**PROJECT NAME – FOODBOX AWS EC2 Deployment**

**Developer Name – Malini Murthy**

**Email id – krmalini77@gmail.com**

**Contact No – 414-629-6041**

**https://github.com/malinimurthy123/FoodBoxDevOps/tree/master**

**https://github.com/malinimurthy123/FoodBoxDevOps.git**

**https://github.com/malinimurthy123/FoodBoxDevOps**

**CI/CD Deployment for Springboot Application.**

Course-end Project 1

DESCRIPTION

As a Full Stack Developer, you have to build a CI/CD pipeline to demonstrate continuous deployment and host the application on AWS EC2 instance.

**Background of the problem statement:**

As the project is in the final stage, management has asked you to automate the integration and deployment of the web application. You are required to set up an environment where the application will be hosted and accessed by users. The source code is supposed to be fetched from a GitHub repository.

**You must use the following:**

* Eclipse
* GitHub
* Jenkins
* AWS EC2/ Virtual machine

**Following requirements should be met:**

* A part of the source code should be tracked on the GitHub repository. You need to document the tracked files that are ignored during the final push to the GitHub repository.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository in the document.
* The step-by-step process involved in completing this task should be documented.

**Project Description: An ecommerce Application design for the food delivery purpose using this application we can buy a food and do payment online and it include admin page and user page user can buy food and give feedback and there is admin dashboard who can handle this application. Example - He can add, delete, update food and maintain the Customer list**

**Environment:**

**Backend – Java 17, Spring Boot 3.0.6, Spring Data JPA 2.7.5**

**Database – MySQL**

**Frontend – Typescript, JavaScript, Angular 17 , HTML , CSS , JQUERY , Angular Material**

**DevOps Tools – GitHub, Jenkins, AWS EC2 Instance (for learning purpose)**

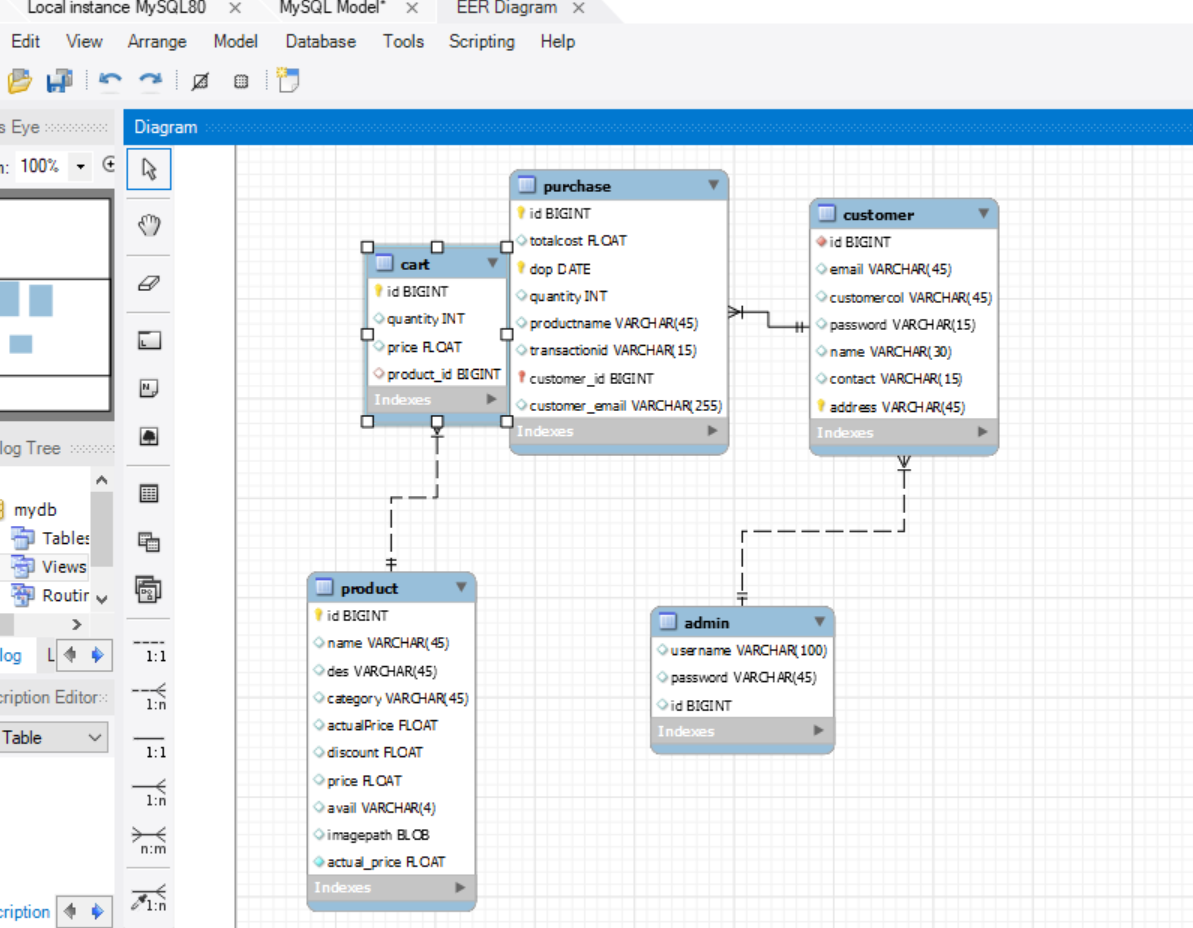
https://github.com/malinimurthy123/FoodBoxDevOps.git

