**Source Code (Backend-SpringBoot)**

**package com.medicare.models**

**File: - Address.java**

**package** com.medicare.models;

**import** javax.persistence.\*;

@Entity

@Table(name = "address")

**public** **class** Address {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id", nullable = **false**)

**private** Long id;

**private** String city;

**private** String street;

**private** String landmark;

**private** String pincode;

@OneToOne

@JoinColumn(name = "user\_id")

**private** User user;

**public** Address() {

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getStreet() {

**return** street;

}

**public** **void** setStreet(String street) {

**this**.street = street;

}

**public** String getLandmark() {

**return** landmark;

}

**public** **void** setLandmark(String landmark) {

**this**.landmark = landmark;

}

**public** String getPincode() {

**return** pincode;

}

**public** **void** setPincode(String pincode) {

**this**.pincode = pincode;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

}

**File: - Authority.java**

**package** com.medicare.models;

**import** org.springframework.security.core.GrantedAuthority;

**public** **class** Authority **implements** GrantedAuthority {

**private** String authority;

**public** Authority(String authority) {

**this**.authority = authority;

}

@Override

**public** String getAuthority() {

**return** **this**.authority;

}

}

**File: - Cart.java**

package com.medicare.models;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "cart")

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

@ManyToMany(cascade = CascadeType.ALL)

@JoinTable(name = "cart\_products",

joinColumns = {@JoinColumn(name = "cart\_id")},

inverseJoinColumns = {@JoinColumn(name = "product\_id")})

private Set<Product> products = new HashSet<>();

@OneToOne

@JoinColumn(name = "user\_id")

private User user;

public Cart() {

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Set<Product> getProducts() {

return products;

}

public void setProducts(Set<Product> products) {

this.products = products;

}

}

**File: Category.java**

package com.medicare.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "category")

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

private String category\_name;

private String category\_description;

private Boolean popular;

@OneToMany(mappedBy = "category",cascade = CascadeType.ALL)

@JsonIgnore

private Set<Product> products = new HashSet<>();

public Category() {

}

public Category(Long id, String category\_name, String category\_description, Set<Product> products) {

this.id = id;

this.category\_name = category\_name;

this.category\_description = category\_description;

this.products = products;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getCategory\_name() {

return category\_name;

}

public void setCategory\_name(String category\_name) {

this.category\_name = category\_name;

}

public String getCategory\_description() {

return category\_description;

}

public void setCategory\_description(String category\_description) {

this.category\_description = category\_description;

}

public Set<Product> getProducts() {

return products;

}

public void setProducts(Set<Product> products) {

this.products = products;

}

public Boolean getPopular() {

return popular;

}

public void setPopular(Boolean popular) {

this.popular = popular;

}

}

**File: JwtRequest.java**

**package** com.medicare.models;

**public** **class** JwtRequest {

String username;

String password;

**public** JwtRequest() {

}

**public** JwtRequest(String username, String password) {

**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;

}

}

**File:- JwtResponse.java**

**package** com.medicare.models;

**public** **class** JwtResponse {

String token;

**public** JwtResponse() {

}

**public** JwtResponse(String token) {

**this**.token = token;

}

**public** String getToken() {

**return** token;

}

**public** **void** setToken(String token) {

**this**.token = token;

}

}

**File:- Notification.java**

**package** com.medicare.models;

**import** javax.persistence.\*;

@Entity

@Table(name = "notification")

**public** **class** Notification {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id", nullable = **false**)

**private** Long id;

**private** String msg;

**private** **boolean** checked = **false**;

@ManyToOne

@JoinColumn(name = "user\_id")

**private** User user;

**public** Notification() {

}

**public** String getMsg() {

**return** msg;

}

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

**public** Boolean getChecked() {

**return** checked;

}

**public** **void** setChecked(Boolean checked) {

**this**.checked = checked;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

}

**File:- Product.java**

package com.medicare.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "product")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

private String product\_name;

@Column(length = 5000)

private String product\_description;

private Double product\_price;

private byte product\_discount;

private Double product\_selling\_price;

private Long product\_stock;

@ManyToOne()

@JoinColumn(name = "category\_id")

private Category category;

@ManyToMany(cascade = CascadeType.ALL)

@JoinTable(name = "product\_images",

joinColumns = {@JoinColumn(name = "product\_id")},

inverseJoinColumns = {@JoinColumn(name = "image\_id")})

private Set<ProductImage> product\_images;

public Product() {

}

public Product(Long id, String product\_name, String product\_description, Double product\_price, byte product\_discount, Long product\_stock, Category category) {

this.id = id;

this.product\_name = product\_name;

this.product\_description = product\_description;

this.product\_price = product\_price;

this.product\_discount = product\_discount;

this.product\_selling\_price = product\_price - (product\_price/100) \* product\_discount;

this.product\_stock = product\_stock;

this.category = category;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getProduct\_name() {

return product\_name;

}

public void setProduct\_name(String product\_name) {

this.product\_name = product\_name;

}

public String getProduct\_description() {

return product\_description;

}

public void setProduct\_description(String product\_description) {

this.product\_description = product\_description;

}

public Double getProduct\_price() {

return product\_price;

}

public void setProduct\_price(Double product\_price) {

this.product\_price = product\_price;

}

public byte getProduct\_discount() {

return product\_discount;

}

public void setProduct\_discount(byte product\_discount) {

this.product\_discount = product\_discount;

}

public Double getProduct\_selling\_price() {

return product\_selling\_price;

}

public Long getProduct\_stock() {

return product\_stock;

}

public void setProduct\_stock(Long product\_stock) {

this.product\_stock = product\_stock;

}

public Category getCategory() {

return category;

}

public void setCategory(Category category) {

this.category = category;

}

public Set<ProductImage> getProduct\_images() {

return product\_images;

}

public void setProduct\_images(Set<ProductImage> product\_images) {

this.product\_images = product\_images;

}

}

**File:- ProductImage.java**

**package** com.medicare.models;

**import** javax.persistence.\*;

@Entity

@Table(name = "product\_image")

**public** **class** ProductImage {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id", nullable = **false**)

**private** Long id;

**private** String name;

**private** String type;

@Column(length = 50000000)

@Lob

**private** **byte**[] imageByte;

**public** ProductImage() {

}

**public** ProductImage(String name, String type, **byte**[] imageByte) {

**this**.name = name;

**this**.type = type;

**this**.imageByte = imageByte;

}

**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 getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** **byte**[] getImageByte() {

**return** imageByte;

}

**public** **void** setImageByte(**byte**[] imageByte) {

**this**.imageByte = imageByte;

}

}

**File:- Role.java**

package com.medicare.models;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "role")

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

private String role\_name;

@OneToMany(cascade = CascadeType.ALL,mappedBy = "role")

private Set<UserRole> userRoles= new HashSet<>();

public Role() {

}

public Role(Long id, String role\_name) {

this.id = id;

this.role\_name = role\_name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getRole\_name() {

return role\_name;

}

public void setRole\_name(String role\_name) {

this.role\_name = role\_name;

}

public Set<UserRole> getUserRoles() {

return userRoles;

}

public void setUserRoles(Set<UserRole> userRoles) {

this.userRoles = userRoles;

}

}

**File:- User.java**

package com.medicare.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;

import com.medicare.config.CustomAuthorityDeserializer;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.\*;

import java.util.Collection;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "users")

public class User implements UserDetails {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

private String username;

private String password;

private String first\_name;

private String last\_name;

private String user\_email;

private String user\_phone;

private boolean enabled=true;

@OneToMany(mappedBy = "user")

private Set<Notification> notifications = new HashSet<>();

@OneToMany(cascade = CascadeType.ALL,mappedBy = "user",fetch = FetchType.EAGER)

@JsonIgnore

private Set<UserRole> userRoles = new HashSet<>();

public User() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

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;

}

public String getFirst\_name() {

return first\_name;

}

public void setFirst\_name(String first\_name) {

this.first\_name = first\_name;

}

public String getLast\_name() {

return last\_name;

}

public void setLast\_name(String last\_name) {

this.last\_name = last\_name;

}

public String getUser\_email() {

return user\_email;

}

public void setUser\_email(String user\_email) {

this.user\_email = user\_email;

}

public String getUser\_phone() {

return user\_phone;

}

public void setUser\_phone(String user\_phone) {

this.user\_phone = user\_phone;

}

public boolean isEnabled() {

return enabled;

}

public void setEnabled(boolean enabled) {

this.enabled = enabled;

}

public Set<UserRole> getUserRoles() {

return userRoles;

}

public void setUserRoles(Set<UserRole> userRoles) {

this.userRoles = userRoles;

}

@Override

public String toString() {

return "User{" +

"id=" + id +

", username='" + username + '\'' +

", password='" + password + '\'' +

", first\_name='" + first\_name + '\'' +

", last\_name='" + last\_name + '\'' +

", user\_email='" + user\_email + '\'' +

", user\_phone='" + user\_phone + '\'' +

", enabled=" + enabled +

", userRoles=" + userRoles +

'}';

}

@JsonDeserialize(using = CustomAuthorityDeserializer.class)

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

Set<Authority> authoritySet=new HashSet<>();

this.userRoles.forEach(userRole -> {

authoritySet.add(new Authority(userRole.getRole().getRole\_name()));

});

return authoritySet;

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

}

**File:- UserOrders.java**

package com.medicare.models;

import javax.persistence.\*;

import java.sql.Date;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "user\_order")

public class UserOrders {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Long id;

@OneToOne

@JoinColumn(name = "user\_id")

private User user;

private String payment\_status;

private String order\_status;

@OneToOne

@JoinColumn(name = "address\_id")

private Address address;

private Boolean isPending;

private Date order\_date;

private Double total;

private Date delivery\_date;

@ManyToMany(cascade = CascadeType.ALL)

@JoinTable(name = "order\_products",

joinColumns = {@JoinColumn(name = "order\_id")},

inverseJoinColumns = {@JoinColumn(name = "product\_id")})

private Set<Product> products = new HashSet<>();

public UserOrders() {

}

public String getPayment\_status() {

return payment\_status;

}

public void setPayment\_status(String payment\_status) {

this.payment\_status = payment\_status;

}

public String getOrder\_status() {

return order\_status;

}

public void setOrder\_status(String order\_status) {

this.order\_status = order\_status;

}

public Date getOrder\_date() {

return order\_date;

}

public void setOrder\_date(Date order\_date) {

this.order\_date = order\_date;

}

public Set<Product> getProducts() {

return products;

}

public void setProducts(Set<Product> products) {

this.products = products;

}

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Boolean getPending() {

return isPending;

}

public void setPending(Boolean pending) {

isPending = pending;

}

@Override

public String toString() {

return "Order{" +

"id=" + id +

", user=" + user +

", payment\_status='" + payment\_status + '\'' +

", order\_status='" + order\_status + '\'' +

", address=" + address +

", order\_date=" + order\_date +

", products=" + products +

'}';

}

public Double getTotal() {

return total;

}

public void setTotal(Double total) {

this.total = total;

}

public Date getDelivery\_date() {

return delivery\_date;

}

public void setDelivery\_date(Date delivery\_date) {

this.delivery\_date = delivery\_date;

}

}

**File:- UserRole.java**

**package** com.medicare.models;

**import** javax.persistence.\*;

@Entity

@Table(name = "user\_role")

**public** **class** UserRole {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id", nullable = **false**)

**private** Long id;

@ManyToOne(cascade = CascadeType.***ALL***)

**private** User user;

@ManyToOne(cascade = CascadeType.***ALL***)

**private** Role role;

**public** UserRole() {

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** Role getRole() {

**return** role;

}

**public** **void** setRole(Role role) {

**this**.role = role;

}

}

**Package:- com.medicare.repositories**

**File:- AddressRepository.java**

package com.medicare.repositories;

import com.medicare.models.Address;

import com.medicare.models.User;

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

import java.util.Set;

public interface AddressRepository extends JpaRepository<Address, Long> {

public Set<Address> findByUser(User user);

}

**File:- CartRepository.java**

package com.medicare.repositories;

import com.medicare.models.Cart;

import com.medicare.models.User;

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

public interface CartRepository extends JpaRepository<Cart,Long> {

public Cart findCartByUser(User user);

}

**File:- CategoryRepository.java**

package com.medicare.repositories;

import com.medicare.models.Category;

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

import java.util.Set;

public interface CategoryRepository extends JpaRepository<Category, Long> {

public Set<Category> findByPopular(Boolean value);

}

**File:- NotificationRepository.java**

package com.medicare.repositories;

import com.medicare.models.Notification;

import com.medicare.models.User;

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

import java.util.Set;

public interface NotificationRepository extends JpaRepository<Notification, Long> {

public Set<Notification> findByUser(User user);

}

**File:- OrderRepository.java**

package com.medicare.repositories;

import com.medicare.models.User;

import com.medicare.models.UserOrders;

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

import java.util.Set;

public interface OrderRepository extends JpaRepository<UserOrders, Long> {

public Set<UserOrders> findByUser(User user);

public Set<UserOrders> findByIsPending(Boolean isPending);

}

**File:- ProductRepository.java**

package com.medicare.repositories;

import com.medicare.models.Category;

import com.medicare.models.Product;

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

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

import org.springframework.data.repository.query.Param;

import java.util.Set;

public interface ProductRepository extends JpaRepository<Product, Long> {

public Set<Product> findByCategory(Category category);

@Query("SELECT p FROM Product p WHERE p.product\_name LIKE :product\_name%")

public Set<Product> findByProduct\_NameStartLike(@Param("product\_name") String product\_name);

@Query("SELECT p FROM Product p WHERE p.product\_name LIKE %:product\_name%")

public Set<Product> findByProduct\_NameLike(@Param("product\_name") String product\_name);

}

**File:- RoleRepository.java**

package com.medicare.repositories;

import com.medicare.models.Role;

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

import org.springframework.stereotype.Repository;

@Repository

public interface RoleRepository extends JpaRepository<Role, Long> {

}

**File:- UserRepository.java**

package com.medicare.repositories;

import com.medicare.models.User;

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

import org.springframework.stereotype.Repository;

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

public User findByUsername(String username);

}

**Package:- com.medicare.services**

**File:- AddressService.java**

package com.medicare.services;

import com.medicare.models.Address;

import com.medicare.models.User;

import java.util.Set;

public interface AddressService {

public Address addAddress(Address address);

public Address updateAddress(Address updatedAddress);

public void deleteAddress(Long address\_id);

public Set<Address> getAddressesOfUser(User user);

public Address getAddress(Long address\_id);

}

**File:- CartService.java**

package com.medicare.services;

import com.medicare.models.Cart;

import com.medicare.models.User;

public interface CartService {

public Cart createCart(Cart cart);

public Cart updateCart(Cart cart);

public Cart getCart(User user);

public void deleteCart(Long cart\_id);

public void deleteProductFromCart(Long cart\_id,Long product\_id);

public void deleteProductsFromCart(Long cart\_id);

}

**File:- CategoryService.java**

package com.medicare.services;

import com.medicare.models.Category;

import java.util.Set;

public interface CategoryService {

public Category addCategory(Category category);

public void deleteCategory(Long category\_id);

public Category updateCategory(Category updatedCategory);

public Category getCategory(Long category\_id);

public Set<Category> getAllCategories();

public Set<Category> getPopularCategories();

}

**File:- OrderService.java**

package com.medicare.services;

import com.medicare.models.Product;

import com.medicare.models.User;

import com.medicare.models.UserOrders;

import java.util.Set;

public interface OrderService {

public UserOrders createOrder(UserOrders order);

public Set<UserOrders> getOrders(User user);

public Set<UserOrders> getAllOrders();

public Set<UserOrders> getAllPendingOrders();

public Set<Product> getProducts(Long user\_order\_id);

public UserOrders completeOrder(Long order\_id);

}

**File:- ProductService.java**

package com.medicare.services;

import com.medicare.models.Category;

import com.medicare.models.Product;

import java.util.Set;

public interface ProductService {

public Product addProduct(Product product);

public Product updateProduct(Product updatedProduct);

public Product getProduct(Long product\_id);

public Set<Product> getAllProducts();

public Set<Product> getProductsOfCategory(Category category);

public void deleteProduct(Long product\_id);

}

**File:- UserService.java**

package com.medicare.services;

import com.medicare.models.User;

import com.medicare.models.UserRole;

import java.util.Set;

public interface UserService {

public User createUser(User user, Set<UserRole> userRoles) throws Exception;

public User getUserByUsername(String username);

public User getUserById(Long user\_id);

public void deleteUser(Long user\_id);

public User updateUser(User updatedUser);

public Set<User> getUsers();

}

**Package:- com.medicare.services.impl**

**File:- AddressServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.Address;

import com.medicare.models.User;

import com.medicare.repositories.AddressRepository;

import com.medicare.services.AddressService;

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

import org.springframework.stereotype.Service;

import java.util.HashSet;

import java.util.Set;

@Service

public class AddressServiceImpl implements AddressService {

@Autowired

private AddressRepository addressRepository;

@Override

public Address addAddress(Address address) {

return addressRepository.save(address);

}

@Override

public Address updateAddress(Address updatedAddress) {

return addressRepository.save(updatedAddress);

}

@Override

public void deleteAddress(Long address\_id) {

addressRepository.deleteById(address\_id);

}

@Override

public Set<Address> getAddressesOfUser(User user) {

return new HashSet<>(addressRepository.findByUser(user));

}

@Override

public Address getAddress(Long address\_id) {

return addressRepository.findById(address\_id).get();

}

}

**File:- CartServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.Cart;

import com.medicare.models.Product;

import com.medicare.models.User;

import com.medicare.repositories.CartRepository;

import com.medicare.services.CartService;

import com.medicare.services.ProductService;

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

import org.springframework.stereotype.Service;

import java.util.HashSet;

import java.util.Set;

@Service

public class CartServiceImpl implements CartService {

@Autowired

private CartRepository cartRepository;

@Autowired

private ProductService productService;

@Override

public Cart createCart(Cart cart) {

return cartRepository.save(cart);

}

@Override

public Cart updateCart(Cart cart) {

return cartRepository.save(cart);

}

@Override

public Cart getCart(User user) {

return cartRepository.findCartByUser(user);

}

@Override

public void deleteCart(Long cart\_id) {

cartRepository.deleteById(cart\_id);

}

@Override

public void deleteProductFromCart(Long cart\_id,Long product\_id) {

Cart cart = cartRepository.findById(cart\_id).get();

Product product = productService.getProduct(product\_id);

// Set<Product> product\_list = new HashSet<>(cart.getProducts()) ;

boolean removed = cart.getProducts().remove(product);

if(removed){

System.out.println("Item Deleted");

cartRepository.save(cart);

}else {

System.out.println("Not deleted");

}

}

@Override

public void deleteProductsFromCart(Long cart\_id) {

Cart cart = cartRepository.findById(cart\_id).get();

cart.getProducts().clear();

cartRepository.save(cart);

}

}

**File:- CategoryServiceImpl.java**

**package** com.medicare.services.impl;

**import** com.medicare.models.Category;

**import** com.medicare.repositories.CategoryRepository;

**import** com.medicare.services.CategoryService;

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

**import** org.springframework.stereotype.Service;

**import** java.util.HashSet;

**import** java.util.Set;

@Service

**public** **class** CategoryServiceImpl **implements** CategoryService {

@Autowired

**private** CategoryRepository categoryRepository;

@Override

**public** Category addCategory(Category category) {

**return** categoryRepository.save(category);

}

@Override

**public** **void** deleteCategory(Long category\_id) {

categoryRepository.deleteById(category\_id);

}

@Override

**public** Category updateCategory(Category updatedCategory) {

**return** categoryRepository.save(updatedCategory);

}

@Override

**public** Category getCategory(Long category\_id) {

**return** categoryRepository.findById(category\_id).get();

}

@Override

**public** Set<Category> getAllCategories() {

**return** **new** HashSet<>(categoryRepository.findAll());

}

@Override

**public** Set<Category> getPopularCategories() {

**return** categoryRepository.findByPopular(**true**);

}

}

**File:- OrderServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.\*;

import com.medicare.repositories.AddressRepository;

import com.medicare.repositories.NotificationRepository;

import com.medicare.repositories.OrderRepository;

import com.medicare.services.CartService;

import com.medicare.services.OrderService;

import com.medicare.services.UserService;

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

import org.springframework.stereotype.Service;

import java.sql.Date;

import java.util.HashSet;

import java.util.Set;

@Service

public class OrderServiceImpl implements OrderService {

@Autowired

private OrderRepository orderRepository;

@Autowired

private UserService userService;

@Autowired

private AddressRepository addressRepository;

@Autowired

private NotificationRepository notificationRepository;

@Autowired

private CartService cartService;

@Override

public UserOrders createOrder(UserOrders order) {

User user = userService.getUserById(order.getUser().getId());

Cart cart = cartService.getCart(user);

Set<Product> products = cart.getProducts();

Address address = addressRepository.findById(order.getAddress().getId()).get();

Date date = new Date(System.currentTimeMillis());

order.setOrder\_date(date);

order.setOrder\_status("ORDER\_PLACED");

order.setAddress(address);

order.getProducts().addAll(products);

order.setUser(user);

order.setPending(true);

return orderRepository.save(order);

}

@Override

public Set<UserOrders> getOrders(User user) {

return new HashSet<>(orderRepository.findByUser(user));

}

@Override

public Set<UserOrders> getAllOrders() {

return new HashSet<>(orderRepository.findAll());

}

@Override

public Set<UserOrders> getAllPendingOrders() {

return orderRepository.findByIsPending(true);

}

@Override

public Set<Product> getProducts(Long user\_order\_id) {

return orderRepository.findById(user\_order\_id).get().getProducts();

}

@Override

public UserOrders completeOrder(Long order\_id) {

UserOrders userOrder = orderRepository.findById(order\_id).get();

System.out.println("Order:"+userOrder);

userOrder.setOrder\_status("COMPLETED");

Date date = new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 24 \* 3 );

userOrder.setDelivery\_date(date);

userOrder.setPending(false);

User user = userOrder.getUser();

Notification notification = new Notification();

notification.setUser(user);

notification.setChecked(false);

notification.setMsg("Order Shipped With Id: "+userOrder.getId() +

" And Total Amount Of : "+Math.round(userOrder.getTotal())+

" Delivery Expected On "+date);

notificationRepository.save(notification);

orderRepository.save(userOrder);

return userOrder;

