Codewits Solutions | VIT | Assignment + Evaluation

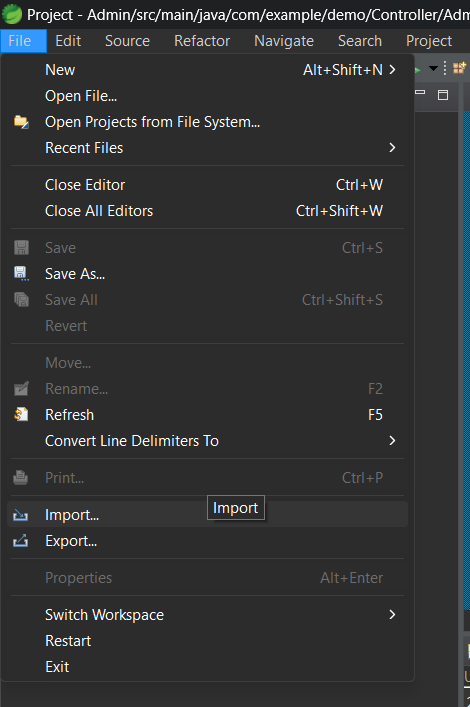
Develop a backend for an e-commerce website that allows users to browse, purchase products, and manage their accounts.

I developed the entire application using **Spring Boot** as the primary framework for building the backend, **MongoDB** as the database for managing data efficiently, and utilized **Postman** to perform and validate API requests throughout the development process.

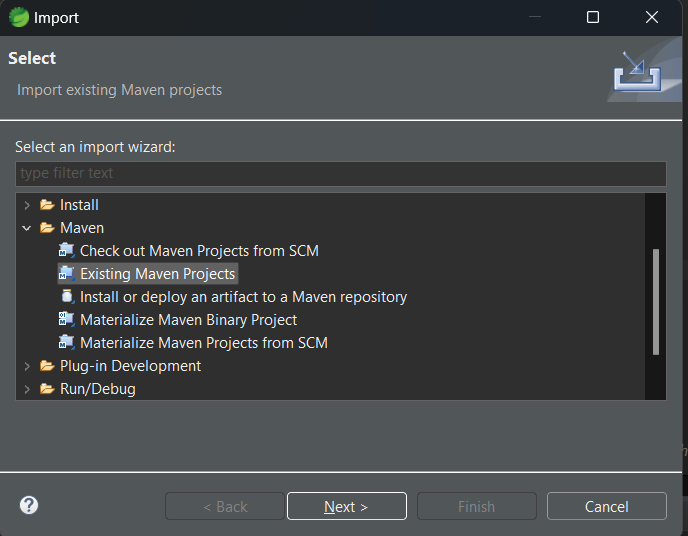
To Run the project first Download the zip of the project from github, then extract the entire project.

I have use **Spring Tool Suite 4** integrated development environment (IDE) for the development of this entire project from the scratch.

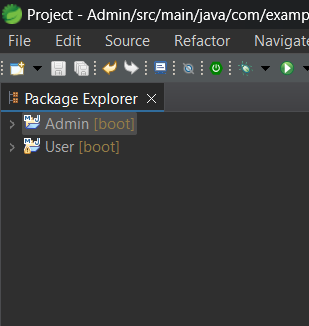
1. Import the entire project.



1. Click on Existing Maven Project:



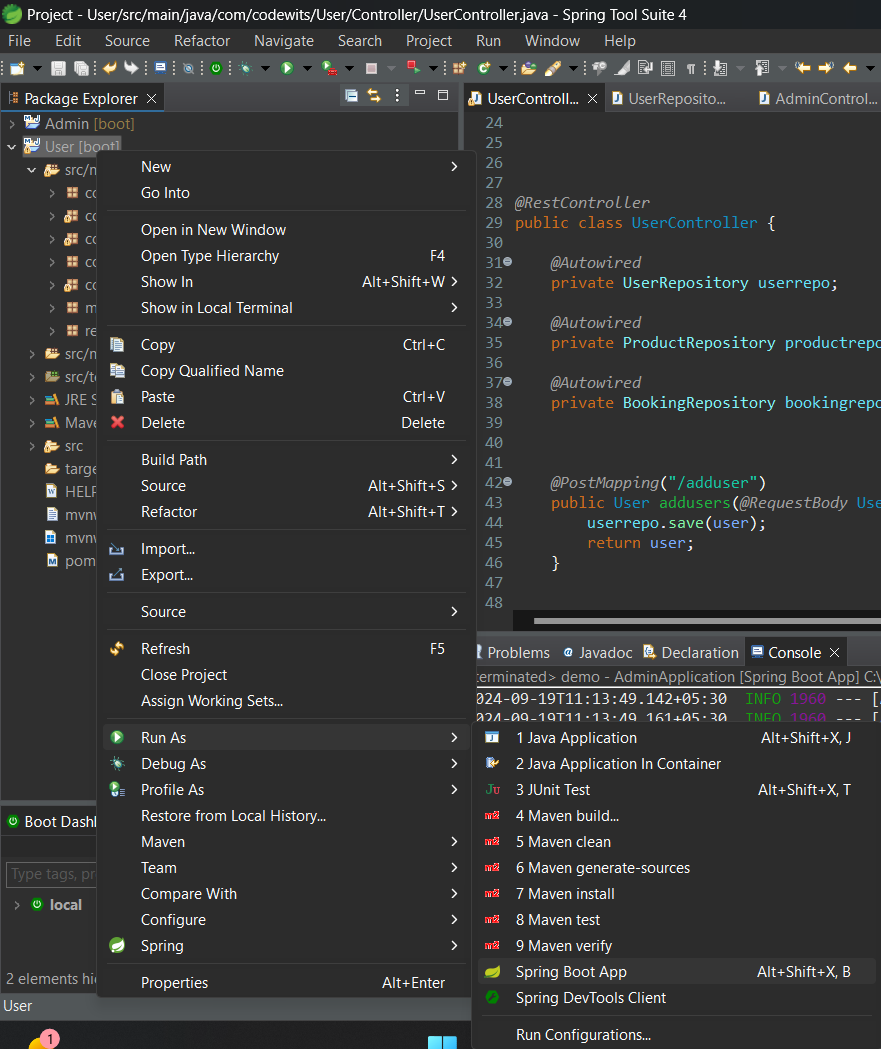
1. Browse the project in your system, and click on finish. After that u will have two Java project **Admin** and **User.**



