# E-commerce Microservice Application

## 1. Introduction

This document provides a comprehensive overview of the microservices architecture used in the eCommerce platform. The system is composed of three primary microservices: **CartService**, **CheckoutService**, and **ProductService**. Each microservice is responsible for a specific domain within the application.

### 1.1 Objective

The objective of this project is to create a eCommerce platform where users can browse products, add them to their cart, and complete purchases. The microservices are designed to handle these operations independently while communicating effectively with each other.

## 2. System Architecture

### 2.1 Overview

The system is built on a microservices architecture, where each service is independently deployed and managed. Communication between services is achieved through RESTful APIs, ensuring loose coupling and allowing for the independent scaling of each service.

### 2.2 Technologies Used

* **Programming Language**: Java
* **Framework**: Spring Boot
* **Database**: MySQL
* **Containerization**: Docker

## 3. Microservices Overview

### 3.1 CartService

##### 3.1.1 Description

The CartService is responsible for managing the shopping cart for users. It handles adding, updating, retrieving, and deleting items in the user's cart.

* **Add to Cart**: Users can add items to their shopping cart by selecting a product and specifying the quantity they wish to purchase.
* **View Cart**: Users can view all the items currently in their cart, including product details and quantities.
* **Update Cart Item**: Users can modify the quantity of items in their cart.
* **Delete Cart Item**: Users can remove items from their cart.
* **Retrieve Cart Item by ID**: Users can view details of a specific cart item by its unique ID.

##### 3.1.2 Endpoints

* GET /api : Retrieves all items in the user's cart.

A screenshot of a computer

Description automatically generated

* GET /api/{id}: Retrieves a specific cart item by ID.

A screenshot of a computer

Description automatically generated

* POST /api/cart: Adds a new item to the cart.

A screenshot of a computer

Description automatically generated

* PUT /api/{id}: Updates the quantity of a cart item.

A screenshot of a computer

Description automatically generated

* DELETE /api/cart/{id}: Removes an item from the cart.

A screenshot of a computer

Description automatically generated

### 3.2 CheckoutService

##### 3.2.1 Description

The CheckoutService handles the checkout process, managing the user's order information, payment details, and order processing.

* **Submit Checkout Form:** Users can enter and submit their checkout information, including personal and payment details, to complete the purchase process.

##### 3.2.2 Endpoints

* POST /api/add : Submit checkout information and save it to the database.

A screenshot of a computer

Description automatically generated

### 3.3 ProductService

##### 3.3.1 Description

The ProductService is responsible for managing the product catalog, including adding, updating, retrieving, and deleting products.

* **Add New Product**: Admin users can add new products to the catalog, including relevant details like name, description, price, and stock.
* **View All Products**: Users can browse and view a list of all available products.
* **View Product by ID**: Users can view detailed information about a specific product by its unique identifier.
* **Update Product**: Admin users can update existing product information.
* **Delete Product**: Admin users can remove products from the catalog.

##### 3.2.2 Endpoints

* GET /api/list: Retrieve a list of all products available in the catalog.

A screenshot of a computer

Description automatically generated

* GET /api/{id}: Retrieve details of a specific product by its ID.

A screenshot of a computer

Description automatically generated

* PUT /api/{id}: Update the details of an existing product.

A screenshot of a computer

Description automatically generated

* POST /api/post: Add a new product to the catalog.

A screenshot of a computer

Description automatically generated

* DELETE /api/{id}: Remove a product from the catalog.

A screenshot of a computer

Description automatically generated

Image Service statred for product service  
A screenshot of a computer

Description automatically generated

* Image service for cart service

A screenshot of a computer

Description automatically generated

* Image service for checkout

A screenshot of a computer

Description automatically generated

* Completely running the docker-compose file

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Front end image running in docker

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Image running on 3000

A screenshot of a computer

Description automatically generated

* Logs os image running

A screenshot of a computer

Description automatically generated

# Frontend

## 1. Key Components

* **AddProductForm**: Form to add new products.
* **Cart**: Cart management, including item removal and quantity adjustment.
* **CheckoutForm**: Form for processing payment and handling checkout.
* **Footer**: Displays footer information.
* **Navbar**: Navigation bar for routing.
* **Products**: Displays a list of products and allows users to add them to the cart.
* **App**: Main application component managing routing and context.

## 2. Component Descriptions

### 2.1 AddProductForm

A screenshot of a computer

Description automatically generated

**Description**: Allows users to add new products to the inventory.

**State Variables**:

* name: Product name.
* description: Product description.
* price: Product price.
* stock: Available stock.
* image: Product image.

**Functionality**:

* Handles form submission.
* Sends product data to the server using axios.
* Redirects to the products page upon successful addition.

### 2.2 Cart

A screenshot of a computer

Description automatically generated

* **Description**: Manages the shopping cart.
* **State Variables**:
  + cart: Array of items in the cart.
* **Functions**:
  + handleCheckout: Navigates to the checkout page.
  + handleRemove: Removes an item from the cart and server.
  + handleQuantityChange: Updates the quantity of items in the cart.

### 2.3 CheckoutForm

A screenshot of a computer screen

Description automatically generated

* **Description**: Processes checkout and payment.
* **State Variables**:
  + name: Customer's name.
  + address: Delivery address.
  + cardNumber: Credit card number.
  + expirationDate: Card expiration date.
  + cvv: Card CVV.
  + loading: Loading state for form submission.
  + error: Error message for payment issues.
* **Functionality**:
  + Submits payment information.
  + Handles success and error states.
  + Navigates to the confirmation page upon successful payment.

### 2.4 Products

A screenshot of a computer

Description automatically generated

* **Description**: Displays a list of products and allows users to add them to the cart.
* **State Variables**:
  + products: Array of products fetched from the server.
* **Functionality**:
  + Fetches product data using axios.
  + Handles adding products to the cart.

### 2.5 App

* **Description**: Main application component.
* **Functionality**:
  + Sets up routing using react-router-dom.
  + Provides context with CartProvider.

## 3. Routing

The application uses react-router-dom for routing:

* **Home**: /
* **Products**: /products
* **Cart**: /cart
* **Add Product**: /add-product
* **Checkout**: /checkout
* **Confirmation**: /confirmation