

# Postman Screenshots

## 1) Category endpoint( /api/categories )

### a) Creating a category

The screenshot shows the Postman interface for creating a new category. The URL is `http://localhost:8080/api/categories`. The request method is `POST`. The body contains the following JSON:

```
1 {  
2   "name": "Electronics",  
3   "description": "the category of devices runs on electricity."  
4 }
```

The response status is `201 Created`, indicating success. The response body is:

```
1 {  
2   "id": 1,  
3   "name": "Electronics",  
4   "description": "the category of devices runs on electricity."  
5 }
```

### b) Creating second category

The screenshot shows the Postman interface for creating a new category. The URL is `http://localhost:8080/api/categories`. The request method is `POST`. The body contains the following JSON:

```
1 {  
2   "name": "Dairy",  
3   "description": "these are basically milk based product."  
4 }
```

The response status is `201 Created`, indicating success. The response body is:

```
1 {  
2   "id": 2,  
3   "name": "Dairy",  
4   "description": "these are basically milk based product."  
5 }
```

### c) Getting all categories

HTTP <http://localhost:8080/api/categories>

GET <http://localhost:8080/api/categories>

Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Body	Cookies	Headers (5)	Test Results

{ } JSON ▾ ▶ Preview Visualize ▾

```
1 [  
2 {  
3   "id": 1,  
4   "name": "Electronics",  
5   "description": "the category of devices runs on electricity."  
6 },  
7 {  
8   "id": 2,  
9   "name": "Dairy",  
10  "description": "these are basically milk based product."  
11 }
```

### d) Deleting a category

HTTP <http://localhost:8080/api/categories/1>

DELETE <http://localhost:8080/api/categories/1>

Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (3) Test Results

{ } Raw ▾ ▶ Preview Visualize ▾

```
1
```

### e) Fetching the category

HTTP <http://localhost:8080/api/categories/2>

GET <http://localhost:8080/api/categories/2>

Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Body	Cookies	Headers (5)	Test Results

{ } JSON ▾ ▶ Preview Visualize ▾

```
1 {  
2   "id": 2,  
3   "name": "Dairy",  
4   "description": "these are basically milk based product."  
5 }
```

## 2) Product Endpoints( /api/products)

### a) Creating products

The screenshot shows the Postman interface for creating a product. The URL is `http://localhost:8080/api/products`. The request method is `POST`. The body contains the following JSON:

```
1 {  
2   "name": "Amul Milk",  
3   "price": 35,  
4   "quantity": 900,  
5   "categoryId": 2  
6 }
```

The response status is `201 Created` with a response time of 192 ms and a size of 346 B. The response body is:

```
1 {  
2   "id": 1,  
3   "name": "Amul Milk",  
4   "price": 35.0,  
5   "quantity": 900,  
6   "category_id": 2,  
7   "category_name": "Dairy",  
8   "createdAt": "2025-12-09T22:23:08.288268",  
9   "updatedAt": "2025-12-09T22:23:08.288268"
```

### b) Fetching product

The screenshot shows the Postman interface for fetching a specific product. The URL is `http://localhost:8080/api/products/4`. The request method is `GET`. The response status is `200 OK` with a response time of 17 ms and a size of 345 B. The response body is:

```
1 {  
2   "id": 4,  
3   "name": "Laptop",  
4   "price": 39578.8,  
5   "quantity": 1,  
6   "category_id": 1,  
7   "category_name": "Electronics",  
8   "createdAt": "2025-12-09T22:46:39.532484",  
9   "updatedAt": "2025-12-09T22:46:39.532484"  
10 }
```

### c) Updating a product

The screenshot shows the Postman interface for updating a product. The URL is `http://localhost:8080/api/products/4`. The method is set to `PUT`. The `Body` tab is selected, showing the following JSON payload:

```
1 {  
2   "name": "Amul Sweet Lassi",  
3   "price": 15,  
4   "quantity": 100,  
5   "categoryId": 2  
6 }
```

The response status is `200 OK` with a response time of `147 ms`, a size of `348 B`, and a timestamp of `2025-12-09T22:46:39.532484`.