1. Have all the dependences in pom.xml
2. <?xml version="1.0" encoding="UTF-8"?>
3. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
5. <modelVersion>4.0.0</modelVersion>
6. <parent>
7. <groupId>org.springframework.boot</groupId>
8. <artifactId>spring-boot-starter-parent</artifactId>
9. <version>3.3.3</version>
10. <relativePath/> <!-- lookup parent from repository -->
11. </parent>
12. <groupId>com.example</groupId>
13. <artifactId>demo</artifactId>
14. <version>0.0.1-SNAPSHOT</version>
15. <name>Admin</name>
16. <description>Demo project for Spring Boot</description>
17. <url/>
18. <licenses>
19. <license/>
20. </licenses>
21. <developers>
22. <developer/>
23. </developers>
24. <scm>
25. <connection/>
26. <developerConnection/>
27. <tag/>
28. <url/>
29. </scm>
30. <properties>
31. <java.version>17</java.version>
32. </properties>
33. <dependencies>
34. <dependency>
35. <groupId>org.springframework.boot</groupId>
36. <artifactId>spring-boot-starter-data-mongodb</artifactId>
37. </dependency>
38. <dependency>
39. <groupId>org.springframework.boot</groupId>
40. <artifactId>spring-boot-starter-web</artifactId>
41. </dependency>
42. <dependency>
43. <groupId>org.springframework.boot</groupId>
44. <artifactId>spring-boot-starter-test</artifactId>
45. <scope>test</scope>
46. </dependency>
47. </dependencies>
48. <build>
49. <plugins>
50. <plugin>
51. <groupId>org.springframework.boot</groupId>
52. <artifactId>spring-boot-maven-plugin</artifactId>
53. </plugin>
54. </plugins>
55. </build>
56. </project>

Above are the dependence copy past this if u have some problem in your pom.xml.

5. To Run the project Right click on user, then click on Run as and then click on Spring boot App.



6. After this your project will run on **localhost:8080:**

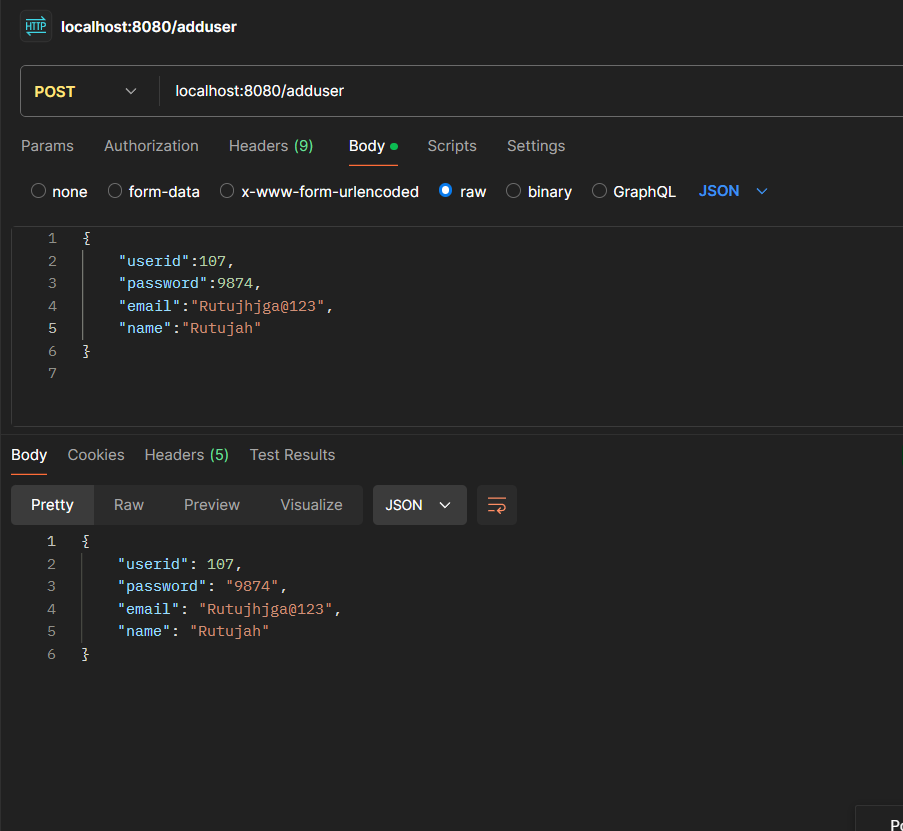
The application is running on **tomcat server 8080** and by default **mongo Db is running on address=localhost:27017**. To check each and every API request I have use **Postman Application** to run every APIs.

