FoodBox
(Project Source Code)

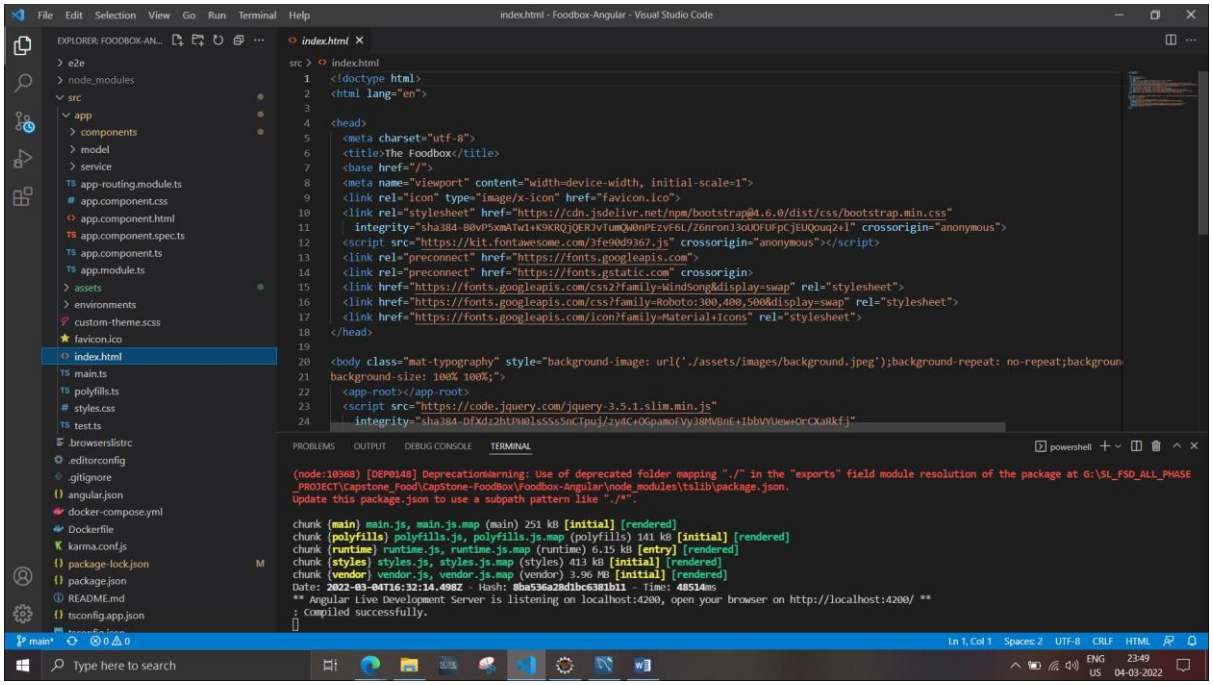
Version History:

