# Phase 1 REST services

## Online Shopping Management System

**Business Utility:**

The Online Shopping Management System is designed to provide a seamless and user-friendly online shopping experience. It allows customers to browse and purchase products from various categories and track their orders. The system aims to simplify the shopping process for customers and provide an efficient platform for businesses to showcase and sell their products online.

**Web Services to Implement:**

Product Services: This service manages the products available for purchase. It provides endpoints for retrieving a list of products, searching for products based on various criteria, adding new products, updating existing products, and deleting products from the system:

* GET /products: Retrieves a list of all products.
* GET /products/{productId}: Retrieves a specific product by its ID.
* POST /products: Creates a new product.
* PUT /products/{productId}: Updates an existing product.
* DELETE /products/{productId}: Deletes an existing product.

Order Services: These services handle the order placement and tracking functionality. It offers endpoints for creating new orders, retrieving order details, updating order status, and canceling orders if necessary.

* GET /orders: Retrieves a list of all orders.
* GET /orders/{orderId}: Retrieves a specific order by its ID.
* POST / customers/{ customerId}/orders: Creates a new order for a customer.
* GET / customers/{ customerId}/orders: Retrieves all orders for a specific customer.
* PUT /orders/{orderId}/status: Updates the status of an existing order.
* DELETE /orders/{orderId}: Cancels an existing order.

Customer Services: These services handle the customers using the system. It offers endpoints for creating, retrieving customer details, updating and deleting customers from the system:

* GET /customers: Retrieves a list of all customers.
* GET /customers/{customerId}: Retrieves a specific customer by their ID.
* POST /customers: Creates a new customer.
* PUT /customers/{ customerId}: Updates an existing customer.
* DELETE /customers/{ customerId}: Deletes an existing customer.

**Data Outputs, Data Inputs:**

The system will provide data outputs such as product listings and order details. Data inputs include product information for creation and updates and order details for order placement.

**Business Processes:**

The implemented web services will support various business processes, including product management (adding, updating, and deleting products), and order placement and tracking.

**Data Structure:**

The system will use JSON for representing product information, and order details.

The Online Shopping Management System aims to be reusable, extensible, and modular. It provides a set of well-defined web services that can be consumed by different client applications. The modular design allows for easy integration with existing systems or future enhancements, ensuring flexibility and scalability.

## Postman Screenshots of Services

### 1. Get all Products:

A screenshot of a computer

Description automatically generated with medium confidence

### 2. Get Product by Product ID:

A screenshot of a computer

Description automatically generated with medium confidence

### 3. Create new Product: A screenshot of a computer Description automatically generated with low confidence

### 4. Update existing Product by Product ID: A screenshot of a computer Description automatically generated with medium confidence

### 5. Delete existing Product by Product ID: A screenshot of a computer Description automatically generated with medium confidence 6. Get all Orders: A screenshot of a computer Description automatically generated with medium confidence

### 7. Get Order by Order ID: A screenshot of a computer Description automatically generated with medium confidence

### 8. Get Orders for a Customer by Customer ID: A screenshot of a computer Description automatically generated with medium confidence

### 9. Create new Order for Customer by Customer ID: A screenshot of a computer Description automatically generated with medium confidence

### 10. Update Status of existing Order by Order ID: A screenshot of a computer Description automatically generated with medium confidence

### 11. Delete existing Order by Order ID: A screenshot of a computer Description automatically generated with medium confidence

### 12. Get all Customers: A screenshot of a computer Description automatically generated with medium confidence

### 13. Get Customer by Customer ID: A screenshot of a computer Description automatically generated with medium confidence

### 14. Create new Customer: A screenshot of a computer Description automatically generated with medium confidence

### 15. Update existing Customer by Customer ID : A screenshot of a computer Description automatically generated with medium confidence

### 16. Delete existing Customer by Customer ID: A screenshot of a computer Description automatically generated with medium confidence

# Phase 2 REST Services

The following forms have been created to consume GET and POST methods.

## HTML Forms

### 1. Search Product by Product ID

**HTTP Request: GET**

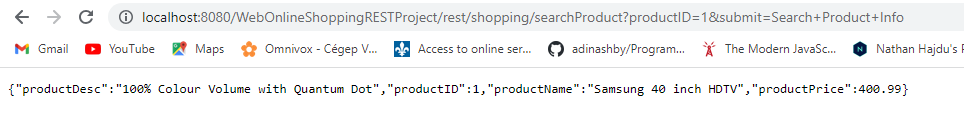
Allows the user to search for a specific product by Product ID:  
A screenshot of a computer

Description automatically generated with medium confidence

**Data Inputs:**

**Product ID:** The ID of the product, provided as input through the form field productId.

**Data Output:**

The output returns the specific product details from the list of products:  


### 2. Add new Product

**HTTP Request: POST**

Allows the user to create a new product and add it to the list of existing products:  
A screenshot of a computer

