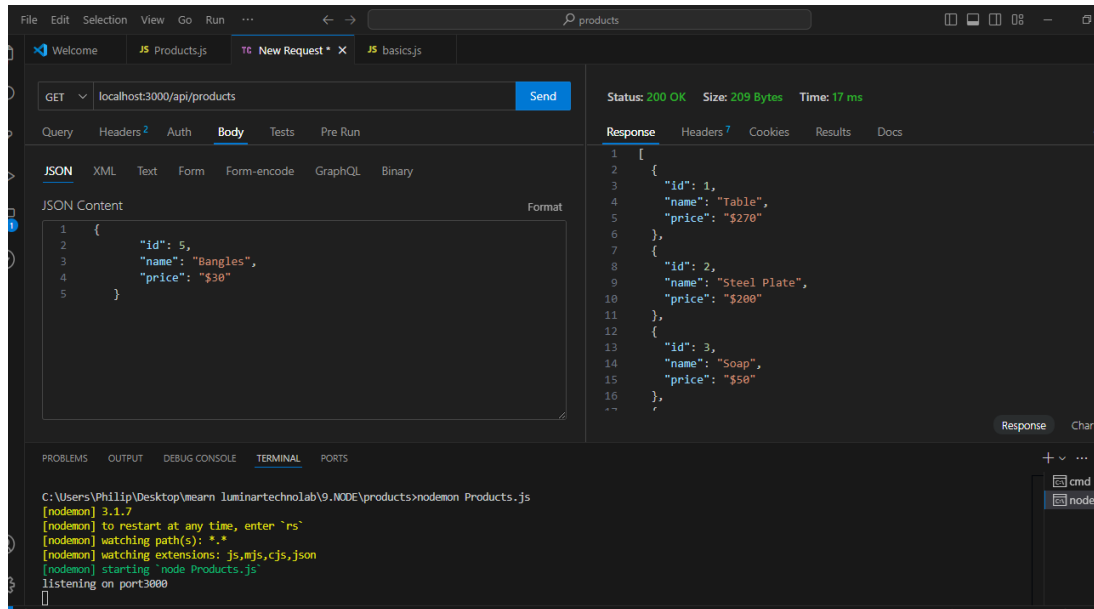


## Task

Implement a basic Node.js Express server that performs CRUD operations on a list of products.

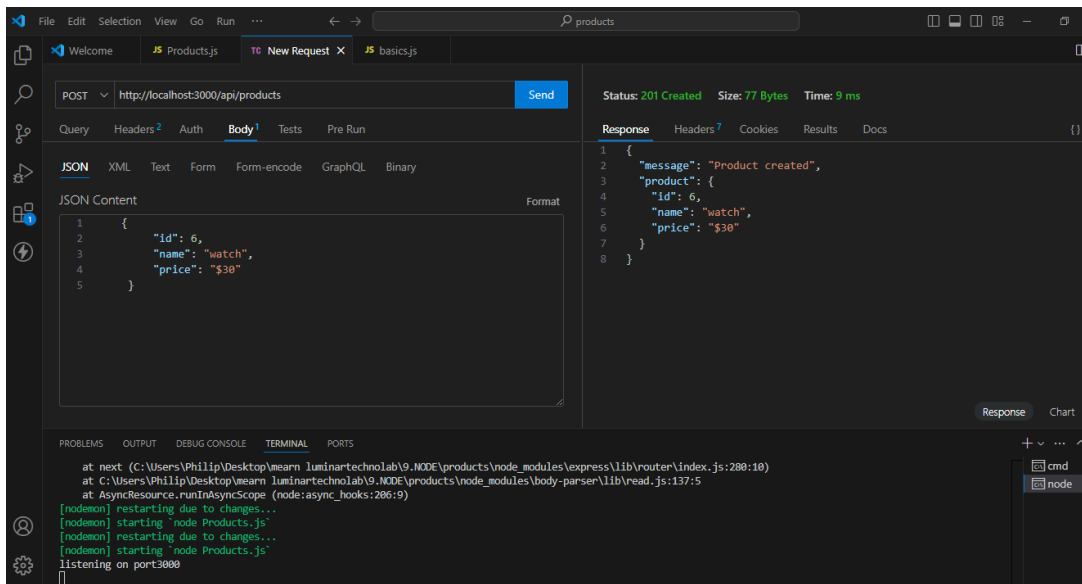
1. Read all products: Implement a GET request to retrieve all products.



