**Capstone-Project**

**Foodbox**

**DESCRIPTION:**

Create a dynamic and responsive online food delivery web application for ordering food items of different cuisines from a restaurant.

**Background of the problem statement:**  
Foodbox is a restaurant chain that delivers food items of different cuisines at affordable prices. It was established in 2014 in Bengaluru, India. It had been serving fine all these years, however, the business analysts noticed a decline in sales since 2016. They found out that the online ordering of food items with companies, such as Swiggy and Foodpanda were gaining more profit by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer to develop an online food delivery web application with a rich and user-friendly interface.  
You are hired as the Full Stack Java developer and are asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

**Introduction:**

FoodBox is an online food delivery web application for ordering food items of different cuisines from a restaurant. FoodBox is a restaurant chain that delivers food items of different cuisines at affordable prices.

**Modules in the Project:**

a. Registration Page.

b. Login Page.

c. User Login.

d. Admin Login.

e. Dashboard.

f. Search Products.

g. Add Cart/View Cart.

h. View Previous Active Orders.

i. Payment Gateway Page.

j. Order Summary Confirmation Page

**Technology Used:**

**a. Front End:** HTML. CSS3 and Bootstrap4. TypeScript. Angular, Angular Material for View.

**b. VS Code:** As an IDE to design frontend of the application.

**c. Git:** To connect and push files from the local system to GitHub.

**d. GitHub:** To store the application code and track its versions 5

**e. Scrum:** An efficient agile framework to deliver the product incrementally.

**f. Back End:** Eclipse IDE. Java Programming. Searching and Sorting Spring Boot DevTools. Spring Web and Spring Data JPA.

**g. Testing and DevOps:** Selenium. Jenkins. Docker and TestNG.

**Source Code:**

**Backend:**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com</groupId>

<artifactId>foodbox</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Foodbox-Spring</name>

<description>Capstone project for simplilearn</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties:**

server.port=8084

spring.datasource.url=jdbc:mysql://localhost:3306/foodbox

spring.datasource.username=root

spring.datasource.password=Uma@1472#la

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto = update

#logging.level.org.hibernate.SQL=DEBUG

#logging.level.org.hibernate.type=TRACE

**FoodboxSpringApplication.java:**

package com.foodbox;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FoodboxSpringApplication {

public static void main(String[] args) {

SpringApplication.run(FoodboxSpringApplication.class, args);

}

}

**ConfigCtrl.java:**

package com.foodbox.config;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.core.Ordered;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

@Component

@Order(Ordered.HIGHEST\_PRECEDENCE)

public class ConfigCtrl implements Filter {

@Override

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {

final HttpServletResponse response = (HttpServletResponse) res;

response.setHeader("Access-Control-Allow-Origin", "\*");

response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");

response.setHeader("Access-Control-Allow-Headers", "Authorization, Content-Type");

response.setHeader("Access-Control-Max-Age", "3600");

if ("OPTIONS".equalsIgnoreCase(((HttpServletRequest) req).getMethod())) {

response.setStatus(HttpServletResponse.SC\_OK);

} else {

chain.doFilter(req, res);

}

}

@Override

public void destroy() {

}

@Override

public void init(FilterConfig config) throws ServletException {

}

}

**AdminController.java:**

package com.foodbox.controller;

import java.util.Map;

import javax.servlet.http.HttpSession;

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

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Admin;

import com.foodbox.repository.AdminRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")

@RestController

public class AdminController {

@Autowired

private AdminRepository adminRepository;

@SuppressWarnings("rawtypes")

@PostMapping("/admin/{username}")

public boolean verifyAdminLogin(@RequestBody Map loginData, @PathVariable(name = "username") String username, HttpSession session) {

String lusername=(String) loginData.get("username");

String lpassword=(String) loginData.get("password");

Admin admin = adminRepository.findByusername(username);

if(admin!=null && admin.getUsername().equals(lusername) && admin.getPassword().equals(lpassword)) {

session.setAttribute("adminUsername", lusername);

return true;

}else {

return false;

}

}

}

**CartController.java:**

package com.foodbox.controller;

import java.util.List;

import javax.servlet.http.HttpSession;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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.RestController;

import com.foodbox.model.Cart;

import com.foodbox.repository.CartRepository;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class CartController {

@Autowired

private CartRepository cartRepository;

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/carts")

public Cart addToCart(@RequestBody Cart cart, HttpSession session) {

float grandTotal=0;

if(session.getAttribute("grandTotal")==null) {

grandTotal=0;

}

else {

grandTotal=(float) session.getAttribute("grandTotal");

}

List<Cart> cartList = cartRepository.findAll();

for(Cart temp:cartList) {

if(temp.getProduct().getId()==cart.getProduct().getId()) {

int tempQuantity = 1+temp.getQuantity();

grandTotal=grandTotal+temp.getPrice();

session.setAttribute("grandTotal", grandTotal);

temp.setQuantity(tempQuantity);

temp.setPrice((temp.getProduct().getPrice()\*tempQuantity));

return cartRepository.save(temp);

}

}

int min = 100;

int max = 999;

int b = (int) (Math.random() \* (max - min + 1) + min);

cart.setId(b);

cart.setQuantity(1);

cart.setPrice(cart.getProduct().getPrice());

grandTotal=grandTotal+cart.getProduct().getPrice();

session.setAttribute("grandTotal", grandTotal);

return cartRepository.save(cart);

}

@GetMapping("/carts")

public List<Cart> getCartItems() {

return cartRepository.findAll();

}

@PutMapping("/carts/add/{id}")

public ResponseEntity<Cart> addByOne(@PathVariable("id") long id,@RequestBody Cart cart){

/\*Cart cart = cartRepository.findById(id)

.orElseThrow(()->new ResourceNotFoundException("Product not found with"+id));\*/

int quantity= cart.getQuantity()+1;

cart.setQuantity(quantity);

cart.setPrice((cart.getProduct().getPrice())\*quantity);

Cart updatedCart = cartRepository.save(cart);

return ResponseEntity.ok(updatedCart);

}

@PutMapping("/carts/minus/{id}")

public ResponseEntity<Cart> lessByOne(@PathVariable("id") long id,@RequestBody Cart cart){

/\*

\* Cart cart = cartRepository.findById(id) .orElseThrow(()->new

\* ResourceNotFoundException("Product not found with"+id));

\*/

int quantity= cart.getQuantity()-1;

if(quantity!=0) {

cart.setQuantity(quantity);

cart.setPrice((cart.getProduct().getPrice())\*quantity);

Cart updatedCart = cartRepository.save(cart);

return ResponseEntity.ok(updatedCart);

}else {

cartRepository.deleteById(id);

return new ResponseEntity<>(HttpStatus.OK);

}

}

@DeleteMapping("/carts/{id}")

public ResponseEntity<?> deleteCart(@PathVariable("id") Long id)

{

cartRepository.deleteById(id);

return new ResponseEntity<>(HttpStatus.OK);

}

@DeleteMapping("/carts")

public void deleteAllCart(){

cartRepository.deleteAll();

}

}

**CustomerController.java:**

package com.foodbox.controller;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpSession;

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

import org.springframework.http.ResponseEntity;

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.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Customer;

import com.foodbox.repository.CustomerRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")

@RestController

public class CustomerController {

@Autowired

private CustomerRepository customerRepository;

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/customers")

public Customer addCustomer(@RequestBody Customer customer, HttpSession session) {

session.setAttribute("cust\_email", customer.getEmail());

return customerRepository.save(customer);

}

@SuppressWarnings("rawtypes")

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/customers/{email}")

public boolean verifyLogin(@RequestBody Map loginData, @PathVariable(name = "email") String email, HttpSession session) {

String lemail = (String) loginData.get("email");

String lpassword = (String) loginData.get("password");

Customer customer = customerRepository.findByEmail(email);

if(customer!= null && customer.getEmail().equals(lemail) && customer.getPassword().equals(lpassword)) {

session.setAttribute("cust\_email", lemail);

return true;

}else {

return false;

}

}

@GetMapping("/customers")

public List<Customer> getAllCustomers(){

return customerRepository.findAll();

}

@GetMapping("/customers/search/{keyword}")

public List<Customer> searchCustomer(@PathVariable String keyword){

return customerRepository.searchCustomer(keyword);

}

@DeleteMapping("/customers/{email}")

public ResponseEntity<Map<String, Boolean>> deleteCustomer(@PathVariable String email){

Customer customer = customerRepository.findByEmail(email);

customerRepository.delete(customer);

Map<String, Boolean> map = new HashMap<>();

map.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(map);

}

@GetMapping("/customers/{cust\_email}")

public Customer getCustomer(@PathVariable String cust\_email) {

return customerRepository.findByEmail(cust\_email);

}

}

**ProductController.java:**

package com.foodbox.controller;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

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

import org.springframework.http.ResponseEntity;

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.RestController;

import com.foodbox.model.Admin;

import com.foodbox.model.Product;

import com.foodbox.repository.AdminRepository;

import com.foodbox.repository.ProductRepository;

import com.foodbox.exception.ResourceNotFoundException;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class ProductController {

@Autowired

private ProductRepository productRepository;

@Autowired AdminRepository adminRepository;

@GetMapping("/products/Admin")

public List<Product> getAdminProducts() {

return productRepository.findAll();

}

@GetMapping("/products/cust")

public List<Product> getAllProducts() {

List<Product> prodList=productRepository.findIfAvail();

if(prodList.isEmpty()) {

List<Admin> adminList = adminRepository.findAll();

if(adminList.isEmpty()) {

adminRepository.save(new Admin("admin","password"));

}

addProdIfEmpty(new Product(1,"Butter Chicken","Chicken infused with butter and spices","Indian",350,0,0,"yes","./assets/images/ButterChicken.png"));

addProdIfEmpty(new Product(2,"Chicken Biryani","Rice Steamed with Chicken and spices","Indian",365,10,0,"yes","./assets/images/biryani.jpg"));

addProdIfEmpty(new Product(3,"Steamed Mince Bun","Steamed Bun with lamb mince","Chinese",250,20,0,"yes","./assets/images/buns.jpg"));

addProdIfEmpty(new Product(4,"Egg Fried Rice","Rice with Egg and Chinese sauses","Chinese",95,5,0,"yes","./assets/images/EggfriedRice.jpg"));

addProdIfEmpty(new Product(2,"Paneer Pizza","Pizza topped with cotted cheese and vegies","Italian",435,0,0,"yes","./assets/images/paneerpizza.jpg"));

addProdIfEmpty(new Product(2,"Red Sause Pasta","Pasta with Tomato and oregano","Italian",435,0,0,"yes","./assets/images/redPasta.jpg"));

addProdIfEmpty(new Product(2,"Ravioli","Ravioli pasta filled with veg mince","Italian",200,18,0,"yes","./assets/images/ravioli.jpg"));

addProdIfEmpty(new Product(2,"Elote de Corn","Corn topped with cream cheese and spice","Mexican",180,7,0,"yes","./assets/images/elote.jpg"));

addProdIfEmpty(new Product(2,"Burrito","Wrapped Tortilla with Meat mince and Mayo","Mexican",350,0,0,"yes","./assets/images/Burrito.jpg"));

prodList=productRepository.findIfAvail();

}

return prodList;

}

public void addProdIfEmpty(Product product) {

int min = 10000;

int max = 99999;

int b = (int) (Math.random() \* (max - min + 1) + min);

product.setId(b);

float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);

float price = product.getActualPrice() - temp;

product.setPrice(price);

productRepository.save(product);

