File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\services\ProductService.java

```java
package com.sportshoes.ecom.services;

import com.sportshoes.ecom.entity.Category;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.exceptions.
ProductNotFoundException;
import com.sportshoes.ecom.repos.ProductRepo;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.stream.Collectors;

@Component
public class ProductService {
    private final ProductRepo productRepo;

    @Autowired
    public ProductService(ProductRepo productRepo) {
        this.productRepo = productRepo;
    }

    public Products addNewProduct(Products product) {
        return productRepo.save(product);
    }

    public Products findProductById(long id) {
        Products products = this.productRepo.findById
(id)
                .orElseThrow(()-> new
ProductNotFoundException("product with ID " + id + "
not found"));
       /* if(products.isDeletedFlag())
            return null;*/
        return products;
    }

    public List<Products> getAllProducts() {
        return this.productRepo.findAll();
    }

    public List<Products> getProductsByCategory(Long
id) {
```

```
39          return this.productRepo.findAllByCategoryId(
   new Category(id));/*.stream()
40                  .filter(a-> !a.isDeletedFlag()).
   collect(Collectors.toList());*/
41      }
42
43      public void deleteProduct(Long id) {
44          System.out.println("deleting product + "+ id
   );
45          /*Products product = this.productRepo.
   findById(id).orElseThrow(()->
46                  new ProductNotFoundException("product
    with ID " + id + " not found"));
47          if(!product.isDeletedFlag()) {
48              product.setDeletedFlag(true);
49              this.productRepo.saveAndFlush(product);
50          }*/
51          this.productRepo.softDeleteProduct(id);
52      }
53
54      public int getProductPrice(Long id) {
55          return this.productRepo.getProductPrice(id);
56      }
57
58      public boolean findProduct(Long id) {
59          if(this.findProductById(id) == null)
60              return false;
61          return this.productRepo.existsById(id);
62      }
63
64  }
65
```

```java
package com.sportshoes.ecom.services;

import com.sportshoes.ecom.entity.Category;
import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.exceptions.ProductNotFoundException;
import com.sportshoes.ecom.repos.CategoryRepo;
import com.sportshoes.ecom.security.ApplicationUserDetails;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.RequestBody;

import javax.transaction.Transactional;
import java.util.List;
import java.util.Optional;

@Component
public class CategoryService {
    private CategoryRepo categoryRepo;
    private CurrentUserService userService;
    private ProductService productService;
    @Autowired
    public CategoryService(CategoryRepo categoryRepo, CurrentUserService userService, ProductService productService) {
        this.categoryRepo = categoryRepo;
        this.userService = userService;
        this.productService = productService;
    }

    public Category addNewCategory(Category category) {
        return categoryRepo.save(category);
    }

    public Category getCategoryById(long id) {
        return categoryRepo.findById(id).orElse(null
```

```
36 );
37     }
38 
39     public List<Category> getAllCategories() {
40         return this.categoryRepo.findAll();
41     }
42     public void deleteCategory(Long id) {
43         this.categoryRepo.deleteById(id);
44     }
45 
46     @Transactional
47     public void softDeleteCategory(Long id) {
48         List<Products> productsByCategory = this.
   productService.getProductsByCategory(id);
49         productsByCategory.forEach(a-> this.
   productService.deleteProduct(a.getID()));
50         this.categoryRepo.softDeleteCategory(id);
51     }
52 }
53 
```

```java
package com.sportshoes.ecom.services;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.repos.CustomerRepo;
import com.sportshoes.ecom.security.
ApplicationUserDetails;
import org.springframework.dao.
DataIntegrityViolationException;
import org.springframework.security.core.context.
SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.
BCryptPasswordEncoder;
import org.springframework.stereotype.Component;

import javax.transaction.Transactional;
import java.util.List;

@Component
public class CustomerService  {

    private CustomerRepo repo;
    private BCryptPasswordEncoder passwordEncoder;

    public CustomerService(CustomerRepo repo,
BCryptPasswordEncoder passwordEncoder) {
        this.repo = repo;
        this.passwordEncoder = passwordEncoder;
    }

    public Customers addNewCustomer(Customers
customers) {
        System.out.println(customers);
        if (customers.getPassword() == null) {
            throw new IllegalArgumentException("
Password cannot be null");
        }
        customers.setPassword(passwordEncoder.encode(
customers.getPassword()));
        return this.repo.saveAndFlush(customers);
    }
    @Transactional
    public String changePassword(String newPassword,
Long userID) {
        /*ApplicationUserDetails principal = (
```

```
35 ApplicationUserDetails) SecurityContextHolder.
   getContext().getAuthentication().getPrincipal();
36         Long userID = principal.getUserID();
37         System.out.println("changing password for
    " + userID + " new Pass " + newPassword );
38         */this.repo.changeMyPassword(this.
   passwordEncoder.encode(newPassword), userID);
39         return newPassword;
40     }
41
42     public List<Customers> getAllRegisteredUsers() {
43         return this.repo.getAllRegisteredUsers();
44     }
45
46     public void softDeleteAccount(Long id) {
47         this.repo.softDeleteAccount(id);
48     }
49
50
51
52 }
53
```

```java
package com.sportshoes.ecom.services;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.entity.Purchase;
import com.sportshoes.ecom.repos.PurchaseRepo;
import com.sportshoes.ecom.security.
ApplicationUserDetails;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.security.core.context.
SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

@Component
public class PurchaseService {

    @Autowired
    PurchaseRepo purchaseRepo;

    public Purchase savePurchase(Purchase purchase) {
        return purchaseRepo.save(purchase);
    }

    public List<Purchase> getAllMyPurchases(Long id
) {
        return purchaseRepo.getPurchasesByCustomerId(
id);
    }

    public List<Purchase> getAllPurchases() {
        return this.purchaseRepo.findAll();
    }

    public List<Purchase> getPurchasesByCategory(Long
 id) {
        List<Purchase> filtered =  new ArrayList<>();
```

```
39          for(Purchase i: this.purchaseRepo.findAll
    ()) {
40              for ( Map.Entry<Products, Integer> j: i.
    getProduct().entrySet()) {
41                  if(j.getKey().getCategoryId().getID
    () == id)
42                      filtered.add(i);
43              }
44          }
45          return filtered;
46      }
47
48      public List<Purchase> filterPurchaseByDate(
    LocalDate date) {
49          return this.purchaseRepo.filterProductsByDate
    (date);
50      }
51  }
52
```

```java
package com.sportshoes.ecom.services;


import com.sportshoes.ecom.security.ApplicationUserDetails;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;

@Component
public class CurrentUserService {
    private final Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

    public Long getIdFromSecurityContext() {
        Object principal = authentication.getPrincipal();
        if(principal instanceof ApplicationUserDetails )
            return ((ApplicationUserDetails) principal).getUserID();
        return 0L;
    }


}

