A REPORT OF AN CAPSTONR PROJECT

SHOP FOR HOME

Welcome to our E-Commerce website

Batch-
Enterprise java
& GCP
Group-07

By-
S. Mounika
P.Prasanna
Md.Ayaz
Nikhil Khore
B.Shreyank

Abstract

The Shop For Home is an E-Commerce Website which is used to shop required products like groceries, home appliances, electronic devices, mobiles, beauty items etc.
This project is developed to make order of required products through online without direct interaction of
the customer to the seller.

The project is developed by using the technologies Angular, Springboot and PostgreSQL. The front is developed by using Angular, the backend is developed by using Springboot and PostgreSQL is the database which is used for data storing.

E-commerce brings convenience for customers as they do not have to leave home and only need to browse
website online, especially for buying the products which are not sold in nearby shops. It could help
customers buy wider range of products and save customers' time.

Consumers also gain power through online
shopping. They research products and compare prices among retailers

Table Of Content

Introduction

E-Commerce is short for electronic commerce, as in online commerce. It’s an umbrella term for any transaction done over the internet. eCommerce includes retail stores, such as clothing and other physical products, and services of all types, from cyber security to booking a hotel

The most common way to facilitate online sales and transactions, is through a dedicated online store, or eCommerce platform.

E-commerce helps create new job opportunities due to information related services, software app and digital products. It also causes job losses. The areas with the greatest predicted job-loss are retail, postal, and travel agencies. The development of e-commerce will create jobs that require highly skilled workers to manage
customer demands, and production processes. In contrast, people with poor technical skills cannot enjoy the wages welfare.

On the other hand, because e-commerce requires sufficient stocks that could be delivered to customers in time, the warehouse becomes an important element. Warehouse needs more staff to manage, supervise and organize, thus the condition of warehouse environment will be concerned by employees.

F-commerce brings convenience for customers as they do not have to leave home and only need to browse website online, especially for buying the products which are not sold in nearby shops. It could help customers buy wider range of products and save customers' time. Consumers also gain power through online shopping.

They research products and compare prices among retailers. Also, online shopping often provides
sales promotion or discounts code, thus it is more price effective for customers. Moreover, e-commerce provides products' detailed information; even the in-store staff cannot offer such detailed explanation. Customers can also review and track the order history online.

Technology Used

1. Angular
2. Springboot
3. postgreSql

Angular:

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into NgModules. NgModules collect related code into functional sets; an Angular application is defined by a set of NgModules. An application always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Modules, components and services are classes that use decorators. These decorators mark their type and provide metadata that tells Angular how to use them

● The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.

● The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI). An application's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities

Springboot:

Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core capabilities

: 1. Autoconfiguration
 2. An opinionated approach to configuration
 3. The ability to create standalone applications

These features work together to provide you with a tool that allows you to set up a Spring-based application with minimal configuration and setup.

Spring is widely used for creating scalable applications. For web applications Spring provides Spring MVC which is a widely used module of spring which is used to create scalable web applications.

But main disadvantage of spring projects is that configuration is really time-consume and can be a bit overwhelming for the new developers. Making the application production-ready takes some time if you are new to the spring.

Solution to this is Spring Boot. Spring Boot is built on the top of the spring and contains all the features of spring. And is becoming favourite of developer’s these days because of it’s a rapid production-ready environment which enables the developers to directly focus on the logic instead of struggling with the configuration and set up.

Spring Boot is a microservice-based framework and making a production-ready application in it takes very less time. Prerequisite for Spring Boot is the basic knowledge Spring framework.

PostgreSQL

PostgreSQL is an enterprise-class open source database management system. It supports both SQL and JSON for relational and non-relational queries for extensibility and SQL compliance.

PostgreSQL supports advanced data types and performance optimization features, which are only available in expensive commercial databases, like Oracle and SQL Server. It is also known as Postgres.

- Helps developers to build applications.
-
- It allows administrators to build fault-tolerant environment by protecting data integrity.
-
- Compatible with various platforms using all major languages and middleware.
-
- It offers a most sophisticated locking mechanism.
-
- Support for multi-version concurrency control.
-
- Mature Server-Side Programming Functionality.
-
- Compliant with the ANSI SQL standard.
-
- Full support for client-server network architecture.

Problem Statement: ShopForHome is a popular Store in the market for shopping the home décor stuff.
Due to Covid 19 all the offline shopping stopped. So, the store wants to move to the online platforms and wants their own web application.
There are 2 users on the application: -

1. User
2. Admin

User Stories –

1. As a user I should be able to login, Logout and Register into the application.
2. As a user I should be able to see the products in different categories.
3. As a user I should be able to sort the products.
4. As a user I should be able to add the products into the shopping cart.
5. As a user I should be able to increase or decrease the quantity added in the cart.
6. As a user I should be able to add "n" number of products in the cart.
7. As a user I should be able to get the Wishlist option where I can add those products which I want but don't want to order now.
8. As a user I should get different discount coupons.

Admin Stories –

1. As an Admin I should be able to login, Logout and Register into the application.
2. .As an Admin I should be able to perform CRUD on Users.
3. As an Admin I should be able to Perform CRUD on the products.
4. As an Admin I should be able to get bulk upload option to upload a csv for products details.
5. As an Admin I should be able to get the stocks.
6. As an Admin I should be able to mail if any stock is less than 10.
7. As an Admin I should be able to get the sales report of a specific duration.
8. As an Admin I should be able to set the discount coupons for the specific set of users.

Instructions –

1. Please use a folder on server to upload the images.
2. Please share the database structure in the .sql file.
3. Please create a separate microservice for reports and discount coupons.
4. Please use separate port to deploy the Angular UI and Spring Boot Microservice.
5. Please use the UI designing tool like (Bootstrap or Material) to make your UI better.
6. Please use Material UI to create the UI

BackEnd Implementation:

1.Initializing Spring Boot:

To start with Spring Boot REST API, you first need to initialize the Spring Boot Project. You can easily initialize a new Spring Boot Project with Spring Initializer.

From your Web Browser, go to start.spring.io. Choose Maven as your Build Tool and Language as Java. Select the specific version of Spring Boot you want to go ahead with.

Add dependencies:

1. Spring Web
2. Spring Boot DevTools
3. Spring Data JPA
4. MySQL Driver
5. JDBC API
6. Web Socket

2.Connecting Spring Boot to the Database:

Next, you need to set up the Database, and you can do it easily with Spring Data JPA.

Add some elementary information in your application.properties file to set up the connection to your preferred Database. Add your JDBC connection URL, provide a username and password for authentication, and set the ddl-auto property to update.

And also add server port within the application.properties by using server.port.

3.Create Required Packages:
Create new packages as per requirement with in the initiated project.
The packages are generated in the project are mentioned below:

1. Config
2. Controller
3. Exception
4. Model
5. Repository
6. Service

4.Develop the code as per requirement to the project:
The code respective to the backend mentioned within the Source Code

5.Creating Discount MicroService:
By adding the required dependencies and packages, developed the code as per the requirement.
Add some elementary information in your application.properties file to set up the connection to your preferred Database.

Add your JDBC connection URL, provide a username and password for authentication, and set the ddl-auto property to update. And also add server port within the application.properties by using server.port.

The code developed for the discount microservice are mentioned within the Source Code section.

FrontEnd Implementation:

Initially Install the Angular within the PC by using the command npm install -g @angular/cli

After installation,

1.Creating Project:

Create an angular project by using the commad ng new Project-Name

The project is generated after the installation of required package. Open the project with the help of Visual Studio Code. After opening the project with the VS code develop the code as per the project requirement.

2.Creating Components:

For creating the components use the command ng generate component component-name.
The Components within the project are:

1. admin
2. adminlogin
3. cart
4. checkout
5. e-commerce
6. header
7. payment
8. products
9. uploadfiles
10. userlogin
11. userregister

12. users
13. wishlist

3.Execution:

The code for the components is mentioned within the Source Code section.
Make sure that app-routing.module.ts is well written and importing of respective component modules are done perfectly.

After the completing of implementation part for executing the project by using the command ng serve

If the code is error free it executes perfectly by showing the port number.
GREAT LEARNING C1-G10Working with Database:
By the successful execution of backend developed project with the Spring Tool Suite then there is generation of data tables within the triggered database location.

After creation of data tables within the database location, Add admin details with in the admin tables by using the database query INSERT, like insert into admin values(‘admin@gmail.com’,’admin’).

This is because of, the project don’t have any admin registration so if we need to perform any operation with respect to admin we need to add admin details within the admin table in the database.

Source Code

BackEnd:

Entity:

User:

```
package eshop.wipro.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.NaturalId;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import java.io.Serializable;

@Entity
@Data
@Table(name = "users")
@NoArgsConstructor
public class User implements Serializable {

    private static final long serialVersionUID = 4887904943282174032L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NaturalId
    @NotEmpty
    private String email;
    @NotEmpty
    @Size(min = 3, message = "Length must be more than 3")
    private String password;
    @NotEmpty
    private String name;
    @NotEmpty
    private String phone;
    @NotEmpty
    private String address;
    @NotNull
    private boolean active;
    @NotEmpty
    private String role = "ROLE_CUSTOMER";

    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL, fetch
= FetchType.LAZY)
    @JsonIgnore  // fix bi-direction toString() recursion problem
    private Cart cart;



    public Long getId() {
        return id;
    }


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


    public String getEmail() {
        return email;
    }


    public void setEmail(String email) {
        this.email = email;
    }
```

```
public String getPassword() {
   return password;
}


public void setPassword(String password) {
   this.password = password;
}


public String getName() {
   return name;
}


public void setName(String name) {
   this.name = name;
}


public String getPhone() {
   return phone;
}


public void setPhone(String phone) {
   this.phone = phone;
}


public String getAddress() {
   return address;
}


public void setAddress(String address) {
   this.address = address;
}


public boolean isActive() {
   return active;
}
```

```
    public void setActive(boolean active) {
        this.active = active;
    }


    public String getRole() {
        return role;
    }


    public void setRole(String role) {
        this.role = role;
    }


    public Cart getCart() {
        return cart;
    }


    public void setCart(Cart cart) {
        this.cart = cart;
    }


    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", email='" + email + '\'' +
                ", password='" + password + '\'' +
                ", name='" + name + '\'' +
                ", phone='" + phone + '\'' +
                ", address='" + address + '\'' +
                ", active=" + active +
                ", role='" + role + '\'' +
                '}';
    }

}
```

ProductInfo

```
package eshop.wipro.entity;

import lombok.Data;
import org.hibernate.annotations.ColumnDefault;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.DynamicUpdate;
import org.hibernate.annotations.UpdateTimestamp;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

@Entity
@Data
@DynamicUpdate
public class ProductInfo implements Serializable {
    @Id
    private String productId;

    @NotNull
    private String productName;

    @NotNull
    private BigDecimal productPrice;

    @NotNull
    @Min(0)
    private Integer productStock;

    private String productDescription;

    private String productIcon;

    /** 0: on-sale 1: off-sale */

    @ColumnDefault("0")
```

```
}

private Integer productStatus;

@ColumnDefault("0")
private Integer categoryType;

@CreationTimestamp
private Date createTime;
@UpdateTimestamp
private Date updateTime;

public ProductInfo() {
}

public String getProductId() {
   return productId;
}

public void setProductId(String productId) {
   this.productId = productId;
}

public String getProductName() {
   return productName;
}

public void setProductName(String productName) {
   this.productName = productName;
}

public BigDecimal getProductPrice() {
   return productPrice;
}

public void setProductPrice(BigDecimal productPrice) {
   this.productPrice = productPrice;
}

public Integer getProductStock() {
   return productStock;
}

public void setProductStock(Integer productStock) {
   this.productStock = productStock;
```

```
}

public String getProductDescription() {
   return productDescription;
}

public void setProductDescription(String productDescription) {
   this.productDescription = productDescription;
}

public String getProductIcon() {
   return productIcon;
}

public void setProductIcon(String productIcon) {
   this.productIcon = productIcon;
}

public Integer getProductStatus() {
   return productStatus;
}

public void setProductStatus(Integer productStatus) {
   this.productStatus = productStatus;
}

public Integer getCategoryType() {
   return categoryType;
}

public void setCategoryType(Integer categoryType) {
   this.categoryType = categoryType;
}

public Date getCreateTime() {
   return createTime;
}

public void setCreateTime(Date createTime) {
   this.createTime = createTime;
}

public Date getUpdateTime() {
        return updateTime;
    }

    public void setUpdateTime(Date updateTime) {
        this.updateTime = updateTime;
    }


}
```

Wishlist:

```
package eshop.wipro.entity;


import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "wishlist")
public class WishList implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
```

```
}


@OneToOne(targetEntity = User.class, fetch = FetchType.EAGER)
@JoinColumn(nullable = false, name = "user_id")
private User user;

@Column(name = "created_date")
private Date createdDate;

@ManyToOne()
@JoinColumn(name = "product_id")
private ProductInfo product;

public WishList() {
}

public WishList(User user, ProductInfo product) {
   this.user = user;
   this.product = product;
   this.createdDate = new Date();
}


public Long getId() {
   return id;
}

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

public User getUser() {
   return user;
}

public void setUser(User user) {
   this.user = user;
}

public Date getCreatedDate() {
   return createdDate;
}

public void setCreatedDate(Date createdDate) {
        this.createdDate = createdDate;
    }

    public ProductInfo getProduct() {
        return product;
    }

    public void setProduct(ProductInfo product) {
        this.product = product;
    }
```

cart

```
package eshop.wipro.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

@Data
@Entity
@NoArgsConstructor
public class Cart implements Serializable {
    @Id
    @NotNull
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long cartId;

    @OneToOne(fetch = FetchType.LAZY)
    @MapsId
    @JsonIgnore
//    @JoinColumn(name = "email", referencedColumnName = "email")
    private User user;

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

@Override
public String toString() {
    return "Cart{" +
            "cartId=" + cartId +
            ", products=" + products +
            '}';
}

public Cart(User user) {
    this.user  = user;
}

public long getCartId() {
    return cartId;
}

public void setCartId(long cartId) {
    this.cartId = cartId;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public Set<ProductInOrder> getProducts() {
    return products;
}

public void setProducts(Set<ProductInOrder> products) {
    this.products = products;
}

public Cart() {

}
```

```
}
```

Discount

```
package eshop.wipro.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@Table(name = "discount")
@Entity
public class Discount {
    /**
     *
     */
    @Id
    @Column(name="id")
    private String id;
    @Column(name="status")
    private Long status;


    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }

    public Long getStatus() {
        return status;
    }
    public void setStatus(Long status) {
        this.status = status;
    }
    public Discount(){

    }

}
```

Product Category

```
package eshop.wipro.entity;

import lombok.Data;
import org.hibernate.annotations.DynamicUpdate;
import org.hibernate.annotations.NaturalId;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.io.Serializable;
import java.util.Date;

@Entity
@Data
@DynamicUpdate
public class ProductCategory implements Serializable {
    @Id
    @GeneratedValue
    private Integer categoryId;

    private String categoryName;

    @NaturalId
    private Integer categoryType;

    private Date createTime;

    private Date updateTime;

    public ProductCategory() {
    }
```

```
}

public ProductCategory(String categoryName, Integer categoryType) {
    this.categoryName = categoryName;
    this.categoryType = categoryType;
}

public Integer getCategoryId() {
    return categoryId;
}

public void setCategoryId(Integer categoryId) {
    this.categoryId = categoryId;
}

public String getCategoryName() {
    return categoryName;
}

public void setCategoryName(String categoryName) {
    this.categoryName = categoryName;
}

public Integer getCategoryType() {
    return categoryType;
}

public void setCategoryType(Integer categoryType) {
    this.categoryType = categoryType;
}

public Date getCreateTime() {
    return createTime;
}

public void setCreateTime(Date createTime) {
    this.createTime = createTime;
}

public Date getUpdateTime() {
    return updateTime;
}

public void setUpdateTime(Date updateTime) {
        this.updateTime = updateTime;
    }


}
```