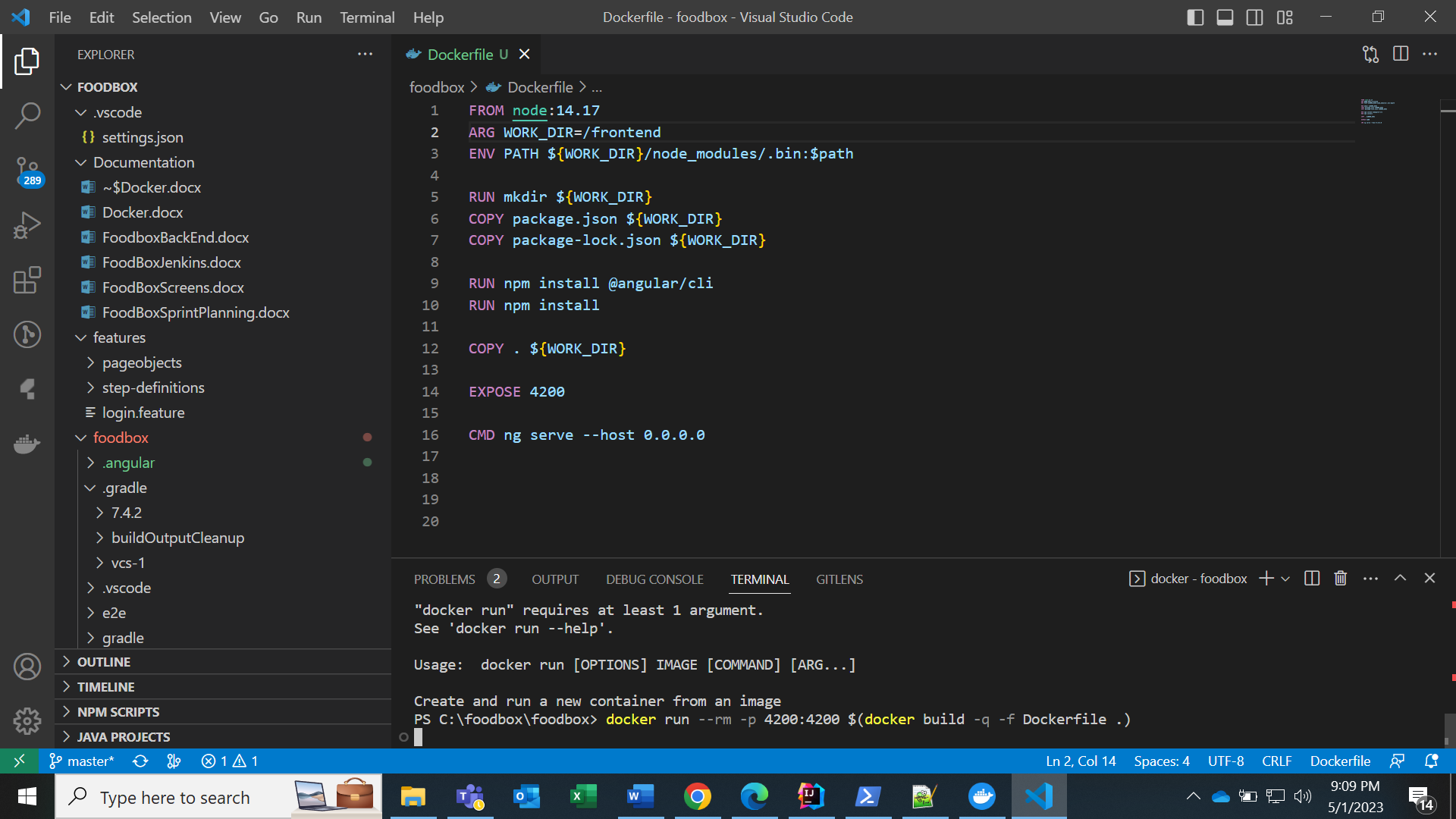
https://github.com/malinimurthy123/FoodBoxDevOps