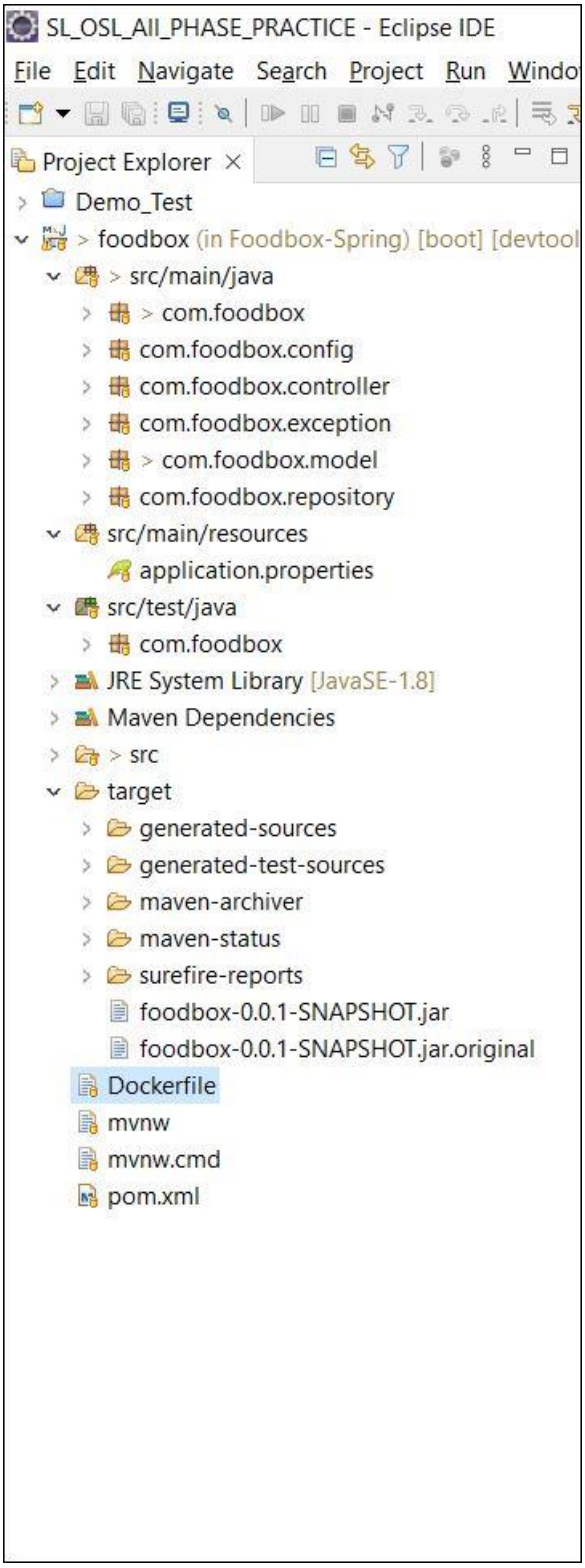
Author	Mayur Bharatkumar Solanki
Purpose	Source Code of the application
Date	28th February 2022
Version	1.0

- Project Link

Repository Name	**FoodBox**
GitHub Link	https://github.com/mursky66/SL_FSD_Capstone_Project.git

- Folder Structure




SL_OSL_All_PHASE_PRACTICE - Eclipse IDE
File Edit Navigate Search Project Run Windo
Project Explorer
Demo_Test
foodbox (in Foodbox-Spring) [boot] [devtool
src/main/java
com.foodbox
com.foodbox.config
com.foodbox.controller
com.foodbox.exception
com.foodbox.model
com.foodbox.repository
src/main/resources
application.properties
src/test/java
com.foodbox
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
foodbox-0.0.1-SNAPSHOT.jar
foodbox-0.0.1-SNAPSHOT.jar.original
Dockerfile
mvnw
mvnw.cmd
pom.xml

- Source Code

package com.foodbox;

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FoodboxSpringApplication {

	public static void main(String[] args) {
		SpringApplication.run(FoodboxSpringApplication.class, args);
	}

}
```

```
package com.foodbox.config;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

@Component
@Order(Ordered.HIGHEST_PRECEDENCE)
public class ConfigCtrl implements Filter {
    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {
        final HttpServletResponse response = (HttpServletResponse) res;
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");
        response.setHeader("Access-Control-Allow-Headers", "Authorization, Content-Type");
        response.setHeader("Access-Control-Max-Age", "3600");
        if ("OPTIONS".equalsIgnoreCase(((HttpServletRequest) req).getMethod())) {
            response.setStatus(HttpServletResponse.SC_OK);
```

```
        } else {
            chain.doFilter(req, res);
        }
    }
    @Override
    public void destroy() {
    }
    @Override
    public void init(FilterConfig config) throws ServletException {
    }
}
```

package com.foodbox.controller;[Admin]

```
import java.util.Map;

import javax.servlet.http.HttpSession;