Order Main

```
package eshop.wipro.entity;

import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.ColumnDefault;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.DynamicUpdate;
import org.hibernate.annotations.UpdateTimestamp;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.io.Serializable;
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.HashSet;
import java.util.Set;

@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;

    @CreationTimestamp
    private LocalDateTime createTime;

    @UpdateTimestamp
    private LocalDateTime updateTime;

    public OrderMain(User buyer) {
        this.buyerEmail = buyer.getEmail();
        this.buyerName = buyer.getName();
        this.buyerPhone = buyer.getPhone();
        this.buyerAddress = buyer.getAddress();
        this.orderAmount = buyer.getCart().getProducts().stream().map(item
-> item.getProductPrice().multiply(new BigDecimal(item.getCount())))
                .reduce(BigDecimal::add)
                .orElse(new BigDecimal(0));
        this.orderStatus = 0;
```

```
}

public Long getOrderId() {
    return orderId;
}

public void setOrderId(Long orderId) {
    this.orderId = orderId;
}

public Set<ProductInOrder> getProducts() {
    return products;
}

public void setProducts(Set<ProductInOrder> products) {
    this.products = products;
}

public String getBuyerEmail() {
    return buyerEmail;
}

public void setBuyerEmail(String buyerEmail) {
    this.buyerEmail = buyerEmail;
}

public String getBuyerName() {
    return buyerName;
}

public void setBuyerName(String buyerName) {
    this.buyerName = buyerName;
}

public String getBuyerPhone() {
    return buyerPhone;
}

public void setBuyerPhone(String buyerPhone) {
    this.buyerPhone = buyerPhone;
}
```

```
}

public String getBuyerAddress() {
    return buyerAddress;
}

public void setBuyerAddress(String buyerAddress) {
    this.buyerAddress = buyerAddress;
}

public BigDecimal getOrderAmount() {
    return orderAmount;
}

public void setOrderAmount(BigDecimal orderAmount) {
    this.orderAmount = orderAmount;
}

public Integer getOrderStatus() {
    return orderStatus;
}

public void setOrderStatus(Integer orderStatus) {
    this.orderStatus = orderStatus;
}

public LocalDateTime getCreateTime() {
    return createTime;
}

public void setCreateTime(LocalDateTime createTime) {
    this.createTime = createTime;
}

public LocalDateTime getUpdateTime() {
    return updateTime;
}

public void setUpdateTime(LocalDateTime updateTime) {
    this.updateTime = updateTime;
}

public OrderMain() {

}
```

```
}
```

Product in Order

```
package eshop.wipro.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.math.BigDecimal;
import java.util.Objects;

@Entity
@Data
@NoArgsConstructor
public class ProductInOrder {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, cascade =
CascadeType.REMOVE)
//    @JoinColumn(name = "cart_id")
    @JsonIgnore
    private Cart cart;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "order_id")
    @JsonIgnore
    private OrderMain orderMain;
```

```
@NotEmpty
private String productId;

@NotEmpty
private String productName;

@NotNull
private String productDescription;

private String productIcon;

@NotNull
private Integer categoryType;

@NotNull
private BigDecimal productPrice;

@Min(0)
private Integer productStock;

@Min(1)
private Integer count;

public ProductInOrder() {

}

public ProductInOrder(ProductInfo productInfo, Integer quantity) {
    this.productId = productInfo.getProductId();
    this.productName = productInfo.getProductName();
    this.productDescription = productInfo.getProductDescription();
    this.productIcon = productInfo.getProductIcon();
    this.categoryType = productInfo.getCategoryType();
    this.productPrice = productInfo.getProductPrice();
    this.productStock = productInfo.getProductStock();
    this.count = quantity;
}

@Override
public String toString() {
    return "ProductInOrder{" +
            "id=" + id +
            ", productId='" + productId + '\'' +
                ", productName='" + productName + '\'' +
                ", productDescription='" + productDescription + '\'' +
                ", productIcon='" + productIcon + '\'' +
                ", categoryType=" + categoryType +
                ", productPrice=" + productPrice +
                ", productStock=" + productStock +
                ", count=" + count +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        ProductInOrder that = (ProductInOrder) o;
        return Objects.equals(id, that.id) &&
                Objects.equals(productId, that.productId) &&
                Objects.equals(productName, that.productName) &&
                Objects.equals(productDescription, that.productDescription)
&&
                Objects.equals(productIcon, that.productIcon) &&
                Objects.equals(categoryType, that.categoryType) &&
                Objects.equals(productPrice, that.productPrice);
    }

    @Override
    public int hashCode() {

        return Objects.hash(super.hashCode(), id, productId, productName,
productDescription, productIcon, categoryType, productPrice);
    }

    public Long getId() {
        return id;
    }

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

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

```
}

public void setCart(Cart cart) {
   this.cart = cart;
}

public OrderMain getOrderMain() {
   return orderMain;
}

public void setOrderMain(OrderMain orderMain) {
   this.orderMain = orderMain;
}

public String getProductId() {
   return productId;
}

public void setProductId(String productId) {
   this.productId = productId;
}

public String getProductName() {
   return productName;
}

public void setProductName(String productName) {
   this.productName = productName;
}

public String getProductDescription() {
   return productDescription;
}

public void setProductDescription(String productDescription) {
   this.productDescription = productDescription;
}

public String getProductIcon() {
   return productIcon;
}

public void setProductIcon(String productIcon) {
        this.productIcon = productIcon;
    }

    public Integer getCategoryType() {
        return categoryType;
    }

    public void setCategoryType(Integer categoryType) {
        this.categoryType = categoryType;
    }

    public BigDecimal getProductPrice() {
        return productPrice;
    }

    public void setProductPrice(BigDecimal productPrice) {
        this.productPrice = productPrice;
    }

    public Integer getProductStock() {
        return productStock;
    }

    public void setProductStock(Integer productStock) {
        this.productStock = productStock;
    }

    public Integer getCount() {
        return count;
    }

    public void setCount(Integer count) {
        this.count = count;
    }


}
```

Repository
User repository
```
package eshop.wipro.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import eshop.wipro.entity.User;

import java.util.Collection;

public interface UserRepository extends JpaRepository<User, String> {
    User findByEmail(String email);
    Collection<User> findAllByRole(String role);

}
```

WishlistCustom repository
```
package eshop.wipro.repository;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.hibernate.Session;
import org.hibernate.query.NativeQuery;
import org.springframework.stereotype.Repository;

import eshop.wipro.entity.User;

@Repository
public class WishListCustomRepository {

    @PersistenceContext
    private EntityManager entityManager;




    public Boolean deleteWishlist(User user, String productId) {
         getSession().createNativeQuery("delete from public.wishlist where
product_id=:productId and user_id=:userId")
                .setParameter("productId", productId)
                .setParameter("userId", user.getId()).executeUpdate();
        return true;
    }

     public Session getSession() {
            return entityManager.unwrap(Session.class);
        }

}
```

wishlist

```
package eshop.wipro.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import eshop.wipro.entity.WishList;

@Repository
public interface WishListRepository extends JpaRepository<WishList,
Integer> {

    Page<WishList> findAllByUserId(Long id, Pageable pageable);




}
```

Product category

```
package eshop.wipro.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

import eshop.wipro.entity.ProductInfo;

public interface ProductInfoRepository extends
JpaRepository<ProductInfo, String> {
    ProductInfo findByProductId(String id);
    // onsale product
    Page<ProductInfo>
findAllByProductStatusOrderByProductIdAsc(Integer productStatus,
Pageable pageable);

    // product in one category
    Page<ProductInfo>
findAllByCategoryTypeOrderByProductIdAsc(Integer categoryType,
Pageable pageable);

    Page<ProductInfo> findAllByOrderByProductId(Pageable pageable);

}
```

product in order

```
package eshop.wipro.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import eshop.wipro.entity.ProductInOrder;

public interface ProductInOrderRepository extends
JpaRepository<ProductInOrder, Long> {

}
```

Product in custom repository

```
package eshop.wipro.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

import eshop.wipro.entity.ProductCategory;

import java.util.List;

public interface ProductCategoryRepository extends
JpaRepository<ProductCategory, Integer> {
    // Some category
    List<ProductCategory>
findByCategoryTypeInOrderByCategoryTypeAsc(List<Integer>
categoryTypes);
    // All category
    List<ProductCategory> findAllByOrderByCategoryType();
    // One category
    ProductCategory findByCategoryType(Integer categoryType);
}
```

Discount

```
package eshop.wipro.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import eshop.wipro.entity.Discount;

public interface DiscountRepository extends JpaRepository<Discount,
String> {

}
```

order

```
package eshop.wipro.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import eshop.wipro.entity.OrderMain;

public interface OrderRepository extends JpaRepository<OrderMain,
Integer> {
    OrderMain findByOrderId(Long orderId);

    Page<OrderMain>
findAllByOrderStatusOrderByCreateTimeDesc(Integer orderStatus,
Pageable pageable);

    Page<OrderMain>
findAllByBuyerEmailOrderByOrderStatusAscCreateTimeDesc(String
buyerEmail, Pageable pageable);

    Page<OrderMain>
findAllByOrderByOrderStatusAscCreateTimeDesc(Pageable pageable);

    Page<OrderMain>
findAllByBuyerPhoneOrderByOrderStatusAscCreateTimeDesc(String
buyerPhone, Pageable pageable);
}
```

Cart Repository

```
package eshop.wipro.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import eshop.wipro.entity.Cart;



public interface CartRepository extends JpaRepository<Cart, Integer> {
}
```

Service

user

```
package eshop.wipro.service;

import java.util.Collection;
import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;

import eshop.wipro.entity.User;

public interface UserService {
   User findOne(String email);

   Collection<User> findByRole(String role);

   User save(User user);

   User update(User user);

   List<User> findAll();

   Object update(Long userId);

   User update(String email);

   Page<User> findAll(PageRequest request);

   User removeAdmin(String email);
}
```

product service

```
package eshop.wipro.service;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import eshop.wipro.entity.ProductInfo;

public interface ProductService {

    ProductInfo findOne(String productId);

    // All selling products
    Page<ProductInfo> findUpAll(Pageable pageable);
    // All products
    Page<ProductInfo> findAll(Pageable pageable);
    // All products in a category
    Page<ProductInfo> findAllInCategory(Integer categoryType, Pageable
pageable);

    // increase stock
    void increaseStock(String productId, int amount);

    //decrease stock
    void decreaseStock(String productId, int amount);

    ProductInfo offSale(String productId);

    ProductInfo onSale(String productId);

    ProductInfo update(ProductInfo productInfo);
    ProductInfo save(ProductInfo productInfo);

    void delete(String productId);

    List<ProductInfo> findAll();
}
```

Category

```
package eshop.wipro.service;

import java.util.List;

import eshop.wipro.entity.ProductCategory;

public interface CategoryService {

    List<ProductCategory> findAll();

    ProductCategory findByCategoryType(Integer categoryType);

    List<ProductCategory> findByCategoryTypeIn(List<Integer>
categoryTypeList);

    ProductCategory save(ProductCategory productCategory);

}
```

cart

```
package eshop.wipro.service;

import java.util.Collection;

import eshop.wipro.entity.Cart;
import eshop.wipro.entity.ProductInOrder;
import eshop.wipro.entity.User;

public interface CartService {
    Cart getCart(User user);

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

    void delete(String itemId, User user);
    void checkout(User user);
}
```

discount

```
package eshop.wipro.service;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;

import eshop.wipro.entity.Discount;

public interface DiscountService {

    Discount createCoupon(String code);

    Page<Discount> findAll(PageRequest request);

    void deleteCoupon(String code);

    List<Discount> findAll();

}
```

Order

```
impopackage eshop.wipro.service;
rt java.util.List;

import eshop.wipro.entity.ProductCategory;

public interface CategoryService {
    List<ProductCategory> findAll();

    ProductCategory findByCategoryType(Integer categoryType);

    List<ProductCategory> findByCategoryTypeIn(List<Integer>
categoryTypeList);

    ProductCategory save(ProductCategory productCategory);

}
```

cart

```
package eshop.wipro.service;

import java.util.Collection;

import eshop.wipro.entity.Cart;
import eshop.wipro.entity.ProductInOrder;
import eshop.wipro.entity.User;

public interface CartService {
    Cart getCart(User user);

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

    void delete(String itemId, User user);

    void checkout(User user);
}
```

discount

```
package eshop.wipro.service;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import eshop.wipro.entity.Discount;

public interface DiscountService {

    Discount createCoupon(String code);

    Page<Discount> findAll(PageRequest request);

    void deleteCoupon(String code);

    List<Discount> findAll();

}
```

Item form

```
package eshop.wipro.form;

import lombok.Data;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;

@Data
public class ItemForm {
    @Min(value = 1)
    private Integer quantity;
    @NotEmpty
    private String productId;
    public Integer getQuantity() {
        return quantity;
    }
    public void setQuantity(Integer quantity) {
        this.quantity = quantity;
    }
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
}
```

Api

Controller

Cart controller

```
package eshop.wipro.api;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.bind.annotation.*;

import eshop.wipro.entity.Cart;
import eshop.wipro.entity.ProductInOrder;
import eshop.wipro.entity.User;
import eshop.wipro.form.ItemForm;
import eshop.wipro.repository.ProductInOrderRepository;
import eshop.wipro.service.CartService;
import eshop.wipro.service.ProductInOrderService;
import eshop.wipro.service.ProductService;
import eshop.wipro.service.UserService;

import java.security.Principal;
import java.util.Collection;
import java.util.Collections;

/**
 * Created By Zhu Lin on 3/11/2018.
 */
@CrossOrigin
@RestController
@RequestMapping("/cart")
public class CartController {
    @Autowired
    CartService cartService;
    @Autowired
    UserService userService;
    @Autowired
    ProductService productService;
    @Autowired
    ProductInOrderService productInOrderService;
    @Autowired
    ProductInOrderRepository productInOrderRepository;

    @PostMapping("")
    public ResponseEntity<Cart> mergeCart(@RequestBody 
Collection<ProductInOrder> productInOrders, Principal principal) {
        User user = userService.findOne(principal.getName());
        try {
            cartService.mergeLocalCart(productInOrders, user);
        } catch (Exception e) {
            ResponseEntity.badRequest().body("Merge Cart Failed");
        }
        return ResponseEntity.ok(cartService.getCart(user));
    }

    @GetMapping("")
    public Cart getCart(Principal principal) {
        User user = userService.findOne(principal.getName());
        return cartService.getCart(user);
    }

    @PostMapping("/add")
    public boolean addToCart(@RequestBody ItemForm form, Principal 
principal) {
        var productInfo = productService.findOne(form.getProductId());
        try {
            mergeCart(Collections.singleton(new ProductInOrder(productInfo, 
form.getQuantity())), principal);
        } catch (Exception e) {
            return false;
        }
        return true;
    }

    @PutMapping("/{itemId}")
    public ProductInOrder modifyItem(@PathVariable("itemId") String
itemId, @RequestBody Integer quantity, Principal principal) {
        User user = userService.findOne(principal.getName());
         productInOrderService.update(itemId, quantity, user);
        return productInOrderService.findOne(itemId, user);
    }

    @DeleteMapping("/{itemId}")
    public void deleteItem(@PathVariable("itemId") String itemId,
Principal principal) {
        User user = userService.findOne(principal.getName());
         cartService.delete(itemId, user);
         // flush memory into DB
    }

    @PostMapping("/checkout")
    public ResponseEntity checkout(Principal principal) {
        User user = userService.findOne(principal.getName());// Email as
username
        cartService.checkout(user);
        return ResponseEntity.ok(null);
    }

}
```

category Controllers

```
package eshop.wipro.api;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.web.bind.annotation.*;

import eshop.wipro.entity.ProductCategory;
import eshop.wipro.entity.ProductInfo;
import eshop.wipro.service.CategoryService;
import eshop.wipro.service.ProductService;
import eshop.wipro.vo.response.CategoryPage;

/**
```

```
 * Created By Zhu Lin on 3/10/2018.
 */
@RestController
@CrossOrigin
public class CategoryController {
    @Autowired
    CategoryService categoryService;
    @Autowired
    ProductService productService;