}

@PostMapping("/products")

public Product addProduct(@RequestBody Product product) {

int min = 10000;

int max = 99999;

int b = (int) (Math.random() \* (max - min + 1) + min);

product.setId(b);

float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);

float price = product.getActualPrice() - temp;

product.setPrice(price);

return productRepository.save(product);

}

@PutMapping("/products/{id}")

public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product productDetails){

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));

product.setName(productDetails.getName());

product.setDesc(productDetails.getDesc());

product.setCategory(productDetails.getCategory());

product.setImagepath(productDetails.getImagepath());

product.setActualPrice(productDetails.getActualPrice());

product.setDiscount(productDetails.getDiscount());

product.setAvail(productDetails.getAvail());

float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);

float price = product.getActualPrice() - temp;

product.setPrice(price);

Product updatedProd = productRepository.save(product);

return ResponseEntity.ok(updatedProd);

}

@DeleteMapping("/products/{id}")

public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id) {

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));

productRepository.delete(product);

Map<String, Boolean> map = new HashMap<>();

map.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(map);

}

@GetMapping("products/{id}")

public ResponseEntity<Product> getProductById(@PathVariable long id) {

Product product = productRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Product Not Found with " + id));

return ResponseEntity.ok(product);

}

@GetMapping("products/search/{keyword}")

public List<Product> getSearchProducts(@PathVariable String keyword) {

return productRepository.homeSearch(keyword);

}

@GetMapping("products/chinese")

public List<Product> getChinese() {

return productRepository.getChinese();

}

@GetMapping("products/indian")

public List<Product> getIndian() {

return productRepository.getIndian();

}

@GetMapping("products/mexican")

public List<Product> getMexican() {

return productRepository.getMexican();

}

@GetMapping("products/italian")

public List<Product> getItalian() {

return productRepository.getItalian();

}

}

**PurchaseController.java:**

package com.foodbox.controller;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

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

import org.springframework.http.ResponseEntity;

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.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.ResourceNotFoundException;

import com.foodbox.model.Cart;

import com.foodbox.model.Customer;

import com.foodbox.model.Purchase;

import com.foodbox.repository.CartRepository;

import com.foodbox.repository.CustomerRepository;

import com.foodbox.repository.PurchaseRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")

@RestController

public class PurchaseController {

@Autowired

private PurchaseRepository purchaseRepository;

@Autowired

private CartRepository cartRepository;

@Autowired

private CustomerRepository customerRepository;

@GetMapping("/purchase/byEmail/{email}")

public List<Purchase> customerOrders(@PathVariable String email) {

return purchaseRepository.getByEmail(email);

}

@GetMapping("/purchase")

public List<Purchase> getAllPurchase(){

return purchaseRepository.findAllByOrderByTransactionidAsc();

}

@DeleteMapping("/purchase/{id}")

public ResponseEntity<Map<String, Boolean>> deletePurchase(@PathVariable Long id) {

Purchase purchase = purchaseRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Purchase Id not found with "+id));

purchaseRepository.delete(purchase);

Map<String, Boolean> map = new HashMap<>();

map.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(map);

}

@GetMapping("/purchase/search/{keyword}")

public List<Purchase> searchPurchase(@PathVariable String keyword){

return purchaseRepository.searchPurchase(keyword);

}

@SuppressWarnings("rawtypes")

@PostMapping("/purchase")

public ResponseEntity<Map<String, Boolean>> buyProducts(@RequestBody Map buyProdMap){

List<Cart> cartList = cartRepository.findAll();

Purchase purchase = new Purchase();

String cust\_email=(String)buyProdMap.get("email");

Customer customer = customerRepository.findByEmail(cust\_email);

String transId = (String)buyProdMap.get("transactionId");

for(Cart cl:cartList) {

java.sql.Date date = new java.sql.Date(new java.util.Date().getTime());

long min=100000;long max=999999;long b = (long)(Math.random()\*(max-min+1)+min);

purchase.setId(b);

purchase.setDop(date);

purchase.setCustomer(customer);

String name = cl.getProduct().getName();

purchase.setProductname(name);

purchase.setQuantity(cl.getQuantity());

purchase.setTotalcost(cl.getPrice());

purchase.setTransactionid(transId);

purchaseRepository.save(purchase);

}

Map<String, Boolean> map = new HashMap<>();

map.put("created",Boolean.TRUE);

return ResponseEntity.ok(map);

}

}

**ResourceNotFoundException.java:**

package com.foodbox.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value=HttpStatus.NOT\_FOUND)

public class ResourceNotFoundException extends RuntimeException{

private static final long serialVersionUID = 1L;

public ResourceNotFoundException(String message) {

super(message);

}

}

**Admin.java:**

package com.foodbox.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Admin {

@Id

private String username;

private String password;

public Admin() {

super();

}

public Admin(String username, String password) {

super();

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**Cart.java:**

package com.foodbox.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity

public class Cart {

@Id

private long id;

private int quantity;

private float price;

@OneToOne

private Product product;

public Cart(long id, int quantity, float price, Product product) {

super();

this.id = id;

this.quantity = quantity;

this.price = price;

this.product = product;

}

public Cart() {

super();

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

**Customer.java:**

package com.foodbox.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Customer {

@Id

private String email;

private String password;

private String name;

private String contact;

private String address;

public Customer(String email, String password, String name, String contact, String address) {

super();

this.email = email;

this.password = password;

this.name = name;

this.contact = contact;

this.address = address;

}

public Customer() {

super();

}

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 getContact() {

return contact;

}

public void setContact(String contact) {

this.contact = contact;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

**Product.java:**

package com.foodbox.model;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Product {

@Id

private long id;

private String name;

private String des;

private String category;

private float actualPrice;

private float discount;

private float price;

private String avail;

private String imagepath;

public Product(long id, String name, String des, String category, float actualPrice, float discount, float price,

String avail, String imagepath) {

super();

this.id = id;

this.name = name;

this.des = des;

this.category = category;

this.actualPrice = actualPrice;

this.discount = discount;

this.price = price;

this.avail = avail;

this.imagepath = imagepath;

}

public Product() {

super();

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesc() {

return des;

}

public void setDesc(String des) {

this.des = des;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public float getActualPrice() {

return actualPrice;

}

public void setActualPrice(float actualPrice) {

this.actualPrice = actualPrice;

}

public float getDiscount() {

return discount;

}

public void setDiscount(float discount) {

this.discount = discount;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public String getAvail() {

return avail;

}

public void setAvail(String avail) {

this.avail = avail;

}

public String getImagepath() {

return imagepath;

}

public void setImagepath(String imagepath) {

this.imagepath = imagepath;

}

}

**Purchase.java:**

package com.foodbox.model;

import java.sql.Date;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity

public class Purchase {

@Id

private long id;

private float totalcost;

private Date dop;

private int quantity;

private String productname;

private String transactionid;

@OneToOne

private Customer customer;

public Purchase() {

super();

}

public Purchase(long id, float totalcost, Date dop, int quantity, String productname, String transactionid,

Customer customer) {

super();

this.id = id;

this.totalcost = totalcost;

this.dop = dop;

this.quantity = quantity;

this.productname = productname;

this.transactionid = transactionid;

this.customer = customer;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public float getTotalcost() {

return totalcost;

}

public void setTotalcost(float totalcost) {

this.totalcost = totalcost;

}

public Date getDop() {

return dop;

}

public void setDop(Date dop) {

this.dop = dop;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public String getProductname() {

return productname;

}

public void setProductname(String productname) {

this.productname = productname;

}

public String getTransactionid() {

return transactionid;

}

public void setTransactionid(String transactionid) {

this.transactionid = transactionid;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

}

**AdminRepository.java:**

package com.foodbox.repository;

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

import com.foodbox.model.Admin;

public interface AdminRepository extends JpaRepository<Admin, String>{

Admin findByusername(String username);

}

**CartRepository.java:**

package com.foodbox.repository;

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

import com.foodbox.model.Cart;

public interface CartRepository extends JpaRepository<Cart, Long> {

}

**CustomerRepository.java:**

package com.foodbox.repository;

import java.util.List;

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

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

import com.foodbox.model.Customer;

public interface CustomerRepository extends JpaRepository<Customer, String> {

Customer findByEmail(String email);

@Query("SELECT c FROM Customer c WHERE c.email LIKE %?1%"

+" OR c.name LIKE %?1%"

+" OR c.contact LIKE %?1%"

+" OR c.address LIKE %?1%")

public List<Customer> searchCustomer(String keyword);

}

**ProductRepository.java:**

package com.foodbox.repository;

import java.util.List;

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

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

import com.foodbox.model.Product;

public interface ProductRepository extends JpaRepository<Product, Long>{

@Query("Select p FROM Product p WHERE p.avail='yes' ORDER BY 'category'")

List<Product> findIfAvail();

@Query("SELECT p FROM Product p WHERE (p.avail LIKE 'yes') AND (p.name LIKE %?1%"

+" OR p.des LIKE %?1%"

+" OR p.price LIKE %?1%"

+" OR p.category LIKE %?1%)")

public List<Product> homeSearch(String keyword);

@Query("SELECT p FROM Product p WHERE p.category LIKE 'Chinese' AND p.avail LIKE 'yes'")

public List<Product> getChinese();

@Query("SELECT p FROM Product p WHERE p.category LIKE 'Indian' AND p.avail LIKE 'yes'")

public List<Product> getIndian();

@Query("SELECT p FROM Product p WHERE p.category LIKE 'Mexican' AND p.avail LIKE 'yes'")

public List<Product> getMexican();

@Query("SELECT p FROM Product p WHERE p.category LIKE 'Italian' AND p.avail LIKE 'yes'")

public List<Product> getItalian();

}

**PurchaseRepository.java:**

package com.foodbox.repository;

import java.util.List;

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

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

import com.foodbox.model.Purchase;

public interface PurchaseRepository extends JpaRepository<Purchase, Long> {

@Query("Select p FROM Purchase p WHERE p.customer.email LIKE %?1%")

public List<Purchase> getByEmail(String email);

public List<Purchase> findAllByOrderByTransactionidAsc();

@Query("Select p FROM Purchase p WHERE p.transactionid LIKE %?1%"

+" OR p.dop LIKE %?1%"

+" OR p.productname LIKE %?1%"

+" OR p.customer LIKE %?1%")

public List<Purchase> searchPurchase(String keyword);

}

**FoodboxSpringApplicationTests.java:**

package com.foodbox;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class FoodboxSpringApplicationTests {

@Test

void contextLoads() {

}

}

**Frontend-Code:**

**index.html:**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>Foodbox-most trusted</title>

    <base href="/" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <link rel="icon" type="image/x-icon" href="favicon.ico" />

    <link

      rel="stylesheet"

      href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"

      integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"

      crossorigin="anonymous"

    />

    <script

      src="https://kit.fontawesome.com/3fe90d9367.js"

      crossorigin="anonymous"

    ></script>

    <link rel="preconnect" href="https://fonts.googleapis.com" />

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

    <link

      href="https://fonts.googleapis.com/css2?family=WindSong&display=swap"

      rel="stylesheet"

    />

    <link

      href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swap"

      rel="stylesheet"

    />

    <link

      href="https://fonts.googleapis.com/icon?family=Material+Icons"

      rel="stylesheet"