I have utilized **MongoDB** as the database to handle all **CRUD** operations within the application, ensuring that each create, read, update, and delete action is accurately reflected in the database, allowing for efficient data management and real-time visibility.

**Users APIs :**

**com.codewits.User.Controller.UserController -> In UserController.java** we have each and every RestController.

1. When we want to add the Users.



Our user project is running on **localhost:8080:**

And we have use the **POST Method** to add the user into our **Mongo Database.**

You can also add this in your system, just have Mongo db in your system, so that when you add the user it should add the user in database **“ Users ”,** which I have created.

Payload :

{

    "userid":107,

    "password":9874,

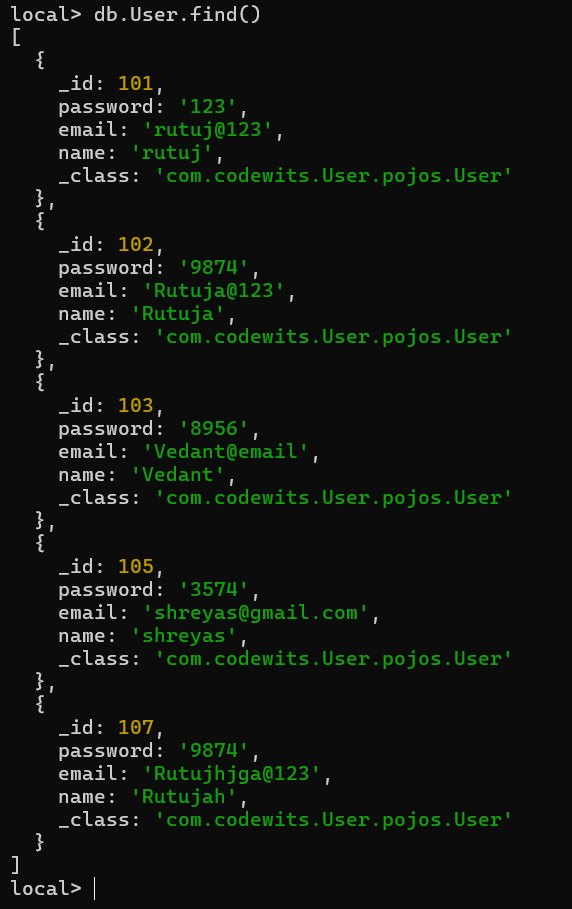
    "email":"Rutujhjga@123",

    "name":"Rutujah"

}

Add this payload in the postman application -> Body -> raw -> JSON.

To see the updates in **mongo db:**

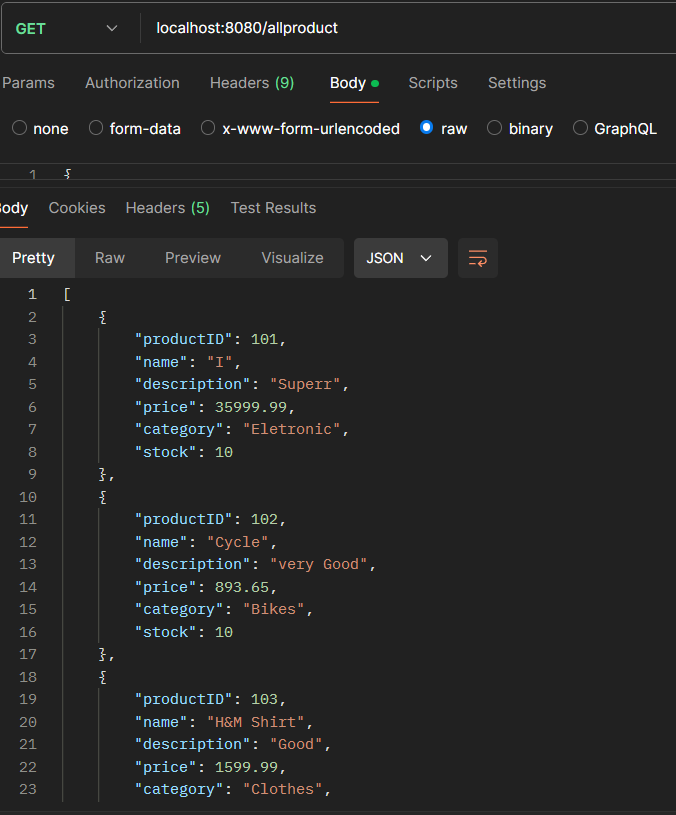


You can see the new use has been add in the Database.