    /**
     * Show products in category
     *
     * @param categoryType
     * @param page
     * @param size
     * @return
     */
    @GetMapping("/category/{type}")
    public CategoryPage showOne(@PathVariable("type") Integer
categoryType,
                                @RequestParam(value = "page", defaultValue = "1")
Integer page,
                                @RequestParam(value = "size", defaultValue = "3")
Integer size) {

        ProductCategory cat =
categoryService.findByCategoryType(categoryType);
        PageRequest request = PageRequest.of(page - 1, size);
        Page<ProductInfo> productInCategory =
productService.findAllInCategory(categoryType, request);
        var tmp = new CategoryPage("", productInCategory);
        tmp.setCategory(cat.getCategoryName());
        return tmp;
    }
}
```

Csv Controllers

```
package eshop.wipro.api;
```

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import eshop.wipro.entity.ProductInfo;
import eshop.wipro.service.impl.CSVService;
import eshop.wipro.vo.helper.CSVHelper;
import eshop.wipro.vo.helper.ResponseMessage;

@CrossOrigin
@Controller
@RequestMapping("/csv")
public class CSVController {

    @Autowired
    CSVService fileService;

    @PostMapping("/upload")
    public ResponseEntity<ResponseMessage>
uploadFile(@RequestParam("file") MultipartFile file) {
        String message = "";
        if (CSVHelper.hasCSVFormat(file)) {
            try {
                fileService.save(file);
                message = "Uploaded the file successfully: " +
file.getOriginalFilename();
                return ResponseEntity.status(HttpStatus.OK).body(new
ResponseMessage(message));
            } catch (Exception e) {
                message = "Could not upload the file: " +
file.getOriginalFilename() + "!";
                return
ResponseEntity.status(HttpStatus.EXPECTATION_FAILED).body(new
ResponseMessage(message));
```

```
          }
       }
       message = "Please upload a csv file!";
       return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new
ResponseMessage(message));
    }

    @GetMapping("/tutorials")
    public ResponseEntity<List<ProductInfo>> getAllTutorials() {
       try {
          List<ProductInfo> tutorials = fileService.getAllTutorials();
          if (tutorials.isEmpty()) {
             return new ResponseEntity<>(HttpStatus.NO_CONTENT);
          }
          return new ResponseEntity<>(tutorials, HttpStatus.OK);
       } catch (Exception e) {
          return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
       }
    }

}
```

discount. controller

```
package eshop.wipro.api;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import eshop.wipro.entity.Discount;
import eshop.wipro.service.DiscountService;

@CrossOrigin
@RestController
public class DiscountController {

    @Autowired
    DiscountService discountService;

    @PostMapping("/add/coupon/{code}")
    public ResponseEntity<Discount>
createCoupon(@PathVariable("code") String code) {

        return ResponseEntity.ok(discountService.createCoupon(code));
    }

    @GetMapping("/coupon/list")
    public Page<Discount> orderList(@RequestParam(value = "page",
defaultValue = "1") Integer page,
                                    @RequestParam(value = "size", defaultValue =
"10") Integer size,
                                    Authentication authentication) {
        PageRequest request = PageRequest.of(page - 1, size);
        Page<Discount> discountPage;
        discountPage = discountService.findAll(request);
        return discountPage;
    }

    @GetMapping("/coupon/alllist")
    public List<Discount> orderList() {




        return discountService.findAll();
    }

    @PostMapping("/delete/coupon/{code}")
    public ResponseEntity<Discount>
deleteCoupon(@PathVariable("code") String code) {
        discountService.deleteCoupon(code);
        return ResponseEntity.ok(null);
    }


}
```

Email. Controller

```
package eshop.wipro.api;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import eshop.wipro.entity.EmailDetails;
import eshop.wipro.service.impl.EmailService;

@RestController
public class EmailController {

    @Autowired private EmailService emailService;

    // Sending a simple Email
    @PostMapping("/sendMail")
    public String
    sendMail(@RequestBody EmailDetails details)
    {
        String status
            = emailService.sendSimpleMail(details);

        return status;
    }

    // Sending email with attachment
    @PostMapping("/sendMailWithAttachment")
    public String sendMailWithAttachment(
        @RequestBody EmailDetails details)
    {
        String status
            = emailService.sendMailWithAttachment(details);
        return status;
    }

}
```

order Controller

```
package eshop.wipro.api;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.web.bind.annotation.*;

import eshop.wipro.entity.OrderMain;
import eshop.wipro.entity.ProductInOrder;
import eshop.wipro.service.OrderService;
import eshop.wipro.service.UserService;

import java.util.Collection;

/**
 * Created By Zhu Lin on 3/14/2018.
 */
@RestController
@CrossOrigin
public class OrderController {
    @Autowired
    OrderService orderService;
    @Autowired
    UserService userService;

    @GetMapping("/order")
    public Page<OrderMain> orderList(@RequestParam(value = "page",
defaultValue = "1") Integer page,
                               @RequestParam(value = "size", defaultValue =
"10") Integer size,
                               Authentication authentication) {
        PageRequest request = PageRequest.of(page - 1, size);
        Page<OrderMain> orderPage;
        if (authentication.getAuthorities().contains(new
SimpleGrantedAuthority("ROLE_CUSTOMER"))) {
            orderPage =
orderService.findByBuyerEmail(authentication.getName(), request);
        } else {
            orderPage = orderService.findAll(request);
        }
        return orderPage;
    }

    @PatchMapping("/order/cancel/{id}")
    public ResponseEntity<OrderMain> cancel(@PathVariable("id") Long
orderId, Authentication authentication) {
        OrderMain orderMain = orderService.findOne(orderId);
        if (!authentication.getName().equals(orderMain.getBuyerEmail())
&& authentication.getAuthorities().contains(new
SimpleGrantedAuthority("ROLE_CUSTOMER"))) {

            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }
        return ResponseEntity.ok(orderService.cancel(orderId));
    }

    @PatchMapping("/order/finish/{id}")
    public ResponseEntity<OrderMain> finish(@PathVariable("id") Long
orderId, Authentication authentication) {
        if (authentication.getAuthorities().contains(new
SimpleGrantedAuthority("ROLE_CUSTOMER"))) {
            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }
        return ResponseEntity.ok(orderService.finish(orderId));
    }

    @GetMapping("/order/{id}")
    public ResponseEntity show(@PathVariable("id") Long orderId,
Authentication authentication) {
        boolean isCustomer = authentication.getAuthorities().contains(new
SimpleGrantedAuthority("ROLE_CUSTOMER"));
        OrderMain orderMain = orderService.findOne(orderId);
        if (isCustomer
&& !authentication.getName().equals(orderMain.getBuyerEmail())) {
            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }

        Collection<ProductInOrder> items = orderMain.getProducts();
        return ResponseEntity.ok(orderMain);
    }
}
```

product Controller

```
package eshop.wipro.api;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import eshop.wipro.entity.ProductInfo;
import eshop.wipro.service.CategoryService;
import eshop.wipro.service.ProductService;
import eshop.wipro.vo.response.ProductInfoResponse;

/**
 * Created By Zhu Lin on 3/10/2018.
 */
@CrossOrigin
@RestController
public class ProductController {
    @Autowired
    CategoryService categoryService;
    @Autowired
    ProductService productService;

    /**
     * Show All Categories
     */

    @GetMapping("/product")
    public Page<ProductInfo> findAll(@RequestParam(value = "page",
defaultValue = "1") Integer page,
                                     @RequestParam(value = "size", defaultValue =
"3") Integer size) {
        PageRequest request = PageRequest.of(page - 1, size);
        return productService.findAll(request);
    }

    @GetMapping("/productall")
    public ProductInfoResponse findAll() {
        ProductInfoResponse response=new ProductInfoResponse();
        response.setProductList(productService.findAll());
        return response;
    }

    @GetMapping("/product/{productId}")
    public ProductInfo showOne(@PathVariable("productId") String
productId) {

        ProductInfo productInfo = productService.findOne(productId);

//        // Product is not available
//       if
(productInfo.getProductStatus().equals(ProductStatusEnum.DOWN.getC
ode())) {
//           productInfo = null;
//       }

        return productInfo;
    }

    @PostMapping("/seller/product/new")
    public ResponseEntity create(@Valid @RequestBody ProductInfo
product,
                                 BindingResult bindingResult) {
        ProductInfo productIdExists =
productService.findOne(product.getProductId());
        if (productIdExists != null) {
            bindingResult
                    .rejectValue("productId", "error.product",
                            "There is already a product with the code provided");
        }
        if (bindingResult.hasErrors()) {
            return ResponseEntity.badRequest().body(bindingResult);
        }
        return ResponseEntity.ok(productService.save(product));
    }

    @PutMapping("/seller/product/{id}/edit")
    public ResponseEntity edit(@PathVariable("id") String productId,
                               @Valid @RequestBody ProductInfo product,
                               BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return ResponseEntity.badRequest().body(bindingResult);
        }
        if (!productId.equals(product.getProductId())) {
            return ResponseEntity.badRequest().body("Id Not Matched");
        }

        return ResponseEntity.ok(productService.update(product));
    }

    @DeleteMapping("/seller/product/{id}/delete")
    public ResponseEntity delete(@PathVariable("id") String productId) {
        productService.delete(productId);
        return ResponseEntity.ok().build();
    }

}
```

Front-End

Pages

Admin

```
<h1 align="center" class="display-4 ">Users</h1>


   Search <input type = "text" name="search" [(ngModel)]="searchText" placeholder
="Enter Some Text To Search" />

<table id="table" class="table table-striped text-center" style="width:100%;">
   <thead>
   <tr>
      <th scope="col">Email</th>
      <th scope="col">Name</th>
      <th scope="col">Role</th>
      <th scope="col">Action</th>

   </tr>
   </thead>
   <!--Search by
   <select ng-model="columns" ng-options="e for e in headers">
      <option value=""></option>
   </select>
   Search <input type = "text" ng-model ="Search[columns]" placeholder ="Enter
Some Text To Search" />-->
   <tbody>
   <tr *ngFor="let user of page?.content | filter:searchText">

      <td class="align-middle">{{user.email}}</td>
      <td class="align-middle">{{user.name}}</td>
      <td class="align-middle">{{user.role}}</td>
      <td class="align-middle">
         <button  *ngIf="user.role=='ROLE_CUSTOMER'"
(click)=addAdmin(user.email)  class="btn btn-success">Add Admin</button>
```

```
          <button  *ngIf="user.role=='ROLE_MANAGER'"
(click)=removeAdmin(user.email)  class="btn btn-success">Remove Admin</button>
          <!-- <a  [ngClass]="{'isDisabled':'(productInfo.productStock >10)?
true:false'}"style="display: block" class="isDisabled"  routerLink="/email">Email</a>-
->
       </td>

    </tbody>
</table>
<app-pagination [currentPage]="page"></app-pagination>
```

admin .ts

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { AdminuserComponent } from './adminuser.component';

describe('AdminuserComponent', () => {
  let component: AdminuserComponent;
  let fixture: ComponentFixture<AdminuserComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ AdminuserComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(AdminuserComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<h1 align="center" class="display-4 mb-5">Order Detail</h1>
```

```
<table class="table table-striped text-center">
   <thead>
   <tr>
      <th scope="col">Photo</th>
      <th scope="col">Name</th>
      <th scope="col">Description</th>
      <th scope="col">Price</th>
      <th scope="col">Quantity</th>
      <th scope="col">Subtotal</th>

   </tr>
   </thead>
   <tbody>
   <tr *ngFor="let item of (order$ | async)?.products">
      <th class="align-middle" scope="row">
         <a routerLink="/seller/product/{{item.productId}}/edit"><img height="100px"
src="{{item.productIcon}}"
                                                        alt="{{item.productName}}"></a>
      </th>
      <td class="align-middle"><a
routerLink="/seller/product/{{item.productId}}/edit">{{item.productName}}</a></td>
      <td class="align-middle">{{item.productDescription}}</td>
      <td class="align-middle">{{item.productPrice | currency}}</td>
      <td class="align-middle">{{item.count}}</td>
      <td class="align-middle">{{(item.productPrice * item.count | currency)}}</td>
   </tr>
   </tbody>
</table>
<h5 style="display: inline" class="float-right">Total: {{(order$ | async)?.orderAmount
| currency}}</h5>
```

order.html

```
<h1 align="center" class="display-4 mb-5">Orders</h1>

<table class="table table-striped text-center">
   <thead>
   <tr>
      <th scope="col">Order #</th>
      <th scope="col">Customer Name</th>
      <th scope="col">Customer Email</th>
      <th scope="col">Customer phone</th>
      <th scope="col">Shipping Address</th>
      <th scope="col">Total</th>
      <th scope="col">Order Data</th>
```

```
        <th scope="col">Status</th>
        <th scope="col">Action</th>
    </tr>
    </thead>
    <tbody>

    <tr *ngFor="let order of page?.content">
        <th class="align-middle" scope="row">
            {{order.orderId}}
        </th>
        <td class="align-middle">{{order.buyerName}}</td>
        <td class="align-middle">{{order.buyerEmail}}</td>
        <td class="align-middle">{{order.buyerPhone}}</td>
        <td class="align-middle">{{order.buyerAddress}}</td>
        <td class="align-middle">{{order.orderAmount | currency}}</td>
        <td class="align-middle">{{order.createTime | date}}</td>
        <td class="align-middle">{{OrderStatus[order.orderStatus]}}</td>
        <td class="align-middle">
            <a *ngIf="!(currentUser.role == Role.Customer && currentUser.name ==
order.buyerEmail)"
               style="display:
            block" href="" routerLink="/order/{{order.orderId}}">
                Show</a>
            <a *ngIf="order.orderStatus == 0" style="display: block"
(click)="cancel(order)" routerLink="./">Cancel</a>
            <a *ngIf="currentUser.role != Role.Customer && order.orderStatus == 0"
               style="display: block"
               (click)="finish(order)"
               routerLink="./">
                Finish</a>
        </td>
    </tr>
    </tbody>
</table>

<app-pagination [currentPage]="page"></app-pagination>
```

order.ts

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {OrderService} from "../../services/order.service";
import {Order} from "../../models/Order";
import {OrderStatus} from "../../enum/OrderStatus";
import {UserService} from "../../services/user.service";
import {JwtResponse} from "../../response/JwtResponse";
import {Subscription} from "rxjs";
import {ActivatedRoute} from "@angular/router";
import {Role} from "../../enum/Role";
import { LoginComponent } from '../login/login.component';

@Component({
  selector: 'app-order',
  templateUrl: './order.component.html',
  styleUrls: ['./order.component.css']
})
export class OrderComponent implements OnInit, OnDestroy {

  page: any;
  OrderStatus = OrderStatus;
  currentUser: JwtResponse;
  Role = Role;
  constructor(private httpClient: HttpClient,
          private orderService: OrderService,
          private userService: UserService,
          private route: ActivatedRoute
  ) {
  }

  querySub: Subscription;

  ngOnInit() {
    this.currentUser = this.userService.currentUserValue;
    this.querySub = this.route.queryParams.subscribe(() => {
      this.update();
    });

  }

  update() {
    let nextPage = 1;
    let size = 10;
    if (this.route.snapshot.queryParamMap.get('page')) {
      nextPage = +this.route.snapshot.queryParamMap.get('page');
      size = +this.route.snapshot.queryParamMap.get('size');
    }
    this.orderService.getPage(nextPage, size).subscribe(page => this.page = page, _
=> {
      console.log("Get Orde Failed")
    });
    }

    cancel(order: Order) {
        this.orderService.cancel(order.orderId).subscribe(res => {
            if (res) {
                order.orderStatus = res.orderStatus;
            }
        });
    }

    finish(order: Order) {
        this.orderService.finish(order.orderId).subscribe(res => {
            if (res) {
                order.orderStatus = res.orderStatus;
            }
        })
    }

    ngOnDestroy(): void {
    }

}
```

