INDEX

INTRODUCTION

1.1 Project Summary

The Product Inventory Module is a web application that enables businesses to manage product information such as stock levels, prices, and descriptions. The module is built using NodeJS language and utilizes a MySQL database. The editor used for development is Visual Studio Code.

1.2 Overview

A product inventory model is an information system that stores and manages data about the products available in a store or warehouse. It consists of tables for products, categories, suppliers, and inventory transactions. The Product Inventory Module provides a platform to input, store and manage this information.

1.3 Purpose

The purpose of the Product Inventory Module is to enable businesses to store and manage information about each product such as name, description, price, category, and other attributes. This information can be used to create, update, and delete products.

SYSTEM ANALYSIS

2.1 Study of Current System and Requirements

To develop the Product Inventory Module, we used NodeJS, MySQL, and Postman. These technologies provided us with the necessary tools to create a reliable and easy-to-use system.

2.2 Operational Feasibility

The system requires a server, a browser, and an internet connection to function. This makes it easily accessible for users and ensures that they can operate the system from anywhere.

2.3 Economic Feasibility

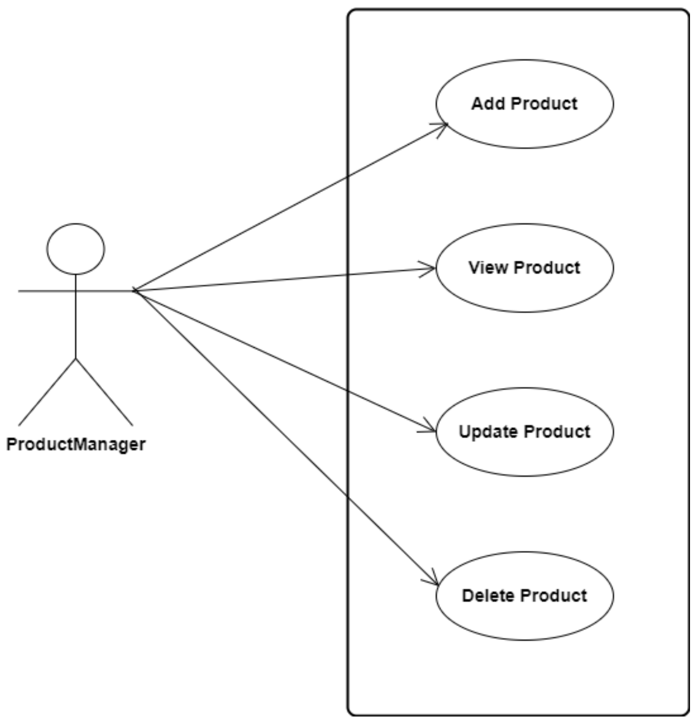
The system is easy to maintain and provides a reliable platform for storing and managing data. This makes it a cost-effective solution for businesses.

2.4 Function of the System

The Product Inventory Module allows businesses to manage product information, load data quickly, and easily manage records.

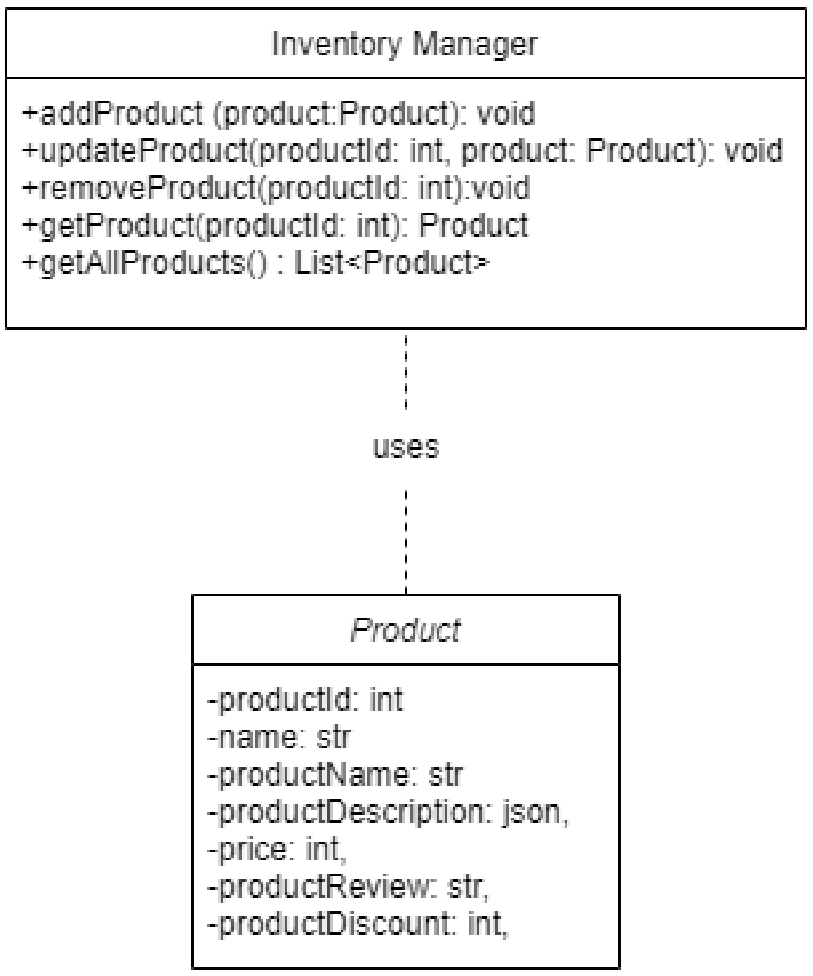
2.5 Use Case

The Product Manager can select desired products, add new products, view products, update product information, and delete products.



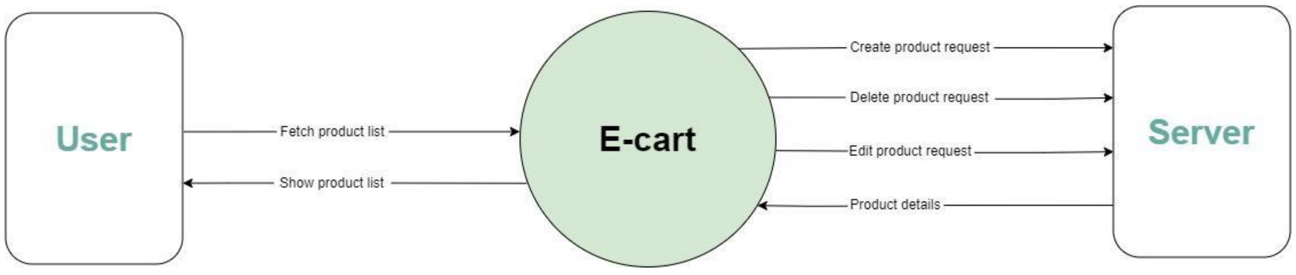
2.6 Class Diagram

The class diagram provides a visual representation of the system's data structure and the relationships between the various entities that make up the product management process.