### Code:

```
// READ all products (GET /api/products)
app.get('/api/products', (req, res) => {
  res.status(200).json(products)
})
```

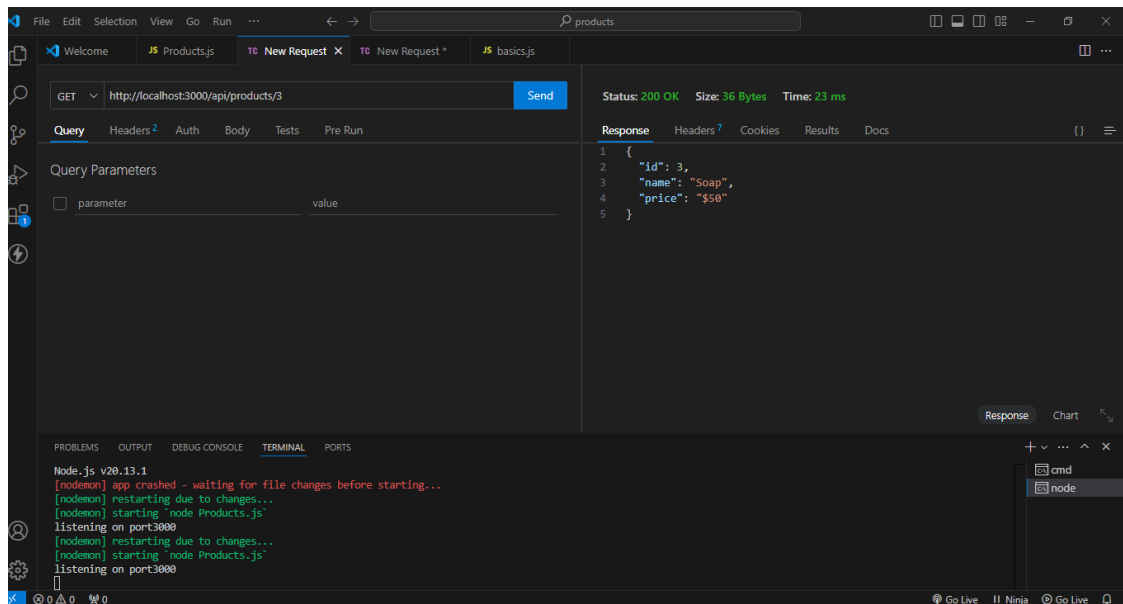
2. Create a product: Implement a POST request to add a new product with fields like id, name, and price.



Code:

```
3
4 // CREATE a new product (POST /api/products)
5 app.post('/api/products', (req, res) => {
6   const { id, name, price } = req.body;
7   if (!id || !name || !price) {
8     return res.status(400).json({ message: "Invalid product data" });
9   }
10  const productExists = products.find(product => product.id === id);
11  if (productExists) {
12    return res.status(400).json({ message: "Product with the same ID already exists" });
13  }
14  const newProduct = { id, name, price };
15  products.push(newProduct);
16  res.status(201).json({ message: "Product created", product: newProduct });
17 });
```

**3. Read a single product: Implement a GET request to fetch a product by its ID.**



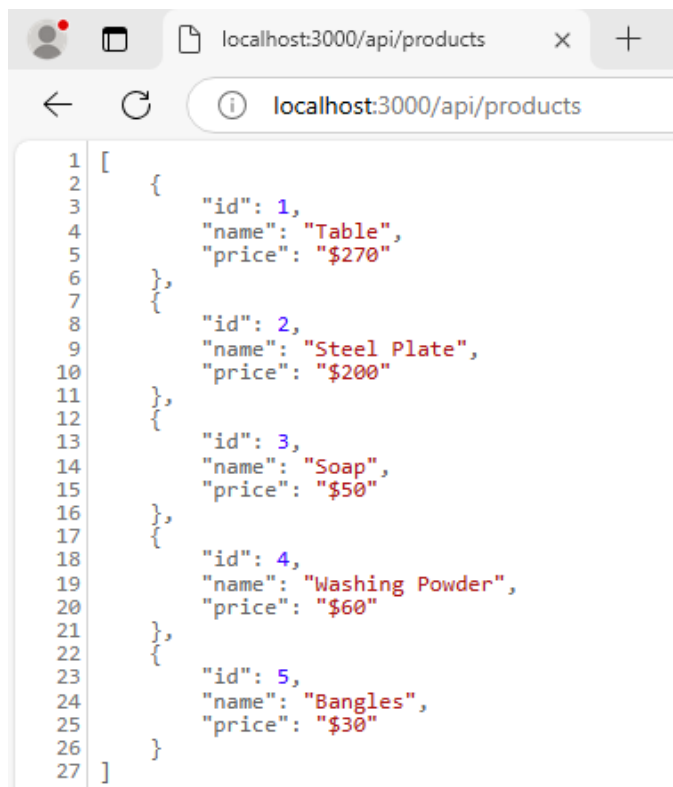
## Code:

```
//Get a specific product by id
app.get('/api/products/:id', (req, res) => {
  const productId = parseInt(req.params.id);
  const product = products.find(product => product.id === productId);

  if (!product) {
    return res.status(404).json({ message: "Product not found" });
  }
  res.status(200).json(product);
});
```

