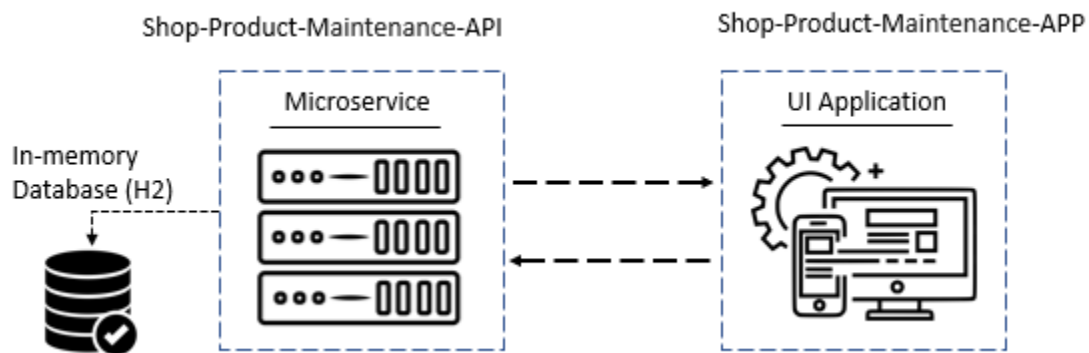


## SHOP-PRODUCT-MAINTAINANCE-APPLICATION

### Technologies Used:

1. Programming Language: Java8, Typescript
2. Backend Framework: SpringBoot2
3. Frontend Framework: Angular
4. Database: H2 (In-Memory Database)
5. Styling and Styling Framework: Bootstrap, CSS
6. Testing Framework: Junit, Mockito

### Application Flow:



### Module Description:

#### 1. Shop-Product-Maintenance-API:

API Description	API to update the product details by providing the id of the record and the updated product details.
Http Method	Put
URL	<a href="http://localhost:8080//shop-product-maintainance/v1/product/{id}">http://localhost:8080//shop-product-maintainance/v1/product/{id}</a>
Sample Request Body	<pre>{   "productName": "Updated Product Name",   "productDesc": "Updated Desc",   "productPrice": 0.0,   "productQuantity": 0,   "productPriceCurrency": "INR" }</pre>
Sample Response Body	<pre>{   "productName": "Updated Product Name",   "productDesc": "Updated Desc",   "productPrice": 0.0,   "productQuantity": 0,   "productPriceCurrency": "INR" }</pre>

	}
--	---

API Description	API to Delete the product details by providing the id of the record.
Http Method	Delete
URL	<a href="http://localhost:8080//shop-product-maintainance/v1/product/{id}">http://localhost:8080//shop-product-maintainance/v1/product/{id}</a>
Sample Request Body	NA
Sample Response Body	<pre>{   "deleted": true }</pre>

API Description	API to get all the product details present in the H2 database.
Http Method	Get
URL	<a href="http://localhost:8080//shop-product-maintainance/v1/product">http://localhost:8080//shop-product-maintainance/v1/product</a>
Sample Request Body	NA
Sample Response Body	<pre>[   {     "id": 1,     "productName": "Updated Product Name",     "productDesc": "Updated Desc",     "productCategory": null,     "productPrice": 0.0,     "productQuantity": 0,     "productPriceCurrency": "INR"   },   {     "id": 2,     "productName": "Updated Product Name",     "productDesc": "Updated Desc",     "productCategory": null,</pre>

	<pre> "productPrice": 0.0, "productQuantity": 0, "productPriceCurrency": "INR" } ] </pre>
--	---

API Description	API to get the product details of a particular product by providing the id of the record.
Http Method	Get
URL	<a href="http://localhost:8080//shop-product-maintainance/v1/product/{id}">http://localhost:8080//shop-product-maintainance/v1/product/{id}</a>
Sample Request Body	NA
Sample Response Body	<pre> {   "id": 2,   "productName": "Updated Product Name",   "productDesc": "Updated Desc",   "productCategory": null,   "productPrice": 0.0,   "productQuantity": 0,   "productPriceCurrency": "INR" } </pre>
Error Response Body	<pre> {   "timestamp": "2021-03-18T15:36:54.861+00:00",   "message": "Product not found for this id :: 1",   "details": "uri=//shop-product-maintainance/v1/product/1" } </pre>