}

}

**File:- ProductServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.Category;

import com.medicare.models.Product;

import com.medicare.repositories.ProductRepository;

import com.medicare.services.ProductService;

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

import org.springframework.stereotype.Service;

import java.util.HashSet;

import java.util.Set;

@Service

public class ProductServiceImpl implements ProductService {

@Autowired

private ProductRepository productRepository;

@Override

public Product addProduct(Product product) {

System.out.println(product.getCategory().getId());

return productRepository.save(product);

}

@Override

public Product updateProduct(Product updatedProduct) {

return productRepository.save(updatedProduct);

}

@Override

public Product getProduct(Long product\_id) {

return productRepository.findById(product\_id).get();

}

@Override

public Set<Product> getAllProducts() {

return new HashSet<>(productRepository.findAll());

}

@Override

public Set<Product> getProductsOfCategory(Category category) {

return new HashSet<>(productRepository.findByCategory(category));

}

@Override

public void deleteProduct(Long product\_id) {

productRepository.deleteById(product\_id);

}

}

**File:- UserDetailsServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.User;

import com.medicare.repositories.UserRepository;

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

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

@Service

public class UserDetailsServiceImpl implements UserDetailsService {

@Autowired

private UserRepository userRepository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userRepository.findByUsername(username);

if(user == null){

System.out.println("User not found");

throw new UsernameNotFoundException("User not found with "+username);

}

return user;

}

}

**File:- UserServiceImpl.java**

package com.medicare.services.impl;

import com.medicare.models.Cart;

import com.medicare.models.User;

import com.medicare.models.UserRole;

import com.medicare.repositories.RoleRepository;

import com.medicare.repositories.UserRepository;

import com.medicare.services.CartService;

import com.medicare.services.UserService;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import java.util.HashSet;

import java.util.Set;

@Service

public class UserServiceImpl implements UserService {

@Autowired

private UserRepository userRepository;

@Autowired

private RoleRepository roleRepository;

@Autowired

private CartService cartService;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

@Override

public User createUser(User user, Set<UserRole> userRoles) throws Exception {

User local = userRepository.findByUsername(user.getUsername());

if(local != null){

System.err.println("User Already Exists");

throw new Exception("User Already Exist");

}else {

local=this.userRepository.save(user);

Cart cart = new Cart();

cart.setUser(local);

cartService.createCart(cart);

}

return local;

}

@Override

public User getUserByUsername(String username) {

return userRepository.findByUsername(username);

}

@Override

public User getUserById(Long user\_id) {

User user = userRepository.findById(user\_id).get();

return user;

}

@Override

public void deleteUser(Long user\_id) {

userRepository.deleteById(user\_id);

}

@Override

public User updateUser(User updatedUser) {

User user= userRepository.findByUsername(updatedUser.getUsername());

return userRepository.save(updatedUser);

}

@Override

public Set<User> getUsers() {

return new HashSet<>(userRepository.findAll());

}

}

**Package:- com.medicare.controllers**

**File:- AuthenticateController.java**

package com.medicare.controllers;

import com.medicare.config.JwtUtils;

import com.medicare.models.JwtRequest;

import com.medicare.models.JwtResponse;

import com.medicare.models.User;

import com.medicare.services.impl.UserDetailsServiceImpl;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.DisabledException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.web.bind.annotation.\*;

import java.security.Principal;

@RestController

@CrossOrigin("\*")

public class AuthenticateController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private UserDetailsServiceImpl userDetailsService;

@Autowired

private JwtUtils jwtUtil;

@PostMapping("/generate-token")

public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws Exception {

try {

aunthenticate(jwtRequest.getUsername(),jwtRequest.getPassword());

}catch (UsernameNotFoundException e){

throw new Exception("User not found with username");

}

UserDetails userDetails = this.userDetailsService.loadUserByUsername(jwtRequest.getUsername());

String token = this.jwtUtil.generateToken(userDetails);

return ResponseEntity.ok(new JwtResponse(token));

}

private void aunthenticate(String username , String password) throws Exception {

try {

authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username,password));

}catch (DisabledException e){

throw new Exception("User is disabled");

}catch (BadCredentialsException e){

throw new Exception("Invalid Credentials "+e.getMessage());

}

}

@GetMapping("/current-user")

public User getCurrentUser(Principal principal){

return (User) this.userDetailsService.loadUserByUsername(principal.getName());

}

}

**File:- CategoryController.java**

package com.medicare.controllers;

import com.medicare.models.Category;

import com.medicare.models.Product;

import com.medicare.services.CategoryService;

import com.medicare.services.ProductService;

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

import org.springframework.web.bind.annotation.\*;

import java.util.Set;

@RestController

@CrossOrigin("\*")

@RequestMapping("/categories")

public class CategoryController {

@Autowired

private ProductService productService;

@Autowired

private CategoryService categoryService;

// ------------- Category Service Endpoints ------------------

@PostMapping("")

public Category addCategory(@RequestBody Category category){

return categoryService.addCategory(category);

}

@PutMapping("")

public Category updateCategory(@RequestBody Category updatedCategory){

return categoryService.updateCategory(updatedCategory);

}

@GetMapping("")

public Set<Category> getCategories(){

return categoryService.getAllCategories();

}

@GetMapping("/popular")

public Set<Category> getPopularCategories(){

return categoryService.getPopularCategories();

}

@GetMapping("/{category\_id}")

public Category getCategory(@PathVariable("category\_id") Long category\_id){

return categoryService.getCategory(category\_id);

}

@GetMapping("/product-list/{category\_id}")

public Set<Product> getProductsOfCategory(@PathVariable("category\_id") Long category\_id){

Category category = categoryService.getCategory(category\_id);

return productService.getProductsOfCategory(category);

}

@DeleteMapping("/{category\_id}")

public void deleteCategory(@PathVariable("category\_id") Long category\_id){

categoryService.deleteCategory(category\_id);

}

}

**File:- ProductController.java**

package com.medicare.controllers;

import com.medicare.models.Category;

import com.medicare.models.Product;

import com.medicare.models.ProductImage;

import com.medicare.repositories.ProductRepository;

import com.medicare.services.CategoryService;

import com.medicare.services.ProductService;

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

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.util.HashSet;

import java.util.Set;

@RestController

@CrossOrigin("\*")

@RequestMapping("/products")

public class ProductController {

@Autowired

private ProductService productService;

@Autowired

private CategoryService categoryService;

@Autowired

private ProductRepository productRepository;

@PostMapping(value = "/",consumes = {MediaType.MULTIPART\_FORM\_DATA\_VALUE})

public Product addProduct(@RequestPart("product") Product product,

@RequestPart("imageFile")MultipartFile[] file){

// return productService.addProduct(product);

try{

Set<ProductImage> images = uploadImage(file);

product.setProduct\_images(images);

return productService.addProduct(product);

}catch (Exception e){

e.printStackTrace();

return null;

}

}

public Set<ProductImage> uploadImage(MultipartFile[] file) throws IOException {

Set<ProductImage> productImages = new HashSet<>();

for(MultipartFile file1:file){

ProductImage productImage = new ProductImage();

productImage.setName(file1.getOriginalFilename());

productImage.setType(file1.getContentType());

productImage.setImageByte(file1.getBytes());

productImages.add(productImage);

}

return productImages;

}

@PutMapping(value="/",consumes = {MediaType.MULTIPART\_FORM\_DATA\_VALUE})

public Product updateProduct(@RequestPart("product") Product product,

@RequestPart("imageFile")MultipartFile[] file){

try{

Set<ProductImage> images = uploadImage(file);

product.setProduct\_images(images);

return productService.updateProduct(product);

}catch (Exception e){

e.printStackTrace();

return null;

}

}

@DeleteMapping("/{product\_id}")

public void deleteProduct(@PathVariable("product\_id") Long product\_id){

productService.deleteProduct(product\_id);

}

@GetMapping("/")

public Set<Product> getAllProducts(){

return productService.getAllProducts();

}

@GetMapping("/{product\_id}")

public Product getProduct(@PathVariable("product\_id") Long product\_id){

return productService.getProduct(product\_id);

}

@GetMapping("/search/{product\_name}")

public Set<Product> searchProducts(@PathVariable("product\_name")String product\_name){

return new HashSet<>(productRepository.findByProduct\_NameLike(product\_name));

}

@GetMapping("/search-startwith/{product\_name}")

public Set<Product> searchProductsStartswith(@PathVariable("product\_name")String product\_name){

return new HashSet<>(productRepository.findByProduct\_NameStartLike(product\_name));

}

}

**File:- UserController.java**

package com.medicare.controllers;

import com.medicare.models.\*;

import com.medicare.repositories.CartRepository;

import com.medicare.repositories.NotificationRepository;

import com.medicare.repositories.ProductRepository;

import com.medicare.services.AddressService;

import com.medicare.services.CartService;

import com.medicare.services.OrderService;

import com.medicare.services.UserService;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.web.bind.annotation.\*;

import java.util.HashSet;

import java.util.Set;

@RestController

@CrossOrigin("\*")

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

@Autowired

private CartRepository cartRepository;

@Autowired

private ProductRepository productRepository;

@Autowired

private CartService cartService;

@Autowired

private AddressService addressService;

@Autowired

private OrderService orderService;

@Autowired

private NotificationRepository notificationRepository;

@PostMapping("/")

public User addUser(@RequestBody User user) throws Exception {

Role role = new Role();

role.setRole\_name("NORMAL");

Set<UserRole> userRoles = new HashSet<>();

UserRole userRole = new UserRole();

userRole.setRole(role);

userRole.setUser(user);

userRoles.add(userRole);

user.setUserRoles(userRoles);

user.setPassword(passwordEncoder.encode(user.getPassword()));

return userService.createUser(user,userRoles);

}

@GetMapping("/{username}")

public User getUserByUsername(@PathVariable("username") String username){

return userService.getUserByUsername(username);

}

@GetMapping("/")

public Set<User> getUsers(){

return userService.getUsers();

}

@PostMapping("/confirm-password")

public Boolean confirmPassword(@RequestBody User user){

User u = userService.getUserByUsername(user.getUsername());

boolean status = passwordEncoder.matches(user.getPassword(),u.getPassword());

System.out.println("User "+u);

System.out.println("Current Password "+user.getPassword());

System.out.println("username"+user.getUsername());

return status;

}

@PostMapping("/change-password")

public Boolean changePassword(@RequestBody User user){

User u = userService.getUserByUsername(user.getUsername());

u.setPassword(passwordEncoder.encode(user.getPassword()));

userService.updateUser(u);

return true;

}

@GetMapping("/user/{user\_id}")

public User getUser(@PathVariable("user\_id")Long user\_id){

User user = userService.getUserById(user\_id);

System.out.println(user);

return user;

}

@DeleteMapping("/{user\_id}")

public void deleteUser(@PathVariable("user\_id")Long user\_id){

userService.deleteUser(user\_id);

}

@PutMapping("/")

public User updateUser(@RequestBody User user) {

return userService.updateUser(user);

}

@PostMapping("/cart/{cart\_id}")

public Product addToCart(@PathVariable("cart\_id") Long cart\_id,@RequestBody Product product){

System.out.println("Add To Cart "+cart\_id);

Cart cart = cartRepository.findById(cart\_id).get();

System.out.println(cart.getId());

Product Iproduct = productRepository.findById(product.getId()).get();

cart.getProducts().add(Iproduct);

cartRepository.save(cart);

return product;

}

@GetMapping("/cart/{user\_id}")

public Cart getCartByUser(@PathVariable("user\_id")Long user\_id){

User user = userService.getUserById(user\_id);

Cart cart = cartRepository.findCartByUser(user);

return cart;

}

@DeleteMapping("/cart/{cart\_id}/{product\_id}")

public void deleteProduct(@PathVariable("cart\_id")Long cart\_id,@PathVariable("product\_id")Long product\_id){

cartService.deleteProductFromCart(cart\_id,product\_id);

}

@DeleteMapping("/cart/{cart\_id}")

public void deleteProducts(@PathVariable("cart\_id")Long cart\_id){

cartService.deleteProductsFromCart(cart\_id);

}

//Address Endpoints

@PostMapping("/address")

public Address addAddress(@RequestBody Address address){

return addressService.addAddress(address);

}

@PutMapping("/address")

public Address updateAddress(@RequestBody Address address){

System.out.println(address.getUser());

return addressService.updateAddress(address);

}

@GetMapping("/addresses/{user\_id}")

public Set<Address> getAddressesOfUser(@PathVariable("user\_id") Long user\_id){

User user = userService.getUserById(user\_id);

return addressService.getAddressesOfUser(user);

}

@GetMapping("/address/{address\_id}")

public Address getAddress(@PathVariable("address\_id") Long address\_id){

return addressService.getAddress(address\_id);

}

@DeleteMapping("/delete-address/{address\_id}")

public void deleteAddress(@PathVariable("address\_id") Long address\_id){

addressService.deleteAddress(address\_id);

}

@PostMapping("/order")

public UserOrders addOrder(@RequestBody UserOrders order){

return orderService.createOrder(order);

}

@GetMapping("/order/{user\_id}")

public Set<UserOrders> getOrders(@PathVariable("user\_id") Long user\_id){

User user = userService.getUserById(user\_id);

return orderService.getOrders(user);

}

@GetMapping("/order-product/{order\_id}")

public Set<Product> getProductsOfOrder(@PathVariable("order\_id") Long order\_id){

return orderService.getProducts(order\_id);

}

@GetMapping("/orders")

public Set<UserOrders> getAllfOrder(){

return orderService.getAllPendingOrders();

}

@GetMapping("/all-orders")

public Set<UserOrders> getAllOrder(){

return orderService.getAllOrders();

}

@GetMapping("/complete-order/{order\_id}")

public UserOrders completeOrder(@PathVariable("order\_id") Long order\_id){

return orderService.completeOrder(order\_id);

}

@GetMapping("/notification/{user\_id}")

public Set<Notification> getNotifications(@PathVariable("user\_id") Long user\_id){

User user = userService.getUserById(user\_id);

return new HashSet<>( notificationRepository.findByUser(user));

}

}

**Package:- com.medicare.config**

**File:- CustomAuthorityDeserializer.java**

package com.medicare.config;

import com.fasterxml.jackson.core.JsonParser;

import com.fasterxml.jackson.databind.DeserializationContext;

import com.fasterxml.jackson.databind.JsonDeserializer;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import java.io.IOException;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

public class CustomAuthorityDeserializer extends JsonDeserializer {

@Override

public Object deserialize(JsonParser jp, DeserializationContext ctxt) throws IOException {

ObjectMapper mapper = (ObjectMapper) jp.getCodec();

JsonNode jsonNode = mapper.readTree(jp);

List<GrantedAuthority> grantedAuthorities = new LinkedList<>();

Iterator<JsonNode> elements = jsonNode.elements();

while (elements.hasNext()) {

JsonNode next = elements.next();

JsonNode authority = next.get("authority");

grantedAuthorities.add(new SimpleGrantedAuthority(authority.asText()));

}

return grantedAuthorities;

}

}

**File:- JwtAuthenticationEntryPoint.java**

package com.medicare.config;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.web.AuthenticationEntryPoint;

import org.springframework.stereotype.Component;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

@Override

public void commence(HttpServletRequest request, HttpServletResponse response, AuthenticationException authException) throws IOException, ServletException {

response.sendError(HttpServletResponse.SC\_UNAUTHORIZED,"Unauthorized: Server");

}

}

**File:- JwtAuthenticationFilter.java**

package com.medicare.config;

import com.medicare.services.impl.UserDetailsServiceImpl;

import io.jsonwebtoken.ExpiredJwtException;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

@Autowired

private UserDetailsServiceImpl userDetailsService;

@Autowired

private JwtUtils jwtUtils;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

final String requestTokenHeader = request.getHeader("Authorization");

System.out.println(requestTokenHeader);

String username=null;

String jwtToken=null;

if(requestTokenHeader!=null && requestTokenHeader.startsWith("Bearer ")){

jwtToken= requestTokenHeader.substring(7);

try {

username = this.jwtUtils.extractUsername(jwtToken);

}catch (ExpiredJwtException expiredJwtException){

System.out.println(" Token Has Expired");

}catch (Exception e){

e.printStackTrace();

}

}else{

System.out.println("Invalidate Token");

}

if(username!=null && SecurityContextHolder.getContext().getAuthentication()==null){

final UserDetails userDetails=this.userDetailsService.loadUserByUsername(username);

if(this.jwtUtils.validateToken(jwtToken,userDetails)){

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken= new UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());

usernamePasswordAuthenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}

}else {

System.out.println("Token Is Not Valid");

}

filterChain.doFilter(request,response);

}

}

**File:- JwtUtils.java**

package com.medicare.config;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Service;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Service

public class JwtUtils {

private String SECRET\_KEY = "onlinetest";

public String extractUsername(String token) {

return extractClaim(token, Claims::getSubject);

}

public Date extractExpiration(String token) {

return extractClaim(token, Claims::getExpiration);

}

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

final Claims claims = extractAllClaims(token);

return claimsResolver.apply(claims);

}

private Claims extractAllClaims(String token) {

return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

}

private Boolean isTokenExpired(String token) {

return extractExpiration(token).before(new Date());

}

public String generateToken(UserDetails userDetails) {

Map<String, Object> claims = new HashMap<>();

return createToken(claims, userDetails.getUsername());

}

private String createToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY).compact();

}

public Boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

}

**File:- MySecurityConfig.java**

package com.medicare.config;

import com.medicare.services.impl.UserDetailsServiceImpl;

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

import org.springframework.cglib.proxy.NoOp;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class MySecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private UserDetailsServiceImpl userDetailsServiceImpl;

@Autowired

private JwtAuthenticationEntryPoint unauthorizedHandler;

@Autowired

private JwtAuthenticationFilter jwtAuthenticationFilter;

@Bean

public BCryptPasswordEncoder passwordEncoder(){

return new BCryptPasswordEncoder();

}

@Bean

protected AuthenticationManager authenticationManager() throws Exception {

return super.authenticationManager();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsServiceImpl).passwordEncoder(passwordEncoder());

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf()

.disable()

.cors()

.disable()

.authorizeRequests()

.antMatchers("/generate-token",

"/users/",

"/products/\*\*",

"/categories/\*\*")

.permitAll()

.antMatchers(HttpMethod.OPTIONS).permitAll()

.anyRequest().authenticated()

.and()

.exceptionHandling().authenticationEntryPoint(unauthorizedHandler)

.and()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

}

}

**Package:- com.medicare**

**File:- MedicareBackendApplication.java**

package com.medicare;

import com.medicare.models.Cart;

import com.medicare.models.Role;

import com.medicare.models.User;

import com.medicare.models.UserRole;

import com.medicare.repositories.CartRepository;

import com.medicare.repositories.UserRepository;

import com.medicare.services.CartService;

import com.medicare.services.UserService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import java.util.HashSet;

import java.util.Set;

@SpringBootApplication

public class MedicareBackendApplication implements CommandLineRunner {

@Autowired

private UserService userService;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

@Autowired

private UserRepository userRepository;

@Autowired

private CartRepository cartRepository;

@Autowired

private CartService cartService;

public static void main(String[] args) {

SpringApplication.run(MedicareBackendApplication.class, args);

}

@Override

public void run(String... args) {

User user = new User();

user.setUsername("admin");

user.setUser\_email("cmallick88@gmail.com");

user.setPassword(passwordEncoder.encode("admin123"));

user.setEnabled(true);

user.setFirst\_name("Chandan");

user.setLast\_name("Mallick");

user.setUser\_phone("7008045937");

Role adminRole = new Role();

adminRole.setRole\_name("ADMIN");

Set<UserRole> userRoles = new HashSet<>();

UserRole userRole = new UserRole();

userRole.setRole(adminRole);

userRole.setUser(user);

userRoles.add(userRole);

user.setUserRoles(userRoles);

User local = userRepository.findByUsername(user.getUsername());

if(local != null){

System.err.println("User Already Exists");

try {

throw new Exception("User Already Exist");

} catch (Exception e) {

System.err.println("User Already Exist");

}

}

else {

try {

User user1=userService.createUser(user, userRoles);

Cart cart = new Cart();

cart.setUser(user1);

cartService.createCart(cart);

} catch (Exception e) {

throw new RuntimeException(e);

}

}

}

}

**File:- application.properties**

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

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.jpa.show-sql=true

spring.jpa.format-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.servlet.multipart.max-file-size=-1

spring.servlet.multipart.max-request-size=-1

**File:- 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>2.7.5</version>

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

</parent>

<groupId>com.</groupId>

<artifactId>backend</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>medicare backend</name>

<description>Capstone rest api</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>

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security -->

<dependency>

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

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

</dependency>

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.49</version>

</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>

**Frontend (Angular)**

**File:- index.html**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

**<title>Frontend</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="preconnect" href="https://fonts.gstatic.com">**

**<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet">**

**<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">**

**<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"**

**integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">**

**</head>**

**<body class="mat-typography main-bg">**

**<app-root></app-root>**

**<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"**

**integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>**

**</body>**

**</html>**

**File:- Style.css**

**/\* You can add global styles to this file, and also import other style files \*/**

**@import "~bootstrap/dist/css/bootstrap.min.css";**

**html, body { height: 100%; }**

**body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif; }**

**.main-bg{**

**background:#757ce8;**

**}**

**mat-dialog-container {**

**border: 1px solid black;**

**}**

**File:- category.model.ts**

**export interface Category{**

**id:number;**

**category\_name:string;**

**category\_description:string**

**}**

**File:- file-handle.model.ts**

**import { SafeUrl } from "@angular/platform-browser";**

**export interface FileHandle{**

**file:File;**

**url:SafeUrl;**

**}**

**File:- product.model.ts**

import { Category } from "./category.model";

import { FileHandle } from "./file-handle.model";

export interface Product {

  id:0;

  product\_name: string;

  product\_description: string;

  product\_price: number;

  product\_discount: number;

  product\_stock: number;

  product\_selling\_price: number;

  productImages:FileHandle[];

  category:Category;

}

**File:- environment.prod.ts**

export const environment = {

  production: true

};