Description automatically generated with medium confidence

**Data Inputs:**

**Product ID:** The ID of the product, provided as input through the form field productId.

**Product Name:** The name of the product, provided as input through the form field productName.

**Product Price:** The price of the product, provided as input through the form field productPrice.

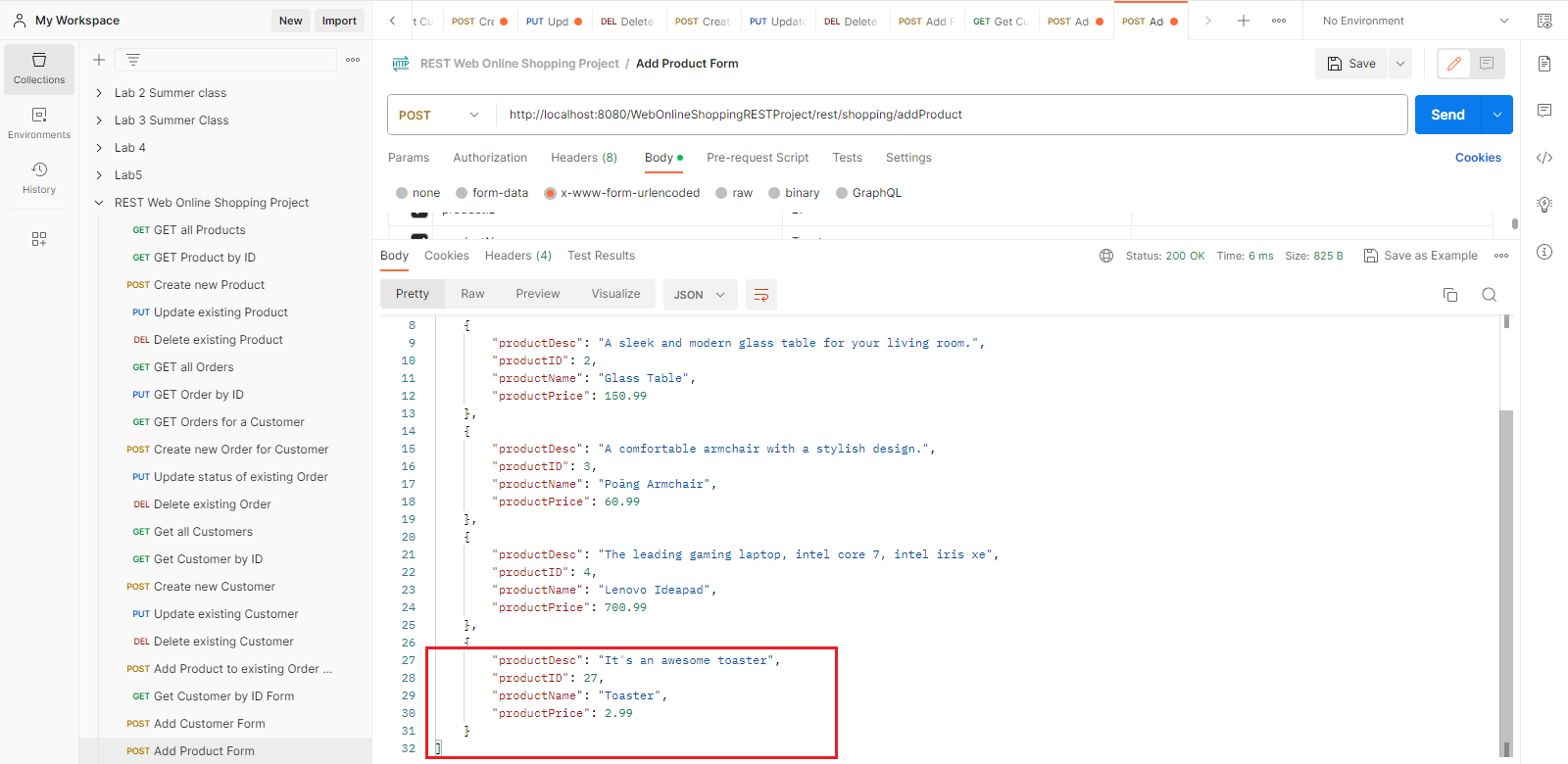
**Product Description:** The description of the product, provided as input through the form field productDesc.

**Data Output:**

**List of Products:** After the new product information is added, a list of products is returned as the output. This list includes all the existing products, including the newly added one:

### 

**Postman:**



### 3. Search Order by Order ID

**HTTP Request: GET**

Allows the user to search for a specific Order by Order ID:

A screenshot of a computer

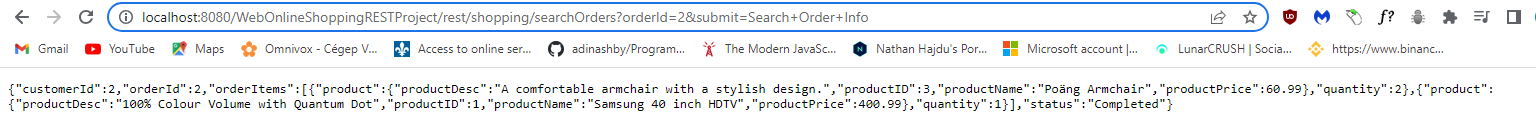
Description automatically generated with medium confidence

**Data Input:**

**Order ID:** The ID of the order, provided as input through the form field orderId.

**Data Output:**

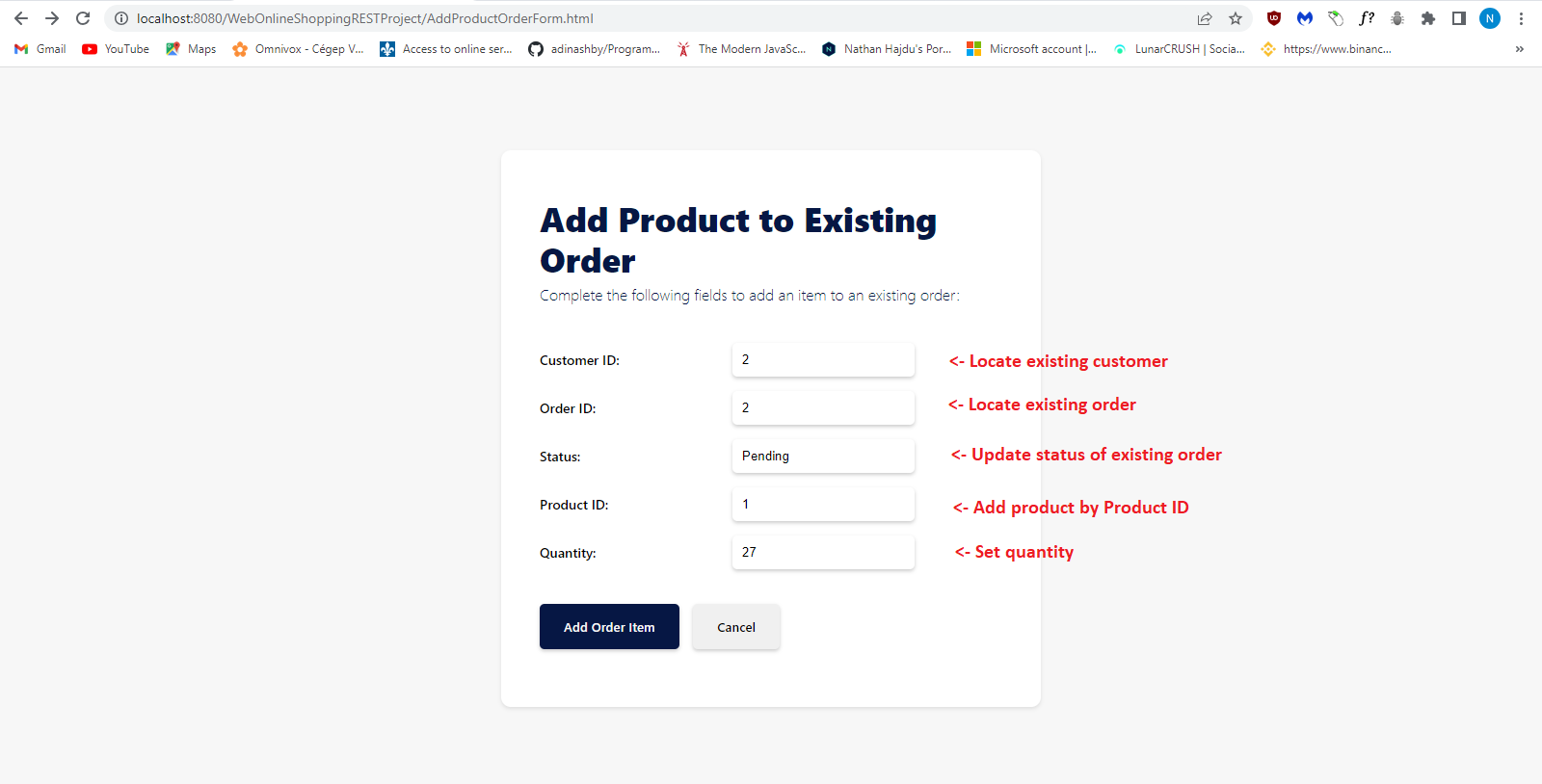
The output returns the specific order details from the list of orders, including the quantity of each product in the order:



### 4. Add new Product to existing Order

**HTTP Request: POST**

Allows the user to add a new product and set quantity to an existing order, as well as update the status of the order (i.e. from “completed” to “pending”):



**Data Inputs:**

**Customer ID:** The ID of the customer, provided as input through the form field customerId.

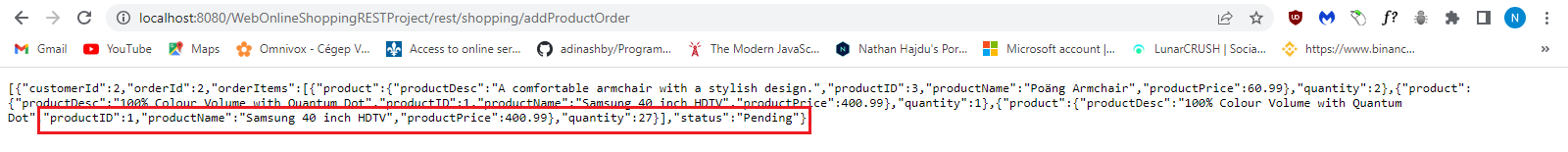
**Order ID:** The ID of the order, provided as input through the form field orderId.

**Status:** The status of the order, provided as input through the form field status.

**Product ID:** The ID of the product to add to the order, provided as input through the form field productId.

**Quantity:** The quantity of the product to add to the order, provided as input through the form field quantity.

**Data Output:**

**List of Orders:** If the customer, order, and product exist, the updated list of orders for the customer is returned as the output:  


**BEFORE:  
A screenshot of a computer

Description automatically generated with medium confidence  
  
AFTER:  
A screenshot of a computer

Description automatically generated with medium confidence**

### 5. Search Customer by Customer ID

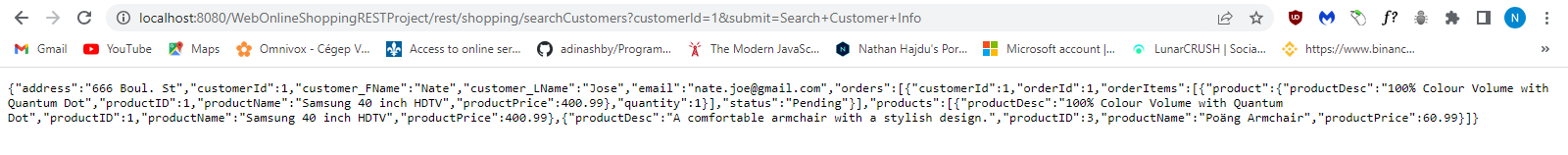
**HTTP Request: GET**Allows the user to search for a specific customer by Customer ID:  
A screenshot of a computer

Description automatically generated with medium confidence

**Data Inputs:**

**Customer ID:** An ID for the customer, provided as input through the form field customerId.

**Data Output:**

The output returns all the details of the customer, including any orders belonging to the customer and the products associated with the customer’s order:  


A screenshot of a computer

Description automatically generated with medium confidence

### 6. Add new Customer

**HTTP REQUEST: POST**

Allows the user to create a new customer and add them to the list of customers:

**A screenshot of a computer

Description automatically generated**

**Data Inputs:**

**Customer ID:** An ID for the customer, provided as input through the form field customerId.

**First Name:** The first name of the customer, provided as input through the form field customer\_FName.

**Last Name:** The last name of the customer, provided as input through the form field customer\_LName.

**Email:** The email address of the customer, provided as input through the form field email.

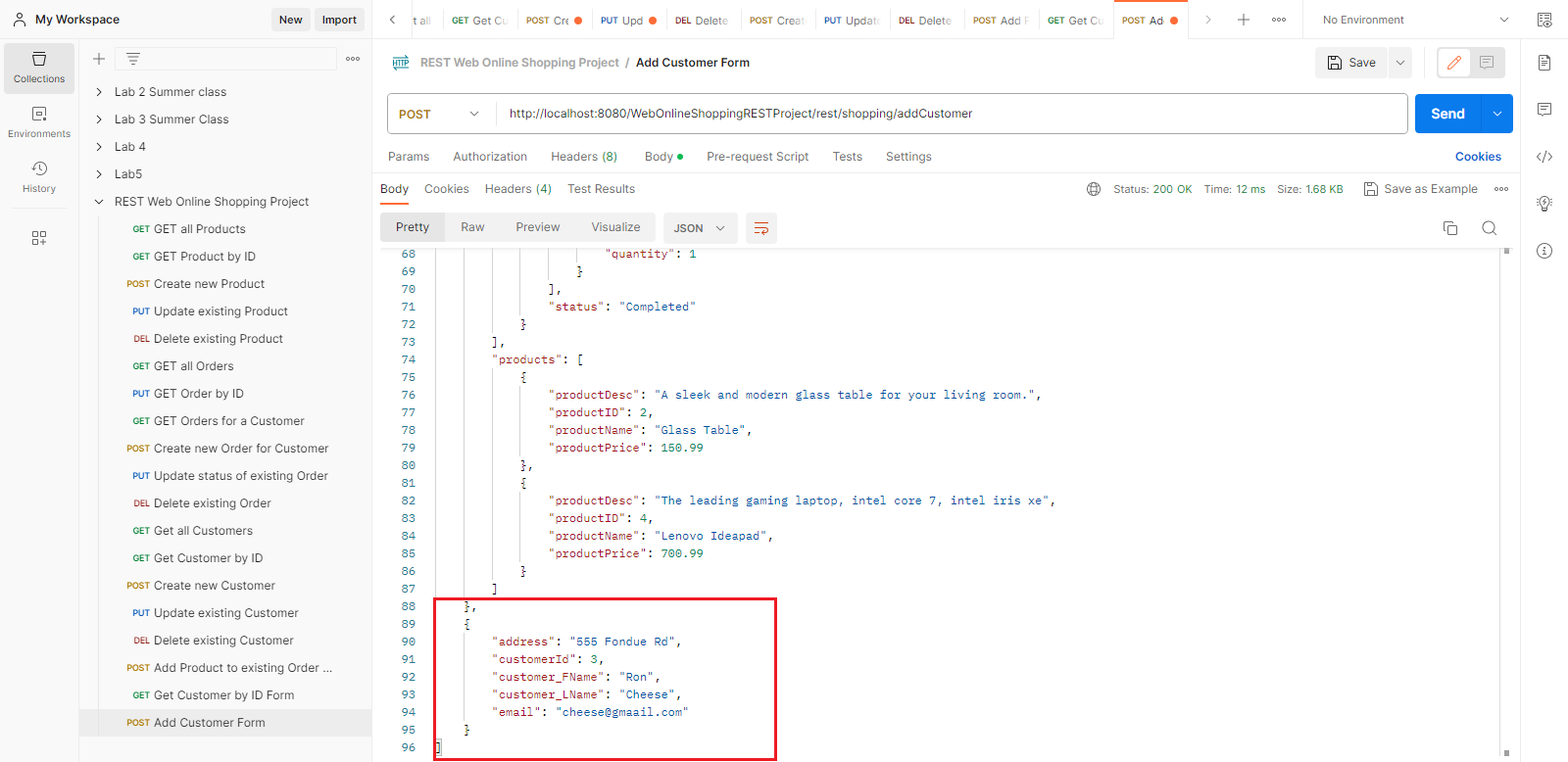
**Address:** The address of the customer, provided as input through the form field address.

