PROJECT REPORT

on

JEAN STATION SHOPPING APP

Submitted by

ANUJA A

KARAN SINGH

SAI BHARADWAJ

CHAPTER 1

Introduction

This project is a web based shopping system for an existing shop. The project objective is to deliver the online shopping application into android platform . Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. It is a form of electronic commerce. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favourite shop.

Shopping has long been considered a recreational activity by many. Shopping online is no exception. The goal of this application is to develop a web based interface for online retailers. The system would be easy to use and hence make the shopping experience pleasant for the users.

The existing business model is insufficient to meet customer demands. To over come the current limitations, we are developing a software for online shopping named JeanStation application that can be globally used to shop the attire you desire . The store will have its own interface through which it will be able to manage the orders pertaining to that store.

CHAPTER 2

Objective

The objective of the project is to make an application in android platform to purchase items in an existing shop. A complete and efficient web application which can provide the online shopping experience is the basic objective of the project. The web application can be implemented in the form of an android application with web view.

The application was designed into two modules first is for the customers who wish to buy the articles. Second is for the storekeepers who maintains and updates the information pertaining to the articles and those of the customers. The end user of this product isa departmental store where the application is hosted on the web and the administrator maintains the database. The application which is deployed at the customer database, the details of the items are brought forward from the database for the customer view based on the selection through the menu and the database of all the products are updated at the end of each transaction.

CHAPTER 3

DESCRIPTION OF EXISTING SYSTEM

Currently, each JeanStation store has a store manager who leads the other staff of the store to perform the day-to-day operations.

To market their products, JeanStation uses the print and electronic media. JeanStation also has a website that they update to showcase the products available in their stores.

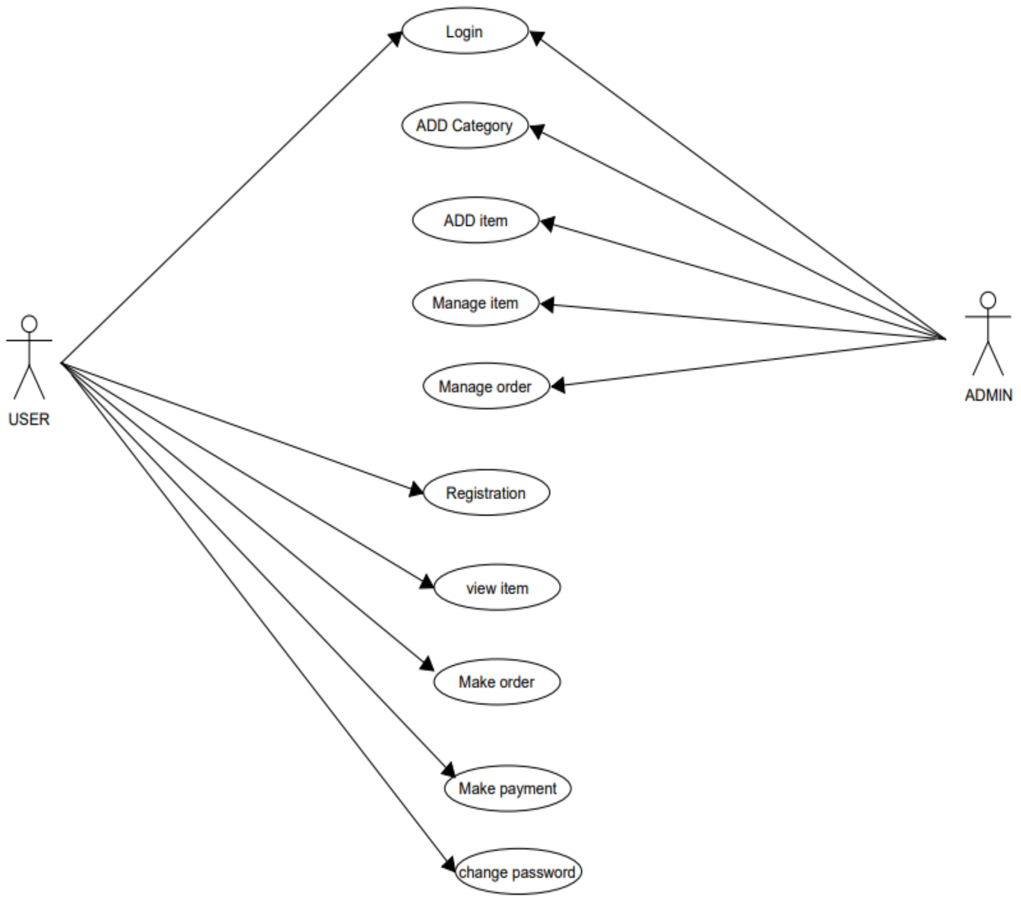
However, the current system of retail marketing has several limitations. Often customers who walk in to a store to buy a new advertised product complain when the product does not arrive at the store. As a result customers return back disappointed. In addition, most of the competitors of JeanStation have launched e-commerce solutions to make it convenient for customers to buy apparels without the need to visit a store.

All these factors are not only affecting the brand image of JeanStation but are also reducing sales, which has resulted in a loss in the company's revenue.

CHAPTER 4

OOAD OF PROJECT

4.1 Use Case Diagram



Above shows the use case diagram of JeanStation shopping cart application. It is a graphical depiction of user’s and admin’s possible interactions with a system.