Inventory Manager
+addProduct (product:Product): void
+updateProduct(productId: int, product: Product): void
+removeProduct(productId: int):void
+getProduct(productId: int): Product
+getAllProducts() : List<Product>
uses
Product
-productId: int
-name: str
-productName: str
-productDescription: json,
-price: int,
-productReview: str,
-productDiscount: int,

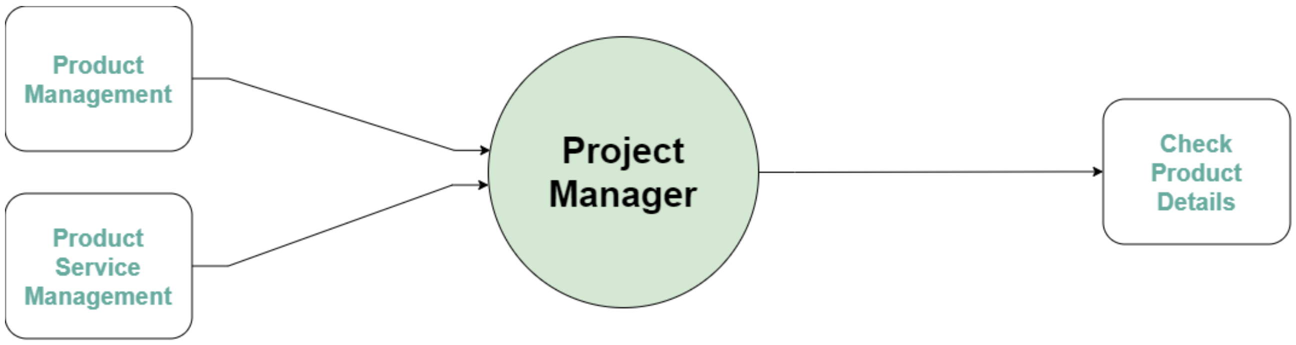
2.7 DFD Level 0

The Data Flow Diagram Level 0 provides a top-level view of the system, illustrating the interactions between the system and its external entities.



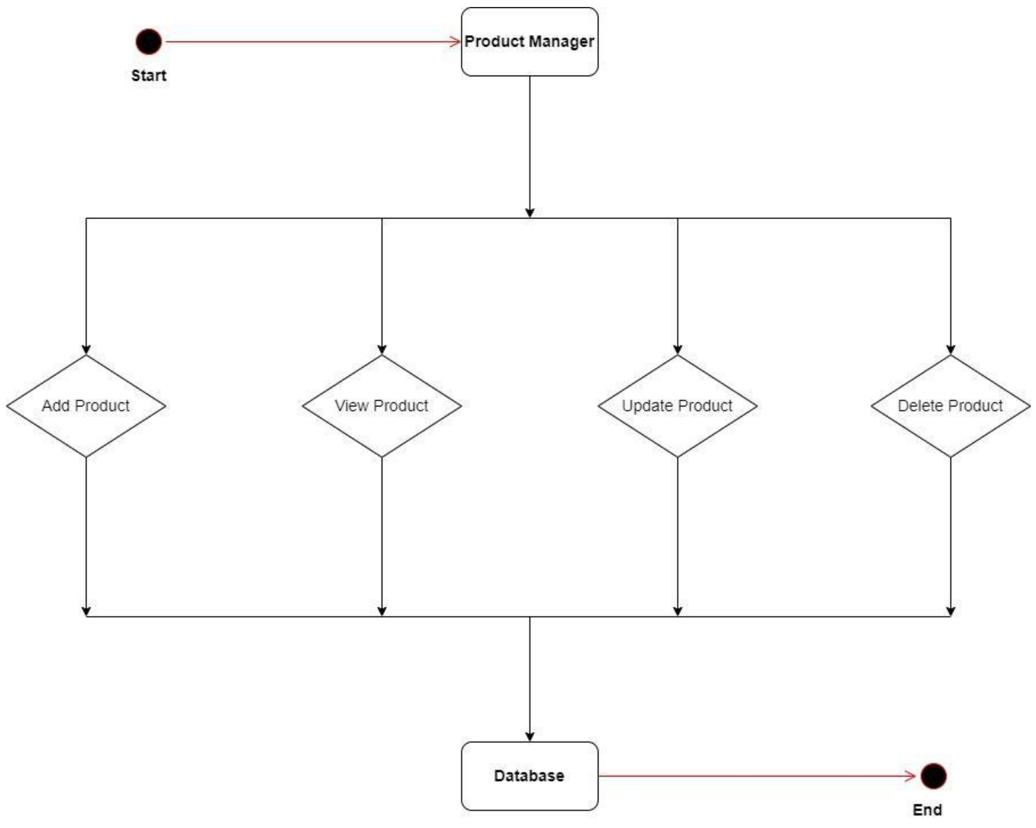
2.8 DFD Level 1

The Data Flow Diagram Level 1 provides



2.9 Activity Diagram:

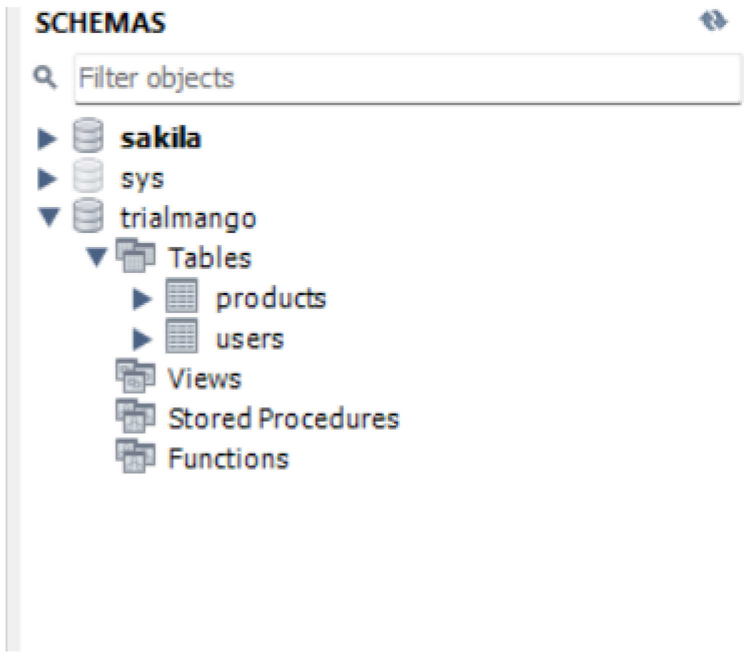
An activity diagram is a behavioural diagram in the Unified Modeling Language (UML) that shows the flow of activities or actions in a system or process. It is used to model the sequence of steps or actions required to complete a task or achieve a goal.