    />

  </head>

  <body class="mat-typography">

    <app-root></app-root>

    <script

      src="https://code.jquery.com/jquery-3.5.1.slim.min.js"

      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"

      crossorigin="anonymous"

    ></script>

    <script

      src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"

      integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns"

      crossorigin="anonymous"

    ></script>

  </body>

</html>

**app.component.html:**

<app-header></app-header>

<router-outlet></router-outlet>

**app-routing.module.ts:**

import { ManageCustomersComponent } from './components/manage-customers/manage-customers.component';

import { AdminDashboardComponent } from './components/admin-dashboard/admin-dashboard.component';

import { AdminLoginComponent } from './components/admin-login/admin-login.component';

import { ProductComponent } from './components/product/product.component';

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { CartComponent } from './components/cart/cart.component';

import { ManagePurchaseComponent } from './components/manage-purchase/manage-purchase.component';

import { PaymentGatewayComponent } from './components/payment-gateway/payment-gateway.component';

import { OrderSummaryComponent } from './components/order-summary/order-summary.component';

const routes: Routes = [

  {path:'',redirectTo:'products',pathMatch:'full'},

  {path:'products',component:ProductComponent},

  {path:'cart', component:CartComponent},

  {path:'admin', component:AdminLoginComponent},

  {path:'adminDashboard', component:AdminDashboardComponent},

  {path:'manageCustomer',component:ManageCustomersComponent},

  {path:'managePurchase',component:ManagePurchaseComponent},

  {path:'paymentGateway',component:PaymentGatewayComponent},

  {path:'orderSummary',component:OrderSummaryComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**app.component.spec.ts:**

import { TestBed } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [

        RouterTestingModule

      ],

      declarations: [

        AppComponent

      ],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have as title 'Foodbox-Angular'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('Foodbox-Angular');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement;

    expect(compiled.querySelector('.content span').textContent).toContain('Foodbox-Angular app is running!');

  });

});

**app.component.ts:**

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'Foodbox-Angular';

}

**app.module.ts:**

import { ProductComponent } from './components/product/product.component';

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpClientModule } from '@angular/common/http';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import {MatSnackBarModule} from '@angular/material/snack-bar';

import { AppRoutingModule } from './app-routing.module';

import { MatSortModule } from '@angular/material/sort';

import { AppComponent } from './app.component';

import { HeaderComponent } from './components/header/header.component';

import { CartComponent } from './components/cart/cart.component';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { AdminLoginComponent } from './components/admin-login/admin-login.component';

import { AdminDashboardComponent } from './components/admin-dashboard/admin-dashboard.component';

import { ManageCustomersComponent } from './components/manage-customers/manage-customers.component';

import { ManagePurchaseComponent } from './components/manage-purchase/manage-purchase.component';

import { PaymentGatewayComponent } from './components/payment-gateway/payment-gateway.component';

import { OrderSummaryComponent } from './components/order-summary/order-summary.component';

@NgModule({

  declarations: [

    AppComponent,

    HeaderComponent,

    ProductComponent,

    CartComponent,

    AdminLoginComponent,

    AdminDashboardComponent,

    ManageCustomersComponent,

    ManagePurchaseComponent,

    PaymentGatewayComponent,

    OrderSummaryComponent,

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule,

    FormsModule,

    ReactiveFormsModule,

    BrowserAnimationsModule,

    MatSnackBarModule,

    MatSortModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**admin-dashboard.component.css:**

.logo{

    position:relative;

    top: 8px;

    height: 40px;

    width: 52px;

    cursor: pointer;

}

.header-title{

    position: relative;

    top: 8px;

    left:25px;

    font-family: 'WindSong', cursive;

    font-size: 32px;

    color: #bf1e2e;

}

.bthHead{

    margin-left: 10px;

}

.active, .bthHead:hover{

    color: #bf1e2e;

}

hr{

    margin: 0;

}

.prodLogo{

    position: relative;

    top: 20px;

    font-size: 36px;

    font-weight: bold;

}

.AddProd{

    margin: 10px;

}

.makeCenter{

    text-align: center;

}

.form-check-input{

    margin-top: 3px;

}

p{

    color: red;

}

.modal-title{

    font-size: 18px;

}

th{

    text-align: center;

}

**admin-dashboard.component.html:**

<div>

    <nav class="navbar navbar-expand-lg navbar-light bg-light">

        <div>

            <img routerLink="/products" class="logo" src="./assets/images/logofoodbox.jpg" alt="">

            <strong class="header-title">uma food-delivery</strong>

        </div>

        <div class="collapse navbar-collapse" id="navbarNav">

            <ul class="navbar-nav ml-auto">

                <li class="nav-item">

                    <button class="btn bthHead active"><i class="fas fa-tasks fa-lg"></i> Manage Products</button>

                </li>

                <li class="nav-item">

                    <button class="btn bthHead" routerLink="/manageCustomer"><i class="fas fa-users fa-lg"></i> Manage Users</button>

                </li>

                <li class="nav-item">

                    <button class="btn bthHead" routerLink="/managePurchase"><i class="fas fa-shopping-basket fa-lg"></i> Manage Purchases</button>

                </li>

                <li class="nav-item">

                    <button class="btn bthHead float-right" (click)="adminLogout()">Logout</button>

                </li>

            </ul>

        </div>

    </nav>

    <hr>

</div>

<div>

    <div class="container">

        <span class="text-center prodLogo">Product List</span>

        <button type="button" (click)="clickAddProduct()" data-toggle="modal" data-target="#exampleModal"

            class="btn btn-primary float-right AddProd">Add Product</button>

    </div>

    <table class="table table-striped" matSort (matSortChange)="sortData($event)">

        <thead>

            <tr>

                <th style="width: 17%;" mat-sort-header="name">Name <i class="fas fa-sort"></i></th>

                <th style="width: 20%;">Description</th>

                <th mat-sort-header="category">Category <i class="fas fa-sort"></i></th>

                <th mat-sort-header="actualPrice">Price <i class="fas fa-sort"></i></th>

                <th mat-sort-header="discount">Discount <i class="fas fa-sort"></i></th>

                <th mat-sort-header="price">Final price <i class="fas fa-sort"></i></th>

                <th mat-sort-header="avail">Availabilty <i class="fas fa-sort"></i></th>

                <th style="width: 18%;">ImagePath</th>

                <th style="width:15%;">Actions</th>

            </tr>

        </thead>

        <tbody>

            <tr \*ngFor="let product of sortedProd">

                <td>{{product.name}}</td>

                <td>{{product.desc}}</td>

                <td class="makeCenter">{{product.category}}</td>

                <td class="makeCenter">₹{{product.actualPrice}}</td>

                <td class="makeCenter">{{product.discount}}%</td>

                <td class="makeCenter">₹{{product.price}}</td>

                <td class="makeCenter"><i \*ngIf="product.avail=='no'" class="fas fa-times"></i>

                    <i \*ngIf="product.avail=='yes'" class="fas fa-check"></i>

                </td>

                <td>{{product.imagepath}}</td>

                <td>

                    <button (click)="onEdit(product)" type="button" data-toggle="modal" data-target="#exampleModal"

                        class="btn btn-sm btn-primary">Update</button>

                    <button (click)="deleteProduct(product)" class="btn btn-sm btn-danger" style="margin-left:3px;">Delete</button>

                </td>

            </tr>

        </tbody>

    </table>

</div>

<!--Add Employee Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

    <div class="modal-dialog">

        <div class="modal-content">

            <div class="modal-header">

                <span class="modal-title float-left" id="exampleModalLabel">Product Details</span>

                <button type="button" class="close float-right" data-dismiss="modal" aria-label="Close">

                    <span aria-hidden="true">&times;</span>

                </button>

            </div>

            <p \*ngIf="message!=''">{{message}}</p>

            <div class="modal-body">

                <form [formGroup]="formValue">

                    <div class="form-group">

                        <label for="name" class="col-form-label">Name</label>

                        <input type="text" formControlName="name" class="form-control" id="name" required>

                    </div>

                    <div class="form-group">

                        <label for="desc" class="col-form-label">Description</label>

                        <input type="text" formControlName="desc" class="form-control" id="desc" required>

                    </div>

                    <div class="form-group">

                        <label for="categery" class="col-form-label">Category</label>

                        <select type="text" name="category" formControlName="category" class="form-control"

                            id="category" required>

                            <option value="Chinese">Chinese</option>

                            <option value="Indian">Indian</option>

                            <option value="Mexican">Mexican</option>

                            <option value="Italian">Italian</option>

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="actualPrice" class="col-form-label">Price</label>

                        <input type="text" formControlName="actualPrice" class="form-control" id="actualPrice" required>

                    </div>

                    <div class="form-group">

                        <label for="discount" class="col-form-label">Discount</label>

                        <input type="text" formControlName="discount" class="form-control" id="discount" required>

                    </div>

                    <div class="form-group">

                        <label for="avail" class="form-check-label">Availabilty</label>

                        <div class="form-check">

                            <input type="radio" value="yes" formControlName="avail" name="avail"

                                class="form-check-input" id="yes">

                            <label class="form-check-label" for="avail">

                                Yes

                            </label>

                            <input style="margin-left: 30px;" type="radio" value="no" formControlName="avail" name="avail" class="form-check-input"

                                id="no">

                            <label style="margin-left:48px;" class="form-check-label" for="avail">

                                No

                            </label>

                        </div>

                    </div>

                    <div class="form-group">

                        <label for="imagepath" class="col-form-label">Image Link</label>

                        <input type="text" formControlName="imagepath" class="form-control" id="imagepath" required>

                    </div>

                </form>

            </div>

            <div class="modal-footer">

                <p \*ngIf="message!=''" class="text-left">{{message}}</p>

                <button type="button" id="cancel" class="btn btn-secondary" data-dismiss="modal">Close</button>

                <button type="button" \*ngIf="showAdd" (click)="addProduct()" class="btn btn-primary">Add</button>

                <button type="button" \*ngIf="showUpdate" (click)="updateProduct()"

                    class="btn btn-primary">Update</button>

            </div>

        </div>

    </div>

</div>

**admin-dashboard.component.ts:**

import { Router } from '@angular/router';

import { LogService } from './../../service/log.service';

import { Component, OnInit } from '@angular/core';

import { ProductService } from 'src/app/service/product.service';

import { Product } from 'src/app/model/product';

import {Sort} from '@angular/material/sort';

import { FormBuilder,FormGroup } from '@angular/forms';

@Component({

  selector: 'app-admin-dashboard',

  templateUrl: './admin-dashboard.component.html',

  styleUrls: ['./admin-dashboard.component.css']

})

export class AdminDashboardComponent implements OnInit {

  public allProducts:Product[];

  public sortedProd:Product[];

  formValue!:FormGroup;

  productModelObj:Product=new Product();

  showAdd !: boolean;

  showUpdate !: boolean;

  message:string='';

  constructor(private logService:LogService,

    private router:Router,

    private productService:ProductService,

    private formBuilder:FormBuilder) {

      this.allProducts=[];

      this.sortedProd=this.allProducts.slice();

     }

  ngOnInit(): void {

    this.getProducts();

    this.logService.sendHeader(0);

    this.logService.loggedAdminId$.subscribe((id)=>{

      console.log('id '+id);

      if(id==''){

        this.router.navigate(['/admin']);