**Data Output:**

The output returns the newly created customer in the list of existing customers:

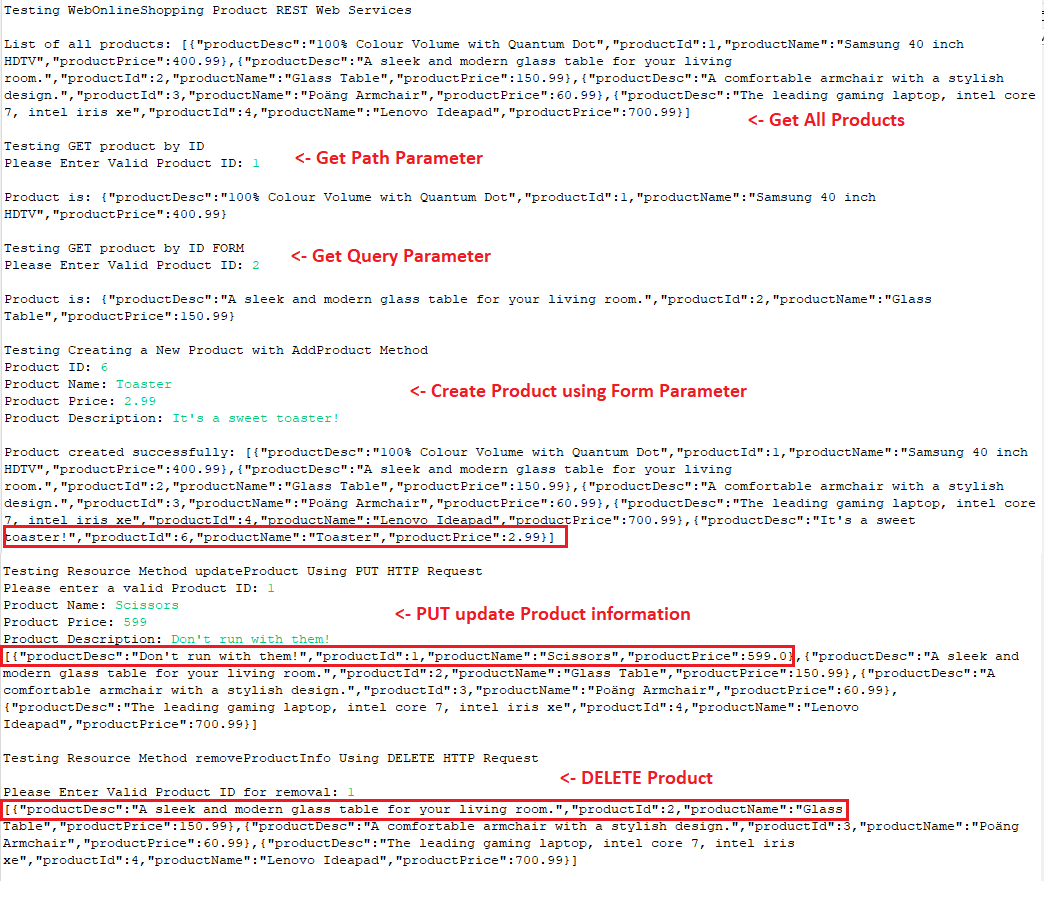
**A screenshot of a computer

Description automatically generated with medium confidence**

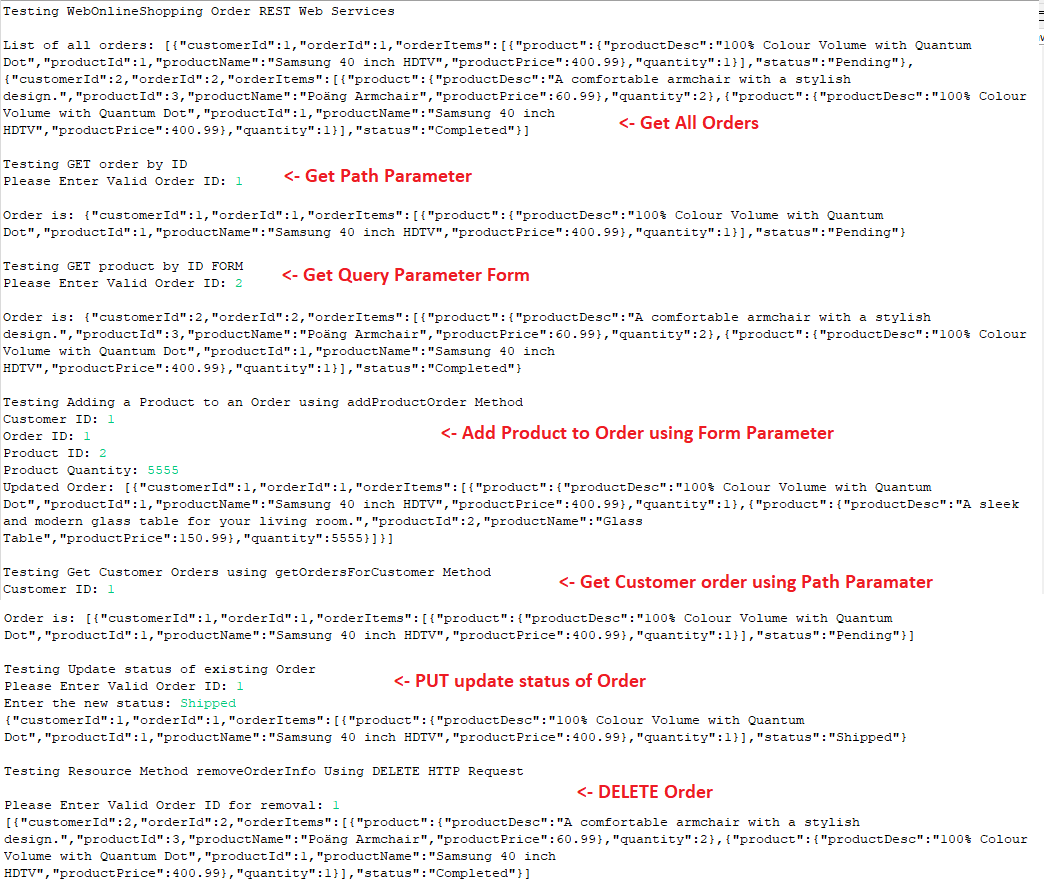
****

## Client WebOnlineShopping

The client is comprised of three classes that test and consume the server-side web services listed previously. The three classes are:  
  
1. ClientProductREST



### 2. ClientOrderREST



### 3. ClientCustomerREST



# Phase 3 Soap Services

The purpose of this phase is to provide further utility through implementing additional SOAP web services.