Front End : [malinimurthy123/foodbox (github.com)](https://github.com/malinimurthy123/foodbox)

Back End : [malinimurthy123/foodbox at FoodBoxBackend (github.com)](https://github.com/malinimurthy123/foodbox/tree/FoodBoxBackend)

My Sql, Back End ER Diagram:





Front end sample code:

1. admin-page
2. admin-products
3. change-password
4. product-form
5. admindashboard
6. login
7. manage-customers
8. nav-bar
9. products
10. search
11. shopping-cart
12. order-success
13. user-login

**Admin:**

1. Admin Login:

admin-login.component.html

<br>

<form class="container col-md-6 mt-5" novalidate (ngSubmit) = "save(regForm)" #regForm = "ngForm">

    <p class = "display-4 lead" style="font-family: Hey October;">Admin Login</p>

      <div class="form-group">

        <label for="emailId" class="lead" style="font-family: Hey October;">Enter Email-id: &nbsp;</label>

        <input [(ngModel)] = adlogin.emailId #EmailIdVar = "ngModel" name="emailId"  id = "emailId" type="text" minlength="4" class="form-control" required

        [ngClass] = "{'is-invalid' : (EmailIdVar.touched || EmailIdVar.dirty) && !EmailIdVar.valid}" pattern="^[a-z0-9.\_%+-]+@[a-z0-9.-]+.[a-z]{2,4}$"/>

        <span class="invalid-feedback">

            <span \*ngIf = EmailIdVar.errors>

                Email-id required

            </span>

            <span \*ngIf = EmailIdVar.errors>

                Enter a valid email

            </span>

        </span>

    </div>

      <div class="form-group">

        <Br>

        <label for="password" class="lead" style="font-family: Hey October;">Enter Password: </label>

        <input [(ngModel)] = adlogin.password #passwordVar = "ngModel" name="password"  id = "password" type="password"  class="form-control" required minlength="5"

        [ngClass] = "{'is-invalid' : (passwordVar.touched || passwordVar.dirty) && !passwordVar.valid}"/>

        <span class="invalid-feedback">

            <span \*ngIf = passwordVar.errors>

                Password required

            </span>

        </span>

    </div>

    <Br>

      <button type="submit" class="btn btn-success"  [disabled] = '!regForm.valid' (click) = "Gotopage(regForm)" >Login</button>

  </form>

**admin-login.component.spec.ts**

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

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

describe('AdminLoginComponent', () => {

  let component: AdminLoginComponent;

  let fixture: ComponentFixture<AdminLoginComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AdminLoginComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AdminLoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