**File:- environments.ts**

export const environment = {

  production: false

};

**File:- app-component.html**

<ngx-ui-loader fgsColor="#757ce8" pbColor="#757ce8" text="Loading" textColor="#757ce8">

</ngx-ui-loader>

<app-navbar></app-navbar>

<router-outlet></router-outlet>

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

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

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

import { AddCategoryComponent } from './pages/admin/add-category/add-category.component';

import { AddProductComponent } from './pages/admin/add-product/add-product.component';

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

import { AdminHomeComponent } from './pages/admin/admin-home/admin-home.component';

import { CategoryListComponent } from './pages/admin/category-list/category-list.component';

import { PendingOrdersComponent } from './pages/admin/pending-orders/pending-orders.component';

import { UpdateCategoryComponent } from './pages/admin/update-category/update-category.component';

import { UpdateProductComponent } from './pages/admin/update-product/update-product.component';

import { ViewProductImagesComponent } from './pages/admin/view-product-images/view-product-images.component';

import { ViewProductComponent } from './pages/admin/view-product/view-product.component';

import { ViewUsersComponent } from './pages/admin/view-users/view-users.component';

import { ChangePasswordComponent } from './pages/change-password/change-password.component';

import { HomeComponent } from './pages/home/home.component';

import { LoginComponent } from './pages/login/login.component';

import { ProfileComponent } from './pages/profile/profile.component';

import { RegisterComponent } from './pages/register/register.component';

import { SearchProductComponent } from './pages/search-product/search-product.component';

import { UpdateUserComponent } from './pages/update-user/update-user.component';

import { AddAddressComponent } from './pages/user/add-address/add-address.component';

import { UpdateAddressComponent } from './pages/user/update-address/update-address.component';

import { UserAddressComponent } from './pages/user/user-address/user-address.component';

import { UserDashboardComponent } from './pages/user/user-dashboard/user-dashboard.component';

import { UserOrdersComponent } from './pages/user/user-orders/user-orders.component';

import { AdminGuard } from './services/admin.guard';

import { NormalGuard } from './services/normal.guard';

const routes: Routes = [

  {

    path: 'register',

    component: RegisterComponent,

    pathMatch: 'full',

  },

  {

    path: 'login',

    component: LoginComponent,

    pathMatch: 'full',

  },

  {

    path: '',

    redirectTo: 'home/0',

    pathMatch: 'full',

  },

  {

    path: 'home/:cat\_id',

    component: HomeComponent,

  },

  {

    path:'search/:model',

    component:SearchProductComponent

  },

  {

    path: 'update-user/:user\_id',

    component: UpdateUserComponent,

    pathMatch: 'full',

  },

  {

    path: 'change-password',

    component: ChangePasswordComponent,

    pathMatch: 'full',

  },

  {

    path: 'admin',

    component: AdminDashboardComponent,

    canActivate: [AdminGuard],

    children: [

      {

        path: 'profile',

        component: ProfileComponent,

      },

      {

        path: '',

        component: AdminHomeComponent,

      },

      {

        path: 'view-categories',

        component: CategoryListComponent,

      },

      {

        path: 'add-category',

        component: AddCategoryComponent,

      },

      {

        path: 'update-category/:cat\_id',

        component: UpdateCategoryComponent,

      },

      {

        path: 'view-products',

        component: ViewProductComponent,

      },

      {

        path: 'add-product',

        component: AddProductComponent,

      },

      {

        path: 'update-product/:product\_id',

        component: UpdateProductComponent,

      },

      {

        path: 'view-images/:product\_id',

        component: ViewProductImagesComponent,

      },

      {

        path: 'view-users',

        component: ViewUsersComponent,

      },

      {

        path: 'view-pending-orders',

        component: PendingOrdersComponent,

      },

    ],

  },

  {

    path: 'user-dashboard',

    component: UserDashboardComponent,

    canActivate: [NormalGuard],

    children: [

      {

        path: '',

        component: ProfileComponent,

      },

      {

        path: 'address',

        component: UserAddressComponent,

      },

      {

        path: 'orders',

        component: UserOrdersComponent,

      },

      {

        path: 'add-address',

        component: AddAddressComponent,

      },

      {

        path: 'update-address/:address\_id',

        component: UpdateAddressComponent,

      },

      {

        path: 'profile',

        component: ProfileComponent,

      },

    ],

  },

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**Package:- services**

**File:- admin.guard.ts**

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

import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';

**import { Observable } from 'rxjs';**

**import { LoginService } from './login.service';**

**@Injectable({**

**providedIn: 'root'**

**})**

**export class AdminGuard implements CanActivate {**

**constructor(private loginServices:LoginService,**

**private route:Router){}**

**canActivate(**

**route: ActivatedRouteSnapshot,**

**state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {**

**if(this.loginServices.isLoggedIn() && this.loginServices.getUserRole()=='ADMIN'){**

**return true;**

**}**

**this.route.navigate(['login']);**

**return false;**

**}**

**}**

**File:- auth.interceptor.ts**

**import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from "@angular/common/http";**

**import { Injectable } from "@angular/core";**

**import { Observable } from "rxjs";**

**import { LoginService } from "./login.service";**

**@Injectable()**

**export class AuthInterceptor implements HttpInterceptor{**

**constructor(private loginService:LoginService){}**

**intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {**

**let authReq=req;**

**const token = this.loginService.getToken();**

**if(token!=null){**

**authReq = authReq.clone({**

**setHeaders: { Authorization: `Bearer ${token}` },**

**});**

**}**

**return next.handle(authReq);**

**}**

**}**

**File:- helper.ts**

**let baseUrl="http://localhost:8080";**

**export default baseUrl;**

**File:- image-processing.service.ts**

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

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

import { FileHandle } from 'src/model/file-handle.model';

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

@Injectable({

  providedIn: 'root'

})

export class ImageProcessingService {

  constructor(private sanitizer:DomSanitizer) { }

public createProductImages(product:any){

    const productImages:any[] = product.product\_images;

    const productImagesToFileHandle:FileHandle[]= [];

    for(let i=0;i<productImages.length;i++){

      const imageData = productImages[i];

      const imageBlob =  this.dataURItoBLOB(imageData.imageByte,imageData.type);

      const file = new File([imageBlob],imageData.name,{type:imageData.type});

      const fileHandle : FileHandle={

        file:file,

        url:this.sanitizer.bypassSecurityTrustUrl(window.URL.createObjectURL(file))

      };

      productImagesToFileHandle.push(fileHandle);

    }

    product.productImages=productImagesToFileHandle;

    return product;

  }

  public dataURItoBLOB(picBytes:any,imageType:any){

    const byteString = window.atob(picBytes);

    const arrayBuffer = new ArrayBuffer(byteString.length);

    const int8array = new Int8Array(arrayBuffer);

    for(let i=0;i<byteString.length;i++){

      int8array[i]=byteString.charCodeAt(i);

    }

    const blob = new Blob([int8array],{type:imageType});

    return blob;

  }

}

**File:- login.service.ts**

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

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

import { Subject } from 'rxjs/internal/Subject';

import baseUrl from './helper';

@Injectable({

  providedIn: 'root'

})

export class LoginService {

  public loginStatusSubject = new Subject<boolean>()

  constructor(private http:HttpClient) { }

  public getCurrentUser(){

    return this.http.get(`${baseUrl}/current-user`)

  }

  public generateToken(loginData:any){

    return this.http.post(`${baseUrl}/generate-token`,loginData);

  }

  public loginUser(token:any){

    localStorage.setItem('token',token);

    return true;

  }

  public isLoggedIn(){

    let tokenstr= localStorage.getItem('token');

    if(tokenstr==undefined || tokenstr==''||tokenstr==null){

      return false;

    }else{

      return true;

    }

  }

  public logout(){

    localStorage.removeItem('token');

    localStorage.removeItem('user');

    return true;

  }

  public getToken(){

    return localStorage.getItem('token');

  }

  public setUser(user:any){

    localStorage.setItem('user',JSON.stringify(user));

  }

  public getUser(){

    let user = localStorage.getItem('user');

    if(user!=null){

      return JSON.parse(user)

    }else{

      this.logout();

      return null;

    }

  }

  public getUserRole(){

    let user = this.getUser();

    if(user!=null){

    return user.authorities[0].authority;

    }

  }

  public isAdmin(){

    if(this.getUserRole() == 'ADMIN'){

      return true;

    }else{

      return false;

    }

  }

  public isNormal(){

    if (this.getUserRole() == 'NORMAL') {

      return true;

    } else {

      return false;

    }

  }

}

F**ile:- normal.guard.ts**

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

import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';

import { Observable } from 'rxjs';

import { LoginService } from './login.service';

@Injectable({

  providedIn: 'root'

})

export class NormalGuard implements CanActivate {

  constructor(private loginService:LoginService,private route:Router){}

  canActivate(

    route: ActivatedRouteSnapshot,

    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {

      if(this.loginService.isLoggedIn() && this.loginService.getUserRole()=='NORMAL'){

        return true;

      }

       this.route.navigate(['login']);

    return false;

  }

}

**File:- product.service.ts**

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

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

import { Subject } from 'rxjs';

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

import baseUrl from './helper';

@Injectable({

  providedIn: 'root',

})

export class ProductService {

  public searchStatus = new Subject<boolean>();

  constructor(private http: HttpClient) {}

  public addProduct(product: FormData) {

    return this.http.post(`${baseUrl}/products/`, product);

  }

  public getProducts() {

    return this.http.get(`${baseUrl}/products/`);

  }

  public searchProducts(model: any) {

    return this.http.get(`${baseUrl}/products/search/${model}`);

  }

  public searchProductsStartWith(model: any) {

    return this.http.get(`${baseUrl}/products/search-startwith/${model}`);

  }

  public getProductsByCategory(cat\_id: any) {

    console.log(cat\_id);

    return this.http.get(`${baseUrl}/categories/product-list/${cat\_id}`);

  }

  public deleteProduct(id: any) {

    return this.http.delete(`${baseUrl}/products/${id}`);

  }

  public getProduct(id: any) {

    return this.http.get(`${baseUrl}/products/${id}`);

  }

  public updateProduct(product: FormData) {

    return this.http.put(`${baseUrl}/products/`, product);

  }

  public addCategory(category: any) {

    return this.http.post(`${baseUrl}/categories`, category);

  }

  public getCategories() {

    return this.http.get(`${baseUrl}/categories`);

  }

  public getPopularCategories() {

    return this.http.get(`${baseUrl}/categories/popular`);

  }

  public getCategory(id: any) {

    return this.http.get(`${baseUrl}/categories/${id}`);

  }

  public deleteCategory(id: any) {

    return this.http.delete(`${baseUrl}/categories/${id}`);

  }

  public updateCategory(category: any) {

    return this.http.put(`${baseUrl}/categories`, category);

  }

}

**File:- user.service.ts**

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

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

import { Subject } from 'rxjs';

import baseUrl from './helper';

@Injectable({

  providedIn: 'root',

})

export class UserService {

  public cartStatus = new Subject<boolean>();

  public orderStatus = false;

  public userStatus = new Subject<boolean>();

  public cartOpen = false;

  constructor(private http: HttpClient) {}

  public addUser(user: any) {

    return this.http.post(`${baseUrl}/users/`, user);

  }

  public updateUser(user: any) {

    return this.http.put(`${baseUrl}/users/`, user);

  }

  public getUsers() {

    return this.http.get(`${baseUrl}/users/`);

  }

  public getUser(user\_id: any) {

    return this.http.get(`${baseUrl}/users/user/${user\_id}`);

  }

  public confirmPassword(user: any) {

    return this.http.post(`${baseUrl}/users/confirm-password`, user);

  }

  public changePassword(user: any) {

    return this.http.post(`${baseUrl}/users/change-password`, user);

  }

  public addProductToCart(cart\_id: any, product: any) {

    return this.http.post(`${baseUrl}/users/cart/${cart\_id}`, product);

  }

  public getCart(user\_id: any) {

    return this.http.get(`${baseUrl}/users/cart/${user\_id}`);

  }

  public deleteProductFromCart(cart\_id: any, product\_id: any) {

    return this.http.delete(`${baseUrl}/users/cart/${cart\_id}/${product\_id}`);

  }

  public deleteProductsFromCart(cart\_id: any) {

    return this.http.delete(`${baseUrl}/users/cart/${cart\_id}`);

  }

  public addAddress(address: any) {

    return this.http.post(`${baseUrl}/users/address`, address);

  }

  public updateAddress(address: any) {

    return this.http.put(`${baseUrl}/users/address`, address);

  }

  public getAddressesOfUser(user\_id: any) {

    return this.http.get(`${baseUrl}/users/addresses/${user\_id}`);

  }

  public getAddress(address\_id: any) {

    return this.http.get(`${baseUrl}/users/address/${address\_id}`);

  }

  public deleteAddress(address\_id: any) {

    return this.http.delete(`${baseUrl}/users/delete-address/${address\_id}`);

  }

  public createOrder(order: any) {

    return this.http.post(`${baseUrl}/users/order`, order);

  }

  public getOrders(user\_id: any) {

    return this.http.get(`${baseUrl}/users/order/${user\_id}`);

  }

  public getOrderProducts(order\_id: any) {

    return this.http.get(`${baseUrl}/users/order-product/${order\_id}`);

  }

  public getPendingOrder() {

    return this.http.get(`${baseUrl}/users/orders`);

  }

  public getAllOrder() {

    return this.http.get(`${baseUrl}/users/all-orders`);

  }

  public completeOrder(order\_id: any) {

    return this.http.get(`${baseUrl}/users/complete-order/${order\_id}`);

  }

  public getNotifications(user\_id: any) {

    return this.http.get(`${baseUrl}/users/notification/${user\_id}`);

  }

}

**Package:- Pipe**

**File:- product-title.pipe.ts**

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'productTitle'

})

export class ProductTitlePipe implements PipeTransform {

  transform(value: string): string {

    if(value.length>=40){

     return value.substring(0, 35);

    }else {

      return value;

    }

  }

}

**Package:- pages.admin.add.** **add-category**

**File:-** **add-category.component.html**

<mat-card class="mt-2">

    <h1 matSubheader>Add New Category</h1>

    <mat-card-content>

        <div class="container mt-3">

  <div class="row">

    <div class="col-md-6 offset-md-3 text-center">

     <mat-card-content>

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

        <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Category Name</mat-label>

          <input matInput placeholder="Category Name "

          [(ngModel)]="category.category\_name"

          name="category\_name" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Category Description</mat-label>

          <textarea matInput placeholder="Category Description"

          [(ngModel)]="category.category\_description"

          name="category\_desc" required cols="25" maxlength="100">

          </textarea>

        </mat-form-field>

          <mat-slide-toggle name="popular" class="example-margin"  [(ngModel)]="category.popular" >

            Popular Category

          </mat-slide-toggle>

        <div class="container mt-5">

            <button type="submit" mat-raised-button color="primary">Add Category</button>

        </div>

      </form>

     </mat-card-content>

    </div>

  </div>

</div>

    </mat-card-content>

</mat-card>

**File:- add-category.component.ts**

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

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

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

import Swal from 'sweetalert2';

@Component({

  selector: 'app-add-category',

  templateUrl: './add-category.component.html',

  styleUrls: ['./add-category.component.css'],

})

export class AddCategoryComponent implements OnInit {

  category = {

    category\_name: '',

    category\_description: '',

    popular:false,

  };

  constructor(private productService: ProductService,

              private router:Router) {}

  ngOnInit(): void {}

  addCategory() {

    console.log(this.category)

    this.productService.addCategory(this.category).subscribe(

      (data) => {

      Swal.fire({

        position: 'top-end',

        icon: 'success',

        title: 'Category added successfully!',

        showConfirmButton: true,

      });

      this.router.navigate(['/admin/view-categories'])

    },

    (error)=>{

        Swal.fire({

          position: 'top-end',

          icon: 'error',

          title: 'Error !',

        });

    }

    );

  }

}

**File:- add-category.component.css**

.example-margin {

    margin: 10px;

}

**Package:- pages.admin.** **add-product**

**File:-** **add-product.component.html**

<mat-card class="mt-2">

  <h1 matSubheader>Add New Product</h1>

  <mat-card-content>

  <form (ngSubmit)="addProduct()">

    <div class="row">

        <div class="col-md-6 text-center">

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Name</mat-label>

            <input matInput placeholder="Product Name " [(ngModel)]="product.product\_name" name="product\_name"

              required />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Description</mat-label>

            <textarea matInput placeholder="Product Description" [(ngModel)]="product.product\_description"

              name="product\_desc" required cols="25" maxlength="100">

                      </textarea>

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Price</mat-label>

            <input matInput placeholder="Product Price " [(ngModel)]="product.product\_price" name="product\_price"

              required type="number" />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Discount</mat-label>

            <input matInput placeholder="Product Discount " [(ngModel)]="product.product\_discount"

              name="product\_discount" required type="number" />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Stock</mat-label>

            <input matInput placeholder="Product Stock " [(ngModel)]="product.product\_stock" name="product\_stock"

              required type="number" />

          </mat-form-field>

          <mat-form-field appearance="fill" class="w-100">

            <mat-label> Category </mat-label>

            <mat-select name="category" [(ngModel)]="product.category.id">

              <mat-option \*ngFor="let c of categories" [value]="c.id">

                {{c.category\_name}}

              </mat-option>

            </mat-select>

          </mat-form-field>

        </div>

        <div class="col-md-6">

          <input type="file" matTooltip="Don't select same image" class="" value="select" #selectFile multiple

            (change)="onFileSelected($event)">

            <mat-grid-list cols="4" rowHeight="2:1" class="mt-2">

              <mat-grid-tile \*ngFor="let productImage of product.productImages ; let i=index">

                <div style="position:relative">

                  <mat-icon class="btn-remove-image" color="warn" (click)="removeImage(i)" matTooltip="Remove Image">

                    remove\_circle</mat-icon>

                  <img [src]="productImage.url" alt="" width="55px" height="55px">

                </div>

              </mat-grid-tile>

            </mat-grid-list>

        </div>

        <div class="container text-center mt-2">

          <button type="submit" mat-raised-button color="primary">

            Add Product

          </button>

        </div>

    </div>

   </form>

  </mat-card-content>

</mat-card>

**File:- add-product.component.ts**

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

import { MatSelectionListChange } from '@angular/material/list';

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

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

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

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

import { FileHandle } from 'src/model/file-handle.model';

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

import Swal from 'sweetalert2';

@Component({

  selector: 'app-add-product',

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

  styleUrls: ['./add-product.component.css'],

})

export class AddProductComponent implements OnInit {

  categories: any = [];

  numberOfImages=0;

  product:Product = {

    id:0,

    product\_name: '',

    product\_description: '',

    product\_price: 0,

    product\_discount: 0,

    product\_stock: 0,

    product\_selling\_price: 0,

    productImages:[],

    category: {

      id: 0,

      category\_name: '',

      category\_description: '',

    },

  };

  constructor(

    private productService: ProductService,

    private snack: MatSnackBar,

    private router:Router,

    private sanitizer:DomSanitizer

  ) {}

  ngOnInit(): void {

    this.numberOfImages=0;

    this.productService.getCategories().subscribe(

      (data: any) => {

        this.categories = data;

      },

      (error) => {

        alert('unable to laod categories data');

        console.log(error);

      }

    );

  }

  addProduct() {

 this.product.product\_selling\_price =

   this.product.product\_price -

   (this.product.product\_price / 100) \* this.product.product\_discount;

    const productFormData= this.prepareFormData(this.product);

    this.productService.addProduct(productFormData).subscribe(

      (data) => {

        Swal.fire({

          position: 'top-end',

          icon: 'success',

          title: 'Product added successfully!',

          showConfirmButton: true,

        });

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

      },

      (error) => {

        this.snack.open('Unable to add product', 'ok');

      }

    );

  }

  prepareFormData(product:Product):FormData{

    const formData= new FormData();

    formData.append(

      'product',

       new Blob([JSON.stringify(product)],{type:'application/json'})

    );

    for(let i=0;i<this.product.productImages.length;i++){

      formData.append(

        'imageFile',

        this.product.productImages[i].file,

        this.product.productImages[i].file.name

      )

    }

    return formData;

  }

 onFileSelected(event:any){

    this.numberOfImages++;

    console.log(event);

    if(event.target.files){

      const file = event.target.files[0];

      const fileHandle:FileHandle={

        file:file,

        url: this.sanitizer.bypassSecurityTrustUrl(

          window.URL.createObjectURL(file)

        )

      }

      if(this.numberOfImages<=4){

        this.product.productImages.push(fileHandle);

      }else{

        this.snack.open("You can add 4 images only.","OK",{

          horizontalPosition:'right',

          verticalPosition:'top'

        })

      }

    }

 }

 removeImage(i:any){

    this.product.productImages.splice(i,1)

    this.numberOfImages--;

 }

}

**File:- add-product.component.css**

mat-grid-tile {

    border: 1px solid lightgrey;

    margin-bottom: 10px;

}

.btn-remove-image{

    position: absolute;

    cursor: pointer;

    top: -2px;

    right: -15px;

}

**Package:- pages.admin.admin-dashboard**

**File:-** **add-dashboard.component.html**

<div class="container-fluid">

  <div class="row">

    <div class="col-md-3">

      <app-admin-sidebar></app-admin-sidebar>

    </div>

    <div class="col-md-9">

      <router-outlet></router-outlet>

    </div>

  </div>

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

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

@Component({

  selector: 'app-admin-dashboard',

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

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

})

export class AdminDashboardComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

**Package:- pages.admin.** **admin-home**

**File:- admin-home.component.html**

<mat-card class="mt-2">

   <mat-card-title>Admin Home</mat-card-title>

</mat-card>

<div class="row row-cols-1 row-cols-md-5 mt-3 mb-5 gap-5">

    <mat-card class="ms-2">

    <div class="text-center">

        <img src="../../../../assets/users\_icon.png" height="100px" width="100px" alt="">

    </div>

     <mat-card-subtitle class="text-center mt-1">Users</mat-card-subtitle>

        <mat-chip-list>

        <span class="empty-space">..................</span><mat-chip >{{users.length}}</mat-chip>