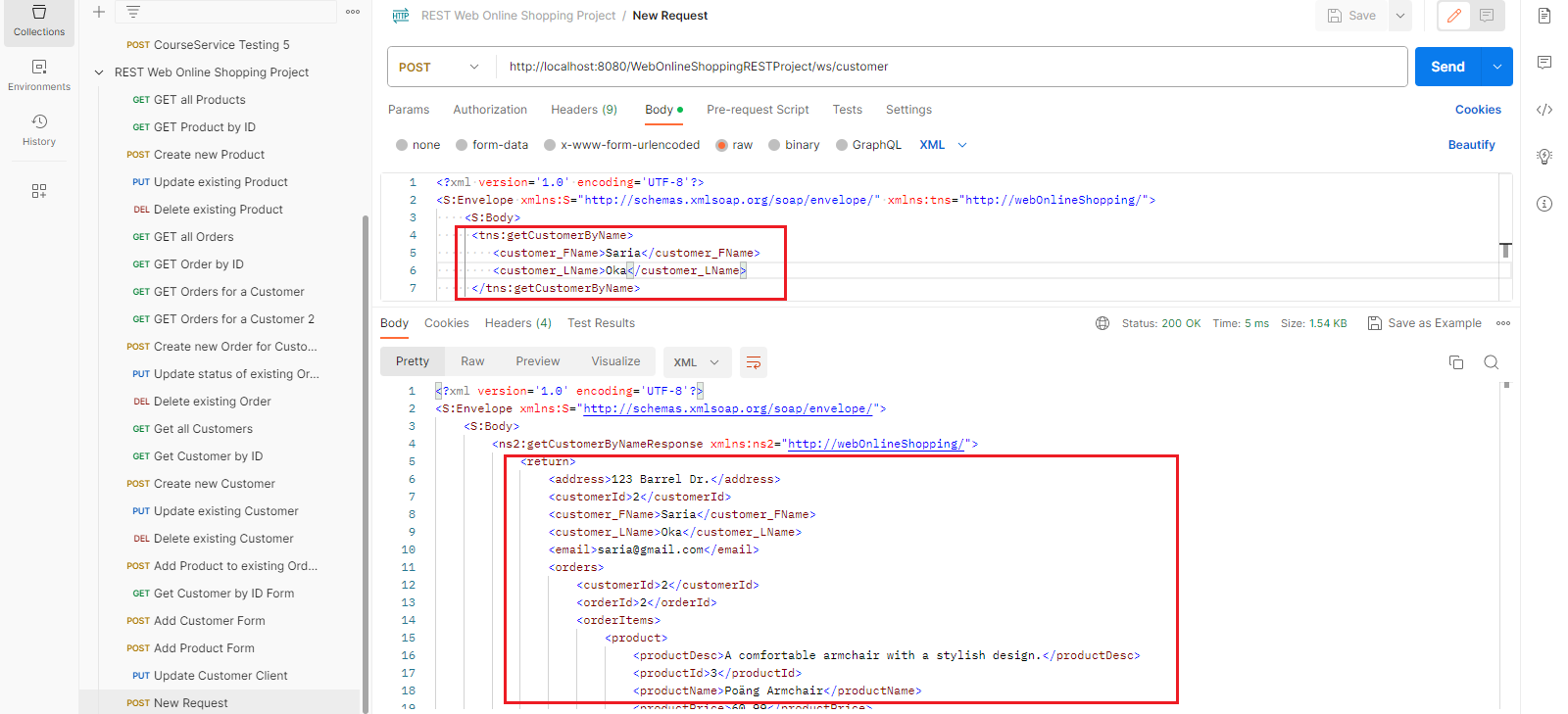
To this end, three new classes have been developed that introduce new functionality to the project:

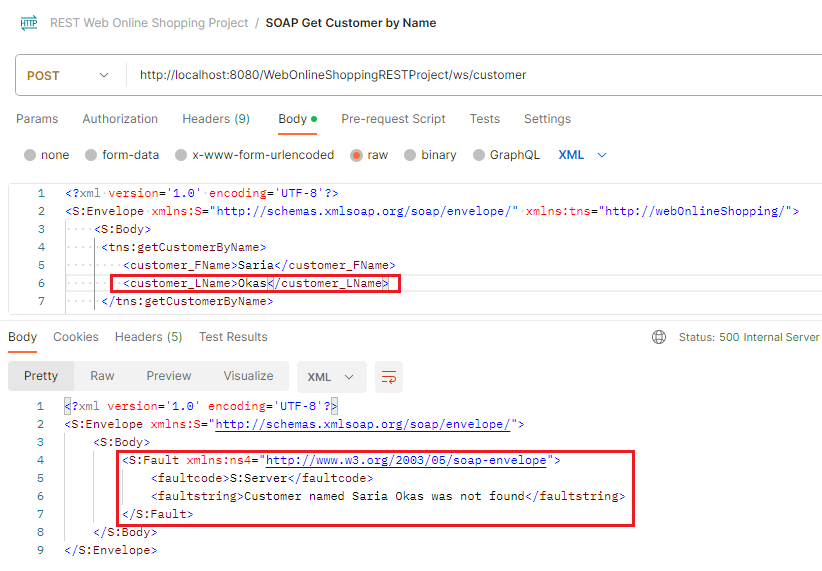
## 1. WebCustomerService

### a) getCustomerByName

This method allows the user to search for a customer by submitting an xml request using the customer’s first and last name as web parameters. If the customer details have been entered correctly and a match is found, the user will be able to see the full details of the customer:  
  
