Documentation

User API:

These API end points are for register and login of user. These both have validation implemented. And the validation for registration and login

username

1. Username must be at least 6 characters long
2. Username must contain at least one number and one special character.

Password

1. Password must be at least 8 characters long
2. Password must contain at least one number and one special character

Here are **API end points**.

Register:

POST---> http://localhost:3000/user/register

Body: (Password and username could be different. I used the same one just for simplicity). Remember we are storing encryptedpasswords in the database**)**.

```
{
"username":"zeeshan2@",
"email": "zeeshan2@gmail.com",
"password": "zeeshan2@",
"role": "customer"
}
```

Login

POST---> http://localhost:3000/user/login

Body:

```
{
  "username":"zeeshan1@",
  "password": "zeeshan1@"
}
```

Products

API endpoints.

Add new Product

Only the authorized user can add a new product.

Endpoint:

http://localhost:3000/product/postProduct

Header:

Authorization: bearer “Token”

Body:

```
{
  "name": "A Laptop",
  "description": "This is a good Laptop",
  "price": 499.99,
  "category": "technology",
  "tags": "lenovo"
```

}

Get All Products

Non authorized can also get all the products in the database. So no need of token in this case.

Endpoint:

GET---> http://localhost:3000/product/getAllProducts

Get Products

Get all the products added by that user in the database.

Endpoint:

GET---> http://localhost:3000/product/getProducts

Header:

Authorization bearer Token

Update Product Based on ID

Endpoint:

PUT---> http://localhost:3000/product/updateProduct/5

Header:

Authorization bearer Token

Body (Updated Object having all fields—>All field logic is that on frontend first we load previous data and then update so we will get whole updated object**)**

```
{
  "name": "A Mobile",
  "description": "This is a good Mobile",
  "price": 499.99,
  "category": "technology",
  "tags": "lenovo"
}
```

Delete A product based on ProductId

Only the user that added the product can delete that. For now this is the logic but in this we can add that admin can also delete the product.

Endpoint:

DELETE---> http://localhost:3000/product/deleteProduct/5

Header:

Authorization bearer Token

CART ENDPOINTS

GetCart

This will return all the products in the cart of the logged in user.

Endpoint

GET ---> http://localhost:3000/cart/getCart

Header:

Authorization bearer Token

Add products to cart

And check that if the product is already in the cart, then it will increase the quantity of the product.

Endpoint:

POST---> http://localhost:3000/cart/addProductToCart

Header:

Authorization bearer Token

Body:

{

“productId”: 1

}

Delete a product from cart

This will delete a product from the cart of the logged in user

Endpoint:

DELETE ---> http://localhost:3000/cart/deleteProductFromCart/4

Header:

Authorization bearer Token

ORDER

Add order

This will add the cart of the logged in user to the order table.

Endpoint:

POST---> http://localhost:3000/order/addOrder

Header:

Authorization bearer Token

Get Orders

This will return all the orders of the logged in user

Endpoint:

GET---> http://localhost:3000/order/getOrders

Header:

Authorization bearer Token

View All orders

This will return all the orders- A requirement in the text document.

Endpoint:

GET---> http://localhost:3000/order/viewAllOrders

Get order by orderId:

Endpoint:

GET ---> http://localhost:3000/order/getOrder/1

Note:

There could be some mistakes or errors in the documentation. To check that you can visit the endpoint and check from the comment in the code.