### d) Deleting a product

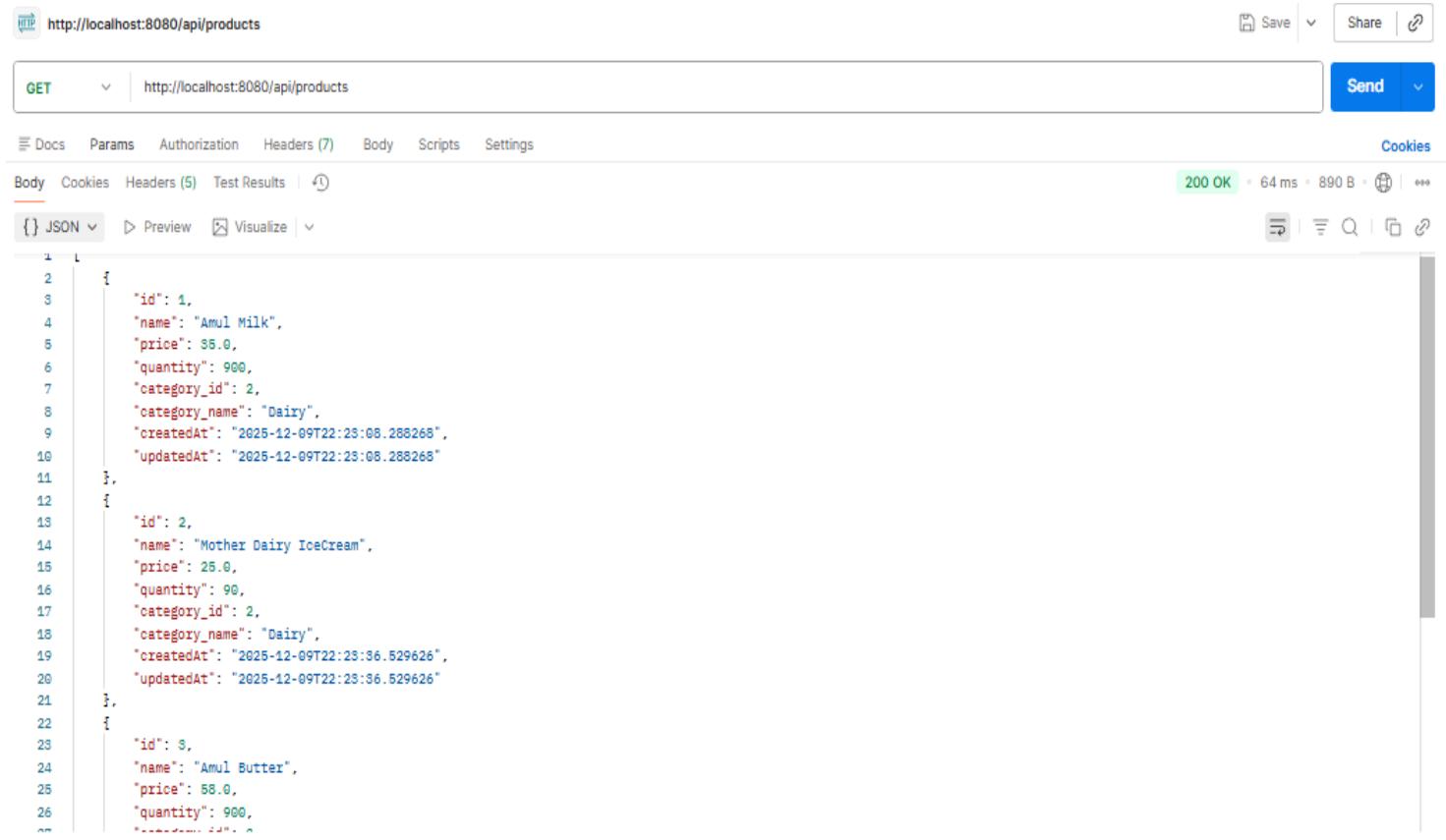
The screenshot shows the Postman interface for deleting a product. The URL is `http://localhost:8080/api/products/4`. The method is set to `DELETE`. The `Body` tab is selected, showing the following JSON payload:

```
1 {  
2   "id": 4,  
3   "name": "Amul Sweet Lassi",  
4   "price": 15.0,  
5   "quantity": 100,  
6   "category_id": 2,  
7   "category_name": "Dairy",  
8   "createdAt": "2025-12-09T22:46:39.532484",  
9   "updatedAt": "2025-12-09T22:48:05.713397"  
10 }
```

The response status is `204 No Content` with a response time of `125 ms`, a size of `112 B`, and a timestamp of `2025-12-09T22:48:05.713397`.

