|  |
| --- |
| **SimpliLearn Capstone Project: MyMoviePlan**  **Movie ticket booking portal**  **Submitted by: KAVIN K R** |
| Date of submission : 21-03-2022  GitHub Project Repository URL : [GitHub Link](https://github.com/kavink-r/SimpliLearn-Capstone-Project-Medicare.git) |
| **Project Structure:**  **Backend API:** |
| package com.medicare.api;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class MedicareApplication {  public static void main(String[] args) {  SpringApplication.run(MedicareApplication.class, args);  }  } |
| package com.medicare.api.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import com.medicare.api.entity.AdminCredential;  import com.medicare.api.entity.Cart;  import com.medicare.api.entity.Customer;  import com.medicare.api.entity.CustomerCredential;  import com.medicare.api.entity.OrderResponse;  import com.medicare.api.entity.ProductList;  import com.medicare.api.service.AdminCredentialService;  import com.medicare.api.service.CartService;  import com.medicare.api.service.CustomerCredentialService;  import com.medicare.api.service.CustomerService;  import com.medicare.api.service.ProductListService;  @CrossOrigin(origins = "http://localhost:4200")  @RestController  @RequestMapping("/api/medicare")  public class MedicareController {    @Autowired  private CustomerService customerService;    @Autowired  private ProductListService productListService;    @Autowired  private CartService cartService;    @Autowired  private CustomerCredentialService customerCredentialService;    @Autowired  private AdminCredentialService adminCredentialService;  // Customer & Product    // 1. POST Method  @PostMapping("/customer")  public Customer addCustomer(@RequestBody Customer customer) {  return customerService.saveCustomer(customer);  }    // 2. GET Method  @GetMapping("/customer")  public List<Customer> findAllCustomers() {  return customerService.getCustomers();  }    @GetMapping("/customer/{id}")  public Customer findCustomerById(@PathVariable int id) {  return customerService.getCustomerById(id);  }    // 3. DELETE Method  @DeleteMapping("/customer/{id}")  public String deleteCustomer(@PathVariable int id) {  return customerService.deleteCustomerById(id);  }    // Join the Customer & Product Table    @GetMapping("/order/list")  public List<OrderResponse> findAllCustomerProductInfos() {  return customerService.getCustomerProductJoinInfos();  }    // ProductList    // 1. POST Method  @PostMapping("/product/list")  public ProductList addProductList(@RequestBody ProductList productlist) {  return productListService.saveProductList(productlist);  }    // 2. GET Method  @GetMapping("/product/list")  public List<ProductList> findAllProductLists() {  return productListService.getProductLists();  }  @GetMapping("/product/list/{id}")  public ProductList findProductListById(@PathVariable int id) {  return productListService.getProductListById(id);  }    // 3. DELETE Method  @DeleteMapping("/product/list/{id}")  public String deleteProductList(@PathVariable int id) {  return productListService.deleteProductListById(id);  }    // 4. PUT Method  @PutMapping("/product/list/{id}")  public ProductList updateProductList(@RequestBody ProductList productlist, @PathVariable int id) {  return productListService.updateProductList(productlist, id);  }    // Cart    // 1. POST Method  @PostMapping("/cart")  public Cart addCart(@RequestBody Cart cart) {  return cartService.saveCart(cart);  }    // 2. GET Method  @GetMapping("/cart")  public List<Cart> findAllCarts() {  return cartService.getCarts();  }    @GetMapping("/cart/{id}")  public Cart findCartById(@PathVariable int id) {  return cartService.getCartById(id);  }    // 3. DELETE Method  @DeleteMapping("/cart/{id}")  public String deleteCart(@PathVariable int id) {  return cartService.deleteCartById(id);  }    // CustomerCredential    // 1. POST Method    @PostMapping("/customer/credential")  public CustomerCredential addCustomerCredential(@RequestBody CustomerCredential customercredential) {  return customerCredentialService.saveCustomerCredential(customercredential);  }    // 2. GET Method    @GetMapping("/customer/credential")  public List<CustomerCredential> findAllCustomerCredentials() {  return customerCredentialService.getCustomerCredentials();  }    @GetMapping("/customer/credential/{id}")  public CustomerCredential findCustomerCredentialById(@PathVariable int id) {  return customerCredentialService.getCustomerCredentialById(id);  }    // 3. PUT Method  @PutMapping("/customer/credential/{id}")  public CustomerCredential updateCustomerCredential(@RequestBody CustomerCredential customercredential, @PathVariable int id) {  return customerCredentialService.updateCustomerCredential(customercredential, id);  }    // AdminCredential    // 1. POST Method    @PostMapping("/admin/credential")  public AdminCredential addAdminCredential(@RequestBody AdminCredential admincredential) {  return adminCredentialService.saveAdminCredential(admincredential);  }    // 2. GET Method    @GetMapping("/admin/credential")  public List<AdminCredential> findAllAdminCredentials() {  return adminCredentialService.getAdminCredentials();  }    @GetMapping("/admin/credential/{id}")  public AdminCredential findAdminCredentialById(@PathVariable int id) {  return adminCredentialService.getAdminCredentialById(id);  }    // 3. PUT Method  @PutMapping("/admin/credential/{id}")  public AdminCredential updateAdminCredential(@RequestBody AdminCredential admincredential, @PathVariable int id) {  return adminCredentialService.updateAdminCredential(admincredential, id);  }    } |
| package com.medicare.api.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  @Entity  public class AdminCredential {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int adminId;  private String username;  private String password;      public AdminCredential() {}  public AdminCredential(String username, String password) {  this.username = username;  this.password = password;  }  public int getAdminId() {  return adminId;  }  public void setAdminId(int adminId) {  this.adminId = adminId;  }  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;  }  @Override  public String toString() {  return "AdminCredential [adminID=" + adminId + ", username=" + username + ", password=" + password + "]";  }  } |
| package com.medicare.api.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  @Entity  public class Cart {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int cartId;  private String medicineName;  private String seller;  private double price;  private String description;  private String imgURL;  private int quantity;    public Cart() {}  public Cart(String medicineName, String seller, double price, String description, String imgURL, int quantity) {  this.medicineName = medicineName;  this.seller = seller;  this.price = price;  this.description = description;  this.imgURL = imgURL;  this.quantity = quantity;  }  public int getCartId() {  return cartId;  }  public void setCartId(int cartId) {  this.cartId = cartId;  }  public String getMedicineName() {  return medicineName;  }  public void setMedicineName(String medicineName) {  this.medicineName = medicineName;  }  public String getSeller() {  return seller;  }  public void setSeller(String seller) {  this.seller = seller;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public String getImgURL() {  return imgURL;  }  public void setImgURL(String imgURL) {  this.imgURL = imgURL;  }  public int getQuantity() {  return quantity;  }  public void setQuantity(int quantity) {  this.quantity = quantity;  }  @Override  public String toString() {  return "Cart [cartId=" + cartId + ", medicineName=" + medicineName + ", seller=" + seller + ", price=" + price  + ", description=" + description + ", imgURL=" + imgURL + ", quantity=" + quantity + "]";  }      } |
