27-Project using Spring | Update and Delete

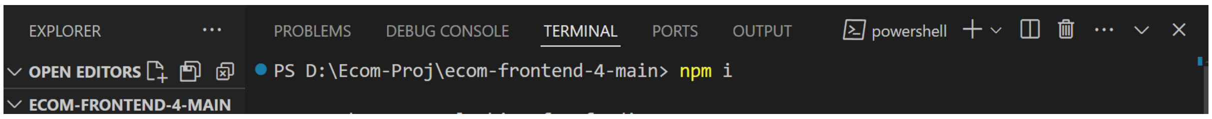
Update and Delete Operations in Our Project

Running the application

To get started, open Visual Studio Code (VS Code) and follow these steps:

1. **Install dependencies:**

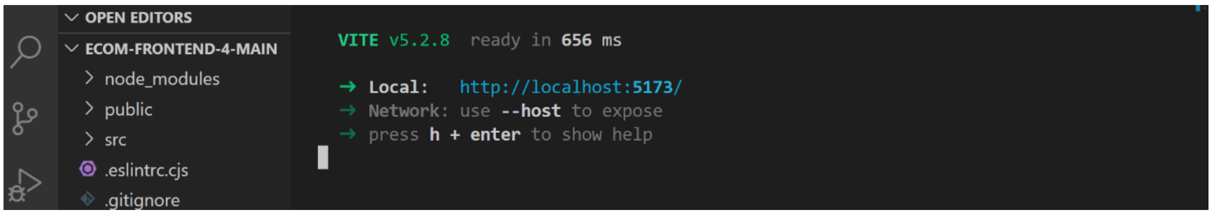
   npm install



```
PS D:\Ecom-Proj\ecom-frontend-4-main> npm i
```