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

import com.foodbox.model.Admin;
import com.foodbox.repository.AdminRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "*")
@RestController
public class AdminController {

	@Autowired
	private AdminRepository adminRepository;

	@SuppressWarnings("rawtypes")
	@PostMapping("/admin/{username}")
	public boolean verifyAdminLogin(@RequestBody Map loginData, @PathVariable(name
= "username") String username, HttpSession session) {
		String lusername=(String) loginData.get("username");
		String lpassword=(String) loginData.get("password");
```

```
            Admin admin = adminRepository.findByusername(username);
            if(admin!=null && admin.getUsername().equals(lusername) &&
admin.getPassword().equals(lpassword)) {
                session.setAttribute("adminUsername", lusername);
                return true;
            }else {
                return false;
            }
        }

}
```

package com.foodbox.controller; [Cart]

```
import java.util.List;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.foodbox.model.Cart;
import com.foodbox.repository.CartRepository;

@CrossOrigin(origins = "http://localhost:4200")
@RestController
public class CartController {

	@Autowired
	private CartRepository cartRepository;

	@CrossOrigin(origins = "http://localhost:4200")
	@PostMapping("/carts")
	public Cart addToCart(@RequestBody Cart cart, HttpSession session) {
		float grandTotal=0;
		if(session.getAttribute("grandTotal")==null) {
			grandTotal=0;
		}
		else {
			grandTotal=(float) session.getAttribute("grandTotal");
		}
		List<Cart> cartList = cartRepository.findAll();
		for(Cart temp:cartList) {
			if(temp.getProduct().getId()==cart.getProduct().getId()) {
				int tempQuantity = 1+temp.getQuantity();
				grandTotal=grandTotal+temp.getPrice();
				session.setAttribute("grandTotal", grandTotal);
				temp.setQuantity(tempQuantity);
				temp.setPrice((temp.getProduct().getPrice()*tempQuantity));
```

```
                return cartRepository.save(temp);
            }
        }
        int min = 100;
        int max = 999;
        int b = (int) (Math.random() * (max - min + 1) + min);
        cart.setId(b);
        cart.setQuantity(1);
        cart.setPrice(cart.getProduct().getPrice());
        grandTotal=grandTotal+cart.getProduct().getPrice();
        session.setAttribute("grandTotal", grandTotal);
        return cartRepository.save(cart);
    }

    @GetMapping("/carts")
    public List<Cart> getCartItems() {
        return cartRepository.findAll();
    }

    @PutMapping("/carts/add/{id}")
    public ResponseEntity<Cart> addByOne(@PathVariable("id") long id,@RequestBody
Cart cart){
        /*Cart cart = cartRepository.findById(id)
                .orElseThrow(()->new ResourceNotFoundException("Product not
found with"+id));*/
        int quantity= cart.getQuantity()+1;
        cart.setQuantity(quantity);
        cart.setPrice((cart.getProduct().getPrice())*quantity);
        Cart updatedCart = cartRepository.save(cart);
        return ResponseEntity.ok(updatedCart);
```

```
	}

	
	@PutMapping("/carts/minus/{id}")
	public ResponseEntity<Cart> lessByOne(@PathVariable("id") long id,@RequestBody Cart cart){
		/*
		 * Cart cart = cartRepository.findById(id) .orElseThrow(()->new
		 * ResourceNotFoundException("Product not found with"+id));
		 */
		int quantity= cart.getQuantity()-1;
		if(quantity!=0) {
			cart.setQuantity(quantity);
			cart.setPrice((cart.getProduct().getPrice())*quantity);
			Cart updatedCart = cartRepository.save(cart);
			return ResponseEntity.ok(updatedCart);
		}else {
			cartRepository.deleteById(id);
			return new ResponseEntity<>(HttpStatus.OK);
		}
	}

	
	@DeleteMapping("/carts/{id}")
	public ResponseEntity<?> deleteCart(@PathVariable("id") Long id)
	{
		cartRepository.deleteById(id);
		return new ResponseEntity<>(HttpStatus.OK);
	}

