

## Testing the API

First, let's create a new Product using a Post request:

The screenshot shows a REST client interface with a POST request to `https://localhost:44350/api/product/AddProduct`. The request body is a JSON object representing a product. The response is a JSON object with an additional `productId` field.

**Request:**

```
1 {
2   "name": "Android Smart TV",
3   "description": "Full HD LED Smart Android TV",
4   "category": "Electronics",
5   "color": "Black",
6   "price": 12000,
7   "availableQuantity": 11
8 }
```

**Response:**

```
1 {
2   "productId": 3,
3   "name": "Android Smart TV",
4   "description": "Full HD LED Smart Android TV",
5   "category": "Electronics",
6   "color": "Black",
7   "price": 12000,
8   "availableQuantity": 11
9 }
```

Status: 200 OK Time: 161 ms

Next, let's do a Get request to get all Products. We can see the new product record which was created in the previous request:

The screenshot shows a REST client interface with a GET request to `https://localhost:44350/api/product/GetProducts`. The response is a JSON array containing three product records, including the one created in the previous request.

**Response:**

```
1 [
2   {
3     "productId": 1,
4     "name": "Air Conditioner",
5     "description": "Hitachi 1.5 Ton Inverter 3 Star",
6     "category": "Electronics",
7     "color": "Black",
8     "price": 40000,
9     "availableQuantity": 12
10  },
11  {
12    "productId": 2,
13    "name": "Mobile",
14    "description": "HD Pin Hole Display, 16 MP Quad Rear Camera",
15    "category": "Electronics",
16    "color": "White",
17    "price": 10000,
18    "availableQuantity": 5
19  },
20  {
21    "productId": 3,
22    "name": "Android Smart TV",
23    "description": "Full HD LED Smart Android TV",
24    "category": "Electronics",
25    "color": "Black",
26    "price": 12000,
27    "availableQuantity": 11
28  }
29 ]
```

Now, let's do a PUT request to test the update functionality by changing the Name and Price:

The screenshot shows a REST client interface with a PUT request to `https://localhost:44350/api/product/UpdateProduct`. The request body is a JSON object with the following fields: `productId` (1), `name` ("Hitachi Air Conditioner"), `description` ("Hitachi 1.5 Ton Inverter 3 Star"), `category` ("Electronics"), `color` ("Black"), `price` (50000), and `availableQuantity` (12). The interface includes tabs for Authorization, Headers (1), Body, Pre-request Script, and Tests. The Body tab is selected, and the request is formatted as raw JSON.

```
1 {  
2   "productId": 1,  
3   "name": "Hitachi Air Conditioner",  
4   "description": "Hitachi 1.5 Ton Inverter 3 Star",  
5   "category": "Electronics",  
6   "color": "Black",  
7   "price": 50000,  
8   "availableQuantity": 12  
9 }
```

Once again let's do a GET request and verify that the Name and Price has changed:

The screenshot shows a REST client interface with a GET request to `https://localhost:44350/api/product/GetProduct?productId=1`. The response is a JSON object with the following fields: `productId` (1), `name` ("Hitachi Air Conditioner"), `description` ("Hitachi 1.5 Ton Inverter 3 Star"), `category` ("Electronics"), `color` ("Black"), `price` (40000), and `availableQuantity` (12). The interface includes tabs for Pretty, Raw, and Preview. The Pretty tab is selected, and the response is formatted as JSON.

```
1 {  
2   "productId": 1,  
3   "name": "Hitachi Air Conditioner",  
4   "description": "Hitachi 1.5 Ton Inverter 3 Star",  
5   "category": "Electronics",  
6   "color": "Black",  
7   "price": 40000,  
8   "availableQuantity": 12  
9 }
```

Now let's do a Delete request to test the delete functionality by deleting product having product Id 3

The screenshot shows a REST client interface with a DELETE request to `https://localhost:44350/api/product/DeleteProduct?productId=3`. The request is formatted as raw text. The interface includes tabs for Authorization, Headers (1), Body, Pre-request Script, and Tests. The Body tab is selected, and the request is formatted as raw text. The response status is 200 OK and the time is 2084 ms.

```
1
```

Once again let's do a GET request and verify that deleted product:

GET

https://localhost:44350/api/product/GetProduct?productId=3

Params

Send

Save

AuthorizationHeaders (1)BodyPre-request ScriptTestsCode

TypeNo Auth

BodyCookiesHeaders (4)Test ResultsStatus: 404 Not FoundTime: 76 ms

PrettyRawPreviewJSON

```
1 {
2   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
3   "title": "Not Found",
4   "status": 404,
5   "traceId": "|889f28eb-483af485c6c9a67a."
6 }
```