      }

    })

    this.formValue=this.formBuilder.group({

      name:[''],

      desc:[''],

      category:[''],

      actualPrice:[''],

      discount:[''],

      avail:[''],

      imagepath:['']

    })

  }

  getProducts(){

    this.productService.getFullProductList().subscribe(data=>{

      this.allProducts=data;

      this.sortedProd=this.allProducts

    })

  }

  sortData(sort:Sort){

    const data=this.allProducts.slice();

    if (!sort.active || sort.direction === '') {

      this.sortedProd = data;

      return;

    }

    this.sortedProd=data.sort((a,b)=>{

      const isAsc = sort.direction === 'asc';

      switch(sort.active){

        case 'name':return this.compare(a.name,b.name,isAsc);

        case 'category':return this.compare(a.category,b.category,isAsc);

        case 'actualPrice': return this.compare(a.actualPrice,b.actualPrice,isAsc);

        case 'discount':return this.compare(a.discount,b.discount,isAsc);

        case 'price': return this.compare(a.price,b.price,isAsc);

        case 'avail': return this.compare(a.avail,b.avail,isAsc);

      }

    })

  }

  compare(a:number|string,b:number|string,isAsc:boolean){

    return (a < b ? -1 : 1) \* (isAsc ? 1 : -1);

  }

  clickAddProduct(){

    this.formValue.reset();

    this.showAdd=true;

    this.showUpdate=false;

  }

  addProduct(){

    if(this.formValue.value.name==null||this.formValue.value.desc==null

      ||this.formValue.value.category==null||this.formValue.value.avail==null

      ||this.formValue.value.imagepath==null||this.formValue.value.actualPrice==null){

      this.message="Fields must not be empty";

      return;

    }

    this.productModelObj.name=this.formValue.value.name;

    this.productModelObj.desc=this.formValue.value.desc;

    this.productModelObj.category=this.formValue.value.category;

    this.productModelObj.actualPrice=this.formValue.value.actualPrice;

    this.productModelObj.discount=this.formValue.value.discount;

    this.productModelObj.avail=this.formValue.value.avail;

    this.productModelObj.imagepath=this.formValue.value.imagepath;

    this.productService.addProduct(this.productModelObj)

    .subscribe(res=>{

      alert("Product Added Sucessfully");

      let ref=document.getElementById('cancel');

      ref?.click();

      this.formValue.reset();

      this.getProducts()

    },

    err=>{

      alert("Something went wrong")

    })

  }

  onEdit(prod:any){

    this.showAdd=false;

    this.showUpdate=true;

    this.productModelObj.id=prod.id;

    this.formValue.controls['name'].setValue(prod.name);

    this.formValue.controls['desc'].setValue(prod.desc);

    this.formValue.controls['category'].setValue(prod.category);

    this.formValue.controls['actualPrice'].setValue(prod.actualPrice);

    this.formValue.controls['discount'].setValue(prod.discount);

    this.formValue.controls['avail'].setValue(prod.avail);

    this.formValue.controls['imagepath'].setValue(prod.imagepath);

  }

  updateProduct(){

    if(this.formValue.value.name==''||this.formValue.value.desc==''

      ||this.formValue.value.category==''||this.formValue.value.avail==''

      ||this.formValue.value.imagepath==''||this.formValue.value.actualPrice==''){

      this.message="Fields must not be empty";

      return;

    }

    this.productModelObj.name=this.formValue.value.name;

    this.productModelObj.desc=this.formValue.value.desc;

    this.productModelObj.category=this.formValue.value.category;

    this.productModelObj.actualPrice=this.formValue.value.actualPrice;

    this.productModelObj.discount=this.formValue.value.discount;

    this.productModelObj.avail=this.formValue.value.avail;

    this.productModelObj.imagepath=this.formValue.value.imagepath;

    this.productService.updateProduct(this.productModelObj.id,this.productModelObj).subscribe(res=>{

      alert("Updated Successfully")

      let ref = document.getElementById('cancel')

      ref?.click();

      this.formValue.reset();

      this.getProducts();

    })

  }

  deleteProduct(prod:any){

    this.productService.deleteProduct(prod.id).subscribe(res=>{

      alert('Product Deleted');

      this.getProducts();

    })

  }

  adminLogout(){

    this.logService.sendId('');

    this.router.navigate(['/admin']);

  }

}

**admin-login.component.html:**

<div class="col-md-6 offset-md-3 shadow-lg mt-5 p-5 text-primary text-center"

style="background-color: white; border-radius: 35px; ">

<h2>Welcome to Admin Login !</h2>

<form (ngSubmit)="loginSubmit()">

    <h5 \*ngIf="error!=''" style="color:red">{{error}}</h5>

    <h2 class="text-center text-lg text-dark"> Username</h2>

    <input [(ngModel)]="loginData.username" name="username" type="username" id="defaultForm-username" class="form-control validate"><br>

    <h2 class="text-center text-dark"> Password </h2>

    <input [(ngModel)]="loginData.password" name="password" type="password" id="defaultForm-pass" class="form-control validate"><br>

    <button type="submit" class="btn btn-dark">Login</button>

</form>

</div>

**admin-login.component.ts:**

import { AdminService } from './../../service/admin.service';

import { LogService } from './../../service/log.service';

import { Component, OnInit } from '@angular/core';

import { FormBuilder,FormGroup } from '@angular/forms';

import { Router } from '@angular/router';

@Component({

  selector: 'app-admin-login',

  templateUrl: './admin-login.component.html',

  styleUrls: ['./admin-login.component.css']

})

export class AdminLoginComponent implements OnInit {

  formValue !: FormGroup;

  adminData!:any;

  error:String='';

  loginData={

    username:'',

    password:''

  }

  auth:any;

  constructor(private logService:LogService, private adminService:AdminService, private router:Router) { }

  ngOnInit(): void {

    this.logService.sendHeader(0);

  }

  loginSubmit(){

    if(this.loginData.username.trim()==''||this.loginData.username==null){

      this.error="Enter Username";

      return

    }

    if(this.loginData.password.trim()==''||this.loginData.password==null){

      this.error="Enter Password";

      return

    }

    this.adminService.adminLogin(this.loginData).subscribe(data=>{

      this.auth=data;

      if(this.auth==true){

        this.logService.sendAdmin(this.loginData.username);

        this.router.navigate(['/adminDashboard']);

      }else{

        this.error="Username or password incorrect"

      }

    })

  }

}

**cart.component.html:**

<ng-container \*ngIf="products.length!=0">

    <div class="container">

        <div class="card-table">

            <div class="card-product">

                <table class="table" matSort (matSortChange)="sortData($event)">

                    <thead>

                        <tr>

                            <th>Sr.No</th>

                            <th mat-sort-header="name">Product Name</th>

                            <th>Product Image</th>

                            <th mat-sort-header="category">Category</th>

                            <th>Price</th>

                            <th mat-sort-header="quantity">Quantity</th>

                            <th mat-sort-header="price">Total</th>

                            <th>Action</th>

                        </tr>

                    </thead>

                    <tbody>

                        <tr \*ngFor="let item of sortedCart;let i=index">

                            <td>{{i+1}}</td>

                            <td>{{item.product.name}}</td>

                            <td><img style="width:120px;" src="{{item.product.imagepath}}" alt=""></td>

                            <td>{{item.product.category}}</td>

                            <td style="width:12%">₹{{item.product.price}}</td>

                            <td style="width:12%"><span class="quantityControl" (click)="lessOneCart(item.id,item)">-</span><span

                                    class="border quantity">{{item.quantity}}</span><span

                                    class="quantityControl" (click)="addOneCart(item.id,item)">+</span></td>

                            <td style="width:12%">₹{{item.price}}</td>

                            <td>

                                <button class="btn btn-danger" (click)="deleteItem(item.id)"><i

                                        class="fas fa-trash-alt"></i></button>

                            </td>

                        </tr>

                        <tr>

                            <td colspan="4"></td>

                            <td><button class="btn btn-danger" (click)="removeAllCart()">Empty Cart</button></td>

                            <td><button routerLink="/products" class="btn btn-primary">Shop More</button></td>

                            <td><button routerLink="/paymentGateway"class="btn btn-success" id="checkout">CheckOut</button></td>

                            <td><strong>Grand Total : ${{grandTotal}}</strong><br><span class="ship">Delivery charges

                                    will be applied</span></td>

                        </tr>

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</ng-container>

<ng-container \*ngIf="products.length==0">

    <div class="container">

        <div class="card">

            <h1 class="card-title">My Cart</h1>

        </div>

        <div class="center">

            <img src="https://rukminim1.flixcart.com/www/800/800/promos/16/05/2019/d438a32e-765a-4d8b-b4a6-520b560971e8.png?q=90"

                alt="">

            <h3>Your cart is empty</h3>

            <h5>Add item into it now</h5>

            <button routerLink="/products" class="btn btn-primary">Shop Now</button>

        </div>

    </div>

</ng-container>

**cart.component.ts:**

import { LogService } from './../../service/log.service';

import { Cart } from './../../model/cart';

import { CartService } from './../../service/cart.service';

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import {Sort} from '@angular/material/sort';

import { CompileMetadataResolver } from '@angular/compiler';

@Component({

  selector: 'app-cart',

  templateUrl: './cart.component.html',

  styleUrls: ['./cart.component.css']

})

export class CartComponent implements OnInit {

  public cart:Cart=new Cart();

  public products:Cart[];

  public sortedCart:Cart[];

  public grandTotal : number = 0;

  constructor(private cartService:CartService, private router:Router, private logService:LogService) {

    this.products=[];

    this.sortedCart=this.products.slice();

   }

  ngOnInit(): void {

    this.logService.sendHeader(1);

    this.logService.loggeduserId$.subscribe(

      (id)=>{

        if(id==''){

            this.router.navigate(['']);

        }});

    this.getCartItemList();

  }

  sortData(sort:Sort){

    const data=this.products.slice();

    if(!sort.active || sort.direction===''){

      this.sortedCart=data;

    }

    this.sortedCart=data.sort((a,b)=>{

      const isAsc = sort.direction === 'asc';

      switch(sort.active){

        case 'name':return this.compare(a.product.name,b.product.name, isAsc);

        case 'category': return this.compare(a.product.category,b.product.category,isAsc);

        case 'price':return this.compare(a.price,b.price,isAsc);

        case 'quantity':return this.compare(a.quantity,b.quantity,isAsc);

        default:return 0;

      }

    })

  }

  compare(a:number|string,b:number|string,isAsc:boolean){

    return (a < b ? -1 : 1) \* (isAsc ? 1 : -1);

  }

  private getCartItemList(){

    this.cartService.getCartItemList().subscribe((res:Cart[])=>{

      this.products=res;

      this.grandTotal=0;

      for(let i=0;i<this.products.length;i++)

      {

        this.grandTotal+=this.products[i].price;

      }

      sessionStorage.setItem('grandTotal',""+this.grandTotal);

      this.sortedCart=this.products;

    });

  }

  deleteItem(id:any){

    this.cartService.deleteItem(id).subscribe(res=>{

      this.getCartItemList();

    })

  }

  removeAllCart(){

    this.cartService.deleteAllCart().subscribe(res=>{

      this.getCartItemList();

    })

  }

  addOneCart(id:any,cart:Cart){

    this.cartService.addOneCart(id,cart).subscribe(res=>{

      this.getCartItemList();

    })