Here use case diagram describe the relationship between users and use cases. A

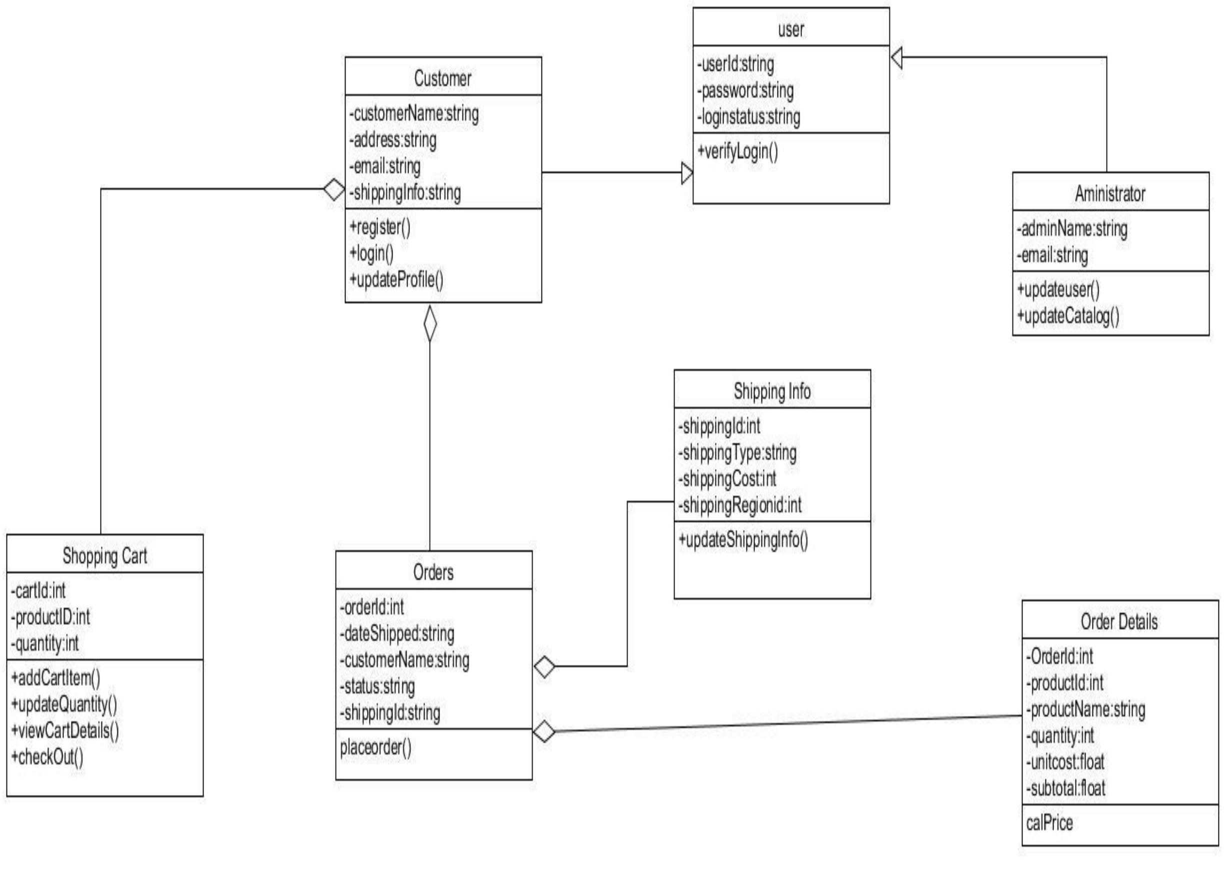
use case is a user activity in the system. It consist of two components, 1 Actor : Actors are represented with a label naming actors role. There may be multiple actors in a diagram.

2 Use case : Represented as ellipse with a label inside, naming the use case.

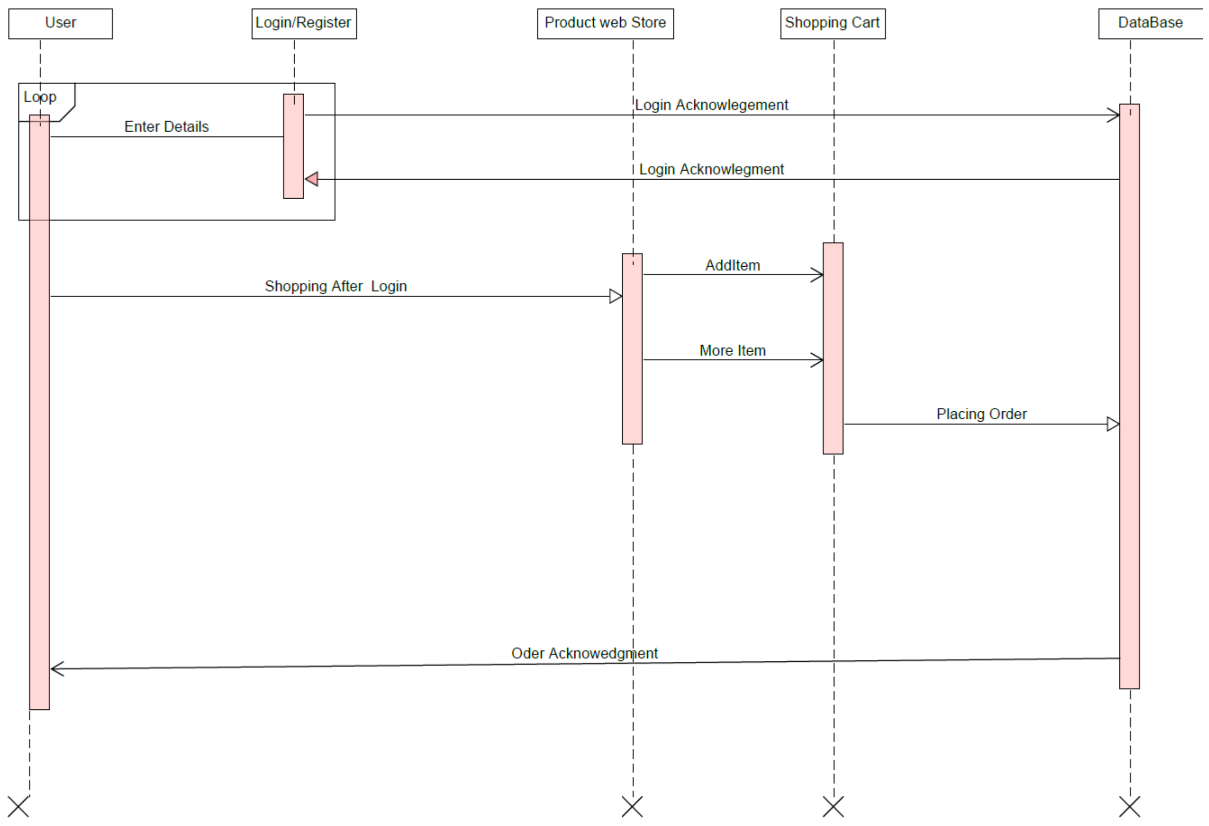
There may be multiple use cases in a diagram.

Actors represent the role that a user might play where each role is represented separately. Actor and Use case names must be unique with in a diagram. A use case describe the activity that is possible. It have several instances of activity throughout it’s life time. In JeanStation application user need to login first before he/she could order and purchase any desired apparels.