## e) Fetching using filters

### 1) Fetching all the products( all filters null)



HTTP <http://localhost:8080/api/products> Save Share ⌂

GET http://localhost:8080/api/products Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

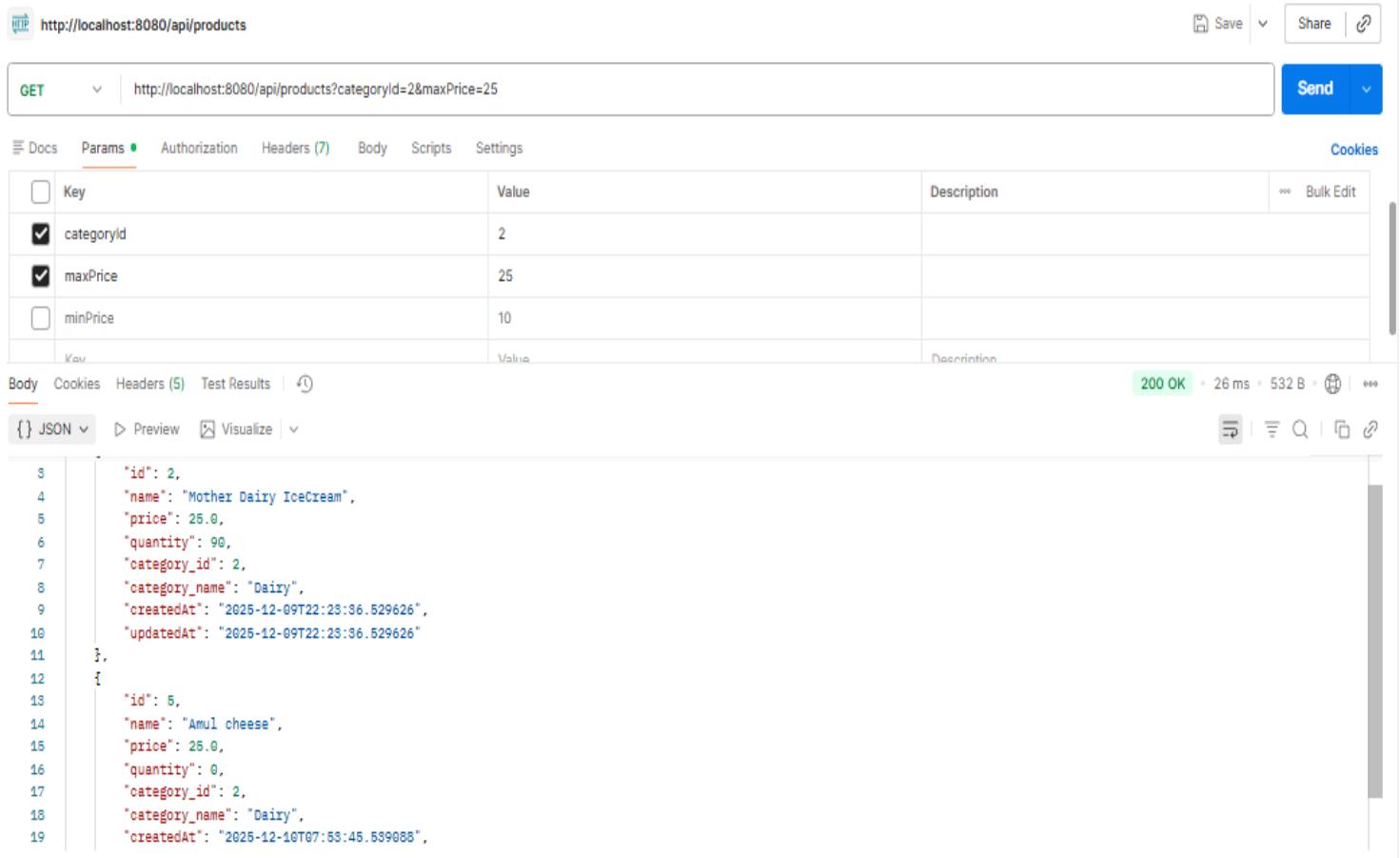
Body Cookies Headers (5) Test Results

200 OK 64 ms 890 B

{ } JSON Preview Visualize

```
1 L
2 [
3   {
4     "id": 1,
5     "name": "Amul Milk",
6     "price": 35.0,
7     "quantity": 900,
8     "category_id": 2,
9     "category_name": "Dairy",
10    "createdAt": "2025-12-09T22:23:08.288268",
11    "updatedAt": "2025-12-09T22:23:08.288268"
12  },
13  {
14    "id": 2,
15    "name": "Mother Dairy IceCream",
16    "price": 25.0,
17    "quantity": 90,
18    "category_id": 2,
19    "category_name": "Dairy",
20    "createdAt": "2025-12-09T22:23:36.529626",
21    "updatedAt": "2025-12-09T22:23:36.529626"
22  },
23  {
24    "id": 3,
25    "name": "Amul Butter",
26    "price": 55.0,
27    "quantity": 900,
28    "category_id": 2,
29    "category_name": "Dairy",
30    "createdAt": "2025-12-09T22:23:36.529626",