        </mat-chip-list>

    </mat-card>

    <mat-card>

        <div class="text-center">

            <img src="../../../../assets/orders\_icon.png" height="100px" width="100px" alt="">

        </div>

        <mat-card-subtitle class="text-center mt-1">Orders</mat-card-subtitle>

        <mat-chip-list>

        <span class="empty-space">..................</span><mat-chip>{{orders.length}} </mat-chip>

        </mat-chip-list>

    </mat-card>

    <mat-card>

        <div class="text-center">

            <img src="../../../../assets/payments.png" height="100px" width="100px" alt="">

        </div>

        <mat-card-subtitle class="text-center mt-1">Payment Received</mat-card-subtitle>

        <mat-chip-list>

            <span class="empty-space"></span> <img src="../../../../assets/cash.gif" width="30px" alt=""><mat-chip>{{paymentRecieved| currency:'INR'}} </mat-chip>

        </mat-chip-list>

    </mat-card>

    <mat-card>

        <div class="text-center">

         <img src="../../../../assets/pending\_payments.png" height="100px" width="100px" alt="">

        </div>

        <mat-card-subtitle class="text-center mt-1">Pending Payment</mat-card-subtitle>

        <mat-chip-list>

        <span class="empty-space">.......</span><img src="../../../../assets/cash.gif" width="30px" alt=""><mat-chip>{{pendingPayment| currency:'INR'}}</mat-chip>

        </mat-chip-list>

    </mat-card>

</div>

<mat-accordion class="example-headers-align">

    <mat-expansion-panel>

        <mat-expansion-panel-header>

            <mat-panel-title>

                All Orders

            </mat-panel-title>

        </mat-expansion-panel-header>

<div class="mt-1">

    <mat-card class="mt-1" \*ngFor="let o of orders">

        <button mat-button color="warn" \*ngIf="o.order\_status != 'COMPLETED'">

            <mat-icon>send</mat-icon> Pending

        </button>

        <button mat-button color="basic" \*ngIf="o.order\_status == 'COMPLETED'">

            <mat-icon>download\_done</mat-icon> Deliverd

        </button>

        <mat-card-header class="mt-1">

            <mat-card-title> Order Amount : <img src="../../../../assets/cash.gif" width="25px" alt=""> {{o.total |

                currency:'INR'}}</mat-card-title>

        </mat-card-header>

        <mat-accordion class="example-headers-align">

            <mat-expansion-panel>

                <mat-expansion-panel-header>

                    <mat-panel-title>

                        Order Details

                    </mat-panel-title>

                </mat-expansion-panel-header>

                <mat-card-subtitle>

                    User : <span class="user-deatils">{{o.user.first\_name}} {{o.user.last\_name}} </span>

                    Phone: <span class="user-deatils">{{o.user.user\_phone}} </span>

                    Email: <span class="user-deatils">{{o.user.user\_email}} </span>

                </mat-card-subtitle>

                <mat-card-subtitle>Address : Om Hostel Vadgoan Road, Near Modern Fitness Center Pune, 411041

                </mat-card-subtitle>

                <mat-card-subtitle>Order Placed : {{o.order\_date}}</mat-card-subtitle>

                <mat-card-subtitle>{{o.order\_status}}</mat-card-subtitle>

                <mat-card-subtitle \*ngIf="o.payment\_status == 'Cash on delivery'">Payment : <img

                        src="../../../../assets/cash.gif" width="25px" alt=""> {{o.payment\_status}}

                </mat-card-subtitle>

                <mat-card-subtitle \*ngIf="o.payment\_status != 'Cash on delivery'">

                    Payment : <img src="../../../../assets/checkicon.gif" width="25px" alt=""> Paid

                </mat-card-subtitle>

                <mat-accordion class="example-headers-align">

                    <mat-expansion-panel>

                        <mat-expansion-panel-header>

                            <mat-panel-title>

                                Purchased Products

                            </mat-panel-title>

                        </mat-expansion-panel-header>

                        <mat-card-content>

                            <mat-list>

                                <mat-list-item \*ngFor="let p of o.product\_list">

                                    <img mat-list-avatar [src]="p.productImages[0].url" alt="">

                                    <div matLine>{{p.product\_name}}</div>

                                    <div matLine>Payable Amount : {{p.product\_selling\_price | currency:'INR' }}</div>

                                </mat-list-item>

                            </mat-list>

                        </mat-card-content>

                    </mat-expansion-panel>

                </mat-accordion>

            </mat-expansion-panel>

        </mat-accordion>

    </mat-card>

</div>

</mat-expansion-panel>

</mat-accordion>

**File:- admin-home.component.ts**

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-admin-home',

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

  styleUrls: ['./admin-home.component.css'],

})

export class AdminHomeComponent implements OnInit {

  orders: any = [];

  products: any = [];

  public users: any = [];

  user: any = {};

  numberOfUser = 0;

  paymentRecieved = 0;

  pendingPayment = 0;

  constructor(

    private userService: UserService,

    private loginService: LoginService,

    private imageService: ImageProcessingService

  ) {}

  ngOnInit(): void {

    this.userService.getAllOrder().subscribe((data: any) => {

      this.orders = data;

      for (let o of this.orders) {

        this.userService.getOrderProducts(o.id).subscribe((data) => {

          this.products = data;

          if (o.payment\_status != 'Cash on delivery') {

            this.paymentRecieved += o.total;

          }

          if (o.payment\_status == 'Cash on delivery') {

            this.pendingPayment += o.total;

          }

          for (let p of this.products) {

            p = this.imageService.createProductImages(p);

          }

          o.product\_list = this.products;

        });

      }

      this.userService.getUsers().subscribe(

        (data) => {

          this.users = data;

        },

        (error) => {

        }

      );

      console.log(this.orders);

    });

  }

}

**File:- admin-home.component.css**

.empty-space {

    color: white;

}

**Package:- pages.admin.** **admin-sidebar**

**File:- admin-sidebar.component.html**

<mat-card class="mt-2 ms-2 ">

    <mat-action-list>

        <div matSubheader>MENU</div>

        <button mat-list-item routerLink="/admin">

            <mat-icon mat-list-icon>home\_outline</mat-icon>Home

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item routerLink="/admin/profile">

            <mat-icon mat-list-icon>account\_box</mat-icon>Profile

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item routerLink="/admin/view-categories">

            <mat-icon mat-list-icon>category</mat-icon> Categories

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item routerLink="/admin/view-products">

            <mat-icon mat-list-icon>inventory</mat-icon> Products

        </button>

         <mat-divider></mat-divider>

        <button mat-list-item routerLink="/admin/view-users">

            <mat-icon mat-list-icon>group</mat-icon> Users

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item

                routerLink="/admin/view-pending-orders">

            <mat-icon mat-list-icon>rule\_sharp</mat-icon>

            Pending Orders

            <span class="ms-5"

                  [matBadge]="orders.length"

                  matBadgeColor="warn"

                  matBadgePosition="above after"

                  \*ngIf="orders.length"></span>

        </button>

         <mat-divider></mat-divider>

        <button mat-list-item (click)="logout()" style="color: #f44337;">

            <mat-icon mat-list-icon color="warn">logout</mat-icon> Logout

        </button>

    </mat-action-list>

</mat-card>

**File:- admin-sidebar.component.ts**

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-admin-sidebar',

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

  styleUrls: ['./admin-sidebar.component.css'],

})

export class AdminSidebarComponent implements OnInit {

  orders:any=[]

  constructor(private loginService:LoginService,

              private userService:UserService) {}

  ngOnInit(): void {

     this.userService.getPendingOrder().subscribe(

      (data:any)=>{ this.orders = data; })

      this.userService.cartStatus.asObservable().subscribe(

        (data)=>

        {

           this.userService.getPendingOrder().subscribe((data: any) => {

             this.orders = data;

           });

        }

      )

  }

  logout() {

    this.loginService.logout();

    window.location.reload();

  }

}

**File:- admin-sidebar.component.css**

.menu{

    color: #3f51b5;

}

**Package:- pages.admin.** **category-list**

**File:- category-listcomponent.html**

<mat-card class="mt-2">

    <h1 matSubheader>Categories</h1>

    <div class="button-container">

        <button mat-raised-button color="primary" routerLink="/admin/add-category" matRipple>

            <mat-icon color="accent">add\_circle</mat-icon>

            New Category

        </button>

    </div>

    <table mat-table [dataSource]="categories" class="w-100 mat-elevation-z8">

        <ng-container matColumnDef="position">

            <th mat-header-cell \*matHeaderCellDef> No. </th>

            <td mat-cell \*matCellDef="let element"> {{element.id}} </td>

        </ng-container>

        <ng-container matColumnDef="category\_name">

            <th mat-header-cell \*matHeaderCellDef>Name</th>

            <td mat-cell \*matCellDef="let element"> {{element.category\_name}} </td>

        </ng-container>

        <ng-container matColumnDef="category\_description">

            <th mat-header-cell \*matHeaderCellDef>Description</th>

            <td mat-cell \*matCellDef="let element"> {{element.category\_description}} </td>

        </ng-container>

        <ng-container matColumnDef="actions">

            <th mat-header-cell \*matHeaderCellDef>  </th>

            <td mat-cell \*matCellDef="let element">

                <button mat-icon-button>

                    <mat-icon color="warn" matTooltip="delete category" (click)="deleteCategory(element.id)">

                        delete

                    </mat-icon>

                </button>

                <button mat-icon-button>

                    <mat-icon color="primary" matTooltip="update category" [routerLink]="'/admin/update-category/'+element.id">

                        edit

                    </mat-icon>

                </button>

            </td>

        </ng-container>

        <tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

        <tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

        <tr class="mat-row" \*matNoDataRow>

            <td class="mat-cell" colspan="4">No data matching the filter</td>

        </tr>

    </table>

</mat-card>

**File:- category-list.component.ts**

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

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

import Swal from 'sweetalert2';

@Component({

  selector: 'app-category-list',

  templateUrl: './category-list.component.html',

  styleUrls: ['./category-list.component.css'],

})

export class CategoryListComponent implements OnInit {

  displayedColumns: string[] = [

    'position',

    'category\_name',

    'category\_description',

    'actions',

  ];

  categories: any[] = [];

  constructor(private productService: ProductService) {}

  ngOnInit(): void {

    this.productService.getCategories().subscribe((data: any) => {

      this.categories = data;

    });

  }

  public deleteCategory(id: any) {

    Swal.fire({

      title: 'Are you sure?',

      text: "You won't be able to revert this!",

      icon: 'warning',

      showCancelButton: true,

      confirmButtonColor: '#3085d6',

      cancelButtonColor: '#d33',

      confirmButtonText: 'Yes, delete it!',

    }).then((result) => {

      if (result.isConfirmed) {

        this.productService.deleteCategory(id).subscribe(

          (data) => {

            Swal.fire({

              position: 'top-end',

              icon: 'success',

              title: 'Category deleted successfully!',

              showConfirmButton: true,

            });

            location.reload();

          },

          (error) => {

            Swal.fire({

              position: 'top-end',

              icon: 'error',

              title: 'Unable to delete category!',

              showConfirmButton: true,

            });

          }

        );

      }

    });

  }

}

**File:- File:- category-list.component.css**

.button-container {

    padding-bottom: 16px;

}

**Package:- pages.admin.** **pending-orders**

**File:- pending-orders.component.html**

<div class="mt-2">

    <mat-card class="mt-1" \*ngFor="let o of orders">

        <button mat-button color="warn" \*ngIf="o.order\_status != 'COMPLETED'">

            <mat-icon>send</mat-icon> Pending

        </button>

        <button mat-button color="basic" \*ngIf="o.order\_status == 'COMPLETED'">

            <mat-icon>download\_done</mat-icon> Deliverd

        </button>

        <mat-card-header class="mt-1">

            <mat-card-title> Order Amount : <img src="../../../../assets/cash.gif" width="25px" alt=""> {{o.total |

                currency:'INR'}}</mat-card-title>

        </mat-card-header>

        <mat-accordion class="example-headers-align">

            <mat-expansion-panel>

                <mat-expansion-panel-header>

                    <mat-panel-title>

                        Order Details

                    </mat-panel-title>

                </mat-expansion-panel-header>

                <mat-card-subtitle>

                    User : <span class="user-deatils">{{o.user.first\_name}} {{o.user.last\_name}} </span>

                    Phone: <span class="user-deatils">{{o.user.user\_phone}} </span>

                    Email: <span class="user-deatils">{{o.user.user\_email}} </span>

                </mat-card-subtitle>

                <mat-card-subtitle>Address : Om Hostel Vadgoan Road, Near Modern Fitness Center Pune, 411041

                </mat-card-subtitle>

                <mat-card-subtitle>Order Placed : {{o.order\_date}}</mat-card-subtitle>

                <mat-card-subtitle>{{o.order\_status}}</mat-card-subtitle>

                <mat-card-subtitle \*ngIf="o.payment\_status == 'Cash on delivery'">Payment : <img

                        src="../../../../assets/cash.gif" width="25px" alt=""> {{o.payment\_status}}

                </mat-card-subtitle>

                <mat-card-subtitle \*ngIf="o.payment\_status != 'Cash on delivery'">

                    Payment : <img src="../../../../assets/checkicon.gif" width="25px" alt=""> Paid

                </mat-card-subtitle>

                <mat-accordion class="example-headers-align">

                    <mat-expansion-panel>

                        <mat-expansion-panel-header>

                            <mat-panel-title>

                                Purchased Products

                            </mat-panel-title>

                        </mat-expansion-panel-header>

                        <mat-card-content>

                            <mat-list>

                                <mat-list-item \*ngFor="let p of o.product\_list">

                                    <img mat-list-avatar [src]="p.productImages[0].url" alt="">

                                    <div matLine>{{p.product\_name}}</div>

                                    <div matLine>Payable Amount : {{p.product\_selling\_price | currency:'INR' }}</div>

                                </mat-list-item>

                            </mat-list>

                        </mat-card-content>

                    </mat-expansion-panel>

                </mat-accordion>

            </mat-expansion-panel>

        </mat-accordion>

        <mat-card-actions class="text-center">

            <button color="primary" (click)="completeOrder(o.id)" mat-raised-button>Complete Order</button>

        </mat-card-actions>

    </mat-card>

</div>

**File:- pending-orders.component.ts**

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-pending-orders',

  templateUrl: './pending-orders.component.html',

  styleUrls: ['./pending-orders.component.css'],

})

export class PendingOrdersComponent implements OnInit {

  orders: any = [];

  products: any = [];

  user: any = {};

  constructor(

    private userService: UserService,

    private loginService: LoginService,

    private imageService: ImageProcessingService

  ) {}

  ngOnInit(): void {

    this.userService.getPendingOrder().subscribe((data: any) => {

      this.orders = data;

      for (let o of this.orders) {

        this.userService.getOrderProducts(o.id).subscribe((data) => {

          this.products = data;

          for (let p of this.products) {

            p = this.imageService.createProductImages(p);

          }

          o.product\_list = this.products;

        });

      }

      console.log(this.orders);

    });

    this.userService.cartStatus.asObservable().subscribe(

      (data)=>{

             this.userService.getPendingOrder().subscribe((data: any) => {

               this.orders = data;

               for (let o of this.orders) {

                 this.userService.getOrderProducts(o.id).subscribe((data) => {

                   this.products = data;

                   for (let p of this.products) {

                     p = this.imageService.createProductImages(p);

                   }

                   o.product\_list = this.products;

                 });

               }

               console.log(this.orders);

             });

      }

    )

  }

  completeOrder(order\_id:any){

Swal.fire({

  title: 'Are you sure?',

  text: "You Want Confirm This Order!",

  icon: 'question',

  showCancelButton: true,

  confirmButtonColor: '#3085d6',

  cancelButtonColor: '#d33',

  confirmButtonText: 'Yes, confirm it!',

}).then((result) => {

  if (result.isConfirmed) {

    this.userService.completeOrder(order\_id).subscribe((data) => {

      Swal.fire('Order Completed!','','success');

      this.userService.cartStatus.next(true);

    });

  }

});

  }

}

**File:-** **pending-orders.component.css**

.user-deatils{

    font-weight: bolder;

}

**Package:- pages.admin.** **update-category**

**File:- update-category.component.html**

<mat-card class="mt-2">

    <h1 matSubheader>Update Category</h1>

    <mat-card-content>

        <div class="container mt-3">

  <div class="row">

    <div class="col-md-6 offset-md-3 text-center">

     <mat-card-content>

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

        <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Category Name</mat-label>

          <input matInput placeholder="Category Name "

          [(ngModel)]="category.category\_name"

          name="category\_name" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Category Description</mat-label>

          <textarea matInput placeholder="Category Description"

          [(ngModel)]="category.category\_description"

          name="category\_desc" required cols="25" maxlength="100">

          </textarea>

        </mat-form-field>

        <mat-slide-toggle name="popular" class="example-margin" [(ngModel)]="category.popular" >

          Popular Category

        </mat-slide-toggle>

        <div class="container mt-5">

            <button type="submit" mat-raised-button color="primary">Update Category</button>

        </div>

      </form>

     </mat-card-content>

    </div>

  </div>

</div>

    </mat-card-content>

</mat-card>

**File:- update-category.component.ts**

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

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

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

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

import Swal from 'sweetalert2';

@Component({

  selector: 'app-update-category',

  templateUrl: './update-category.component.html',

  styleUrls: ['./update-category.component.css'],

})

export class UpdateCategoryComponent implements OnInit {

  category = {

    category\_name: '',

    category\_description: '',

    popular: false,

  };

  cat\_id = null;

  constructor(

    private productService: ProductService,

    private route: ActivatedRoute,

    private snack: MatSnackBar,

    private router: Router

  ) {}

  ngOnInit(): void {

    this.cat\_id = this.route.snapshot.params['cat\_id'];

    this.productService.getCategory(this.cat\_id).subscribe(

      (data: any) => {

        this.category = data;

        console.log(this.category.popular);

      },

      (error) => {

        this.snack.open('Something Went Wrong', 'ok');

      }

    );

  }

  public updateCategory() {

    this.productService.updateCategory(this.category).subscribe(

      (data) => {

        Swal.fire({

          position: 'top-end',

          icon: 'success',

          title: 'Category updated successfully!',

          showConfirmButton: true,

        });

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

      },

      (error) => {

        Swal.fire({

          position: 'top-end',

          icon: 'error',

          title: 'Unable To Update Category!',

          showConfirmButton: true,

        });

      }

    );

  }

}

**Package:- pages.admin** **update-product**

**File:- update-product.component.html**

<mat-card class="mt-2">

  <h1 matSubheader>Update Product</h1>

  <mat-card-content>

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

      <div class="row">

        <div class="col-md-6 text-center">

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Name</mat-label>

            <input matInput placeholder="Product Name " [(ngModel)]="product.product\_name" name="product\_name"

              required />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Description</mat-label>

            <textarea matInput placeholder="Product Description" [(ngModel)]="product.product\_description"

              name="product\_desc" required cols="25" maxlength="100">

                      </textarea>

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Price</mat-label>

            <input matInput placeholder="Product Price " [(ngModel)]="product.product\_price" name="product\_price"

              required type="number" />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Discount</mat-label>

            <input matInput placeholder="Product Discount " [(ngModel)]="product.product\_discount"

              name="product\_discount" required type="number" />

          </mat-form-field>

          <mat-form-field appearance="outline" class="w-100">

            <mat-label>Enter Product Stock</mat-label>

            <input matInput placeholder="Product Stock " [(ngModel)]="product.product\_stock" name="product\_stock"

              required type="number" />

          </mat-form-field>

          <mat-form-field appearance="fill" class="w-100">

            <mat-label> Category </mat-label>

            <mat-select name="category" [(ngModel)]="product.category.id">

              <mat-option \*ngFor="let c of categories" [value]="c.id">

                {{c.category\_name}}

              </mat-option>

            </mat-select>

          </mat-form-field>

        </div>

        <div class="col-md-6">

          <input type="file" matTooltip="Don't select same image" class="" value="select" #selectFile multiple

            (change)="onFileSelected($event)">

          <mat-grid-list cols="4" rowHeight="2:1" class="mt-2">

            <mat-grid-tile \*ngFor="let productImage of product.productImages ; let i=index">

              <div style="position:relative">

                <mat-icon class="btn-remove-image" color="warn" (click)="removeImage(i)" matTooltip="Remove Image">

                  remove\_circle</mat-icon>

                <img [src]="productImage.url" alt="" width="55px" height="55px">

              </div>

            </mat-grid-tile>

          </mat-grid-list>

        </div>

        <div class="container text-center mt-2">

          <button type="submit" mat-raised-button color="primary">

            Update Product

          </button>

        </div>

      </div>

    </form>

  </mat-card-content>

</mat-card>

**File:- update-product.component.ts**

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

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

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

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

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

import { FileHandle } from 'src/model/file-handle.model';

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

import Swal from 'sweetalert2';

@Component({

  selector: 'app-update-product',

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

  styleUrls: ['./update-product.component.css'],

})

export class UpdateProductComponent implements OnInit {

  categories: any = [];

  numberOfImages = 0;

  product\_id = null;

  product:Product = {

    id:0,

    product\_name: '',

    product\_description: '',

    product\_price: 0,

    product\_discount: 0,

    product\_stock: 0,

    product\_selling\_price: 0,

    productImages:[],

    category: {

      id: 0,

      category\_name: '',

      category\_description: '',

    },

  };

  constructor(

    private productService: ProductService,

    private route: ActivatedRoute,

    private snack: MatSnackBar,

    private router: Router,

    private imageProcessing:ImageProcessingService,

    private sanitizer:DomSanitizer

  ) {}

  ngOnInit(): void {

    this.product\_id = this.route.snapshot.params['product\_id'];

    this.productService.getCategories().subscribe(

      (data: any) => {

        this.categories = data;

      },

      (error) => {

        this.snack.open('Unable to load product data', 'Ok');

      }

    );

    this.productService.getProduct(this.product\_id).subscribe(

      (data: any) => {

        this.product = data;

        this.product = this.imageProcessing.createProductImages(this.product);

      },

      (error) => {

        this.snack.open('Unable to load categories data', 'Ok');

      }

    );