2. **Run the application:**

   npm run dev

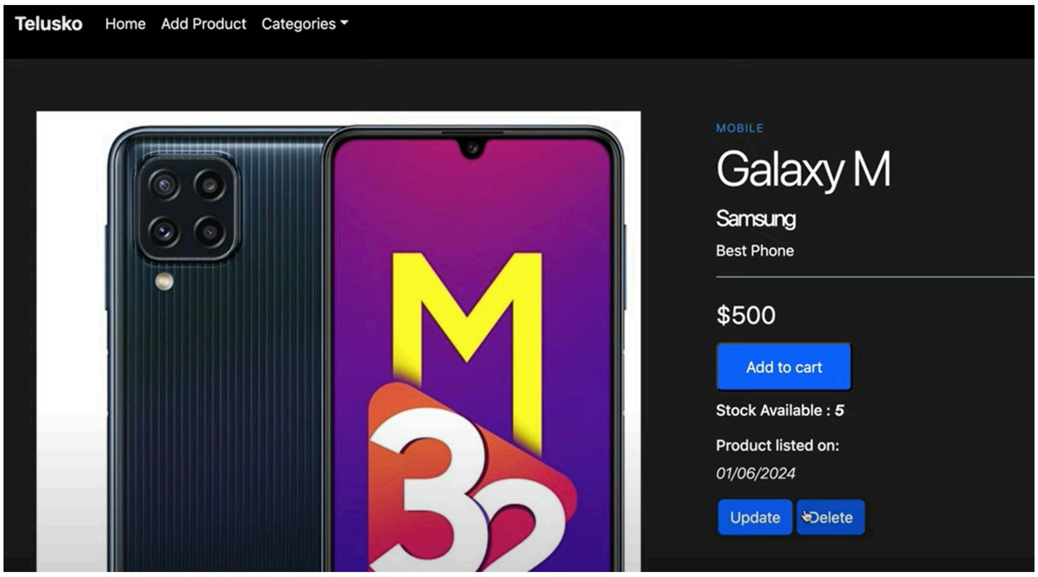


```
  VITE v5.2.8  ready in 656 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

The application will run on https://localhost:5173/.

Updated UI

After loading the application, you should see the UI displaying two different products with images. Clicking on any product will show the product description and other details. Two new buttons have been added: one for updating and one for deleting the product.



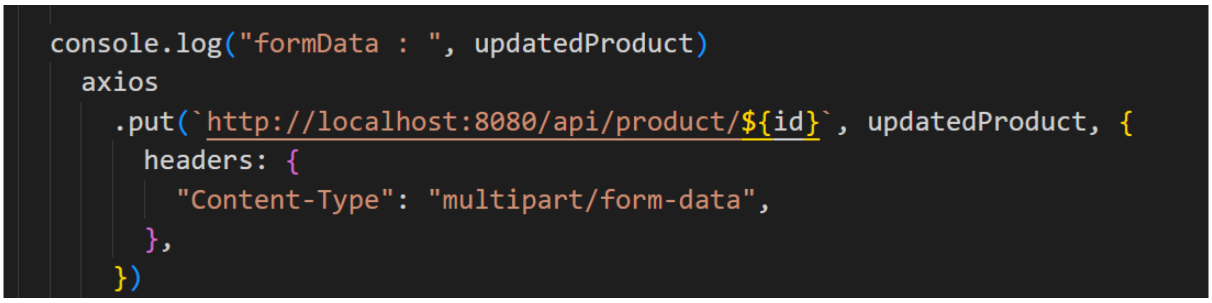
Till now, we have not made any changes to the backend code; only some modifications have been done on the frontend side.

Update Functionality

Clicking on the update button opens an update form with existing data that needs to be changed if required. After submitting the update form, a pop-up displays that our product failed to update. This is because we haven't updated our backend for the corresponding URL, and we don't have any logic to handle this API right now on the backend part.

Before heading towards the backend coding, In UI let's see what URL has been created for the update and delete functionalities. In the update method, a PUT mapping request is used along with the URL:

localhost:8080/api/product/{id}

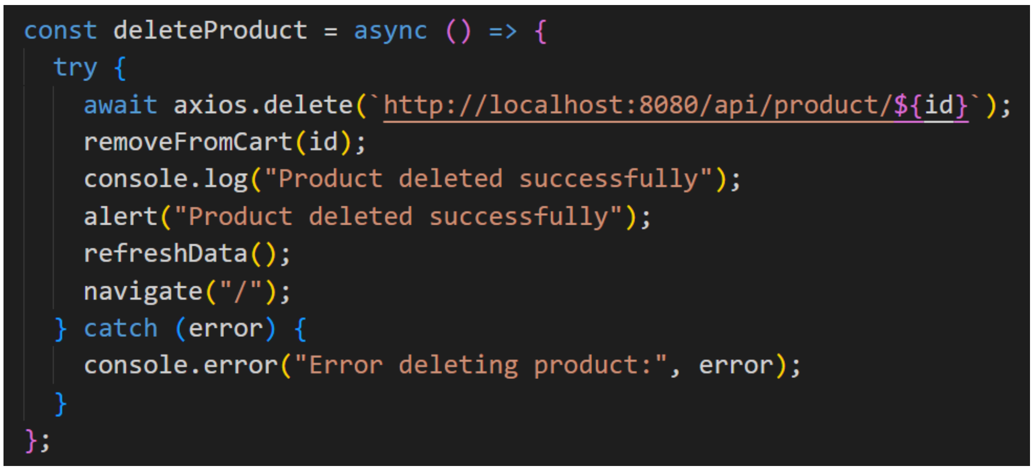


```
console.log("formData : ", updatedProduct)
  axios
    .put(`http://localhost:8080/api/product/${id}`, updatedProduct, {
      headers: {
        "Content-Type": "multipart/form-data",
      },
    })
```

Delete Functionality

For the delete method, the URL request mapping is:

localhost:8080/api/product/{id}