31    "updatedAt": "2025-12-09T22:23:36.529626"
32 }
```

### 2) Filter(categoryId:2 , maxPrice :25)



HTTP <http://localhost:8080/api/products> Save Share ⌂

GET http://localhost:8080/api/products?categoryId=2&maxPrice=25 Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (5) Test Results

200 OK 26 ms 532 B

{ } JSON Preview Visualize

<input type="checkbox"/> Key	Value	Description	<input type="checkbox"/> Bulk Edit
<input checked="" type="checkbox"/> categoryId	2		
<input checked="" type="checkbox"/> maxPrice	25		
<input type="checkbox"/> minPrice	10		

```
3 L
4 [
5   {
6     "id": 2,
7     "name": "Mother Dairy IceCream",
8     "price": 25.0,
9     "quantity": 90,
10    "category_id": 2,
11    "category_name": "Dairy",
12    "createdAt": "2025-12-09T22:23:36.529626",
13    "updatedAt": "2025-12-09T22:23:36.529626"
14  },
15  {
16    "id": 5,
17    "name": "Amul cheese",
18    "price": 25.0,
19    "quantity": 0,
20    "category_id": 2,
21    "category_name": "Dairy",
22    "createdAt": "2025-12-10T07:53:45.539088",
23    "updatedAt": "2025-12-10T07:53:45.539088"
24 }
```