  }

  updateProduct() {

    this.product.product\_selling\_price =

      this.product.product\_price -

      (this.product.product\_price / 100) \* this.product.product\_discount;

    const productFormData = this.prepareFormData(this.product);

    this.productService.updateProduct(productFormData).subscribe(

      (data) => {

        Swal.fire({

          position: 'top-end',

          icon: 'success',

          title: 'Product updated successfully!',

          showConfirmButton: true,

        });

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

      },

      (error) => {

        Swal.fire({

          position: 'top-end',

          icon: 'error',

          title: 'Unable To Update Product!',

          showConfirmButton: true,

        });

      }

    );

  }

  removeImage(i: any) {

    this.product.productImages.splice(i, 1);

  }

  onFileSelected(event: any) {

    this.numberOfImages++;

    console.log(event);

    if (event.target.files) {

      const file = event.target.files[0];

      const fileHandle: FileHandle = {

        file: file,

        url: this.sanitizer.bypassSecurityTrustUrl(

          window.URL.createObjectURL(file)

        ),

      };

      console.log(this.product.productImages);

      if (this.numberOfImages <= 4) {

        this.product.productImages.push(fileHandle);

      } else {

        this.snack.open('You can add 4 images only.', 'OK', {

          horizontalPosition: 'right',

          verticalPosition: 'top',

        });

      }

    }

  }

  prepareFormData(product: Product): FormData {

    const formData = new FormData();

    formData.append(

      'product',

      new Blob([JSON.stringify(product)], { type: 'application/json' })

    );

    for (let i = 0; i < this.product.productImages.length; i++) {

      formData.append(

        'imageFile',

        this.product.productImages[i].file,

        this.product.productImages[i].file.name

      );

    }

    return formData;

  }

}

**Package:- pages.admin.view-product**

**File:- view-product.component.html**

<mat-card class="mt-2">

    <h1 matSubheader>Product List</h1>

  <div class="button-container">

    <button mat-raised-button color="primary" routerLink="/admin/add-product">

        <mat-icon color="accent">add\_circle</mat-icon> New Product

    </button>

 </div>

    <table mat-table [dataSource]="products" class="w-100 mat-elevation-z8">

        <ng-container matColumnDef="position">

            <th mat-header-cell \*matHeaderCellDef> No. </th>

            <td mat-cell \*matCellDef="let element"> {{element.id}} </td>

        </ng-container>

        <ng-container matColumnDef="product\_name">

            <th mat-header-cell \*matHeaderCellDef>Name</th>

            <td mat-cell \*matCellDef="let element"> {{element.product\_name}} </td>

        </ng-container>

        <ng-container matColumnDef="product\_price">

            <th mat-header-cell \*matHeaderCellDef> Price </th>

            <td mat-cell \*matCellDef="let element"> {{element.product\_price | currency:'INR'}} </td>

        </ng-container>

        <ng-container matColumnDef="product\_discount">

            <th mat-header-cell \*matHeaderCellDef> Discount </th>

            <td mat-cell \*matCellDef="let element"> {{element.product\_discount}} </td>

        </ng-container>

        <ng-container matColumnDef="product\_selling\_price">

            <th mat-header-cell \*matHeaderCellDef> Selling Price </th>

            <td mat-cell \*matCellDef="let element"> {{element.product\_selling\_price | currency:'INR'}} </td>

        </ng-container>

        <ng-container matColumnDef="category">

            <th mat-header-cell \*matHeaderCellDef> Category </th>

            <td mat-cell \*matCellDef="let element"> {{element.category.category\_name}} </td>

        </ng-container>

        <ng-container matColumnDef="actions">

            <th mat-header-cell \*matHeaderCellDef> actions </th>

            <td mat-cell \*matCellDef="let element">

                <mat-icon color="warn" (click)="deleteProduct(element.id)">delete</mat-icon>

                <mat-icon class="ms-1" color="primary" [routerLink]="'/admin/update-product/'+element.id">edit</mat-icon>

                <mat-icon class="ms-1" (click)="showImages(element)">images</mat-icon>

            </td>

        </ng-container>

        <tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

        <tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

        <tr class="mat-row" \*matNoDataRow>

            <td class="mat-cell" colspan="4">No data matching the filter</td>

        </tr>

    </table>

**File:- view-product.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

import { ImageProcessingService } from 'src/app/services/image-processing.service';

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

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

import Swal from 'sweetalert2';

import { ViewProductImagesComponent } from '../view-product-images/view-product-images.component';

@Component({

  selector: 'app-view-product',

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

  styleUrls: ['./view-product.component.css'],

})

export class ViewProductComponent implements OnInit {

  displayedColumns: string[] = [

    'position',

    'product\_name',

    'product\_price',

    'product\_discount',

    'product\_selling\_price',

    'category',

    'actions'

  ];

  products: any = [];

  constructor(

    private productService: ProductService,

    private imageDialog: MatDialog,

    private imageProcessingService: ImageProcessingService

  ) {}

  ngOnInit(): void {

    this.productService.getProducts().subscribe(

      (data: any) => {

        this.products = data;

        for (let p of this.products) {

          p = this.imageProcessingService.createProductImages(p);

        }

      },

      (error) => {

        alert('unable to load products');

      }

    );

  }

  deleteProduct(id: any) {

    Swal.fire({

      title: 'Are you sure?',

      text: "You won't be able to revert this!",

      icon: 'warning',

      showCancelButton: true,

      confirmButtonColor: '#3085d6',

      cancelButtonColor: '#d33',

      confirmButtonText: 'Yes, delete it!',

    }).then((result) => {

      if (result.isConfirmed) {

        this.productService.deleteProduct(id).subscribe(

          (data) => {

            Swal.fire({

              position: 'top-end',

              icon: 'success',

              title: 'Product deleted successfully!',

              showConfirmButton: true,

            });

            location.reload();

          },

          (error) => {

            Swal.fire({

              position: 'top-end',

              icon: 'error',

              title: 'Unable to delete product!',

              text:'Already orders placed with this product, if you want you can update this product.',

              showConfirmButton: true,

            });

          }

        );

      }

    });

  }

  showImages(product: Product) {

    console.log(product);

    this.imageDialog.open(ViewProductImagesComponent, {

      data: {

        images: product.productImages,

        productTitle: product.product\_name,

      },

      height: '250px',

      width: '1000px',

      disableClose: false,

    });

  }

}

**File:- view-product.component.css**

.product-box{

    border: 1px solid #e1e1e1;

    margin-bottom: 10px;

    box-shadow: 2px #BDBDBD;

}

image-preview{

    border: 1px solid grey;

}

.button-container {

    padding-bottom: 16px;

}

**Package:- pages.admin. view-product-images**

**File:- view-product-images.component.html**

<h1 mat-dialog-title> {{data.productTitle}} Images </h1>

<mat-grid-list cols="4" rowHeight="2:1">

    <mat-grid-tile \*ngFor="let fileHandle of data.images">

        <img [src]="fileHandle.url" alt="" width="60px" height="60px">

    </mat-grid-tile>

</mat-grid-list>

**File:- view-product-images.component.ts**

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

import { MatDialog , MAT\_DIALOG\_DATA } from '@angular/material/dialog';

@Component({

  selector: 'app-view-product-images',

  templateUrl: './view-product-images.component.html',

  styleUrls: ['./view-product-images.component.css']

})

export class ViewProductImagesComponent implements OnInit {

  constructor(@Inject(MAT\_DIALOG\_DATA) public data:any) { }

  ngOnInit(): void {

  }

}

**File:- view-product-images.component.css**

mat-grid-tile{

    border: 1.5px solid grey;

}

mat-dialog{

    border: 2.5px solid grey;

}

          title: 'Unable To Update Product!',

          showConfirmButton: true,

        });

      }

    );

  }

  removeImage(i: any) {

    this.product.productImages.splice(i, 1);

  }

  onFileSelected(event: any) {

    this.numberOfImages++;

    console.log(event);

    if (event.target.files) {

      const file = event.target.files[0];

      const fileHandle: FileHandle = {

        file: file,

        url: this.sanitizer.bypassSecurityTrustUrl(

          window.URL.createObjectURL(file)

        ),

      };

      console.log(this.product.productImages);

      if (this.numberOfImages <= 4) {

        this.product.productImages.push(fileHandle);

      } else {

        this.snack.open('You can add 4 images only.', 'OK', {

          horizontalPosition: 'right',

          verticalPosition: 'top',

        });

      }

    }

  }

  prepareFormData(product: Product): FormData {

    const formData = new FormData();

    formData.append(

      'product',

      new Blob([JSON.stringify(product)], { type: 'application/json' })

    );

    for (let i = 0; i < this.product.productImages.length; i++) {

      formData.append(

        'imageFile',

        this.product.productImages[i].file,

        this.product.productImages[i].file.name

      );

    }

    return formData;

  }

}

**Package:- pages.admin. view-users**

**File:- view-users.component.html**

<mat-card>

    <h1>Users List</h1>

    <table mat-table [dataSource]="users" class="mat-elevation-z8">

        <!-- Position Column -->

        <ng-container matColumnDef="position">

            <th mat-header-cell \*matHeaderCellDef> No. </th>

            <td mat-cell \*matCellDef="let element"> {{element.id}} </td>

        </ng-container>

        <!-- Name Column -->

        <ng-container matColumnDef="name">

            <th mat-header-cell \*matHeaderCellDef> Name </th>

            <td mat-cell \*matCellDef="let element"> {{element.first\_name}} {{element.last\_name}}</td>

        </ng-container>

        <!-- Weight Column -->

        <ng-container matColumnDef="user\_role">

            <th mat-header-cell \*matHeaderCellDef> Role </th>

            <td mat-cell \*matCellDef="let element"> {{element.authorities[0].authority}} </td>

        </ng-container>

        <!-- Symbol Column -->

        <ng-container matColumnDef="email">

            <th mat-header-cell \*matHeaderCellDef> Email </th>

            <td mat-cell \*matCellDef="let element"> {{element.user\_email}} </td>

        </ng-container>

        <ng-container matColumnDef="phone">

            <th mat-header-cell \*matHeaderCellDef> Phone </th>

            <td mat-cell \*matCellDef="let element"> {{element.user\_phone}} </td>

        </ng-container>

        <tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

        <tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

        <!-- Row shown when there is no matching data. -->

        <tr class="mat-row" \*matNoDataRow>

            <td class="mat-cell" colspan="4">No data matching the filter</td>

        </tr>

    </table>

</mat-card>

**File:- view-users.component.ts**

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

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

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-view-users',

  templateUrl: './view-users.component.html',

  styleUrls: ['./view-users.component.css'],

})

export class ViewUsersComponent implements OnInit {

  displayedColumns: string[] = ['position', 'name', 'user\_role', 'email','phone'];

  public users: any = [];

  dataSource:any = [];

  constructor(private userService: UserService, private snack: MatSnackBar) {}

  ngOnInit(): void {

    this.userService.getUsers().subscribe(

      (data) => {

        this.users = data;

      },

      (error) => {

        this.snack.open('Unable to load users', 'OK');

      }

    );

  }

  applyFilter(event: Event) {

    const filterValue = (event.target as HTMLInputElement).value;

    console.log(filterValue);

    this.users.filter = filterValue.trim().toLowerCase();

  }

}

**File:- view-users.component.css**

table {

    width: 100%;

}

.mat-form-field {

    font-size: 14px;

    width: 100%;

}

**Package:- pages.change-password**

**File:-change-password.component.html**

<div class="container mt-3">

    <div class="row">

        <div class="col-md-6 offset-md-3 text-center">

            <mat-card class="bg-light mb-3">

                <mat-card-title>

                    <div class="container text-center">

                        <img src="./../../../assets/logo.jpg" alt="medicare" width="75px" />

                    </div>

                    <h1 class="text-center">Change Password</h1>

                </mat-card-title>

                <mat-card-content>

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

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Current Password</mat-label>

                            <input matInput placeholder="Current Password" [(ngModel)]="current\_password" name="current-password"

                                required />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>New Password</mat-label>

                            <input matInput placeholder="New Password" [(ngModel)]="new\_password" name="password"

                                required  />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Confrim Password</mat-label>

                            <input matInput

                                   placeholder="Confirm Password"

                                   [(ngModel)]="confirm\_password"

                                   name="password"

                                   required

                                   (keyup)="compare()"

                                />

                            <mat-hint \*ngIf="!status" class="error">

                                New Password & Confirm Must Be Same

                            </mat-hint>

                        </mat-form-field>

                        <div class="container mt-3">

                            <button type="submit" mat-raised-button color="primary">Change Password</button>

                        </div>

                    </form>

                </mat-card-content>

            </mat-card>

        </div>

    </div>

</div>

**File:- change-password.component.ts**

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

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-change-password',

  templateUrl: './change-password.component.html',

  styleUrls: ['./change-password.component.css'],

})

export class ChangePasswordComponent implements OnInit {

  current\_password: string = '';

  new\_password:string = '';

  confirm\_password:string = '';

  status= true;

  user:any={

    password:''

  };

  check\_current\_password:any=true;

  constructor(

    private userService: UserService,

    private loginService: LoginService,

    private snack:MatSnackBar,

    private router:Router

  ) {}

  ngOnInit(): void {

    this.userService.getUser(this.loginService.getUser().id).subscribe(

      (data)=>{

        this.user=data;

      }

    )

  }

  compare() {

    if( this.new\_password.substring(0,this.confirm\_password.length ) == this.confirm\_password  ){

      this.status=true;

    }else{

      this.status = false;

    }

  }

  changePassword() {

    if(!this.status){

      this.snack.open('Confirm Password and New Password must same','OK');

      return;

    }

    this.user = this.loginService.getUser();

    this.user.password= this.current\_password;

    this.userService.confirmPassword(this.user).subscribe(

      (data)=>{

        this.check\_current\_password = data;

        console.log(this.check\_current\_password);

      }

    )

      if(this.check\_current\_password){

        this.user.password=this.confirm\_password;

        this.userService.changePassword(this.user).subscribe(

          (data)=>{

            this.snack.open('Password changed successfully', 'OK');

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

            this.loginService.logout();

          }

        )

      }else{

        this.snack.open('Current password is not correct','OK')

      }

  }

}

**File:- change-password.component.css**

.error{

    color: brown;

}

**Package:- pages. home**

**File:-home.component.html**

<div id="carouselExampleSlidesOnly" class="carousel slide mt-3" data-bs-ride="carousel">

    <div class="carousel-inner">

        <div class="carousel-item active " data-bs-interval="1500">

            <img src="../../../assets/slide\_one.jpg" class="d-block w-100" alt="...">

        </div>

        <div class="carousel-item" data-bs-interval="1500">

            <img src="../../../assets/slide\_two.jpg" class="d-block w-100" alt="...">

        </div>

        <div class="carousel-item" data-bs-interval="1500">

            <img src="../../../assets/slide\_three.jpg" class="d-block w-100" alt="...">

        </div>

        <div class="carousel-item" data-bs-interval="1500">

            <img src="../../../assets/slide\_four.jpg" class="d-block w-100" alt="...">

        </div>

        <div class="carousel-item" data-bs-interval="1500">

            <img src="../../../assets/slide\_six.jpg" class="d-block w-100" alt="...">

        </div>

    </div>

    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleSlidesOnly"

        data-bs-slide="prev">

        <span class="carousel-control-prev-icon" aria-hidden="true"></span>

        <span class="visually-hidden">Previous</span>

    </button>

    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleSlidesOnly"

        data-bs-slide="next">

        <span class="carousel-control-next-icon" aria-hidden="true"></span>

        <span class="visually-hidden">Next</span>

    </button>

</div>

<div class="container mt-3 ">

    <div class="row">

        <div class="col-md-12">

            <ul class="list-group list-group-horizontal-sm">

                <button class="cat-btn ms-1" \*ngFor="let c of categories" [routerLink]="'/home/'+c.id" >

                    {{c.category\_name}}

                </button>

                <mat-chip-list>

                    <mat-chip color="accent" selected><span  routerLink="/home/0">View All</span></mat-chip>

                </mat-chip-list>

            </ul>

        </div>

    </div>

</div>

<div class="container-fluid showcase ms-2 mt-3 ">

    <div class="row mt-3">

        <div class="col-md-12 mt-3">

                    <div class="row row-cols-1 row-cols-md-6 ms-5 mt-2 mb-5" >

                        <mat-card \*ngFor="let p of products" class="ms-2 mt-1" >

                            <mat-card-header class="text-center">

                            </mat-card-header>

                            <mat-card-content class="text-center">

                                <img (click)="showDetails(p)" [src]="p.productImages[0].url" alt="" width="75px" height="75px">

                                <mat-divider></mat-divider>

                                <mat-card-subtitle class="mt-1" (click)="showDetails(p)">

                                    {{p.product\_name|productTitle}}

                                    <span \*ngIf="p.product\_name.length >= 35" [matTooltip]="p.product\_name">...</span>

                                    <span class="empty-space" \*ngIf="p.product\_name.length <= 22" >..........</span>

                                </mat-card-subtitle>

                                <div matLine>

                                    Price: {{p.product\_selling\_price|currency:'INR'}}

                                    <s class="ms-1" \*ngIf="p.product\_selling\_price != p.product\_price">{{p.product\_price|currency:'INR'}}</s>

                                </div>

                                <span class="discount" \*ngIf="p.product\_discount" >

                                    Upto  {{p.product\_discount/100|percent}} off

                                </span>

                                <span class="no-discount" \*ngIf="! p.product\_discount" >

                                    Upto {{p.product\_discount/100|percent}} off

                                </span>

                            </mat-card-content>

                            <mat-card-actions class="text-center">

                                <button mat-button color="primary" (click)="addToCart(p)" \*ngIf="!productInCart(p.id)">

                                    <mat-icon>add\_shopping\_cart</mat-icon>

                                    Add To Cart

                                </button>

                                <button mat-flat-button class="cart" (click)="gotoCart()" \*ngIf="productInCart(p.id)">

                                    <mat-icon class="cart">check</mat-icon>

                                    Go To Cart

                                </button>

                            </mat-card-actions>

                        </mat-card>

                    </div>

        </div>

    </div>

</div>

**File:-home.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

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

import { ActivatedRoute, NavigationEnd, NavigationStart, Router } from '@angular/router';

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

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

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

import { ProductDetailsComponent } from '../product-details/product-details.component';

import { ViewCartComponent } from '../user/view-cart/view-cart.component';

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css'],

})

export class HomeComponent implements OnInit {

  products: any = [];

  categories: any = [];

  cart\_products: any = [];

  cat\_id = 0;

  cart\_id = 0;

  constructor(

    private productService: ProductService,

    private imageProcessingService: ImageProcessingService,

    private route: ActivatedRoute,

    private snack: MatSnackBar,

    private router: Router,

    private loginService: LoginService,

    private userService: UserService,

    private cartDialog: MatDialog

  ) {}

  ngOnInit(): void {

    this.productService.getPopularCategories().subscribe(

      (data) => {

        this.categories = data;

      },

      (error) => {

          Swal.fire({

            icon: 'error',

            title: 'Server Error !',

            text: 'Server is down, sorry for the inconvenience!',

          });

        this.loginService.logout();

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

      }

    );

    if (this.cat\_id == undefined) {

      this.cat\_id = 0;

    }

    this.cat\_id = this.route.snapshot.params['cat\_id'];

    if (this.cat\_id == 0) {

      this.productService.getProducts().subscribe(

        (data) => {

          this.products = data;

          for (let p of this.products) {

            p = this.imageProcessingService.createProductImages(p);

          }

        },

        (error) => {

            this.loginService.logout();

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

        }

      );

    } else if (this.cat\_id > 0) {

      this.productService.getProductsByCategory(this.cat\_id).subscribe(

        (data) => {

          this.products = data;

          for (let p of this.products) {

            p = this.imageProcessingService.createProductImages(p);

          }

        },

        (error) => {

           Swal.fire({

             icon: 'error',

             title: 'Server Error !',

             text: 'Server is down, sorry for the inconvenience!',

           });

            this.loginService.logout();

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

        }

      );

    }

    this.router.events.subscribe((event: any) => {

      if (event instanceof NavigationEnd) {

        this.cat\_id = this.route.snapshot.params['cat\_id'];

        if (this.cat\_id == 0) {

          this.productService.getProducts().subscribe(

            (data) => {

              this.products = data;

              for (let p of this.products) {

                p = this.imageProcessingService.createProductImages(p);

              }

            },

            (error) => {

                this.loginService.logout();

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

            }

          );

        } else if (this.cat\_id > 0) {

          this.productService.getProductsByCategory(this.cat\_id).subscribe(

            (data) => {

              this.products = data;

              for (let p of this.products) {

                p = this.imageProcessingService.createProductImages(p);

              }

            },

            (error) => {

               this.loginService.logout();

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

            }

          );

        }

      }

    });

    this.userService

      .getCart(this.loginService.getUser().id)

      .subscribe((data: any) => {

        this.cart\_products = data.products;

        console.log(this.cart\_products);

      });

    this.userService.cartStatus.asObservable().subscribe((data) => {

      this.userService

        .getCart(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.cart\_products = data.products;

          console.log(this.cart\_products);

        });

    });

    this.userService.getCart(this.loginService.getUser().id).subscribe(

      (data: any) => {

        this.cart\_id = data.id;

        console.log('Cart ID' + this.cart\_id);

      },

      (error) => console.log(error)

    );

    console.log(this.cart\_products);

  }

  public addToCart(product: any) {

    if (!this.loginService.isLoggedIn()) {

        this.snack.open('You must login to add product to cart', 'OK');

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

      return;

    }

    if (this.loginService.getUserRole() == 'NORMAL') {

      const user\_id = this.loginService.getUser().id;

      console.log('User ID' + user\_id);

      this.userService.getCart(user\_id).subscribe(

        (data: any) => {

          this.cart\_id = data.id;

          console.log('Cart ID' + this.cart\_id);

          this.userService.cartStatus.next(true);

        },

        (error) => console.log(error)

      );

      this.userService.addProductToCart(this.cart\_id, product).subscribe(

        (data) => {

          this.userService.cartStatus.next(true);

          this.userService

            .getCart(this.loginService.getUser().id)

            .subscribe((data: any) => {

              this.cart\_products = data.products;

            });

        },

        (error) => {

            Swal.fire({

              icon: 'error',

              title: 'Server Down !',

              text: 'Something went wrong product not added to cart, sorry for the inconvenience!',