**admin-login.component.ts**

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

import { NgForm } from '@angular/forms';

import { SearchService } from 'src/app/search.service';

import { adminLogin } from './adminlogin';

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

@Component({

  selector: 'app-admin-login',

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

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

})

export class AdminLoginComponent implements OnInit {

  adlogin = new adminLogin();

  constructor(private router: Router){

  }

  ngOnInit(): void {

  }

  Gotopage(userForm : NgForm)

  {

    if(this.adlogin.emailId == "admin@gmail.com" && this.adlogin.password == "kitchenadmin"){

      if(window.confirm("Do you want to visit Admin Page?")){

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

      }

      else{

        userForm.reset();

        console.log(this.adlogin.password);

      }

    }

    else{

      userForm.reset();

      alert("Invalid EmailId or Password");

    }

  }

  save(userForm : NgForm){

  }

}

**adminlogin.ts**

export class adminLogin {

    [x: string]: any;

    constructor(

        public emailId = '',

        public password = '',

    ) {

    }

}

**Back end sample code:**

package foodboxspringb;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class FoodboxspringbApplicationTests {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(FoodboxspringbApplicationTests.class, args);  
 }  
  
}

package foodboxspringb.config;  
  
import org.springframework.core.Ordered;  
import org.springframework.core.annotation.Order;  
import org.springframework.stereotype.Component;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import javax.servlet.Filter;  
import javax.servlet.FilterChain;  
import javax.servlet.FilterConfig;  
import javax.servlet.ServletException;  
import javax.servlet.ServletRequest;  
import javax.servlet.ServletResponse;  
  
@Component  
@Order(Ordered.*HIGHEST\_PRECEDENCE*)  
public class ConfigCtrl implements Filter {  
 @Override  
 public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException, ServletException {  
 final HttpServletResponse response = (HttpServletResponse) res;  
 response.setHeader("Access-Control-Allow-Origin", "\*");  
 response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");  
 response.setHeader("Access-Control-Allow-Headers", "Authorization, Content-Type");  
 response.setHeader("Access-Control-Max-Age", "3600");  
 if ("OPTIONS".equalsIgnoreCase(((HttpServletRequest) req).getMethod())) {  
 response.setStatus(HttpServletResponse.*SC\_OK*);  
 } else {  
 chain.doFilter(req, res);  
 }  
 }  
 @Override  
 public void destroy() {  
 }  
 @Override  
 public void init(FilterConfig config) throws ServletException {  
 }  
}

package foodboxspringb.controller;  
  
import foodboxspringb.model.Admin;  
import foodboxspringb.repository.AdminRepository;  
import javax.servlet.http.HttpSession;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.Map;  
  
@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")  
@RestController  
public class AdminController {  
  
 @Autowired  
 private AdminRepository adminRepository;  
  
 @SuppressWarnings("rawtypes")  
 @PostMapping("/admin/{username}")  
 public boolean verifyAdminLogin(@RequestBody Map loginData, @PathVariable(name = "username") String username, HttpSession session) {  
 String lusername=(String) loginData.get("username");  
 String lpassword=(String) loginData.get("password");  
 Admin admin = adminRepository.findByusername(username);  
  
 if(admin!=null && admin.getUsername().equals(lusername) && admin.getPassword().equals(lpassword)) {  
 session.setAttribute("adminUsername", lusername);  
 return true;  
 }else {  
 return false;  
 }  
 }  
}

package foodboxspringb.controller;  
  
import foodboxspringb.model.Cart;  
import foodboxspringb.repository.CartRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import javax.servlet.http.HttpSession;  
import java.util.List;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
@CrossOrigin(origins = "http://localhost:4200")  
@RestController  
public class CartController {  
 @Autowired  
 private CartRepository cartRepository;  
  