A computer code with text

Description automatically generated



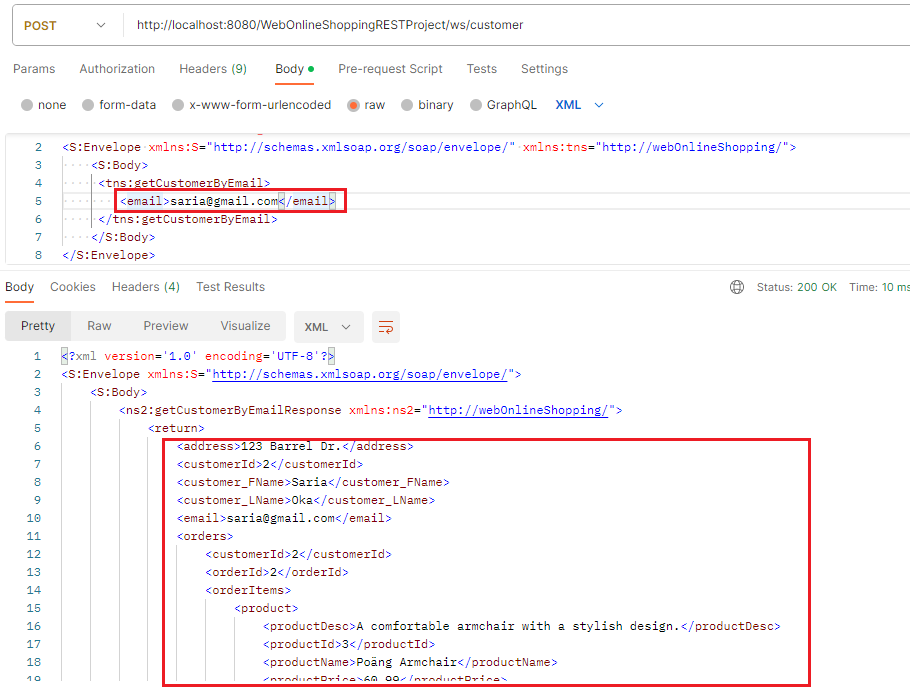
This method is not case-sensitive. However, if the user enters the wrong name, the response alerts the user that the customer was not found:  
  


### b) searchCustomerByEmail

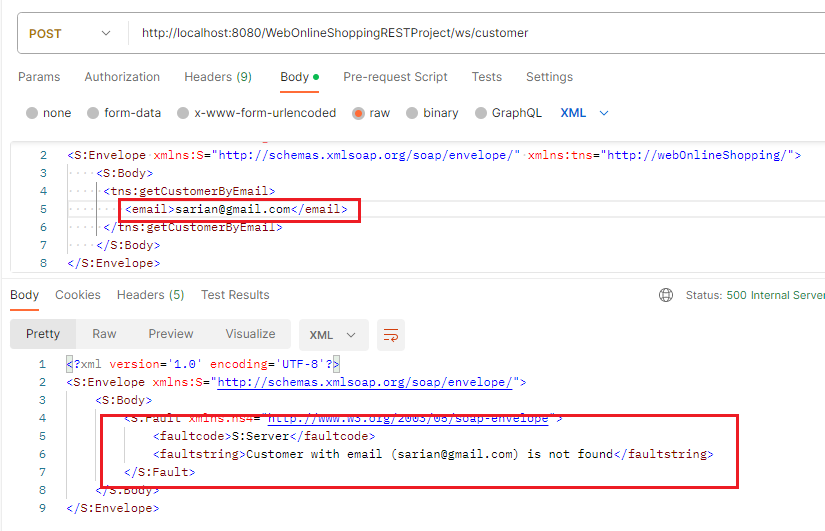
This method allows the user to search for a customer by email. If the customer details have been entered correctly and a match is found, the user will be able to see the full details of the customer.

A computer screen shot of text

Description automatically generated



This method is not case-sensitive. However, if the user enters the wrong name, the response alerts the user that the customer was not found:

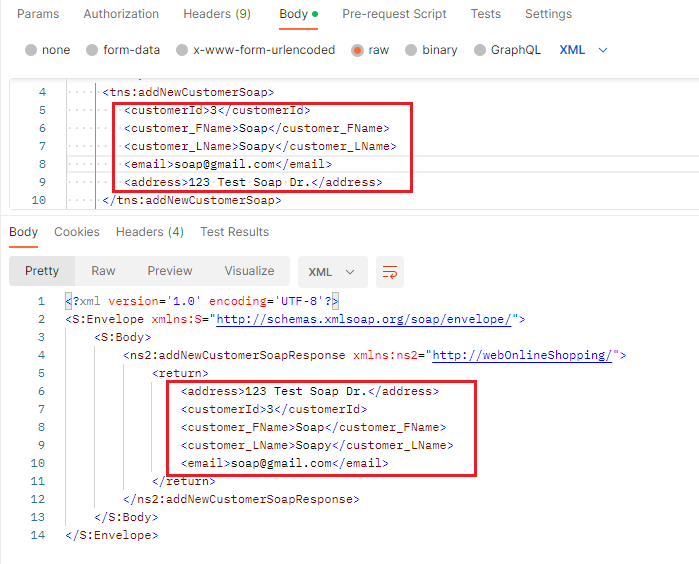


### c) addNewCustomerSoap

This method allows the user to create a new customer entry and add them to the list of customers.

A computer screen shot of a program code

Description automatically generated



This method is not case-sensitive. However, if the user enters a customer ID or email address that already exists in the list of customers, they will receive an error:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

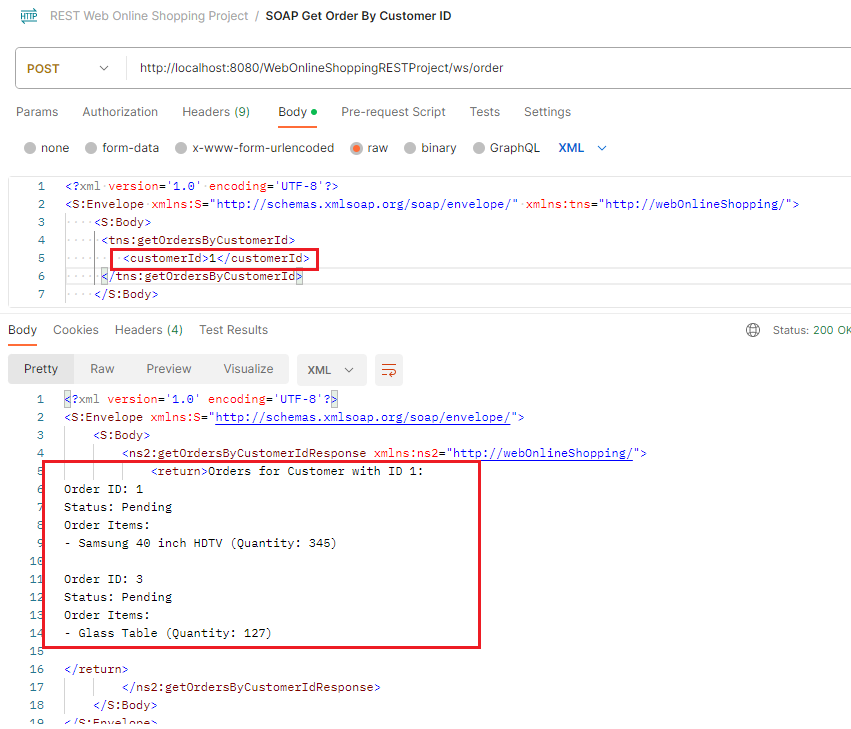
## 2. WebOrderService

### a) getOrderByCustomerId

This method allows the user to return all the orders placed by a customer based on their ID.