DATABASE

3.1 Data Dictionary:

3.1.1 Database Table:

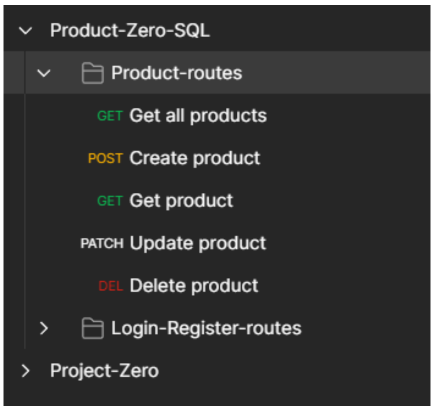


3.1.2 Product Table:

APIs

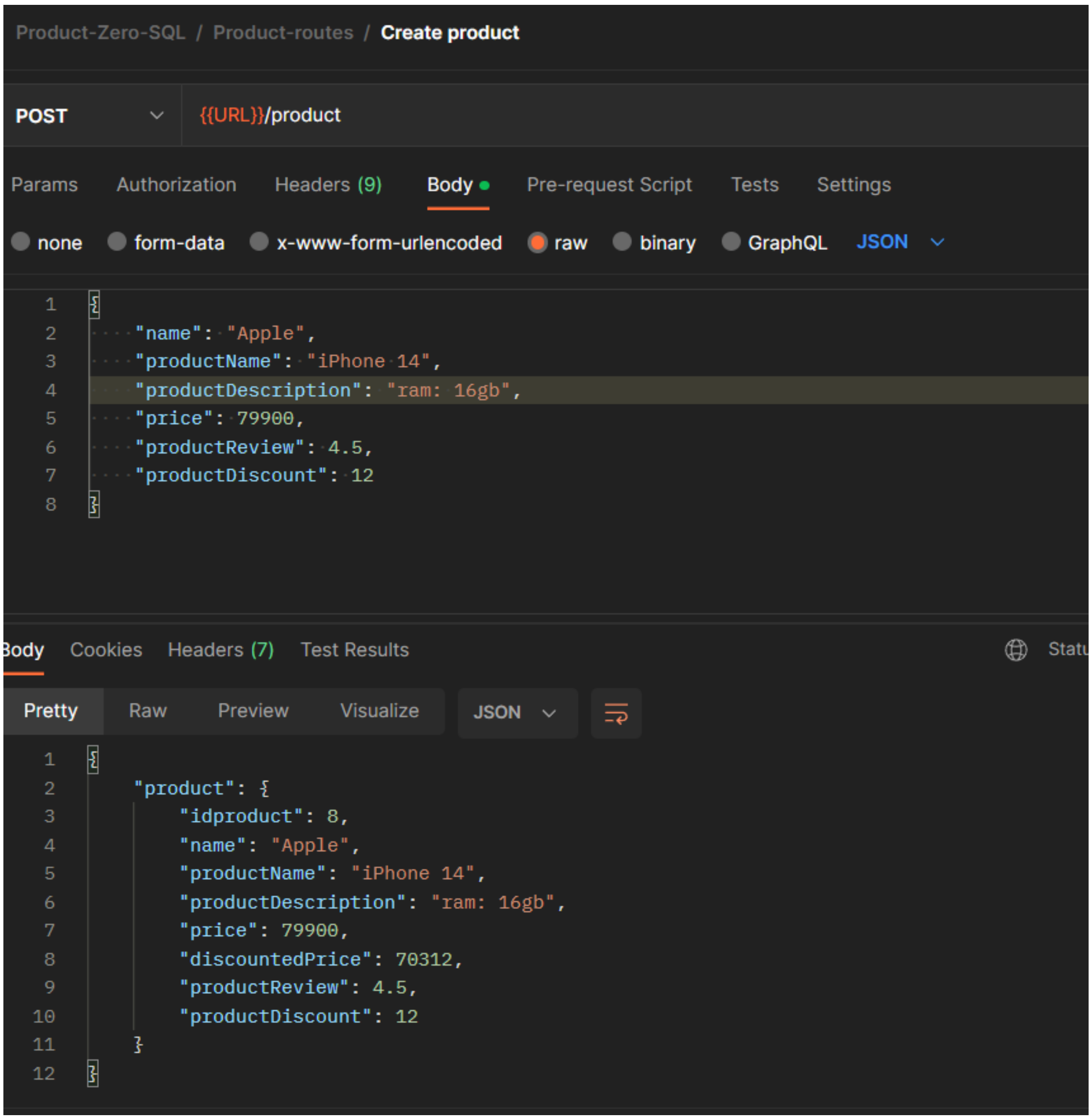
4.1 Product:

In this API, we can add, remove, update, or make changes to product details.



```
Product-Zero-SQL
  Product-routes
    GET Get all products
    POST Create product
    GET Get product
    PATCH Update product
    DEL Delete product
  Login-Register-routes
Project-Zero
```

4.1.1 Add New Product:



Product-Zero-SQL / Product-routes / Create product

POST {{URL}}/product

```
{
    "name": "Apple",
    "productName": "iPhone 14",
    "productDescription": "ram: 16gb",
    "price": 79900,
    "productReview": 4.5,
    "productDiscount": 12
}
```

