



In Business since 1995

# Service Layer

Majed Hassan

November 05, 2023

## Overview

### Route Layer:

In the route layer of the backend of this hotel website, this will create particular paths (URL endpoints) that the frontend can use to communicate with the server. These routes are accountable for processing HTTP requests and directing them to the relevant controllers in the system. This can provide access to certain information and capabilities related to hotels, bookings, and user administration by defining routes such as `/api/user`, `/api/bookings`, and `/api/category`, for example. These routes can be used to provide access to specific information.

### Controller Layer:

In the controller layer, this will implement the logic that will be used to handle incoming HTTP requests and return appropriate responses to the frontend of the application. Each route that establishes in the route layer will eventually lead to a particular controller, also known as an endpoint. This may, for instance, have a `hotelsController` that handles requests pertaining to hotels, a `bookingsController` that manages bookings, and a `usersController` that handles actions pertaining to users. These controllers take in incoming requests, perform necessary data processing, contact the appropriate service methods, and then relay the results of their work to the client. The controllers act as a connecting point between the routes and the service layer of the network.

### Service Layer:

The service layer of the hotel website backend and the business logic of the website. Each controller is responsible for delegating its activities to the appropriate service methods. In order

to obtain, alter, or otherwise work with the data, the service methods connect with the database, which serves as the repository layer. For instance, this may have methods within the `hotelsService` that allow this to retrieve hotel information, add new hotels, update current hotels, or delete hotels. In a similar way, the `bookingsService` would be in charge of handling booking-related tasks such as checking availability, making reservations, and canceling existing bookings. Using these service methods, this can be certain that the fundamental functionality of this application is kept entirely distinct from the routing and controller logic.

---

## Method: Post

---

*Purpose: New users or guests can be added to the hotel's system via the API's "create user" method. This key action enables account creation, which in turn enables reservation making, individualized service delivery, and data management for individual users.*

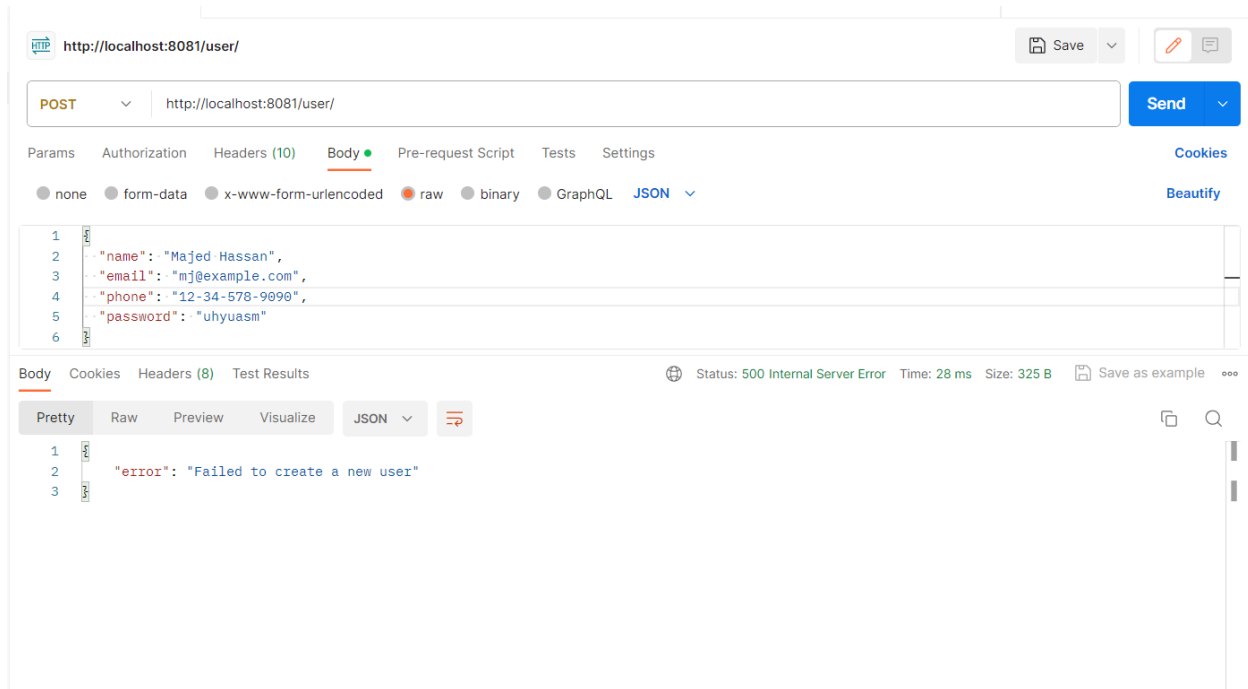
The screenshot displays a REST client interface with the following details:

- URL:** `http://localhost:8081/user/`
- Method:** `POST`
- Body (JSON):**

```
1 {
2   "name": "Majed Hassan",
3   "email": "mj@example.com",
4   "phone": "123457890",
5   "password": "uhyuasm"
6 }
```
- Status:** 200 OK, Time: 115 ms, Size: 317 B
- Response Body (JSON):**

```
1 {
2   "message": "User created successfully",
3   "userId": 0
4 }
```

## Error Handling



---

## Method: Get

---

*Purpose: In an application programming interface (API) designed for a hotel development project, the "get" method has the responsibility of getting information about the various types of rooms, services, and facilities that are provided by the hotel.*

⌵ Explore

Search Postman

Invite ⚙️ 🔔 🔴 Upgrade ⌵

⌵

⌵

GET http://localhost:8081/c

No Environment ⌵

⌵

http://localhost:8081/category

Save ⌵

✎

⌵

GET

http://localhost:8081/category

Send ⌵

💡

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key

Value

Description

⋮ Bulk Edit

Body

Cookies

Headers (8)

Test Results

Status: 200 OK Time: 9 ms Size: 934 B Save as example ⋮

Pretty

Raw

Preview

Visualize

JSON ⌵

⌵

⌵

⌵

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24


{

"name": "Business Class ",

"type": "Double Bed",

"cost": 1200,

"available": 11,

,"

"dec": "Non AC Room"

}

,

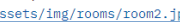
{

"name": "Business Class ",

"type": "Double Bed",

"cost": 2005,

"available": 9,

,"

"dec": "AC Room"

}

,


{

"name": "Business Class ",

"type": "Single Bed",

"cost": 800,

"available": 10,

,"

"dec": "Non AC Room"

}

Body

Cookies

Headers (8)

Test Results

Status: 200 OK Time: 9 ms Size: 934 B Save as example ⋮

Pretty

Raw

Preview

Visualize

JSON ⌵

⌵

⌵

⌵

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

"name": "Business Class ",

"type": "Single Bed",

"cost": 800,

"available": 10,

,"

"dec": "Non AC Room"

}

,

{

"name": "Business Class ",

"type": "Single Bed",

"cost": 1200,

"available": 10,

,"

"dec": "AC Room"

}

,

{

"name": "First class",

"type": "Double Bed",

"cost": 1357,

"available": 24,

,"

"dec": "This is a good room"

}

Postbot

Runner

Start Proxy

Cookies

Trash



http://localhost:8081/category

GET



http://localhost:8081/category

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
--	-----	-------

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {
2   "code": "ER_NO_SUCH_TABLE",
3   "errno": 1146,
4   "sqlMessage": "Table 'hotel.category1' doesn't exist",
5   "sqlState": "42S02",
6   "index": 0,
7   "sql": "select * from category1"
8 }
```

---

## Method: Get

---

*Purpose: Admins can access and edit their own profile information, as well as those of their users, with this feature.*

GET http://localhost:8081/u + No Environment

http://localhost:8081/user Save Send

GET http://localhost:8081/user

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body Cookies Headers (8) Test Results Status: 200 OK Time: 15 ms Size: 978 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "name": "admin",
4     "email": "admin@admin.com",
5     "phone": "1",
6     "password": "1"
7   },
8   {
9     "name": "Alice Johnson",
10    "email": "alice@example.com",
11    "phone": "5551234567",
12    "password": "secret1234"
13  },
14  {
15    "name": "Bob Williams",
16    "email": "bob@example.com",
17    "phone": "4448889999",
18    "password": "mypassword"
19  },
20  {
21    "name": "Eva Davis",
22    "email": "eva@example.com",
23    "phone": "7771113333",
24    "password": "pass123"
```

body Cookies Headers (8) Test Results Status: 200 OK Time: 15 ms Size: 978 B Save as example

Pretty Raw Preview Visualize JSON

```
22    "email": "eva@example.com",
23    "phone": "7771113333",
24    "password": "pass123"
25  },
26  {
27    "name": "Sabit",
28    "email": "iamsabit99@gmail.com",
29    "phone": "01744248058",
30    "password": "1"
31  },
32  {
33    "name": "Jane Smith",
34    "email": "janesmith@example.com",
35    "phone": "9876543210",
36    "password": "securepwd"
37  },
38  {
39    "name": "John Doe",
40    "email": "johndoe@example.com",
41    "phone": "1234567890",
42    "password": "123"
43  },
```