```

```java
package com.sportshoes.ecom.AuthModels;

import lombok.*;

@NoArgsConstructor
@AllArgsConstructor
@Getter
public class AuthRequest {
    private  String emailId;
    private  String password;
}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\AuthModels\AuthResponse.java

```java
package com.sportshoes.ecom.AuthModels;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;

@AllArgsConstructor
@NoArgsConstructor
@Getter
public class AuthResponse {
    public  String jwtToken;


}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\exceptions\CustomExceptionHandlers.java

```java
package com.sportshoes.ecom.exceptions;

import io.jsonwebtoken.MalformedJwtException;
import lombok.extern.slf4j.Slf4j;
import org.springframework.dao.
DataIntegrityViolationException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.http.converter.
HttpMessageNotReadableException;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.
MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.
ControllerAdvice;
import org.springframework.web.bind.annotation.
ExceptionHandler;
import org.springframework.web.bind.annotation.
ResponseStatus;
import org.springframework.web.servlet.handler.
AbstractHandlerExceptionResolver;

import javax.validation.ConstraintViolation;
import javax.validation.ConstraintViolationException;
import java.util.HashMap;
import java.util.Map;

@Slf4j
@ControllerAdvice
public class CustomExceptionHandlers extends
RuntimeException {
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(HttpMessageNotReadableException
.class)
    public  ResponseEntity<String> JsonParseException
(HttpMessageNotReadableException e) {
        System.out.println("caught this exception");
        log.info(e.getMessage());
        return ResponseEntity.status(HttpStatus.
CONFLICT)
                .body("Invalid JSON Object Provided"
);
    }
    @ResponseStatus(HttpStatus.BAD_REQUEST)
```

```
33      @ExceptionHandler(DataIntegrityViolationException
    .class)
34      public ResponseEntity<String>
    handleDataIntegrityException(
    DataIntegrityViolationException e) {
35          e.getMostSpecificCause();
36          return ResponseEntity.status(HttpStatus.
    CONFLICT).body(e.getMostSpecificCause().getMessage
    ());
37      }
38
39      @ExceptionHandler(MethodArgumentNotValidException
    .class)
40      public ResponseEntity<String>
    handleValidationException(
    MethodArgumentNotValidException e) {
41          Map<String, String> errorMsg = new HashMap
    <>();
42          e.getBindingResult().getFieldErrors().stream
    ().forEach(a->errorMsg.put(a.getField(), a.
    getDefaultMessage()));
43          return  ResponseEntity.status(HttpStatus.
    BAD_REQUEST).body(errorMsg.toString());
44      }
45      @ExceptionHandler(ProductNotFoundException.class)
46      public ResponseEntity<String>
    handleValidationException(ProductNotFoundException e
    ) {
47          System.out.println("catcing JST Exception");
48          return ResponseEntity.status(HttpStatus.
    BAD_REQUEST).body(e.getMessage());
49      }
50
51      @ExceptionHandler(io.jsonwebtoken.
    MalformedJwtException.class)
52      public ResponseEntity<String> handleJWTException(
    MalformedJwtException e) {
53
54          return ResponseEntity.status(HttpStatus.
    FORBIDDEN).body("invalid JWt token");
55      }
56  }
57
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\exceptions\ProductNotFoundException.java