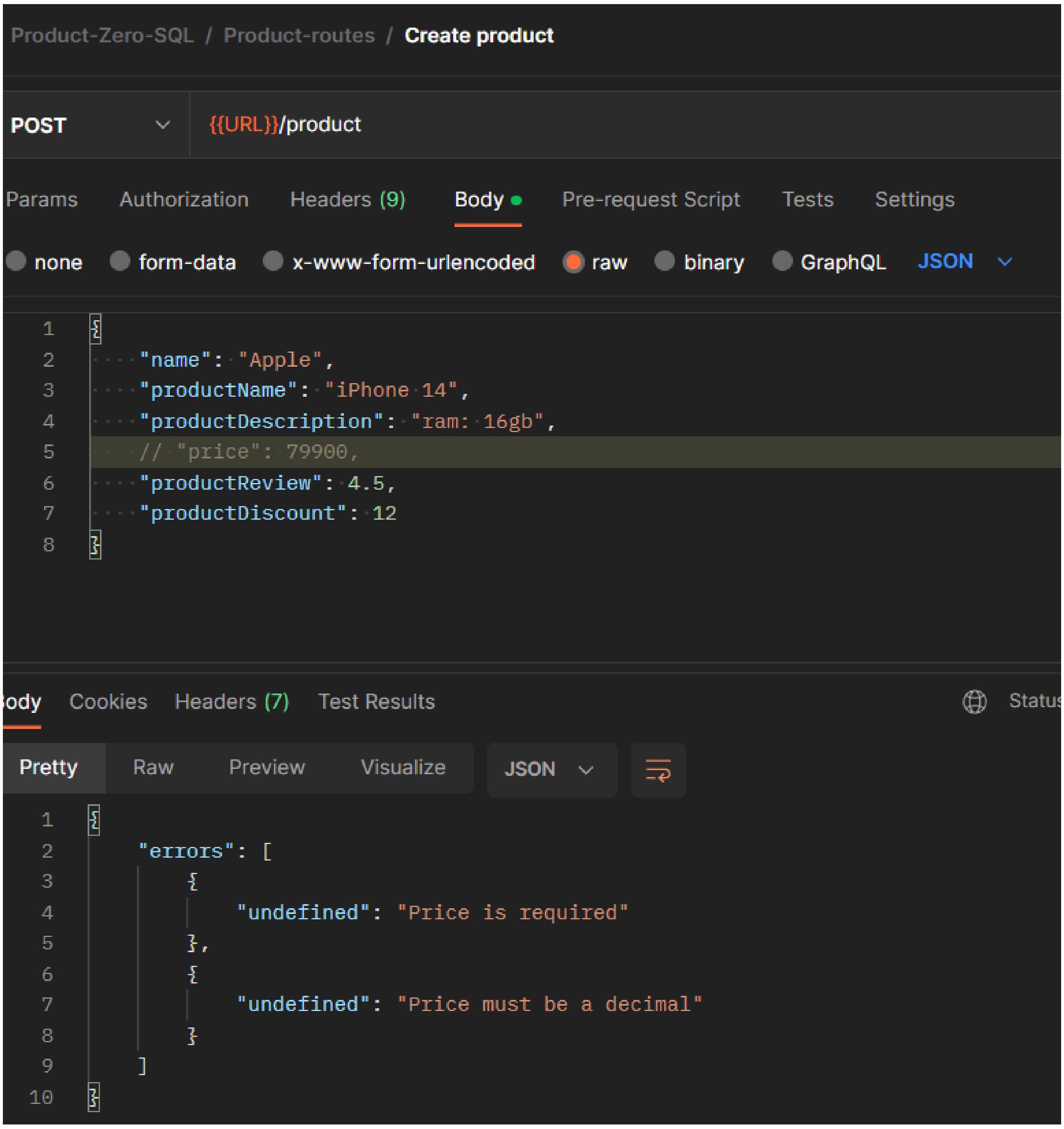
Response Body (Pretty, JSON):

```
{
    "product": {
        "idproduct": 8,
        "name": "Apple",
        "productName": "iPhone 14",
        "productDescription": "ram: 16gb",
        "price": 79900,
        "discountedPrice": 70312,
        "productReview": 4.5,
        "productDiscount": 12
    }
}
```

4.1.1.1 Add Complete Data:

If the product manager does not add proper information, an error message will be displayed.

Here, if a product's price is not given, we show this message.



Product-Zero-SQL / Product-routes / Create product
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Body Cookies Headers (7) Test Results Status
Pretty Raw Preview Visualize JSON

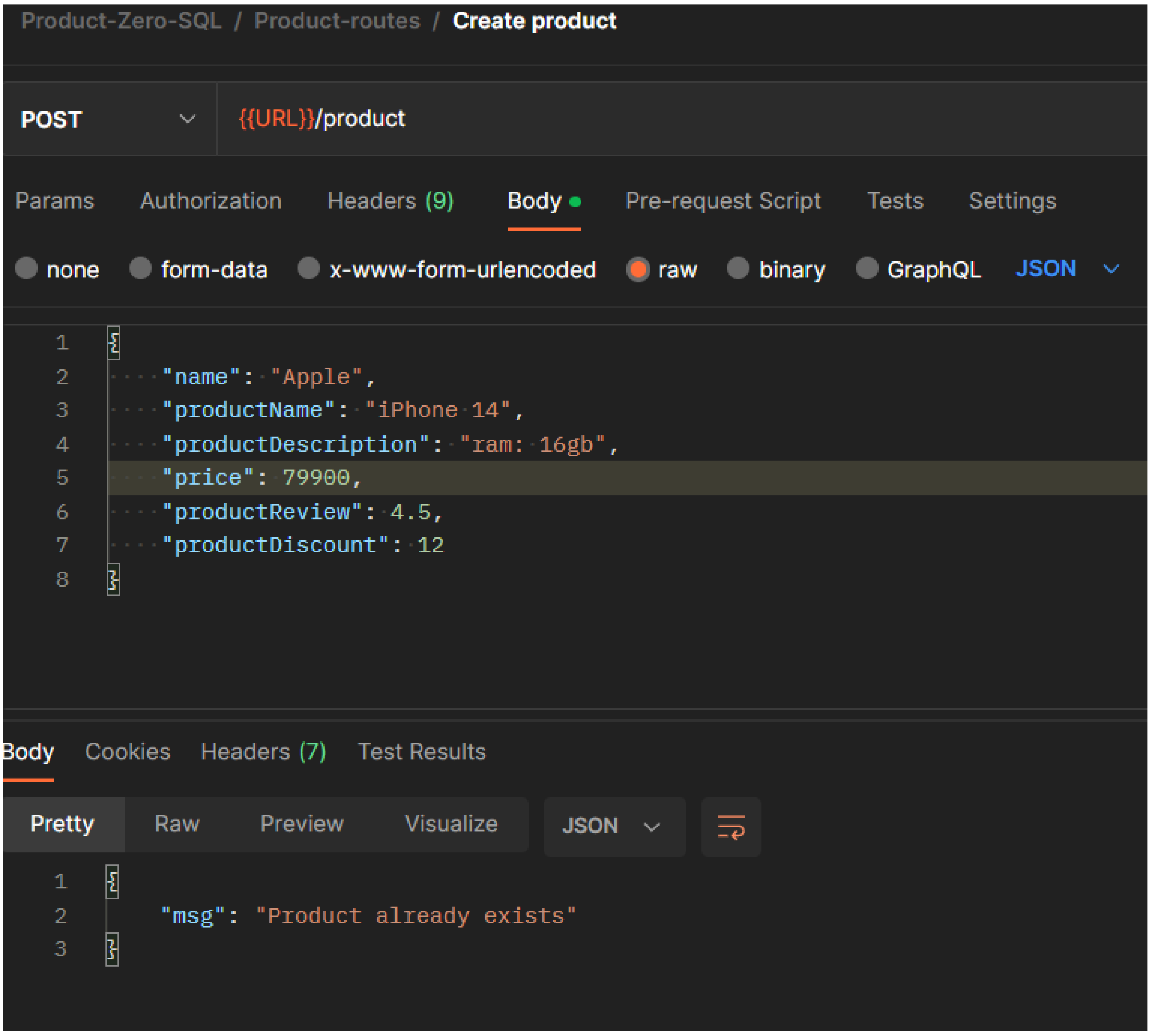
POST {{URL}}/product

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "name": "Apple",
    "productName": "iPhone 14",
    "productDescription": "ram: 16gb",
    // "price": 79900,
    "productReview": 4.5,
    "productDiscount": 12
}
```