```javascript
const deleteProduct = async () => {
  try {
    await axios.delete(`http://localhost:8080/api/product/${id}`);
    removeFromCart(id);
    console.log("Product deleted successfully");
    alert("Product deleted successfully");
    refreshData();
    navigate("/");
  } catch (error) {
    console.error("Error deleting product:", error);
  }
};
```

For the delete mapping request, only the ID is sent, and the product is deleted based on the provided ID.

Backend Preparation

What data are we going to send through the UI is mainly three things:

- Product
- Product ID
- Image as a multipart file

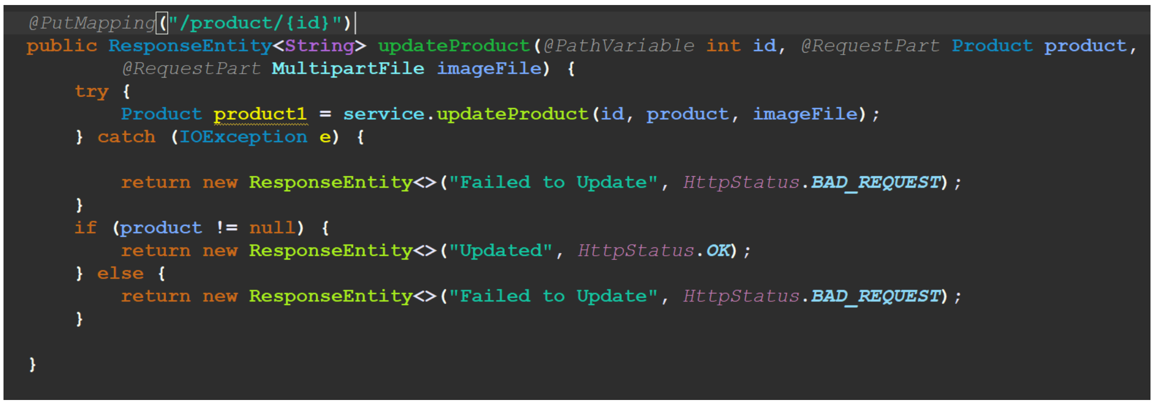
For the delete request, we are just sending the product ID along with the method, and later our product is deleted.

Update Request

In the Controller:

- **Method:** updateProduct
- **Return Type:** ResponseEntity
- **Annotation:** @PutMapping("/product/{id}")
- **Arguments:**
  - Product
  - Product ID
  - Multipart file as image

Implementation:

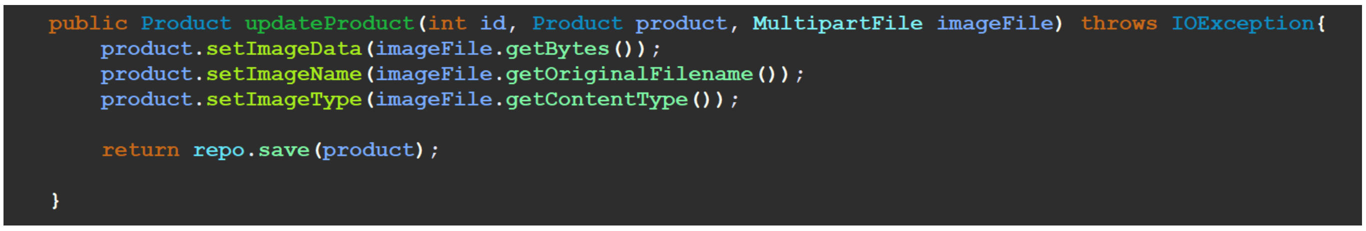


```
@PutMapping("/product/{id}")
public ResponseEntity<String> updateProduct(@PathVariable int id, @RequestPart Product product,
        @RequestPart MultipartFile imageFile) {
    try {
        Product product1 = service.updateProduct(id, product, imageFile);
    } catch (IOException e) {

        return new ResponseEntity<>("Failed to Update", HttpStatus.BAD_REQUEST);
    }
    if (product != null) {
        return new ResponseEntity<>("Updated", HttpStatus.OK);
    } else {
        return new ResponseEntity<>("Failed to Update", HttpStatus.BAD_REQUEST);
    }

}
```

In the Service:

- **Method:** updateProduct
- **Return Type:** Product
- **Parameters:** id, product, imageFile

Implementation:



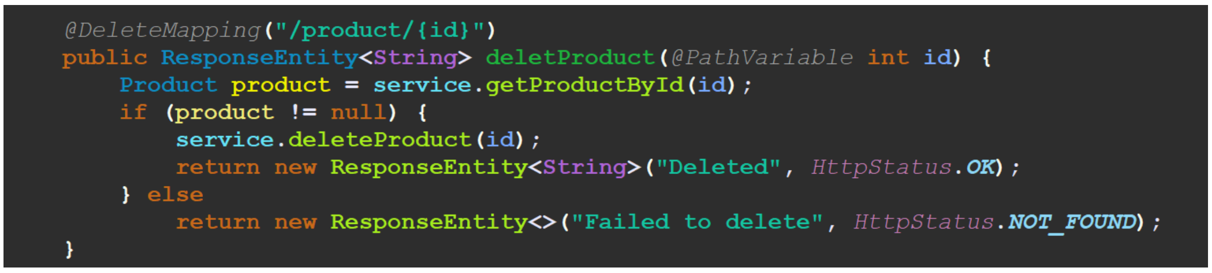
```
public Product updateProduct(int id, Product product, MultipartFile imageFile) throws IOException{
    product.setImageData(imageFile.getBytes());
    product.setImageName(imageFile.getOriginalFilename());
    product.setImageType(imageFile.getContentType());

    return repo.save(product);