4.2ClassDiagram



4.3 Sequence Diagram



CHAPTER 5

TECHNOLOGIES USED IN PROJECT

- Java 11
- Spring Boot
- Maven
- Spring Security
- Spring Data JPA
- JWT Authentication
- Hibernate
- Bootstrap
- MySql-Database

Limitations

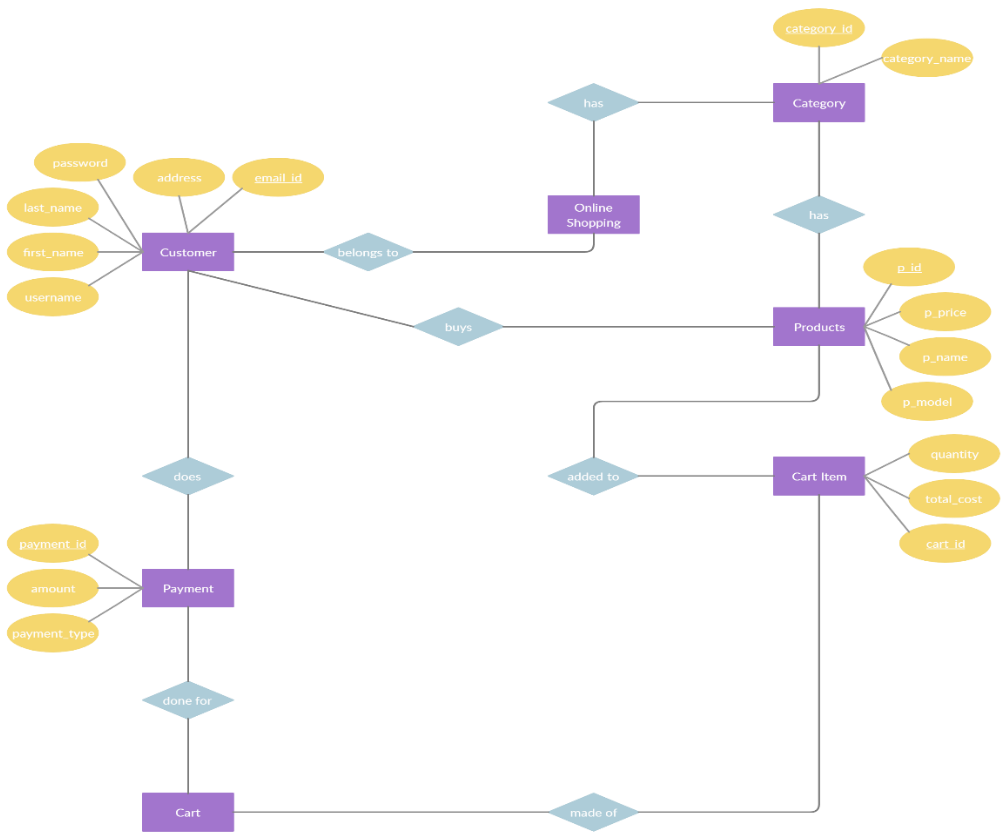
- Issues of security
- Technical glitch
- Delay in the delivery
- You can't touch the product
- Lack of interaction
- Returning the product

CHAPTER 6

DATABASE DESIGN OF PROJECT

6.1 ER Diagrams

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology system.