```
{
    "errors": [
        {
            "undefined": "Price is required"
        },
        {
            "undefined": "Price must be a decimal"
        }
    ]
}
```

4.1.1.2 Same Details Used More Than Once:

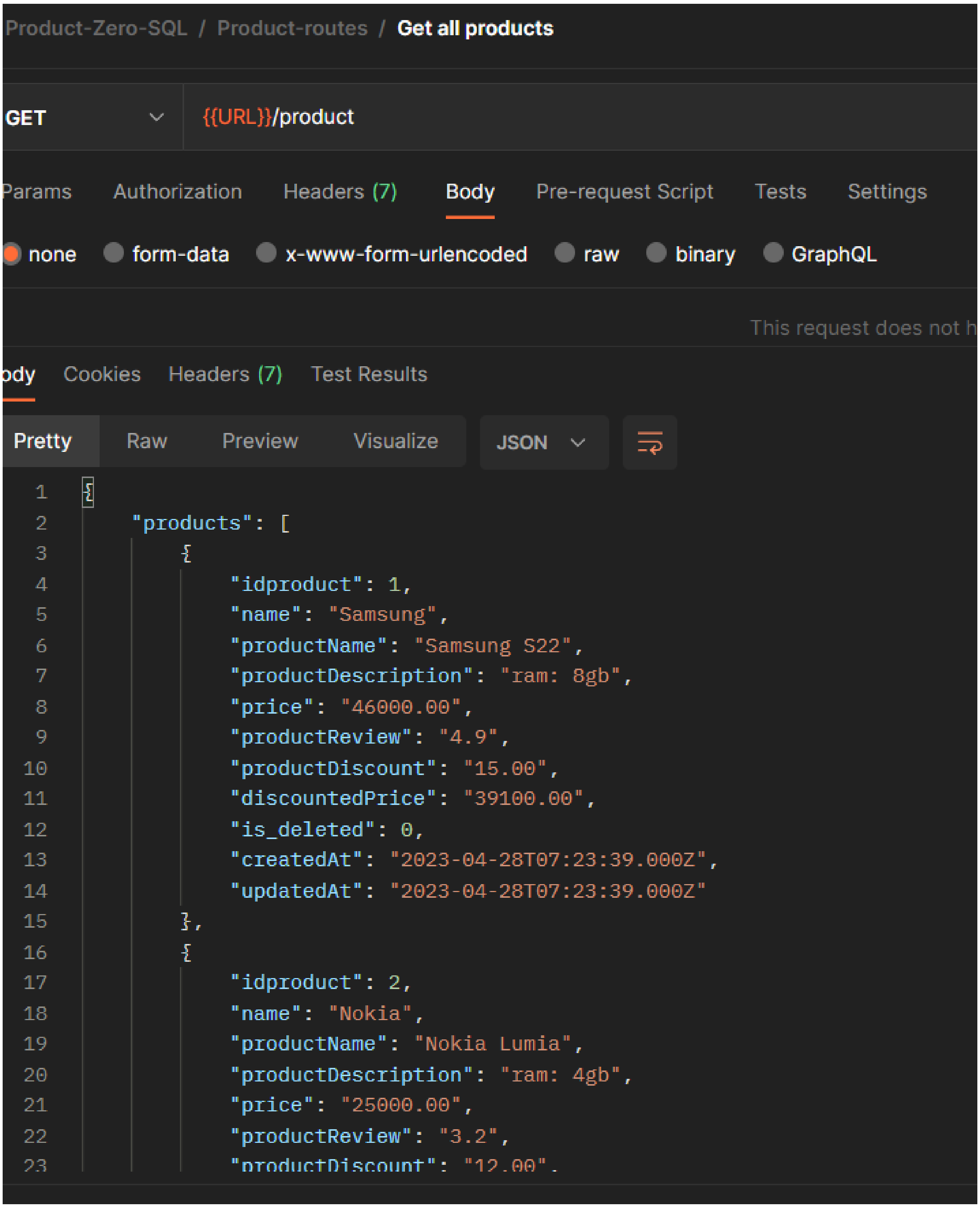

Product-Zero-SQL / Product-routes / Create product
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Body Cookies Headers (7) Test Results
Pretty Raw Preview Visualize JSON

POST {{URL}}/product

```
{
    "name": "Apple",
    "productName": "iPhone 14",
    "productDescription": "ram: 16gb",
    "price": 79900,
    "productReview": 4.5,
    "productDiscount": 12
}
```

```
{
    "msg": "Product already exists"
}
```

4.1.2 Get All Listed Products List:



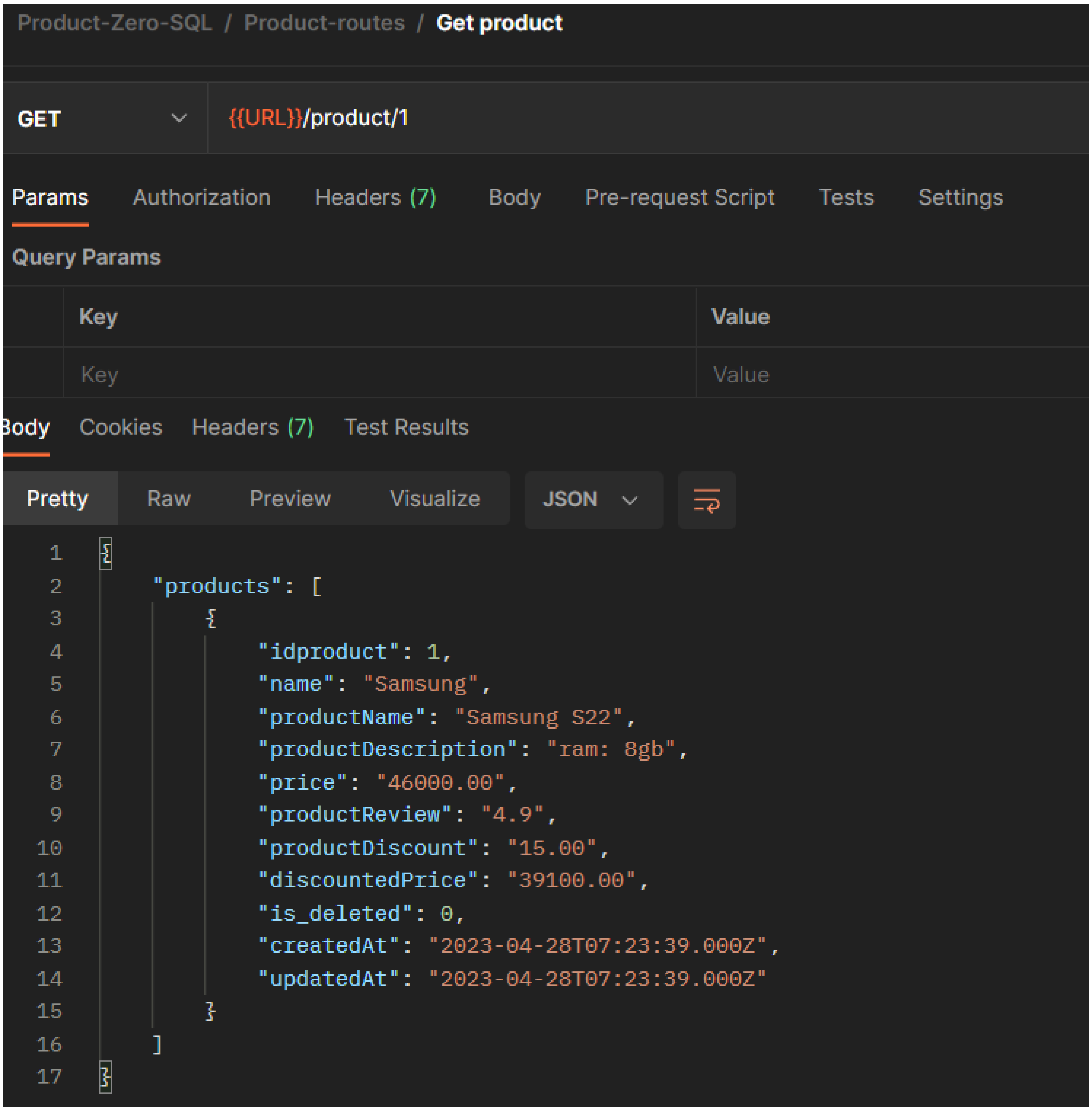
Product-Zero-SQL / Product-routes / **Get all products**

GET {{URL}}/product

This request does not h