### 3) With All Three filters

HTTP <http://localhost:8080/api/products> Save Share Send

GET <http://localhost:8080/api/products?categoryId=2&maxPrice=60&minPrice=36>

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

Key	Value	Description	Bulk Edit
categoryId	2		
maxPrice	60		
minPrice	36		

Body Cookies Headers (5) Test Results ①

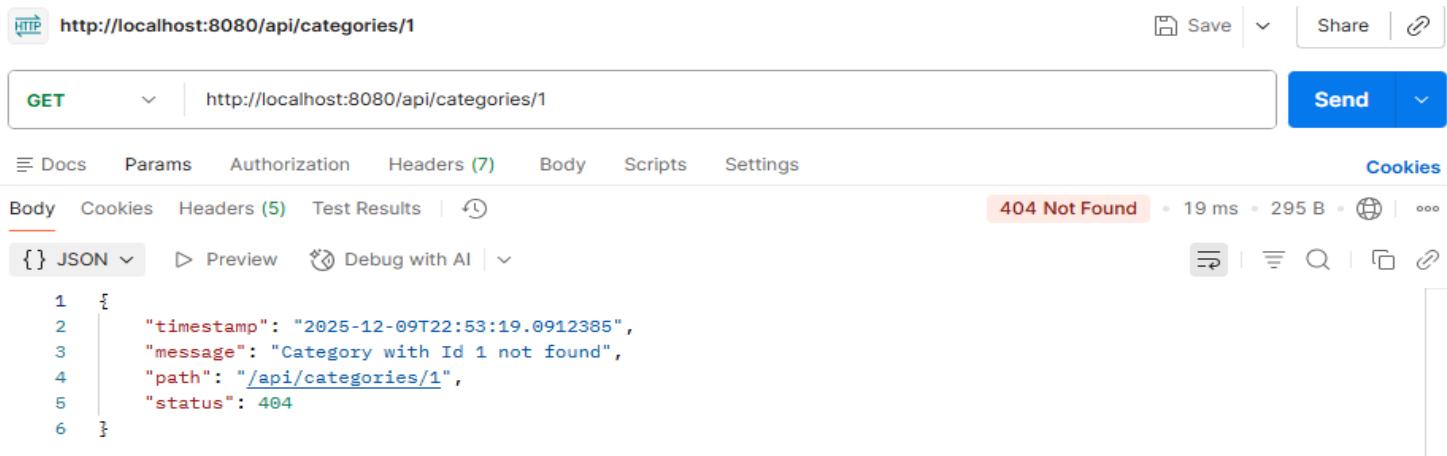
{ } JSON ▶ Preview Visualize ②

```
1 [  
2   {  
3     "id": 3,  
4     "name": "Amul Butter",  
5     "price": 58.0,  
6     "quantity": 900,  
7     "category_id": 2,  
8     "category_name": "Dairy",  
9     "createdAt": "2025-12-09T22:25:50.847677",  
10    "updatedAt": "2025-12-09T22:25:50.847677"  
11  }  
12 ]
```

200 OK 46 ms 345 B ③

### 3) Validation and Exception

#### a) Fetching category after deleting it



HTTP <http://localhost:8080/api/categories/1> Save Share

GET http://localhost:8080/api/categories/1 Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

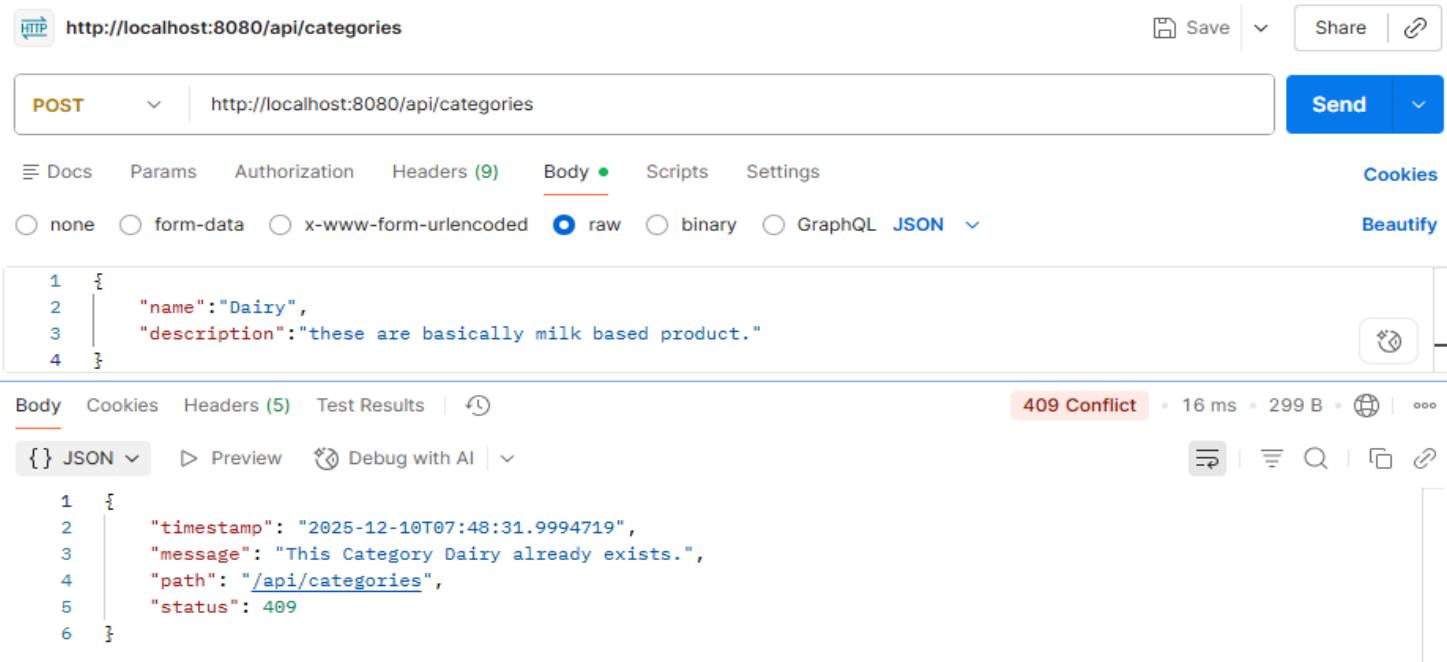
Body Cookies Headers (5) Test Results Debug with AI