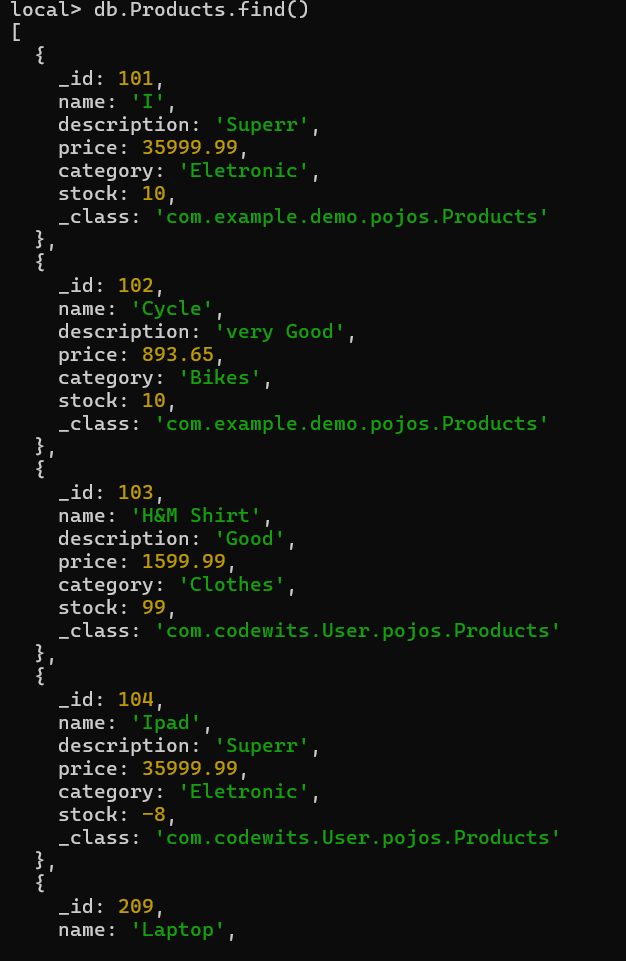
1. To see the products.

To see the Product use API – **localhost:8080/allproduct**

Its a **GET mapping** so we will use **GET** so that it will fetch the entire data from monog db.



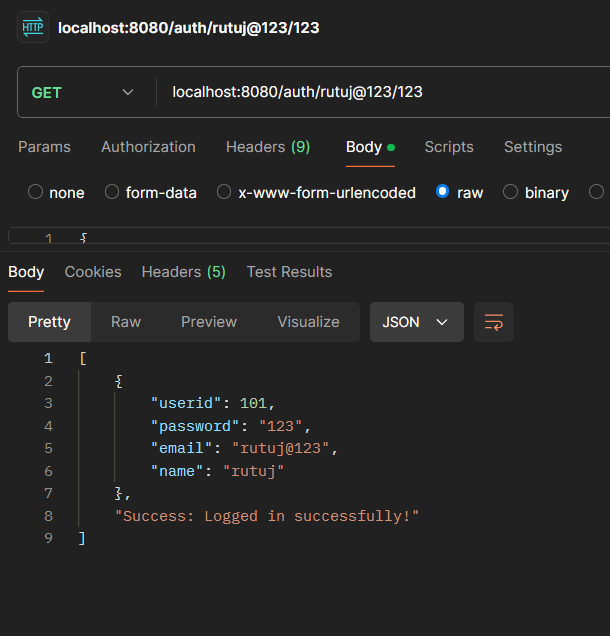
The product we have retrieve by GET method :



1. To do the Authentication of the user.

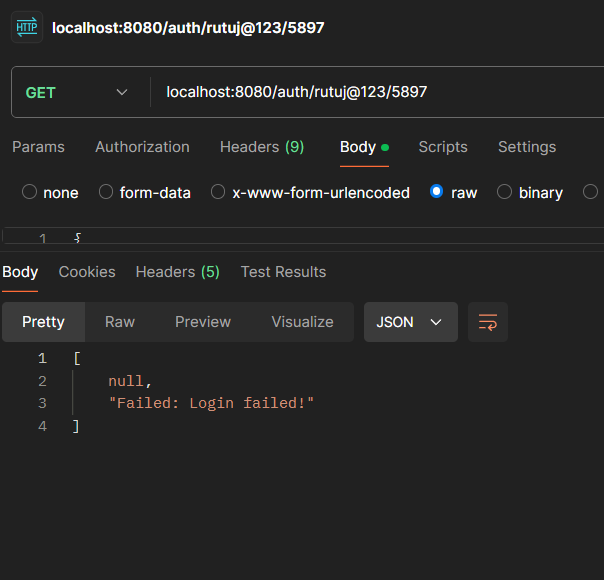
To do the Authentication of the user use API – **localhost:8080/auth/rutuj@123/123(Example of one user from DB).**

Its a **GET mapping** so we will use **GET** so that it will fetch the entire data from monog db and will see **if the user is store in DB or not.**



The particular user is present in DB so we have got successfully logged in.

**Expectation : If the person is not present in the DB**

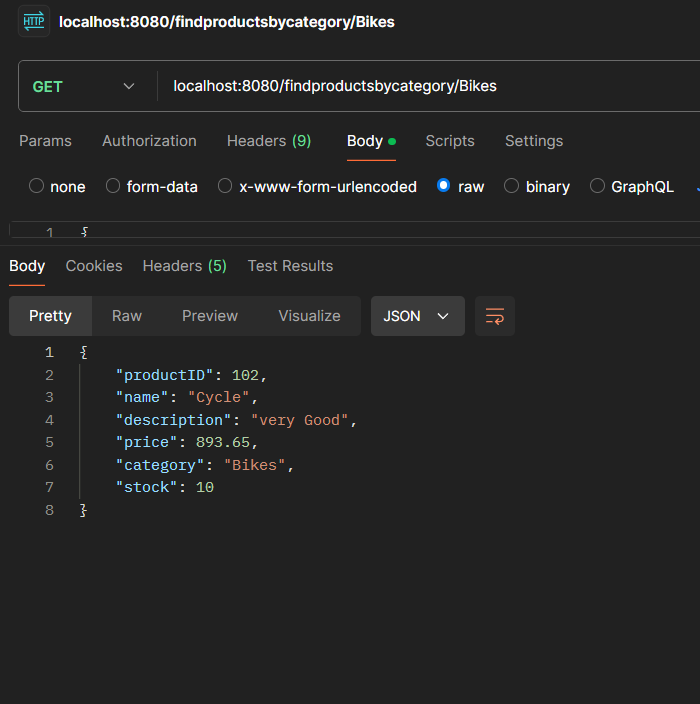
****

You can see I have change the password for the user.So it is giving me Failed: to login because the user does not exist.