LoginComponent.html

```
<h1 align="center" class="display-4 mb-5">Login</h1>
<div style="width:40%; margin: 25px auto">

    <div class="alert alert-danger" *ngIf="isInvalid">
        Incorrect username and password.
    </div>
    <div class="alert alert-info" *ngIf="isLogout">
        You have been logged out.
    </div>

    <form #form='ngForm' (ngSubmit)="onSubmit()">
        <div class="form-group">
            <label>Email address</label>
            <input type="text" class="form-control form-control-lg" id="email"
name="email" placeholder="Enter email"
                   required autofocus [(ngModel)]="model.username" #email="ngModel"
autocomplete="email" >
            <div [hidden]="email.valid || email.pristine" class="alert alert-danger">
                Email is required
```

```
          </div>
        </div>

        <div class="form-group">
          <label>Password</label>
          <input type="password" class="form-control form-control-lg" id="password"
name="password"  autocomplete="password"
                 placeholder="Password" required [(ngModel)]="model.password"
#password='ngModel'>
          <div [hidden]="password.valid || password.pristine" class="alert alert-
danger">
            Email is required
          </div>
        </div>

        <div class="form-group">
          <div>
            <input type="checkbox" id="remember_me" name="remember-me"
[(ngModel)]="model.remembered">
            <label for="remember_me" class="inline">Remember me</label>
            <a class="float-right" routerLink="/register">Sign Up</a>
          </div>
        </div>

        <div class="form-group">
          <button [disabled]="!form.form.valid" type="submit" class="btn btn-lg btn-
primary btn-block">Sign In</button>
        </div>
    </form>



</div>




<login class="ts">

    import {Component, OnInit} from '@angular/core';
import {UserService} from "../../services/user.service";
import {ActivatedRoute, Router} from "@angular/router";
import {Role} from "../../enum/Role";
import { DiscountComponent } from '../discount/discount.component';

@Component({
    selector: 'app-login',
    templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  isInvalid: boolean;
  isLogout: boolean;
  submitted = false;
  model: any = {
    username: '',
    password: '',
    remembered: false
  };

  returnUrl = '/';

  constructor(private userService: UserService,
              private router: Router,
              private route: ActivatedRoute) {
  }

  ngOnInit() {
    let params = this.route.snapshot.queryParamMap;
    this.isLogout = params.has('logout');
    this.returnUrl = params.get('returnUrl');
  }

  onSubmit() {
    this.submitted = true;
    this.userService.login(this.model).subscribe(
      user => {
        if (user) {
          if (user.role != Role.Customer) {

            this.returnUrl = '/seller';
          }

          this.router.navigateByUrl(this.returnUrl);
        } else {
          this.isLogout = false;
          this.isInvalid = true;
        }

      }
    );
  }

  fillLoginFields(u, p) {
    this.model.username = u;
    this.model.password = p;
    this.onSubmit();
  }
}
```

DiscountComponent.html
</login>

Order detail.html

```
<h1 align="center" class="display-4 mb-5">Orders</h1>

<table class="table table-striped text-center">
    <thead>
    <tr>
        <th scope="col">Order #</th>
        <th scope="col">Customer Name</th>
        <th scope="col">Customer Email</th>
        <th scope="col">Customer phone</th>
        <th scope="col">Shipping Address</th>
        <th scope="col">Total</th>
        <th scope="col">Order Data</th>
        <th scope="col">Status</th>
        <th scope="col">Action</th>
    </tr>
    </thead>
    <tbody>

    <tr *ngFor="let order of page?.content">
        <th class="align-middle" scope="row">
            {{order.orderId}}
        </th>
        <td class="align-middle">{{order.buyerName}}</td>
        <td class="align-middle">{{order.buyerEmail}}</td>
        <td class="align-middle">{{order.buyerPhone}}</td>
        <td class="align-middle">{{order.buyerAddress}}</td>
        <td class="align-middle">{{order.orderAmount | currency}}</td>
        <td class="align-middle">{{order.createTime | date}}</td>
        <td class="align-middle">{{OrderStatus[order.orderStatus]}}</td>
        <td class="align-middle">
            <a *ngIf="!(currentUser.role == Role.Customer && currentUser.name ==
order.buyerEmail)"
               style="display:
```

```
            block" href="" routerLink="/order/{{order.orderId}}">
                Show</a>
            <a *ngIf="order.orderStatus == 0" style="display: block"
(click)="cancel(order)" routerLink="./">Cancel</a>
            <a *ngIf="currentUser.role != Role.Customer && order.orderStatus == 0"
               style="display: block"
               (click)="finish(order)"
               routerLink="./">
                Finish</a>
        </td>
    </tr>
    </tbody>
</table>

<app-pagination [currentPage]="page"></app-pagination>
```

Orderdetail.ts

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {OrderService} from "../../services/order.service";
import {Order} from "../../models/Order";
import {OrderStatus} from "../../enum/OrderStatus";
import {UserService} from "../../services/user.service";
import {JwtResponse} from "../../response/JwtResponse";
import {Subscription} from "rxjs";
import {ActivatedRoute} from "@angular/router";
import {Role} from "../../enum/Role";
import { LoginComponent } from '../login/login.component';

@Component({
    selector: 'app-order',
    templateUrl: './order.component.html',
    styleUrls: ['./order.component.css']
})
export class OrderComponent implements OnInit, OnDestroy {

    page: any;
    OrderStatus = OrderStatus;
    currentUser: JwtResponse;
    Role = Role;
    constructor(private httpClient: HttpClient,
             private orderService: OrderService,
             private userService: UserService,
             private route: ActivatedRoute
    ) {
    }

    querySub: Subscription;

    ngOnInit() {
       this.currentUser = this.userService.currentUserValue;
       this.querySub = this.route.queryParams.subscribe(() => {
          this.update();
       });

    }

    update() {
       let nextPage = 1;
       let size = 10;
       if (this.route.snapshot.queryParamMap.get('page')) {
          nextPage = +this.route.snapshot.queryParamMap.get('page');
          size = +this.route.snapshot.queryParamMap.get('size');
       }
       this.orderService.getPage(nextPage, size).subscribe(page => this.page = page, _
=> {
          console.log("Get Orde Failed")
       });
    }

    cancel(order: Order) {
       this.orderService.cancel(order.orderId).subscribe(res => {
          if (res) {
             order.orderStatus = res.orderStatus;
          }
       });
    }

    finish(order: Order) {
       this.orderService.finish(order.orderId).subscribe(res => {
          if (res) {
             order.orderStatus = res.orderStatus;
          }
       })
    }

    ngOnDestroy(): void {
    }
```

```
}
```

LoginComponent.html

```
<h1 align="center" class="display-4 mb-5">Login</h1>
<div style="width:40%; margin: 25px auto">

   <div class="alert alert-danger" *ngIf="isInvalid">
      Incorrect username and password.
   </div>
   <div class="alert alert-info" *ngIf="isLogout">
      You have been logged out.
   </div>

   <form #form='ngForm' (ngSubmit)="onSubmit()">
      <div class="form-group">
         <label>Email address</label>
         <input type="text" class="form-control form-control-lg" id="email"
name="email" placeholder="Enter email"
              required autofocus [(ngModel)]="model.username" #email="ngModel"
autocomplete="email" >
         <div [hidden]="email.valid || email.pristine" class="alert alert-danger">
            Email is required
         </div>
      </div>

      <div class="form-group">
         <label>Password</label>
         <input type="password" class="form-control form-control-lg" id="password"
name="password"  autocomplete="password"
              placeholder="Password" required [(ngModel)]="model.password"
#password='ngModel'>
         <div [hidden]="password.valid || password.pristine" class="alert alert-
danger">
            Email is required
         </div>
      </div>

      <div class="form-group">
         <div>
            <input type="checkbox" id="remember_me" name="remember-me"
[(ngModel)]="model.remembered">
            <label for="remember_me" class="inline">Remember me</label>
            <a class="float-right" routerLink="/register">Sign Up</a>
        </div>
    </div>

    <div class="form-group">
        <button [disabled]="!form.form.valid" type="submit" class="btn btn-lg btn-
primary btn-block">Sign In</button>
    </div>
  </form>




</div>




<login class="ts">

  import {Component, OnInit} from '@angular/core';
import {UserService} from "../../services/user.service";
import {ActivatedRoute, Router} from "@angular/router";
import {Role} from "../../enum/Role";
import { DiscountComponent } from '../discount/discount.component';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  isInvalid: boolean;
  isLogout: boolean;
  submitted = false;
  model: any = {
    username: '',
    password: '',
    remembered: false
  };

  returnUrl = '/';

  constructor(private userService: UserService,
            private router: Router,
            private route: ActivatedRoute) {
  }
  ngOnInit() {
    let params = this.route.snapshot.queryParamMap;
    this.isLogout = params.has('logout');
    this.returnUrl = params.get('returnUrl');
  }

  onSubmit() {
    this.submitted = true;
    this.userService.login(this.model).subscribe(
      user => {
        if (user) {
          if (user.role != Role.Customer) {

            this.returnUrl = '/seller';
          }

          this.router.navigateByUrl(this.returnUrl);
        } else {
          this.isLogout = false;
          this.isInvalid = true;
        }

      }
    );
  }

  fillLoginFields(u, p) {
    this.model.username = u;
    this.model.password = p;
    this.onSubmit();
  }
}
```

DiscountComponent.html
</login>

Sign up.html

```
<h1 align="center" class="display-4 mb-5">Sign Up</h1>
<div style="width:40%; margin: 25px auto" >
 <form #form="ngForm" (ngSubmit)="onSubmit()">
  <div class="form-group">
   <label><b>Email address</b></label>
```

```
    <input [(ngModel)]="user.email" type="email" class="form-control form-control-
lg" id="email" name="email" placeholder="Enter email" email required
autofocus  #email="ngModel">
    <div  *ngIf="email.invalid && (email.dirty ||email.touched)" >
       <div *ngIf="email.errors.required" >
          Email is required.
       </div>
       <div *ngIf="email.errors.email">
          Invalid Email.
       </div>
    </div>
   </div>
   <div class="form-group">
    <label><b>Name</b></label>
    <input [(ngModel)]="user.name" type="text" class="form-control form-control-
lg" id="name" name="name" placeholder="Your name" required #name="ngModel">
      <div  *ngIf="name.invalid && (name.dirty ||name.touched)">
         <div *ngIf="name.errors.required">
            Name is required.
         </div>
         <div *ngIf="name.errors.minlength">
            Name must be at least 3 characters long.
         </div>
      </div>
   </div>
   <div class="form-group">
    <label><b>Password</b></label>
    <input  [(ngModel)]="user.password" type="password" class="form-control form-
control-lg" id="password" name="password" placeholder="Password"
minlength="3" required #password="ngModel">
      <div  *ngIf="password.invalid && (password.dirty ||password.touched)">
         <div *ngIf="password.errors.required">
            Password is required.
         </div>
         <div *ngIf="password.errors.minlength">
            Password must be at least 3 characters long.
         </div>
      </div>
   </div>
   <div class="form-group">
    <label><b>Phone</b></label>
    <input [(ngModel)]="user.phone" type="text" class="form-control form-control-
lg" id="phone" name="phone" placeholder="Phone" required #phone="ngModel" >
      <div  *ngIf="phone.invalid && (phone.dirty ||phone.touched)">
         <div *ngIf="phone.errors.required">
            Phone Number is required.
         </div>
```

```
        </div>
    </div>
    <div class="form-group">
      <label><b>Address</b></label>
      <input [(ngModel)]="user.address" type="text" class="form-control form-control-
lg" id="address" name="address" placeholder="Address" required
#address="ngModel">
        <div  *ngIf="address.invalid && (address.dirty ||address.touched)">
            <div *ngIf="address.errors.required">
                Address is required.
            </div>
        </div>
    </div>
    <div class="form-group">
      <button type="submit" class="btn btn-lg btn-primary btn-block"
[disabled]="!form.form.valid" >Sign Up</button>
    </div>
  </form>
</div>
```

sign up.ts

```
import {Component, OnInit} from '@angular/core';
import {Location} from '@angular/common';
import {User} from "../../models/User";
import {UserService} from "../../services/user.service";
import {Router} from "@angular/router";

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css']
})
export class SignupComponent implements OnInit {

  user: User;

  constructor( private location: Location,
              private userService: UserService,
              private router: Router) {
    this.user = new User();

  }
  ngOnInit() {

  }
  onSubmit() {
    this.userService.signUp(this.user).subscribe(u => {
      this.router.navigate(['/login']);
    },
        e => {});
  }

}
```

Productlist.html

```
<h1 align="center" class="display-4 ">Products</h1>
<a *ngIf="currentUser?.role == Role.Customer" style="color: inherit"
   routerLink="/seller/product/new" class="float-right mb-3"><i class="fas fa-plus fa-
2x">Add</i>
</a>
<button (click)="exportAsXLSX()" class="float-right mb-3">
    <i class="fa fa-download" aria-hidden="true" style="font-
size:42px;color:blue"></i></button>
    <form action="/" method="post" enctype="multipart/form-data">
<input class="form-control" type="file"
            (change)="onChange($event)">
  </form>
    <button (click)="onUpload()"
        class="btn btn-success">
        Upload
    </button>



  <!--  Search <input type = "text" name="search" [(ngModel)]="searchText"
placeholder ="Enter Some Text To Search" /> -->

<table id="table" class="table table-striped text-center" style="width:100%;">
    <thead>
    <tr>
        <th scope="col">Photo</th>
        <th scope="col">Code</th>
        <th scope="col">Name</th>
        <th scope="col">Type</th>
        <th scope="col">Description</th>
        <th scope="col">Price</th>
        <th scope="col">Stock</th>
        <th scope="col">Status</th>
        <th scope="col">Action</th>
        <th scopr="col">Mail</th>
    </tr>
    </thead>
    <!--Search by
    <select ng-model="columns" ng-options="e for e in headers">
        <option value=""></option>
    </select>
    Search <input type = "text" ng-model ="Search[columns]" placeholder ="Enter
Some Text To Search" />-->
    <tbody>
    <tr *ngFor="let productInfo of page?.content | filter:searchText">
        <th class="align-middle" scope="row">
            <img height="100px" src="{{productInfo.productIcon}}"
alt="{{productInfo.productName}}">
        </th>
        <td class="align-middle">{{productInfo.productId}}</td>
        <td class="align-middle">{{productInfo.productName}}</td>
        <td class="align-middle">{{CategoryType[productInfo.categoryType]}}</td>
        <td class="align-middle">{{productInfo.productDescription}}</td>
        <td class="align-middle">{{productInfo.productPrice | currency}}</td>
        <td class="align-middle">{{productInfo.productStock}}</td>
        <td class="align-middle">{{ProductStatus[productInfo.productStatus]}}</td>
        <td class="align-middle">
            <a style="display: block"
routerLink="/seller/product/{{productInfo.productId}}/edit">
                Edit</a>

            <a *ngIf="currentUser?.role == Role.Customer" style="display: block"
               (click)="remove(page.content, productInfo.productId)" routerLink="./">
                Remove</a>
        </td>
        <td class="align-middle">
            <button (click)=onSubmit() [disabled]="(productInfo.productStock >10)?
true:false" class="btn btn-success">email</button>
           <!-- <a  [ngClass]="{'isDisabled':'(productInfo.productStock >10)?
true:false'}"style="display: block" class="isDisabled"  routerLink="/email">Email</a>-
->
        </td>

    </tbody>
</table>
<app-pagination [currentPage]="page"></app-pagination>
```

productlist.ts

```
import {Component, Injectable, OnDestroy, OnInit} from '@angular/core';
import {UserService} from "../../services/user.service";
import {ProductService} from "../../services/product.service";
import {JwtResponse} from "../../response/JwtResponse";
import {Subscription} from "rxjs";
import {ActivatedRoute} from "@angular/router";
import {CategoryType} from "../../enum/CategoryType";
import {ProductStatus} from "../../enum/ProductStatus";
import {ProductInfo} from "../../models/productInfo";
import {Role} from "../../enum/Role";
import {ExcelService} from "../../services/ExcelService"
import { JsonpClientBackend } from '@angular/common/http';
import { Router } from '@angular/router';
@Injectable({
  providedIn: 'root'
})
@Component({
  selector: 'app-product.list',
  templateUrl: './product.list.component.html',
  styleUrls: ['./product.list.component.css']
})
export class ProductListComponent implements OnInit, OnDestroy {

  constructor(private userService: UserService,
          private productService: ProductService,
          private route: ActivatedRoute,
          private excelService: ExcelService,
          private router:Router) {
  }