404 Not Found 19 ms 295 B

{ } JSON Preview Debug with AI

```
1 {  
2   "timestamp": "2025-12-09T22:53:19.0912385",  
3   "message": "Category with Id 1 not found",  
4   "path": "/api/categories/1",  
5   "status": 404  
6 }
```

#### b) Creating duplicate category



HTTP <http://localhost:8080/api/categories> Save Share

POST http://localhost:8080/api/categories Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

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

Beautify

{ } JSON Preview Debug with AI

```
1 {  
2   "name": "Dairy",  
3   "description": "these are basically milk based product."  
4 }
```

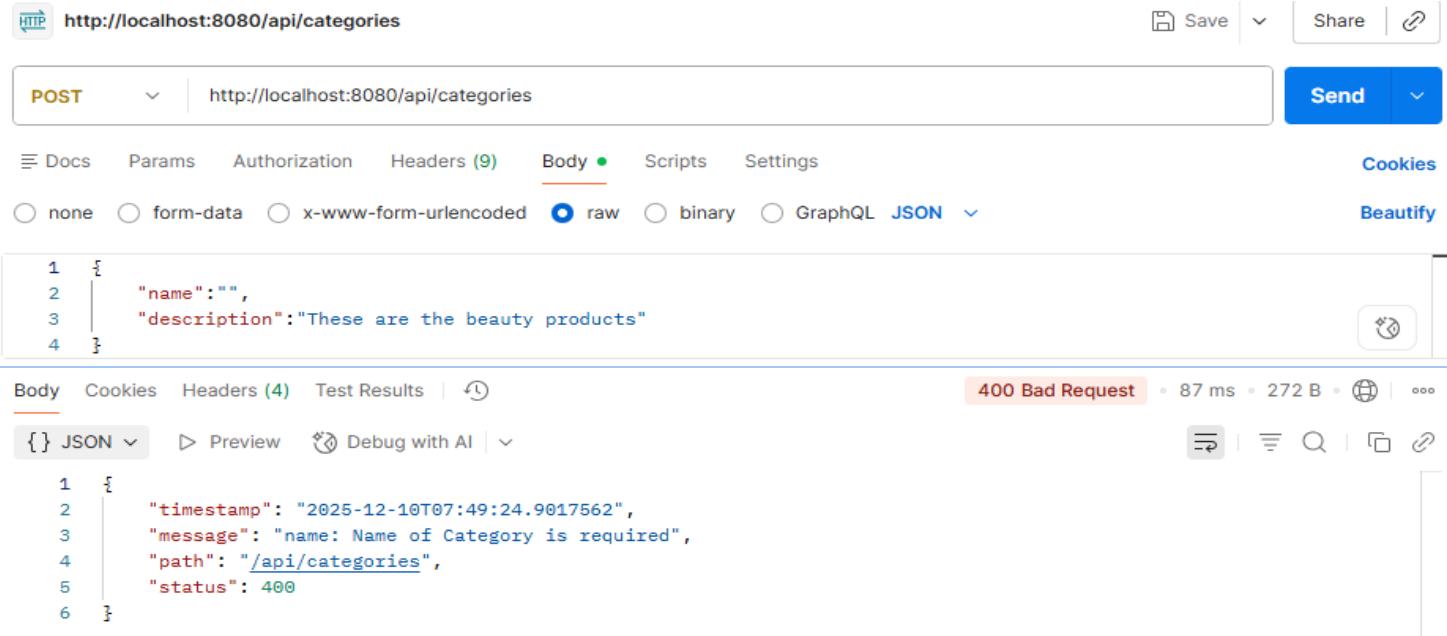
409 Conflict 16 ms 299 B

Body Cookies Headers (5) Test Results Debug with AI

{ } JSON Preview Debug with AI

```
1 {  
2   "timestamp": "2025-12-10T07:48:31.9994719",  
3   "message": "This Category Dairy already exists.",  
4   "path": "/api/categories",  
5   "status": 409  
6 }
```

#### c) Putting name of category as null



HTTP <http://localhost:8080/api/categories> Save Share

POST http://localhost:8080/api/categories Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

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

Beautify

{ } JSON Preview Debug with AI

```
1 {  
2   "name": "",  
3   "description": "These are the beauty products"  
4 }
```

400 Bad Request 87 ms 272 B

Body Cookies Headers (4) Test Results Debug with AI

{ } JSON Preview Debug with AI

```
1 {  
2   "timestamp": "2025-12-10T07:49:24.9017562",  
3   "message": "name: Name of Category is required",  
4   "path": "/api/categories",  
5   "status": 400  
6 }
```

d) Putting name of Product as null

HTTP <http://localhost:8080/api/products> Save Share

POST <http://localhost:8080/api/products> Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

Body Type: raw (selected), binary, GraphQL, JSON

Beautify

```
1 {  
2   "name": "",  
3   "price": 25,  
4   "quantity": 90,  
5   "categoryId": 2  
6 }
```