## Error Handling

GET http://localhost:8081/u + ... No Environment

http://localhost:8081/user Save

GET http://localhost:8081/user Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
-----	-------	-------------

Body Cookies Headers (8) Test Results Status: 200 OK Time: 53 ms Size: 414 B Save as example

Pretty Raw Preview Visualize JSON





```
1  {
2    "code": "ER_NO_SUCH_TABLE",
3    "errno": 1146,
4    "sqlMessage": "Table 'hotel.user1' doesn't exist",
5    "sqlState": "42S02",
6    "index": 0,
7    "sql": "select * from user1"
8  }
```



---



## Method: Put










---

*Purpose: To make it possible to change and update individual room numbers or details in the hotel's room management system, an API for a hotel project includes a "PUT updating rooms/room\_number" function.*




HTTP  http://localhost:8081/rooms/3  Save  



PUT  http://localhost:8081/rooms/3 

Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings 

 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON  





```
1 [{"id": 3, "room_number": 104}]
```



**Body** Cookies Headers (8) Test Results  Status: 200 OK Time: 439 ms Size: 313 B  Save as example 



Pretty Raw Preview Visualize **JSON**  










```
1 [{"message": "Room number updated successfully"}]
```

## Error Handling




HTTP  http://localhost:8081/rooms/5  Save  



PUT  http://localhost:8081/rooms/5 

Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings 

 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON  

```
1 [{"id": 3, "room_number": 104}]
```

**Body** Cookies Headers (8) Test Results  Status: 404 Not Found Time: 9 ms Size: 300 B  Save as example 

Pretty Raw Preview Visualize **JSON**  

```
1 [{"error": "Room not found"}]
```



---

## Method: Delete

---

*Purpose: The system is useful for keeping an updated and precise room count. The hotel's inventory is adjusted to reflect the reduction in available rooms after a room is removed.*

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8081/rooms/4`
- Method:** `DELETE`
- Status:** 200 OK
- Time:** 149 ms
- Size:** 306 B
- Body:**

```
{  "message": "Room deleted successfully"}
```

## Error Handling

The screenshot shows a REST client interface with the following details:

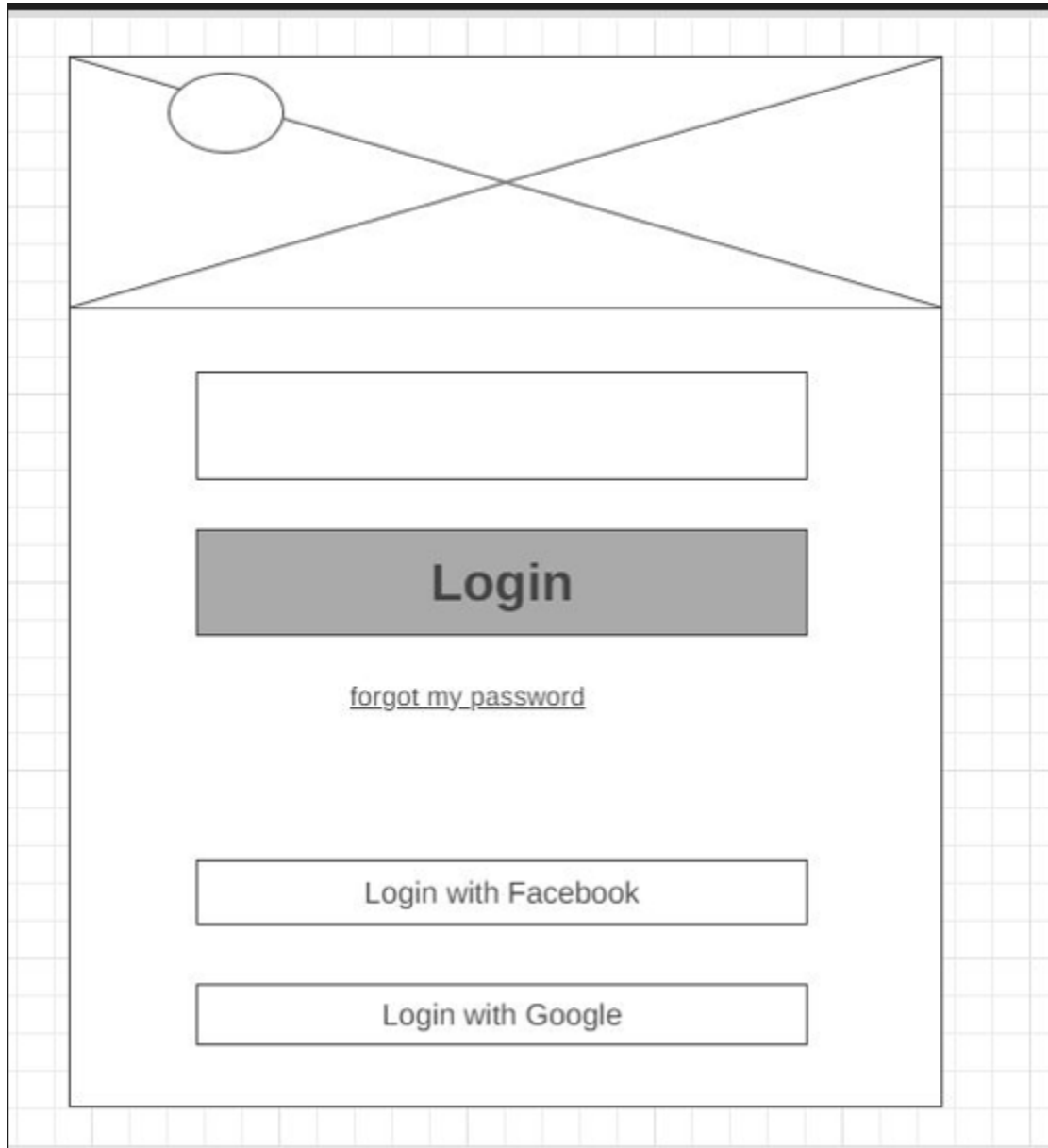
- URL:** `http://localhost:8081/rooms/5`
- Method:** `DELETE`
- Status:** 404 Not Found
- Time:** 13 ms
- Size:** 300 B
- Body:**

```
{  "error": "Room not found"}
```