```
{
    "products": [
        {
            "idproduct": 1,
            "name": "Samsung",
            "productName": "Samsung S22",
            "productDescription": "ram: 8gb",
            "price": "46000.00",
            "productReview": "4.9",
            "productDiscount": "15.00",
            "discountedPrice": "39100.00",
            "is_deleted": 0,
            "createdAt": "2023-04-28T07:23:39.000Z",
            "updatedAt": "2023-04-28T07:23:39.000Z"
        },
        {
            "idproduct": 2,
            "name": "Nokia",
            "productName": "Nokia Lumia",
            "productDescription": "ram: 4gb",
            "price": "25000.00",
            "productReview": "3.2",
            "productDiscount": "12.00",
```

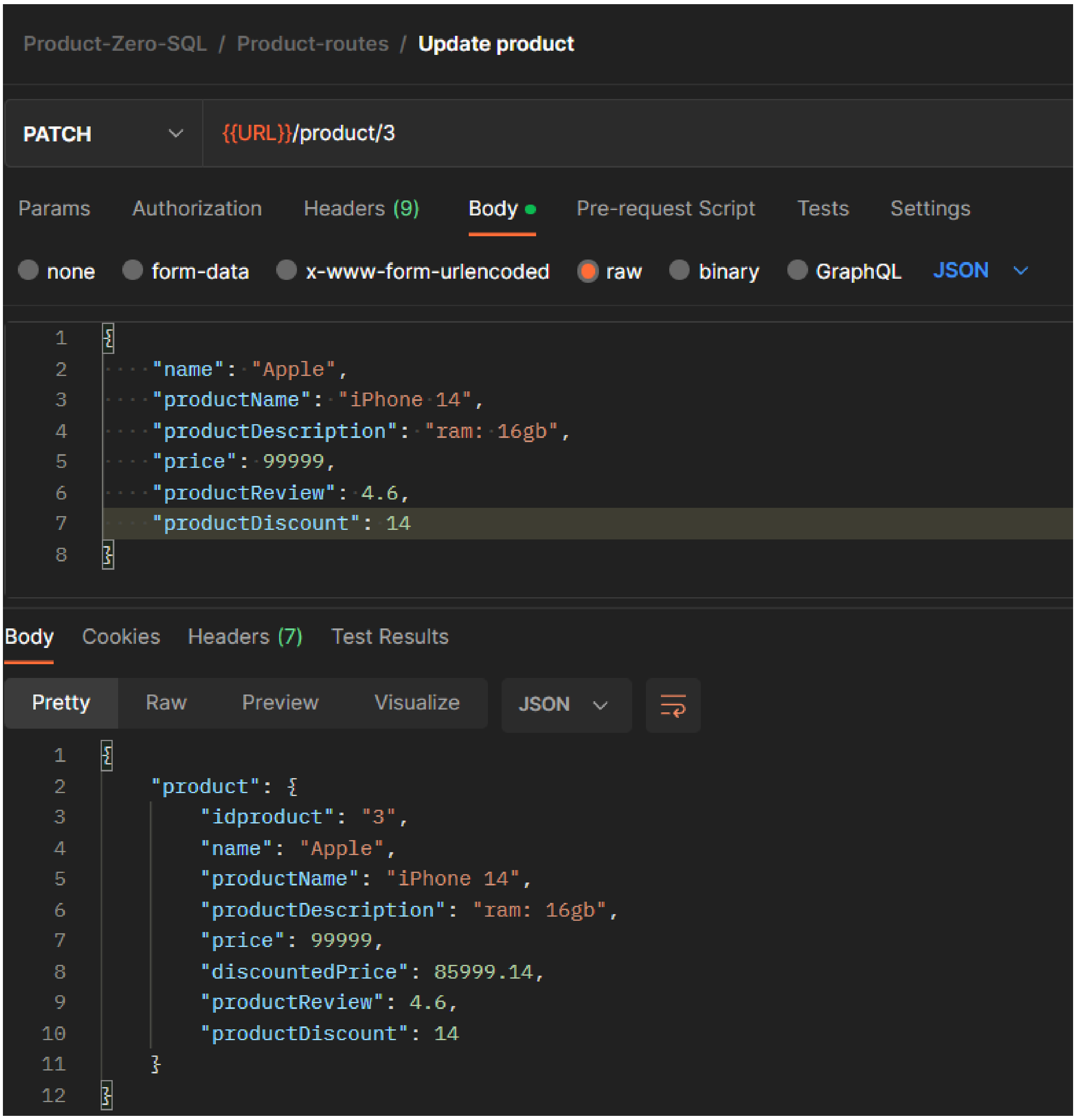
4.1.3 Get Single Product by Their Product Id:



```
{{URL}}/product/1
```

```
{
    "products": [
        {
            "idproduct": 1,
            "name": "Samsung",
            "productName": "Samsung S22",
            "productDescription": "ram: 8gb",
            "price": "46000.00",
            "productReview": "4.9",
            "productDiscount": "15.00",
            "discountedPrice": "39100.00",
            "is_deleted": 0,
            "createdAt": "2023-04-28T07:23:39.000Z",
            "updatedAt": "2023-04-28T07:23:39.000Z"
        }
    ]
}
```

4.1.4 Update Product:



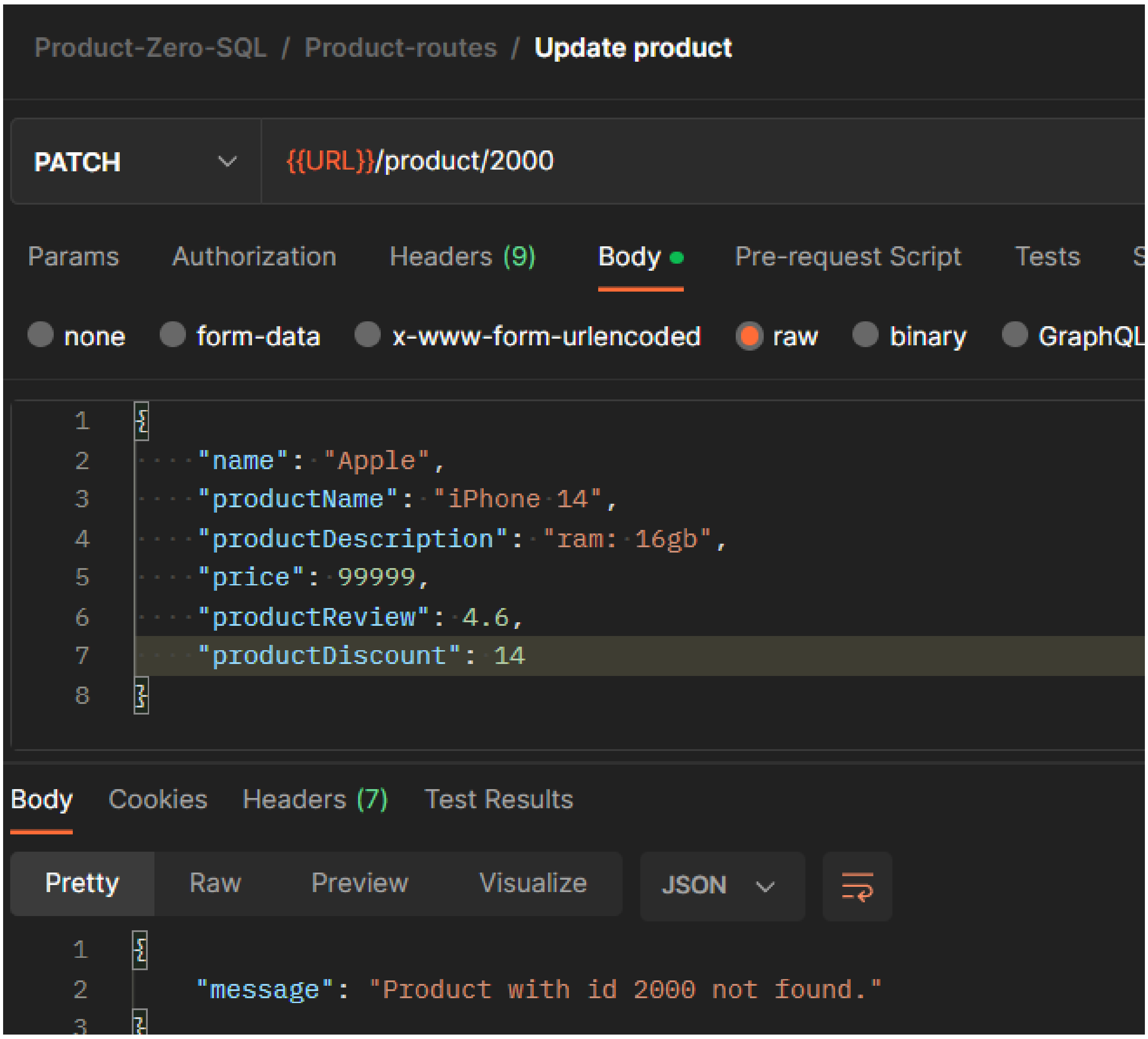
PATCH {{URL}}/product/3

```
{
    "name": "Apple",
    "productName": "iPhone 14",
    "productDescription": "ram: 16gb",
    "price": 99999,
    "productReview": 4.6,
    "productDiscount": 14
}
```

Response (JSON):

```
{
    "product": {
        "idproduct": "3",
        "name": "Apple",
        "productName": "iPhone 14",
        "productDescription": "ram: 16gb",
        "price": 99999,
        "discountedPrice": 85999.14,
        "productReview": 4.6,
        "productDiscount": 14
    }
}
```

4.1.4.1 Update Specific Records



Product-Zero-SQL / Product-routes / Update product

PATCH {{URL}}/product/2000

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL

```
{
    "name": "Apple",
    "productName": "iPhone 14",
    "productDescription": "ram: 16gb",
    "price": 99999,
    "productReview": 4.6,
    "productDiscount": 14
}
```

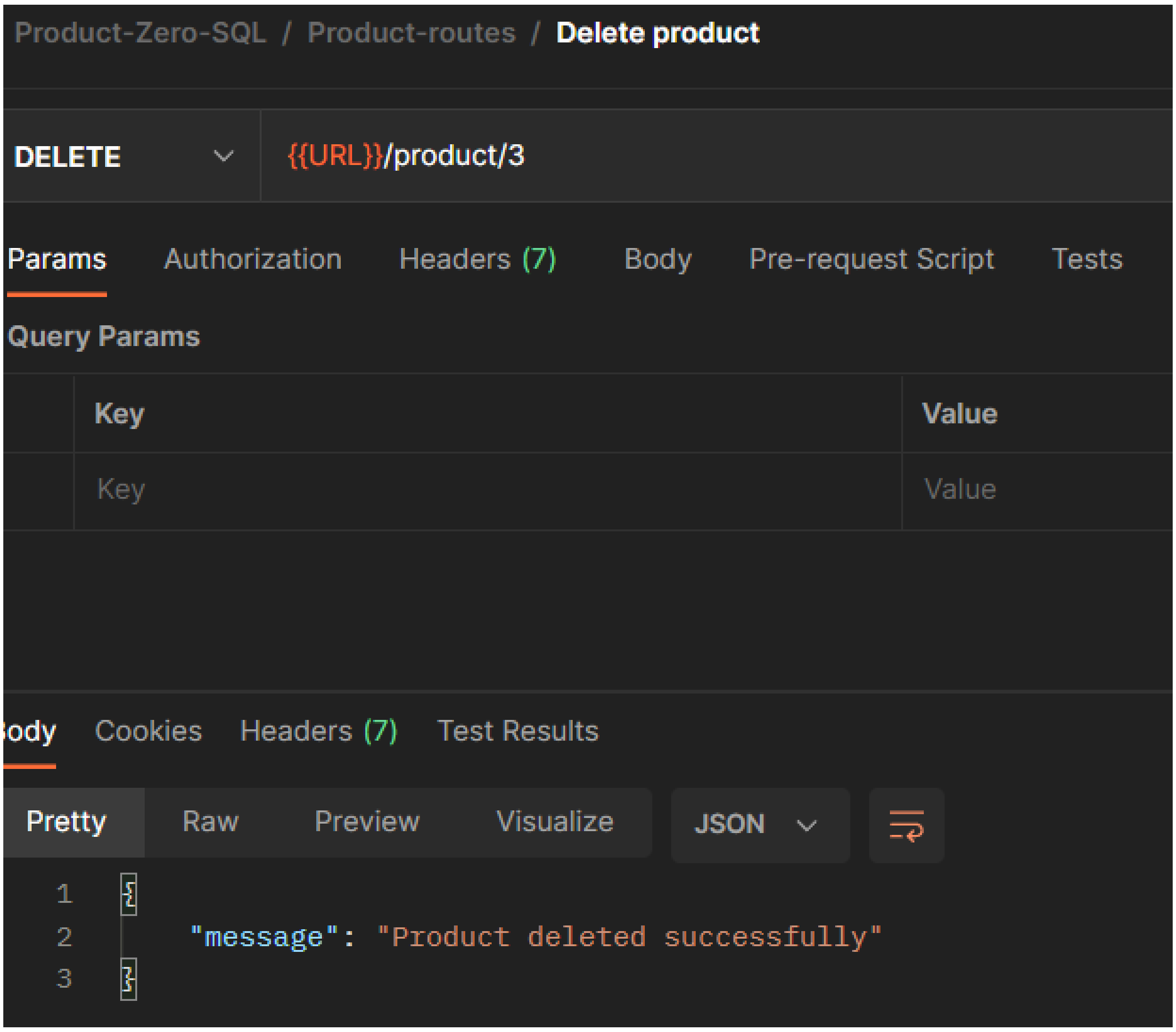
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "Product with id 2000 not found."
}
```