  }

  lessOneCart(id:any,cart:Cart){

    this.cartService.lessOneCart(id,cart).subscribe(res=>{

      this.getCartItemList();

    })

  }

}

**header.component.html:**

<header \*ngIf="showHeader==1">

    <div class="navbar shadow-sm">

        <div class="container">

            <div class="navbar-brand align-item-center d-flex">

                <div>

                    <img routerLink="/products" class="logo" src="./assets/images/logofoodbox.jpg" alt="">

                </div>

                <strong class="header-title">UMA'S-Delicious Food</strong>

            </div>

            <div \*ngIf="login==0" class="text-right">

                <button class="btn btnMain" type="button" data-toggle="modal" data-target="#loginModal" id="custLogin">Login</button>

                <button class="btn btnMain" type="button" data-toggle="modal" data-target="#registerModal" id="custRegister">Register</button>

            </div>

            <div \*ngIf="login==1" class="text-right">

                <button class="btn btnMain" (click)="getActiveOrders()" type="button" data-toggle="modal" data-target="#activeOrders">Previous Orders</button>

                <button class="btn btnMain" (click)="CustomerLogout()">Logout</button>

            </div>

        </div>

        <button \*ngIf="login==1" routerLink="cart" class="btn cart">

            <i style="font-size: 50px;color:#372861;" class="fas fa-cart-plus"></i>

        </button>

    </div>

    <div class="modal fade" id="loginModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel"

        aria-hidden="true">

        <div class="modal-dialog" role="document">

            <div class="modal-content">

                <div class="modal-header text-center">

                    <h4 class="modal-title w-100 font-weight-bold">Sign in</h4>

                    <button type="button" class="close" id="Lcancel" data-dismiss="modal" aria-label="Close">

                        <span aria-hidden="true">&times;</span>

                    </button>

                </div>

                <p \*ngIf="message!= ''">{{message}}</p>

                <form (ngSubmit)="loginSubmit()">

                    <div class="modal-body mx-3">

                        <div class="md-form mb-3">

                            <i class="fas fa-envelope prefix grey-text"></i>

                            <input [(ngModel)]="loginData.email" name="email" type="email" id="defaultForm-email"

                                class="form-control validate">

                            <label data-error="wrong" data-success="right" for="defaultForm-email">Your email</label>

                        </div>

                        <div class="md-form mb-3">

                            <i class="fas fa-lock prefix grey-text"></i>

                            <input [(ngModel)]="loginData.password" name="password" type="password"

                                id="defaultForm-pass" class="form-control validate">

                            <label data-error="wrong" data-success="right" for="defaultForm-pass">Your password</label>

                        </div>

                    </div>

                    <div class="modal-footer d-flex justify-content-center">

                        <button type="submit" id="loginSubmit" class="btn btn-default">Login</button>

                    </div>

                </form>

            </div>

        </div>

    </div>

    <div class="modal fade" id="registerModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel"

        aria-hidden="true">

        <div class="modal-dialog" role="document">

            <div class="modal-content">

                <div class="modal-header text-center">

                    <h4 class="modal-title w-100 font-weight-bold">Sign up</h4>

                    <p \*ngIf="message!= ''">{{message}}</p>

                    <button type="button" class="close" id="Rcancel" data-dismiss="modal" aria-label="Close">

                        <span aria-hidden="true">&times;</span>

                    </button>

                </div>

                <form [formGroup]="formValue">

                    <div class="modal-body mx-3">

                        <div class="md-form form-group mb-2">

                            <i class="fas fa-envelope prefix grey-text"> Email Id</i>

                            <input type="email" formControlName="email" id="orangeForm-Remail"

                                class="form-control validate">

                        </div>

                        <div class="md-form form-group mb-2">

                            <i class="fas fa-lock prefix grey-text"> Password</i>

                            <input type="password" formControlName="password" id="orangeForm-Rpass"

                                class="form-control validate">

                        </div>

                        <div class="md-form form-group mb-2">

                            <i class="fas fa-user prefix grey-text"> Full Name</i>

                            <input type="text" formControlName="name" id="orangeForm-name"

                                class="form-control validate">

                        </div>

                        <div class="md-form form-group mb-2">

                            <i class="fas fa-user prefix grey-text"> Contact Number</i>

                            <input type="text" formControlName="contact" id="orangeForm-contact"

                                class="form-control validate">

                        </div>

                        <div class="md-form form-group mb-2">

                            <i class="fas fa-user prefix grey-text"> Address</i>

                            <input type="text" formControlName="address" id="orangeForm-address"

                                class="form-control validate">

                        </div>

                    </div>

                </form>

                <div class="modal-footer d-flex justify-content-center">

                    <button class="btn btn-deep-orange" (click)=addCustomer()>Register</button>

                </div>

            </div>

        </div>

    </div>

    <div class="modal" id="activeOrders">

        <div class="modal-dialog modal-lg">

            <div class="modal-content">

                <!-- Modal Header -->

                <div class="modal-header">

                    <h4 class="modal-title">Previous Orders</h4>

                    <button type="button" class="close" data-dismiss="modal">&times;</button>

                </div>

                <!-- Modal body -->

                <div class="modal-body">

                    <div \*ngIf="activeOrders.length!=0">

                        <table class="table">

                            <thead class="thead-#bf1e2e">

                                <tr>

                                    <th>Product Name</th>

                                    <th>Quantity</th>

                                    <th>Cost</th>

                                    <th>Purchase Data</th>

                                    <th>Transaction Id</th>

                                </tr>

                            </thead>

                            <tbody>

                                <tr \*ngFor="let pur of activeOrders">

                                    <td>{{pur.productname}}</td>

                                    <td>{{pur.quantity}}</td>

                                    <td>{{pur.totalcost}}</td>

                                    <td>{{pur.dop}}</td>

                                    <td>{{pur.transactionid}}</td>

                                </tr>

                            </tbody>

                        </table>

                    </div>

                    <div \*ngIf="activeOrders.length==0">

                        <h5 class="text-center text-danger">No Previous orders</h5>

                    </div>

                </div>

                <!-- Modal footer -->

                <div class="modal-footer">

                    <button type="button" class="btn btn-sm btn-danger" data-dismiss="modal">Close</button>

                </div>

            </div>

        </div>

    </div>

</header>

**header.component.ts:**

import { LogService } from './../../service/log.service';

import { Cart } from './../../model/cart';

import { CustomerService } from './../../service/customer.service';

import { CartService } from './../../service/cart.service';

import { Customer } from './../../model/customer';

import { Component, OnInit } from '@angular/core';

import { FormBuilder,FormGroup } from '@angular/forms';

import { Router } from '@angular/router';

import { ProductService } from 'src/app/service/product.service';

import { Purchase } from 'src/app/model/purchase';

import { PurchaseService } from 'src/app/service/purchase.service';

@Component({

  selector: 'app-header',

  templateUrl: './header.component.html',

  styleUrls: ['./header.component.css']

})

export class HeaderComponent implements OnInit {

  showHeader:number=0;

  message:string='';

  formValue !: FormGroup;

  customer:Customer = new Customer();

  auth:any;

  loginData={

    email:'',

    password:''};

  login:number=0;

  productList:any;

  public cart:Cart=new Cart();

  public products:Cart[];

  public activeOrders:Purchase[]=[];

  public totalItem:number=0;

  constructor(private cartService:CartService,

    private productService:ProductService,

    private customerService:CustomerService,

    private logService:LogService,

    private purchaseService:PurchaseService,

    private formbuilder:FormBuilder,

    private router:Router) { }

  ngOnInit(): void {

    this.logService.headerId$.subscribe((id)=>{

      this.showHeader=id;

    })

    sessionStorage.setItem('cust\_email',null);

    this.cartService.deleteAllCart().subscribe(res=>{

      console.log('deletedAllCart')

    })

    this.cartService.getProduct().subscribe(res=>{

      this.products=res;

      this.totalItem=this.products.length;

    });

    this.productService.getProductList().subscribe(res=>{

      this.productList=res;

    })

    this.formValue=this.formbuilder.group({

      email:[''],

      password:[''],

      name:[''],

      contact:[''],

      address:['']

    })

  }

  addCustomer(){

    this.customer.email=this.formValue.value.email;

    this.customer.password=this.formValue.value.password;

    this.customer.name=this.formValue.value.name;

    this.customer.contact=this.formValue.value.contact;

    this.customer.address=this.formValue.value.address;

    this.customerService.addCustomer(this.customer).subscribe(data=>{

      this.login=1;

      this.productService.login.next(1);

      sessionStorage.setItem('cust\_email',this.customer.email);

      sessionStorage.setItem('cust\_name',this.customer.name);

      this.logService.sendId(this.customer.email);

      let ref = document.getElementById('Rcancel')

      ref?.click();

      this.formValue.reset();

      this.goToProducts();

    },

    error=> this.message='User already exist, Please Login' )

  }

  goToProducts(){

    this.router.navigate(['/products']);

  }

  loginSubmit(){

    if(this.loginData.email.trim()==''|| this.loginData.email==null)

    {

      this.message="Enter Username"

      return;

    }

    if(this.loginData.password.trim()==''|| this.loginData.password==null)

    {

      this.message="Enter password"

      return;

    }

    this.customerService.customerLogin(this.loginData).subscribe(data =>{

      this.auth=data;

      if(this.auth==true){

        this.login=1;

        this.productService.login.next(1);

        sessionStorage.setItem('cust\_email',this.loginData.email);

        this.logService.sendId('cust\_email');

        let ref = document.getElementById('Lcancel')

      ref?.click();

      this.formValue.reset();

      this.goToProducts();

      }else{

        this.message='Username or password incorrect'

      }

    })

  }

  CustomerLogout(){

    this.login=0;

    this.cartService.deleteAllCart().subscribe(data=>{

      console.log('cart empty')

    });

    sessionStorage.setItem('cust\_email',null);

    this.logService.sendId('');

  }

  getActiveOrders(){

    var email:string;

    email=sessionStorage.getItem("cust\_email");

    this.purchaseService.getCustomerOrders(email).subscribe(data=>{

      this.activeOrders=data;

    })

  }

}

**header.component.css:**

.logo{

    position:relative;

    top: 8px;

    height: 80px;

    width: 105px;

    cursor: pointer;

}

.header-title{

    position: relative;

    top: 40px;

    left:25px;

    font-family: "Papyrus", Fantasy;

    font-style: italic;

    font-size: 64px;

    color: #ac0b0b;

}

.btnMain{

    font-size: 20px;

}

.btnMain:hover{

    color: #bf1e2e;

}

.modal-footer button{

    background-color: #bf1e2e;

    color:white

}

th{

    color:#bf1e2e;

}

**manage-customers.component.html:**

<div>

    <nav class="navbar navbar-expand-lg navbar-light bg-light">

        <div>

            <img routerLink="/products" class="logo" src="./assets/images/logofoodbox.jpg" alt="">

            <strong class="header-title">uma food-delivery</strong>

        </div>

        <div class="collapse navbar-collapse" id="navbarNav">

            <ul class="navbar-nav ml-auto">

                <li class="nav-item">

                    <button class="btn btnHead" routerLink="/adminDashboard"><i class="fas fa-tasks fa-lg"></i> Manage

                        Products</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead active"><i class="fas fa-users fa-lg"></i> Manage Users</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead" routerLink="/managePurchase"><i

                            class="fas fa-shopping-basket fa-lg"></i> Manage Purchases</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead float-right" (click)="adminLogout()">Logout</button>