 @CrossOrigin(origins = "http://localhost:4200")  
 @PostMapping("/carts")  
 public Cart addToCart(@RequestBody Cart cart, HttpSession session) {  
 float grandTotal=0;  
 if(session.getAttribute("grandTotal")==null) {  
 grandTotal=0;  
 }  
 else {  
 grandTotal=(float) session.getAttribute("grandTotal");  
 }  
 List<Cart> cartList = cartRepository.findAll();  
 for(Cart temp:cartList) {  
 if(temp.getProduct().getId()==cart.getProduct().getId()) {  
 int tempQuantity = 1+temp.getQuantity();  
 grandTotal=grandTotal+temp.getPrice();  
 session.setAttribute("grandTotal", grandTotal);  
 temp.setQuantity(tempQuantity);  
 temp.setPrice((temp.getProduct().getPrice()\*tempQuantity));  
 return cartRepository.save(temp);  
 }  
 }  
 int min = 50;  
 int max = 899;  
 int b = (int) (Math.*random*() \* (max - min + 1) + min);  
 cart.setId(b);  
 cart.setQuantity(1);  
 cart.setPrice(cart.getProduct().getPrice());  
 grandTotal=grandTotal+cart.getProduct().getPrice();  
 session.setAttribute("grandTotal", grandTotal);  
 return cartRepository.save(cart);  
 }  
 @GetMapping("/carts")  
 public List<Cart> getCartItems() {  
 return cartRepository.findAll();  
 }  
  
 @PutMapping("/carts/add/{id}")  
 public ResponseEntity<Cart> addByOne(@PathVariable("id") long id,@RequestBody Cart cart){  
  
 int quantity= cart.getQuantity()+1;  
 cart.setQuantity(quantity);  
 cart.setPrice((cart.getProduct().getPrice())\*quantity);  
 Cart updatedCart = cartRepository.save(cart);  
 return ResponseEntity.*ok*(updatedCart);  
 }  
 @PutMapping("/carts/minus/{id}")  
 public ResponseEntity<Cart> lessByOne(@PathVariable("id") long id,@RequestBody Cart cart){  
  
 int quantity= cart.getQuantity()-1;  
 if(quantity!=0) {  
 cart.setQuantity(quantity);  
 cart.setPrice((cart.getProduct().getPrice())\*quantity);  
 Cart updatedCart = cartRepository.save(cart);  
 return ResponseEntity.*ok*(updatedCart);  
 }else {  
 cartRepository.deleteById(id);  
 return new ResponseEntity<>(HttpStatus.*OK*);  
 }  
 }  
 @DeleteMapping("/carts/{id}")  
 public ResponseEntity<?> deleteCart(@PathVariable("id") Long id)  
 {  
 cartRepository.deleteById(id);  
 return new ResponseEntity<>(HttpStatus.*OK*);  
 }  
 @DeleteMapping("/carts")  
 public void deleteAllCart(){  
 cartRepository.deleteAll();  
 }  
}

package foodboxspringb.controller;  
  
import foodboxspringb.model.Customer;  
import foodboxspringb.repository.CustomerRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import javax.servlet.http.HttpSession;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")  
@RestController  
public class CustomerController {  
  
 @Autowired  
 private CustomerRepository customerRepository;  
 @CrossOrigin(origins = "http://localhost:4200")  
 @PostMapping("/customers")  
 public Customer addCustomer(@RequestBody Customer customer, HttpSession session) {  
 session.setAttribute("custemail", customer.getEmail());  
 return customerRepository.save(customer);  
 }  
  
 @SuppressWarnings("rawtypes")  
 @CrossOrigin(origins = "http://localhost:4200")  
 @PostMapping("/customers/{email}")  
 public boolean verifyLogin(@RequestBody Map loginData, @PathVariable(name = "email") String email, HttpSession session) {  
 String lemail = (String) loginData.get("email");  
 String lpassword = (String) loginData.get("password");  
 Customer customer = customerRepository.findByEmail(email);  
 if(customer!= null && customer.getEmail().equals(lemail) && customer.getPassword().equals(lpassword)) {  
 session.setAttribute("custemail", lemail);  
 return true;  
 }else {  
 return false;  
 }  
 }  
  
 @GetMapping("/customers")  
 public List<Customer> getAllCustomers(){  
 return customerRepository.findAll();  
 }  
  
 @GetMapping("/customers/search/{keyword}")  
 public List<Customer> searchCustomer(@PathVariable String keyword){  
 return customerRepository.searchCustomer(keyword);  
 }  
  