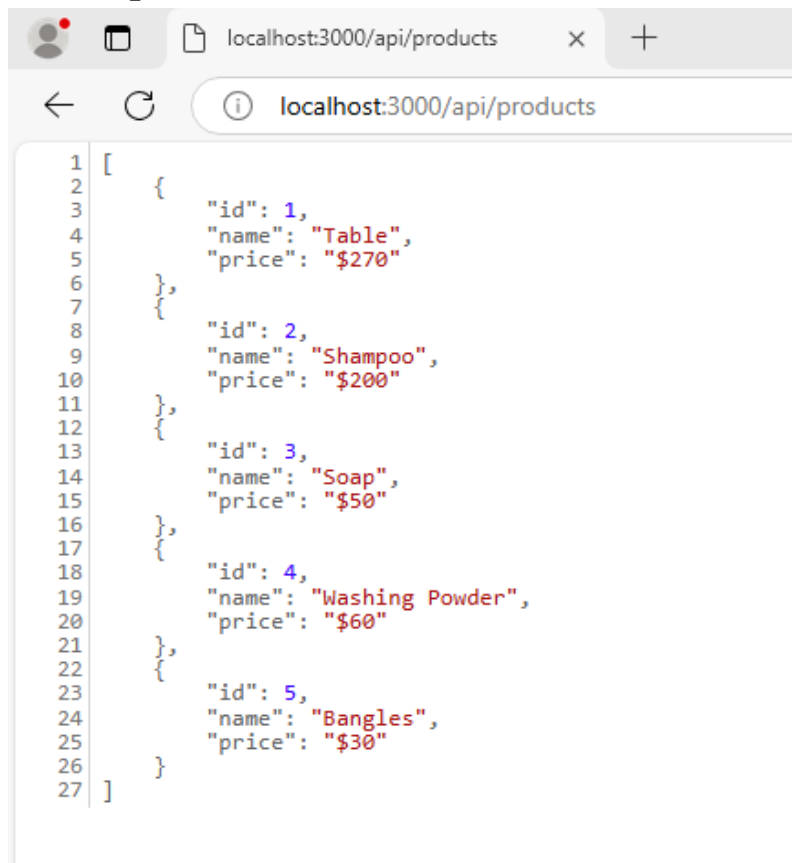
4. Update a product: Implement a PUT request to update the name and price of a product by its ID.

## Before Update:



```
1 [
2   {
3     "id": 1,
4     "name": "Table",
5     "price": "$270",
6   },
7   {
8     "id": 2,
9     "name": "Steel Plate",
10    "price": "$200",
11  },
12  {
13    "id": 3,
14    "name": "Soap",
15    "price": "$50",
16  },
17  {
18    "id": 4,
19    "name": "Washing Powder",
20    "price": "$60",
21  },
22  {
23    "id": 5,
24    "name": "Bangles",
25    "price": "$30",
26  }
27 ]
```

### After Update:



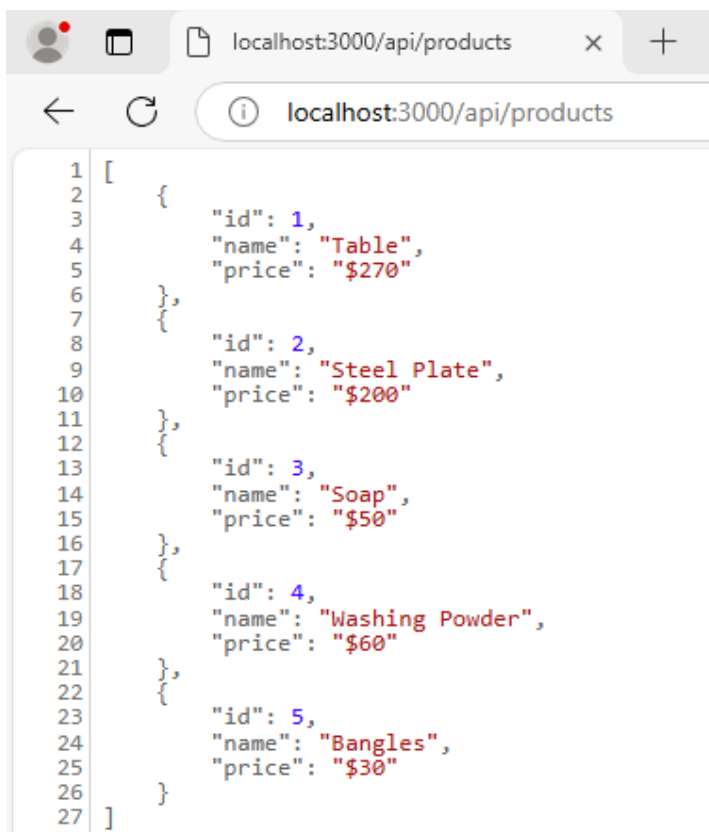
```
1 [
2   {
3     "id": 1,
4     "name": "Table",
5     "price": "$270",
6   },
7   {
8     "id": 2,
9     "name": "Shampoo",
10    "price": "$200",
11  },
12  {
13    "id": 3,
14    "name": "Soap",
15    "price": "$50",
16  },
17  {
18    "id": 4,
19    "name": "Washing Powder",
20    "price": "$60",
21  },
22  {
23    "id": 5,
24    "name": "Bangles",
25    "price": "$30",
26  }
27 ]
```

