:7005/reserve?name=Room1&day=x&hour=y&duration=z
Enter day value between 1 and 7
```

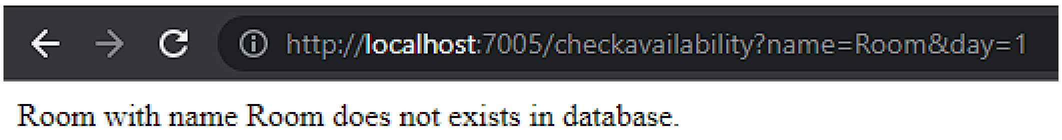

```
http://localhost:7005/reserve?name=Room1&day=1&hour=y&duration=z
Enter hour value between 9 and 17
```

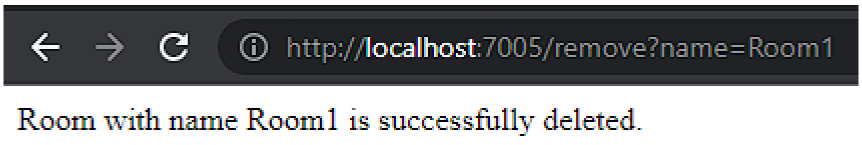

```
http://localhost:7005/reserve?name=Room1&day=1&hour=9&duration=z
Enter duration value as an integer
```


```
http://localhost:7005/reserve?name=Room1&day=1&hour=12&duration=3
Room with name Room1 is already reserved!
```

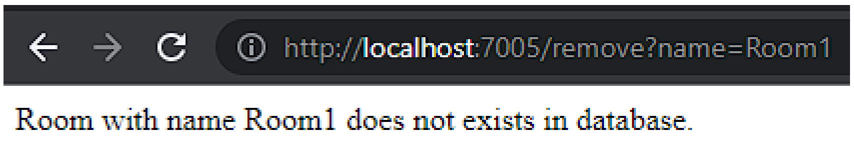

```
http://localhost:7005/checkavailability?name=Room1&day=1
On day Monday room with name Room1 is available for the following hours: 9 10 11 15 16 17
```


```
http://localhost:7005/checkavailability?name=Room1&day=30
Enter day value between 1 and 7
```


```
http://localhost:7005/checkavailability?name=Room&day=1
Room with name Room does not exists in database.
```


http://localhost:7005/remove?name=Room1

Room with name Room1 is successfully deleted.


http://localhost:7005/remove?name=Room1

Room with name Room1 does not exists in database.

Activity Server

activity.py

This is an activity class where we can create an instance for the activity. It contains a constructor which has an attribute name .

activity_db.json

This is created by TinyDB which holds the data for created activity instances.

main.py

In main we defined socket host and port. Server port is 7003. We created a socket then bind it to the host and port address. After that we started listening to incoming connection requests. When a client connects to the socket then he sends a request. This request is parsed and sent to activity_controller.py. Then a response is sent back to main.py which will be sent to the user.

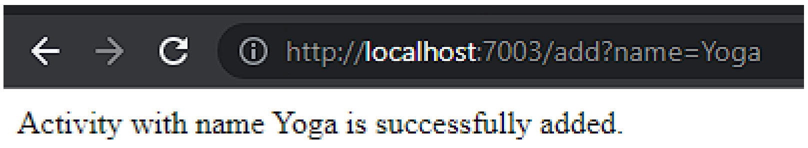
activity_controller.py

There are three methods in this class. They handle the request coming from main.py and send back an HTML response message. To send an HTML message it calls the methods in activity_http_messages.py.

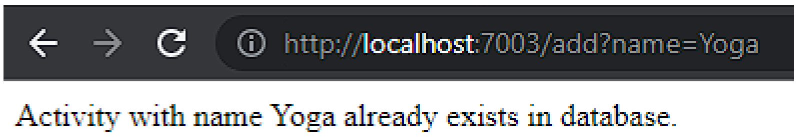
activity_http_messages.py

This class contains methods that return HTML strings according to the coming request and status of HTTP codes.

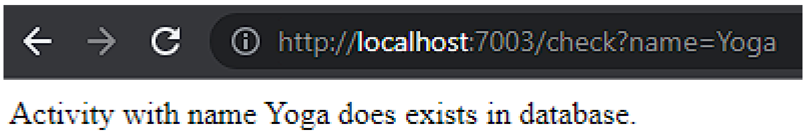
Screenshots of the Requests in Activity Server


http://localhost:7003/add?name=Yoga

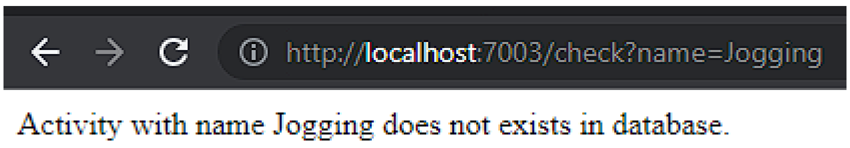
Activity with name Yoga is successfully added.


http://localhost:7003/add?name=Yoga

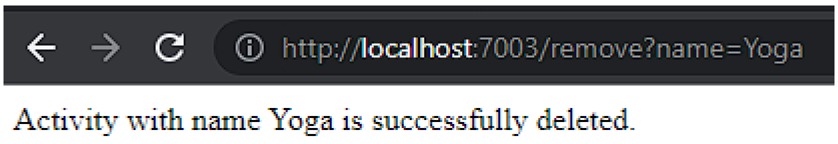
Activity with name Yoga already exists in database.



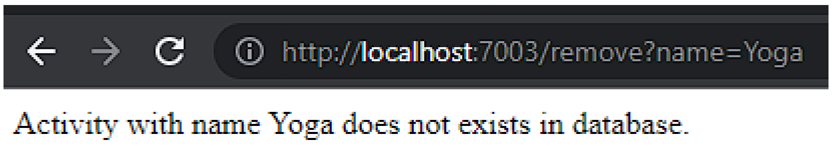
```
http://localhost:7003/check?name=Yoga
Activity with name Yoga does exists in database.
```