}
```

Delete Request

In the Controller:

- **Method:** deleteProduct
- **Return Type:** ResponseEntity<String>
- **Annotation:** @DeleteMapping("/product/{id}")
- **Argument:** @PathVariable Long id

Implementation:

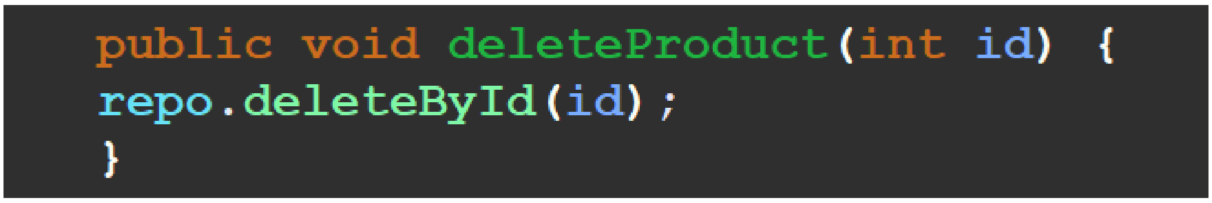


```
@DeleteMapping("/product/{id}")
public ResponseEntity<String> deletProduct(@PathVariable int id) {
    Product product = service.getProductById(id);
    if (product != null) {
        service.deleteProduct(id);
        return new ResponseEntity<String>("Deleted", HttpStatus.OK);
    } else
        return new ResponseEntity<>("Failed to delete", HttpStatus.NOT_FOUND);
}
```

In the service:

- **Method:** deleteProduct
- **Return Type:** void
- **Parameter:** Long id

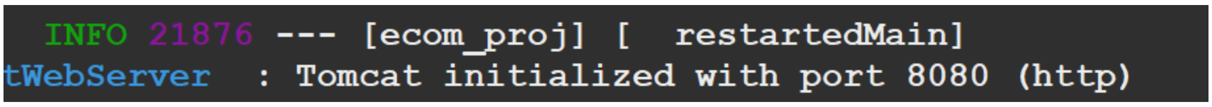
Implementation:



```
public void deleteProduct(int id) {
repo.deleteById(id);
}
```

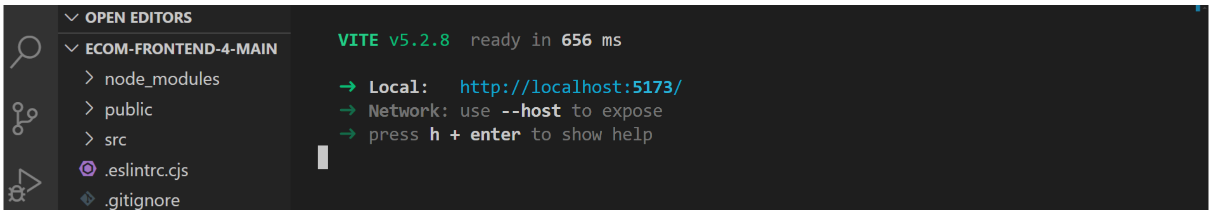
Restart the application.

1. **Backend Server:** Start the Tomcat server on port 8080.