 @DeleteMapping("/customers/{email}")  
 public ResponseEntity<Map<String, Boolean>> deleteCustomer(@PathVariable String email){  
 Customer customer = customerRepository.findByEmail(email);  
 customerRepository.delete(customer);  
 Map<String, Boolean> map = new HashMap<>();  
 map.put("deleted", Boolean.*TRUE*);  
 return ResponseEntity.*ok*(map);  
 }  
 @GetMapping("/customers/{email}")  
 public Customer getCustomer(@PathVariable String email) {  
 return customerRepository.findByEmail(email);  
 }  
}

package foodboxspringb.controller;  
  
import foodboxspringb.exception.ResourceNotFoundException;  
import foodboxspringb.model.Admin;  
import foodboxspringb.model.Product;  
import foodboxspringb.repository.AdminRepository;  
import foodboxspringb.repository.ProductRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
@CrossOrigin(origins = "http://localhost:4200")  
@RestController  
public class ProductController {  
  
 @Autowired  
 private ProductRepository productRepository;  
 @Autowired AdminRepository adminRepository;  
  
 @GetMapping("/products")  
 public List<Product> getAllProducts(){  
 return productRepository.findAll();  
 }  
 @GetMapping("/products/Admin")  
 public List<Product> getAdminProducts() {  
 return productRepository.findAll();  
 }  
  
 @GetMapping("/products/cust")  
 public List<Product> getAllProducts1() {  
 List<Product> prodList=productRepository.findIfAvail();  
 if(prodList.isEmpty()) {  
 List<Admin> adminList = adminRepository.findAll();  
 if(adminList.isEmpty()) {  
 adminRepository.save(new Admin("admin","password"));  
 }  
 addProdIfEmpty(new Product(1,"Butter Tofu","Tofu infused with butter and spices","Indian",350,0,0,"yes","./assets/images/ButterChicken.png"));  
 addProdIfEmpty(new Product(2,"Biryani","Rice Steamed with Vegetables and spices","Indian",365,10,0,"yes","./assets/images/biryani.jpg"));  
 addProdIfEmpty(new Product(3,"Steamed Mince Burger","Steamed Bun with Veggie mince","Chinese",250,20,0,"yes","./assets/images/buns.jpg"));  
 addProdIfEmpty(new Product(4,"Fried Rice","Rice with Egg and Chinese sauses","Chinese",95,5,0,"yes","./assets/images/EggfriedRice.jpg"));  
 addProdIfEmpty(new Product(2,"Paneer Pizza","Pizza topped with cotted cheese and vegies","Italian",435,0,0,"yes","./assets/images/paneerpizza.jpg"));  
 addProdIfEmpty(new Product(2,"Red Sause Pasta","Pasta with Tomato and oregano","Italian",435,0,0,"yes","./assets/images/redPasta.jpg"));  
 addProdIfEmpty(new Product(2,"Ravioli","Ravioli pasta filled with veg mince","Italian",200,18,0,"yes","./assets/images/ravioli.jpg"));  
 addProdIfEmpty(new Product(2,"Elote de Corn","Corn topped with cream cheese and spice","Mexican",180,7,0,"yes","./assets/images/elote.jpg"));  
 addProdIfEmpty(new Product(2,"Burrito","Wrapped Tortilla with mince and Mayo","Mexican",350,0,0,"yes","./assets/images/Burrito.jpg"));  
 prodList=productRepository.findIfAvail();  
 }  
 return prodList;  
 }  
  
 public void addProdIfEmpty(Product product) {  
 int min = 10000;  
 int max = 99999;  
 int b = (int) (Math.*random*() \* (max - min + 1) + min);  
 product.setId(b);  
 float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);  
 float price = product.getActualPrice() - temp;  
 product.setPrice(price);  
 productRepository.save(product);  
 }  
  