API Description	Event-Producer-Service produces and commits scheduled events at every 2 mins to Kafka broker. As of now for demo purpose, it is configured to Mumbai as cityName.
Http Method	POST
URL	<a href="http://localhost:8080//shop-product-maintainance/v1/product">http://localhost:8080//shop-product-maintainance/v1/product</a>
Sample Request Body	<pre> {   "productName": "Updated Product Name",   "productDesc": "Updated Desc", </pre>

	<pre>"productPrice": 0.0, "productQuantity": 0, "productPriceCurrency": "INR" }</pre>
Sample Response Body	<pre>{   "id": 1,   "productName": "Updated Product Name",   "productDesc": "Updated Desc",   "productCategory": null,   "productPrice": 0.0,   "productQuantity": 0,   "productPriceCurrency": "INR" }</pre>

## JUnit Test Cases and Code Coverage Report:

Debug Project Explorer Servers JUnit

Finished after 13.521 seconds

Runs: 14/14 Errors: 0 Failures: 0

▼ ProductControllerTest [Runner: JUnit 5] (2.300 s)

- Test-9: To test if exception is thrown when we try to delete a record based on id but record is not present (0.001 s)
- Test-8: To test if the call is successful when we try to delete a record (0.001 s)
- Test-6: To test if the call is successful when we try to update a record (0.000 s)
- Test-7: To test if exception is thrown when we try to update a record based on id but record is not present (0.000 s)
- Test-4: To test if the call is successful when we try to fetch a record based on id (0.000 s)
- Test-3: To test if exception is thrown when we try to fetch a record based on id but record is not present (0.000 s)
- Test-2: To test if all available products in H2 are received (0.000 s)
- Test-1: To test if exception is thrown when we try to fetch all records but no records present (0.000 s)
- Test-5: To test if the call is successful when we try to create a new record (2.298 s)

▼ ProductControllerIntegrationTest [Runner: JUnit 5] (0.634 s)

- Test-3: To test if the call is successful when we try to create a new record (0.552 s)
- Test-4: To test if the call is successful when we try to update a record (0.030 s)
- Test-2: To test if the call is successful when we try to fetch a record based on id (0.013 s)
- Test-5: To test if the call is successful when we try to delete a record (0.027 s)
- Test-1: To test if all available products in H2 are received (0.012 s)

Coverage

Element	Coverage	Covered Instruction...	Missed Instructions	Tot
▼ shop-product-maintenance-api	88.4 %	725	95	
▼ src/main/java	80.5 %	273	66	
> com.shopapplication.services.exceptions	10.9 %	7	57	
> com.shopapplication.services	37.5 %	3	5	
> com.shopapplication.services.model	95.6 %	86	4	
> com.shopapplication.services.controller	100.0 %	177	0	
> src/test/java	94.0 %	452	29	

### CASE-1: Adding a product, Getting HTTP-Status <202> and new product is added.

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/shop-product-maintenance/v1/product`. The request body is a JSON object with the following fields: `productName`, `productDesc`, `productCategory`, `productPrice`, `productQuantity`, and `productPriceCurrency`. The response status is `202 Accepted` with a response time of `56 ms` and a body size of `770 B`. The response body is a JSON object with an `id` field set to `1`.

```
POST http://localhost:8080/shop-product-maintenance/v1/product

{
  "productName": "Fabelle - The Bars Treasury, Pack of 2x235g, Assorted Chocolate Gift Pack.",
  "productDesc": "Fabelle - The Bars Treasury is an assorted chocolate gift pack filled with specially curated collection of centre filled bars and layered chocolate bars. Handcrafted by Master Chocolatiers, Fabelle - The Bars is perfect for gifting as it makes moments even more special filled with goodness.",
  "productCategory": "Chocolate",
  "productPrice": 560.0,
  "productQuantity": 1,
  "productPriceCurrency": "INR"
}
```

```
202 Accepted 56 ms 770 B

{
  "id": 1,
  "productName": "Fabelle - The Bars Treasury, Pack of 2x235g, Assorted Chocolate Gift Pack.",
  "productDesc": "Fabelle - The Bars Treasury is an assorted chocolate gift pack filled with specially curated collection of centre filled bars and layered chocolate bars. Handcrafted by Master Chocolatiers, Fabelle - The Bars is perfect for gifting as it makes moments even more special filled with goodness.",
  "productCategory": "Chocolate",
  "productPrice": 560.0,
  "productQuantity": 1,
  "productPriceCurrency": "INR"
}
```