                </li>

            </ul>

        </div>

    </nav>

    <hr>

</div>

<div>

    <div class="container">

        <span class="text-center prodLogo">Users List</span>

        <input type="text" class="float-right form-control" name="searchKey" [(ngModel)]="searchKey"

            (ngModelChange)="searchCustomer()" placeholder="Search Customer">

    </div>

    <div class="container">

        <table class="table table-striped table-bordered">

            <thead>

                <tr>

                    <th>Email</th>

                    <th>Name</th>

                    <th>Contact No</th>

                    <th>Address</th>

                    <th>Actions</th>

                </tr>

            </thead>

            <tbody>

                <tr \*ngFor="let cust of allCustomers">

                    <td>{{cust.email}}</td>

                    <td>{{cust.name}}</td>

                    <td>{{cust.contact}}</td>

                    <td>{{cust.address}}</td>

                    <td><button (click)="getActiveOrders(cust.email)" class="btn btn-primary btn-sm" type="button"

                            data-toggle="modal" data-target="#activeOrders">View active Orders</button>

                        <button (click)="deleteCustomer(cust.email)" class="btn btn-danger btn-sm"

                            style="margin-left:3px;">Delete</button>

                    </td>

                </tr>

            </tbody>

        </table>

    </div>

    <div class="modal" id="activeOrders">

        <div class="modal-dialog modal-lg">

            <div class="modal-content">

                <!-- Modal Header -->

                <div class="modal-header">

                    <h4 class="modal-title">Previous Orders</h4>

                    <button type="button" class="close" data-dismiss="modal">&times;</button>

                </div>

                <!-- Modal body -->

                <div class="modal-body">

                    <div \*ngIf="activeOrders.length!=0">

                        <table class="table">

                            <thead class="thead-#bf1e2e">

                                <tr>

                                    <th>Product Name</th>

                                    <th>Quantity</th>

                                    <th>Cost</th>

                                    <th>Purchase Data</th>

                                    <th>Transaction Id</th>

                                </tr>

                            </thead>

                            <tbody>

                                <tr \*ngFor="let pur of activeOrders">

                                    <td>{{pur.productname}}</td>

                                    <td>{{pur.quantity}}</td>

                                    <td>{{pur.totalcost}}</td>

                                    <td>{{pur.dop}}</td>

                                    <td>{{pur.transactionid}}</td>

                                </tr>

                            </tbody>

                        </table>

                    </div>

                    <div \*ngIf="activeOrders.length==0">

                        <h5 class="text-center text-danger">No Previous orders</h5>

                    </div>

                </div>

                <!-- Modal footer -->

                <div class="modal-footer">

                    <button type="button" class="btn btn-sm btn-danger" data-dismiss="modal">Close</button>

                </div>

            </div>

        </div>

    </div>

</div>

**manage-customers.component.ts:**

import { Router } from '@angular/router';

import { LogService } from './../../service/log.service';

import { Component, OnInit } from '@angular/core';

import { Customer } from 'src/app/model/customer';

import { CustomerService } from 'src/app/service/customer.service';

import { Purchase } from 'src/app/model/purchase';

import { PurchaseService } from 'src/app/service/purchase.service';

import { error } from '@angular/compiler/src/util';

@Component({

  selector: 'app-manage-customers',

  templateUrl: './manage-customers.component.html',

  styleUrls: ['./manage-customers.component.css']

})

export class ManageCustomersComponent implements OnInit {

  searchKey:string;

  public allCustomers:Customer[];

  public activeOrders:Purchase[]=[];

  constructor(private customerService:CustomerService,

    private purchaseService:PurchaseService,

    private logService:LogService,

    private router:Router) { }

  ngOnInit(): void {

    this.logService.loggedAdminId$.subscribe((id)=>{

      console.log('id '+id);

      if(id==''){

        this.router.navigate(['/admin']);

      }

    })

    this.getCustomers();

  }

  getCustomers(){

    this.customerService.getCustomers().subscribe(data=>{

      this.allCustomers=data;

    })

  }

  adminLogout(){

    this.logService.sendId('');

    this.router.navigate(['/admin']);

  }

  searchCustomer(){

    if(this.searchKey==''){

      this.getCustomers();

    }else{

      this.customerService.searchCustomer(this.searchKey).subscribe(data=>{

        this.allCustomers=data;

      })

    }

  }

  getActiveOrders(email:string){

    this.purchaseService.getCustomerOrders(email).subscribe(data=>{

      this.activeOrders=data;

    })

  }

  deleteCustomer(email:string){

    this.customerService.deleteCustomer(email).subscribe(data=>{

      alert("Customer Deleted");

      this.getCustomers();

    },error=>{

      alert("Customer can't be deleted until their orders are deleted")

    }

    )

  }

}

**manage-customers.component.css:**

.logo{

    position:relative;

    top: 8px;

    height: 40px;

    width: 52px;

    cursor: pointer;

}

.header-title{

    position: relative;

    top: 8px;

    left:25px;

    font-family: 'WindSong', cursive;

    font-size: 32px;

    color: #bf1e2e;

}

.btnHead{

    margin-left: 10px;

}

.active, .btnHead:hover{

    color: #bf1e2e;

}

hr{

    margin: 0;

}

.prodLogo{

    position: relative;

    top: 20px;

    font-size: 36px;

    font-weight: bold;

}

.form-control{

    width:30%;

    margin: 10px 0;

}

**manage-purchase.component.html:**

<div>

    <nav class="navbar navbar-expand-lg navbar-light bg-light">

        <div>

            <img routerLink="/products" class="logo" src="./assets/images/logofoodbox.jpg" alt="">

            <strong class="header-title">uma food-delivery</strong>

        </div>

        <div class="collapse navbar-collapse" id="navbarNav">

            <ul class="navbar-nav ml-auto">

                <li class="nav-item">

                    <button class="btn btnHead" routerLink="/adminDashboard"><i class="fas fa-tasks fa-lg"></i> Manage

                        Products</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead" routerLink="/manageCustomer"><i class="fas fa-users fa-lg"></i> Manage

                        Users</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead active" routerLink="/managePurchase"><i

                            class="fas fa-shopping-basket fa-lg"></i> Manage Purchases</button>

                </li>

                <li class="nav-item">

                    <button class="btn btnHead float-right" (click)="adminLogout()">Logout</button>

                </li>

            </ul>

        </div>

    </nav>

    <hr>

</div>

<div>

    <div class="container">

        <span class="text-center prodLogo">Users List</span>

        <input type="text" class="float-right form-control" [(ngModel)]="keyword"

        (ngModelChange)="searchPurchase()" name="searchKey" id="searchKey"

            placeholder="Search Purchase">

    </div>

    <div class="container">

        <table class="table table-striped table-bordered">

            <thead>

                <tr>

                    <th>Transaction Id</th>

                    <th>Date of Purchase</th>

                    <th>Product Name</th>

                    <th>Quantity</th>

                    <th>Total Cost</th>

                    <th>Customer Email</th>

                    <th>Action</th>

                </tr>

            </thead>

            <tbody>

                <tr \*ngFor="let item of allPurchase">

                    <td>{{item.transactionid}}</td>

                    <td>{{item.dop}}</td>

                    <td>{{item.productname}}</td>

                    <td>{{item.quantity}}</td>

                    <td>{{item.totalcost}}</td>

                    <td>{{item.customer.email}}</td>

                    <td><button (click)="deletePurchase(item.id)" class="btn btn-sm btn-danger">Delete</button></td>

                </tr>

            </tbody>

        </table>

    </div>

</div>

**manage-purchase.component.css:**

.logo{

    position:relative;

    top: 8px;

    height: 40px;

    width: 52px;

    cursor: pointer;

}

.header-title{

    position: relative;

    top: 8px;

    left:25px;

    font-family: 'WindSong', cursive;

    font-size: 32px;

    color: #bf1e2e;

}

.btnHead{

    margin-left: 10px;

}

.active, .btnHead:hover{

    color: #bf1e2e;

}

hr{

    margin: 0;

}

.prodLogo{

    position: relative;

    top: 20px;

    font-size: 36px;

    font-weight: bold;

}

.form-control{

    width:30%;

    margin: 10px 0;

}

**manage-purchase.component.ts:**

import { LogService } from './../../service/log.service';

import { Component, OnInit } from '@angular/core';

import { Purchase } from 'src/app/model/purchase';

import { PurchaseService } from 'src/app/service/purchase.service';

import { Router } from '@angular/router';

@Component({

  selector: 'app-manage-purchase',

  templateUrl: './manage-purchase.component.html',

  styleUrls: ['./manage-purchase.component.css']

})

export class ManagePurchaseComponent implements OnInit {

  public allPurchase:Purchase[];

  keyword:any;

  constructor(private purchaseService:PurchaseService,

              private logService:LogService,

              private router:Router) { }

  ngOnInit(): void {

    this.logService.loggedAdminId$.subscribe((id)=>{

      console.log('id '+id);

      if(id==''){

        this.router.navigate(['/admin']);

      }

    })

    this.getAllPurchase();

  }

  getAllPurchase(){

    this.purchaseService.getAllPurchase().subscribe(data=>{

      this.allPurchase=data;

    })

  }

  adminLogout(){

    this.logService.sendId('');

    this.router.navigate(['/admin']);

  }

  deletePurchase(id:any){

    this.purchaseService.deletePurchase(id).subscribe(data=>{

      alert("Purchase Deleted");

      this.getAllPurchase();

    })

  }

  searchPurchase(){

    if(this.keyword==''){

      this.getAllPurchase();

    }else{

      this.purchaseService.searchPurchase(this.keyword).subscribe(data=>{

        this.allPurchase=data;

      });

    }

  }

}

**order-summary.component.css:**

@import url('https://fonts.googleapis.com/css2?family=Montserrat&display=swap');

body {

    background-color: #ffe8d2;

    font-family: 'Montserrat', sans-serif

}

.card {

    border: none

}

.logo {

    background-color: #eeeeeea8

}

.totals tr td {

    font-size: 13px

}

.footer {

    background-color: #eeeeeea8

}

.footer span {

    font-size: 12px

}

.product-qty span {

    font-size: 12px;

    color: #dedbdb

}

.shopMore{

    position: relative;

    top:10px;

   left:400px

}

.printOrder{

    position: relative;

    top:10px;

    left:380px

}

**order-summary.component.html:**

<div class="container mt-5 mb-5">

    <div class="row d-flex justify-content-center">

        <div class="col-md-8">

            <div class="card">

                <div class="text-left logo p-2 px-5"> <img src="https://i.imgur.com/2zDU056.png" width="50"> </div>

                <div class="invoice p-5">

                    <h5>Your order Confirmed!</h5> <span class="font-weight-bold d-block mt-4">Hello, {{customer.name}}</span> <span>You order has been confirmed and will be shipped in next two hours!</span>

                    <div class="payment border-top mt-3 mb-3 border-bottom table-responsive">

                        <table class="table table-borderless">

                            <tbody>

                                <tr>

                                    <td>

                                        <div class="py-2"> <span class="d-block text-muted">Order Date</span> <span>{{todayDate|date:'shortDate'}}</span> </div>

                                    </td>

                                    <td>

                                        <div class="py-2"> <span class="d-block text-muted">Transaction Id</span> <span>{{transId}}</span> </div>