	
	@DeleteMapping("/carts")
	public void deleteAllCart(){
```

```
        cartRepository.deleteAll();
    }
}
```

package com.foodbox.controller; [Customer]

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Customer;
import com.foodbox.repository.CustomerRepository;

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

```
public class CustomerController {


	@Autowired
	private CustomerRepository customerRepository;

	
	@CrossOrigin(origins = "http://localhost:4200")
	@PostMapping("/customers")
	public Customer addCustomer(@RequestBody Customer customer, HttpSession session) {
		session.setAttribute("cust_email", customer.getEmail());
		return customerRepository.save(customer);
	}

	
	@SuppressWarnings("rawtypes")
	@CrossOrigin(origins = "http://localhost:4200")
	@PostMapping("/customers/{email}")
	public boolean verifyLogin(@RequestBody Map loginData, @PathVariable(name = "email") String email, HttpSession session) {
		String lemail = (String) loginData.get("email");
		String lpassword = (String) loginData.get("password");
		Customer customer = customerRepository.findByEmail(email);
		if(customer!= null && customer.getEmail().equals(lemail) && customer.getPassword().equals(lpassword)) {
			session.setAttribute("cust_email", lemail);
			return true;
		}else {
			return false;
		}
	}

	
```

```
	@GetMapping("/customers")
	public List<Customer> getAllCustomers(){
		return customerRepository.findAll();
	}

	@GetMapping("/customers/search/{keyword}")
	public List<Customer> searchCustomer(@PathVariable String keyword){
		return customerRepository.searchCustomer(keyword);
	}

	@DeleteMapping("/customers/{email}")
	public ResponseEntity<Map<String, Boolean>> deleteCustomer(@PathVariable String email){
		Customer customer = customerRepository.findByEmail(email);
		customerRepository.delete(customer);
		Map<String, Boolean> map = new HashMap<>();
		map.put("deleted", Boolean.TRUE);
		return ResponseEntity.ok(map);
	}

	@GetMapping("/customers/{cust_email}")
	public Customer getCustomer(@PathVariable String cust_email) {
		return customerRepository.findByEmail(cust_email);

