

Task 2 - Hotel Room Service Request API

POST /requests to add a new service request

The screenshot shows a REST client interface for a "Room Service" API. The endpoint is `http://localhost:3000/api/requests` with a `POST` method. The request body is a JSON object: `{ "guestName": "Srin Doe", "roomNumber": 502, "requestDetails": "Repair fan", "priority": 1 }`. The response status is `201 Created` with a response time of 43 ms and a size of 1.2 KB. The response body is a JSON object: `{ "success": true, "message": "Request added successfully", "data": { "id": "1727692029558", "guestName": "Srin Doe", "roomNumber": 502, "requestDetails": "Repair fan", "priority": 1, "status": "received" } }`.

```
1 {
2   "guestName": "Srin Doe",
3   "roomNumber": 502,
4   "requestDetails": "Repair fan",
5   "priority": 1
6 }
```


```
1 {
2   "success": true,
3   "message": "Request added successfully",
4   "data": {
5     "id": "1727692029558",
6     "guestName": "Srin Doe",
7     "roomNumber": 502,
8     "requestDetails": "Repair fan",
9     "priority": 1,
10    "status": "received"
11  }
12 }
```

GET /requests to retrieve all requests, showing them sorted by priority.

The screenshot shows a REST client interface for a "Room Service" API. The endpoint is `http://localhost:3000/api/requests` with a `GET` method. The response status is `200 OK` with a response time of 14 ms and a size of 1.72 KB. The response body is a JSON object: `{ "success": true, "message": "Requests retrieved successfully", "data": [{ "id": "1727686524685", "guestName": "John Doe", "roomNumber": 101, "requestDetails": "Extra towels", "priority": 1, "status": "completed" }, { "id": "1727691123635", "guestName": "Srin Doe", "roomNumber": 502, "requestDetails": "Extra bed", "priority": 1, "status": "received" }, { "id": "1727692548611", "guestName": "Srin Joy", "roomNumber": 603, "requestDetails": "Repair shower", "priority": 2, "status": "received" }] }`.

```
1 {
2   "success": true,
3   "message": "Requests retrieved successfully",
4   "data": [
5     {
6       "id": "1727686524685",
7       "guestName": "John Doe",
8       "roomNumber": 101,
9       "requestDetails": "Extra towels",
10      "priority": 1,
11      "status": "completed"
12    },
13    {
14      "id": "1727691123635",
15      "guestName": "Srin Doe",
16      "roomNumber": 502,
17      "requestDetails": "Extra bed",
18      "priority": 1,
19      "status": "received"
20    },
21    {
22      "id": "1727692548611",
23      "guestName": "Srin Joy",
24      "roomNumber": 603,
25      "requestDetails": "Repair shower",
26      "priority": 2,
27      "status": "received"
28    }
29  ]
30 }
```

GET /requests/{id} to retrieve a specific request by its ID.

 Room Service / **get room service by id**

GET

http://localhost:3000/api/requests/1727689874615


Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Body Cookies Headers (21) Test Results 404 Not Found 25 ms 1.09 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": false,
3   "message": "Request not found for the id",
4   "data": null
5 }
```

 Room Service / **get room service by id**

GET

http://localhost:3000/api/requests/1727689874614

Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Body Cookies Headers (21) Test Results 200 OK 16 ms 1.21 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Request fetched successfully",
4   "data": {
5     "id": "1727689874614",
6     "guestName": "Srin Doe",
7     "roomNumber": 502,
8     "requestDetails": "Extra shampoo and pillows",
9     "priority": 3,
10    "status": "received"
11  }
12 }
```

PUT /requests/{id} to update details or the priority of an existing request.

The screenshot shows a REST client interface for a "Room Service" API. The endpoint is `http://localhost:3000/api/requests/1727692770460` and the method is `PUT`. The request body is a JSON object: `{ "status": "canceled", "priority": 4 }`. The response status is `200 OK` with a response time of 31 ms and a size of 1.2 KB. The response body is a JSON object: `{ "success": true, "message": "Request updated successfully", "data": { "id": "1727692770460", "guestName": "Srin Ghosh", "roomNumber": 803, "requestDetails": "Extra chips", "priority": 4, "status": "canceled" } }`.

```
1 {
2   "status": "canceled",
3   "priority": 4
4 }
```

```
1 {
2   "success": true,
3   "message": "Request updated successfully",
4   "data": {
5     "id": "1727692770460",
6     "guestName": "Srin Ghosh",
7     "roomNumber": 803,
8     "requestDetails": "Extra chips",
9     "priority": 4,
10    "status": "canceled"
11  }
12 }
```

DELETE /requests/{id} to remove a completed or canceled request.