```java
package com.sportshoes.ecom.exceptions;

public class ProductNotFoundException extends RuntimeException {
    public ProductNotFoundException(String message) {
        super(message);
    }
}

```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\controllers\Authentication.java

```java
package com.sportshoes.ecom.controllers;

import com.sportshoes.ecom.AuthModels.AuthRequest;
import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.exceptions.
ProductNotFoundException;
import com.sportshoes.ecom.security.JWTUtil;
import com.sportshoes.ecom.security.
UserDetailsServiceImpl;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.context.annotation.Profile
;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.
AuthenticationManager;
import org.springframework.security.authentication.
BadCredentialsException;
import org.springframework.security.authentication.
UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.
UserDetails;
import org.springframework.security.core.userdetails.
UserDetailsService;
import org.springframework.web.bind.annotation.
PostMapping;
import org.springframework.web.bind.annotation.
RequestBody;
import org.springframework.web.bind.annotation.
RequestMapping;
import org.springframework.web.bind.annotation.
RestController;

@RestController
@Profile("!test")
public class Authentication {

    private AuthenticationManager
authenticationManager;
    @Autowired
    private UserDetailsServiceImpl userDetailsService
;

    @Autowired
```

```
30      private JWTUtil jwtUtil;
31      @Autowired
32      public void setAuthenticationManager(
    AuthenticationManager authenticationManager) {
33          this.authenticationManager =
    authenticationManager;
34      }
35
36      @PostMapping("authenticate")
37      public ResponseEntity<?> createAuthToken(@
    RequestBody AuthRequest request) {
38          try {
39              authenticationManager.authenticate(
40                      new
    UsernamePasswordAuthenticationToken(request.
    getEmailId(), request.getPassword()));
41          } catch (BadCredentialsException e) {
42              throw new ProductNotFoundException("
    Incorect credentials");
43          }
44          UserDetails user = userDetailsService.
    loadUserByUsername(request.getEmailId());
45          final String token = jwtUtil.generateToken(
    user);
46          return ResponseEntity.ok(token);
47      }
48 }
49
```

```java
package com.sportshoes.ecom.controllers;

import com.sportshoes.ecom.entity.Category;
import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.JSONMappers.
CategoryJSONMapper;
import com.sportshoes.ecom.entity.JSONMappers.
ProductJSONFilter;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.entity.Purchase;
import com.sportshoes.ecom.security.
ApplicationUserDetails;
import com.sportshoes.ecom.services.CategoryService;
import com.sportshoes.ecom.services.CustomerService;
import com.sportshoes.ecom.services.ProductService;
import com.sportshoes.ecom.services.PurchaseService;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.format.annotation.
DateTimeFormat;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.
AuthenticationPrincipal;
import org.springframework.security.core.context.
SecurityContextHolder;
import org.springframework.security.core.userdetails.
User;
import org.springframework.validation.annotation.
Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;


@RestController
@RequestMapping("/api/admin")
public class AdminController {

    private final ProductService productService;
    private final CustomerService customerService;
    private final CategoryService categoryService;
```

```
36      private final PurchaseService purchaseService;
37      @Autowired
38      public AdminController(ProductService
    productService,
39                             CustomerService
    customerservice,
40                             CategoryService
    categoryService,
41                             PurchaseService
    purchaseService) {
42          this.productService = productService;
43          this.customerService = customerservice;
44          this.categoryService = categoryService;
45          this.purchaseService = purchaseService;
46      }
47
48      @PostMapping("add-new-product")
49      public Products addProduct(@RequestBody
    ProductJSONFilter productFilter, @
    AuthenticationPrincipal ApplicationUserDetails
    details) {
50          /*Object principal = SecurityContextHolder.
    getContext().getAuthentication().getPrincipal();
51          Long userID = ((ApplicationUserDetails)
    principal).getUserID();*/
52          Products product = new Products(productFilter
    .getCategoryId(), productFilter.getName(),
    productFilter.getPrice());
53          product.setAdmin(new Customers(details.
    getUserID()));
54          return this.productService.addNewProduct(
    product);
55      }
56
57      @Validated
58      @PostMapping("add-new-category")
59      public Category addNewCategory(@Valid @
    RequestBody CategoryJSONMapper categoryMapper,
60                                     @
    AuthenticationPrincipal ApplicationUserDetails
    details) {
61          Category category = new Category(details.
    getUserID(), categoryMapper.getName());
62          //category.setAdmin(new Customers(details.
```