4.1.5 Soft-Delete Product

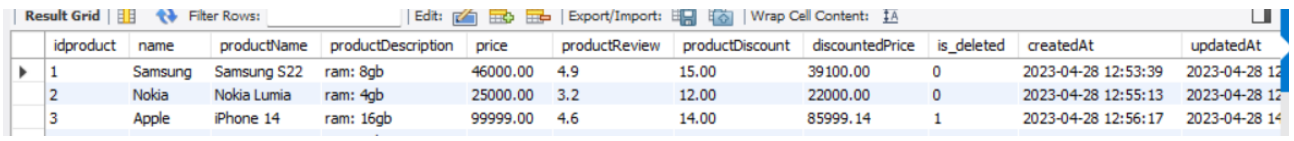
Product is deleted successfully for the user to see but still exists in the database.



```
DELETE {{URL}}/product/3
```

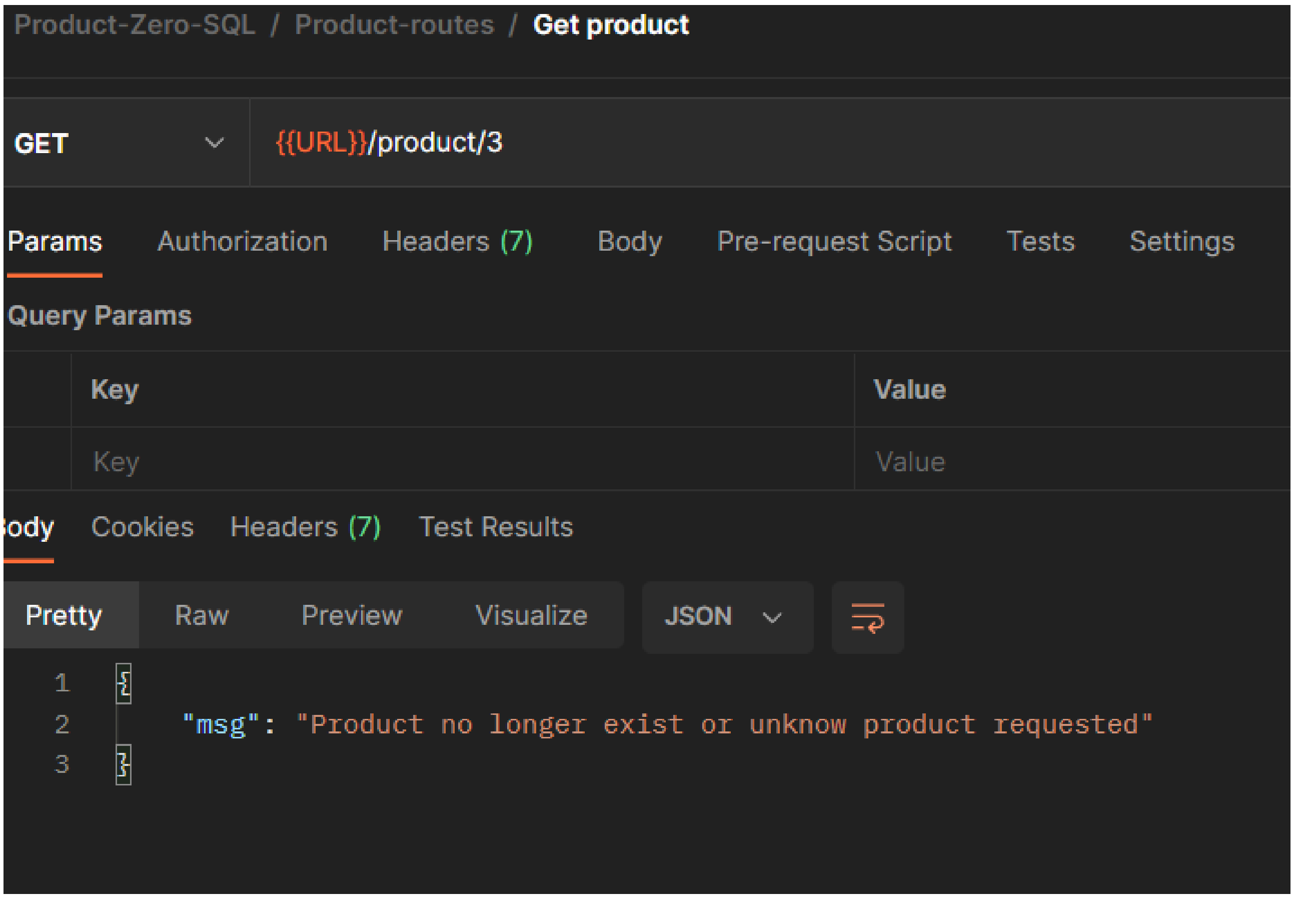
```
{
    "message": "Product deleted successfully"
}
```

4.1.5.2 Delete Product but Keep in Database



idproduct	name	productName	productDescription	price	productReview	productDiscount	discountedPrice	is_deleted	createdAt	updatedAt
1	Samsung	Samsung S22	ram: 8gb	46000.00	4.9	15.00	39100.00	0	2023-04-28 12:53:39	2023-04-28 12
2	Nokia	Nokia Lumia	ram: 4gb	25000.00	3.2	12.00	22000.00	0	2023-04-28 12:55:13	2023-04-28 12
3	Apple	iPhone 14	ram: 16gb	99999.00	4.6	14.00	85999.14	1	2023-04-28 12:56:17	2023-04-28 14

4.1.5.3 Delete Product and Remove from GET API:



```
GET {{URL}}/product/3
```

```
{
    "msg": "Product no longer exist or unknow product requested"
}
```