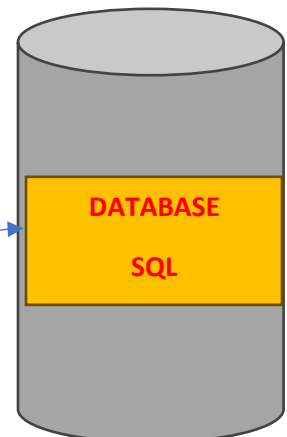
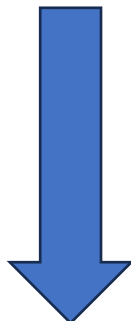
---

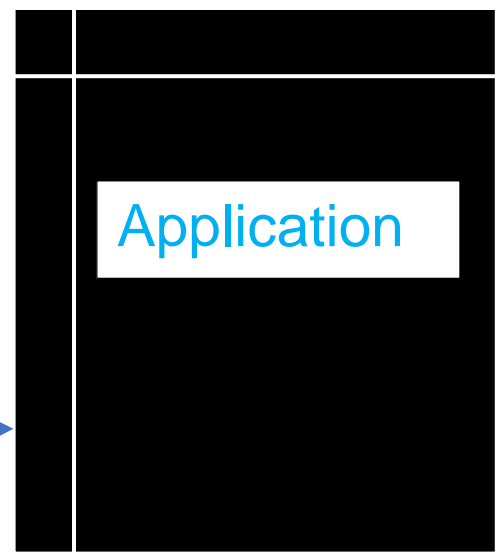
## *UI Design Planning / Implementation*

---



A wireframe diagram of a login page on a grid background. The page features a header area with a circular logo placeholder and a large 'X' across it. Below the header is a text input field, followed by a grey 'Login' button. Under the button is a link labeled 'forgot my password'. At the bottom are two buttons: 'Login with Facebook' and 'Login with Google'.



A wireframe of a web application interface. At the top, there is a header bar with a hamburger menu icon on the left, the text "Some daily updates with notifications." in the center, and a circular profile photo placeholder on the right labeled "Profile photo". Below the header is a large white content area. Inside this area, there is a long rectangular input field at the top. Below it is a section with several gray rectangular blocks of varying sizes, representing text or image placeholders. At the bottom of the content area is a form section. This section contains a dropdown menu with the text "Select" and a downward arrow, followed by the text "Select your options from the list". Below the dropdown is a horizontal slider control with a black knob. To the right of the slider is a gray button with the text "SUBMIT" in white, underlined.