| package com.medicare.api.entity;  import java.util.List;  import javax.persistence.CascadeType;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.JoinTable;  import javax.persistence.ManyToMany;  @Entity  public class Customer {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int CusId;  private String name;  private int age;  private String gender;  private String email;  private String address;  private String mobile;    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)  @JoinTable(name = "CUSTOMER\_PRODUCT\_TABLE",  joinColumns = {  @JoinColumn(name ="customer\_id", referencedColumnName = "CusId")  },  inverseJoinColumns = {  @JoinColumn(name = "product\_id", referencedColumnName = "pid")  }  )  private List<Product> products;  public Customer() {}  public Customer(String name, int age, String gender, String email, String address, String mobile,  List<Product> products) {  this.name = name;  this.age = age;  this.gender = gender;  this.email = email;  this.address = address;  this.mobile = mobile;  this.products = products;  }  public int getCusId() {  return CusId;  }  public void setCusId(int cusId) {  CusId = cusId;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String getGender() {  return gender;  }  public void setGender(String gender) {  this.gender = gender;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public String getMobile() {  return mobile;  }  public void setMobile(String mobile) {  this.mobile = mobile;  }  public List<Product> getProducts() {  return products;  }  public void setProducts(List<Product> products) {  this.products = products;  }  @Override  public String toString() {  return "Customer [CusId=" + CusId + ", name=" + name + ", age=" + age + ", gender=" + gender + ", email="  + email + ", address=" + address + ", mobile=" + mobile + ", products=" + products + "]";  }            } |
| package com.medicare.api.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  @Entity  public class CustomerCredential {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int CustomerId;  private String username;  private String password;      public CustomerCredential() {}  public CustomerCredential(String username, String password) {  this.username = username;  this.password = password;  }  public int getCustomerId() {  return CustomerId;  }  public void setCustomerId(int customerId) {  CustomerId = customerId;  }  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;  }  @Override  public String toString() {  return "CustomerCredential [CustomerId=" + CustomerId + ", username=" + username + ", password=" + password  + "]";  }      } |
| package com.medicare.api.entity;  import java.util.Date;  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  import com.fasterxml.jackson.annotation.JsonInclude;  import com.fasterxml.jackson.annotation.JsonInclude.Include;  @JsonIgnoreProperties(ignoreUnknown = true)  @JsonInclude(value = Include.NON\_DEFAULT)  public class OrderResponse {    private int CusId;  private String name;  private String email;  private String mobile;  private String address;  private String medicineName;  private String seller;  private double price;  private String description;  private int quantity;  private Date orderDateTime;      public OrderResponse() {}  public OrderResponse(int CusId, String name, String email, String mobile, String address, String medicineName,  String seller, double price, String description, int quantity, Date orderDateTime) {  this.CusId = CusId;  this.name = name;  this.email = email;  this.mobile = mobile;  this.address = address;  this.medicineName = medicineName;  this.seller = seller;  this.price = price;  this.description = description;  this.quantity = quantity;  this.orderDateTime = orderDateTime;  }  public int getCusId() {  return CusId;  }  public void setCusId(int CusId) {  this.CusId = CusId;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getMobile() {  return mobile;  }  public void setMobile(String mobile) {  this.mobile = mobile;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public String getMedicineName() {  return medicineName;  }  public void setMedicineName(String medicineName) {  this.medicineName = medicineName;  }  public String getSeller() {  return seller;  }  public void setSeller(String seller) {  this.seller = seller;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public int getQuantity() {  return quantity;  }  public void setQuantity(int quantity) {  this.quantity = quantity;  }  public Date getOrderDateTime() {  return orderDateTime;  }  public void setOrderDateTime(Date orderDateTime) {  this.orderDateTime = orderDateTime;  }      } |
| package com.medicare.api.entity;  import java.util.Date;  import java.util.List;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.FetchType;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.ManyToMany;  import javax.persistence.PrePersist;  import javax.persistence.Temporal;  import javax.persistence.TemporalType;  import com.fasterxml.jackson.annotation.JsonBackReference;  @Entity  public class Product {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int pid;  private String medicineName;  private String seller;  private double price;  private String description;  private int quantity;    @ManyToMany(mappedBy = "products", fetch = FetchType.LAZY)  @JsonBackReference  private List<Customer> customers;    @Temporal(TemporalType.TIMESTAMP)  @Column(nullable = false)  private Date orderDateTime;    @PrePersist  private void onCreate() {  orderDateTime = new Date();  }  public Product() {}  public Product(String medicineName, String seller, double price, String description, int quantity,  List<Customer> customers, Date orderDateTime) {  this.medicineName = medicineName;  this.seller = seller;  this.price = price;  this.description = description;  this.quantity = quantity;  this.customers = customers;  this.orderDateTime = orderDateTime;  }  public int getPid() {  return pid;  }  public void setPid(int pid) {  this.pid = pid;  }  public String getMedicineName() {  return medicineName;  }  public void setMedicineName(String medicineName) {  this.medicineName = medicineName;  }  public String getSeller() {  return seller;  }  public void setSeller(String seller) {  this.seller = seller;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public int getQuantity() {  return quantity;  }  public void setQuantity(int quantity) {  this.quantity = quantity;  }  public List<Customer> getCustomers() {  return customers;  }  public void setCustomers(List<Customer> customers) {  this.customers = customers;  }  public Date getOrderDateTime() {  return orderDateTime;  }  public void setOrderDateTime(Date orderDateTime) {  this.orderDateTime = orderDateTime;  }  @Override  public String toString() {  return "Product [pid=" + pid + ", medicineName=" + medicineName + ", seller=" + seller + ", price=" + price  + ", description=" + description + ", quantity=" + quantity + ", customers=" + customers  + ", orderDateTime=" + orderDateTime + "]";  }        } |
| package com.medicare.api.entity;  import java.util.Date;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.PrePersist;  import javax.persistence.Temporal;  import javax.persistence.TemporalType;  @Entity  public class ProductList {    @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int productListId;  private String medicineName;  private String seller;  private String qtyDesc;  private double price;  private String description;  private String imgURL;  private int active=1;    @Temporal(TemporalType.TIMESTAMP)  @Column(nullable = false)  private Date CreationDateTime;    @PrePersist  private void onCreate() {  CreationDateTime = new Date();  }  public ProductList() {}  public ProductList(String medicineName, String seller, String qtyDesc, double price, String description, String imgURL,  int active, Date creationDateTime) {  this.medicineName = medicineName;  this.seller = seller;  this.qtyDesc = qtyDesc;  this.price = price;  this.description = description;  this.imgURL = imgURL;  this.active = active;  this.CreationDateTime = creationDateTime;  }  public int getProductListId() {  return productListId;  }  public void setProductListId(int productListId) {  this.productListId = productListId;  }  public String getMedicineName() {  return medicineName;  }  public void setMedicineName(String medicineName) {  this.medicineName = medicineName;  }  public String getSeller() {  return seller;  }  public void setSeller(String seller) {  this.seller = seller;  }  public String getQtyDesc() {  return qtyDesc;  }  public void setQtyDesc(String qtyDesc) {  this.qtyDesc = qtyDesc;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public String getImgURL() {  return imgURL;  }  public void setImgURL(String imgURL) {  this.imgURL = imgURL;  }  public int getActive() {  return active;  }  public void setActive(int active) {  this.active = active;  }  public Date getCreationDateTime() {  return CreationDateTime;  }  public void setCreationDateTime(Date creationDateTime) {  this.CreationDateTime = creationDateTime;  }  @Override  public String toString() {  return "ProductList [productListId=" + productListId + ", medicineName=" + medicineName + ", seller=" + seller  + ", qtyDesc=" + qtyDesc + ", price=" + price + ", description=" + description + ", imgURL=" + imgURL  + ", active=" + active + ", CreationDateTime=" + CreationDateTime + "]";  }    } |