1. To find the product by category.

To do the searching of the product by the user we will use the API – **localhost:8080/findproductsbycategory/Bikes (Example of one product from our DB).**

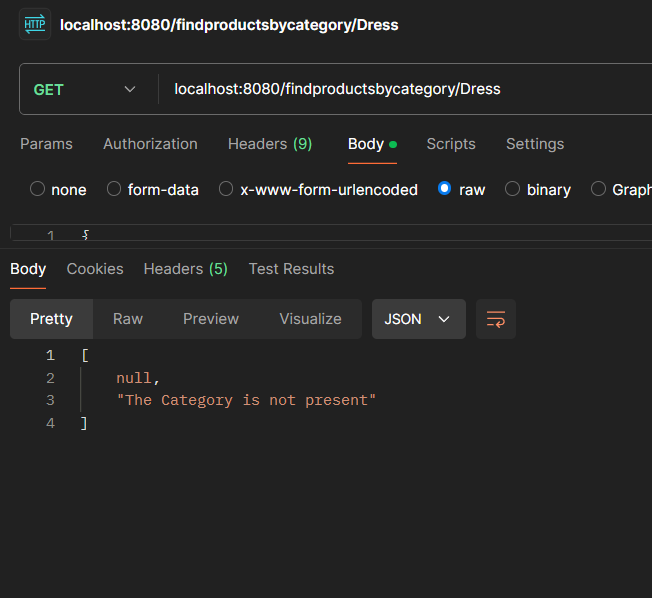
Its a **GET mapping** so we will use **GET** so that it will fetch the entire data from monog db and will see **if the category is store in DB or not.**

****

**Exception : If the category is not present in the DB.**

Likethe API I have use, have Dress has category and we don’t have dress has category.

**localhost:8080/findproductsbycategory/Dress**

****

1. Now we will do the Buying of the product by the user

To do the Buying of the product by the user we will use the API – **localhost:8080/fiaddtocart (Example of one product from our DB).**

Its a **Post mapping** so we will use **POST** so that it will take data from the payload and store it in our DB.

Payload:

{

    "bookingId": "BOOK-123456",

    "totalcost": 5000,

    "quantity": 2,

    "user": {

      "userid": 100,

      "name": "Rutuj",

      "email": "rutuj@example.com"

    },

    "product": {

      "productID": 103,

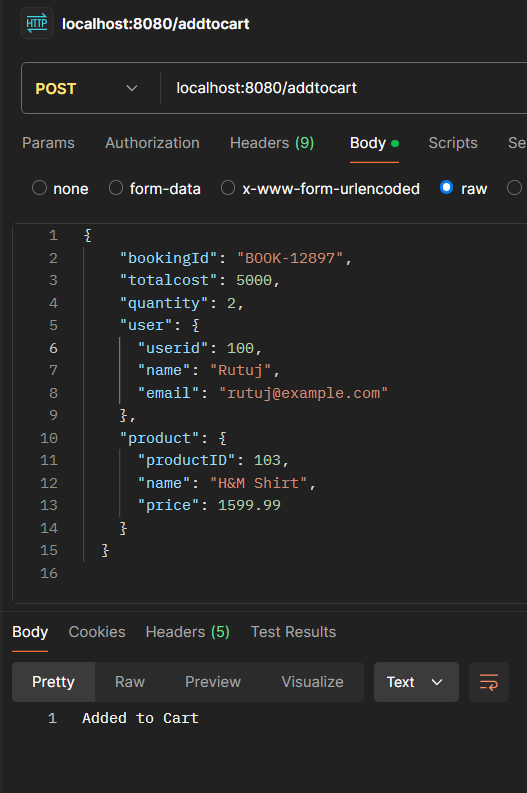
      "name": "H&M Shirts",

      "price": 1599.99

    }

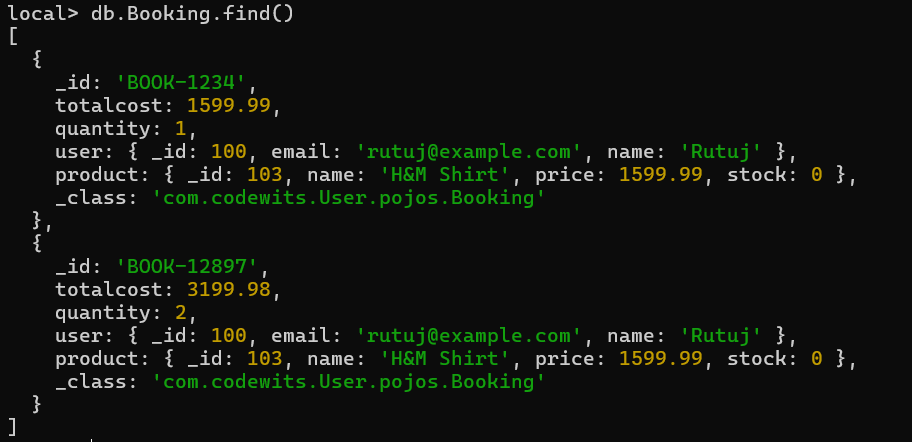
  }

You need to give this payload. Which we will take from frontend by the user.