                                    </td>

                                    <td>

                                        <div class="py-2"> <span class="d-block text-muted">Payment</span> <span><img src="https://img.icons8.com/color/48/000000/mastercard.png" width="20" /></span> </div>

                                    </td>

                                    <td>

                                        <div class="py-2"> <span class="d-block text-muted">Shiping Address</span> <span>{{customer.address}}</span> </div>

                                    </td>

                                </tr>

                            </tbody>

                        </table>

                    </div>

                    <div class="product border-bottom table-responsive">

                        <table class="table table-borderless">

                            <tbody>

                                <tr \*ngFor="let item of cartItems">

                                    <td width="20%"> <img src="{{item.product.imagepath}}" width="90"> </td>

                                    <td width="60%"> <span class="font-weight-bold">{{item.product.name}}</span>

                                        <div class="product-qty"> <span class="d-block">{{item.quantity}}</span> <span>{{item.product.category}}</span> </div>

                                    </td>

                                    <td width="20%">

                                        <div class="text-right"> <span class="font-weight-bold">₹{{item.price}}</span> </div>

                                    </td>

                                </tr>

                            </tbody>

                        </table>

                    </div>

                    <div class="row d-flex justify-content-end">

                        <div class="col-md-5">

                            <table class="table table-borderless">

                                <tbody class="totals">

                                    <tr>

                                        <td>

                                            <div class="text-left"> <span class="text-muted">Subtotal</span> </div>

                                        </td>

                                        <td>

                                            <div class="text-right"> <span>₹{{total}}</span> </div>

                                        </td>

                                    </tr>

                                    <tr>

                                        <td>

                                            <div class="text-left"> <span class="text-muted">Shipping Fee</span> </div>

                                        </td>

                                        <td>

                                            <div class="text-right"> <span>₹60</span> </div>

                                        </td>

                                    </tr>

                                    <tr class="border-top border-bottom">

                                        <td>

                                            <div class="text-left"> <span class="font-weight-bold">Total</span> </div>

                                        </td>

                                        <td>

                                            <div class="text-right"> <span class="font-weight-bold">₹{{grandTotal}}</span> </div>

                                        </td>

                                    </tr>

                                </tbody>

                            </table>

                        </div>

                    </div>

                    <p>We will be sending shipping confirmation email when the item shipped successfully!</p>

                    <p class="font-weight-bold mb-0">Thanks for shopping with us!</p>

                    <button class="btn btn-dark printOrder" (click)="printOrder()">Print Order</button>

                    <button routerLink="/products" class="btn btn-dark shopMore">Shop More</button>

                </div>

            </div>

        </div>

    </div>

</div>

**order-summary.component.ts:**

import { LogService } from './../../service/log.service';

import { CartService } from './../../service/cart.service';

import { Cart } from './../../model/cart';

import { Component, OnInit } from '@angular/core';

import { Customer } from 'src/app/model/customer';

import { CustomerService } from 'src/app/service/customer.service';

import { formatDate } from '@angular/common';

import { Router } from '@angular/router';

@Component({

  selector: 'app-order-summary',

  templateUrl: './order-summary.component.html',

  styleUrls: ['./order-summary.component.css']

})

export class OrderSummaryComponent implements OnInit {

  customer:Customer=new Customer();

  cust\_email:string;

  cartItems:Cart[];

  total:number;

  grandTotal:number;

  todayDate:any;

  transId:string;

  constructor(private customerService:CustomerService,

              private cartService:CartService,

              private logService:LogService,

              private router:Router) { }

  ngOnInit(): void {

    this.cust\_email=sessionStorage.getItem('cust\_email');

    this.customerService.getCustomer(this.cust\_email).subscribe(data=>{

      this.customer=data

    })

    this.cartService.getCartItemList().subscribe(data=>{

      this.cartItems=data

    })

    this.cartService.deleteAllCart().subscribe(data=>{

      console.log('');

    })

    this.total=Number(sessionStorage.getItem('grandTotal'));

    this.grandTotal=this.total+60;

    this.todayDate=formatDate(new Date(),'dd MMM, yyyy','en');

    this.transId=sessionStorage.getItem('transId')

  }

  printOrder(){

    window.print();

  }

}

**payment-gateway.component.css:**

body {

    color: #000;

    overflow-x: hidden;

    height: 100%;

    background-color: #EF5350;

    background-repeat: no-repeat

}

.card {

    padding: 30px 25px 35px 50px;

    border-radius: 30px;

    box-shadow: 0px 4px 8px 0px #B71C1C;

    margin-top: 50px;

    margin-bottom: 50px

}

.border-line {

    border-right: 1px solid #BDBDBD

}

.text-sm {

    font-size: 13px

}

.text-md {

    font-size: 18px

}

.image {

    width: 60px;

    height: 30px

}

::placeholder {

    color: grey;

    opacity: 1

}

:-ms-input-placeholder {

    color: grey

}

::-ms-input-placeholder {

    color: grey

}

input {

    padding: 2px 0px;

    border: none;

    border-bottom: 1px solid lightgrey;

    margin-bottom: 5px;

    margin-top: 2px;

    box-sizing: border-box;

    color: #000;

    font-size: 16px;

    letter-spacing: 1px;

    font-weight: 500

}

input:focus {

    -moz-box-shadow: none !important;

    -webkit-box-shadow: none !important;

    box-shadow: none !important;

    border-bottom: 1px solid #EF5350;

    outline-width: 0

}

button:focus {

    -moz-box-shadow: none !important;

    -webkit-box-shadow: none !important;

    box-shadow: none !important;

    outline-width: 0

}

.btn-red {

    background-color: #EF5350;

    color: #fff;

    padding: 8px 25px;

    border-radius: 50px;

    font-size: 18px;

    letter-spacing: 2px;

    border: 2px solid #fff

}

.btn-red:hover {

    box-shadow: 0 0 0 2px #EF5350

}

.btn-red:focus {

    box-shadow: 0 0 0 2px #EF5350 !important

}

.custom-checkbox .custom-control-input:checked~.custom-control-label::before {

    background-color: #EF5350

}

@media screen and (max-width: 575px) {

    .border-line {

        border-right: none;

        border-bottom: 1px solid #EEEEEE

    }

}

**payment-gateway.component.html:**

<div class="container-fluid px-1 px-md-2 px-lg-4 py-5 mx-auto">

    <div class="row d-flex justify-content-center">

        <div class="col-xl-7 col-lg-8 col-md-9 col-sm-11">

            <div class="card border-0">

                <div class="row justify-content-center">

                    <h3 class="mb-4">Credit Card Checkout</h3>

                </div>

                <div class="row">

                    <div class="col-sm-7 border-line pb-3">

                        <div class="form-group">

                            <p class="text-muted text-sm mb-0">Name on the card</p> <input type="text" name="name" placeholder="Name" size="15">

                        </div>

                        <div class="form-group">

                            <p class="text-muted text-sm mb-0">Card Number</p>

                            <div class="row px-3"> <input type="text" name="card-num" placeholder="0000 0000 0000 0000" size="18" id="cr\_no" minlength="19" maxlength="19">

                                <p class="mb-0 ml-3">/</p> <img class="image ml-1" src="https://i.imgur.com/WIAP9Ku.jpg">

                            </div>

                        </div>

                        <div class="form-group">

                            <p class="text-muted text-sm mb-0">Expiry date</p> <input type="text" name="exp" placeholder="MM/YY" size="6" id="exp" minlength="5" maxlength="5">

                        </div>

                        <div class="form-group">

                            <p class="text-muted text-sm mb-0">CVV/CVC</p> <input type="password" name="cvv" placeholder="000" size="1" minlength="3" maxlength="3">

                        </div>

                    </div>

                    <div class="col-sm-5 text-sm-center justify-content-center pt-4 pb-4"> <small class="text-sm text-muted">Transaction Id</small>

                        <h5 class="mb-5">{{transId}}</h5> <small class="text-sm text-muted">Payment amount</small>

                        <div class="row px-3 justify-content-sm-center">

                            <h2 class=""><span class="text-md mr-2">₹</span><span class="text-danger">{{totalCost}}</span></h2>

                        </div>

                        <h6 class="text-danger">Delivery charges included</h6>

                        <button type="submit" (click)="buyProducts()" class="btn btn-red text-center mt-4" id="pay">PAY</button>

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

**payment-gateway.component.ts:**

import { LogService } from './../../service/log.service';

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { PurchaseService } from 'src/app/service/purchase.service';

@Component({

  selector: 'app-payment-gateway',

  templateUrl: './payment-gateway.component.html',

  styleUrls: ['./payment-gateway.component.css']

})

export class PaymentGatewayComponent implements OnInit {

  totalCost:any;

  cust\_email:string;

  transId:string;

  buyProdMap={email:'',transactionId:''};

  constructor(private logService:LogService,

              private router:Router,

              private purchaseService:PurchaseService) { }

  ngOnInit(): void {

    this.logService.loggeduserId$.subscribe(

      (id)=>{

        if(id==''){

            this.router.navigate(['']);

        }});

    this.totalCost=60+Number(sessionStorage.getItem('grandTotal'));

    this.cust\_email=sessionStorage.getItem('cust\_email');

    this.generateTransId();

    this.buyProdMap.email=this.cust\_email;

    this.buyProdMap.transactionId=this.transId

  }

  buyProducts(){

    this.purchaseService.buyProducts(this.buyProdMap).subscribe(data=>{

      console.log('products added to Purchase');

    })

    this.router.navigate(['/orderSummary']);

  }

  generateTransId(){

    var num1 = Math.floor(Math.random() \* (999999 - 100000)) + 100000;

    var num2 = Math.floor(Math.random() \* (999999 - 100000)) + 100000;

    this.transId='TFIB'+(num1.toString())+(num2.toString());

    sessionStorage.setItem('transId',this.transId);

  }

}

**product.component.css:**

.card-top{

    position: relative;

    display: flex;

    flex-direction: column;

    min-width: 0;

    word-wrap: break-word;

    background: #fff;

    background-clip: border-box;

    border-radius: .25rem;

    border:1px solid rgba(0, 0, 0, 0.2)

}

.card{

    padding: 20px;

    margin: 15px auto;

}

.navSec a:hover{

   color: #bf1e2e;

    cursor: pointer;

}

.card img{

    width: 250px;

    height:200px;

    margin-bottom:15px;

    transition: 0.3s ease-in-out;

}

.card img:hover{

    transition: 0.3s ease-in-out;

    transform: scale(1.1);

}

.form-control{

    border-radius: 3px;

    width: 350px;

    margin-left: 50px;

    margin-top: 10px;

}

.carousel-item {

    height:300px;

    min-height: 350px;

    background: no-repeat center center scroll;

    -webkit-background-size: cover;

    -moz-background-size: cover;

    -o-background-size: cover;

    background-size: cover

}

/\* .addcart{

    position: absolute;

    bottom: 5px;

    display: table;

} \*/

**product.component.html:**

<div id="carouselExampleSlidesOnly" class="carousel slide" data-interval="3000" data-ride="carousel">