	}
}
```

package com.foodbox.controller; [Product]

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Admin;
import com.foodbox.model.Product;
import com.foodbox.repository.AdminRepository;
import com.foodbox.repository.ProductRepository;

import com.foodbox.exception.ResourceNotFoundException;

@CrossOrigin(origins = "http://localhost:4200")
@RestController
public class ProductController {

	@Autowired
```

```
	private ProductRepository productRepository;

	@Autowired AdminRepository adminRepository;


	@GetMapping("/products/Admin")
	public List<Product> getAdminProducts() {
		return productRepository.findAll();
	}


	@GetMapping("/products/cust")
	public List<Product> getAllProducts() {
		List<Product> prodList=productRepository.findIfAvail();
		if(prodList.isEmpty()) {
			List<Admin> adminList = adminRepository.findAll();
			if(adminList.isEmpty()) {
				adminRepository.save(new Admin("admin","password"));
			}
			addProdIfEmpty(new Product(1,"Butter Chicken","Chicken infused with butter and spices","Indian",350,0,0,"yes","./assets/images/ButterChicken.png"));
			addProdIfEmpty(new Product(2,"Chicken Biryani","Rice Steamed with Chicken and spices","Indian",365,10,0,"yes","./assets/images/biryani.jpg"));
			addProdIfEmpty(new Product(3,"Steamed Mince Bun","Steamed Bun with lamb mince","Chinese",250,20,0,"yes","./assets/images/buns.jpg"));
			addProdIfEmpty(new Product(4,"Egg Fried Rice","Rice with Egg and Chinese sauses","Chinese",95,5,0,"yes","./assets/images/EggfriedRice.jpg"));
			addProdIfEmpty(new Product(2,"Paneer Pizza","Pizza topped with cotted cheese and vegies","Italian",435,0,0,"yes","./assets/images/paneerpizza.jpg"));
			addProdIfEmpty(new Product(2,"Red Sause Pasta","Pasta with Tomato and oregano","Italian",435,0,0,"yes","./assets/images/redPasta.jpg"));
			addProdIfEmpty(new Product(2,"Ravioli","Ravioli pasta filled with veg mince","Italian",200,18,0,"yes","./assets/images/ravioli.jpg"));
```

```
                addProdIfEmpty(new Product(2,"Elote de Corn","Corn topped with cream cheese and spice","Mexican",180,7,0,"yes","./assets/images/elote.jpg"));
                addProdIfEmpty(new Product(2,"Burrito","Wrapped Tortilla with Meat mince and Mayo","Mexican",350,0,0,"yes","./assets/images/Burrito.jpg"));
                prodList=productRepository.findIfAvail();
            }
            return prodList;
        }

    public void addProdIfEmpty(Product product) {
            int min = 10000;
            int max = 99999;
            int b = (int) (Math.random() * (max - min + 1) + min);
            product.setId(b);
            float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
            float price = product.getActualPrice() - temp;
            product.setPrice(price);
            productRepository.save(product);
        }

    @PostMapping("/products")
    public Product addProduct(@RequestBody Product product) {
            int min = 10000;
            int max = 99999;
            int b = (int) (Math.random() * (max - min + 1) + min);
            product.setId(b);
            float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
            float price = product.getActualPrice() - temp;
            product.setPrice(price);
            return productRepository.save(product);
```

```
    }

    @PutMapping("/products/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable Long id,
@RequestBody Product productDetails){
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Employee
Not Found with " + id));
        product.setName(productDetails.getName());
        product.setDesc(productDetails.getDesc());
        product.setCategory(productDetails.getCategory());
        product.setImagepath(productDetails.getImagepath());
        product.setActualPrice(productDetails.getActualPrice());
        product.setDiscount(productDetails.getDiscount());
        product.setAvail(productDetails.getAvail());
        float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
        float price = product.getActualPrice() - temp;
        product.setPrice(price);

        Product updatedProd = productRepository.save(product);
        return ResponseEntity.ok(updatedProd);

    }


    @DeleteMapping("/products/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id) {
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Employee
Not Found with " + id));
        productRepository.delete(product);
```

```
        Map<String, Boolean> map = new HashMap<>();
        map.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(map);
    }

    @GetMapping("products/{id}")
    public ResponseEntity<Product> getProductById(@PathVariable long id) {
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Product
Not Found with " + id));
        return ResponseEntity.ok(product);
    }

    @GetMapping("products/search/{keyword}")
    public List<Product> getSearchProducts(@PathVariable String keyword) {
        return productRepository.homeSearch(keyword);
    }

    @GetMapping("products/chinese")
    public List<Product> getChinese() {
        return productRepository.getChinese();
    }

    @GetMapping("products/indian")
    public List<Product> getIndian() {
        return productRepository.getIndian();
    }

    @GetMapping("products/mexican")
    public List<Product> getMexican() {
```

```
        return productRepository.getMexican();
    }

    @GetMapping("products/italian")
    public List<Product> getItalian() {
        return productRepository.getItalian();
    }
}
```

package com.foodbox.controller;[Purchase]

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.ResourceNotFoundException;
import com.foodbox.model.Cart;
```

```
import com.foodbox.model.Customer;
import com.foodbox.model.Purchase;
import com.foodbox.repository.CartRepository;
import com.foodbox.repository.CustomerRepository;
import com.foodbox.repository.PurchaseRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "*")
@RestController
public class PurchaseController {

	@Autowired
	private PurchaseRepository purchaseRepository;

	@Autowired
	private CartRepository cartRepository;

	@Autowired
	private CustomerRepository customerRepository;