| package com.medicare.api.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.AdminCredential;  @Repository  public interface AdminCredentialRepository extends JpaRepository<AdminCredential, Integer>{  } |
| package com.medicare.api.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.Cart;  @Repository  public interface CartRepository extends JpaRepository<Cart, Integer>{  } |
| package com.medicare.api.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.CustomerCredential;  @Repository  public interface CustomerCredentialRepository extends JpaRepository<CustomerCredential, Integer>{  } |
| package com.medicare.api.repository;  import java.util.List;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.Customer;  import com.medicare.api.entity.OrderResponse;  @Repository  public interface CustomerRepository extends JpaRepository<Customer, Integer>{    @Query("SELECT new com.medicare.api.entity.OrderResponse(c.CusId,c.name,c.email, c.mobile, c.address, p.medicineName, p.seller, p.price, p.description, p.quantity, p.orderDateTime) FROM Customer c JOIN c.products p")  public List<OrderResponse> joinCustomerProductTable();  } |
| package com.medicare.api.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.ProductList;  @Repository  public interface ProductListRepository extends JpaRepository<ProductList, Integer>{  } |
| package com.medicare.api.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.medicare.api.entity.Product;  @Repository  public interface ProductRepository extends JpaRepository<Product, Integer>{  } |
| package com.medicare.api.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.medicare.api.entity.AdminCredential;  import com.medicare.api.repository.AdminCredentialRepository;  @Service  public class AdminCredentialService {    @Autowired  private AdminCredentialRepository adminCreRepo;  // POST Method  public AdminCredential saveAdminCredential(AdminCredential admincredential) {  return adminCreRepo.save(admincredential);  }    // GET Method  public List<AdminCredential> getAdminCredentials() {  return adminCreRepo.findAll();  }    public AdminCredential getAdminCredentialById(int id) {  return adminCreRepo.findById(id).orElse(null);  }    // PUT Method  public AdminCredential updateAdminCredential(AdminCredential admincredential, int id) {    AdminCredential existingAdminCredential = adminCreRepo.findById(id).orElse(null);  existingAdminCredential.setUsername(admincredential.getUsername());  existingAdminCredential.setPassword(admincredential.getPassword());  return adminCreRepo.save(existingAdminCredential);  }  } |
| package com.medicare.api.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.medicare.api.entity.Cart;  import com.medicare.api.repository.CartRepository;  @Service  public class CartService {    @Autowired  private CartRepository cartRepo;    // 1. insert/save the cart data to the database (POST Method)    public Cart saveCart(Cart cart) {  return cartRepo.save(cart);  }    // 2. get the cart data from database (GET Method)    public List<Cart> getCarts() {  return cartRepo.findAll();  }    public Cart getCartById(int id) {  return cartRepo.findById(id).orElse(null);  }    // 3. delete the cart data from database (DELETE Method)    public String deleteCartById(int id) {  cartRepo.deleteById(id);  return "Data "+id+" Removed Successfully";  }  } |
| package com.medicare.api.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.medicare.api.entity.CustomerCredential;  import com.medicare.api.repository.CustomerCredentialRepository;  @Service  public class CustomerCredentialService {    @Autowired  private CustomerCredentialRepository cusCreRepo;    // POST Method  public CustomerCredential saveCustomerCredential(CustomerCredential customercredential) {  return cusCreRepo.save(customercredential);  }    // GET Method  public List<CustomerCredential> getCustomerCredentials() {  return cusCreRepo.findAll();  }    public CustomerCredential getCustomerCredentialById(int id) {  return cusCreRepo.findById(id).orElse(null);  }    // PUT Method  public CustomerCredential updateCustomerCredential(CustomerCredential customercredential, int id) {  CustomerCredential existingCustomerCredential = cusCreRepo.findById(id).orElse(null);  existingCustomerCredential.setUsername(customercredential.getUsername());  existingCustomerCredential.setPassword(customercredential.getPassword());  return cusCreRepo.save(existingCustomerCredential);  }  } |
| package com.medicare.api.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.medicare.api.entity.Customer;  import com.medicare.api.entity.OrderResponse;  import com.medicare.api.repository.CustomerRepository;  @Service  public class CustomerService {    @Autowired  private CustomerRepository customerRepo;    // 1. Save the Customer to the database (POST Method)    public Customer saveCustomer(Customer customer) {  return customerRepo.save(customer);  }    // 2. fetch the Customer from database (GET Method)    public List<Customer> getCustomers() {  return customerRepo.findAll();  }    public Customer getCustomerById(int id) {  return customerRepo.findById(id).orElse(null);  }    // 3. delete the Customer from database (Delete)  public String deleteCustomerById(int id) {  customerRepo.deleteById(id);  return "Data "+id+" Deleted Successfully !";  }    // Join the Customer & Product Table    public List<OrderResponse> getCustomerProductJoinInfos() {  return customerRepo.joinCustomerProductTable();  }  } |
| package com.medicare.api.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.medicare.api.entity.ProductList;  import com.medicare.api.repository.ProductListRepository;  @Service  public class ProductListService {    @Autowired  private ProductListRepository productListRepo;    // 1. Save the product list to the database (POST Method)    public ProductList saveProductList(ProductList productlist) {  return productListRepo.save(productlist);  }    // 2. get the product list from database (GET Method)    public List<ProductList> getProductLists() {  return productListRepo.findAll();  }    public ProductList getProductListById(int id) {  return productListRepo.findById(id).orElse(null);  }    // 3. delete the product list from database (DELETE Method)    public String deleteProductListById(int id) {  productListRepo.deleteById(id);  return "Data "+id+" Deleted Successfully";  }  // 4. update the product list to the database (PUT Method)    public ProductList updateProductList(ProductList productlist, int id) {  ProductList existingProductList = productListRepo.findById(id).orElse(null);  existingProductList.setMedicineName(productlist.getMedicineName());  existingProductList.setSeller(productlist.getSeller());  existingProductList.setQtyDesc(productlist.getQtyDesc());  existingProductList.setPrice(productlist.getPrice());  existingProductList.setDescription(productlist.getDescription());  existingProductList.setImgURL(productlist.getImgURL());  return productListRepo.save(existingProductList);  }  } |
| **Project Structure:**  **Frontend UI:** |
| **About Component HTML:**  <section>  <div class="content-indentation">  <div class="data">  <div class="data-header">  <h1>About</h1>  </div>  <div class="data-body">  <p>  Medicare is a company that supplies medicines and a couple of other healthcare essentials at an affordable price.  It was established in 2012 in Delhi, India.  </p>  </div>  </div>  </div>  </section> |