            });

        }

      );

    } else if (this.loginService.getUserRole() == 'ADMIN') {

      this.snack.open('This service is for normal user only', 'OK');

    }

  }

  public productInCart(product\_id: any) {

    let result = false;

    for (let p of this.cart\_products) {

      if (p.id == product\_id) {

        result = true;

      }

    }

    return result;

  }

  gotoCart() {

      this.cartDialog.open(ViewCartComponent, {

        height: '500px',

        width: '900px',

        position: {

          right: '0',

          top: '0',

        },

      });

  }

  showDetails(product:any){

     this.cartDialog.open(ProductDetailsComponent, {

       height: '700px',

       width:  '900px',

       data: product,

       position:{

        top: '50px',

       }

     });

  }

}

**File:-home.component.css**

mat-card{

    border: 1.5px solid grey;

    border-radius: 15px;

}

.categories{

    border: 1.5px solid grey;

    background: white;

}

.discount{

 color: green;

}

.no-discount {

    color: white;

}

.menu{

    border-bottom: 1px solid lightgrey;

}

.cart{

    color: green;

}

.list-group-item {

    cursor: pointer;

}

.cat-btn{

    color: white;

    outline: none;

    border: none;

    background-color: #757ce8 ;

}

.empty-space{

    color: white;

}

.all-cat{

    font-size: large;

}

**Package:- pages. login**

**File:-login.component.html**

<div class="container mt-3">

  <div class="row">

    <div class="col-md-6 offset-md-3 text-center">

      <mat-card class="bg-light mb-3">

        <mat-card-title>

             <div class="container text-center">

        <img src="./../../../assets/logo.jpg" alt="medicare" width="75px" />

      </div>

      <h1  class="text-center">Login Here</h1>

      </mat-card-title>

     <mat-card-content>

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

        <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Username</mat-label>

          <input matInput placeholder="Username "

          [(ngModel)]="loginData.username"

          name="username" required/>

          <mat-hint>Username must be unique !!!!</mat-hint>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Password</mat-label>

          <input matInput placeholder="Password "

          [(ngModel)]="loginData.password"

          name="password" required type="password"/>

        </mat-form-field>

        <div class="container">

            <button type="submit" mat-raised-button color="primary">Login</button>

        </div>

      </form>

     </mat-card-content>

      </mat-card>

    </div>

  </div>

</div>

**File:-login.component.ts**

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

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  loginData={

    username:'',

    password:''

  }

  constructor(private snackbar:MatSnackBar,

              private loginService:LoginService,

              private route:Router,

              private userService:UserService) { }

  ngOnInit(): void {

  }

  loginUser(){

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

      this.snackbar.open("Username is required","Ok");

      return

    }

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

        this.snackbar.open('Password is required', 'Ok');

        return

      }

      this.loginService.generateToken(this.loginData).subscribe(

        (data:any)=>{

          console.log('success');

          console.log(data);

          this.loginService.loginUser(data.token);

          this.loginService.getCurrentUser().subscribe(

            (user)=>{

              this.loginService.setUser(user);

              console.log(user);

              if(this.loginService.getUserRole() =='ADMIN'){

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

                 location.href = "admin";

                this.loginService.loginStatusSubject.next(true);

              }else if(this.loginService.getUserRole() == 'NORMAL'){

                //  this.route.navigate(['']);

                 location.href="";

                 this.loginService.loginStatusSubject.next(true);

                 this.userService.cartStatus.next(true);

              }else{

                this.loginService.logout();

                location.reload();

              }

            },

            (error)=>{

              Swal.fire({

                icon: 'error',

                title: 'Invalid Username or Password',

                text: 'If you are new ? .. Register',

              });

            }

          )

        },

        (error)=>{

           Swal.fire({

             icon: 'error',

             title: 'Invalid Username or Password',

             text: 'If you are new ? .. Register',

           });

        }

      )

  }

}

**Package:- pages. Product-details**

**File:-product-details.component.html**

<mat-icon color="warn" mat-dialog-close>cancel</mat-icon>

<mat-card class="example-card mt-2">

    <div class="container-fluid">

        <div class="row">

            <div class="col-md-6">

                <div class="text-center">

                    <mat-icon \*ngIf="!prev\_btn" class="float-start" (click)="prev()" color="primary">arrow\_back\_ios

                    </mat-icon>

                    <mat-icon \*ngIf="!next\_btn" class="float-end" (click)="next()" color="primary">arrow\_forward\_ios

                    </mat-icon>

                </div>

                <div class="text-center"><img height="350px" width="350px" [src]="productImages[position].url" alt="IMAGE"></div>

                <img class="mt-1 ms-1 preview-image" (click)="changePosition(n)"

                    \*ngFor="let i of productImages; let n = index" [src]="i.url" width="60px" alt="">

            </div>

            <div class="col-md-6">

                <h2 class="product\_title mt-5" mat-card-subtitle>{{product.product\_name}}</h2>

              <mat-chip-list>

                 <mat-chip color="accent" selected>

                     {{product.category.category\_name}}

                </mat-chip></mat-chip-list>

                <mat-card-subtitle class="pricing mt-2">

                    <h3 matLine>

                        MRP: {{product.product\_selling\_price|currency:'INR'}}

                        <s class="ms-1"

                            \*ngIf="product.product\_selling\_price != product.product\_price">{{product.product\_price|currency:'INR'}}</s>

                        <span class="discount" \*ngIf="product.product\_discount">

                            Upto {{product.product\_discount/100|percent}} off

                        </span>

                        <span class="no-discount" \*ngIf="! product.product\_discount">

                            Upto {{product.product\_discount/100|percent}} off

                        </span>

                    </h3>

                </mat-card-subtitle>

                <h5 mat-card-subtitle>(Inclusive of all taxes)</h5>

                <h5 mat-card-subtitle>\* Mkt: Medicare Pvt Ltd</h5>

                <h5 mat-card-subtitle>\* Country of Origin: India</h5>

                <h5 mat-card-subtitle>\* Delivery charges if applicable will be applied at checkout</h5>

                <mat-card-actions>

                    <button mat-button color="primary" (click)="addToCart(product)" \*ngIf="! productInCart(product.id)">

                        <mat-icon>add\_shopping\_cart</mat-icon>

                        Add To Cart

                    </button>

                    <button mat-flat-button mat-dialog-close class="cart" (click)="gotoCart()" \*ngIf="productInCart(product.id)">

                        <mat-icon class="cart">check</mat-icon>

                        Go To Cart

                    </button>

                </mat-card-actions>

            </div>

        </div>

    </div>

   <h2 class="mt-3" mat-card-title>Product Details :</h2>

   <p>{{product.product\_description}}</p>

</mat-card>

<!-- <mat-card-header>

        <div class="container text-center">

        </div>

    </mat-card-header>

    <div class="conatiner text-center mt-2">

    </div>

    <div class="conatiner text-center mt-2">

    </div>

    <mat-card-content class="mt-2">

        <div matSubheader>Description :</div>

        <p>

            {{product.product\_description}}

        </p>

    </mat-card-content> -->

**File:-product-details.component.ts**

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

import { MatDialog, MAT\_DIALOG\_DATA } from '@angular/material/dialog';

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

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

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

import { UserService } from 'src/app/services/user.service';

import { ViewCartComponent } from '../user/view-cart/view-cart.component';

@Component({

  selector: 'app-product-details',

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

  styleUrls: ['./product-details.component.css'],

})

export class ProductDetailsComponent implements OnInit{

  products: any = [];

  categories: any = [];

  cart\_products: any = [];

  cat\_id = 0;

  cart\_id = 0;

  productImages: any = [];

  position = 0;

  max = 0;

  prev\_btn = false;

  next\_btn = false;

  constructor(

    private productService: ProductService,

    private imageProcessingService: ImageProcessingService,

    private route: ActivatedRoute,

    private snack: MatSnackBar,

    private router: Router,

    private loginService: LoginService,

    private userService: UserService,

    @Inject(MAT\_DIALOG\_DATA) public product: any,

    private cartDialog: MatDialog

  ) {}

  ngOnInit(): void {

    this.productImages = this.product.productImages;

    this.max = this.productImages.length;

    this.userService.getCart(this.loginService.getUser().id).subscribe(

      (data: any) => {

        this.cart\_id = data.id;

        console.log('Cart ID' + this.cart\_id);

      },

      (error) => console.log(error)

    );

    this.userService

      .getCart(this.loginService.getUser().id)

      .subscribe((data: any) => {

        this.cart\_products = data.products;

        console.log(this.cart\_products);

      });

    this.userService.cartStatus.asObservable().subscribe((data) => {

      this.userService

        .getCart(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.cart\_products = data.products;

          console.log(this.cart\_products);

        });

    });

    console.log(this.cart\_products);

  }

  public addToCart(product: any) {

    if (!this.loginService.isLoggedIn()) {

      this.snack.open('You must login to add product to cart', 'OK');

      return;

    }

    if (this.loginService.getUserRole() == 'NORMAL') {

      const user\_id = this.loginService.getUser().id;

      console.log('User ID' + user\_id);

      this.userService.getCart(user\_id).subscribe(

        (data: any) => {

          this.cart\_id = data.id;

          console.log('Cart ID' + this.cart\_id);

          this.userService.cartStatus.next(true);

        },

        (error) => console.log(error)

      );

      this.userService.addProductToCart(this.cart\_id, product).subscribe(

        (data) => {

          this.snack.open('Product added to cart', 'OK');

          this.userService.cartStatus.next(true);

          this.userService

            .getCart(this.loginService.getUser().id)

            .subscribe((data: any) => {

              this.cart\_products = data.products;

            });

        },

        (error) => {

          this.snack.open('Product not added to cart', 'OK');

        }

      );

    } else if (this.loginService.getUserRole() == 'ADMIN') {

      this.snack.open('This service is for normal user only', 'OK');

    }

  }

  public productInCart(product\_id: any) {

    let result = false;

    for (let p of this.cart\_products) {

      if (p.id == product\_id) {

        result = true;

      }

    }

    return result;

  }

  gotoCart() {

    this.cartDialog.open(ViewCartComponent, {

      height: '500px',

      width: '900px',

      position: {

        right: '0',

        top: '0',

      },

    });

  }

  prev() {

    if (this.position == 0) {

      this.prev\_btn = true;

      return;

    }

    this.next\_btn = false;

    this.position--;

  }

  next() {

    if (this.position == this.max - 1) {

      this.next\_btn = true;

      return;

    }

    this.prev\_btn = false;

    this.position++;

  }

  changePosition(n:any){

    this.position=n;

  }

}

**File:-product-details.component.css**

.discount {

    color: green;

}

.no-discount {

    color: white;

}

.cart {

    color: green;

}

 .preview-image{

    border: 0.5px solid black;

    border-collapse: collapse;

 }

 .product\_title{

    font-weight: bolder;

 }

 .pricing{

    font-weight: bolder;

 }

**Package:- pages. profile**

**File:-profile.component.html**

<mat-card class="mt-2">

    <h1 class="text-center" matSubheader>Profile Information</h1>

    <mat-card-content>

        <div class="container">

            <div class="container text-center">

                <img src="./../../../assets/account\_icon.png" alt="" width="200px" class="profile">

                <h1>{{user.first\_name}} {{user.last\_name}}</h1>

            </div>

            <div class="row">

                <div class="col-md-10 offset-md-1" >

                    <table class="profile-table">

                        <tr>

                            <td>Username</td>

                            <td>{{user.username}}</td>

                        </tr>

                         <tr>

                            <td>Email</td>

                            <td>{{user.user\_email}}</td>

                        </tr>

                        <tr>

                            <td>Phone</td>

                            <td>{{user.user\_phone}}</td>

                        </tr>

                    </table>

                </div>

            </div>

        </div>

    </mat-card-content>

    <mat-card-actions class="text-center">

        <button mat-raised-button color="primary" [routerLink]="'/update-user/'+user.id">Update Profile</button>

        <button mat-raised-button color="accent" routerLink="/change-password">Change Password</button>

    </mat-card-actions>

</mat-card>

**File:-profile.component.ts**

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-profile',

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

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

})

export class ProfileComponent implements OnInit {

  user:any=null;

  constructor(private loginService:LoginService,

              private userService:UserService) { }

  ngOnInit(): void {

     this.userService

       .getUser(this.loginService.getUser().id)

       .subscribe((data: any) => {

         console.log(data);

         console.log(this.loginService.getUser().id);

         this.user = data;

       });

    this.userService.userStatus.asObservable().subscribe(

      (data)=>{

        this.userService.getUser(this.loginService.getUser().id).subscribe(

          (data:any)=>{

            console.log(data);

            this.user=data;

          }

        )

      }

    )

  }

}

**File:-profile.component.css**

.profile-table{

    width: 100%;

}

.profile-table tr td{

    padding: 20px 20px;

    border: 0.5px solid #E0E0E0;

}

.profile{

    border: 2.5px solid #607D8B;

    border-radius: 50%;

    max-width: 250px;

    max-height: 250px;

}

**Package:- pages. register**

**File:-register.component.html**

<div class="container mt-3">

  <div class="row">

    <div class="col-md-6 offset-md-3 text-center">

      <mat-card class="bg-light mb-3">

        <mat-card-title>

             <div class="container text-center">

        <img src="./../../../assets/logo.jpg" alt="medicare" width="75px" />

      </div>

      <h1  class="text-center">Register Here</h1>

      </mat-card-title>

     <mat-card-content>

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

        <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Username</mat-label>

          <input matInput placeholder="Username "

          [(ngModel)]="user.username"

          name="username" required/>

          <mat-hint>Username must be unique !!!!</mat-hint>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Password</mat-label>

          <input matInput placeholder="Password "

          [(ngModel)]="user.password"

          name="password" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your First Name</mat-label>

          <input matInput placeholder="First Name "

          [(ngModel)]="user.first\_name"

          name="first\_name" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Last Name</mat-label>

          <input matInput placeholder="Last Name "

          [(ngModel)]="user.last\_name"

          name="last\_name" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Email</mat-label>

          <input matInput placeholder="Email "

          [(ngModel)]="user.user\_email"

          name="user\_email" required/>

        </mat-form-field>

         <mat-form-field appearance="outline" class="w-100">

          <mat-label>Enter Your Phone</mat-label>

          <input matInput placeholder="Phone "

          [(ngModel)]="user.user\_phone"

          name="user\_phone" required/>

        </mat-form-field>

        <div class="container">

            <button type="submit" mat-raised-button color="primary">Register</button>

            <button type="reset" mat-raised-button color="warn" class="ms-3">Clear</button>

        </div>

      </form>

     </mat-card-content>

      </mat-card>

    </div>

  </div>

</div>

**File:-register.component.ts**

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

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

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

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-register',

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

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

})

export class RegisterComponent implements OnInit {

  user={

    username:'',

    password:'',

    first\_name:'',

    last\_name:'',

    user\_email:'',

    user\_phone:''

  }

  constructor(private userService:UserService,private snack:MatSnackBar,

              private router:Router) { }

  ngOnInit(): void {

  }

  registerUser(){

    if (this.user.username == '' || this.user.username == null ) {

      this.snack.open('Username is required','ok')

      return;

    }if (this.user.username.length>13 || this.user.username.length < 5 ) {

      this.snack.open('Username length should be less than 13 character or greater than 5 character', 'ok');

      return;

    }

    if (this.user.password == '' || this.user.password == null) {

      this.snack.open('Password is required', 'ok');

      return;

    } if (this.user.first\_name == '' || this.user.first\_name == null) {

      this.snack.open('First name is required', 'ok');

      return;

    }  if (this.user.last\_name == '' || this.user.last\_name == null) {

      this.snack.open('Last name is required', 'ok');

      return;

    }  if (this.user.user\_email == '' || this.user.user\_email == null) {

      this.snack.open('Email is required', 'ok');

      return;

    }  if (this.user.user\_phone == '' || this.user.user\_phone == null) {

      this.snack.open('Phone is required', 'ok');

      return;

    }

    this.userService.addUser(this.user).subscribe(

      (data)=>{

        console.log(data);

        Swal.fire({

          position: 'top-end',

          icon: 'success',

          title: 'Registration successful !',

          showConfirmButton: false,

          timer: 1500,

        });

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

      },

      (error)=>{

        console.log(error);

        Swal.fire({

          icon: 'error',

          title: 'Server Error !',

          text: 'Server is down, sorry for the inconvenience!',

        });

      }

    )

  }

}

**Package:- pages. search-product**

**File:-search-product.component.html**

<div class="row ">

    <div class="col-md-12 ">

        <div class="row row-cols-1 row-cols-md-6 ms-5 mt-2 mb-5">

            <mat-card \*ngFor="let p of products" class="ms-2 mt-1">

                <mat-card-header class="text-center">

                </mat-card-header>

                <mat-card-content class="text-center">

                    <img (click)="showDetails(p)" [src]="p.productImages[0].url" alt="" width="75px" height="75px">

                    <mat-divider></mat-divider>

                    <mat-card-subtitle class="mt-1" (click)="showDetails(p)">

                        {{p.product\_name|productTitle}}

                        <span \*ngIf="p.product\_name.length >= 35" [matTooltip]="p.product\_name">...</span>

                        <span class="empty-space" \*ngIf="p.product\_name.length <= 22">..........</span>

                    </mat-card-subtitle>

                    <div matLine>

                        Price: {{p.product\_selling\_price|currency:'INR'}}

                        <s class="ms-1"

                            \*ngIf="p.product\_selling\_price != p.product\_price">{{p.product\_price|currency:'INR'}}</s>

                    </div>

                    <span class="discount" \*ngIf="p.product\_discount">

                        Upto {{p.product\_discount/100|percent}} off

                    </span>

                    <span class="no-discount" \*ngIf="! p.product\_discount">

                        Upto {{p.product\_discount/100|percent}} off

                    </span>

                </mat-card-content>

                <mat-card-actions class="text-center">

                    <button mat-button color="primary" (click)="addToCart(p)" \*ngIf="!productInCart(p.id)">

                        <mat-icon>add\_shopping\_cart</mat-icon>

                        Add To Cart

                    </button>

                    <button mat-flat-button class="cart" (click)="gotoCart()" \*ngIf="productInCart(p.id)">

                        <mat-icon class="cart">check</mat-icon>

                        Go To Cart

                    </button>

                </mat-card-actions>

            </mat-card>

        </div>

        <div class="row row-cols-1 row-cols-md-6 ms-5 mt-1 mb-5">

            <mat-card \*ngFor="let p of rel\_products" class="ms-2 mt-1">

                <mat-card-header class="text-center">

                </mat-card-header>

                <mat-card-content class="text-center">

                    <img (click)="showDetails(p)" [src]="p.productImages[0].url" alt="" width="75px" height="75px">

                    <mat-divider></mat-divider>

                    <mat-card-subtitle class="mt-1" (click)="showDetails(p)">

                        {{p.product\_name|productTitle}}

                        <span \*ngIf="p.product\_name.length >= 35" [matTooltip]="p.product\_name">...</span>

                        <span class="empty-space" \*ngIf="p.product\_name.length <= 22">..........</span>

                    </mat-card-subtitle>

                    <div matLine>

                        Price: {{p.product\_selling\_price|currency:'INR'}}

                        <s class="ms-1"

                            \*ngIf="p.product\_selling\_price != p.product\_price">{{p.product\_price|currency:'INR'}}</s>

                    </div>

                    <span class="discount" \*ngIf="p.product\_discount">

                        Upto {{p.product\_discount/100|percent}} off

                    </span>

                    <span class="no-discount" \*ngIf="! p.product\_discount">

                        Upto {{p.product\_discount/100|percent}} off

                    </span>

                </mat-card-content>

                <mat-card-actions class="text-center">

                    <button mat-button color="primary" (click)="addToCart(p)" \*ngIf="!productInCart(p.id)">

                        <mat-icon>add\_shopping\_cart</mat-icon>

                        Add To Cart

                    </button>

                    <button mat-flat-button class="cart" (click)="gotoCart()" \*ngIf="productInCart(p.id)">

                        <mat-icon class="cart">check</mat-icon>

                        Go To Cart

                    </button>

                </mat-card-actions>

            </mat-card>

        </div>

    </div>

</div>

**File:-search-product.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

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

import { ActivatedRoute, NavigationEnd, Router } from '@angular/router';

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

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

import { UserService } from 'src/app/services/user.service';

import { ProductDetailsComponent } from '../product-details/product-details.component';

import { ViewCartComponent } from '../user/view-cart/view-cart.component';

@Component({

  selector: 'app-search-product',

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

  styleUrls: ['./search-product.component.css'],

})

export class SearchProductComponent implements OnInit {

  products: any = [];

  rel\_products: any = [];

  cart\_products: any = [];

  search\_model = '';

  cart\_id = 0;

  constructor(

    private productService: ProductService,

    private imageProcessingService: ImageProcessingService,

    private route: ActivatedRoute,

    private snack: MatSnackBar,

    private router: Router,

    private loginService: LoginService,

    private userService: UserService,

    private cartDialog: MatDialog

  ) {}

  ngOnInit(): void {

    this.search\_model = this.route.snapshot.params['model'];

    this.productService.searchProductsStartWith(this.search\_model).subscribe(

      (data) => {

        this.products = data;

        for (let p of this.products) {

          p = this.imageProcessingService.createProductImages(p);

        }

      },

      (error) => {}

    );

      this.productService.searchProducts(this.search\_model).subscribe(

        (data) => {

          this.rel\_products = data;

          for (let p of this.rel\_products) {

            p = this.imageProcessingService.createProductImages(p);

          }

        },

        (error) => {}

      );

    this.productService.searchStatus.asObservable().subscribe((data) => {

      this.productService.searchProductsStartWith(this.search\_model).subscribe(

        (data) => {

          this.products = data;

          for (let p of this.products) {

            p = this.imageProcessingService.createProductImages(p);

          }

        },

        (error) => {}

      );

      this.productService.searchProducts(this.search\_model).subscribe(

        (data) => {

          this.rel\_products = data;

          for (let p of this.rel\_products) {

            p = this.imageProcessingService.createProductImages(p);

          }

        },

        (error) => {}

      );

    });

    this.userService

      .getCart(this.loginService.getUser().id)