```
http://localhost:7003/check?name=Jogging
Activity with name Jogging does not exists in database.
```



```
http://localhost:7003/remove?name=Yoga
Activity with name Yoga is successfully deleted.
```



```
http://localhost:7003/remove?name=Yoga
Activity with name Yoga does not exists in database.
```

reservation.py

This is a reservation class where we can create an instance for the reservation. It contains a constructor which has six attributes as name, activity, days, hour duration and id .

reservation_db.json

This is created by TinyDB which holds the data for created reservation instances.

main.py

In main we defined socket host and port. Server port is 7004. We created a socket then bind it to the host and port address. After that we started listening to incoming connection requests. When a client connects to the socket then he sends a request. This request is parsed and sent to reservation_controller.py. In the reservation controller, we create a socket and send relative url and listen to either the room server or activity server socket. According to the response http message redirected to screen.

reservation_controller.py
There are three methods in this class.

```
def list_availability(params,client_address,client_connection):
```

This function lists the available hours of a given day via reaching the room controller’s list availability function. if the day is not specified it lists all days of the week.

```
def reserve(params,client_address,client_connection):
```

This function reserves an activity on the specified room within the given hours. First it contacts the activity controller, checks for whether the given activity exists or not. If the given activity does not exist in the database it sends back relative http messages to the user.
If the given activity exists, then it contacts the room controller and is available in the given hours, day or not. if all conditions are true it reserves and stores it to the database.