| import { Component } from '@angular/core';  @Component({  selector: 'app-about',  templateUrl: './about.component.html',  styleUrls: ['./about.component.css']  })  export class AboutComponent {  } |
| **Card Component HTML:**  <div class="container">  <div class="row row-cols-3 mb-5">    <!-- single item -->  <div class="col" \*ngFor="let prod of productList | filter: filterData">  <section class="mx-auto cus-inden" style="max-width: 23rem;">  <div class="card">  <div class="bg-image hover-overlay ripple" data-bs-ripple-color="light">  <img src={{prod.imgURL}} class="img-fluid" />  </div>  <div class="card-body">  <h5 class="card-title font-weight-bold"><a>{{prod.medicineName}}</a></h5>  <p class="mb-2">{{prod.seller}}</p>  <ul class="list-unstyled list-inline mb-0">  <li class="list-inline-item">  <p class="text-muted">{{prod.qtyDesc}} Tablet(s) in Strip</p>  </li>  <li class="list-inline-item">  <h4><span class="badge bg-warning text-dark rounded-pill">{{prod.price | currency: 'INR'}}</span></h4>  </li>    </ul>  <p class="card-text">  {{prod.description}}  </p>  <button class="cus-btn" (click)="order(prod.medicineName, prod.seller, prod.price, prod.description, prod.imgURL, prod.qtyDesc)">  <span class="label">Add to Cart</span>  <span class="icon">  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24"><path fill="none" d="M0 0h24v24H0z"></path><path fill="currentColor" d="M16.172 11l-5.364-5.364 1.414-1.414L20 12l-7.778 7.778-1.414-1.414L16.172 13H4v-2z"></path></svg>  </span>  </button>  </div>  </div>  </section>  </div>  <!-- Single Item -->    </div><br><br>  </div> |
| import { MedicareService } from 'src/app/services/medicare.service';  import { Router } from '@angular/router';  import { Component, Input, OnInit } from '@angular/core';  import { ProductList } from 'src/app/Model/product-list/product-list';  import { Cart } from 'src/app/Model/Cart/cart';  @Component({  selector: 'app-card',  templateUrl: './card.component.html',  styleUrls: ['./card.component.css']  })  export class CardComponent implements OnInit{    @Input() filterData:string;  productList: ProductList[];  proList:ProductList;  proCart: Cart;  constructor(private router:Router, private medicareService: MedicareService) {}  ngOnInit(): void {  this.medicareService.getAllProductList().subscribe(data => this.productList = data);  }  order(medicine: any, seller: any, price: any, desc: any, url:any, qtyDesc: any ) {  this.proCart = {  medicineName: medicine,  seller: seller,  price: price,  description: desc,  imgURL: url,  quantity: qtyDesc  }  this.medicareService.saveCart(this.proCart).subscribe(x=>console.log(x));  this.router.navigate(['/cart']);  // console.log(this.proCart);  // console.log(JSON.stringify(proCart))  }  } |
| **Cart Component HTML:**  <Section>  <div class="content-indentation">  <app-cart-card></app-cart-card>  </div>  </Section> |
| import { Component } from '@angular/core';  @Component({  selector: 'app-cart',  templateUrl: './cart.component.html',  styleUrls: ['./cart.component.css']  })  export class CartComponent {  } |
| **Cart-Card Component HTML:**  <section class="h-100 gradient-custom">  <div class="container py-5">  <div class="row d-flex justify-content-center my-4">  <div class="col-md-8">  <div class="card mb-4">  <div class="card-header py-3">  <h5 class="mb-0">Cart - {{cartData.length}} items</h5>  </div>  <div class="card-body">  <!-- Single item -->  <div class="row" \*ngFor="let cart of cartData">  <div class="col-lg-3 col-md-12 mb-4 mb-lg-0">  <div class="bg-image hover-overlay hover-zoom ripple rounded" data-mdb-ripple-color="light">  <img src={{cart.imgURL}} class="w-100" alt="Blue Jeans Jacket" />  <a href="#!">  <div class="mask" style="background-color: rgba(251, 251, 251, 0.2)"></div>  </a>  </div>  </div>  <div class="col-lg-5 col-md-6 mb-4 mb-lg-0">  <p><strong>Medicine Name: </strong>{{cart.medicineName | titlecase}}</p>  <!-- <p>{{cart.cartId}}</p> -->  <p><strong>Seller/Brand: </strong>{{cart.seller | titlecase}}</p>  <p><strong>Quantity: </strong>{{cart.quantity}} Tablet(s) in Strip</p>  <p><strong>Description: </strong>{{cart.description | titlecase}}</p>  <button type="button" (click)="deleteCart(cart)" class="btn btn-danger btn-sm me-1 mb-2"  data-bs-toggle="tooltip" title="Remove item">  <i class="bi bi-trash3-fill"></i>  </button>  </div>  <div class="col-lg-4 col-md-6 mb-4 mb-lg-0">  <div class="d-flex mb-4" style="max-width: 300px">  <button class="btn btn-custom px-3 me-2" style="height: 37px;" (click)="dec(qty.value)"  onclick="this.parentNode.querySelector('input[type=number]').stepDown()">  <i class="bi bi-dash-lg"></i>  </button>  <div class="form-outline">  <input id="form1" min="0" name="quantity" value="1" #qty type="number" class="form-control" />  <label class="form-label" for="form1">Quantity</label>  </div>  <button class="btn btn-custom px-3 ms-2" style="height: 37px;" (click)="dec(qty.value)"  onclick="this.parentNode.querySelector('input[type=number]').stepUp()">  <i class="bi bi-plus-lg"></i>  </button>  </div>  <p class="text-start text-md-center">  <strong>{{cart.price\*qtyVal | currency: 'INR'}}</strong>  </p>  </div>  <hr class="my-4" />  </div>  <!-- Single item -->  </div>  </div>  </div>  <div class="col-md-4">  <div class="card mb-4">  <div class="card-header py-3">  <h5 class="mb-0">Customer Details</h5>  </div>  <div class="card-body">  <form>  <div class="row g-3">  <div class="col-lg-8">  <label class="form-label">Name</label>  <input type="text" required class="form-control" placeholder="name" #name >  </div>  <div class="col-lg-4">  <label class="form-label">Age</label>  <input type="text" required class="form-control" placeholder="age" #age>  </div>  <div class="col-lg-4">  <label class="form-label">Gender</label>  <input type="text" required class="form-control" placeholder="gender" #gender>  </div>  <div class="col-lg-8">  <label class="form-label">E-mail</label>  <input type="text" required class="form-control" placeholder="email" #email>  </div>  <div class="col-lg-12">  <label class="form-label">Address</label>  <input type="text" required class="form-control" placeholder="address" #address>  </div>  <div class="col-lg-12">  <label class="form-label">Mobile</label>  <input type="text" required class="form-control" placeholder="mobile number" #mobile>  </div>  <div class="col-lg-12">  <button type="button"  (click)="placeOrder(cartData, name.value, age.value, gender.value, email.value, address.value, mobile.value)"  class="btn btn-custom">  Place Order  </button>  </div>  </div>  </form>  </div>  </div>  </div>  </div>  </div>  </section> |
| import { Cart } from './../../Model/Cart/cart';  import { MedicareService } from 'src/app/services/medicare.service';  import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  import { Customer } from 'src/app/Model/Customer/customer';  @Component({  selector: 'app-cart-card',  templateUrl: './cart-card.component.html',  styleUrls: ['./cart-card.component.css']  })  export class CartCardComponent implements OnInit{  cartData: Cart[];  customer: Customer;  constructor(private router: Router, private medicareService: MedicareService) {  // this.proId = this.router.getCurrentNavigation()?.extras.state?.['pid'];  }    ngOnInit(): void {  this.medicareService.getAllCarts().subscribe(x => this.cartData = x);  }  deleteCart(cart: Cart) {  // console.log(cart.cartId);  this.cartData = this.cartData.filter(data => data.cartId !== cart.cartId);  this.medicareService.deleteCartById(Number(cart.cartId)).subscribe(x=> console.log(x));    this.router.navigate(['/cart']);  }  qtyVal: number = 1;  dec(dec: any) {  this.qtyVal = Number(dec);  }  placeOrder(cartAllData: any, name: any, age: any, gen: any, email: any, add: any, mob: any) {  // console.log(name);  // console.log(age);  // console.log(gen);  // console.log(email);  // console.log(add);  // console.log(mob);  // console.log(this.qtyVal);  // console.log(cartAllData);  this.customer = {  name: name,  age: age,  gender: gen,  email: email,  address: add,  mobile: mob,  products: cartAllData  }  // console.log(this.customer);  this.medicareService.placeOrder(this.customer).subscribe(x=>console.log(x));  alert("Your Order Placed Successfully !");  this.router.navigate(['/home']);  }    } |