A screenshot of a computer code

Description automatically generated



If the ID entered does not match an existing record, the user will receive a warning:  
  
A screenshot of a computer program

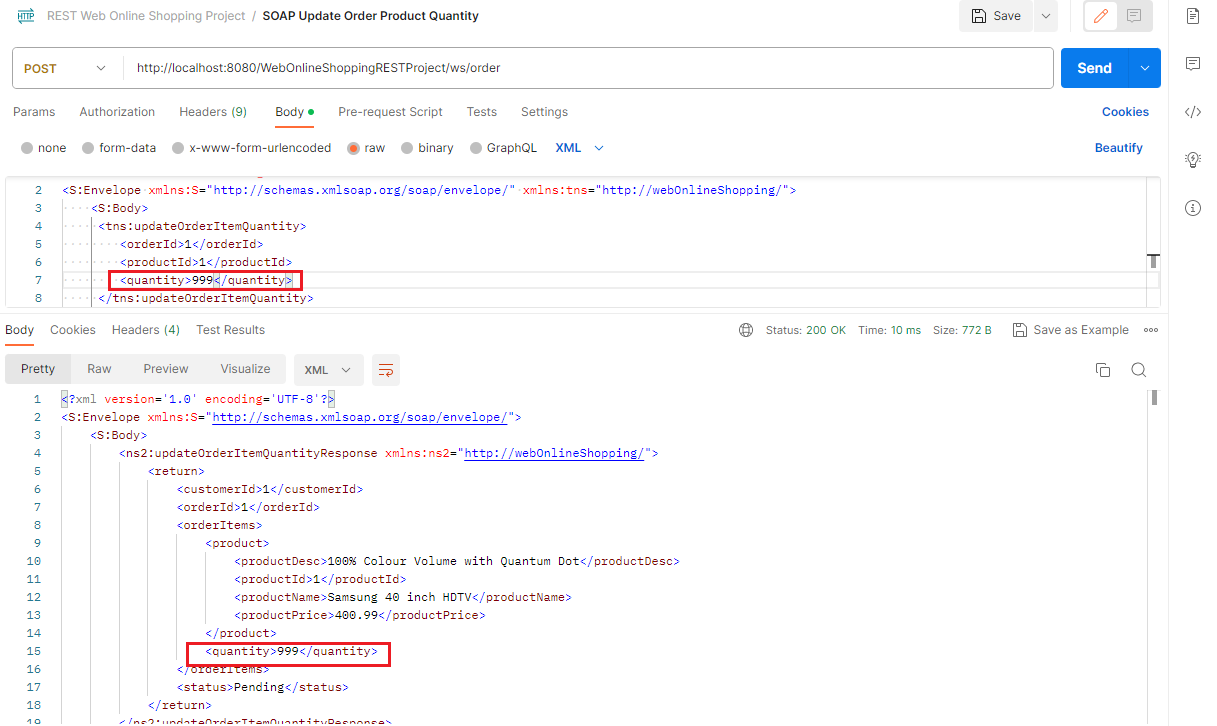
Description automatically generated

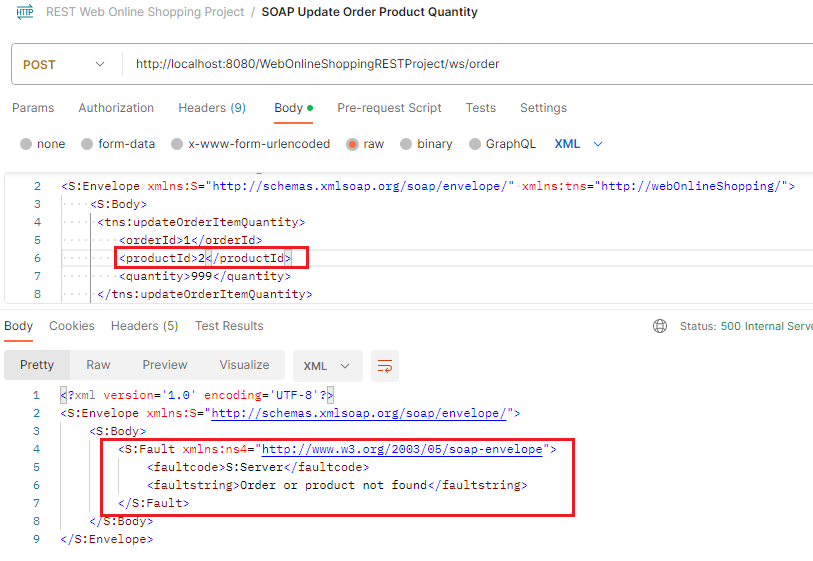
### b) updateOrderItemQuantity

This method allows the user to search for a product in an existing order and update the quantity of that product.

A computer screen shot of a code

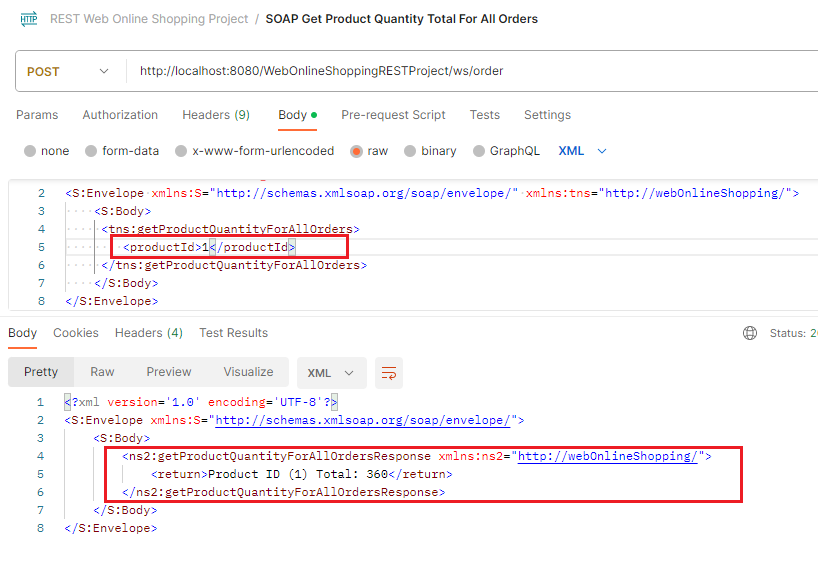
Description automatically generated



If the user enters an order ID that doesn’t exist, or if they entered a correct order ID but the product is not found in the order (based on product ID), the user will receive a warning:  
  


### c) getProductQuantityForAllOrders

This method allows the user to discover the total number of a specific product ordered across all orders.



A computer screen shot of a program

Description automatically generated

As shown in the screenshot above, order 1 has a quantity of 345 for product 1, while order 2 has 15.   
345 + 15 = 360.