```
def display(params,conn):
```

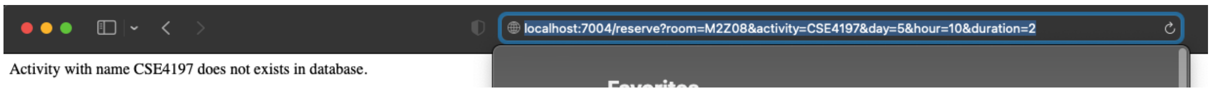
This function lists the reservation that is done on the databases with the given id. After that, it sends back relative http messages to the user.

These functions handle the request coming from main.py and send back an HTML response message. To send an HTML message it calls the methods in reservation_http_messages.py.

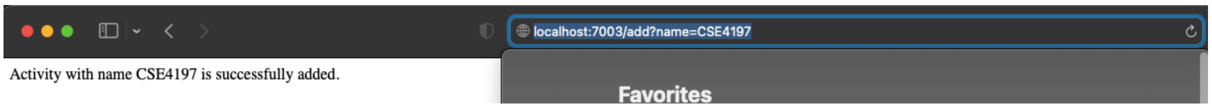
reservation_http_messages.py

This class contains methods that return HTML strings according to the coming request and status of HTTP codes.

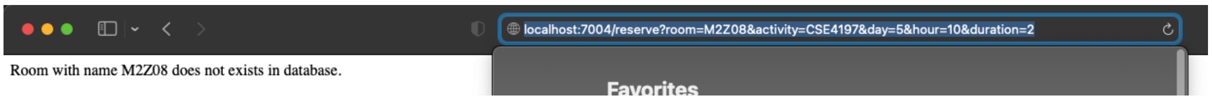
Screenshots of the Requests in Reservation Server


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=5&hour=10&duration=2

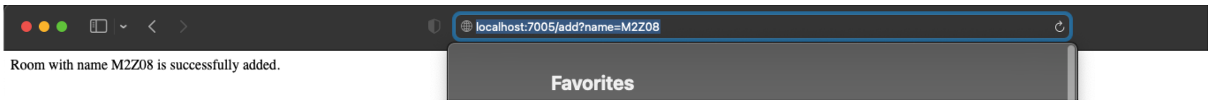
Activity with name CSE4197 does not exists in database.


localhost:7003/add?name=CSE4197

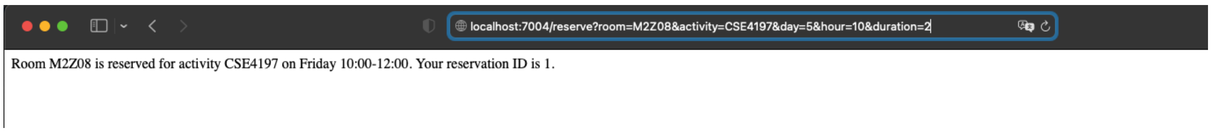
Activity with name CSE4197 is successfully added.


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=5&hour=10&duration=2

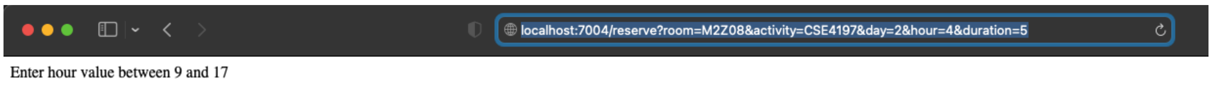
Room with name M2Z08 does not exists in database.


localhost:7005/add?name=M2Z08

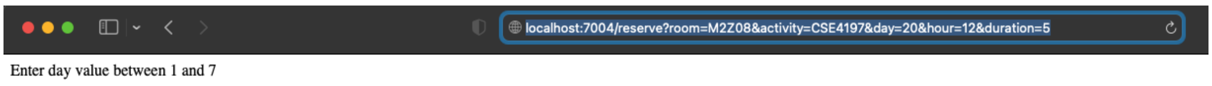
Room with name M2Z08 is successfully added.


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=5&hour=10&duration=2

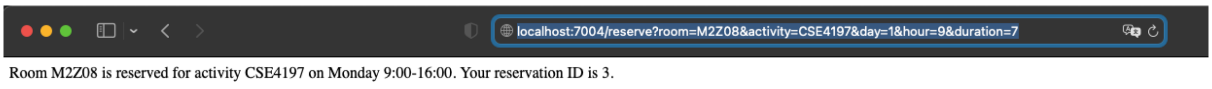
Room M2Z08 is reserved for activity CSE4197 on Friday 10:00-12:00. Your reservation ID is 1.


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=2&hour=4&duration=5

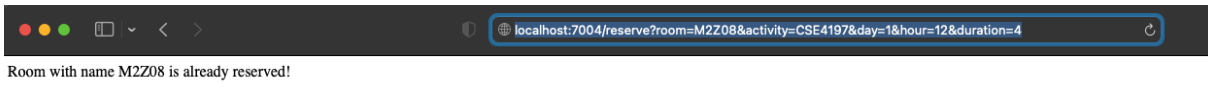
```
Enter hour value between 9 and 17
```


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=20&hour=12&duration=5

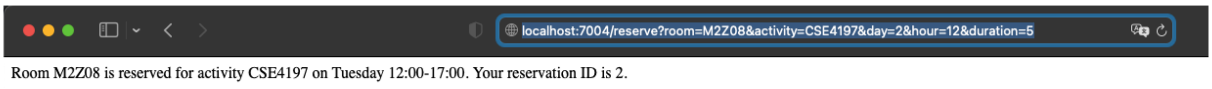
```
Enter day value between 1 and 7
```


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=1&hour=9&duration=7