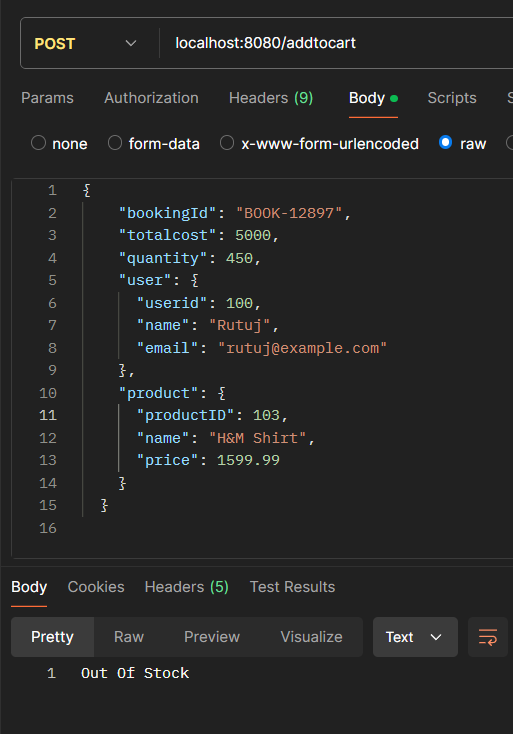
You can see the changes in the Mongo DB also

In the new database name as **Booking**



**Exception:**

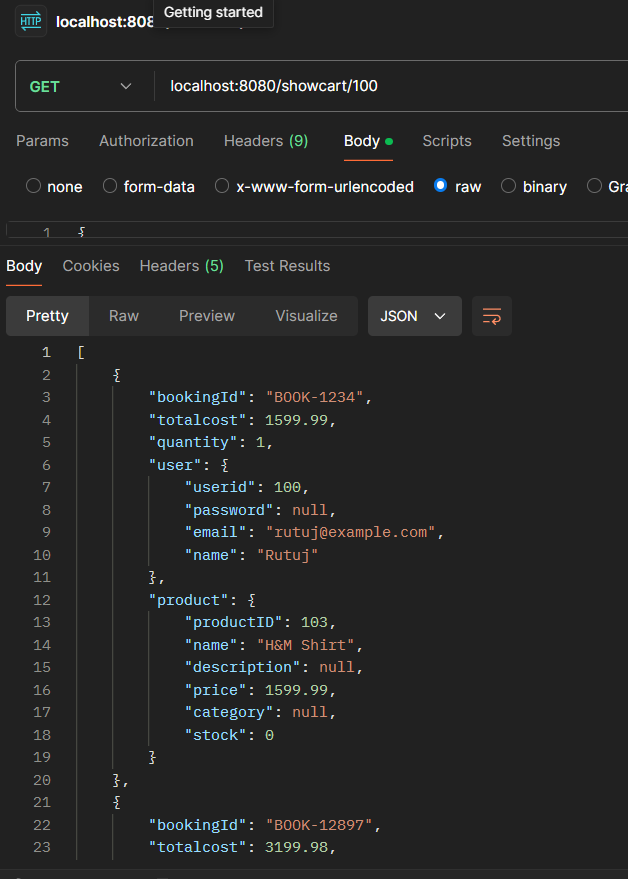
And if a user will buy a lot of quantity of a same product and in the stock we don’t have that much stock remain, it will give the expectation **“ Out of stock “**

****

1. Now we will do the Buying of the product by the user

To do the Buying of the product by the user we will use the API – **localhost:8080/** **showcart/100 (Example of one product from our DB).**

Its a **Get mapping** so we will use **Get** so that it will take data from the Database and store it in our DB.



For the Admin

For the admin we have the same as user but the difference is that it runs on the Local host 8082:

This have this API same as users, but work differently for the admin and have other database.

**localhost:8082/addadmin (POST method)**

**localhost:8082/allproduct ( GET method )**

**localhost:8082//auth/{email}/{password} ( GET method)**

**localhost:8082//findproductsbycategory/{category} ( GET method)**

This are the Same API like users, works same as I show above but different database.

