

NAME: Akash K Raj  
EMP ID: 4009205

#### Week 4 Assessment

##### Question:

I. Project template is provided .

Need to implement

1. RestController
2. ProductService
3. ProductDao with one custom query

3. Create ProductNotFoundException in com.verizon.expection package

```
Perform All 1. void AddProduct(),  
                2. List<Product> GetAllProducts(),  
                3. Product GetProductById(Integer id) throws  
ProductNotFoundException  
                4. List<Product> GetAllProductsBetweenPrice(Integer  
low,Integer high);  
                3. void UpdateProduct(Integer id,Product  
product)throws ProductNotFoundException  
                4. void DeleteProduct(Integer id)throws  
ProductNotFoundException
```

submit project source code in Zip format

Send screenshots of postman for all above operations in Word doc.

**Screenshots of Postman Implementations:**

## 1. Insertion of a product into database:

The screenshot shows the Postman interface with a POST request configured. The URL is `http://localhost:8080/api/v1/products`. The request body is set to JSON and contains the following data:

```
{  "name": "Cupboard",  "price": 399.99}
```

The response status is 201 Created, with a time of 14 ms and a size of 128 B. The response body is currently empty.

## 2. Retrieving All products from database:

The screenshot shows the Postman interface with a GET request configured. The URL is `http://localhost:8080/api/v1/products`. The response status is 200 OK, with a time of 14 ms and a size of 363 B. The response body is displayed in JSON format, showing an array of three product objects:

```
[  {    "id": 5,    "name": "Bed",    "price": 299.99  },  {    "id": 6,    "name": "Bean Bag",    "price": 99.99  },  {    "id": 8,    "name": "Cupboard",    "price": 399.99  }]
```

### 3. Retrieving product with a specific ID. Example ID = 3:

Postman interface showing a successful GET request to `http://localhost:8080/api/v1/products/3`. The response status is 200 OK, Time: 10 ms, Size: 202 B. The response body is displayed in JSON format:

```
1 {
2   "id": 3,
3   "name": "Table",
4   "price": 249.99
5 }
```

### 4. ProductNotFoundException:

Postman interface showing a failed GET request to `http://localhost:8080/api/v1/products/2`. The response status is 404 Not Found, Time: 12 ms, Size: 272 B. The response body is displayed in JSON format:

```
1 {
2   "status": 404,
3   "error": "Not Found",
4   "message": "Product not found with ID: 2",
5   "path": "/api/v1/products"
6 }
```

5. Retrieving products between a specific price range. The price is passed as query parameters in this case.

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/v1/products/price?low=100&high=300`. The response status is 200 OK, and the body contains a JSON array of three products.

**Request:**

- Method: GET
- URL: `http://localhost:8080/api/v1/products/price?low=100&high=300`

**Query Params:**

Key	Value
low	100
high	300

**Response Body (JSON):**

```
1 {
2   {
3     "id": 3,
4     "name": "Table",
5     "price": 249.99
6   },
7   {
8     "id": 4,
9     "name": "Chair",
10    "price": 119.99
11  },
12  {
13    "id": 5,
14    "name": "Bed",
15    "price": 299.99
16  }
17 }
```

**Status:** 200 OK, Time: 8 ms, Size: 280 B

6. Updating a specific product, using productID. In this case we update the product with ID 3:  
Before updating:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/v1/products/3`. The response status is 200 OK, and the body is a JSON object:

```
{
  "id": 3,
  "name": "Table",
  "price": 199.99
}
```

Performing Update operation:

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/v1/products/3`. The request body is a JSON object:

```
{
  "name": "Dining Table",
  "price": 349.99
}
```

The response status is 204 No Content.

## After update operation:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/v1/products/3`. The request is successful, returning a 200 OK status. The response body is displayed in JSON format:

```
{
  "id": 3,
  "name": "Dining Table",
  "price": 349.99
}
```

The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, showing the response in Pretty, Raw, Preview, and Visualize formats. The response status is 200 OK, with a time of 10 ms and a size of 209 B.

## Trying to update a product with a productID that doesn't exist in the database:

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/v1/products/2`. The request fails, returning a 404 Not Found status. The response body is displayed in JSON format:

```
{
  "status": 404,
  "error": "Not Found",
  "message": "Product not found with ID: 2",
  "path": "/api/v1/products"
}
```

The interface includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, showing the response in Pretty, Raw, Preview, and Visualize formats. The response status is 404 Not Found, with a time of 12 ms and a size of 272 B.

## 7. Deleting a product successfully:

All the products before deleting a product with ID 8:

Postman interface showing a GET request to `http://localhost:8080/api/v1/products`. The response is a JSON array of three products:

```
[{"id": 5, "name": "Bed", "price": 299.99}, {"id": 6, "name": "Bean Bag", "price": 99.99}, {"id": 8, "name": "Cupboard", "price": 399.99}]
```

Status: 200 OK Time: 10 ms Size: 370 B

## Performing Delete Operation:

Postman interface showing a DELETE request to `http://localhost:8080/api/v1/products/8`. The response is a 204 No Content status.

Status: 204 No Content Time: 19 ms Size: 112 B

Now displaying all the products in the database:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/v1/products`. The response is a JSON array of three product objects, each with `id`, `name`, and `price` fields. The status is 200 OK, and the response is displayed in the 'Body' tab using the 'Pretty' view.

Key	Value	Bulk Edit
Key	Value	

```
1 {  
2   "id": 4,  
3   "name": "Chair",  
4   "price": 119.99  
5 },  
6 {  
7   "id": 5,  
8   "name": "Bed",  
9   "price": 299.99  
10 },  
11 {  
12   "id": 6,  
13   "name": "Bean Bag",  
14   "price": 99.99  
15 }  
16 ]
```

Trying to delete a product with a productID that doesn't exist in the database:

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/v1/products/8`. The response is a JSON object indicating a 404 Not Found status, with an error message: "Product not found with ID: 8". The status is 404 Not Found, and the response is displayed in the 'Body' tab using the 'Pretty' view.

Key	Value	Bulk Edit
Key	Value	

```
1 {  
2   "status": 404,  
3   "error": "Not Found",  
4   "message": "Product not found with ID: 8",  
5   "path": "/api/v1/products"  
6 }
```

THANK YOU