Body Cookies Headers (4) Test Results | [Debug with AI](#)

400 Bad Request 33 ms 266 B [View Response](#)

{ } JSON Preview Debug with AI

```
1 {  
2   "timestamp": "2025-12-10T07:51:01.1348201",  
3   "message": "name: Product name is required",  
4   "path": "/api/products",  
5   "status": 400  
6 }
```

e) Putting Price of product as null

HTTP <http://localhost:8080/api/products> Save Share

POST <http://localhost:8080/api/products> Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

Body Type: raw (selected), binary, GraphQL, JSON

Beautify

```
1 {  
2   "name": "Amul cheese",  
3   "price": "",  
4   "quantity": 90,  
5   "categoryId": 2  
6 }
```

Body Cookies Headers (4) Test Results | [Debug with AI](#)

400 Bad Request 15 ms 260 B [View Response](#)

{ } JSON Preview Debug with AI

```
1 {  
2   "timestamp": "2025-12-10T07:51:49.7893988",  
3   "message": "price: Price is required",  
4   "path": "/api/products",  
5   "status": 400  
6 }
```

f) Price < 0 (Negative price )

The screenshot shows the Postman interface. The URL is `http://localhost:8080/api/products`. The method is set to `POST`. The `Body` tab is selected, showing a raw JSON payload:

```
1 {  
2   "name": "Amul cheese",  
3   "price": -2,  
4   "quantity": 90,  
5   "categoryId": 2  
6 }
```

The response status is `400 Bad Request` with a timestamp of `2025-12-10T07:52:37.643643`, message `"price: Price must be > 0"`, path `"/api/products"`, and status `400`.

g) Trying to add duplicate product

The screenshot shows the Postman interface. The URL is `http://localhost:8080/api/products`. The method is set to `POST`. The `Body` tab is selected, showing a raw JSON payload:

```
1 {  
2   "name": "Amul cheese",  
3   "price": 25,  
4   "quantity": 0,  
5   "categoryId": 2  
6 }
```

The response status is `409 Conflict` with a timestamp of `2025-12-10T09:13:40.6175791`, message `"Duplicate product: Amul cheese already exists"`, path `"/api/products"`, and status `409`.