1. To add a product by admin :

**localhost:8082//addproduct ( POST METHOD )**

**Payload:**

{

    "productID": 108,

    "name": "Pants",

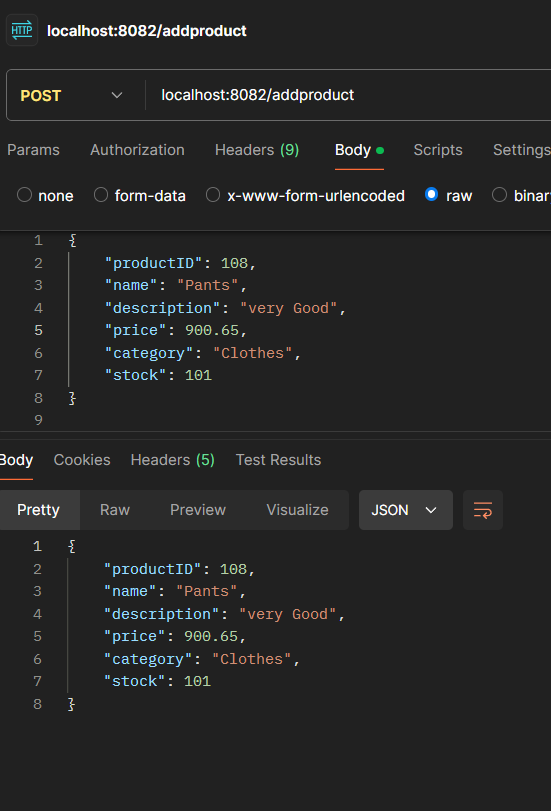
    "description": "very Good",

    "price": 900.65,

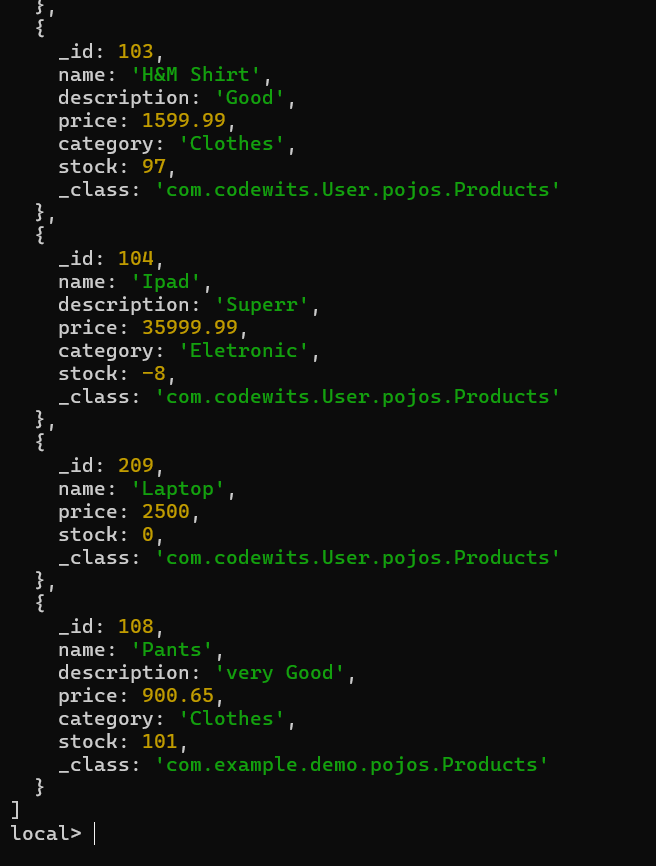
    "category": "Clothes",

    "stock": 101

}

****

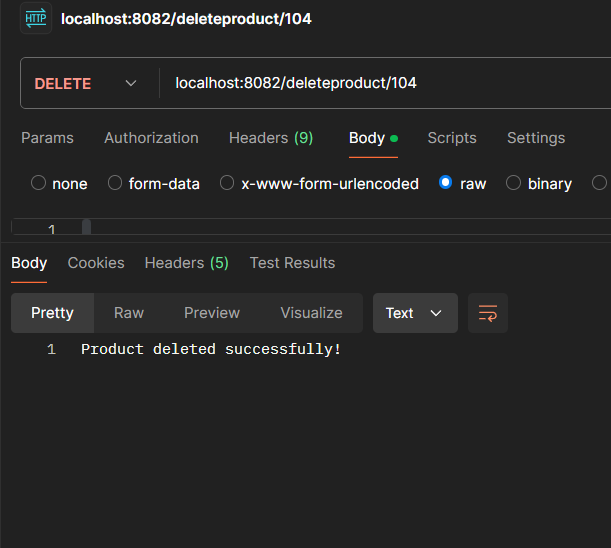
**Data base :**

****

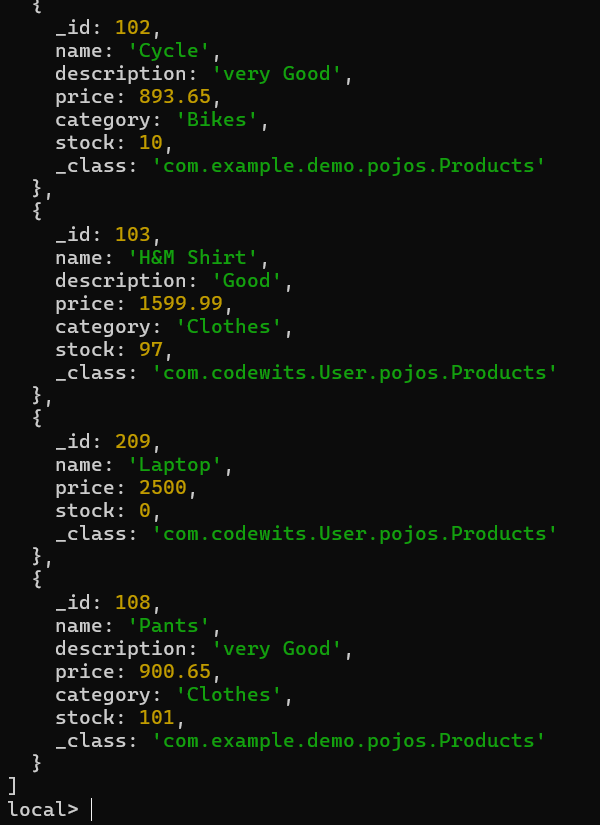
**You can see the new product have been add**