	@GetMapping("/purchase/byEmail/{email}")
	public List<Purchase> customerOrders(@PathVariable String email) {
		return purchaseRepository.getByEmail(email);
	}

	@GetMapping("/purchase")
	public List<Purchase> getAllPurchase(){
		return purchaseRepository.findAllByOrderByTransactionidAsc();
	}

	
```

```
	@DeleteMapping("/purchase/{id}")
	public ResponseEntity<Map<String, Boolean>> deletePurchase(@PathVariable Long id)
{
		Purchase purchase = purchaseRepository.findById(id)
				.orElseThrow(() -> new ResourceNotFoundException("Purchase
Id not found with "+id));
		purchaseRepository.delete(purchase);
		Map<String, Boolean> map = new HashMap<>();
		map.put("deleted", Boolean.TRUE);
		return ResponseEntity.ok(map);
	}

	@GetMapping("/purchase/search/{keyword}")
	public List<Purchase> searchPurchase(@PathVariable String keyword){
		return purchaseRepository.searchPurchase(keyword);
	}

	@SuppressWarnings("rawtypes")
	@PostMapping("/purchase")
	public ResponseEntity<Map<String, Boolean>> buyProducts(@RequestBody Map
buyProdMap){
		List<Cart> cartList = cartRepository.findAll();
		Purchase purchase = new Purchase();
		String cust_email=(String)buyProdMap.get("email");
		Customer customer = customerRepository.findByEmail(cust_email);
		String transId = (String)buyProdMap.get("transactionId");
		for(Cart cl:cartList) {
			java.sql.Date date = new java.sql.Date(new java.util.Date().getTime());
			long min=100000;long max=999999;long b =
(long)(Math.random()*(max-min+1)+min);
			purchase.setId(b);
```

```
                    purchase.setDop(date);
                    purchase.setCustomer(customer);
                    String name = cl.getProduct().getName();
                    purchase.setProductname(name);
                    purchase.setQuantity(cl.getQuantity());
                    purchase.setTotalcost(cl.getPrice());
                    purchase.setTransactionid(transId);
                    purchaseRepository.save(purchase);
                }
                Map<String, Boolean> map = new HashMap<>();
                map.put("created",Boolean.TRUE);
                return ResponseEntity.ok(map);
        }
}
```

```
package com.foodbox.exception;

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


@ResponseStatus(value=HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException{
	private static final long serialVersionUID = 1L;
	
	public ResourceNotFoundException(String message) {
		super(message);
	}

}
```

package com.foodbox.model;[Admin]

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Admin {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private Long userId;
	private String username;
	private String password;

	public Admin() {
		super();
	}

	public Admin(String username, String password) {
		super();
		this.username = username;
		this.password = password;
	}

	public String getUsername() {
		return username;
	}
```

```
	public void setUsername(String username) {
		this.username = username;
	}

	public String getPassword() {
		return password;
	}

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

}
```

package com.foodbox.model;[Cart]

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToOne;

@Entity
public class Cart {
	@Id
	private long id;
	private int quantity;
	private float price;
	@OneToOne
	private Product product;
	
```

```
	public Cart(long id, int quantity, float price, Product product) {
		super();
		this.id = id;
		this.quantity = quantity;
		this.price = price;
		this.product = product;
	}

	public Cart() {
		super();
	}

	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}
	public float getPrice() {
		return price;
```

```
	}
	public void setPrice(float price) {
		this.price = price;
	}
	public Product getProduct() {
		return product;
	}
	public void setProduct(Product product) {
		this.product = product;
	}
	
	
	
	
	
	
}
```

package com.foodbox.model;[Customer]

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Customer {
	@Id
```