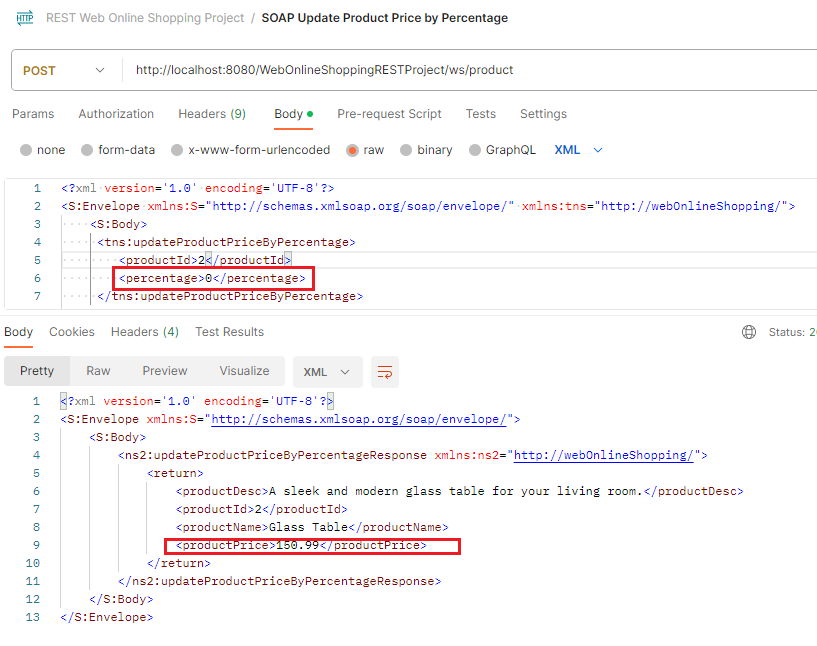
## 3. WebProductService

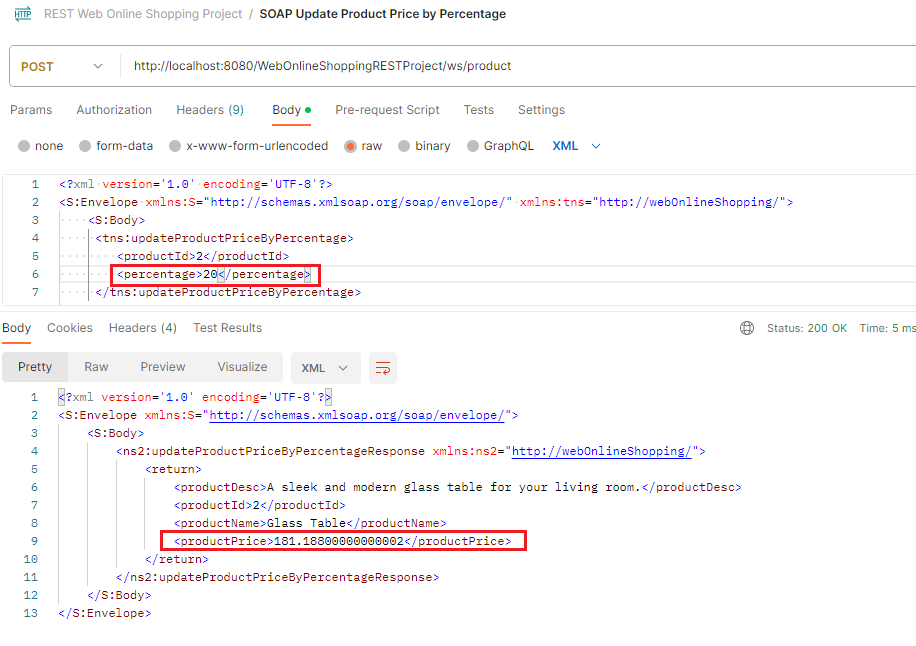
### a) updateProductPriceByPercentage

This method allows the user to raise or reduce the price of a Product by a percentage.

A screen shot of text

Description automatically generated

For example, Product 2 is originally listed at $150.99:  
  


By changing the percentage to 20, we effectively increase the price of the item by 20%:  
  
  
**NOTE:** After raising or reducing the price, the user can reset the price of the Product to its default value by making a request with 0 as the percentage. Currently Does not Work In Postman.

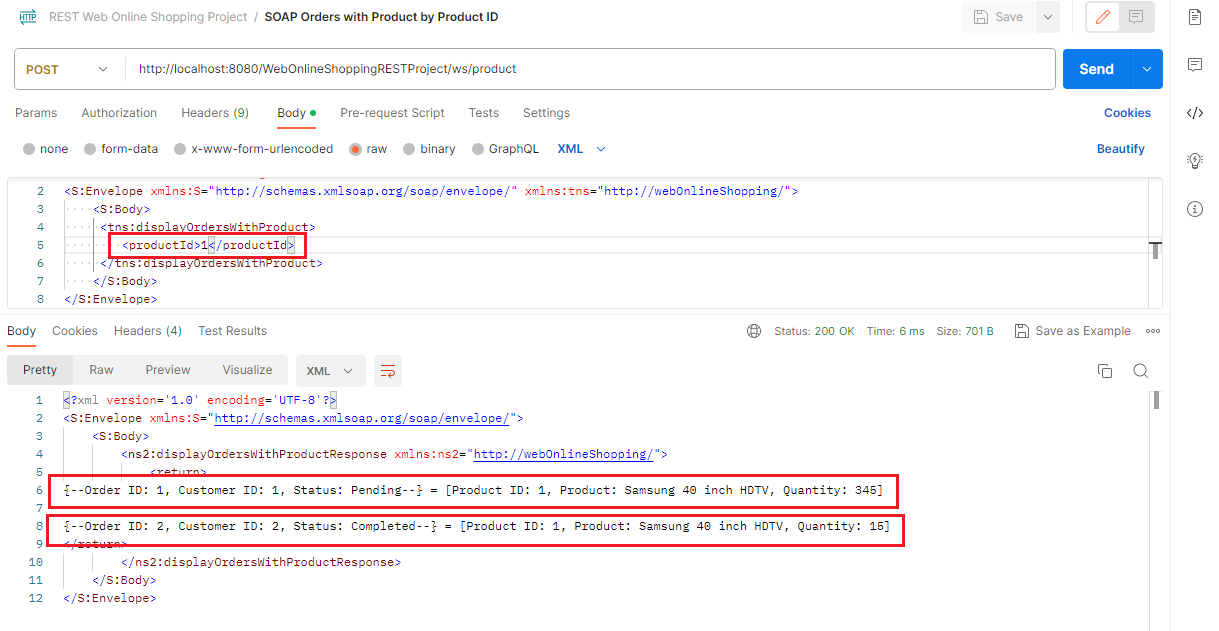
### b) displayOrdersWithProduct

This method allows the user to search through all Orders that contain a specific Product ID and outputs the details of the Order and Product searched for (i.e. Product ID, Product Name, Quantity).

This is useful if the user wants to check the total quantity of a product in orders against inventory.

A screenshot of a computer code

Description automatically generated



### c) getProductsByPriceRange

This method allows the user to search for all products within a specific price range (min/max).  
  
A screen shot of text

Description automatically generated

The program currently has 4 products ranging in price between $1-800:  
  
A screenshot of a computer

Description automatically generated

If the users narrows the range down to 300:  
  
A screenshot of a web page

Description automatically generated

If no items fall within the price range specified, the user will receive a warning:  
  
A screenshot of a web page

Description automatically generated