```
 INFO 21876 --- [ecom_proj] [  restartedMain]
tWebServer   : Tomcat initialized with port 8080 (http)
```

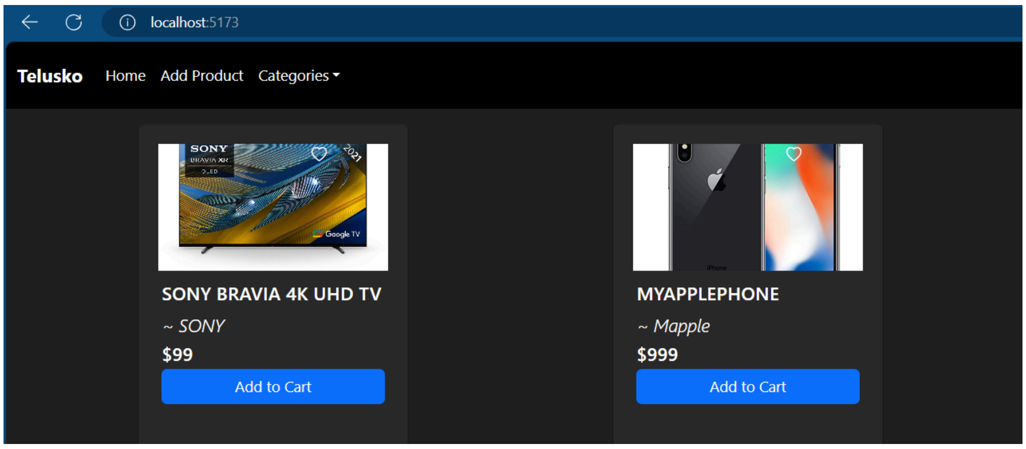
2. **Frontend Application:** Load the application on port 5173.



```
VITE v5.2.8  ready in 656 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Verifying the Updates

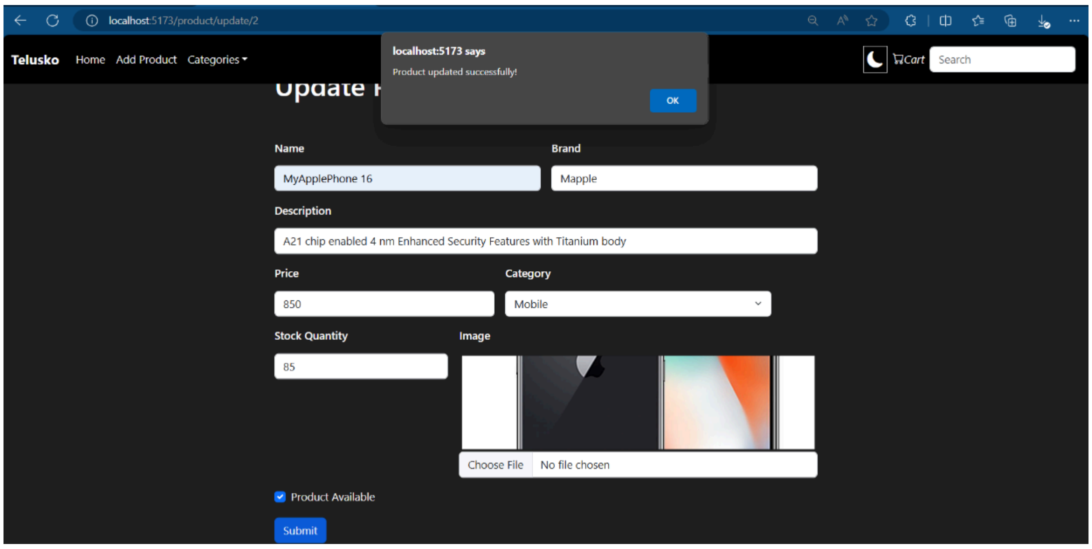
- **Home Page:**
  - o Check that product details, including images, are displayed correctly.



SONY BRAVIA 4K UHD TV
~ SONY
$99

MYAPPLEPHONE
~ Mapple
$999

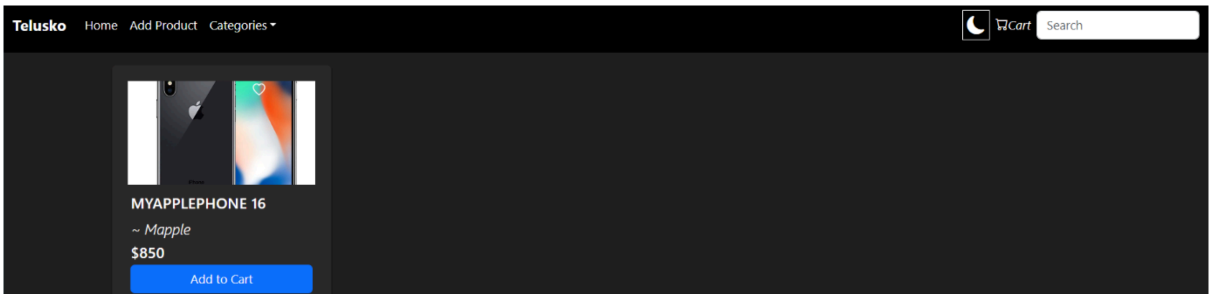
- **Update Product:**
  - Click on the "Update" button to open the update form.
  - Modify the necessary fields and submit the form.
  - Verify that a pop-up message confirms the successful update.
  - Check the home page to see the updated product information.



localhost:5173 says
Product updated successfully!