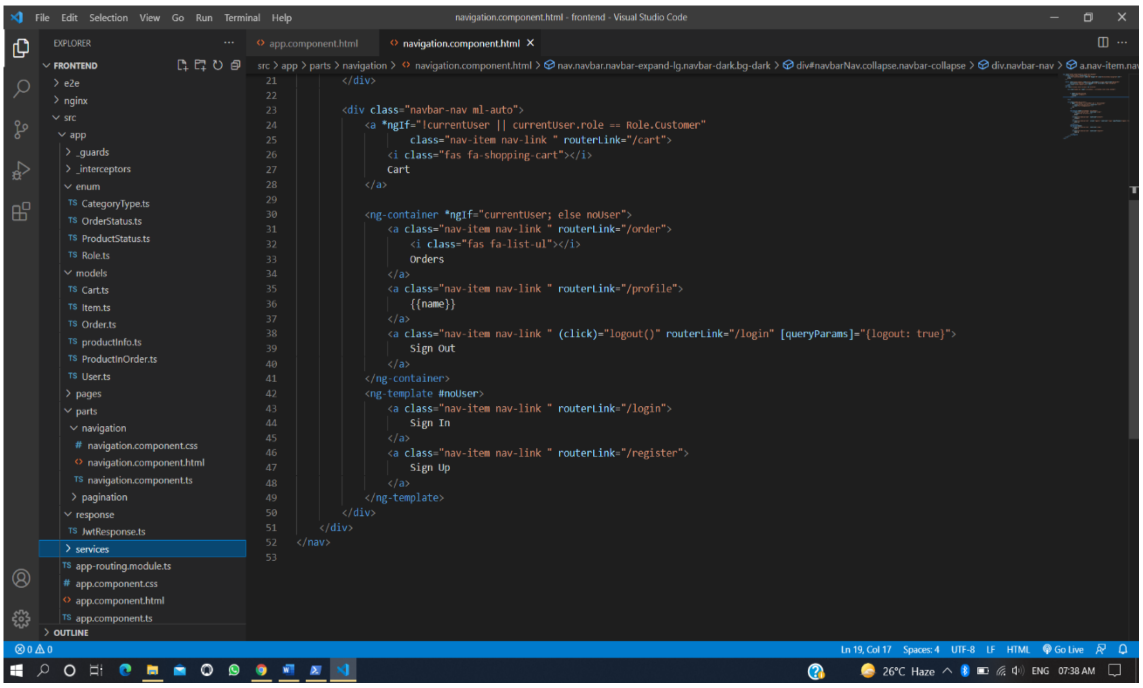


CHAPTER 7

SCOPE OF FUTURE RESEARCH

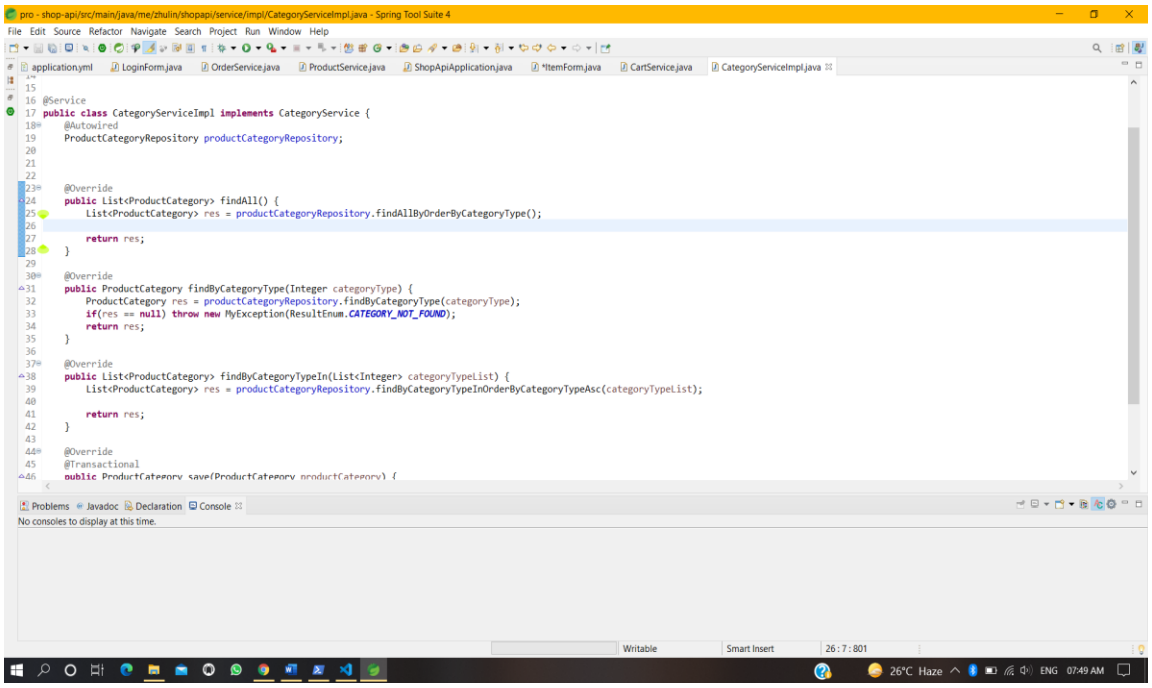
- **Artificial Intelligence :** Companies are now able to gather and investigate data in real-time thus facilitating competence and efficiency in the business. Machines itself are assisting businesses by performing all routine tasks, payments in a quick manner. Chatbots, CRM, internet of things are changing e-commerce domain. AI is connecting customers together and reducing efforts.
- **Google's Buy Now Button :** 'Buy Now' style button allow e-shoppers to search for any products on Google and purchasing can be done using single click only. The button offers customization and thus shopping process becomes flexible online. Thus without any headache product can be availed to the customer in less time.
- **App only Approach :** App only model is going to be of great use as the future of the internet lies in the mobiles. Mobile technologies are becoming a hub for the customers/ brand engagement creating a holistic experience. App only e-commerce model is proving itself as the best digital solution for business growth over the web.

Working Screenshots



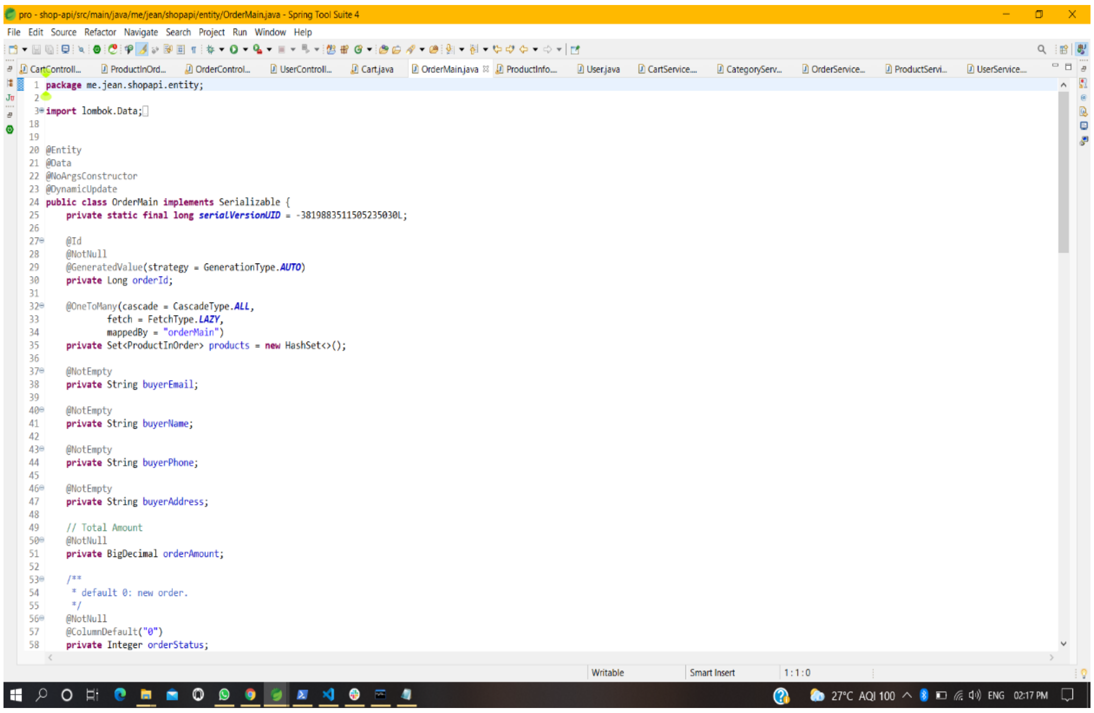
```
            </div>

            <div class="navbar-nav ml-auto">
                <a *ngIf="!currentUser || currentUser.role == Role.Customer"
                        class="nav-item nav-link " routerLink="/cart">
                    <i class="fas fa-shopping-cart"></i>
                    Cart
                </a>

                <ng-container *ngIf="currentUser; else noUser">
                    <a class="nav-item nav-link " routerLink="/order">
                        <i class="fas fa-list-ul"></i>
                        Orders
                    </a>
                    <a class="nav-item nav-link " routerLink="/profile">
                        {{name}}
                    </a>
                    <a class="nav-item nav-link " (click)="logout()" routerLink="/login" [queryParams]="{logout: true}">
                        Sign Out
                    </a>
                </ng-container>
                <ng-template #noUser>
                    <a class="nav-item nav-link " routerLink="/login">
                        Sign In
                    </a>
                    <a class="nav-item nav-link " routerLink="/register">
                        Sign Up
                    </a>
                </ng-template>
            </div>
        </div>
</nav>
```

Front end



```
@Service
public class CategoryServiceImpl implements CategoryService {
    @Autowired
    ProductCategoryRepository productCategoryRepository;



    @Override
    public List<ProductCategory> findAll() {
        List<ProductCategory> res = productCategoryRepository.findAllByOrderByCategoryType();

        return res;
    }

    @Override
    public ProductCategory findByCategoryType(Integer categoryType) {
        ProductCategory res = productCategoryRepository.findByCategoryType(categoryType);
        if(res == null) throw new MyException(ResultEnum.CATEGORY_NOT_FOUND);
        return res;
    }

    @Override
    public List<ProductCategory> findByCategoryTypeIn(List<Integer> categoryTypeList) {
        List<ProductCategory> res = productCategoryRepository.findByCategoryTypeInOrderByCategoryTypeAsc(categoryTypeList);

        return res;
    }

    @Override
    @Transactional
    public ProductCategory save(ProductCategory productCategory) {
```