  Role = Role;
  currentUser: JwtResponse;
  page: any;
  CategoryType = CategoryType;
  ProductStatus = ProductStatus;
  private querySub: Subscription;
  response:any;
  productInfo = [];
 // private router: Router;

  data : any[] ;
  // Variable to store shortLink from api response
  shortLink: string = "";
  loading: boolean = false; // Flag variable
  file: File = null; // Variable to store file
  ngOnInit() {
    this.querySub = this.route.queryParams.subscribe(() => {
      this.update();
    });
  }

  ngOnDestroy(): void {

  }

  update() {
    if (this.route.snapshot.queryParamMap.get('page')) {
      const currentPage = +this.route.snapshot.queryParamMap.get('page');
      const size = +this.route.snapshot.queryParamMap.get('size');
      this.getProds(currentPage, size);
    } else {
      this.getProds();
    }
  }

  getProds(page: number = 1, size: number = 5) {
    this.productService.getAllInPage(+page, +size)
      .subscribe(page => {
        this.page = page;
      });

  }

  remove(productInfos: ProductInfo[], productInfo) {
    this.productService.delelte(productInfo).subscribe(_ => {
        productInfos = productInfos.filter(e => e.productId != productInfo);
      },
      err => {
      });
  }

  exportAsXLSX():void{
      this.productService.getAll().subscribe(response => this.productInfo=
response);
      console.log(this.data);
      this.excelService.exportAsExcelFile(this.productInfo,"Report");
  }

  afuConfig = {
    uploadAPI: {
     url: "http://localhost:8080/api/csv/upload"
```

```
        }
    };

    // On file Select
    onChange(event) {
        this.file = event.target.files[0];
    }

    // OnClick of button Upload
    onUpload() {
        this.loading = !this.loading;
        console.log(this.file);
        this.productService.upload(this.file).subscribe(
            (event: any) => {
                if (typeof (event) === 'object') {

                    // Short link via api response
                    this.shortLink = event.link;

                    this.loading = false; // Flag variable
                }
            }
        );
    }

    onSubmit(){
        this.router.navigate(['/email']);
    }

}
```

Part

Pagination.html

```
<div class="col-md-12 column">
  <ul class="pagination justify-content-end">
      <li class="page-item" *ngIf="currentPage?.number > 0; else prev">
          <a
                class="page-link"
                [routerLink]="['./']"
                [queryParams]="{ page: currentPage?.number, size:
currentPage?.size }"
```

```
        >Previous</a
        >
    </li>
      <ng-template #prev>
        <li class="page-item  disabled"><a class="page-
link">Previous</a></li>
      </ng-template>

      <ng-container *ngFor="let item of counter(currentPage?.totalPages);
let i = index">
        <li class="page-item" *ngIf="currentPage &&
currentPage.number != i; else active">
          <a class="page-link "
                routerLink="./"
                [queryParams]="{ page: i + 1, size: currentPage?.size }"
          >{{ i + 1 }}</a
          >
        </li>
        <ng-template #active>
          <li
                class="page-item active "

          >
            <button class="page-link " disabled>{{ i + 1 }}</button>
          </li>
        </ng-template>
      </ng-container>

      <li
            class="page-item"
            *ngIf="currentPage?.number + 1 < currentPage?.totalPages; else
next"
      >
        <a
              class="page-link"
              [routerLink]="['./']"
              [queryParams]="{
        page: currentPage?.number + 2,
        size: currentPage?.size
       }"
```

```
          >Next</a
          >
    </li>
    <ng-template #next>
      <li class="page-item disabled "><a class="page-link">Next</a></li>
    </ng-template>
  </ul>
</div>
```

pagination.ts

```
import {Component, Input, OnInit} from '@angular/core';
import { navigationCancelingError } from '@angular/router/src/shared';

@Component({
  selector: 'app-pagination',
  templateUrl: './pagination.component.html',
  styleUrls: ['./pagination.component.css']
})
export class PaginationComponent implements OnInit {

  @Input() currentPage: any;

  constructor() {
  }

  ngOnInit() {
  }

  counter(i = 1) {
    return new Array(i);
  }
}
```

navigation.html

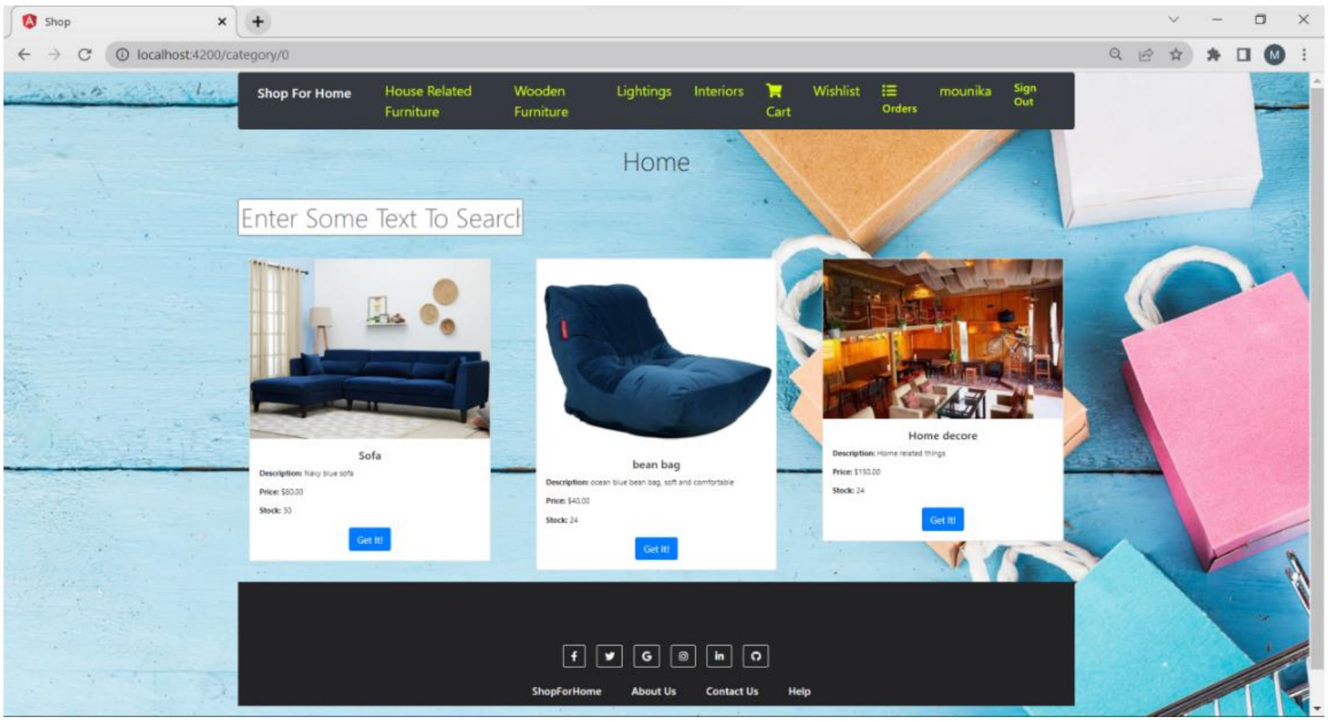
```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" [routerLink]="root">
    <img src="/assets/brand.png" width="30" height="30" class="d-
inline-block align-top" alt="">
    Shop_For_Home
  </a>

  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav"
        aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">

    <div class="navbar-nav" *ngIf="!currentUser || currentUser.role ==
Role.Customer">

        <a class="nav-item nav-link "
          routerLink="/category/0">
          House Related Furniture
        </a>
        <a class="nav-item nav-link"
          routerLink="/category/1">
          Wooden Furniture
        </a>
        <a class="nav-item nav-link "
          routerLink="/category/2">
          Lightings
        </a>
        <a class="nav-item nav-link"
          routerLink="/category/3">
          Interiors
        </a>



    </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>

      <div class="navbar-nav" *ngIf="currentUser && currentUser.role
== Role.Manager">

        <a class="nav-item nav-
link"  routerLink="/discount">Discount</a>
        <a class="nav-item nav-link" (click)=getUsers()
routerLink="/admin/user">Users</a>
        <a class="nav-item nav-link"
routerLink="/seller/product">Stocks</a>
      </div>

      <div class="navbar-nav" *ngIf="currentUser && currentUser.role
== Role.Customer">
        <a class="nav-item nav-link"  routerLink="/wishlist">Wishlist</a>
      </div>

      <ng-container *ngIf="currentUser; else noUser">

        <!-- <a class="nav-item nav-
link"  routerLink="/discount">Discount</a>
        <a class="nav-item nav-link" (click)=getUsers()
routerLink="/admin/user">Users</a>
        <a class="nav-item nav-link"
routerLink="/seller/product">Stocks</a> -->
        <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">

                Login
            </a>
            <a class="nav-item nav-link " routerLink="/register">
                Register
            </a>
        </ng-template>

        </div>
    </div>
</nav>
```

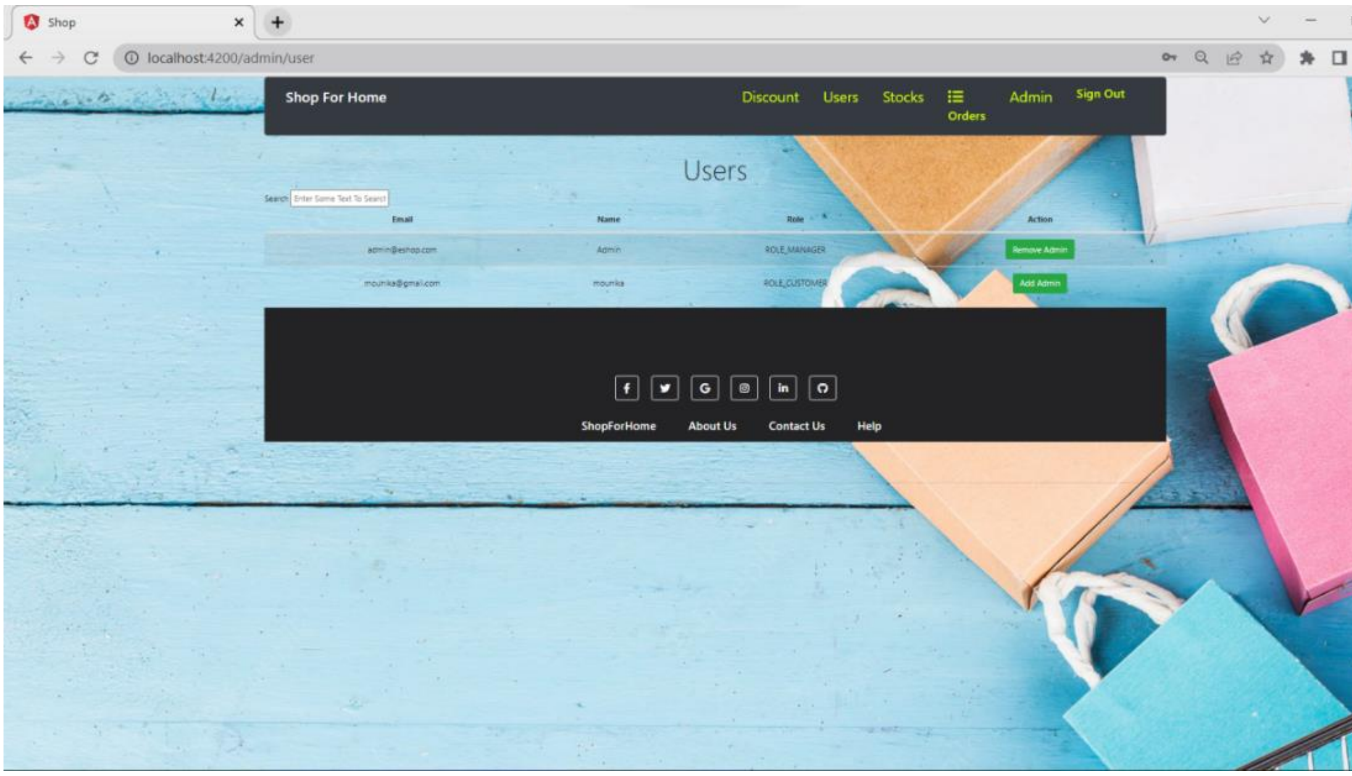
navigation.ts

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {UserService} from "../../services/user.service";
import {Subscription} from "rxjs";
import {JwtResponse} from "../../response/JwtResponse";
import {Router} from "@angular/router";
import {Role} from "../../enum/Role";
import { User } from 'src/app/models/User';

@Component({
    selector: 'app-navigation',
    templateUrl: './navigation.component.html',
    styleUrls: ['./navigation.component.css']
})
export class NavigationComponent implements OnInit, OnDestroy {
    currentUserSubscription: Subscription;
    name$;
    name: string;
    currentUser: JwtResponse;
    root = '/';
    Role = Role;
    userList : User[];

    constructor(private userService: UserService,
                private router: Router,

    ) {

    }

    ngOnInit() {
        this.name$ = this.userService.name$.subscribe(aName => this.name
= aName);
        this.currentUserSubscription =
this.userService.currentUser.subscribe(user => {
            this.currentUser = user;
            if (!user || user.role == Role.Customer) {
                this.root = '/';
            } else {
                this.root = '/seller';
            }
        });
    }

    ngOnDestroy(): void {
        this.currentUserSubscription.unsubscribe();
        // this.name$.unsubscribe();
    }

    logout() {
        this.userService.logout();
        // this.router.navigate(['/login'], {queryParams: {logout: 'true'}} );
    }

    getUsers(){
        this.userService.getUsers().subscribe(response => this.userList=
response);
    }


}
```

Home Page



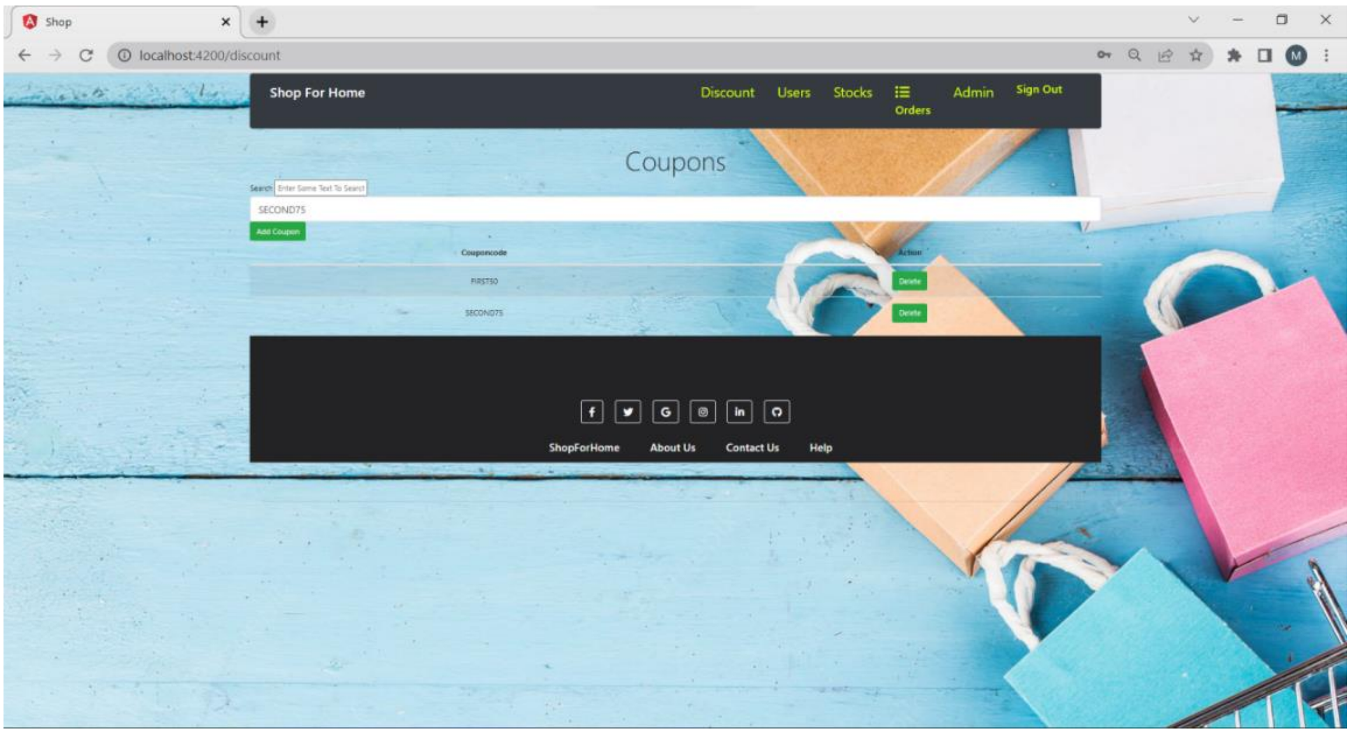
User data page



Users

Email	Name	Role	Action
admin@eshop.com	Admin	ROLE_MANAGER	Remove Admin
mounika@gmail.com	mounika	ROLE_CUSTOMER	Add Admin