| **Contact Component HTML:**  <section>  <div class="content-indentation">  <div class="data">  <div class="data-header">  <h1>Contact</h1>  </div>  <div class="data-body">  <div class="data-text">  <p class="icon me-5"><strong><i class="bi bi-envelope-at-fill"></i></strong></p>  <p>care@medicare.com</p>  </div>  <div class="data-text">  <p class="icon me-5"><i class="bi bi-telephone-outbound-fill"></i></p>  <p>+91 99999-88888</p>  </div>  <div class="data-text">  <p class="icon me-5"><i class="bi bi-buildings-fill"></i></p>  <p>Delhi, India - 110096</p>  </div>  </div>  </div>  </div>  </section> |
| import { Component } from '@angular/core';  @Component({  selector: 'app-contact',  templateUrl: './contact.component.html',  styleUrls: ['./contact.component.css']  })  export class ContactComponent {  } |
| **Admin Component HTML:**  <section class="content-center">  <div class="login-box">  <p>Admin</p>  <form>  <div class="user-box">  <input required type="text" #user>  <label>Username</label>  </div>  <div class="user-box">  <input required type="text" #pass>  <label>Password</label>  </div>  <a class="custom-btn" (click)="adminCred(user.value, pass.value)">  <span></span>  <span></span>  <span></span>  <span></span>  Submit  </a>  </form>  <p>Change Your Password? <a class="a2" (click)="changeAdmin()">Click</a></p>  </div>  </section> |
| import { MedicareService } from 'src/app/services/medicare.service';  import { AdminCreden } from './../../../Model/Credential/Admin/admin-creden';  import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-admin',  templateUrl: './admin.component.html',  styleUrls: ['./admin.component.css']  })  export class AdminComponent implements OnInit{  adminCreden: AdminCreden[];  constructor(private router: Router, private medicareService: MedicareService) {}  ngOnInit(): void {  this.medicareService.getAllAdmin().subscribe(data => this.adminCreden = data);  }  adminCred(user: any, pass: any) {  // console.log(this.adminCreden[0]);  for(let x of this.adminCreden){  // console.log(x.username);  // console.log(x.password);  // console.log(user+" "+pass);  if(x.username == user && x.password == pass) {  // console.log("you are Logged In");  this.router.navigate(['/manage-product'],{state: {u: x.username}});  }else{  alert("You Entered Wrong username & Password ! Try Again !");  }  }  }  changeAdmin() {  this.router.navigate(['/manage-product/change-password']);  }  } |
| **Login Component HTML:**  <section class="content-center">  <div class="login-box">  <p>Login</p>  <form>  <div class="user-box">  <input required type="text" #custUser>  <label>Username</label>  </div>  <div class="user-box">  <input required type="text" #custPass>  <label>Password</label>  </div>  <a class="custom-btn" (click)="login(custUser.value, custPass.value)">  <span></span>  <span></span>  <span></span>  <span></span>  Login  </a>  </form>  <p>Don't have an account? <a class="a2" (click)="switchSignUp()">Sign Up</a></p>  </div>  </section>  <!-- Forgot Your Password --> |
| import { MedicareService } from 'src/app/services/medicare.service';  import { CustomerCreden } from './../../../Model/Credential/Customer/customer-creden';  import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-login',  templateUrl: './login.component.html',  styleUrls: ['./login.component.css']  })  export class LoginComponent implements OnInit{  customerCreden: CustomerCreden[];  constructor(private router: Router, private medicareService: MedicareService) {}    ngOnInit(): void {  this.medicareService.getAllCustomerCreden().subscribe(data => this.customerCreden = data);  }  login(custUser:any, custPass:any) {  console.log(custUser+" "+custPass);  for(let x of this.customerCreden) {  if(x.username == custUser && x.password == custPass) {  this.router.navigate(['/product']);  }  }  }  switchSignUp(): any {  this.router.navigate(['signup']);  }  } |
| **Signup component HTML:**  <section class="content-center">  <div class="login-box">  <p>Sign Up</p>  <form>  <div class="user-box">  <input required type="text" #username>  <label>Username</label>  </div>  <div class="user-box">  <input required type="text" #password>  <label>Password</label>  </div>  <a class="custom-btn" (click)="signup(username.value, password.value)">  <span></span>  <span></span>  <span></span>  <span></span>  Sign Up  </a>  </form>  <p>Already you have an account? <a class="a2" (click)="switchLogin()">Login</a></p>  </div>  </section> |
| import { MedicareService } from 'src/app/services/medicare.service';  import { CustomerCreden } from './../../../Model/Credential/Customer/customer-creden';  import { Router } from '@angular/router';  import { Component } from '@angular/core';  @Component({  selector: 'app-signup',  templateUrl: './signup.component.html',  styleUrls: ['./signup.component.css']  })  export class SignupComponent {  custCreden: CustomerCreden;  constructor(private router: Router, private medicareService: MedicareService) {}  signup(username:any, password:any) {  // console.log(username+" "+password);  this.custCreden = {  username: username,  password: password  }  this.medicareService.saveCustomerCreden(this.custCreden).subscribe(x=> alert("you Data Save Successfully ! Now You Can Logged In."));  }  switchLogin(): any {  this.router.navigate(['login']);  }  } |
| **Footer Component HTML:**  <footer class="text-center text-white fixed-bottom" style="background-color: #3cb876;">  <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2);">  Design and Developed by : Kavin K R  </div>  </footer> |
| **Header Component HTML:**  <nav class="navbar navbar-expand-lg navbar-light fixed-top mask-custom shadow-0">  <div class="container">  <a class="navbar-brand" routerLink="home">  <span class="logo-1">Medi</span>  <span class="logo-2">Care</span>  </a>  <button class="navbar-toggler mob-nav-bar" type="button" data-bs-toggle="collapse"  data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"  aria-label="Toggle navigation">  <i class="bi bi-list-nested"></i>  </button>  <div class="collapse navbar-collapse" id="navbarSupportedContent">  <ul class="navbar-nav me-auto ms-5" >  <li class="nav-item">  <a class="nav-link medicare-navlink ms-5" routerLink="product">Product</a>  </li>  <li class="nav-item">  <a class="nav-link medicare-navlink ms-5" routerLink="about">About</a>  </li>  <li class="nav-item">  <a class="nav-link medicare-navlink ms-5" routerLink="contact">Contact</a>  </li>  <li class="nav-item">  <a class="nav-link medicare-navlink ms-5" routerLink="manage-product" \*ngIf="adminName">Manage Product</a>  </li>  </ul>  <ul class="navbar-nav d-flex flex-row">  <li class="nav-item me-3 me-lg-0">  <a class="nav-link credential" routerLink="signup">  <i class="bi bi-person"></i> Sign Up  </a>  </li>  <li class="nav-item me-3 me-lg-0">  <a class="nav-link credential" routerLink="login">  <i class="bi bi-box-arrow-in-right"></i> Login  </a>  </li>  <li class="nav-item me-3 me-lg-0">  <a class="nav-link credential" routerLink="admin">  <i class="bi bi-shield-lock-fill"></i> Admin  </a>  </li>  <li class="nav-item me-3 me-lg-0">  <a class="nav-link credential" routerLink="cart">  <i class="bi bi-cart-fill"></i> Cart  </a>  </li>  </ul>  </div>  </div>  </nav> |
| import { Component } from '@angular/core';  @Component({  selector: 'app-header',  templateUrl: './header.component.html',  styleUrls: ['./header.component.css']  })  export class HeaderComponent{  adminName: any = true;  } |
| **Home Component HTML:**  <div class="content-indentation">  <div class="content-center">  <ngx-typed-js [strings]="['MediCare']" [loop]="true" [typeSpeed]="170" [backSpeed]="170" [startDelay]="300" [backDelay]="1000">  <h1>Welcome to <span class="typing"></span></h1>  </ngx-typed-js>  <button (click)="product()">  <span>ORDER NOW</span><i></i>  </button>  </div>  </div> |
