

Step 1

```
JS Index1.js U •
Week 3 > JS Index1.js > ...
10  async function connectToMongoDB() {
18      } catch (err) {
19          console.error("Error connecting to MongoDB:", err);
20      }
21  }
22
23  connectToMongoDB();
24
25  app.listen(port, () => {
26      console.log(`Server running on port ${port}`);
27  });
28
29  app.get('/rides', async (req, res) => {
30      try {
31          const rides = await db.collection('rides').find().toArray();

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Wickson Goh\Desktop\Cloud System\Cloud\berr2243-2025gmc\Week 3> node index1.js
Server running on port 3000
Connected to MongoDB!
PS C:\Users\Wickson Goh\Desktop\Cloud System\Cloud\berr2243-2025gmc\Week 3> |
```

POST

POST http://localhost:3000/rides

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1  {
2      ... "pickupLocation": "Central Park",
3      ... "destination": "Times Square",
4      ... "driverId": "DRIVER123",
5      ... "status": "requested"
6  }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview

```
1  [{"id": "690c19c911ca62384a3abc5b"}]
```

POST (Output)

berr2243-2025gmc > testDB > rides

Documents 1 Aggregations Schema Indexes 1 Validation

🕒 ▾ Type a query: { field: 'value' } or [Generate query](#) ⚡

ADD DATA ▾ **EXPORT DATA** ▾ **UPDATE** **DELETE**

```
_id: ObjectId('690c19c911ca62384a3abc5b')
pickupLocation: "Central Park"
destination: "Times Square"
driverId: "DRIVER123"
status: "requested"
```

GET From MongoDB

GET <http://localhost:3000/rides>

Params Authorization Headers (7) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

1

Body Cookies Headers (7) Test Results Status: 200 OK Time: 6 ms Size:

Pretty Raw Preview **JSON** ▾

1 [{"_id":"690c19c911ca62384a3abc5b","pickupLocation":"Central Park","destination":"Times Square","driverId":"DRIVER123","status":"requested"}]

PATCH

PATCH <http://localhost:3000/rides/690c19c911ca62384a3abc5b>

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

1 {
2 "status": "cancelled"
3 }

Body Cookies Headers (7) Test Results

Pretty Raw Preview **JSON** ▾

1 [{"updated":1}]

berr2243-2025gmc > testDB > rides

Documents 1 Aggregations Schema Indexes 1 Validation

🕒 ▾ Type a query: { field: 'value' } or [Generate query](#) ⚡

ADD DATA ▾

EXPORT DATA ▾

UPDATE

DELETE

```
_id: ObjectId('690c19c911ca62384a3abc5b')
pickupLocation: "Central Park"
destination: "Times Square"
driverId: "DRIVER123"
status: "cancelled"
```

DELETE

DELETE

▼

http://localhost:3000/rides/690c19c911ca62384a3abc5b

ParamsAuthorizationHeaders (7)BodyScriptsSettings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQLJSON ▼

1

BodyCookiesHeaders (7)Test Results

PrettyRawPreviewJSON ▼

1 [{"deleted":1}]

berr2243-2025gmc > testDB > rides

Documents 0AggregationsSchemaIndexes 1Validation

🕒 ▼ Type a query: { field: 'value' } or [Generate query](#) ⚡

Explain

➕ ADD DATA ▼📄 EXPORT DATA ▼✎ UPDATE🗑️ DELETE

25 ▼ 0 – 0 of 0



This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

Import data

Lab Questions

Answer by testing your API in Postman and observing responses.

1. POST Request:

o What HTTP status code is returned when a ride is created successfully?

= **201 Created**

o What is the structure of the response body?

= **json({ id: result.insertedId })**

2. GET Request:

o What happens if the rides collection is empty?

= **empty array ([])**

o What data type is returned in the response (array/object)?

= **array**

3. Fix PATCH and DELETE Error:

o Catch the error when requesting PATCH or DELETE API, then try to fix the issue reported.

= **error (400 bad request)**

To fix, add the ObjectId.

```
const { MongoClient, ObjectId } = require('mongodb');
```

o If you try to update a non-existent ride ID, what status code is returned?

= **Invalid ride Id or data**

o What is the value of updated in the response if the update succeeds?