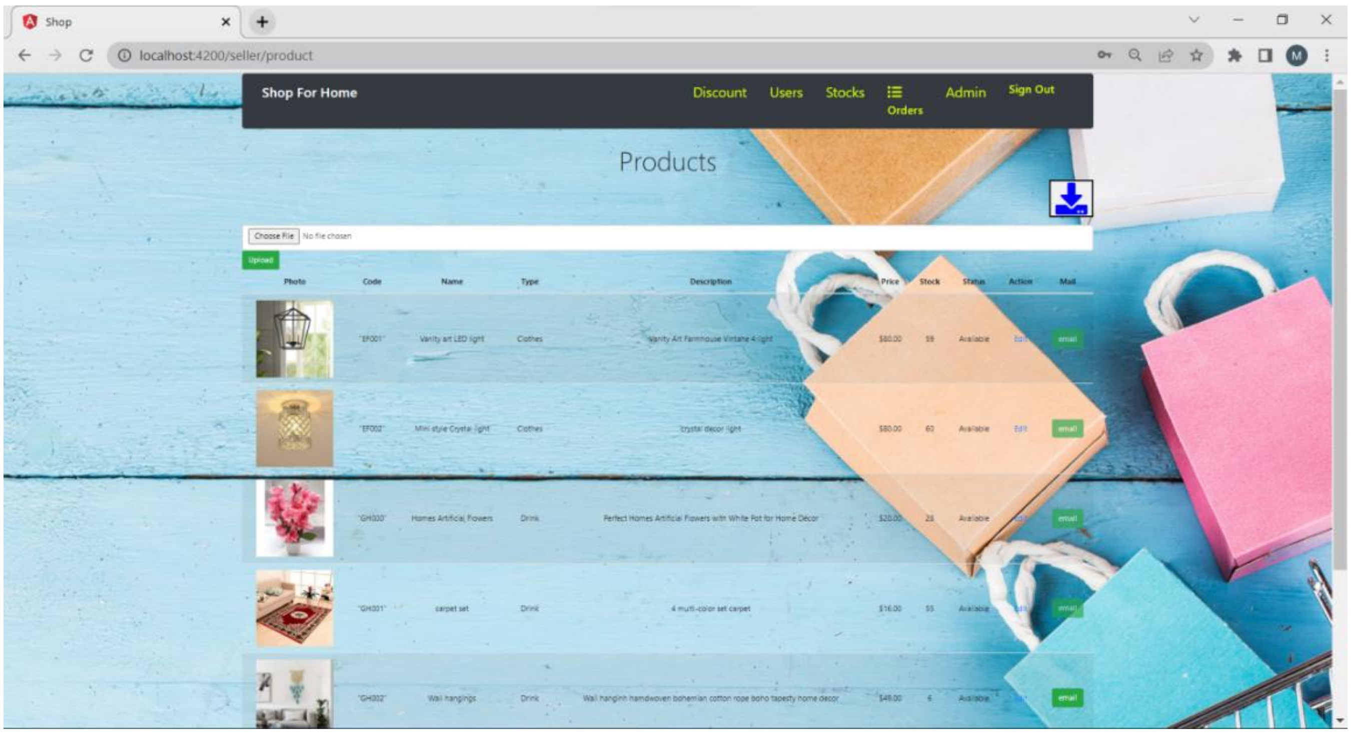
Coupons



Coupons

Couponcode	Action
FIRST50	Delete
SECOND75	Delete

Products



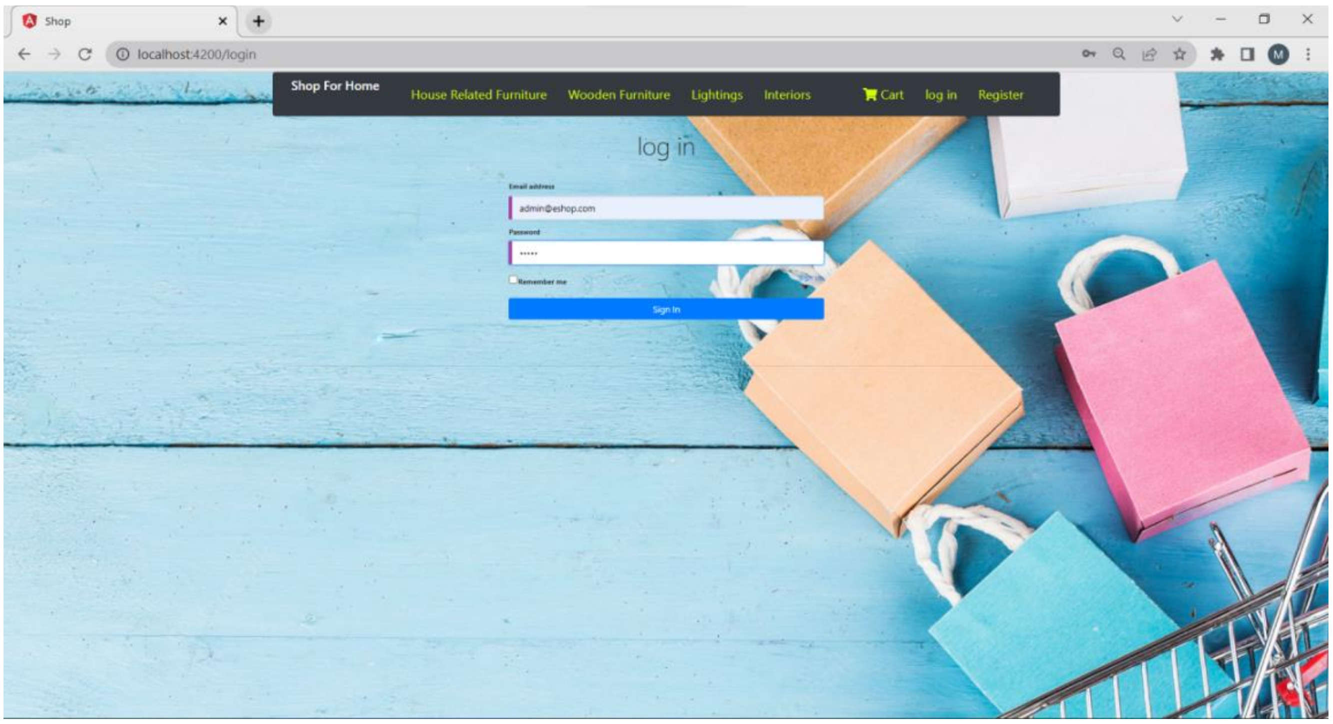
Products

Choose File No file chosen

Upload

Photo	Code	Name	Type	Description	Price	Stock	Status	Action	Mail
	"EF001"	Vanity art LED light	Clothes	Vanity Art Farmhouse Vintahe 4 light	$80.00	59	Available	Edit	email
	"EF002"	Mini style Crystal light	Clothes	crystal decor light	$80.00	60	Available	Edit	email
	"GH000"	Homes Artificial Flowers	Drink	Perfect Homes Artificial Flowers with White Pot for Home Decor	$20.00	28	Available	Edit	email
	"GH001"	carpet set	Drink	4 multi-color set carpet	$16.00	55	Available	Edit	email
	"GH002"	Wall hangings	Drink	Wall hanginn handwoven bohemian cotton rope boho tapesty home decor	$49.00	6	Available	Edit	email

Login page



log in

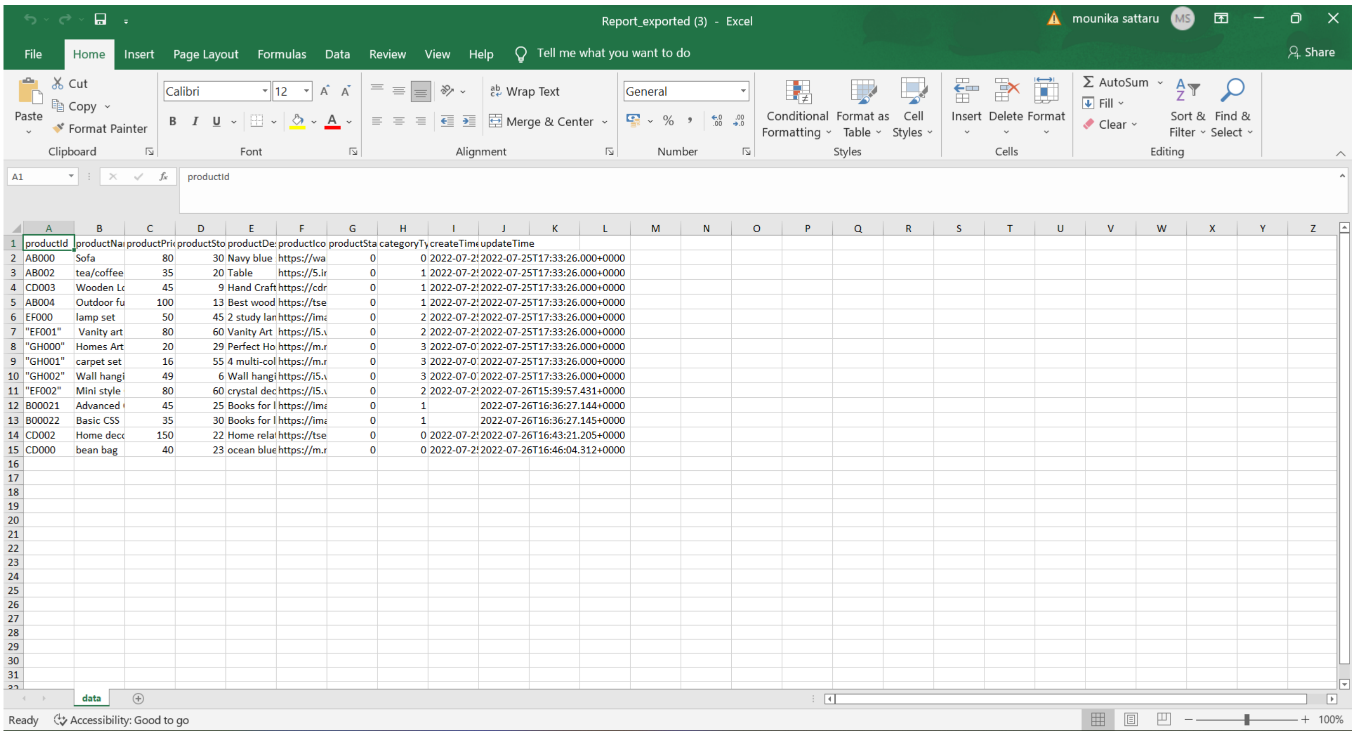
Email address: admin@eshop.com

Password: •••••

Remember me

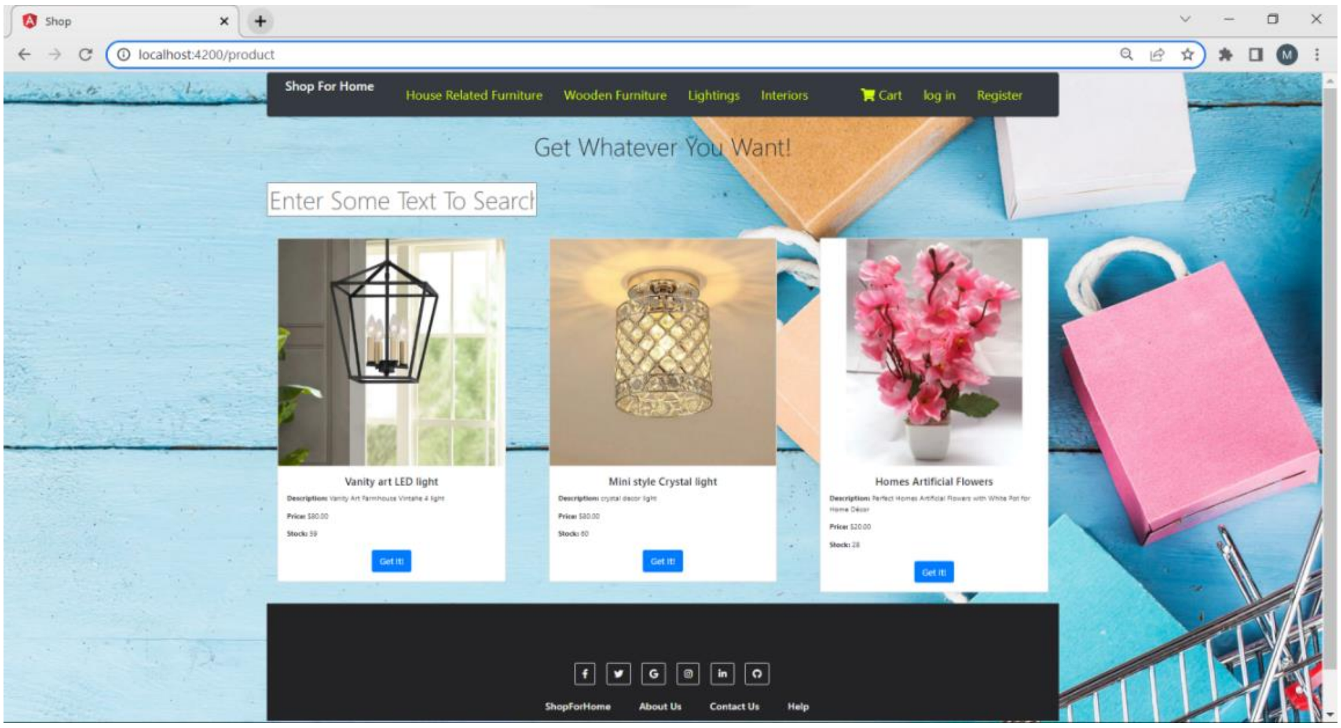
Sign In

Sales report data



	A	B	C	D	E	F	G	H	I	J
1	productId	productNa	productPri	productSto	productDe	productIco	productSta	categoryTy	createTime	updateTime
2	AB000	Sofa	80	30	Navy blue	https://wa	0	0	2022-07-2	2022-07-25T17:33:26.000+0000
3	AB002	tea/coffee	35	20	Table	https://5.ir	0	1	2022-07-2	2022-07-25T17:33:26.000+0000
4	CD003	Wooden Lo	45	9	Hand Craft	https://cdr	0	1	2022-07-2	2022-07-25T17:33:26.000+0000
5	AB004	Outdoor fu	100	13	Best wood	https://tse	0	1	2022-07-2	2022-07-25T17:33:26.000+0000
6	EF000	lamp set	50	45	2 study lan	https://ima	0	2	2022-07-2	2022-07-25T17:33:26.000+0000
7	"EF001"	Vanity art	80	60	Vanity Art	https://i5.v	0	2	2022-07-2	2022-07-25T17:33:26.000+0000
8	"GH000"	Homes Art	20	29	Perfect Ho	https://m.r	0	3	2022-07-0	2022-07-25T17:33:26.000+0000
9	"GH001"	carpet set	16	55	4 multi-col	https://m.r	0	3	2022-07-0	2022-07-25T17:33:26.000+0000
10	"GH002"	Wall hangi	49	6	Wall hangi	https://i5.v	0	3	2022-07-0	2022-07-25T17:33:26.000+0000
11	"EF002"	Mini style	80	60	crystal dec	https://i5.v	0	2	2022-07-2	2022-07-26T15:39:57.431+0000
12	B00021	Advanced	45	25	Books for l	https://ima	0	1		2022-07-26T16:36:27.144+0000
13	B00022	Basic CSS	35	30	Books for l	https://ima	0	1		2022-07-26T16:36:27.145+0000
14	CD002	Home deco	150	22	Home relat	https://tse	0	0	2022-07-2	2022-07-26T16:43:21.205+0000
15	CD000	bean bag	40	23	ocean blue	https://m.r	0	0	2022-07-2	2022-07-26T16:46:04.312+0000