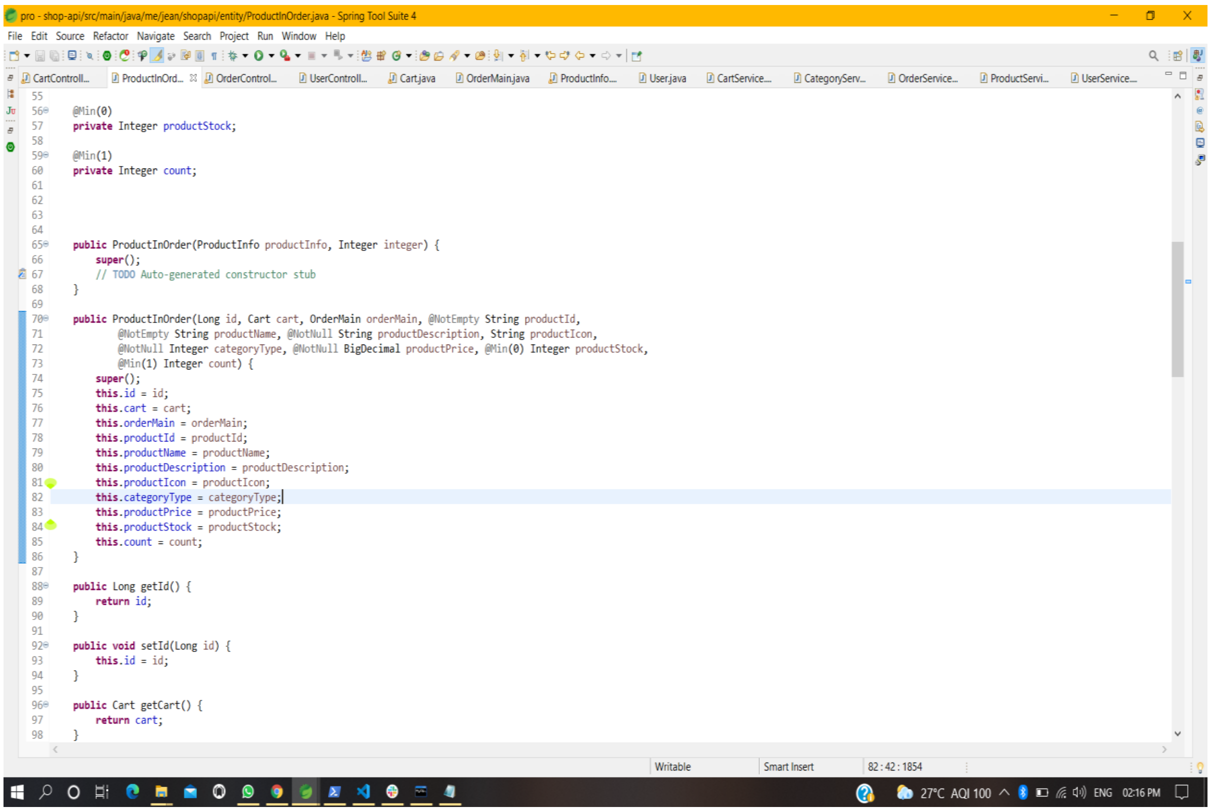
```
No consoles to display at this time.
```



pro - shop-api/src/main/java/me/jean/shopapi/entity/OrderMain.java - Spring Tool Suite 4

```java
package me.jean.shopapi.entity;

import lombok.Data;


@Entity
@Data
@NoArgsConstructor
@DynamicUpdate
public class OrderMain implements Serializable {
    private static final long serialVersionUID = -3819883511505235030L;

    @Id
    @NotNull
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long orderId;

    @OneToMany(cascade = CascadeType.ALL,
            fetch = FetchType.LAZY,
            mappedBy = "orderMain")
    private Set<ProductInOrder> products = new HashSet<>();

    @NotEmpty
    private String buyerEmail;

    @NotEmpty
    private String buyerName;

    @NotEmpty
    private String buyerPhone;

    @NotEmpty
    private String buyerAddress;

    // Total Amount
    @NotNull
    private BigDecimal orderAmount;

    /**
     * default 0: new order.
     */
    @NotNull
    @ColumnDefault("0")
    private Integer orderStatus;
```

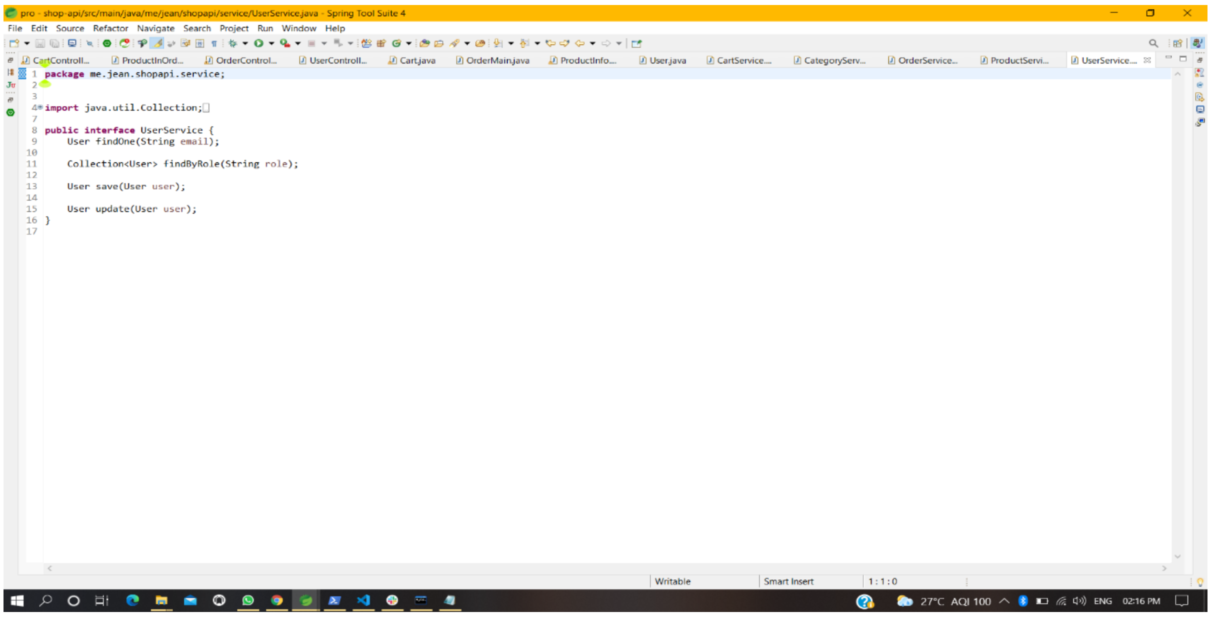
Writable   Smart Insert   1 : 1 : 0



pro - shop-api/src/main/java/me/jean/shopapi/entity/ProductInOrder.java - Spring Tool Suite 4

```java

    @Min(0)
    private Integer productStock;

    @Min(1)
    private Integer count;




    public ProductInOrder(ProductInfo productInfo, Integer integer) {
        super();
        // TODO Auto-generated constructor stub
    }

    public ProductInOrder(Long id, Cart cart, OrderMain orderMain, @NotEmpty String productId,
            @NotEmpty String productName, @NotNull String productDescription, String productIcon,
            @NotNull Integer categoryType, @NotNull BigDecimal productPrice, @Min(0) Integer productStock,
            @Min(1) Integer count) {
        super();
        this.id = id;
        this.cart = cart;
        this.orderMain = orderMain;
        this.productId = productId;
        this.productName = productName;
        this.productDescription = productDescription;
        this.productIcon = productIcon;
        this.categoryType = categoryType;
        this.productPrice = productPrice;
        this.productStock = productStock;
        this.count = count;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Cart getCart() {
        return cart;
    }
```