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

	private String email;
	private String password;
	private String name;
	private String contact;
	private String address;

	public Customer(String email, String password, String name, String contact, String address) {
		super();
		this.email = email;
		this.password = password;
		this.name = name;
		this.contact = contact;
		this.address = address;
	}

	public Customer() {
		super();
	}

	public String getEmail() {
		return email;
	}

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

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

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

	public String getName() {
		return name;
	}

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

	public String getContact() {
		return contact;
	}

	public void setContact(String contact) {
		this.contact = contact;
	}

	public String getAddress() {
		return address;
	}
```

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

}
```

package com.foodbox.model; [Product]

```
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Product {
	@Id
	private long id;
	private String name;
	private String des;
	private String category;
	private float actualPrice;
	private float discount;
	private float price;
	private String avail;
	private String imagepath;
	

	public Product(long id, String name, String des, String category, float actualPrice, float discount, float price,
			String avail, String imagepath) {
		super();
```

```
                this.id = id;
                this.name = name;
                this.des = des;
                this.category = category;
                this.actualPrice = actualPrice;
                this.discount = discount;
                this.price = price;
                this.avail = avail;
                this.imagepath = imagepath;
        }
        public Product() {
                super();
        }

        public long getId() {
                return id;
        }

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

        public String getName() {
                return name;
        }

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

```
	public String getDesc() {
		return des;
	}

	public void setDesc(String des) {
		this.des = des;
	}

	public String getCategory() {
		return category;
	}

	public void setCategory(String category) {
		this.category = category;
	}

	public float getActualPrice() {
		return actualPrice;
	}

	public void setActualPrice(float actualPrice) {
		this.actualPrice = actualPrice;
	}

	public float getDiscount() {
		return discount;
	}
```

```
	public void setDiscount(float discount) {
		this.discount = discount;
	}

	public float getPrice() {
		return price;
	}

	public void setPrice(float price) {
		this.price = price;
	}

	public String getAvail() {
		return avail;
	}

	public void setAvail(String avail) {
		this.avail = avail;
	}

	public String getImagepath() {
		return imagepath;
	}

	public void setImagepath(String imagepath) {
		this.imagepath = imagepath;
	}

}
```

package com.foodbox.model;[Purchase]

```
import java.sql.Date;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToOne;
@Entity
public class Purchase {
	@Id
	private long id;
	private float totalcost;
	private Date dop;
	private int quantity;
	private String productname;
	private String transactionid;
	@OneToOne
	private Customer customer;

	public Purchase() {
		super();
	}
	public Purchase(long id, float totalcost, Date dop, int quantity, String productname,
String transactionid,
			Customer customer) {
		super();
		this.id = id;
		this.totalcost = totalcost;
		this.dop = dop;
		this.quantity = quantity;
```

```
        this.productname = productname;
        this.transactionid = transactionid;
        this.customer = customer;
    }

    public long getId() {
        return id;
    }

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

    public float getTotalcost() {
        return totalcost;
    }

    public void setTotalcost(float totalcost) {
        this.totalcost = totalcost;
    }

    public Date getDop() {
        return dop;
    }

    public void setDop(Date dop) {
        this.dop = dop;
    }
```

```
	public int getQuantity() {
		return quantity;
	}

	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}

	public String getProductname() {
		return productname;
	}

	public void setProductname(String productname) {
		this.productname = productname;
	}

	public String getTransactionid() {
		return transactionid;
	}
	public void setTransactionid(String transactionid) {
		this.transactionid = transactionid;
	}
	public Customer getCustomer() {
		return customer;
	}
	public void setCustomer(Customer customer) {
		this.customer = customer;
	}
}
```

package com.foodbox.repository;[Admin]

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

import com.foodbox.model.Admin;

public interface AdminRepository extends JpaRepository<Admin, String>{

	Admin findByusername(String username);

}
```

package com.foodbox.repository;[Cart]

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

import com.foodbox.model.Cart;

public interface CartRepository extends JpaRepository<Cart, Long> {

}
```

package com.foodbox.repository;[Customer]

```
import java.util.List;

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

import com.foodbox.model.Customer;

public interface CustomerRepository extends JpaRepository<Customer, String> {
	Customer findByEmail(String email);
	