The first screenshot shows a REST client interface for a "Room Service" API. The endpoint is `http://localhost:3000/api/requests/1727692029558` and the method is `DELETE`. The response status is `400 Bad Request` with a response time of 9 ms and a size of 1.12 KB. The response body is a JSON object: `{ "success": false, "message": "Cannot delete request unless it is completed or canceled", "data": null }`.

```
1 {
2   "success": false,
3   "message": "Cannot delete request unless it is completed or canceled",
4   "data": null
5 }
```

The second screenshot shows a REST client interface for a "Room Service" API. The endpoint is `http://localhost:3000/api/requests/1727692770460` and the method is `DELETE`. The response status is `200 OK` with a response time of 20 ms and a size of 1.08 KB. The response body is a JSON object: `{ "success": true, "message": "Request deleted successfully", "data": null }`.

```
1 {
2   "success": true,
3   "message": "Request deleted successfully",
4   "data": null
5 }
```

POST /requests/{id}/complete to mark a request as completed.

The screenshot shows a REST client interface for a 'Room Service' API. The endpoint is `POST http://localhost:3000/api/requests/1727692029558/complete`. The response is a 200 OK status with a 13 ms response time and 1.2 KB of data. The response body is a JSON object:

```
1 {
2   "success": true,
3   "message": "Request marked as completed",
4   "data": {
5     "id": "1727692029558",
6     "guestName": "Srin Doe",
7     "roomNumber": 502,
8     "requestDetails": "Repair fan",
9     "priority": 1,
10    "status": "completed"
11  }
12 }
```

Task 1 - Meeting Scheduler

Input 1

[[9, 12], [11, 13], [14, 17], [16, 18]]

Output 1

[[9, 13], [14, 18]]

Input 2

[]

Output 2

[]

Input 3

[

```
[1, 4], [5, 6], [7, 8], [9, 11], [10, 13],  
[14, 16], [15, 18], [17, 19], [20, 22], [21, 25],  
[24, 26], [27, 30], [29, 31], [32, 34], [33, 35],  
[36, 39], [38, 40], [41, 43], [42, 44], [45, 47],  
[46, 48], [49, 50], [51, 53], [52, 54], [55, 57],  
[56, 58], [59, 61], [60, 62], [63, 65], [64, 66]  
]
```

Output 3

```
[  
[ 1, 4 ], [ 5, 6 ],  
[ 7, 8 ], [ 9, 13 ],  
[ 14, 19 ], [ 20, 26 ],  
[ 27, 31 ], [ 32, 35 ],  
[ 36, 40 ], [ 41, 44 ],  
[ 45, 48 ], [ 49, 50 ],  
[ 51, 54 ], [ 55, 58 ],  
[ 59, 62 ], [ 63, 66 ]  
]
```

Input 4

```
[  
[ 0, 3 ], [ 2, 5 ], [ 4, 7 ], [ 6, 9 ], [ 8, 11 ],  
[ 10, 13 ], [ 12, 15 ], [ 14, 17 ], [ 16, 19 ], [ 18, 21 ],  
[ 20, 23 ], [ 22, 25 ], [ 24, 27 ], [ 26, 29 ], [ 28, 31 ],  
[ 30, 33 ], [ 32, 35 ], [ 34, 37 ], [ 36, 39 ], [ 38, 41 ],  
[ 40, 43 ], [ 42, 45 ], [ 44, 47 ], [ 46, 49 ], [ 48, 51 ],  
[ 50, 53 ], [ 52, 55 ], [ 54, 57 ], [ 56, 59 ], [ 58, 61 ],  
[ 60, 63 ], [ 62, 65 ], [ 64, 67 ], [ 66, 69 ], [ 68, 71 ],  
[ 70, 73 ], [ 72, 75 ], [ 74, 77 ], [ 76, 79 ], [ 78, 81 ],  
[ 80, 83 ], [ 82, 85 ], [ 84, 87 ], [ 86, 89 ], [ 88, 91 ],  
[ 90, 93 ], [ 92, 95 ], [ 94, 97 ], [ 96, 99 ], [ 98, 101 ],  
[ 100, 103 ], [ 102, 105 ], [ 104, 107 ], [ 106, 109 ], [ 108, 111 ],  
[ 110, 113 ], [ 112, 115 ], [ 114, 117 ], [ 116, 119 ], [ 118, 121 ],  
[ 120, 123 ], [ 122, 125 ], [ 124, 127 ], [ 126, 129 ], [ 128, 131 ],  
[ 130, 133 ], [ 132, 135 ], [ 134, 137 ], [ 136, 139 ], [ 138, 141 ],  
[ 140, 143 ], [ 142, 145 ], [ 144, 147 ], [ 146, 149 ], [ 148, 151 ],  
[ 150, 153 ], [ 152, 155 ], [ 154, 157 ], [ 156, 159 ], [ 158, 161 ],  
[ 160, 163 ], [ 162, 165 ], [ 164, 167 ], [ 166, 169 ], [ 168, 171 ],  
[ 170, 173 ], [ 172, 175 ], [ 174, 177 ], [ 176, 179 ], [ 178, 181 ],  
[ 180, 183 ], [ 182, 185 ], [ 184, 187 ], [ 186, 189 ], [ 188, 191 ],  
[ 190, 193 ], [ 192, 195 ], [ 194, 197 ], [ 196, 199 ], [ 198, 201 ],  
]
```

... 200 more items
]

Output 4

[[0, 601]]