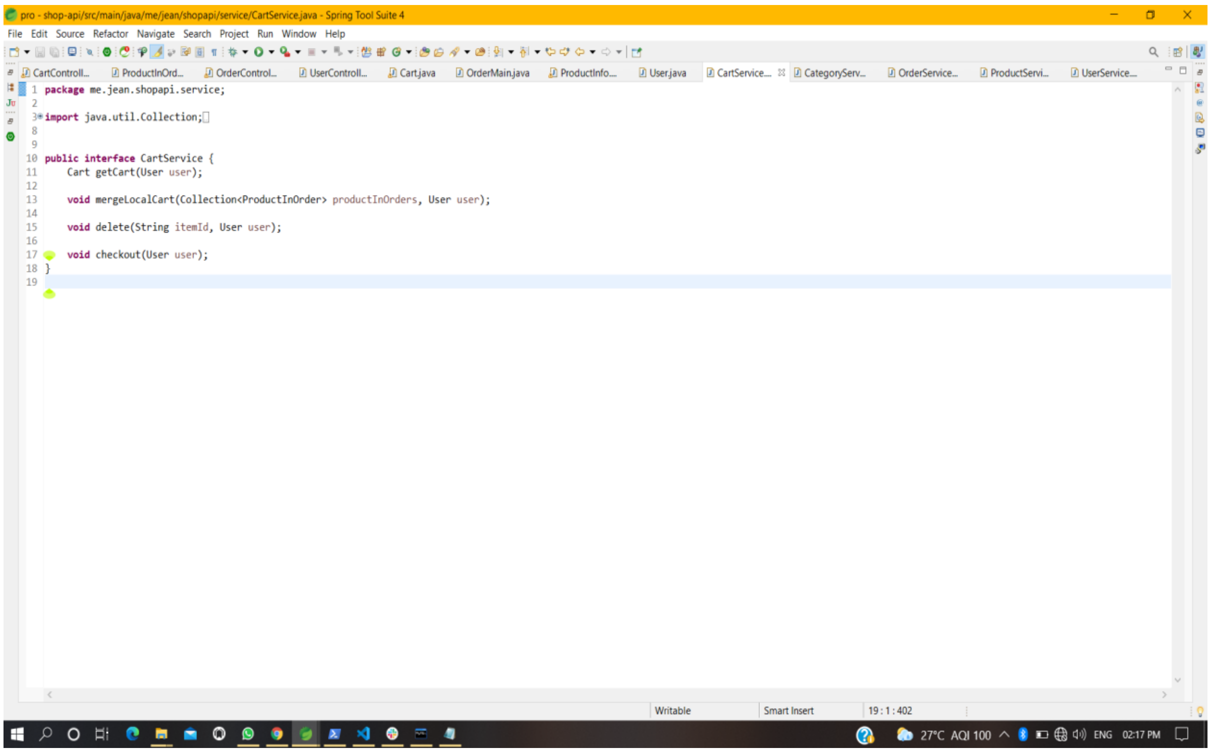
Writable   Smart Insert   82 : 42 : 1854


pro - shop-api/src/main/java/me/jean/shopapi/service/UserService.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help
CartControll... ProductInOrd... OrderControl... UserControll... Cart.java OrderMain.java ProductInfo... User.java CartService... CategoryServ... OrderService... ProductServi... UserService...

```java
package me.jean.shopapi.service;


import java.util.Collection;

public interface UserService {
    User findOne(String email);

    Collection<User> findByRole(String role);

    User save(User user);

    User update(User user);
}
```

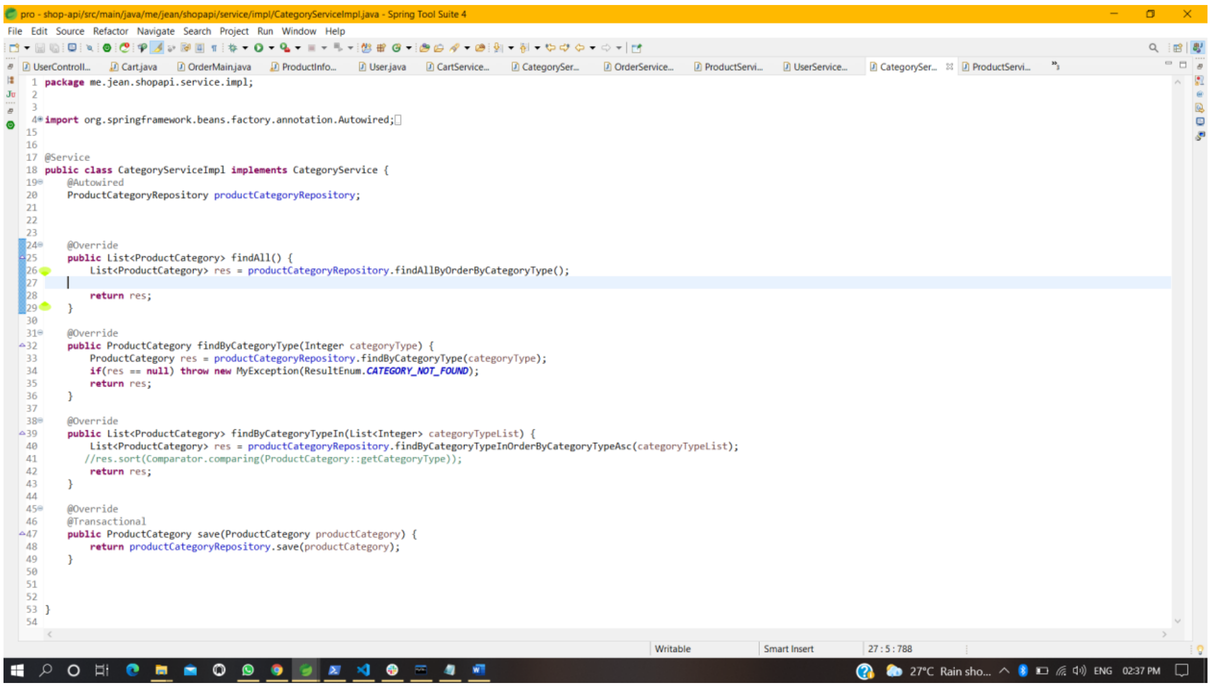
Writable
Smart Insert
1 : 1 : 0
27°C AQI 100
ENG
02:16 PM


pro - shop-api/src/main/java/me/jean/shopapi/service/CartService.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help
CartControll... ProductInOrd... OrderControl... UserControll... Cart.java OrderMain.java ProductInfo... User.java CartService... CategoryServ... OrderService... ProductServi... UserService...