| import { Component } from '@angular/core';  import { Router } from '@angular/router';  @Component({  selector: 'app-home',  templateUrl: './home.component.html',  styleUrls: ['./home.component.css']  })  export class HomeComponent {  constructor(private router: Router) {}  product(): any {  this.router.navigate(['/login']);  }  } |
| **Add Product Component HTML:**  <section>  <div class="add-pro">  <h1>Add Product</h1>  <div class="form-container">  <form [formGroup]="addProductForm" (ngSubmit)="onSubmit()">  <div class="row g-3">  <div class="col-md-4">  <label class="form-label">Medicine Name</label>  <input type="text" placeholder="medicine name" formControlName="name" class="form-control" [(ngModel)]="productList.medicineName"  [ngClass]="{'is-invalid':submitted && form['name'].errors}">  <div \*ngIf="submitted && form['name'].errors" class="invalid-feedback">  Medicine Name Cannot be Empty  </div>  </div>  <div class="col-md-4">  <label class="form-label">Seller</label>  <input type="text" placeholder="Seller/Brand name" formControlName="seller" class="form-control" [(ngModel)]="productList.seller"  [ngClass]="{'is-invalid':submitted && form['seller'].errors}">  <div \*ngIf="submitted && form['seller'].errors" class="invalid-feedback">  Seller/Brand Cannot be Empty  </div>  </div>  <div class="col-md-4">  <label class="form-label">Image URL</label>  <input type="text" placeholder="medicine image url" formControlName="url" class="form-control" [(ngModel)]="productList.imgURL"  [ngClass]="{'is-invalid':submitted && form['url'].errors}">  <div \*ngIf="submitted && form['url'].errors" class="invalid-feedback">  Medicine Image URL Cannot be Empty  </div>  </div>  <div class="col-md-4">  <label class="form-label">Medicine Qty (in a Strip)</label>  <input type="text" placeholder="medicine Quantity in a strip" formControlName="qty" class="form-control" [(ngModel)]="productList.qtyDesc"  [ngClass]="{'is-invalid':submitted && form['qty'].errors}">  <div \*ngIf="submitted && form['qty'].errors" class="invalid-feedback">  Medicine Quantity Cannot be Empty  </div>  </div>  <div class="col-md-2">  <label class="form-label">Price</label>  <input type="text" placeholder="medicine price" formControlName="price" class="form-control" [(ngModel)]="productList.price"  [ngClass]="{'is-invalid':submitted && form['price'].errors}">  <div \*ngIf="submitted && form['price'].errors" class="invalid-feedback">  Price Cannot be Empty  </div>  </div>  <div class="col-md-6">  <label class="form-label">Description</label>  <input type="text" placeholder="description of medicine" formControlName="desc" class="form-control" [(ngModel)]="productList.description"  [ngClass]="{'is-invalid':submitted && form['desc'].errors}">  <div \*ngIf="submitted && form['desc'].errors" class="invalid-feedback">  Description Cannot be Empty  </div>  </div>  <br>  <div class="col-12">  <button class="btn btn-success">Submit</button>  </div>  </div>  </form>  </div>  </div>  </section> |
| import { MedicareService } from './../../../services/medicare.service';  import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  import { ProductList } from 'src/app/Model/product-list/product-list';  @Component({  selector: 'app-add-product',  templateUrl: './add-product.component.html',  styleUrls: ['./add-product.component.css']  })  export class AddProductComponent implements OnInit {  productList: ProductList = new ProductList();  constructor(private builder:FormBuilder, private router: Router, private medicareService: MedicareService) {}    addProductForm:FormGroup;  submitted:boolean=false;  ngOnInit(): void {  this.addProductForm=this.builder.group({  name:['',Validators.required],  seller:['',Validators.required],  url:['',Validators.required],  qty:['',Validators.required],  price:['',Validators.required],  desc:['',Validators.required],  })  }  get form(){  return this.addProductForm.controls;  }  onSubmit() {  // console.log(this.productList);  this.submitted=true;  if(this.addProductForm.invalid){  return  }else {  // console.log(this.productList);  this.medicareService.saveProductList(this.productList).subscribe(data => console.log(data));  this.router.navigate(['/manage-product']);  }  }  } |
| **Change Password Component HTML:**  <section>  <div class="change-pass">  <h1>Change Admin Password</h1>  <div class="login-box">  <form>  <div class="user-box">  <input required="" name="" type="text">  <label>Enter Old Password</label>  </div>  <div class="user-box">  <input required="" name="" type="password">  <label>Set New Password</label>  </div>  <div>  <a href="#">  SEND  <span></span>  </a>  </div>  </form>  </div>  </div>  </section> |
| **Edit Product component HTML:**  <section>  <div class="update-pro">  <h1>Update My Product : {{prodList.productListId}}</h1>  <div class="form-container">    <form [formGroup]="updateProductForm" (ngSubmit)="onUpdate()">  <div class="row g-3">  <div class="col-md-4">  <label class="form-label">Medicine Name</label>  <input type="text" placeholder="medicine name" formControlName="name" class="form-control" [(ngModel)]="prodList.medicineName"  [ngClass]="{'is-invalid':submitted && form['name'].errors}">    <div \*ngIf="submitted && form['name'].errors" class="invalid-feedback">  Medicine Name Cannot be Empty  </div>  </div>    <div class="col-md-4">  <label class="form-label">Seller</label>  <input type="text" placeholder="Seller/Brand name" formControlName="seller" class="form-control" [(ngModel)]="prodList.seller"  [ngClass]="{'is-invalid':submitted && form['seller'].errors}">    <div \*ngIf="submitted && form['seller'].errors" class="invalid-feedback">  Seller/Brand Cannot be Empty  </div>  </div>    <div class="col-md-4">  <label class="form-label">Image URL</label>  <input type="text" placeholder="medicine image url" formControlName="url" class="form-control" [(ngModel)]="prodList.imgURL"  [ngClass]="{'is-invalid':submitted && form['url'].errors}">    <div \*ngIf="submitted && form['url'].errors" class="invalid-feedback">  Medicine Image URL Cannot be Empty  </div>  </div>    <div class="col-md-4">  <label class="form-label">Medicine Qty (in a Strip)</label>  <input type="text" placeholder="medicine Quantity in a strip" formControlName="qty" class="form-control" [(ngModel)]="prodList.qtyDesc"  [ngClass]="{'is-invalid':submitted && form['qty'].errors}">    <div \*ngIf="submitted && form['qty'].errors" class="invalid-feedback">  Medicine Quantity Cannot be Empty  </div>  </div>    <div class="col-md-2">  <label class="form-label">Price</label>  <input type="text" placeholder="medicine price" formControlName="price" class="form-control" [(ngModel)]="prodList.price"  [ngClass]="{'is-invalid':submitted && form['price'].errors}">    <div \*ngIf="submitted && form['price'].errors" class="invalid-feedback">  Price Cannot be Empty  </div>  </div>    <div class="col-md-6">  <label class="form-label">Description</label>  <input type="text" placeholder="description of medicine" formControlName="desc" class="form-control" [(ngModel)]="prodList.description"  [ngClass]="{'is-invalid':submitted && form['desc'].errors}">    <div \*ngIf="submitted && form['desc'].errors" class="invalid-feedback">  Description Cannot be Empty  </div>  </div>  <br>  <div class="col-12">  <button class="btn btn-success">Update</button>  </div>  </div>  </form>  </div>  </div>    </section> |