2. To Delete a product by admin :

**localhost:8082/deleteproduct/104 ( 104 is the id of the product) ( DELETE METHOD)**

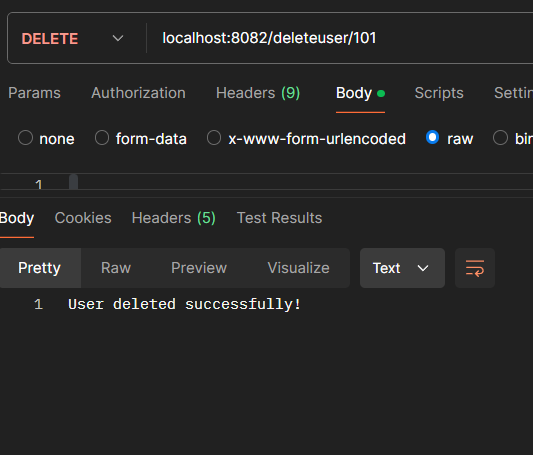
****

**In db you can see the id 104 is not available :**

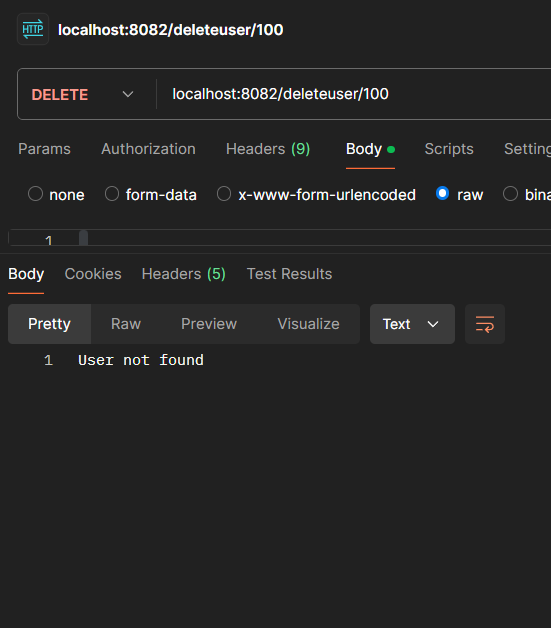
****

2. To Remove a user by admin :

**localhost:8082/deleteuser/101 ( DELETE METHOD)**



Expection if the users is not present:



Database :

You can see the user 101 is not present.



3 . To Update the products by admin :

**localhost:8082/updateproduct/102 ( PUT METHOD)**

**Before Update :**



Now we are going to update the product 102 :

Payload :

{

    "productID": 102,

    "name": "Cycle",

    "description": "very bad",

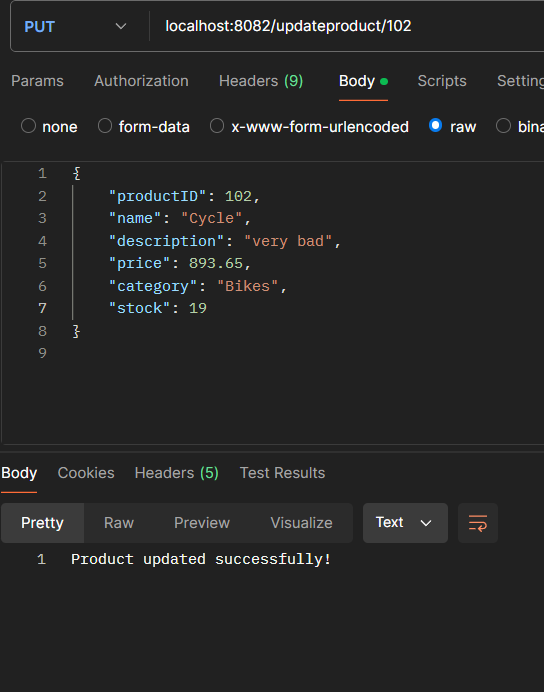
    "price": 893.65,

    "category": "Bikes",

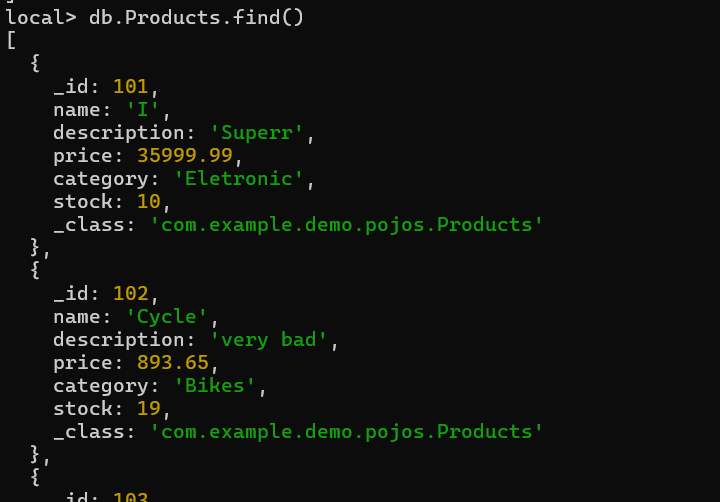
    "stock": 19

}

We have change the description and stock :



New DB with updated values of the product:



You can the product\_id = 102 has change.