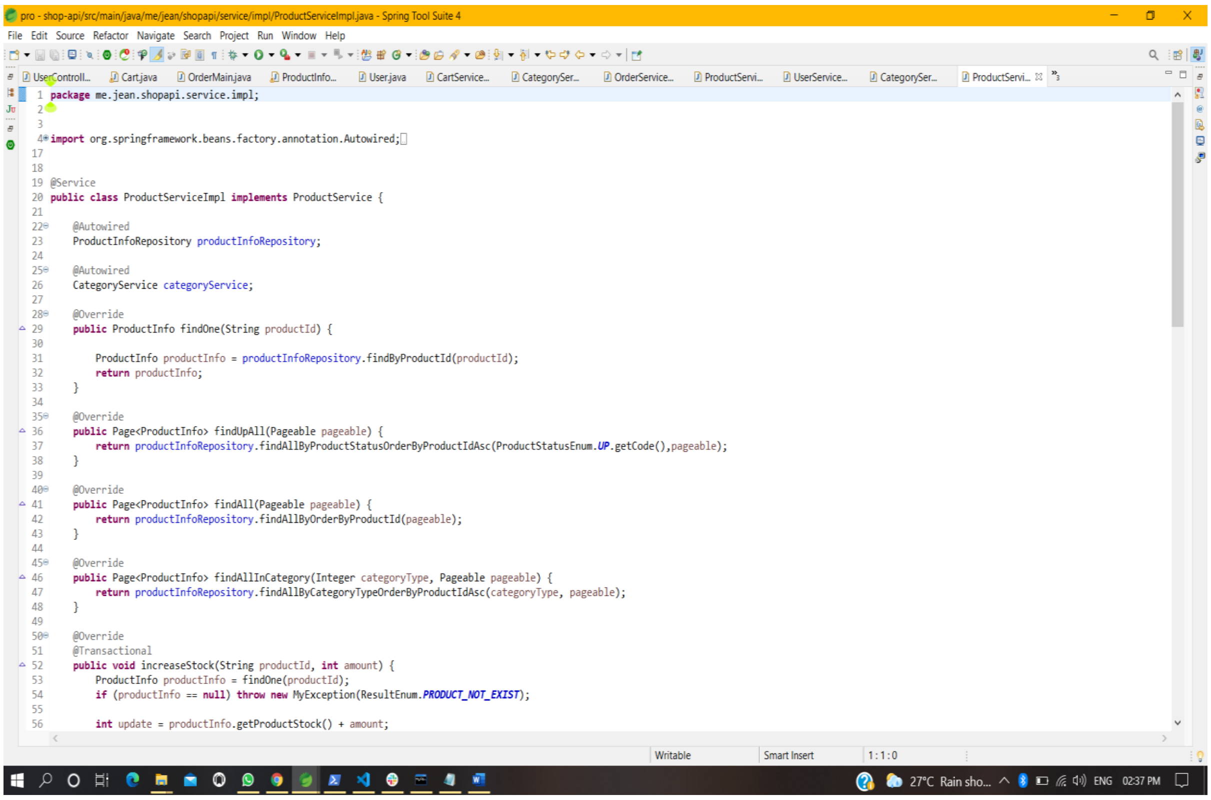
```java
package me.jean.shopapi.service;

import java.util.Collection;


public interface CartService {
    Cart getCart(User user);

    void mergeLocalCart(Collection<ProductInOrder> productInOrders, User user);

    void delete(String itemId, User user);

    void checkout(User user);
}
```

Writable
Smart Insert
19 : 1 : 402
27°C AQI 100
ENG
02:17 PM


pro - shop-api/src/main/java/me/jean/shopapi/service/impl/CategoryServiceImpl.java - Spring Tool Suite 4

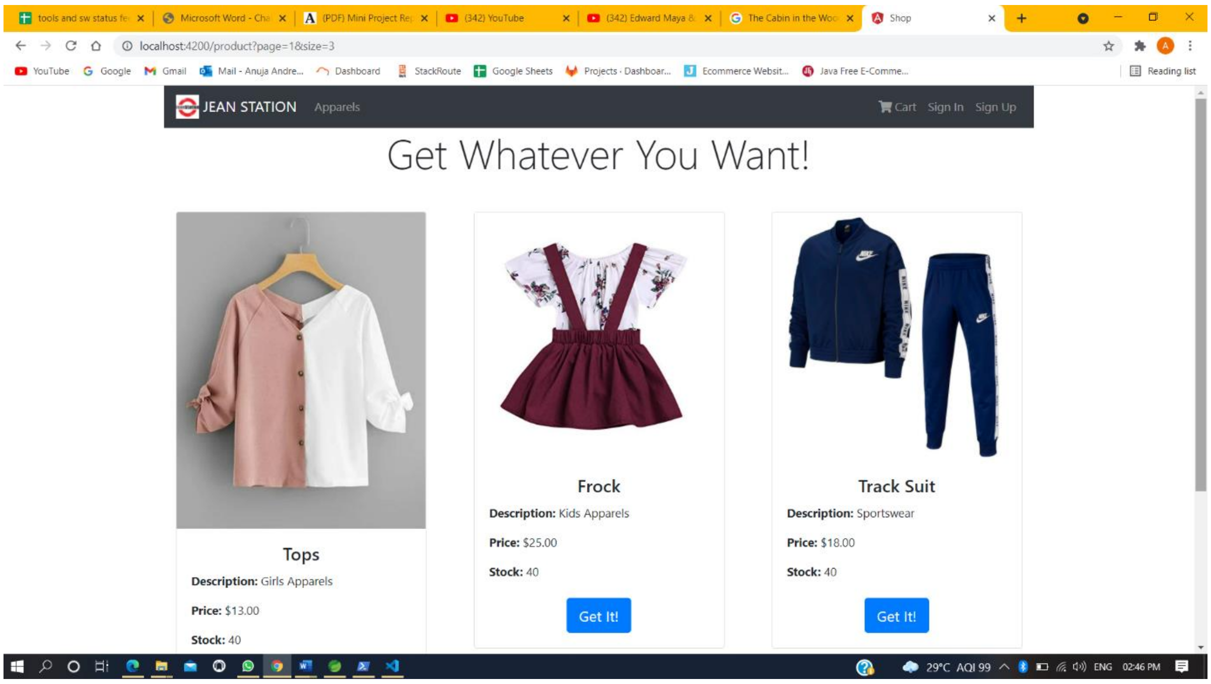
```java
package me.jean.shopapi.service.impl;


import org.springframework.beans.factory.annotation.Autowired;


@Service
public class CategoryServiceImpl implements CategoryService {
    @Autowired
    ProductCategoryRepository productCategoryRepository;



    @Override
    public List<ProductCategory> findAll() {
        List<ProductCategory> res = productCategoryRepository.findAllByOrderByCategoryType();

        return res;
    }

    @Override
    public ProductCategory findByCategoryType(Integer categoryType) {
        ProductCategory res = productCategoryRepository.findByCategoryType(categoryType);
        if(res == null) throw new MyException(ResultEnum.CATEGORY_NOT_FOUND);
        return res;
    }

    @Override
    public List<ProductCategory> findByCategoryTypeIn(List<Integer> categoryTypeList) {
        List<ProductCategory> res = productCategoryRepository.findByCategoryTypeInOrderByCategoryTypeAsc(categoryTypeList);
       //res.sort(Comparator.comparing(ProductCategory::getCategoryType));
        return res;
    }

    @Override
    @Transactional
    public ProductCategory save(ProductCategory productCategory) {
        return productCategoryRepository.save(productCategory);
    }



}
```