    <div class="carousel-inner">

        <div class="carousel-item active">

            <img class="d-block w-100" src="./assets/images/CCar.jpg" alt="First slide">

        </div>

        <div class="carousel-item">

            <img class="d-block w-100" src="./assets/images/ICar.jpg" alt="Second slide">

        </div>

        <div class="carousel-item">

            <img class="d-block w-100" src="./assets/images/InCar.jpg" alt="Third slide">

        </div>

        <div class="carousel-item">

            <img class="d-block w-100" src="./assets/images/mCar.jpg" alt="Fourth slide">

        </div>

    </div>

</div>

<div class="card-top container-fluid">

    <div class="container d-flex">

        <div class="container d-flex">

            <nav class="navbar navbar-expand-lg navbar-light col-md-7">

                <div class="navSec">

                    <a (click)="getProducts()" id="allProducts" class="navbar-brand">All Products</a>

                </div>

                <div class="navSec">

                    <a (click)="categoryChinese()" id="chinese" class="navbar-brand">Chinese</a>

                </div>

                <div class="navSec">

                    <a (click)="categoryIndian()" id="indian" class="navbar-brand">Indian</a>

                </div>

                <div class="navSec">

                    <a (click)="categoryMexican()" class="navbar-brand">Mexican</a>

                </div>

                <div class="navSec">

                    <a (click)="categoryItalian()" class="navbar-brand">Italian</a>

                </div>

            </nav>

            <div class="form-group col-md-3">

                <input type="text" class="form-control" name="searchKey" [(ngModel)]="searchKey"

                    (ngModelChange)="Search()" placeholder="Search Product">

            </div>

        </div>

    </div>

</div>

<div class="container">

    <div class="row">

        <ng-container \*ngFor="let item of products">

            <div class="card col-md-3 flex-column d-flex">

                <img src="{{item.imagepath}}" alt="">

                <h2>{{item.name}} </h2>

                <p>{{item.desc}}</p>

                <p><strong>Price: </strong>₹{{item.actualPrice}}

                    <span \*ngIf="item.discount>0" style="color: red;">({{item.discount}}% Discount)</span>

                </p>

                <div \*ngIf="item.discount>0">

                    <p><strong>After Discount Price: </strong>₹{{item.price}}</p>

                </div>

                <div \*ngIf="item.discount==0">

                    <p></p>

                    <br>

                </div>

                <button \*ngIf="login==0" class="align-self-end mt-auto btn addcart btn-primary"

                    (click)="showToLogin()">Add to Cart</button>

                <button \*ngIf="login==1" class="align-self-end mt-auto btn addcart btn-primary"

                    (click)="addToCart(item)" id="addToCart">Add to Cart</button>

            </div>

        </ng-container>

    </div>

</div>

**product.component.ts:**

import { LogService } from './../../service/log.service';

import { Cart } from './../../model/cart';

import { CartService } from './../../service/cart.service';

import { Product } from './../../model/product';

import { Component, OnInit, ɵCompiler\_compileModuleAndAllComponentsAsync\_\_POST\_R3\_\_ } from '@angular/core';

import { ProductService } from 'src/app/service/product.service';

import { MatSnackBar, MatSnackBarConfig } from '@angular/material/snack-bar';

@Component({

  selector: 'app-product',

  templateUrl: './product.component.html',

  styleUrls: ['./product.component.css']

})

export class ProductComponent implements OnInit {

  searchKey:string;

  cart:Cart=new Cart();

  login:number=0;

  products:Product[];

  setAutoHide: boolean = true;

  autoHide: number = 2000;

  constructor(private productService:ProductService,

    private cartService: CartService,

    private logService:LogService,

    public snackBar:MatSnackBar) { }

  ngOnInit(): void {

    this.logService.sendHeader(1);

    this.getProducts();

    this.productService.login.subscribe(res=>{

      this.login=res;

      console.log(this.login);

    })

  }

  public getProducts(){

    this.productService.getProductList().subscribe(data=>{

      this.products=data;

    })

  }

  addToCart(product:any) {

    this.cart.product=product;

    this.cartService.addToCart(this.cart).subscribe(data=>{

      console.log(data);

    });

    let config = new MatSnackBarConfig();

    config.duration = this.setAutoHide ? this.autoHide : 0;

    this.snackBar.open('Product added to cart, View cart to Checkout',true ? 'Ok' : undefined, config);

  }

  Search(){

    if(this.searchKey==""){

      this.ngOnInit();

    }else{

      this.productService.getProductSearch(this.searchKey).subscribe(data=>{

        this.products=data;

      })

    }

  }

  categoryChinese(){

    this.productService.getChinese().subscribe(data=>{

      this.products=data;

    })

  }

  categoryIndian(){

    this.productService.getIndian().subscribe(data=>{

      this.products=data;

    })

  }

  categoryMexican(){

    this.productService.getMexican().subscribe(data=>{

      this.products=data;

    })

  }

  categoryItalian(){

    this.productService.getItalian().subscribe(data=>{

      this.products=data;

    })

  }

  showToLogin(){

    alert("Please login or Register to continue shopping")

  }

}

**model:**

**admin.ts:**

export class Admin{

    username:string;

    password:string;

}

**cart.ts:**

import { Product } from "./product";

export class Cart{

    id:number;

    quantity:number;

    price:number;

    product:Product;

}

**customer.ts:**

export class Customer{

    email:string;

    password:string;

    name:string;

    contact:string;

    address:string;

}

**product.ts:**

export class Product{

    id:number;

    name:string;

    desc:string;

    category:string;

    actualPrice:number;

    discount:number;

    price:number;

    avail:string;

    imagepath:string;

}

**purchase.ts:**

import { Customer } from "./customer";

export class Purchase {

    id:number;

    dop:Date;

    productname:string;

    quantity:number;

    totalcost:number;

    transactionid:string;

    customer:Customer;

}

**admin.service.ts:**

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AdminService {

  private baseURL="http://localhost:8084/admin"

  constructor(private httpClient:HttpClient) { }

  adminLogin(loginData:any):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}/${loginData.username}`,loginData);

  }

}

**cart.service.ts:**

import { HttpClient } from '@angular/common/http';

import { Cart } from './../model/cart';

import { Injectable } from '@angular/core';

import { BehaviorSubject, Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class CartService {

  public cartItemList:any=[]

  public productList = new BehaviorSubject<any>([]);

  public cart:Cart;

  private baseURL = "http://localhost:8084/carts";

  constructor(private httpClient:HttpClient) { }

  addToCart(cart:any):Observable<any>{

    return this.httpClient.post(`${this.baseURL}`, cart);

  }

  getProduct(){

    return this.productList.asObservable();

  }

  getCartItemList():Observable<Cart[]>{

    return this.httpClient.get<Cart[]>(`${this.baseURL}`);

  }

  deleteItem(id:any):Observable<Cart>{

    return this.httpClient.delete<Cart>(`${this.baseURL}/${id}`);

  }

  deleteAllCart():Observable<Object>{

    return this.httpClient.delete<Cart[]>(`${this.baseURL}`);

  }

  lessOneCart(id:any,cart:Cart):Observable<Object>{

    return this.httpClient.put<Cart>(`${this.baseURL}/minus/${id}`,cart);

  }

  addOneCart(id:any,cart:Cart):Observable<Object>{

    return this.httpClient.put<Cart>(`${this.baseURL}/add/${id}`,cart);

  }

}

**Customer.service.ts:**

import { Observable } from 'rxjs';

import { Customer } from './../model/customer';

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

@Injectable({

  providedIn: 'root'

})

export class CustomerService {

  private baseURL="http://localhost:8084/customers";

  constructor(private httpClient:HttpClient) { }

  addCustomer(customer:Customer):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}`,customer);

  }

  customerLogin(loginData:any):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}/${loginData.email}`,loginData);

  }

  getCustomers():Observable<Customer[]>{

    return this.httpClient.get<Customer[]>(`${this.baseURL}`);

  }

  searchCustomer(keyword:any):Observable<Customer[]>{

    return this.httpClient.get<Customer[]>(`${this.baseURL}/search/${keyword}`);

  }

  deleteCustomer(email:string):Observable<Customer>{

    return this.httpClient.delete<Customer>(`${this.baseURL}/${email}`);

  }

  getCustomer(cust\_email:string):Observable<Customer>{

    return this.httpClient.get<Customer>(`${this.baseURL}/${cust\_email}`);

  }

}

**log.service.ts:**

import { Injectable } from '@angular/core';

import { BehaviorSubject } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class LogService {

  private logUserIdsource=new BehaviorSubject<String>('');

  loggeduserId$=this.logUserIdsource.asObservable();

  private showHeader=new BehaviorSubject<number>(1);

  headerId$=this.showHeader.asObservable();

  private logAdminsource=new BehaviorSubject<String>('');

  loggedAdminId$=this.logAdminsource.asObservable();

  constructor() { }

  public sendId(loggedId :string){

    this.logUserIdsource.next(loggedId);

 }

 public sendHeader(isHeader:number){

   this.showHeader.next(isHeader);

 }

 public sendAdmin(loggedUname:string){

  this.logAdminsource.next(loggedUname)

 }

}

**product.service.ts:**

import { Product } from './../model/product';

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { BehaviorSubject,Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class ProductService {

  public login = new BehaviorSubject<any>([]);

  private baseURL = "http://localhost:8084/products";

  private adminURL= "http://localhost:8084/AdminProducts";

  constructor(private httpClient: HttpClient) { }

  getLogin(){

    return this.login.asObservable();

  }

  getProductList():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/cust`);

  }

  public getProductById(id : number) : Observable<Product> {

    return this.httpClient.get<Product>(`${this.baseURL}/${id}`);

  }

  public getProductSearch(keyword:string):Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/search/${keyword}`);

  }

  getChinese():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/chinese`);

  }

  getIndian():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/indian`);

  }

  getMexican():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/mexican`);

  }

  getItalian():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/italian`);

  }

  getFullProductList():Observable<Product[]>{

    return this.httpClient.get<Product[]>(`${this.baseURL}/Admin`);

  }

  addProduct(product:Product):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}`,product);

  }

  updateProduct(id:number,product:Product):Observable<Object>{

    return this.httpClient.put<Product>(`${this.baseURL}/${id}`,product);

  }

  deleteProduct(id:number):Observable<Product>{

    return this.httpClient.delete<Product>(`${this.baseURL}/${id}`);

  }

}

**purchase.service.ts:**

import { Purchase } from 'src/app/model/purchase';

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class PurchaseService {

  private baseURL="http://localhost:8084/purchase";

  constructor( private httpClient:HttpClient) { }

  getCustomerOrders(email:string):Observable<Purchase[]>{

    return this.httpClient.get<Purchase[]>(`${this.baseURL}/byEmail/${email}`);

  }

  getAllPurchase():Observable<Purchase[]>{

    return this.httpClient.get<Purchase[]>(`${this.baseURL}`);

  }

  deletePurchase(id:any):Observable<Purchase>{

    return this.httpClient.delete<Purchase>(`${this.baseURL}/${id}`);

  }

  searchPurchase(keyword:string):Observable<Purchase[]>{

    return this.httpClient.get<Purchase[]>(`${this.baseURL}/search/${keyword}`);

  }

  buyProducts(buyProdMap:any):Observable<Object>{

    return this.httpClient.post(`${this.baseURL}`,buyProdMap);

  }

}

**main.ts:**

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

  enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule)

  .catch(err => console.error(err));

**tests.css:**

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

  context(path: string, deep?: boolean, filter?: RegExp): {

    keys(): string[];

    <T>(id: string): T;

  };

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting()

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

**styles.css:**

/\* You can add global styles to this file, and also import other style files \*/

html, body { height: 100%; }

body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif;background-color:

    #eef2f5 }

hr { border-top: 1px dashed grey;}