= **{"updated":1}**

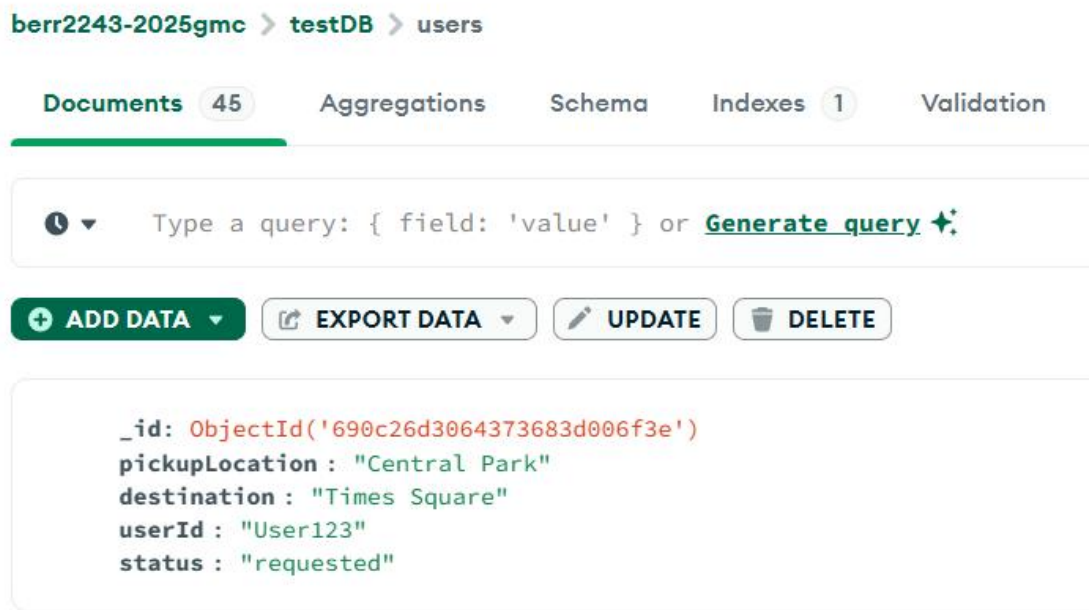
o How does the API differentiate between a successful deletion and a failed one?

= **if successful {"deleted":1} and status 200 OK if unsuccessful {"error":"Ride not found"} and status 404 not found.**

4. Users Endpoints:

o Based on the exercise above, create the endpoints to handle the CRUD operations for users account

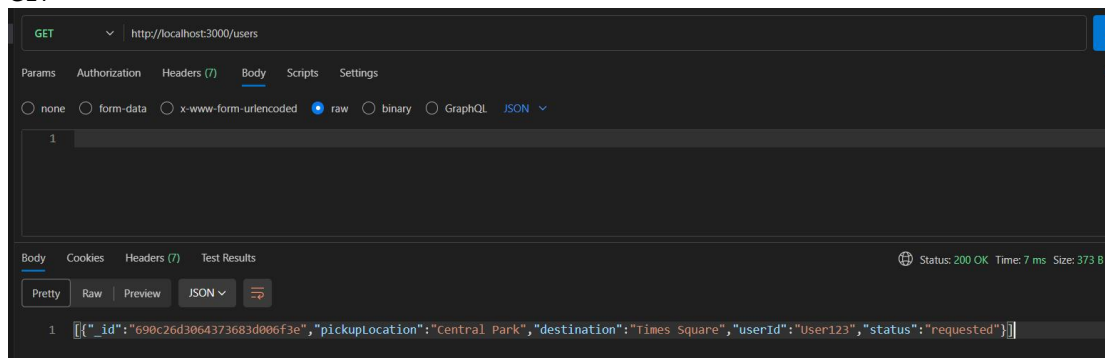
POST



The screenshot shows the MongoDB Compass interface. The breadcrumb path is **berr2243-2025gmc > testDB > users**. The 'Documents' tab is selected, showing 45 documents. Below the tabs is a search bar with a clock icon and the text 'Type a query: { field: 'value' } or [Generate query](#)'. Below the search bar are four buttons: 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. The main area displays a single document in a code editor:

```
_id: ObjectId('690c26d3064373683d006f3e')
pickupLocation: "Central Park"
destination: "Times Square"
userId: "User123"
status: "requested"
```

GET



The screenshot shows the Postman interface. The method is 'GET' and the URL is 'http://localhost:3000/users'. The 'Body' tab is selected. The response is shown in the 'Test Results' section, with a status of '200 OK', time of '7 ms', and size of '373 B'. The response body is a JSON array containing one object:

```
1 [{"_id": "690c26d3064373683d006f3e", "pickupLocation": "Central Park", "destination": "Times Square", "userId": "User123", "status": "requested"}]
```

PATCH

HTTP **PATCH** `http://localhost:3000/users/690c26d3064373683d006f3e`

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {  
2   ... "status": "Cancelled"  
3 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview **JSON** ▾

```
1 { "updated": 1 }
```

berr2243-2025gmc > testDB > users

Documents 1 Aggregations Schema Indexes 1 Validation

🕒 ▾ Type a query: { field: 'value' } or [Generate query](#) ⚡

ADD DATA ▾ **EXPORT DATA** ▾ **UPDATE** **DELETE**

```
_id: ObjectId('690c26d3064373683d006f3e')  
pickupLocation: "Central Park"  
destination: "Times Square"  
userId: "User123"  
status: "Cancelled"
```

5. FrontEnd:

- o Upload the Postman JSON to any AI tools, and generate a simple HTML and JS Dashboard for you

←

🔍

📁

File

C:\Users\Nickson\k20Goh\Desktop\Cloud\k20System\Cloud\beer2243-2025gmc\Week202\SDASH.html

Dashboard

Rides

Load Rides

ID	Pickup	Destination	Driver ID	Status	Actions
6912201957e3cb9cb5f1fc55	Central Park	Times Square	DRIVER123	Cancelled	<div><div>Update</div><div>Delete</div></div>

Add Ride

Add Ride

Users

Load Users

ID	Pickup	Destination	User ID	Status	Actions
690c26d3064373683d006f3e	Central Park	Times Square	User123	Cancelled	<div><div>Update</div><div>Delete</div></div>
691204c63b7b24b5faca97f3	Central Park	Times Square	user123	requested	<div><div>Update</div><div>Delete</div></div>

Add User

Add User