Writable   Smart Insert   27 : 5 : 788


pro - shop-api/src/main/java/me/jean/shopapi/service/impl/ProductServiceImpl.java - Spring Tool Suite 4

```java
package me.jean.shopapi.service.impl;


import org.springframework.beans.factory.annotation.Autowired;


@Service
public class ProductServiceImpl implements ProductService {

    @Autowired
    ProductInfoRepository productInfoRepository;

    @Autowired
    CategoryService categoryService;

    @Override
    public ProductInfo findOne(String productId) {

        ProductInfo productInfo = productInfoRepository.findByProductId(productId);
        return productInfo;
    }

    @Override
    public Page<ProductInfo> findUpAll(Pageable pageable) {
        return productInfoRepository.findAllByProductStatusOrderByProductIdAsc(ProductStatusEnum.UP.getCode(),pageable);
    }

    @Override
    public Page<ProductInfo> findAll(Pageable pageable) {
        return productInfoRepository.findAllByOrderByProductId(pageable);
    }

    @Override
    public Page<ProductInfo> findAllInCategory(Integer categoryType, Pageable pageable) {
        return productInfoRepository.findAllByCategoryTypeOrderByProductIdAsc(categoryType, pageable);
    }

    @Override
    @Transactional
    public void increaseStock(String productId, int amount) {
        ProductInfo productInfo = findOne(productId);
        if (productInfo == null) throw new MyException(ResultEnum.PRODUCT_NOT_EXIST);

        int update = productInfo.getProductStock() + amount;
```

Writable   Smart Insert   1 : 1 : 0



Get Whatever You Want!

Tops

Description: Girls Apparels

Price: $13.00

Stock: 40

Frock

Description: Kids Apparels

Price: $25.00

Stock: 40

Get It!

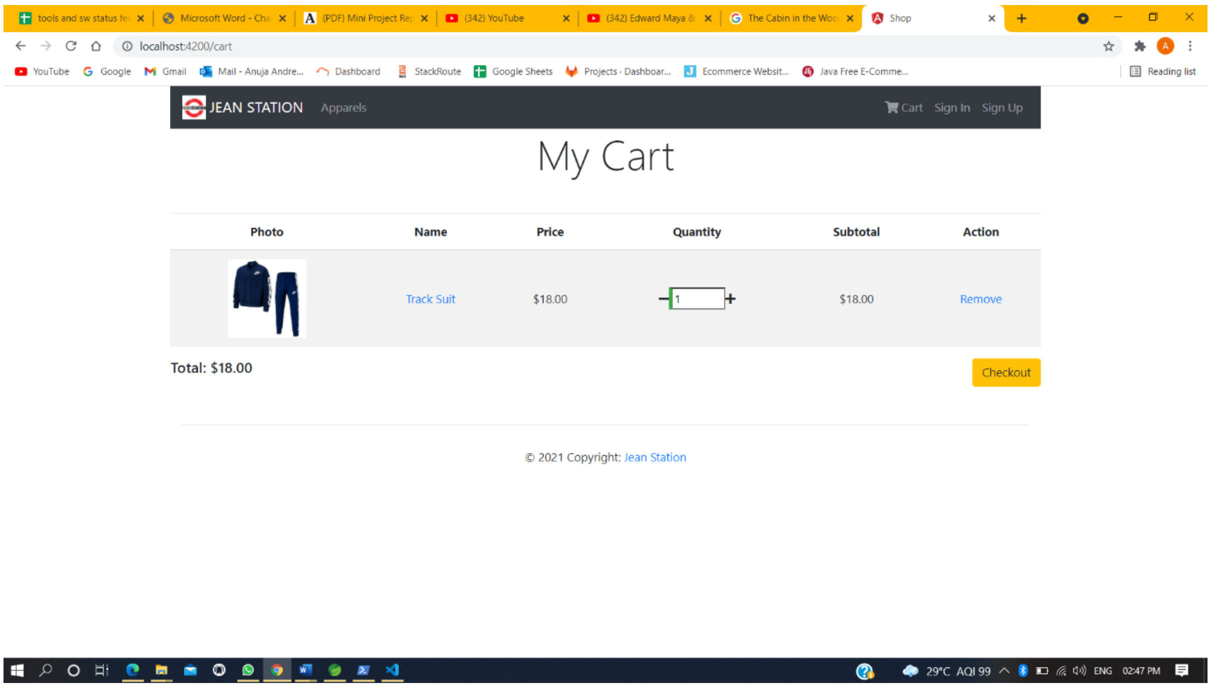
Track Suit

Description: Sportswear

Price: $18.00

Stock: 40

Get It!



My Cart

Photo	Name	Price	Quantity	Subtotal	Action
	Track Suit	$18.00	1	$18.00	Remove

Total: $18.00

Checkout

© 2021 Copyright: Jean Station

References

- https://www.javainuse.com/fullstack/ecommerce/shop
- https://medium.com/adyen/building-an-e-commerce-application-using-java-react-54015b81d6c9
- https://www.sivalabs.in/categories/microservices/
- https://www.javaguides.net/p/spring-boot-tutorial.html