| import { ActivatedRoute, Router } from '@angular/router';  import { MedicareService } from 'src/app/services/medicare.service';  import { FormGroup, FormBuilder, Validators } from '@angular/forms';  import { Component, OnInit } from '@angular/core';  import { ProductList } from 'src/app/Model/product-list/product-list';  @Component({  selector: 'app-edit-product',  templateUrl: './edit-product.component.html',  styleUrls: ['./edit-product.component.css']  })  export class EditProductComponent implements OnInit {  prodList: ProductList;  id: string|null;  updateProductForm:FormGroup;  submitted: boolean = false;  constructor(private medicareService: MedicareService, private activatedRoute: ActivatedRoute, private builder: FormBuilder, private router: Router) {}  ngOnInit(): void {  this.id = this.activatedRoute.snapshot.paramMap.get('id');  // alert(this.id);  this.medicareService.getProductListById(Number(this.id)).subscribe(x=> this.prodList=x);  this.updateProductForm = this.builder.group({  name:['',Validators.required],  seller:['',Validators.required],  url:['',Validators.required],  qty:['',Validators.required],  price:['',Validators.required],  desc:['',Validators.required],  })    }  get form() {  return this.updateProductForm.controls;  }  onUpdate() {  this.submitted = true;  if(this.updateProductForm.invalid){  return  }  else{  // console.log(this.prodList);  this.medicareService.updateProductList(this.prodList, Number(this.id)).subscribe(x=>console.log(x));  alert("Data Updated Successfully !");  this.router.navigate(['manage-product']);  }  }  } |
| **Home-product component HTML:**  <section>  <div class="content-indentation">  <div class="row">  <div class="col-lg-2 nav-product">  <ul class="nav flex-column">  <li class="nav-item">  {{username}}  </li>  <li class="nav-item">  <a class="nav-link active" routerLink="product-list">  <button>Product List</button>  </a>  </li>    <li class="nav-item">  <a class="nav-link" routerLink="add-product">  <button>Add Product</button>  </a>  </li>  <li class="nav-item">  <a class="nav-link" routerLink="order-list">  <button>Order List</button>  </a>  </li>  <li class="nav-item">  <a class="nav-link" routerLink="change-password">  <button>Change Password</button>  </a>  </li>  </ul>  </div>  <div class="col-lg-10">  <router-outlet></router-outlet>  </div>  </div>  </div>  </section> |
| import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-home-product',  templateUrl: './home-product.component.html',  styleUrls: ['./home-product.component.css']  })  export class HomeProductComponent implements OnInit{  username: any;  constructor(private router: Router) {  this.username = this.router.getCurrentNavigation()?.extras.state?.['u'];  // console.log(this.username);  }  ngOnInit(): void {    }  } |
| **Order-List component HTML:**  <section>  <div class="or-list">  <h1>Order List</h1>  <div class="table-responsive">  <table class="table table-dark table-striped table-hover table-borderless align-middle">  <thead>  <tr>  <th>C.ID</th>  <th>Name</th>  <th>email</th>  <th>Mobile</th>  <th>Address</th>  <th>Medicine Name</th>  <th>Seller</th>  <th>Price</th>  <th>Description</th>  <th>Quantity</th>  <th>Date/Time</th>  <th>Operations</th>  </tr>  </thead>  <tbody>  <!-- single -->  <tr \*ngFor="let order of orderList">  <td>{{order.cusId}}</td>  <td>{{order.name | titlecase}}</td>  <td>{{order.email}}</td>  <td>{{order.mobile}}</td>  <td>{{order.address | titlecase}}</td>  <td>{{order.medicineName | titlecase}}</td>  <td>{{order.seller | titlecase}}</td>  <td>{{order.price | currency: 'INR'}}</td>  <td>{{order.description}}</td>  <td>{{order.quantity}}</td>  <td>{{order.orderDateTime | date: 'd MMM y, h:mm:ss a'}}</td>  <td>  <button class="btn btn-danger" (click)="orderDelete(order.cusId)">Delete</button>  </td>  </tr>  <!-- single -->  </tbody>  </table>  </div>  </div>  </section> |
| import { Customer } from './../../../Model/Customer/customer';  import { MedicareService } from 'src/app/services/medicare.service';  import { Router } from '@angular/router';  import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-order-list',  templateUrl: './order-list.component.html',  styleUrls: ['./order-list.component.css']  })  export class OrderListComponent implements OnInit{  orderList: any;  constructor(private router: Router, private medicareService: MedicareService) {}  ngOnInit(): void {  this.medicareService.getAllOrderList().subscribe(data => this.orderList = data);  }  orderDelete(id: number) {  this.medicareService.deleteCustomerProductDetailsById(Number(id)).subscribe(data => console.log(data));  this.orderList = this.orderList.filter((x: { cusId: number; }) => x.cusId !== id)  }  } |
| **Product-details component HTML:**  <section>  <div class="pro-details">  <table class="table table-dark table-borderless">    <div class="custom-content">  <tr>  <td>  <h1>Product Details : {{productList.productListId}}</h1>  </td>  </tr>  <tr>  <td><strong>Medicine Name: </strong> {{productList.medicineName}}</td>    </tr>  <tr>  <td><strong>Seller: </strong> {{productList.seller}}</td>  </tr>  <tr>  <td><strong>Price: </strong>{{productList.price | currency: 'INR'}}</td>  </tr>  <tr>  <td><strong>Qty Desc: </strong>{{productList.qtyDesc}}</td>  </tr>  <tr>  <td><strong>Active: </strong>{{productList.active}}</td>  </tr>  <tr>  <td><strong>Date/Time: </strong>{{ productList.creationDateTime | date:'d MMM y, h:mm:ss a'}}</td>  </tr>  <tr>  <td><strong>Image URL: </strong>{{productList.imgURL}}</td>  </tr>  <tr>  <td><strong>Description: </strong>{{productList.description}}</td>  </tr>  </div>  </table>  </div>  </section> |
| import { MedicareService } from './../../../services/medicare.service';  // import { IProductList } from './../../../interfaces/product-list/product-list';  import { Component, OnInit } from '@angular/core';  import { ActivatedRoute } from '@angular/router';  @Component({  selector: 'app-product-details',  templateUrl: './product-details.component.html',  styleUrls: ['./product-details.component.css']  })  export class ProductDetailsComponent implements OnInit {    productList: any;  constructor(private activatedRoute: ActivatedRoute, private medicareService: MedicareService) {}  ngOnInit(): void {  const id = this.activatedRoute.snapshot.paramMap.get('id');  this.medicareService.getProductListById(Number(id)).subscribe(d => this.productList = d);    // this.activatedRoute.params.subscribe( (params) => {  // this.medicareService.getProductListById(+params['id']).subscribe(data => this.productList = data);  // });  }  } |
| **Product-list component HTML:**  <section>  <div class="pro-list">  <h1>Product List</h1>  <table class="table table-dark table-striped table-hover table-borderless">  <thead>  <tr>  <th scope="col">P. ID</th>  <th scope="col">Medicine Name</th>  <th scope="col">Seller</th>  <th scope="col">Price</th>  <th scope="col">Description</th>  <th scope="col">Image URL</th>  <th scope="col">Qty Desc</th>  <th scope="col">Active</th>  <th scope="col">Date/Time</th>  <th scope="col">Operations</th>  </tr>  </thead>  <tbody>  <!-- Single Cell -->  <tr \*ngFor="let proList of productList">  <td>{{ proList.productListId }}</td>  <td>{{ proList.medicineName | slice:0:16 }}</td>  <td>{{ proList.seller | slice:0:10}}</td>  <td>{{ proList.price | currency: 'INR' }}</td>  <td>{{ proList.description | slice:0:19}}</td>  <td>{{ proList.imgURL | slice:0:20}}</td>  <td>{{ proList.qtyDesc | slice:0:9 }}</td>  <td>{{ proList.active }}</td>  <td>{{ proList.creationDateTime | date:'d MMM y, h:mm:ss a'}}</td>  <td>  <button class="btn btn-success" routerLink="/product-details/{{ proList.productListId }}">View</button>    <button class="btn btn-primary mx-2" routerLink="./edit-product/{{proList.productListId}}">Edit</button>    <button class="btn btn-danger" (click)="delete(proList.productListId)">Delete</button>  </td>  </tr>  <!-- Single Cell -->  </tbody>  </table>  </div>  </section> |
| // import { IProductList } from './../../../interfaces/product-list/product-list';  import { MedicareService } from './../../../services/medicare.service';  import { Component, OnInit } from '@angular/core';  import { ProductList } from 'src/app/Model/product-list/product-list';  @Component({  selector: 'app-product-list',  templateUrl: './product-list.component.html',  styleUrls: ['./product-list.component.css']  })  export class ProductListComponent implements OnInit{  // Injected medicare service dependency in component(here...)  constructor(private medicareService: MedicareService) {}  productList: ProductList[];  ngOnInit(): void {  this.medicareService.getAllProductList().subscribe(data => this.productList = data);  }  delete(id: number) {  this.medicareService.deleteProductList(id).subscribe(data => console.log(data));  this.productList = this.productList.filter((data) => {  return data.productListId != id;  });      }  } |