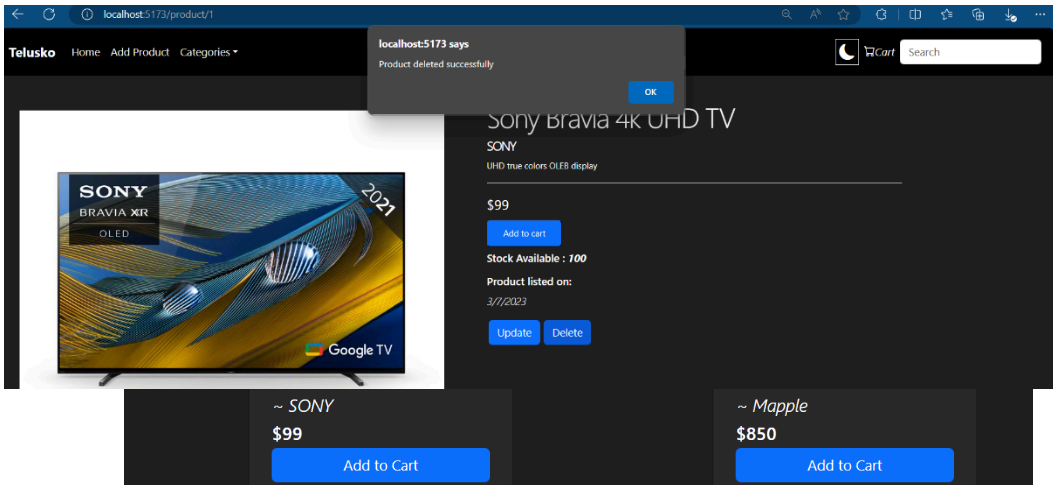
Update P

Name: MyApplePhone 16
Brand: Mapple
Description: A21 chip enabled 4 nm Enhanced Security Features with Titanium body
Price: 850
Category: Mobile
Stock Quantity: 85
Image: No file chosen
✓ Product Available



MYAPPLEPHONE 16
~ Mapple
$850

- **Delete Product:**
  - Click on the "Delete" button to remove a product.
  - Verify that a pop-up message confirms the successful deletion.
  - Check the home page to ensure the product has been removed.



localhost:5173 says
Product deleted successfully

Sony Bravia 4k UHD TV
SONY
UHD true colors OLED display
$99
Stock Available : *100*
Product listed on:
3/7/2023

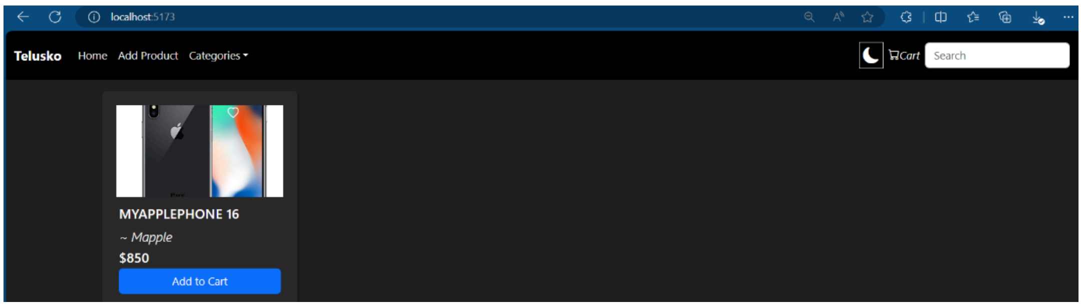
~ SONY
$99

~ Mapple
$850

Conclusion

In this chapter, we implemented two methods: Update and Delete.

- **Update:** requires product, product ID, and image as a multipart file.
- **Delete:** Only requires the product ID.



Next Steps

In the next chapter, we will work on the search feature, allowing users to search for products using the search bar in the navigation menu.

Stay curious and keep coding! ☺ 👾