	@Query("SELECT c FROM Customer c WHERE c.email LIKE %?1%"
			+" OR c.name LIKE %?1%"
			+" OR c.contact LIKE %?1%"
			+" OR c.address LIKE %?1%")
	public List<Customer> searchCustomer(String keyword);
}
```

package com.foodbox.repository;[Product]

```
import java.util.List;

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

import com.foodbox.model.Product;
```

```
public interface ProductRepository extends JpaRepository<Product, Long>{

	@Query("Select p FROM Product p WHERE p.avail='yes' ORDER BY 'category'")
	List<Product> findIfAvail();

	@Query("SELECT p FROM Product p WHERE (p.avail LIKE 'yes') AND (p.name LIKE %?1%"
			+" OR p.des LIKE %?1%"
			+" OR p.price LIKE %?1%"
			+" OR p.category LIKE %?1%)")
	public List<Product> homeSearch(String keyword);

	@Query("SELECT p FROM Product p WHERE p.category LIKE 'Chinese' AND p.avail LIKE 'yes'")
	public List<Product> getChinese();

	@Query("SELECT p FROM Product p WHERE p.category LIKE 'Indian' AND p.avail LIKE 'yes'")
	public List<Product> getIndian();

	@Query("SELECT p FROM Product p WHERE p.category LIKE 'Mexican' AND p.avail LIKE 'yes'")
	public List<Product> getMexican();

	@Query("SELECT p FROM Product p WHERE p.category LIKE 'Italian' AND p.avail LIKE 'yes'")
	public List<Product> getItalian();
}
```

package com.foodbox.repository;[Purchase]

```
import java.util.List;

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

import com.foodbox.model.Purchase;

public interface PurchaseRepository extends JpaRepository<Purchase, Long> {
	@Query("Select p FROM Purchase p WHERE p.customer.email LIKE %?1%")
	public List<Purchase> getByEmail(String email);
	
	public List<Purchase> findAllByOrderByTransactionidAsc();
	
	@Query("Select p FROM Purchase p WHERE p.transactionid LIKE %?1%"
			+" OR p.dop LIKE %?1%"
			+" OR p.productname LIKE %?1%"
			+" OR p.customer LIKE %?1%")
	public List<Purchase> searchPurchase(String keyword);
}
```

Application.Properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/foodbox
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.jpa.show-sql = true
logging.level.org.springframework = info
spring.jpa.hibernate.ddl-auto = update
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.6.3</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>foodbox</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Foodbox-Spring</name>
	<description>FoodBox Spring Boot</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-rest</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<scope>runtime</scope>
		</dependency>

		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
```

```
                <scope>test</scope>
            </dependency>

            <dependency>
                <groupId>org.springframework.restdocs</groupId>
                <artifactId>spring-restdocs-mockmvc</artifactId>
                <scope>test</scope>
            </dependency>

        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.asciidoctor</groupId>
                    <artifactId>asciidoctor-maven-plugin</artifactId>
                    <version>1.5.8</version>
                    <executions>
                        <execution>
                            <id>generate-docs</id>
                            <phase>prepare-package</phase>
                            <goals>
                                <goal>process-asciidoc</goal>
                            </goals>
                            <configuration>
                                <backend>html</backend>
                                <doctype>book</doctype>
                            </configuration>
                        </execution>
                    </executions>
                    <dependencies>
                        <dependency>

        <groupId>org.springframework.restdocs</groupId>
                            <artifactId>spring-restdocs-
asciidoctor</artifactId>
                            <version>${spring-
restdocs.version}</version>
                        </dependency>
                    </dependencies>
                </plugin>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                    <configuration>
                        <excludes>
                            <exclude>
                                <groupId>org.projectlombok</groupId>
                                <artifactId>lombok</artifactId>
                            </exclude>
                        </excludes>
                    </configuration>
                </plugin>
            </plugins>
        </build>

    </project>
```