| **Product component HTML:**  <section>  <div class="content-indentation">  <div class="container">  <div>  <div class="input-group rounded">  <input type="text" class="form-control rounded" placeholder="Search" [(ngModel)]="enteredData"/>  </div>  </div>  </div>  <app-card [filterData] = "enteredData"></app-card>  </div>  </section> |
| import { Component } from '@angular/core';  @Component({  selector: 'app-product',  templateUrl: './product.component.html',  styleUrls: ['./product.component.css']  })  export class ProductComponent {  enteredData: string;  } |
| export class Cart {  cartId?: number;  medicineName: string;  seller: string;  price: number;  description: string;  imgURL: string;  quantity: number  } |
| export class AdminCreden {  adminId: number;  username: string;  password: string;  } |
| export class CustomerCreden {  customerId?: number;  username: string;  password: string;  } |
| export class Customer {  CusId?: number;  name: string;  age: number;  gender: string;  email: string;  address: string;  mobile: string;  products: [  {  pid?: number;  medicineName: string;  seller: string;  price: number;  description: string;  quantity: number  }  ]  } |
| export class ProductList {  productListId: number;  medicineName: string;  seller: string;  qtyDesc?: string;  price: number;  description: string;  imgURL: string;  active: number;  creationDateTime: Date;  } |
| import { CustomerCreden } from './../Model/Credential/Customer/customer-creden';  // import { IProductList } from './../interfaces/product-list/product-list';  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { Observable } from 'rxjs';  import { ProductList } from '../Model/product-list/product-list';  import { Cart } from '../Model/Cart/cart';  import { Customer } from '../Model/Customer/customer';  import { AdminCreden } from '../Model/Credential/Admin/admin-creden';  @Injectable({  providedIn: 'root'  })  export class MedicareService {  private \_productListUrl = "http://localhost:8080/api/medicare/product/list";  private \_orderListUrl = "http://localhost:8080/api/medicare/order/list";  private \_cartUrl = "http://localhost:8080/api/medicare/cart";  private \_customerUrl="http://localhost:8080/api/medicare/customer";  private \_adminCredentialUrl = "http://localhost:8080/api/medicare/admin/credential";  private \_customerCredentialUrl = "http://localhost:8080/api/medicare/customer/credential";    constructor(private http: HttpClient) { }  // -----------------Customer & Product (Place Order)---------------------  // 1. Get Method  getAllOrderList(): Observable<Customer[]> {  return this.http.get<Customer[]>(this.\_orderListUrl);  }  // 2. Post Method  placeOrder(customer: Customer): Observable<Customer> {  return this.http.post<Customer>(this.\_customerUrl, customer);  }  // 3. Delete Method  deleteCustomerProductDetailsById(id: number): Observable<Customer> {  return this.http.delete<Customer>(`${this.\_customerUrl}/${id}`)  }  // ----------------------------Product List------------------------------------  // 1. GET Method  getAllProductList(): Observable<ProductList[]> {  return this.http.get<ProductList[]>(this.\_productListUrl);  }  getProductListById(id: number): Observable<ProductList> {  return this.http.get<ProductList>(`${this.\_productListUrl}/${id}`);  }  // 2. Post Method  saveProductList(productlist: ProductList): Observable<ProductList> {  return this.http.post<ProductList>(this.\_productListUrl, productlist);  }  // 3. Put Method  updateProductList(productlist: ProductList, id: number): Observable<ProductList> {  return this.http.put<ProductList>(`${this.\_productListUrl}/${id}`,productlist);  }  // 4. Delete Method  deleteProductList(id: number): Observable<ProductList> {  return this.http.delete<ProductList>(`${this.\_productListUrl}/${id}`);  }  // ---------------------------------Cart------------------------------------------  // 1. Get Method  getAllCarts(): Observable<Cart[]> {  return this.http.get<Cart[]>(this.\_cartUrl);  }  getCartById(id: number): Observable<Cart> {  return this.http.get<Cart>(`${this.\_cartUrl}/${id}`);  }  // 2. Post Method  saveCart(cart: Cart): Observable<ProductList> {  return this.http.post<ProductList>(this.\_cartUrl, cart);  }  // 3. Delete Method  deleteCartById(id: number): Observable<Cart> {  return this.http.delete<Cart>(`${this.\_cartUrl}/${id}`);  }  // ---------------------Admin Credential------------------------  // 1. Get Method  getAllAdmin(): Observable<AdminCreden[]> {  return this.http.get<AdminCreden[]>(this.\_adminCredentialUrl);  }  getAdminById(id: number): Observable<AdminCreden> {  return this.http.get<AdminCreden>(`${this.\_adminCredentialUrl}/${id}`);  }  // 2. Put method  updateAdminById(admin: AdminCreden, id: number): Observable<AdminCreden> {  return this.http.put<AdminCreden>(`${this.\_adminCredentialUrl}/${id}`,admin);  }  //----------------------Customer Credential----------------------  // 1. Get Method  getAllCustomerCreden(): Observable<CustomerCreden[]> {  return this.http.get<CustomerCreden[]>(this.\_customerCredentialUrl);  }  // 2. Post Method  saveCustomerCreden(customercreden: CustomerCreden): Observable<CustomerCreden> {  return this.http.post<CustomerCreden>(this.\_customerCredentialUrl, customercreden);  }    } |
| import { EditProductComponent } from './components/ManageProduct/edit-product/edit-product.component';  import { ProductDetailsComponent } from './components/ManageProduct/product-details/product-details.component';  import { OrderListComponent } from './components/ManageProduct/order-list/order-list.component';  import { ChangePasswordComponent } from './components/ManageProduct/change-password/change-password.component';  import { AddProductComponent } from './components/ManageProduct/add-product/add-product.component';  import { ProductListComponent } from './components/ManageProduct/product-list/product-list.component';  import { HomeProductComponent } from './components/ManageProduct/home-product/home-product.component';  import { SignupComponent } from './components/credential/signup/signup.component';  import { LoginComponent } from './components/credential/login/login.component';  import { AdminComponent } from './components/credential/admin/admin.component';  import { CartComponent } from './components/cart/cart.component';  import { DescCardComponent } from './components/desc-card/desc-card.component';  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { AboutComponent } from './components/about/about.component';  import { ContactComponent } from './components/contact/contact.component';  import { HomeComponent } from './components/home/home.component';  import { ProductComponent } from './components/product/product.component';  const routes: Routes = [  {  path: 'product-details/:id',  component: ProductDetailsComponent  },  {  path: 'manage-product',  component: HomeProductComponent,  children: [  {  path: 'edit-product/:id',  component: EditProductComponent  },  {  path: 'order-list',  component: OrderListComponent  },  {  path: 'change-password',  component: ChangePasswordComponent  },  {  path: 'product-list',  component: ProductListComponent,  },  {  path: 'add-product',  component: AddProductComponent  },  {  path: '',  component: ProductListComponent  },  {  path: '\*\*',  component: ProductListComponent  }  ]  },  {  path: 'signup',  component: SignupComponent  },  {  path: 'login',  component: LoginComponent  },  {  path: 'admin',  component: AdminComponent  },  {  path: 'cart',  component: CartComponent  },  {  path: 'desc-card',  component: DescCardComponent  },  {  path: 'product',  component: ProductComponent  },  {  path: 'about',  component: AboutComponent  },  {  path: 'contact',  component: ContactComponent  },  {  path: 'home',  component: HomeComponent  },  {  path: '',  component: HomeComponent  },  {  path: '\*\*',  component: HomeComponent  }  ];  @NgModule({  imports: [RouterModule.forRoot(routes)],  exports: [RouterModule]  })  export class AppRoutingModule { } |
| **App.component.html:**  <section>  <app-header></app-header>  <router-outlet></router-outlet>  </section>  <section>  <app-footer></app-footer>  </section> |