Welcome page



Vanity art LED light

Description: Vanity Art Farmhouse Vintage 4 light

Price: $80.00

Stock: 59

Mini style Crystal light

Description: crystal decor light

Price: $80.00

Stock: 60

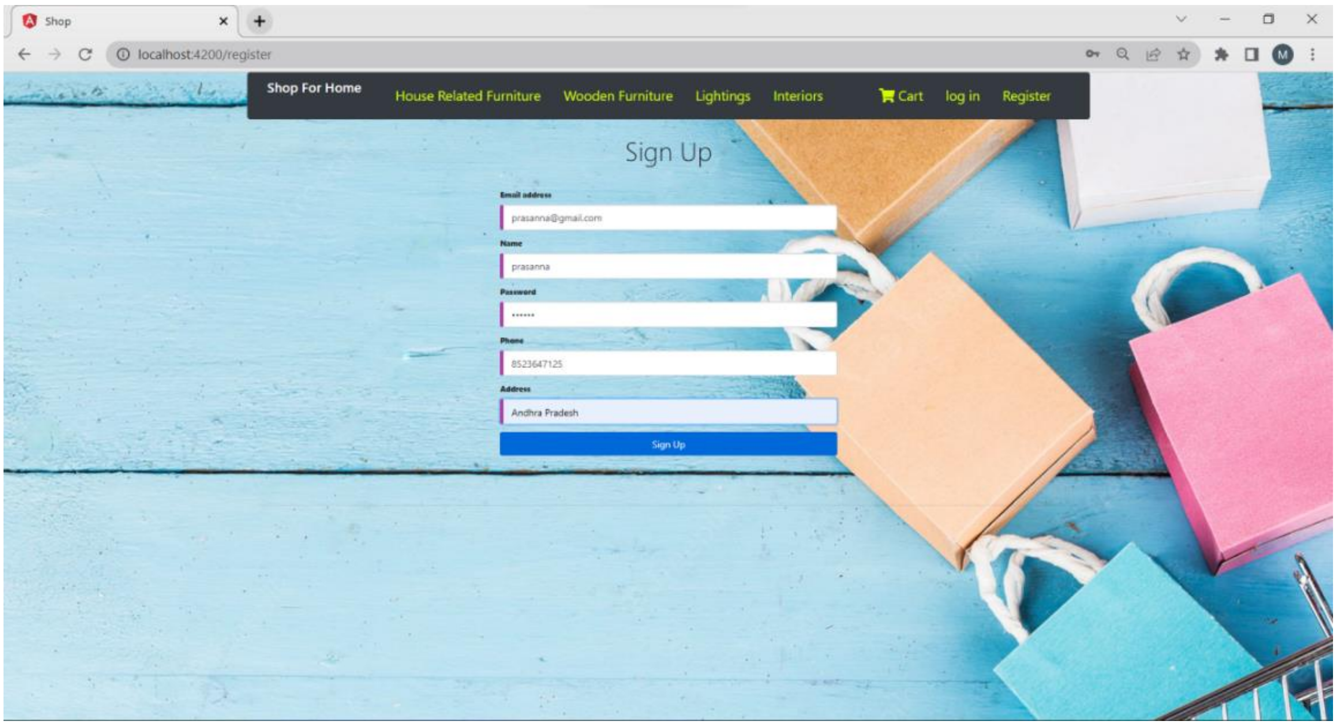
Homes Artificial Flowers

Description: Perfect Homes Artificial Flowers with White Pot for Home Décor

Price: $20.00

Stock: 28

Registration page



Sign Up

Email address
prasanna@gmail.com

Name
prasanna

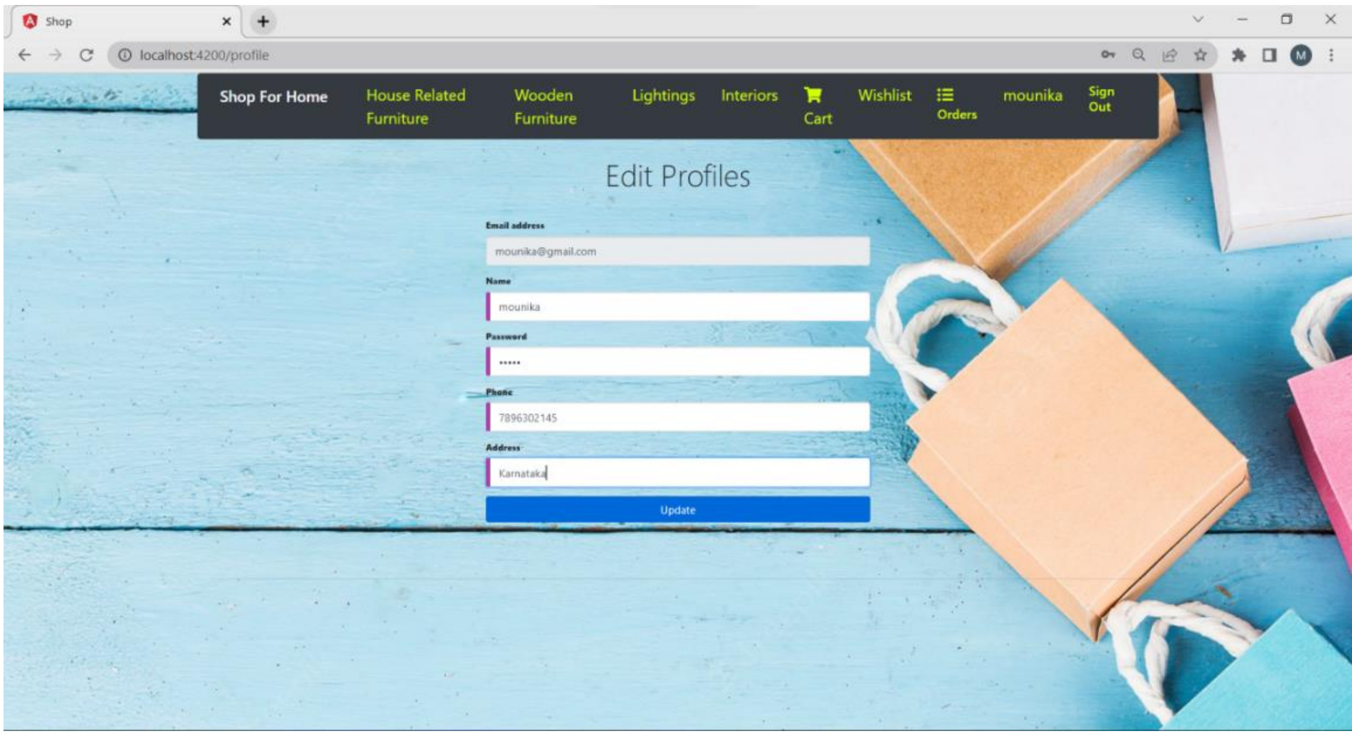
Password
••••••

Phone
8523647125

Address
Andhra Pradesh

Sign Up

Edit profile



Edit Profiles

Email address
mounika@gmail.com

Name
mounika

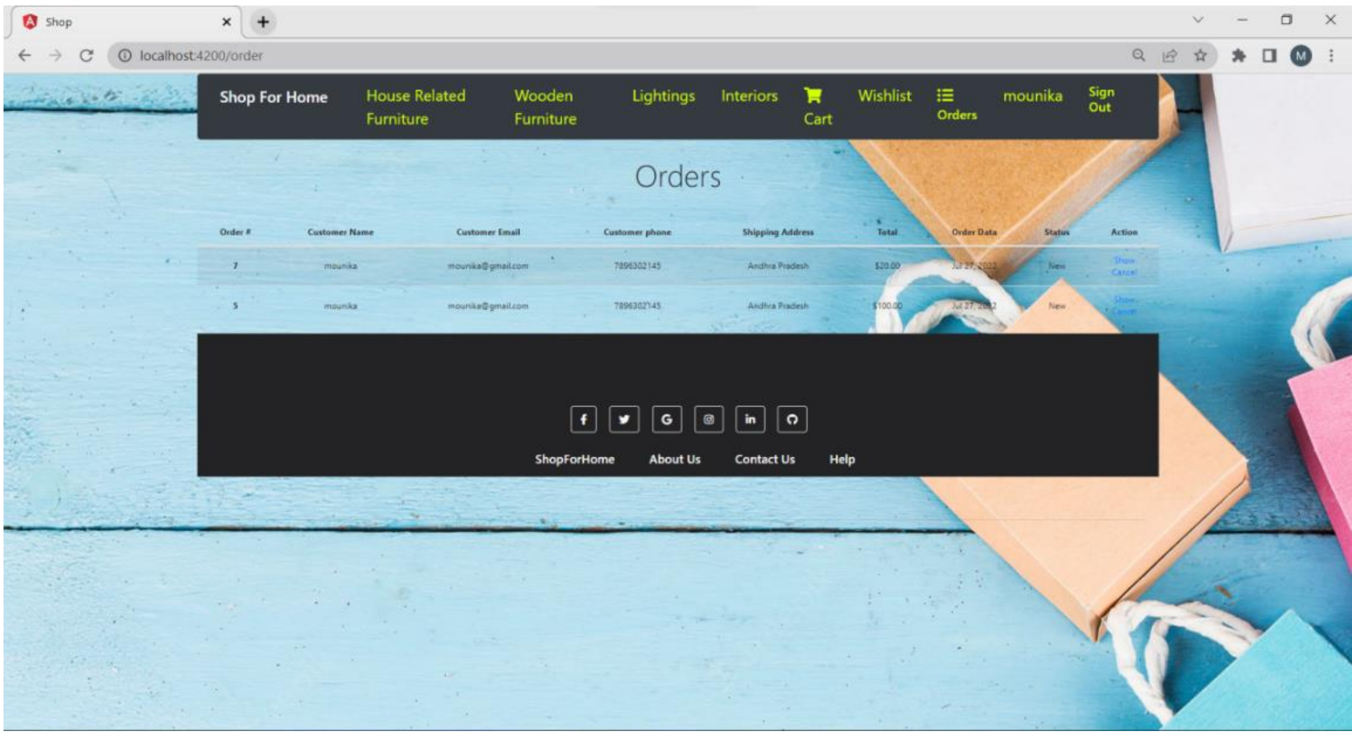
Password
•••••

Phone
7896302145

Address
Karnataka

Update

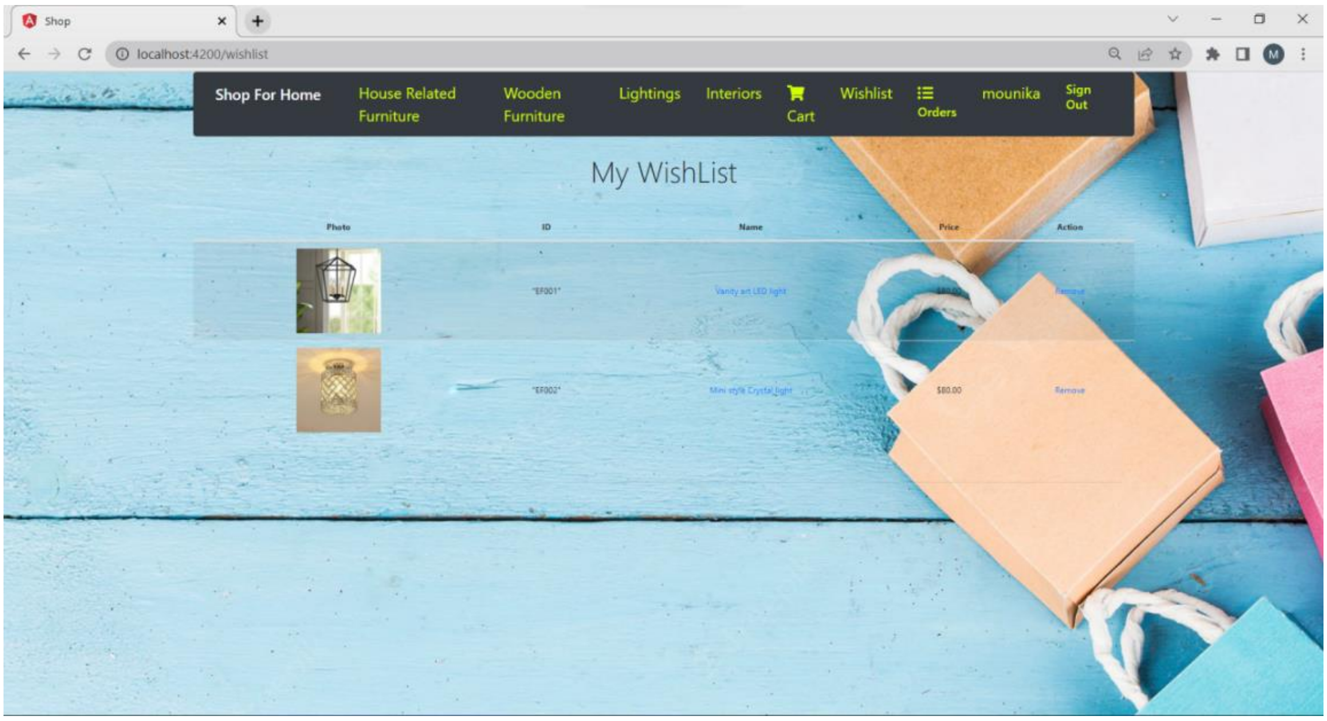
Order page



Orders

Order #	Customer Name	Customer Email	Customer phone	Shipping Address	Total	Order Data	Status	Action
7	mounika	mounika@gmail.com	7896302145	Andhra Pradesh	$20.00	Jul 27, 2022	New	Show Cancel
5	mounika	mounika@gmail.com	7896302145	Andhra Pradesh	$100.00	Jul 27, 2022	New	Show Cancel

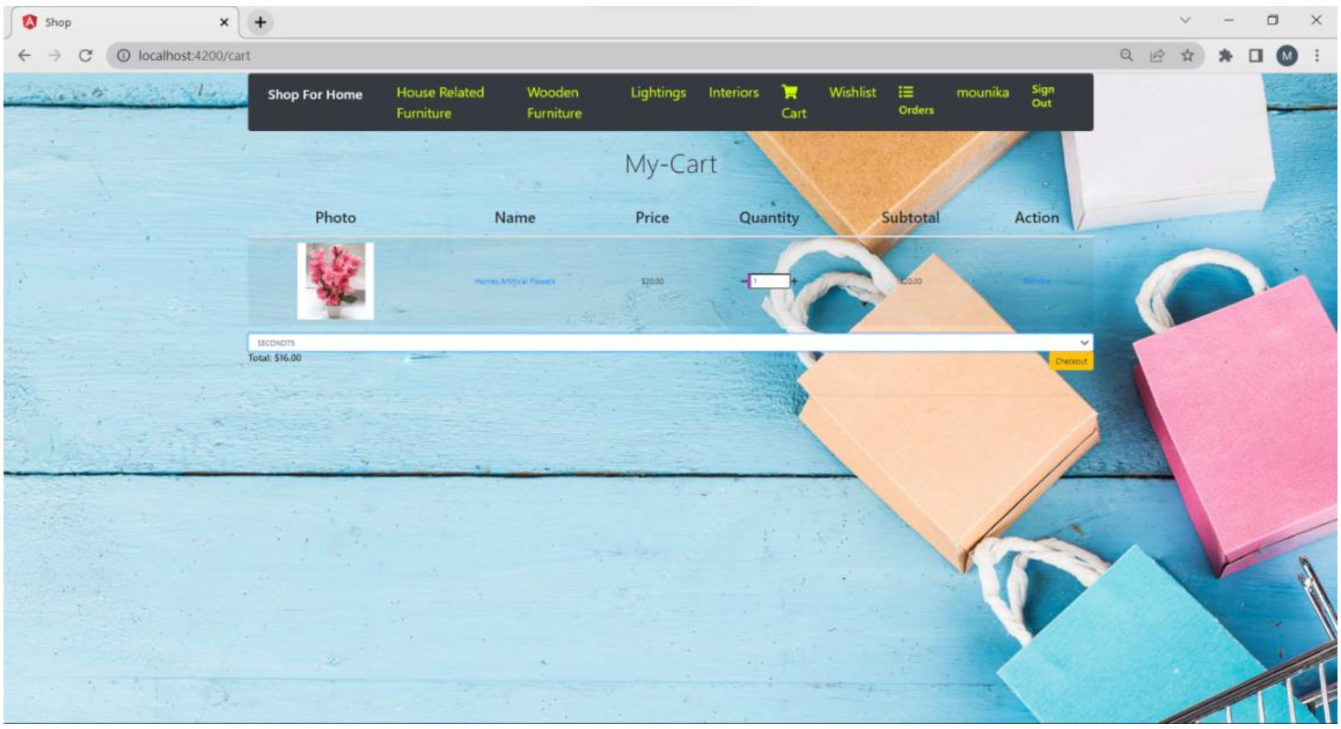
My Wishlist page



My WishList

Photo	ID	Name	Price	Action
	"EF001"	Vandy art LED light	$80.00	Remove
	"EF002"	Mini style Crystal light	$80.00	Remove