 @PostMapping("/products")  
 public Product addProduct(@RequestBody Product product) {  
 int min = 10000;  
 int max = 99999;  
 int b = (int) (Math.*random*() \* (max - min + 1) + min);  
 product.setId(b);  
 float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);  
 float price = product.getActualPrice() - temp;  
 product.setPrice(price);  
 return productRepository.save(product);  
 }  
  
 @PutMapping("/products/{id}")  
 public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product productDetails){  
 Product product = productRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));  
 product.setName(productDetails.getName());  
 product.setDesc(productDetails.getDesc());  
 product.setCategory(productDetails.getCategory());  
 product.setImagepath(productDetails.getImagepath());  
 product.setActualPrice(productDetails.getActualPrice());  
 product.setDiscount(productDetails.getDiscount());  
 product.setAvail(productDetails.getAvail());  
 float temp = (product.getActualPrice()) \* (product.getDiscount() / 100);  
 float price = product.getActualPrice() - temp;  
 product.setPrice(price);  
  
 Product updatedProd = productRepository.save(product);  
 return ResponseEntity.*ok*(updatedProd);  
  
 }  
  
 @DeleteMapping("/products/{id}")  
 public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id) {  
 Product product = productRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));  
 productRepository.delete(product);  
 Map<String, Boolean> map = new HashMap<>();  
 map.put("deleted", Boolean.*TRUE*);  
 return ResponseEntity.*ok*(map);  
 }  
  
 @GetMapping("products/{id}")  
 public ResponseEntity<Product> getProductById(@PathVariable long id) {  
 Product product = productRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException("Product Not Found with " + id));  
 return ResponseEntity.*ok*(product);  
 }  
  
 @GetMapping("products/search/{keyword}")  
 public List<Product> getSearchProducts(@PathVariable String keyword) {  
 return productRepository.homeSearch(keyword);  
 }  
  
 @GetMapping("products/chinese")  
 public List<Product> getChinese() {  
 return productRepository.getChinese();  
 }  
  
 @GetMapping("products/indian")  
 public List<Product> getIndian() {  
 return productRepository.getIndian();  
 }  
  
 @GetMapping("products/mexican")  
 public List<Product> getMexican() {  
 return productRepository.getMexican();  
 }  
  
 @GetMapping("products/italian")  
 public List<Product> getItalian() {  
 return productRepository.getItalian();  
 }  
}

package foodboxspringb.controller;  
  
import foodboxspringb.exception.ResourceNotFoundException;  
import foodboxspringb.model.Cart;  
import foodboxspringb.model.Customer;  
import foodboxspringb.model.Purchase;  
import foodboxspringb.repository.CartRepository;  
import foodboxspringb.repository.CustomerRepository;  
import foodboxspringb.repository.PurchaseRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "\*")  
@RestController  
public class PurchaseController {  
 @Autowired  
 private PurchaseRepository purchaseRepository;  
 @Autowired  
 private CartRepository cartRepository;  
 @Autowired  
 private CustomerRepository customerRepository;  
  
 @GetMapping("/purchase/byEmail/{email}")  
 public List<Purchase> customerOrders(@PathVariable String email) {  
 return purchaseRepository.getByEmail(email);  
 }  
  
 @GetMapping("/purchase")  
 public List<Purchase> getAllPurchase(){  
 return purchaseRepository.findAllByOrderByTransactionidAsc();  
 }  
  
 @DeleteMapping("/purchase/{id}")  
 public ResponseEntity<Map<String, Boolean>> deletePurchase(@PathVariable Long id) {  
 Purchase purchase = purchaseRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException("Purchase Id not found with "+id));  
 purchaseRepository.delete(purchase);  
 Map<String, Boolean> map = new HashMap<>();  
 map.put("deleted", Boolean.*TRUE*);  
 return ResponseEntity.*ok*(map);  
 }  
  
 @GetMapping("/purchase/search/{keyword}")  
 public List<Purchase> searchPurchase(@PathVariable String keyword){  
 return purchaseRepository.searchPurchase(keyword);  
 }  
  