      .subscribe((data: any) => {

        this.cart\_products = data.products;

        console.log(this.cart\_products);

      });

    this.userService.cartStatus.asObservable().subscribe((data) => {

      this.userService

        .getCart(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.cart\_products = data.products;

          console.log(this.cart\_products);

        });

    });

    this.userService.getCart(this.loginService.getUser().id).subscribe(

      (data: any) => {

        this.cart\_id = data.id;

        console.log('Cart ID' + this.cart\_id);

      },

      (error) => console.log(error)

    );

    console.log(this.cart\_products);

  }

  showDetails(product: any) {

    this.cartDialog.open(ProductDetailsComponent, {

      height: '900px',

      width: '900px',

      data: product,

      position: {

        top: '50px',

      },

    });

  }

  public addToCart(product: any) {

    if (!this.loginService.isLoggedIn()) {

      this.snack.open('You must login to add product to cart', 'OK');

      return;

    }

    if (this.loginService.getUserRole() == 'NORMAL') {

      const user\_id = this.loginService.getUser().id;

      console.log('User ID' + user\_id);

      this.userService.getCart(user\_id).subscribe(

        (data: any) => {

          this.cart\_id = data.id;

          console.log('Cart ID' + this.cart\_id);

          this.userService.cartStatus.next(true);

        },

        (error) => console.log(error)

      );

      this.userService.addProductToCart(this.cart\_id, product).subscribe(

        (data) => {

          this.snack.open('Product added to cart', 'OK');

          this.userService.cartStatus.next(true);

          this.userService

            .getCart(this.loginService.getUser().id)

            .subscribe((data: any) => {

              this.cart\_products = data.products;

            });

        },

        (error) => {

          this.snack.open('Product not added to cart', 'OK');

        }

      );

    } else if (this.loginService.getUserRole() == 'ADMIN') {

      this.snack.open('This service is for normal user only', 'OK');

    }

  }

  public productInCart(product\_id: any) {

    let result = false;

    for (let p of this.cart\_products) {

      if (p.id == product\_id) {

        result = true;

      }

    }

    return result;

  }

  gotoCart() {

    this.cartDialog.open(ViewCartComponent, {

      height: '500px',

      width: '900px',

      position: {

        right: '0',

        top: '0',

      },

    });

  }

}

**Package:- pages. update-user**

**File:-update-user.component.html**

<div class="container mt-3">

    <div class="row">

        <div class="col-md-6 offset-md-3 text-center">

            <mat-card class="bg-light mb-3">

                <mat-card-title>

                    <div class="container text-center">

                        <img src="./../../../assets/logo.jpg" alt="medicare" width="75px" />

                    </div>

                    <h1 class="text-center">Update Profile Details</h1>

                </mat-card-title>

                <mat-card-content>

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

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Enter Your Username</mat-label>

                            <input matInput placeholder="Username " [(ngModel)]="user.username" name="username"

                                required />

                            <mat-hint>Username must be unique !!!!</mat-hint>

                        </mat-form-field>

                        <mat-form-field hidden appearance="outline" class="w-100">

                            <mat-label>Enter Your Password</mat-label>

                            <input hidden matInput placeholder="Password " [(ngModel)]="user.password" name="password"

                                required />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Enter Your First Name</mat-label>

                            <input matInput placeholder="First Name " [(ngModel)]="user.first\_name" name="first\_name"

                                required />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Enter Your Last Name</mat-label>

                            <input matInput placeholder="Last Name " [(ngModel)]="user.last\_name" name="last\_name"

                                required />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Enter Your Email</mat-label>

                            <input matInput placeholder="Email " [(ngModel)]="user.user\_email" name="user\_email"

                                required />

                        </mat-form-field>

                        <mat-form-field appearance="outline" class="w-100">

                            <mat-label>Enter Your Phone</mat-label>

                            <input matInput placeholder="Phone " [(ngModel)]="user.user\_phone" name="user\_phone"

                                required />

                        </mat-form-field>

                        <div class="container">

                            <button type="submit" mat-raised-button color="primary">Update</button>

                            <button type="reset" mat-raised-button color="warn" class="ms-3">Clear</button>

                        </div>

                    </form>

                </mat-card-content>

            </mat-card>

        </div>

    </div>

</div>

**File:-update-user.component.ts**

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

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-update-user',

  templateUrl: './update-user.component.html',

  styleUrls: ['./update-user.component.css'],

})

export class UpdateUserComponent implements OnInit {

  user = {

    username: '',

    password: '',

    first\_name: '',

    last\_name: '',

    user\_email: '',

    user\_phone: '',

  };

  constructor(private snack:MatSnackBar,

              private userService:UserService,

              private route:ActivatedRoute,

              private loginService:LoginService,

              private router:Router) {}

  ngOnInit(): void {

      this.userService

        .getUser(this.loginService.getUser().id)

        .subscribe((data: any) => {

          console.log(data);

          this.user = data;

        });

  }

  updateUser() {

    if (this.user.username == '' || this.user.username == null) {

      this.snack.open('Username is required', 'ok');

      return;

    }

    if (this.user.username.length > 13 || this.user.username.length < 5) {

      this.snack.open(

        'Username length should be less than 13 character or greater than 5 character',

        'ok'

      );

      return;

    }

    if (this.user.password == '' || this.user.password == null) {

      this.snack.open('Password is required', 'ok');

      return;

    }

    if (this.user.first\_name == '' || this.user.first\_name == null) {

      this.snack.open('First name is required', 'ok');

      return;

    }

    if (this.user.last\_name == '' || this.user.last\_name == null) {

      this.snack.open('Last name is required', 'ok');

      return;

    }

    if (this.user.user\_email == '' || this.user.user\_email == null) {

      this.snack.open('Email is required', 'ok');

      return;

    }

    if (this.user.user\_phone == '' || this.user.user\_phone == null) {

      this.snack.open('Phone is required', 'ok');

      return;

    }

    this.userService.updateUser(this.user).subscribe(

      (data) => {

        console.log(data);

        Swal.fire({

          position: 'top-end',

          icon: 'success',

          title: 'Updated successful !',

          showConfirmButton: false,

          timer: 2500,

        });

        if(this.loginService.isAdmin()){

          this.userService.userStatus.next(true);

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

        }

          if(this.loginService.isNormal()){

            this.userService.userStatus.next(true);

            this.router.navigate(['/user-dashboard/profile'])

          }

      },

      (error) => {

        console.log(error);

        Swal.fire({

          icon: 'error',

          title: 'Error !',

          text: 'Something went wrong!',

        });

      }

    );

  }

}

**Package:- pages.user.add-address**

**File: add-address.component.html**

<mat-card class="mt-2">

    <h1 matSubheader>Add New Address</h1>

    <mat-card-content>

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

            <div class="row">

                <div class="col-md-10 offset-md-1">

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Street</mat-label>

                        <textarea matInput placeholder="Street" [(ngModel)]="address.street" name="street" required

                            rows="4" maxlength="100">

                        </textarea>

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter City Name</mat-label>

                        <input matInput placeholder="City Name " [(ngModel)]="address.city" name="city\_name" required />

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Nearby Landmark</mat-label>

                        <input matInput placeholder="Landmark " [(ngModel)]="address.landmark" name="landmark" required

                            type="text" />

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Pincode</mat-label>

                        <input matInput placeholder="Pincode " [(ngModel)]="address.pincode" name="pincode"

                            required type="number" />

                    </mat-form-field>

                    <div class="container text-center mt-2">

                        <button type="submit" mat-raised-button color="primary">

                            Add Address

                        </button>

                    </div>

                </div>

            </div>

        </form>

    </mat-card-content>

</mat-card>

**File: add-address.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import { OrderComponent } from '../order/order.component';

@Component({

  selector: 'app-add-address',

  templateUrl: './add-address.component.html',

  styleUrls: ['./add-address.component.css']

})

export class AddAddressComponent implements OnInit {

  address={

    street:'',

    city:'',

    landmark:'',

    pincode:'',

    user:{

      id:0

    }

  }

  constructor(private loginService:LoginService,

              private userService:UserService,

              private snack:MatSnackBar,

              private router:Router,

              private cartDialog:MatDialog) { }

  ngOnInit(): void {

     this.address.user.id=this.loginService.getUser().id;

  }

  addAddress(){

    this.userService.addAddress(this.address).subscribe(

      (data)=>{

        this.snack.open('Address added successfully','OK');

        if(this.userService.orderStatus){

          this.userService.orderStatus=false;

           this.cartDialog.open(OrderComponent, {

             height: '600px',

             width: '1200px',

           });

        }else{

        this.router.navigate(['/user-dashboard/address']);

        }

      },

      (error)=>{

        this.snack.open('Error','OK')

      }

    )

  }

}

**Package:- pages.user.notification**

<span class="mt-3"></span>

<mat-list class="mt-5">

    <div mat-subheader>Notifications <button class="ms-2" mat-button color="primary">mark all as read</button></div>

     <mat-list-item class="mt-2" \*ngFor="let n of notifications">

           <mat-icon color="accent" mat-list-icon>sms</mat-icon>

           <div class="ms-2 mt-2" matCardSubtitle matTooltipPosition="left" [matTooltip]="n.msg">{{n.msg}}</div>

     </mat-list-item>

     <mat-divider></mat-divider>

</mat-list>

**File: notification.component.html**

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-notification',

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

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

})

export class NotificationComponent implements OnInit {

  notifications:any=null;

  constructor(private userService:UserService,

              private loginService:LoginService) { }

  ngOnInit(): void {

    this.userService.getNotifications(this.loginService.getUser().id).subscribe(

      (data)=>{

        this.notifications=data;

      }

    )

  }

}

**File: notification.component.ts**

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-notification',

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

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

})

export class NotificationComponent implements OnInit {

  notifications:any=null;

  constructor(private userService:UserService,

              private loginService:LoginService) { }

  ngOnInit(): void {

    this.userService.getNotifications(this.loginService.getUser().id).subscribe(

      (data)=>{

        this.notifications=data;

      }

    )

  }

}

**File: notification.component.css**

mat-list-item{

    border: 2.5px solid pink;

    border-radius: 10px;

    border-collapse: collapse;

    font-weight: bolder;

}

**Package:- pages.user.order**

**File: order.component.html**

<button mat-raised-button mat-dialog-close color="accent">

    <mat-icon>close</mat-icon>

</button>

<mat-stepper linear [labelPosition]="'bottom'">

    <mat-step [completed]="false"

              [stepControl]="firstFormGroup">

       <ng-template matStepLabel>Address</ng-template>

    <mat-card >

        <h1 matSubheader>Your Address List</h1>

        <mat-card-actions>

            <button mat-raised-button

                    color="primary"

                    (click)="addAddress()"

                    mat-dialog-close>

                <mat-icon color="accent">add\_circle</mat-icon> New Address

            </button>

        </mat-card-actions>

        <form [formGroup]="firstFormGroup">

        <mat-list \*ngIf="address.length">

            <mat-radio-group formControlName="firstCtrl" name="address" required >

                <mat-radio-button requied \*ngFor="let a of address" (change)="addressChanged(a.id)"   [value]="a.id">

                        <mat-list-item>

                            <h4 mat-line>{{a.street}} , </h4>

                            <h4 mat-line>{{a.city}} , </h4>

                            <h4 mat-line>Near {{a.landmark}} , </h4>

                            <h4 mat-line>{{a.pincode}} </h4>

                            <button mat-icon-button mat-dialog-close (click)="updateAddress(a.id)">

                                <mat-icon color="primary">edit</mat-icon>

                            </button>

                        </mat-list-item>

                </mat-radio-button>

            </mat-radio-group>

        </mat-list>

    </form>

        <h3 mat-line \*ngIf="! address.length">You don't have any address</h3>

    </mat-card>

       <div class="container text-center mt-3">

        <button mat-raised-button color="primary" matStepperNext>

            Payment Method <mat-icon>arrow\_forward</mat-icon>

        </button>

       </div>

    </mat-step>

    <mat-step  [completed]="false"

              [stepControl]="secondFormGroup"

              >

        <ng-template matStepLabel>Payment</ng-template>

        <mat-card >

            <h1 matSubheader>Choose Payment Method</h1>

            <form [formGroup]="secondFormGroup">

                <div class="text-center">

                <mat-list >

                    <mat-radio-group class="pay ms-2" formControlName="firstCtrl" name="payment" required>

                        <mat-radio-button requied (change)="paymentDone(1)" [value]="1">

                            <mat-list-item>

                              Cash On Delivery

                            </mat-list-item>

                        </mat-radio-button>

                        <mat-radio-button requied (change)="paymentDone(2)" [value]="2">

                            <mat-list-item>

                                Card Payment

                            </mat-list-item>

                        </mat-radio-button>

                        <mat-radio-button requied (change)="paymentDone(3)" [value]="3">

                            <mat-list-item>

                                UPI Payment

                            </mat-list-item>

                        </mat-radio-button>

                    </mat-radio-group>

                    <h1 mat-line>Dummy Payment Gateway</h1>

                </mat-list>

                </div>

            </form>

        </mat-card>

        <div class="container text-center mt-3">

            <button mat-raised-button color="primary"

                    (click)="placeOrder()"

                    matStepperNext>

                Continue <mat-icon >arrow\_forward</mat-icon>

            </button>

        </div>

    </mat-step>

    <mat-step class="text-center">

        <ng-template matStepLabel>Order Placed</ng-template>

        <div class="container text-center">

        <h1><img src="./../../../../assets/checkicon.gif" width="75px" alt="gif not supported" >

            Order Confirmed, Delivery expected in 72 Hours</h1>

            <h3>Total order amount: <img src="./../../../../assets/cash.gif" width="25px" alt="gif not supported">

                {{ total\_bill | currency:'INR'}}</h3>

            <button mat-raised-button mat-dialog-close color="primary" routerLink="/user-dashboard/orders">

                My Orders

            </button>

        </div>

    </mat-step>

</mat-stepper>

**File: order. component.ts**

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

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

import { MAT\_DIALOG\_DATA } from '@angular/material/dialog';

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

import { MatStepper } from '@angular/material/stepper';

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-order',

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

  styleUrls: ['./order.component.css'],

})

export class OrderComponent implements OnInit {

  order = {

    payment\_status: '',

    total:0,

    user: {

      id: 0,

    },

    address: {

      id: 0,

    },

  };

  cart\_id = 0;

  address: any = [];

  address\_id: any;

  payment\_id: any;

  payment\_status = '';

  payment\_status\_done = false;

  firstFormGroup = this.formBuilder.group({

    firstCtrl: ['', Validators.required],

  });

  secondFormGroup = this.formBuilder.group({

    firstCtrl: ['', Validators.required],

  });

  constructor(

    private userService: UserService,

    private loginService: LoginService,

    private snack: MatSnackBar,

    private router: Router,

    private formBuilder: FormBuilder,

    @Inject(MAT\_DIALOG\_DATA) public total\_bill: any,

  ) {}

  ngOnInit(): void {

    console.log(this.total\_bill);

    this.userService.getCart(this.loginService.getUser().id).subscribe(

      (data: any) => {

        console.log(data.id);

        this.cart\_id = data.id;

        console.log('cart\_id' + this.cart\_id);

      },

      (error) => {

        this.snack.open('Cart Not Getting');

      }

    );

    this.userService

      .getAddressesOfUser(this.loginService.getUser().id)

      .subscribe(

        (data) => {

          this.address = data;

        },

        (error) => {

          this.snack.open('Unable load addresses', 'OK');

        }

      );

  }

  deleteAddress(address\_id: any) {

    this.userService.deleteAddress(address\_id).subscribe(

      (data) => {

        this.snack.open('Address deleted successfully', 'OK');

        location.reload();

      },

      (error) => {

        this.snack.open('Address not deleted', 'OK');

      }

    );

  }

  addAddress() {

    this.userService.orderStatus = true;

    this.router.navigate(['/user-dashboard/add-address']);

  }

  updateAddress(address\_id: any) {

    this.userService.orderStatus = true;

    this.router.navigate(['/user-dashboard/update-address/' + address\_id]);

  }

  addressChanged(id: any) {

    this.address\_id = id;

  }

  paymentDone(id: number) {

    if (id == 1) {

      this.payment\_status = 'Cash on delivery';

    }

    if (id == 2) {

      this.payment\_status = 'Card Payment';

    }

    if (id == 3) {

      this.payment\_status = 'Upi Payment';

    }

  }

  placeOrder() {

    console.log(this.payment\_status);

    this.order.payment\_status = this.payment\_status;

    this.order.total= this.total\_bill;

    if (this.order.payment\_status == '') {

      this.snack.open('Payment Not Done Yet', 'OK');

      return;

    }

    if (this.address\_id == undefined) {

      this.snack.open('Address No Selected Yet', 'OK');

      return;

    }

    this.order.address.id = this.address\_id;

    this.order.user.id = this.loginService.getUser().id;

    this.userService.createOrder(this.order).subscribe(

      (data) => {

        this.userService.deleteProductsFromCart(this.cart\_id).subscribe(

          (data) => {

            this.userService.cartStatus.next(true);

          },

          (error) => {

            this.snack.open('Cart Not Cleared', 'OK');

          }

        );

      },

      (error) => {

        this.snack.open('Order not placed','ok');

      }

    );

  }

}

**File: order.component.css**

.mat-stepper-horizontal {

    margin-top: 8px;

}

.mat-form-field {

    margin-top: 16px;

}

.example-margin {

    margin: 0 12px;

}

**Package:- pages.user.update-address**

**File: update-address.component.html**

<mat-card class="mt-2">

    <h1 matSubheader>Update Address</h1>

    <mat-card-content>

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

            <div class="row">

                <div class="col-md-10 offset-md-1">

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Street</mat-label>

                        <textarea matInput placeholder="Street" [(ngModel)]="address.street" name="street" required

                            rows="4" maxlength="100">

                        </textarea>

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter City Name</mat-label>

                        <input matInput placeholder="City Name " [(ngModel)]="address.city" name="city\_name" required />

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Nearby Landmark</mat-label>

                        <input matInput placeholder="Landmark " [(ngModel)]="address.landmark" name="landmark" required

                            type="text" />

                    </mat-form-field>

                    <mat-form-field appearance="outline" class="w-100">

                        <mat-label>Enter Pincode</mat-label>

                        <input matInput placeholder="Pincode " [(ngModel)]="address.pincode" name="pincode" required

                            type="number" />

                    </mat-form-field>

                    <div class="container text-center mt-2">

                        <button type="submit" mat-raised-button color="primary">

                            Update Address

                        </button>

                    </div>

                </div>

            </div>

        </form>

    </mat-card-content>

</mat-card>

**File: update-address.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import { OrderComponent } from '../order/order.component';

@Component({

  selector: 'app-update-address',

  templateUrl: './update-address.component.html',

  styleUrls: ['./update-address.component.css'],

})

export class UpdateAddressComponent implements OnInit {

  address = {

    street: '',

    city: '',

    landmark: '',

    pincode: '',

    user:{

      id:0

    }

  }

  address\_id=0;

  constructor(private userService:UserService,

              private route:ActivatedRoute,

              private snack:MatSnackBar,

              private loginService:LoginService,

              private router:Router,

              private cartDialog:MatDialog) {}

  ngOnInit(): void {

    this.address\_id= this.route.snapshot.params['address\_id'];

    this.address.user.id=this.loginService.getUser().id;

    this.userService.getAddress(this.address\_id).subscribe(

      (data:any)=>{

        this.address=data;

        console.log("Address: "+this.address.user);

      },

      (error)=>{

        this.snack.open('Unable to load address','OK');

      }

    )

  }

  updateAddress(){

    console.log(this.address)

    this.userService.updateAddress(this.address).subscribe(

      (data)=>{

        this.snack.open('Successfully Updated Address','OK')

        if(this.userService.orderStatus){

            this.userService.orderStatus = false;

            this.cartDialog.open(OrderComponent, {

              height: '600px',

              width: '1200px',

            });

        }else{

          this.router.navigate(['/user-dashboard/address']);

        }

      },

      (error)=>{

        this.snack.open('Unable to update  address','OK')

      }

    )

  }

}

**Package:- pages.user.user-address**

**File:user-address.component.html**

<mat-card class="mt-2">

        <h1 matSubheader>Your Address List</h1>

        <mat-card-actions>

            <button mat-raised-button color="primary" routerLink="/user-dashboard/add-address">

                <mat-icon color="accent">add\_circle</mat-icon> New Address

            </button>

        </mat-card-actions>

        <mat-list>

            <mat-list-item \*ngFor="let a of address">

                <h4 mat-line>{{a.street}} , </h4>

                <h4 mat-line>{{a.city}} , </h4>

                <h4 mat-line>Near {{a.landmark}} , </h4>

                <h4 mat-line>{{a.pincode}}  </h4>

                <button mat-icon-button (click)="deleteAddress(a.id)">

                    <mat-icon color="warn">delete</mat-icon>

                </button>

                <button mat-icon-button [routerLink]="'/user-dashboard/update-address/'+a.id">

                    <mat-icon color="primary" >edit</mat-icon>

                </button>

            </mat-list-item>

        </mat-list>

</mat-card>

**File:user-address.component.ts**

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

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

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import Swal from 'sweetalert2';

@Component({

  selector: 'app-user-address',

  templateUrl: './user-address.component.html',

  styleUrls: ['./user-address.component.css']

})

export class UserAddressComponent implements OnInit {

  address:any=[]

  constructor(private userService:UserService,

              private loginService:LoginService,

              private snack:MatSnackBar) { }

  ngOnInit(): void {

    this.userService.getAddressesOfUser(this.loginService.getUser().id).subscribe(

      (data)=>{

        this.address= data;

      },

      (error)=>{

        this.snack.open('Unable load addresses','OK')

      }

    )

  }

  deleteAddress(address\_id:any){

Swal.fire({

  title: 'Are you sure?',

  text: 'You Want To Delete Address!',

  icon: 'question',

  showCancelButton: true,

  confirmButtonColor: '#3085d6',

  cancelButtonColor: '#d33',

  confirmButtonText: 'Yes, delete it!',

}).then((result) => {

  if (result.isConfirmed) {

    this.userService.deleteAddress(address\_id).subscribe(

      (data) => {

        location.reload();