Cartpage



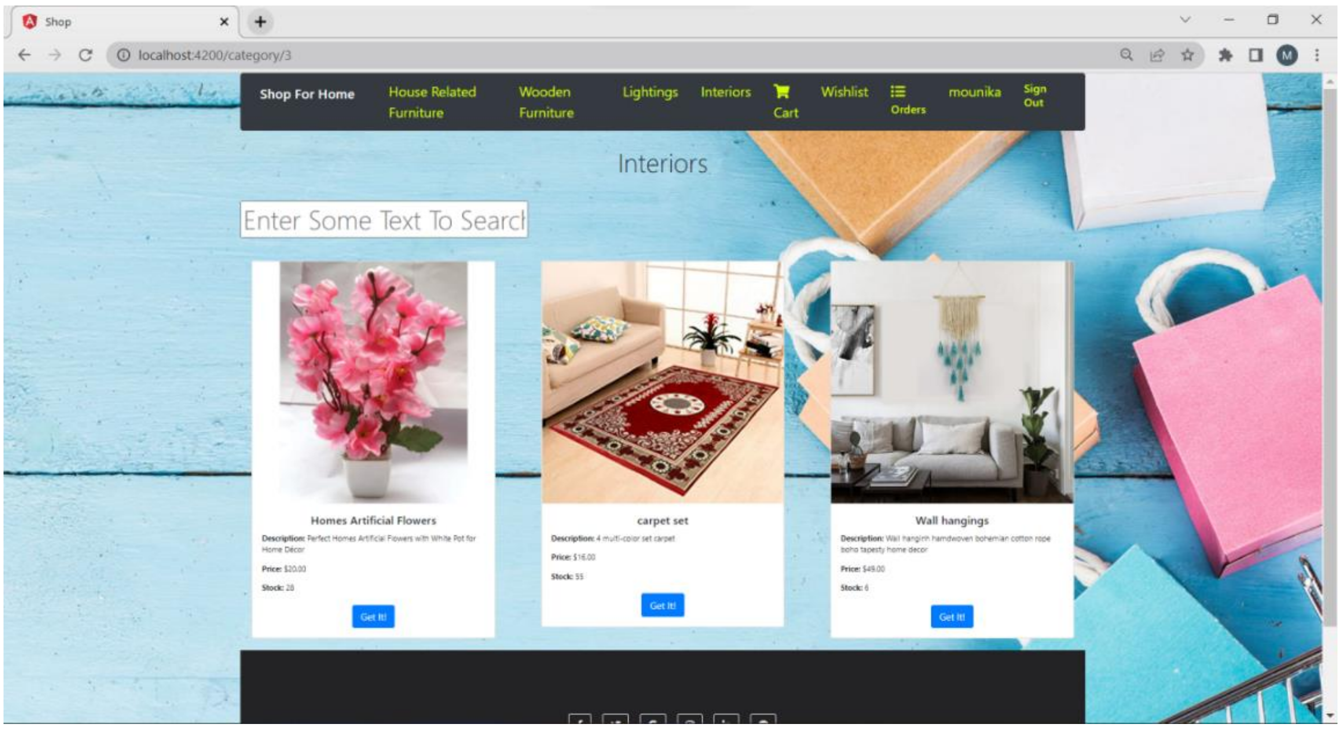
My-Cart

Photo	Name	Price	Quantity	Subtotal	Action
	Homes Artificial Flowers	$20.00	1	$20.00	Remove

SECONDTS

Total: $16.00

Interior page



Homes Artificial Flowers

Description: Perfect Homes Artificial Flowers with White Pot for Home Décor

Price: $20.00

Stock: 28

carpet set

Description: 4 multi-color set carpet

Price: $16.00

Stock: 55

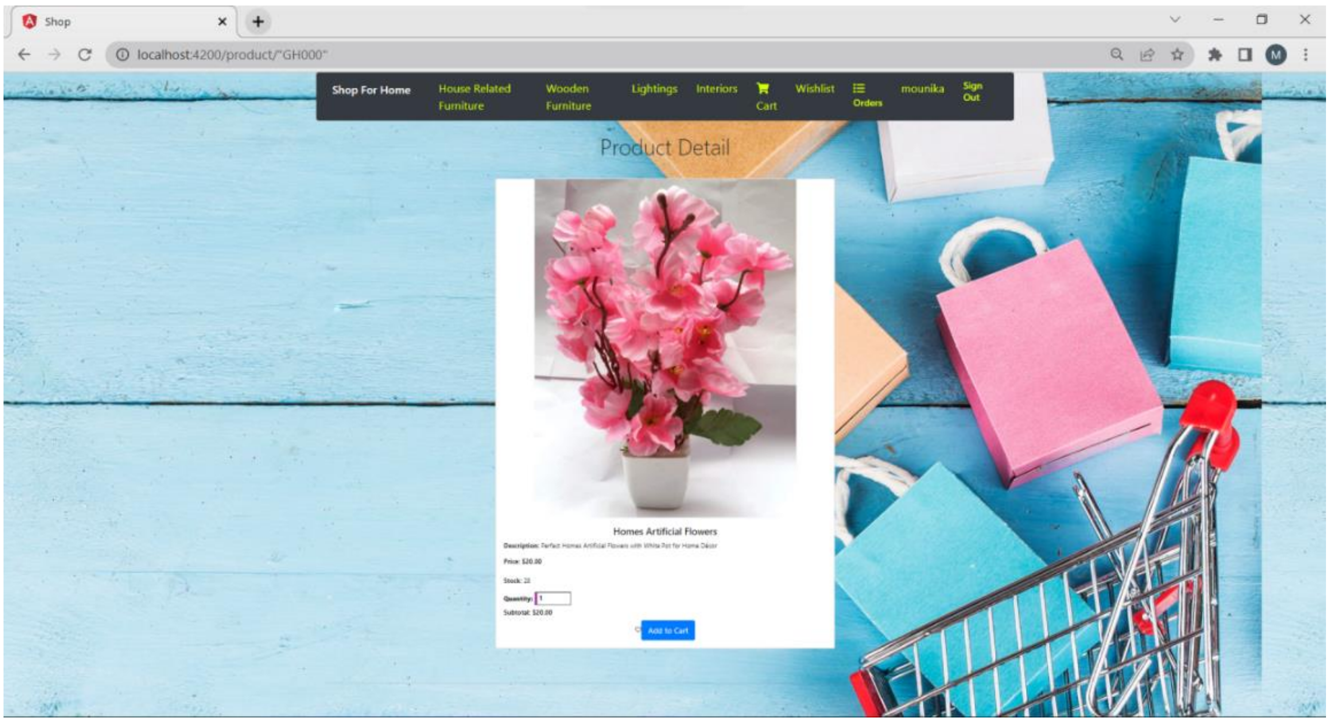
Wall hangings

Description: Wall hangirh handwoven bohemian cotton rope boho tapesty home decor

Price: $49.00

Stock: 6

Products Detail



Product Detail

Homes Artificial Flowers

Description: Perfect Homes Artificial Flowers with White Pot for Home Décor

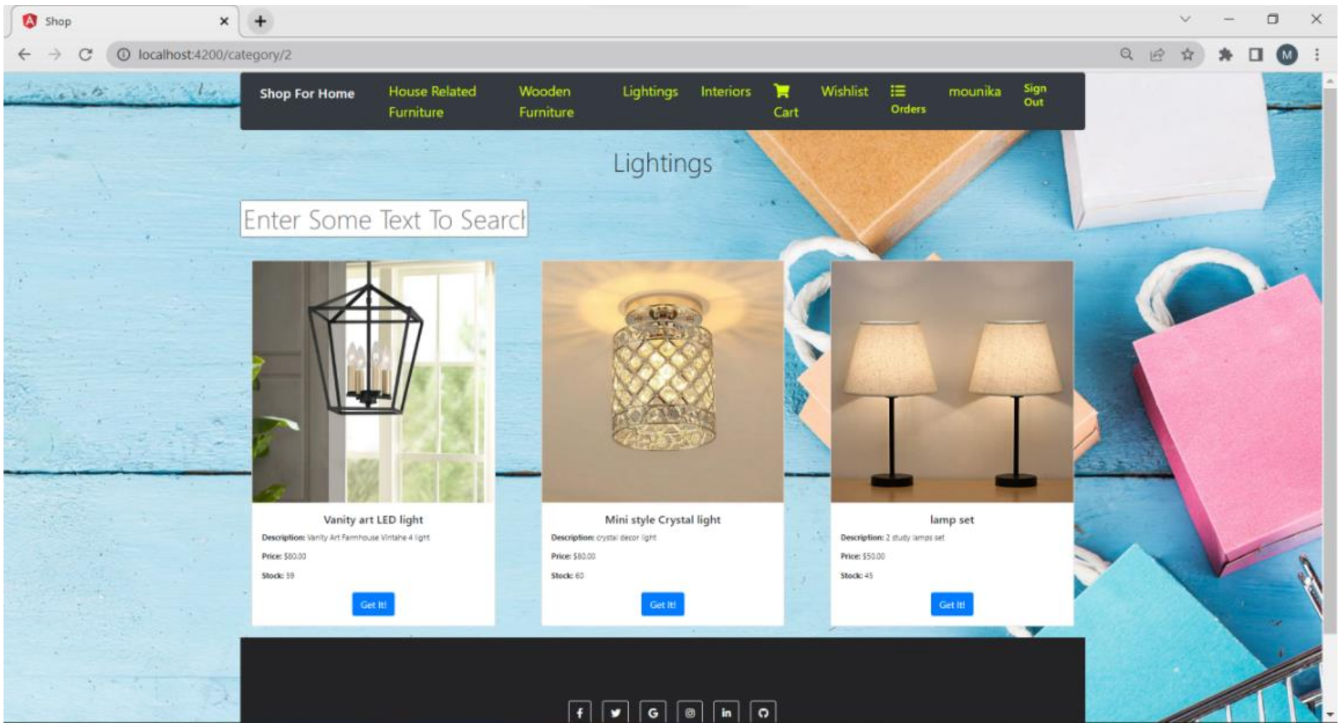
Price: $20.00

Stock: 20

Quantity: 1

Subtotal: $20.00

Lighting page



Lightings

Vanity art LED light

Description: Vanity Art Farmhouse Vintahe 4 light

Price: $80.00

Stock: 59

Mini style Crystal light

Description: crystal decor light

Price: $80.00

Stock: 60

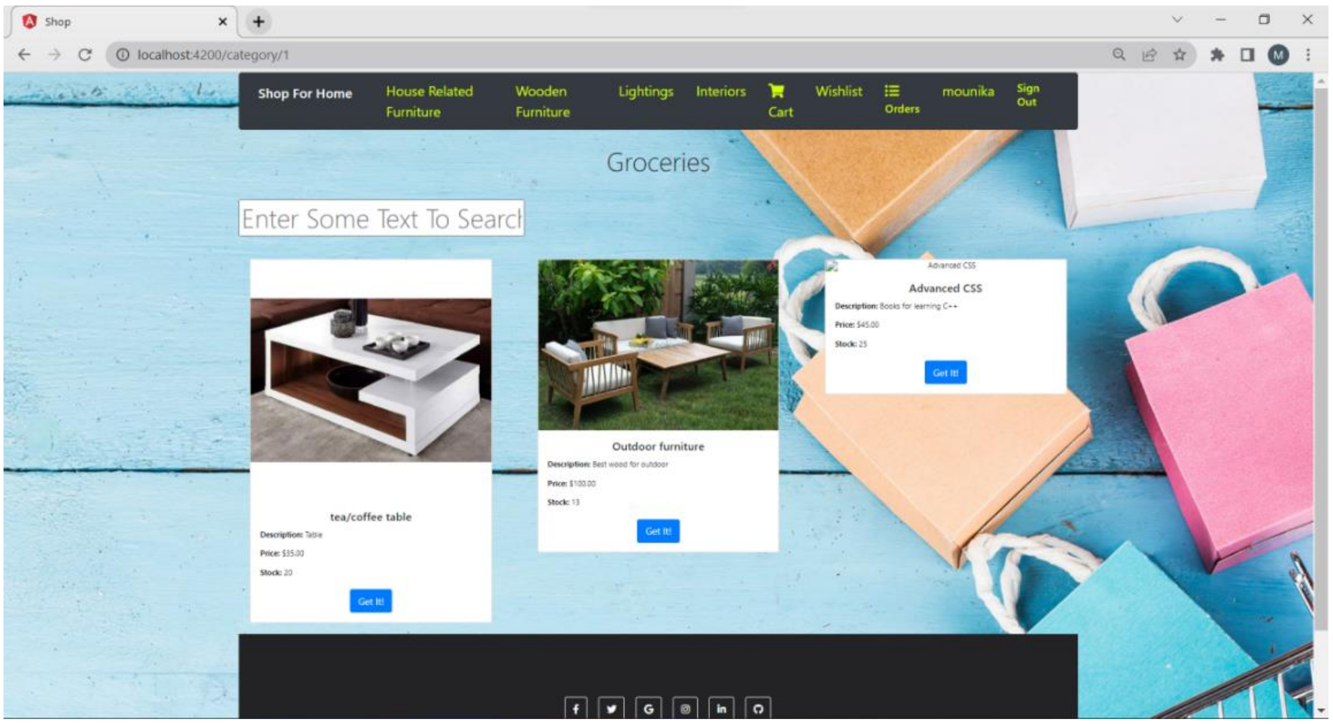
lamp set

Description: 2 study lamps set

Price: $50.00

Stock: 45

Wooden furniture page



tea/coffee table

Description: Table

Price: $35.00

Stock: 20

Outdoor furniture

Description: Best wood for outdoor

Price: $100.00

Stock: 13

Advanced CSS

Description: Books for learning C++

Price: $45.00

Stock: 25

Conclusion

In general, today’s businesses must always strive to create the next best thing that consumers will want because consumers continue to desire their products, services etc. to continuously be better, faster, and cheaper. In this world of new technology, businesses need to accommodate to the new types of consumer needs and trends because it will prove to be vital to their business’ success and survival.

E-commerce is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented. From the inception of the Internet and e-commerce, the possibilities have become endless for both businesses and consumers.

Creating more opportunities for profit and advancements for businesses, while creating more options for consumers. However, just like anything else, e-commerce has its disadvantages including consumer uncertainties, but nothing that cannot be resolved or avoided by good decision-making and business practices.

There are several factors and variables that need to be considered and decided upon when starting an e-commerce business. Some of these include: types of e-commerce, marketing strategies, and countless more. If the correct methods and practices are followed, a business will prosper in an e-commerce setting with much success and profitability.