### CASE-2: Editing a product details, Getting HTTP-Status <200> and user details are updated successfully.

The screenshot displays a REST client interface with a PUT request to `http://localhost:8080/shop-product-maintenance/v1/product/1`. The request body is a JSON object with the same fields as in Case 1, but the `productPrice` is updated to `600.0`. The response status is `200 OK` with a response time of `27 ms` and a body size of `764 B`. The response body is a JSON object with an `id` field set to `1`.

```
PUT http://localhost:8080/shop-product-maintenance/v1/product/1

{
  "productName": "Fabelle - The Bars Treasury, Pack of 2x235g, Assorted Chocolate Gift Pack.",
  "productDesc": "Fabelle - The Bars Treasury is an assorted chocolate gift pack filled with specially curated collection of centre filled bars and layered chocolate bars. Handcrafted by Master Chocolatiers, Fabelle - The Bars is perfect for gifting as it makes moments even more special filled with goodness.",
  "productCategory": "Chocolate",
  "productPrice": 600.0,
  "productQuantity": 1,
  "productPriceCurrency": "INR"
}
```

```
200 OK 27 ms 764 B

{
  "id": 1,
  "productName": "Fabelle - The Bars Treasury, Pack of 2x235g, Assorted Chocolate Gift Pack.",
  "productDesc": "Fabelle - The Bars Treasury is an assorted chocolate gift pack filled with specially curated collection of centre filled bars and layered chocolate bars. Handcrafted by Master Chocolatiers, Fabelle - The Bars is perfect for gifting as it makes moments even more special filled with goodness.",
  "productCategory": "Chocolate",
  "productPrice": 600.0,
  "productQuantity": 1,
  "productPriceCurrency": "INR"
}
```

**CASE-3: Getting a product details based on product id, Getting HTTP-Status <200> and product details received successfully.**

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/shop-product-maintenance/v1/product/1`. The response status is 200 OK, with a response time of 7 ms and a body size of 764 B. The response body is displayed in a pretty-printed JSON format.

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Key	Value	Description			

```
{
  "id": 1,
  "productName": "Fabelle - The Bars Treasury, Pack of 2x235g, Assorted Chocolate Gift Pack.",
  "productDesc": "Fabelle - The Bars Treasury is an assorted chocolate gift pack filled with specially curated collection of centre filled bars and layered chocolate bars. Handcrafted by Master Chocolatiers, Fabelle - The Bars is perfect for gifting as it makes moments even more special filled with goodness.",
  "productCategory": "Chocolate",
  "productPrice": 600.0,
  "productQuantity": 1,
  "productPriceCurrency": "INR"
}
```

**CASE-4: Getting a product details based on product id, Getting HTTP-Status <404> Not Found as product details were not found in database for the given id.**

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/shop-product-maintenance/v1/product/2`. The response status is 404 Not Found, with a response time of 6 ms and a body size of 409 B. The response body is displayed in a pretty-printed JSON format.

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Key	Value	Description			

```
{
  "timestamp": "2021-03-18T13:34:39.184+00:00",
  "message": "Product not found for this id :: 2",
  "details": "uri=/shop-product-maintenance/v1/product/2"
}
```

**CASE-5: Deleting product details based on product id, Getting HTTP-Status <404> Not Found as product details were not found in database for the given id.**

The screenshot shows a REST client interface with a DELETE method selected for the URL `http://localhost:8080/shop-product-maintenance/v1/product/2`. The response status is `404 Not Found` with a response time of 15 ms and a body size of 409 B. The response body is displayed in JSON format:

```
{  "timestamp": "2021-03-18T13:35:17.564+00:00",  "message": "Product not found for this id :: 2",  "details": "uri=//shop-product-maintenance/v1/product/2"}
```

**CASE-6: Deleting product details based on product id, Getting HTTP-Status <200> and the given record is deleted successfully from the H2 Database.**

The screenshot shows a REST client interface with a DELETE method selected for the URL `http://localhost:8080/shop-product-maintenance/v1/product/1`. The response status is `200 OK` with a response time of 14 ms and a body size of 269 B. The response body is displayed in JSON format:

```
{  "deleted": true}
```