        Swal.fire('Address deleted successfully!', '', 'success');

        this.userService.cartStatus.next(true);

      },

      (error) => {

        this.snack.open("You can't delete this address rigth now.Already order is pending on same address.", 'OK');

      }

    );

  }

});

  }

}

**Package:- pages.user.user-dashboard**

**File:user-dashboard.component.html**

<div class="container-fluid">

  <div class="row">

    <div class="col-md-3">

      <app-user-sidebar></app-user-sidebar>

    </div>

    <div class="col-md-9">

      <router-outlet></router-outlet>

    </div>

  </div>

</div>

**File:user-dashboard.component.ts**

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

@Component({

  selector: 'app-user-dashboard',

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

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

})

export class UserDashboardComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

**Package:- pages.user.user-orders**

**File:user-orders.component.html**

<div class="mt-2">

    <mat-card class="mt-1" \*ngFor="let o of orders">

    <button mat-button color="warn" \*ngIf="o.order\_status != 'COMPLETED'"> <mat-icon>hourglass\_top</mat-icon> Ongoing</button>

    <button mat-button color="basic" \*ngIf="o.order\_status == 'DELIVERED'">

        <mat-icon>download\_done</mat-icon> Deliverd

    </button>

    <button mat-button class="on-the-way-label" \*ngIf="o.order\_status == 'COMPLETED'">

        <mat-icon>local\_shipping</mat-icon> On The Way

    </button>

        <mat-card-header class="mt-1">

            <mat-card-title \*ngIf="o.delivery\_date">Delivery Expected : {{o.delivery\_date}}</mat-card-title>

            <mat-card-title \*ngIf="! o.delivery\_date">Order Not Shipped Yet</mat-card-title>

        </mat-card-header>

            <mat-accordion class="example-headers-align">

                <mat-expansion-panel>

                    <mat-expansion-panel-header>

                        <mat-panel-title class="float-end">

                            Details

                        </mat-panel-title>

                    </mat-expansion-panel-header>

                    <mat-card-subtitle>Order Placed On :{{o.order\_date}}</mat-card-subtitle>

                    <mat-card-subtitle>{{o.order\_status}}</mat-card-subtitle>

                    <mat-card-subtitle>Total Order Amount: <strong>{{o.total | currency:'INR'}}</strong> </mat-card-subtitle>

                    <mat-card-subtitle \*ngIf="o.payment\_status == 'Cash on delivery'">Payment : <img src="../../../../assets/cash.gif"

                            width="25px" alt=""> {{o.payment\_status}} </mat-card-subtitle>

                    <mat-card-subtitle \*ngIf="o.payment\_status != 'Cash on delivery'">

                        Payment : <img src="../../../../assets/checkicon.gif" width="25px" alt=""> Paid

                    </mat-card-subtitle>

                    <mat-card-content>

                        <mat-list>

                            <mat-list-item \*ngFor="let p of o.product\_list">

                                <img mat-list-avatar [src]="p.productImages[0].url" alt="">

                                <div matLine>{{p.product\_name}}</div>

                                <div matLine>Price Amount : {{p.product\_selling\_price | currency:'INR' }}</div>

                            </mat-list-item>

                        </mat-list>

                    </mat-card-content>

                </mat-expansion-panel>

            </mat-accordion>

    </mat-card>

</div>

**File:user-orders.component.ts**

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-user-orders',

  templateUrl: './user-orders.component.html',

  styleUrls: ['./user-orders.component.css']

})

export class UserOrdersComponent implements OnInit {

  orders:any=[]

  products:any=[]

  constructor(private userService:UserService,

              private loginService:LoginService,

              private imageService:ImageProcessingService) { }

  ngOnInit(): void {

    this.userService.getOrders(this.loginService.getUser().id).subscribe(

      (data:any)=>{ this.orders = data;

        for(let o of this.orders){

           this.userService.getOrderProducts(o.id).subscribe(

            (data)=>{

              this.products = data;

              for(let p of this.products){

                p = this.imageService.createProductImages(p);

              }

                o.product\_list = this.products;

            }

           )

        }

        console.log(this.orders);

      } );

  }

}

**File:user-orders.component.css**

.on-the-way-label{

    color: green;

}

**Package:- pages.user.user-slidebar**

**File:user-slidebar.component.html**

<mat-card class="mt-2 ms-2">

    <mat-action-list>

        <button mat-list-item routerLink="">

            <mat-icon mat-list-icon>home</mat-icon>Home

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item routerLink="/user-dashboard">

            <mat-icon mat-list-icon>account\_box</mat-icon>Profile

        </button>

       <mat-divider></mat-divider>

        <button mat-list-item routerLink="/user-dashboard/address">

            <mat-icon mat-list-icon>business</mat-icon>Address

        </button>

        <mat-divider></mat-divider>

        <button mat-list-item routerLink="/user-dashboard/orders">

            <mat-icon mat-list-icon>send\_time</mat-icon>Orders

        </button>

        <mat-divider></mat-divider>

         <mat-divider></mat-divider>

        <button mat-list-item (click)="logout()" style="color: #f44337;">

            <mat-icon mat-list-icon color="warn">logout</mat-icon> Logout

        </button>

    </mat-action-list>

</mat-card>

**File:user-slidebar.component.ts**

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

import { LoginService } from 'src/app/services/login.service';

@Component({

  selector: 'app-user-sidebar',

  templateUrl: './user-sidebar.component.html',

  styleUrls: ['./user-sidebar.component.css'],

})

export class UserSidebarComponent implements OnInit {

  constructor(private loginService: LoginService) {}

  ngOnInit(): void {}

  logout() {

    this.loginService.logout();

    window.location.reload();

  }

}

**Package:- pages.user.view-cart**

**File:view-cart.component.html**

<button mat-raised-button mat-dialog-close color="accent">

    <mat-icon>close</mat-icon>

</button>

<mat-card-title class="mt-1 text-center">Products In Cart</mat-card-title>

<section  class="example-container mat-elevation-z8" tabindex="0">

<table  mat-table [dataSource]="products\_in\_cart" >

    <ng-container matColumnDef="image">

        <th mat-header-cell \*matHeaderCellDef>

            <mat-icon>image</mat-icon>

        </th>

        <td mat-cell \*matCellDef="let element">

            <img [src]="element.productImages[0].url" alt="" width="75px" height="75px">

        </td>

        <td mat-footer-cell \*matFooterCellDef></td>

    </ng-container>

    <ng-container matColumnDef="name">

        <th mat-header-cell \*matHeaderCellDef>

            Product Name

        </th>

        <td mat-cell \*matCellDef="let element">{{element.product\_name}}</td>

        <td mat-footer-cell \*matFooterCellDef>Total:</td>

    </ng-container>

    <ng-container matColumnDef="price">

        <th mat-header-cell \*matHeaderCellDef>

            Price

        </th>

        <td mat-cell \*matCellDef="let element">{{element.product\_price|currency:'INR'}}</td>

        <td mat-footer-cell \*matFooterCellDef></td>

    </ng-container>

    <ng-container matColumnDef="discount">

        <th mat-header-cell \*matHeaderCellDef>

            Discount

        </th>

        <td mat-cell \*matCellDef="let element">{{element.product\_discount/100|percent}}</td>

        <td mat-footer-cell \*matFooterCellDef></td>

    </ng-container>

    <ng-container matColumnDef="payable">

        <th mat-header-cell \*matHeaderCellDef>

            Payable

        </th>

        <td mat-cell \*matCellDef="let element">{{element.product\_selling\_price|currency:'INR'}}</td>

        <td mat-footer-cell \*matFooterCellDef>  </td>

    </ng-container>

    <ng-container matColumnDef="quantity">

        <th mat-header-cell \*matHeaderCellDef>

            Quantity

        </th>

        <td mat-cell \*matCellDef="let element">1

        </td>

        <td mat-footer-cell \*matFooterCellDef></td>

    </ng-container>

    <ng-container matColumnDef="action">

        <th mat-header-cell \*matHeaderCellDef>

        </th>

        <td mat-cell \*matCellDef="let element">

            <button class="ml-2" mat-icon-button (click)="delete(element.id)">

                <mat-icon color="accent">delete</mat-icon>

            </button>

        </td>

        <td mat-footer-cell \*matFooterCellDef>{{getTotalCost() | currency:'INR'}}</td>

    </ng-container>

    <tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

    <tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

    <tr  mat-footer-row \*matFooterRowDef="displayedColumns; sticky: true"></tr>

</table>

</section>

<div \*ngIf="products\_in\_cart" class="container text-center" mat-dialog-close>

<button class="text-center mt-3 mb-1" mat-raised-button

                                      mat-dialog-close

                                      color="primary"

                                      (click)="openOrder()">

    <mat-icon>shopping\_bag</mat-icon>

    Place Order Now

</button>

</div>

**File:view-cart.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

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

import { ImageProcessingService } from 'src/app/services/image-processing.service';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

import { OrderComponent } from '../order/order.component';

@Component({

  selector: 'app-view-cart',

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

  styleUrls: ['./view-cart.component.css'],

})

export class ViewCartComponent implements OnInit {

  displayedColumns: string[] = [

    'image',

    'name',

    'price',

    'discount',

    'payable',

    'quantity',

    'action',

  ];

  resultColumns: string[] = ['price', 'saved', 'payable', 'action'];

  products\_in\_cart: any = [];

  quantity = 1;

  cart\_id = 0;

  constructor(

    private userService: UserService,

    private loginService: LoginService,

    private snack: MatSnackBar,

    private imageProcessingService: ImageProcessingService,

    private cartDialog: MatDialog

  ) {}

  ngOnInit(): void {

    this.userService

      .getCart(this.loginService.getUser().id)

      .subscribe((data: any) => {

        this.products\_in\_cart = data.products;

        this.cart\_id = data.id;

        for (let p of this.products\_in\_cart) {

          p = this.imageProcessingService.createProductImages(p);

        }

        console.log(this.products\_in\_cart);

      });

    this.userService.cartStatus.asObservable().subscribe((event) => {

      this.userService

        .getCart(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.products\_in\_cart = data.products;

          this.cart\_id = data.id;

          for (let p of this.products\_in\_cart) {

            p = this.imageProcessingService.createProductImages(p);

          }

          console.log(this.products\_in\_cart);

        });

    });

  }

  addQuantity() {

    this.quantity++;

  }

  subtractQuantity() {

    this.quantity--;

  }

  delete(product\_id: any) {

    this.userService.deleteProductFromCart(this.cart\_id, product\_id).subscribe(

      (data: any) => {

        this.userService.cartStatus.next(true);

      },

      (error) => {

        alert('Item Not Removed');

      }

    );

  }

  openOrder() {

    console.log(this.getTotalCost());

    if(this.products\_in\_cart.length>0){

    this.cartDialog.open(OrderComponent, {

      data:this.getTotalCost(),

      height: '600px',

      width: '1200px',

    });

  }else{

    this.snack.open('No products in cart','OK');

  }

  }

  getTotalCost(){

    let total=0;

    for(let p of this.products\_in\_cart){

      total= total + p.product\_selling\_price;

    }

    return total;

  }

}

**File:view-cart.component.css**

table {

    width: 100%;

    height: 100%;

}

.quantity-btn-add{

    font-size:xx-large;

    color: green;

    cursor: pointer;

}

.quantity-btn-minus {

    font-size: xx-large;

    letter-spacing: -3px;

    color: red;

    cursor: pointer;

}

.quantity\_number{

    font-size: x-large;

    text-align: center;

}

.mat-stepper-horizontal {

    margin-top: 8px;

}

.mat-form-field {

    margin-top: 16px;

}

.example-container {

    height: 270px;

    overflow: auto;

}

tr.mat-footer-row {

    font-weight: bold;

}

.mat-table-sticky {

    border-top: 1px solid #e0e0e0;

}

**Package:- pages.user.view-user-address**

**File:view-user-address.component.html**

<p>view-user-address works!</p>

**Package:- material**

**File:-material.module.ts**

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

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

import { MatInputModule } from '@angular/material/input';

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatButtonModule } from '@angular/material/button';

import { MatCardModule } from '@angular/material/card';

import { MatIconModule } from '@angular/material/icon';

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

import { MatToolbarModule } from '@angular/material/toolbar';

import { MatListModule } from '@angular/material/list';

import { MatTooltipModule } from '@angular/material/tooltip';

import { MatRippleModule } from '@angular/material/core';

import { MatSelectModule } from '@angular/material/select';

import { MatGridListModule } from '@angular/material/grid-list';

import { MatDialogModule } from '@angular/material/dialog';

import { MatTableModule } from '@angular/material/table';

import { MatBadgeModule } from '@angular/material/badge';

import { MatStepperModule } from '@angular/material/stepper';

import { MatCheckboxModule } from '@angular/material/checkbox';

import { MatRadioModule } from '@angular/material/radio';

import { MatSlideToggleModule } from '@angular/material/slide-toggle';

import { MatAccordion, MatExpansionModule, MAT\_ACCORDION } from '@angular/material/expansion';

import { MatChipsModule } from '@angular/material/chips';

import { MatPaginatorModule } from '@angular/material/paginator';

@NgModule({

  declarations: [],

  exports: [

    CommonModule,

    MatInputModule,

    MatFormFieldModule,

    MatButtonModule,

    MatCardModule,

    MatIconModule,

    MatSnackBarModule,

    MatToolbarModule,

    MatListModule,

    MatTooltipModule,

    MatRippleModule,

    MatSelectModule,

    MatGridListModule,

    MatDialogModule,

    MatTableModule,

    MatBadgeModule,

    MatStepperModule,

    MatCheckboxModule,

    MatRadioModule,

    MatSlideToggleModule,

    MatExpansionModule,

    MatChipsModule,

    MatPaginatorModule

  ]

})

export class MaterialModule { }

**Package:- components.navbar**

**File:- navbar.component.html**

<mat-toolbar color="primary"

             class="m-auto sticky-top">

  <span class="app-title-color" routerLink="" mat-card-title >

    <img src="./../../../assets/logo.jpg" alt=""

        width="30px"

        style="background-color: white ;"

        routerLink="">

     MEDICARE

  </span>

  <span class="nav-spacer"></span>

      <input [(ngModel)]="search"

             type="search" placeholder="Search" aria-label="Search"

             (keyup)="searchProducts()"

             (change)="searchProducts()"

             \*ngIf="!isAdmin">

      <button \*ngIf="!isAdmin" mat-raised-button class="search-btn"><mat-icon color="primary">search</mat-icon></button>

  <span class="nav-spacer"></span>

<button mat-button [disabled]="cartOpen" \*ngIf="isNormal && products\_in\_cart" (click)="showCart()" >

  Cart <mat-icon [matBadge]="products\_in\_cart"  class="demo-section" matBadgeColor="accent">shopping\_cart</mat-icon>

</button>

<button [disabled]="msgOpen" color="basic" \*ngIf="isNormal notifications" mat-button (click)="openNotifications()">

  <mat-icon \*ngIf="notifications" [matBadge]="notifications.length" matBadgeColor="accent" matBadgeOverlap="true">notifications</mat-icon>

</button>

<button color="basic" \*ngIf="isAdmin"

          mat-button

          (click)="openDashboard()">

    <mat-icon matTooltip="Go To Dashboard">admin\_panel\_settings</mat-icon>

    Dashboard

</button>

<button color="basic"  \*ngIf="isNormal" mat-button (click)="openDashboard()">

  <mat-icon>account\_circle</mat-icon>

  {{user?.first\_name}} {{user?.last\_name}}

</button>

  <a \*ngIf="!isLoggedIn"

     mat-icon-button

     routerLink="login">

    <mat-icon>login</mat-icon>

    Login

</a>

   <a    \*ngIf="!isLoggedIn"

          mat-icon-button

          class="me-5 ms-5"

          routerLink="register">

    <mat-icon> account\_circle </mat-icon>

    Register

</a>

</mat-toolbar>

**File:- navbar.component.ts**

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

import { MatDialog } from '@angular/material/dialog';

import { NavigationEnd, NavigationStart, Router } from '@angular/router';

import { NotificationComponent } from 'src/app/pages/user/notification/notification.component';

import { ViewCartComponent } from 'src/app/pages/user/view-cart/view-cart.component';

import { LoginService } from 'src/app/services/login.service';

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

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-navbar',

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

  styleUrls: ['./navbar.component.css'],

})

export class NavbarComponent implements OnInit {

  user: any = {

  };

  notifications: any = null;

  cartOpen = false;

  msgOpen = false;

  search = '';

  isNormal=false;

  isAdmin=false;

  isLoggedIn=false;

  products\_in\_cart = 0;

  constructor(

    public loginService: LoginService,

    private router: Router,

    private userService: UserService,

    private cartDialog: MatDialog,

    private productService:ProductService

  ) {}

  ngOnInit(): void {

  this.loadUser()

  this.loginService.loginStatusSubject.asObservable().subscribe((data) => {

      this.userService

        .getUser(this.loginService.getUser().id)

        .subscribe((data: any) => {

          console.log(data);

          this.user = data;

          this.isAdmin=this.loginService.isAdmin();

          this.isNormal=this.loginService.isNormal();

          this.isLoggedIn = this.loginService.isLoggedIn();

          console.log(this.isAdmin)

          console.log(this.isLoggedIn);

          console.log(this.isNormal);

        });

      this.userService

        .getNotifications(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.notifications = data;

        });

           this.userService.getCart(this.user.id).subscribe((data: any) => {

             this.products\_in\_cart = data.products.length;

             console.log(this.products\_in\_cart);

           });

    });

    this.userService.userStatus.asObservable().subscribe((data) => {

      this.userService

        .getUser(this.loginService.getUser().id)

        .subscribe((data: any) => {

          console.log(data);

          this.user = data;

          this.isAdmin = this.loginService.isAdmin();

          this.isNormal = this.loginService.isNormal();

        });

      this.userService

        .getNotifications(this.loginService.getUser().id)

        .subscribe((data: any) => {

          this.notifications = data;

        });

    });

    this.userService.cartStatus.asObservable().subscribe((data: any) => {

      this.userService.getCart(this.user.id).subscribe((data: any) => {

        this.products\_in\_cart = data.products.length;

        console.log(this.products\_in\_cart);

        this.isAdmin = this.loginService.isAdmin();

        this.isNormal = this.loginService.isNormal();

      });

    });

       this.isLoggedIn=this.loginService.isLoggedIn();

       this.isAdmin = this.loginService.isAdmin();

       this.isNormal = this.loginService.isNormal();

      this.userService

        .getNotifications(this.loginService?.getUser()?.id)

        .subscribe((data: any) => {

          this.notifications = data;

        });

        this.userService

          .getUser(this.loginService?.getUser()?.id)

          .subscribe((data: any) => {

            this.isAdmin = this.loginService.isAdmin();

            this.isNormal = this.loginService.isNormal();

          });

          if(this.user.id>0){

          this.userService.getCart(this.user.id).subscribe((data: any) => {

            this.products\_in\_cart = data.products.length;

            console.log(this.products\_in\_cart);

            this.isAdmin = this.loginService.isAdmin();

            this.isNormal = this.loginService.isNormal();

          },

          (error)=>{

          });

        }

         if(this.user!=null){

         this.userService.getCart(this.user?.id).subscribe((data: any) => {

           this.products\_in\_cart = data.products.length;

           console.log(this.products\_in\_cart);

         });

         }

  }

  loadUser(){

    this.userService.getUser(this.loginService?.getUser()?.id).subscribe(

      (data:any)=>{

        this.user=data;

          this.userService.getCart(this.user?.id).subscribe((data: any) => {

            this.products\_in\_cart = data.products.length;

            console.log(this.products\_in\_cart);

          });

      }

    )

  }

  logout() {

    this.loginService.logout();

    window.location.reload();

  }

  openDashboard() {

    if (this.loginService.getUserRole() == 'ADMIN') {

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

    }

    if (this.loginService.getUserRole() == 'NORMAL') {

      this.router.navigate(['/user-dashboard']);

    }

  }

  public showCart() {

    this.cartOpen = true;

    const dialogREf = this.cartDialog.open(ViewCartComponent, {

      height: '450px',

      width: '1000px',

      position: {

        right: '0',

        top: '0',

      },

    });

    dialogREf.afterClosed().subscribe((data) => {

      this.cartOpen = false;

    });

  }

  openNotifications() {

    this.msgOpen = true;

    const ref = this.cartDialog.open(NotificationComponent, {

      height: '450px',

      width: '450px',

      position: {

        right: '0',

        top: '0',

      },

    });

    ref.afterClosed().subscribe((data) => {

      this.msgOpen = false;

    });

  }

  searchProducts(){

    if(this.search.length == 0){

      this.router.navigate(['/home/0']);

    }else{

      this.router.navigate(['/search/'+this.search]);

    }

  }

}

**File:- navbar.component.css**

.nav-spacer {

    flex: 1 1 auto;

}

.app-title-color {

    color: #00C853;

    font-size: x-large;

    font-weight: bold;

    cursor: pointer;

    font-display: inherit;

    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;

}

.navbar{

    background: #3F51B5;

}

.dashboard{

    color: #00C853;

}

.search-box{

    background-color: white;

}

input{

   border: 0.5px solid #3F51B5;

   flex: 1;

   border-right: 0px;

   font-family: Roboto, "Helvetica Neue",sans-serif;

   font-size: small;

   border-radius: 10px;

   border-top-right-radius: 0px;

   border-bottom-right-radius: 0px;

   height: 40px;

   outline: none;

   width: 100%;

}

input:focus{

    border: 0.5px solid #3F51B5;

    flex: 1;

    border-right: 0px solid white;

    font-family: Roboto, "Helvetica Neue", sans-serif;

    font-size: small;

    border-radius: 10px;

    border-top-right-radius: 0px;

    border-bottom-right-radius: 0px;

    height: 40px;

    outline: none;

}

.search-btn{

    height: 40px;

    border-radius: 10px;

    border-top-left-radius: 0px;

    border-bottom-left-radius: 0px;

    color: white;

    border: 0.5px solid #3F51B5;

    border-left: 0px solid white;

}

.fixed{

    position: fixed;

    width: 100%;

    transition: 0.2s linear;

    top: 0px;

    left: 0px;

}

 .navbar-nav-scroll{

    --bs-scroll-height: 100px;

 }