## Code:

```
4
5 // UPDATE a product by ID (PUT /api/products/:id) http://localhost:3000/api/products/1
6 app.put('/api/products/:id', (req, res) => {
7   const productId = parseInt(req.params.id);
8   const { name, price } = req.body;
9   const product = products.find(product => product.id === productId);
10  if (!product) {
11    return res.status(404).json({ message: "Product not found" });
12  }
13  if (name) product.name = name;
14  if (price) product.price = price;
15  res.status(200).json({ message: "Product updated", product });
16 });
17
```

5. Delete a product: Implement a DELETE request to remove a product by its ID.

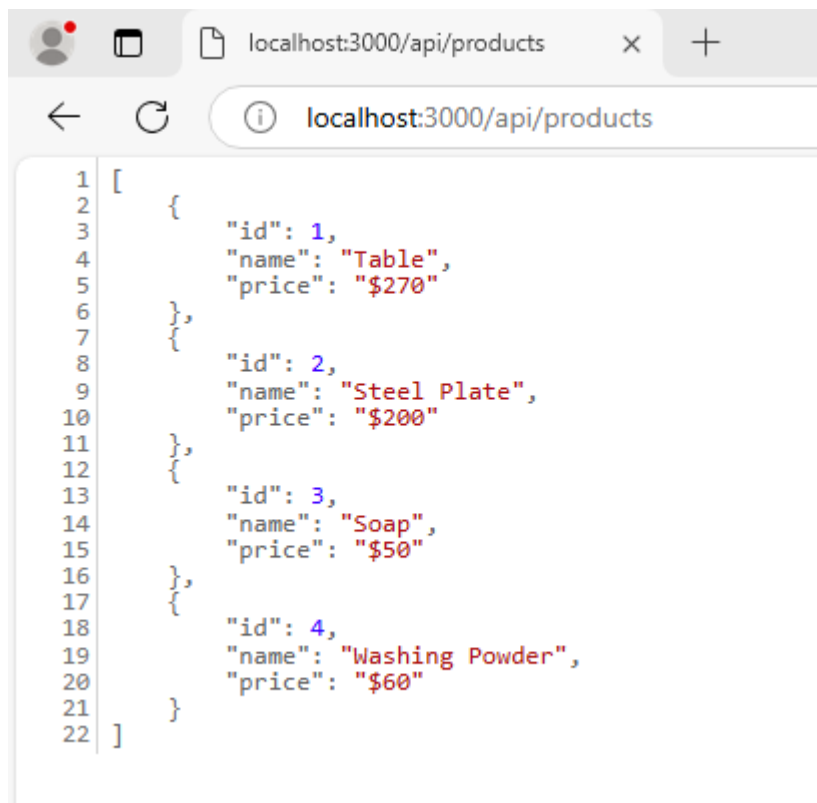
## Before Delete:



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/api/products'. The page content shows a JSON array of 5 products:

```
1 [
2   {
3     "id": 1,
4     "name": "Table",
5     "price": "$270"
6   },
7   {
8     "id": 2,
9     "name": "Steel Plate",
10    "price": "$200"
11  },
12  {
13    "id": 3,
14    "name": "Soap",
15    "price": "$50"
16  },
17  {
18    "id": 4,
19    "name": "Washing Powder",
20    "price": "$60"
21  },
22  {
23    "id": 5,
24    "name": "Bangles",
25    "price": "$30"
26  }
27 ]
```

## After Delete



```
1 [
2   {
3     "id": 1,
4     "name": "Table",
5     "price": "$270"
6   },
7   {
8     "id": 2,
9     "name": "Steel Plate",
10    "price": "$200"
11  },
12  {
13    "id": 3,
14    "name": "Soap",
15    "price": "$50"
16  },
17  {
18    "id": 4,
19    "name": "Washing Powder",
20    "price": "$60"
21  }
22 ]
```

**Code:**

```
//delete a post from the array
app.delete('/api/products/:id',(req,res)=>{
  ⚡ const pid = parseInt(req.params.id)
  const product =products.find((p)=>p.id === pid)
  if(!product){
    res.status(404).send("No product found")
  }
  else{
    products=products.filter((p)=>p.id !== pid)
    res.status(200).json(products)
  }
})
```