 @SuppressWarnings("rawtypes")  
 @PostMapping("/purchase")  
 public ResponseEntity<Map<String, Boolean>> buyProducts(@RequestBody Map buyProdMap){  
 List<Cart> cartList = cartRepository.findAll();  
 Purchase purchase = new Purchase();  
 String cust\_email=(String)buyProdMap.get("email");  
 Customer customer = customerRepository.findByEmail(cust\_email);  
 String transId = (String)buyProdMap.get("transactionId");  
 for(Cart cl:cartList) {  
 java.sql.Date date = new java.sql.Date(new java.util.Date().getTime());  
 long min=100000;long max=999999;long b = (long)(Math.*random*()\*(max-min+1)+min);  
 purchase.setId(b);  
 purchase.setDop(date);  
 purchase.setCustomer(customer);  
 String name = cl.getProduct().getName();  
 purchase.setProductname(name);  
 purchase.setQuantity(cl.getQuantity());  
 purchase.setTotalcost(cl.getPrice());  
 purchase.setTransactionid(transId);  
 purchaseRepository.save(purchase);  
 }  
 Map<String, Boolean> map = new HashMap<>();  
 map.put("created",Boolean.*TRUE*);  
 return ResponseEntity.*ok*(map);  
 }  
}

package foodboxspringb.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value=HttpStatus.*NOT\_FOUND*)  
public class ResourceNotFoundException extends RuntimeException{  
 private static final long *serialVersionUID* = 1L;  
  
 public ResourceNotFoundException(String message) {  
 super(message);  
 }  
  
}

package foodboxspringb.repository;  
  
import foodboxspringb.model.Purchase;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import java.util.List;  
  
public interface PurchaseRepository extends JpaRepository<Purchase, Long> {  
 @Query("Select p FROM Purchase p WHERE p.customer.email LIKE %?1%")  
 public List<Purchase> getByEmail(String email);  
  
 public List<Purchase> findAllByOrderByTransactionidAsc();  
  
 @Query("Select p FROM Purchase p WHERE p.transactionid LIKE %?1%"  
 +" OR p.productname LIKE %?1%"  
 +" OR p.customer.name LIKE %?1%")  
 public List<Purchase> searchPurchase(String keyword);  
}

package foodboxspringb.repository;  
  
import foodboxspringb.model.Product;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import java.util.List;  
  
public interface ProductRepository extends JpaRepository<Product, Long>{  
  
 @Query("Select p FROM Product p WHERE p.avail='yes' ORDER BY 'category'")  
 List<Product> findIfAvail();  
  
 @Query("SELECT p FROM Product p WHERE (p.avail LIKE 'yes') AND (p.name LIKE %?1%"  
 +" OR p.des LIKE %?1%"  
 +" OR p.category LIKE %?1%)")  
 public List<Product> homeSearch(String keyword);  
  
 @Query("SELECT p FROM Product p WHERE p.category LIKE 'Chinese' AND p.avail LIKE 'yes'")  
 public List<Product> getChinese();  
  
 @Query("SELECT p FROM Product p WHERE p.category LIKE 'Indian' AND p.avail LIKE 'yes'")  
 public List<Product> getIndian();  
  
 @Query("SELECT p FROM Product p WHERE p.category LIKE 'Mexican' AND p.avail LIKE 'yes'")  
 public List<Product> getMexican();  
  
 @Query("SELECT p FROM Product p WHERE p.category LIKE 'Italian' AND p.avail LIKE 'yes'")  
 public List<Product> getItalian();  
}

package foodboxspringb.repository;  
  
import foodboxspringb.model.Customer;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import java.util.List;  
  
public interface CustomerRepository extends JpaRepository<Customer, String> {  
 Customer findByEmail(String email);  
 @Query("SELECT c FROM Customer c WHERE c.email LIKE %?1%"  
 +" OR c.name LIKE %?1%"  
 +" OR c.contact LIKE %?1%"  
 +" OR c.address LIKE %?1%")  
 public List<Customer> searchCustomer(String keyword);  
}

package foodboxspringb.repository;  
import foodboxspringb.model.Admin;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import java.util.List;  
public interface AdminRepository extends JpaRepository<Admin, String>{  
  
 Admin findByusername(String username);  
  
}