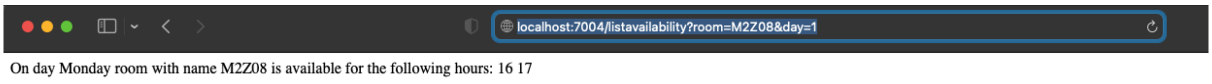
```
Room M2Z08 is reserved for activity CSE4197 on Monday 9:00-16:00. Your reservation ID is 3.
```


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=1&hour=12&duration=4

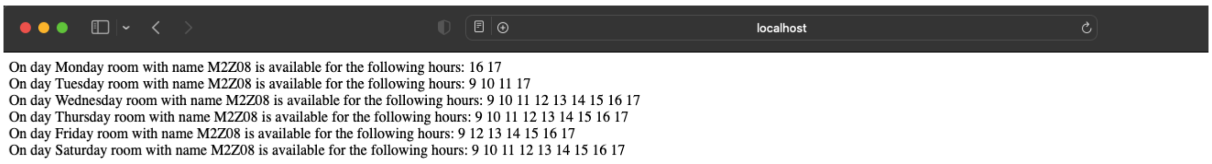
```
Room with name M2Z08 is already reserved!
```


localhost:7004/reserve?room=M2Z08&activity=CSE4197&day=2&hour=12&duration=5

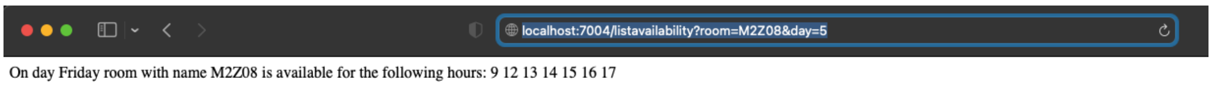
```
Room M2Z08 is reserved for activity CSE4197 on Tuesday 12:00-17:00. Your reservation ID is 2.
```


localhost:7004/listavailability?room=M2Z08&day=1

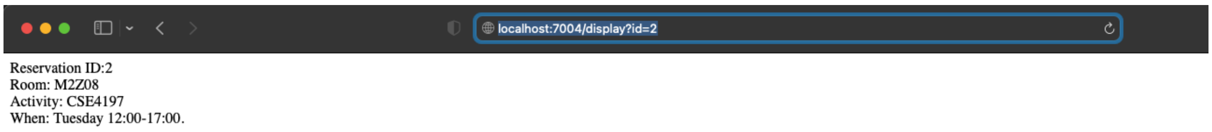
```
On day Monday room with name M2Z08 is available for the following hours: 16 17
```


localhost

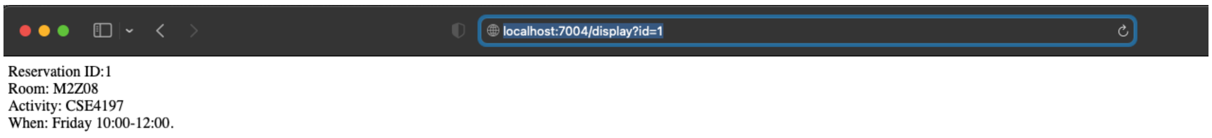
```
On day Monday room with name M2Z08 is available for the following hours: 16 17
On day Tuesday room with name M2Z08 is available for the following hours: 9 10 11 17
On day Wednesday room with name M2Z08 is available for the following hours: 9 10 11 12 13 14 15 16 17
On day Thursday room with name M2Z08 is available for the following hours: 9 10 11 12 13 14 15 16 17
On day Friday room with name M2Z08 is available for the following hours: 9 12 13 14 15 16 17
On day Saturday room with name M2Z08 is available for the following hours: 9 10 11 12 13 14 15 16 17
```


localhost:7004/listavailability?room=M2Z08&day=5

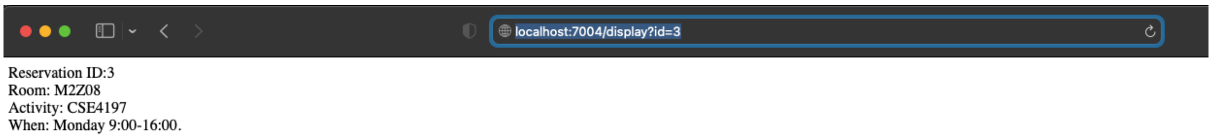
```
On day Friday room with name M2Z08 is available for the following hours: 9 12 13 14 15 16 17
```


localhost:7004/display?id=2

```
Reservation ID:2
Room: M2Z08
Activity: CSE4197
When: Tuesday 12:00-17:00.
```


localhost:7004/display?id=1

```
Reservation ID:1
Room: M2Z08
Activity: CSE4197
When: Friday 10:00-12:00.
```


localhost:7004/display?id=3

```
Reservation ID:3
Room: M2Z08
Activity: CSE4197
When: Monday 9:00-16:00.
```