```
62 getUserID()));
63         return this.categoryService.addNewCategory(
   category);
64     }
65 
66 
67     @GetMapping("get-all-registered-users")
68     public List<Customers> getRegisteredUsers() {
69         return this.customerService.
   getAllRegisteredUsers();
70     }
71 
72 
73     @GetMapping(value = "category/{id}")
74     public Category getCategoryById(@PathVariable
   Long id) {
75         return this.categoryService.getCategoryById(
   id);
76     }
77 
78     @GetMapping("category/get-all-categories")
79     public List<Category> getAddCategories() {
80          return this.categoryService.
   getAllCategories();
81     }
82 
83     @GetMapping("purchase-summary")
84     public List<Purchase> getPurchaseSummary() {
85         return this.purchaseService.getAllPurchases
   ();
86     }
87 
88     /*@DeleteMapping("category/delete-category/{id
   }")
89     public ResponseEntity deleteCategory(@
   PathVariable("id") Long id) {
90         categoryService.deleteCategory(id);
91         return ResponseEntity.ok("Category " + id
    + "deleted");
92     }*/
93 
94     @DeleteMapping("product/delete-product/{id}")
95     public ResponseEntity deleteProduct(@
   PathVariable("id") Long id) {
```

```
 96          this.productService.deleteProduct(id);
 97          return ResponseEntity.ok("Product " + id + "
    deleted");
 98      }
 99
100      @DeleteMapping("account/delete-account/{id}")
101      public void deleteAccount(@PathVariable("id")
    Long id) {
102          this.customerService.softDeleteAccount(id);
103      }
104
105      @DeleteMapping("category/delete-category/{
    categoryId}")
106      public void deleteCategory(@PathVariable("
    categoryId") long id) {
107          this.categoryService.softDeleteCategory(id);
108      }
109
110      @GetMapping("purchase/filter-purchase-by-
    category/{id}")
111      public ResponseEntity filterByCategory(@
    PathVariable("id") Long id) {
112          return ResponseEntity.ok(this.
    purchaseService.getPurchasesByCategory(id));
113      }
114
115      @GetMapping("purchase/filter-purchase-by-date/{
    date}")
116      public ResponseEntity filterByCategory(@
    PathVariable("date")  @DateTimeFormat(pattern = "
    yyyy-MM-dd") LocalDate date) {
117          System.out.println("localDate " + date);
118          return ResponseEntity.ok(this.
    purchaseService.filterPurchaseByDate(date));
119      }
120  }
121
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\controllers\PublicController.java

```java
package com.sportshoes.ecom.controllers;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.JSONMappers.customerMapper;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.exceptions.ProductNotFoundException;
import com.sportshoes.ecom.services.CustomerService;
import com.sportshoes.ecom.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api")
public class PublicController {

    private ProductService productService;
    private CustomerService customerService;

    @Autowired
    public PublicController(ProductService productService, CustomerService customerService) {
        this.productService = productService;
        this.customerService = customerService;
    }

    @GetMapping(value = "product/{id}")
    public Products getProductById(@PathVariable("id") Long id) {
        Products product = productService.findProductById(id);
        if(product == null)
            throw new ProductNotFoundException("Product with " + id + " Not found" );
        return product;
    }

    @GetMapping("products")
```

```
38      public List<Products> getAllProducts() {
39          return productService.getAllProducts();
40      }
41
42      @GetMapping("products/search-by-category/{id}")
43      public List<Products> filterProductsByCategory(@
    PathVariable("id") long id) {
44          return productService.getProductsByCategory(
    id);
45      }
46
47      @PostMapping("register")
48      public ResponseEntity<String> AddCustomer(@Valid
     @RequestBody customerMapper customer) {
49          Customers customerEntity = this.
    customerService.addNewCustomer(customer.mapToCustomer
    ());
50          return ResponseEntity.ok("Registration
    success\n " + customerEntity);
51
52      }
53 }
54
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\controllers\ProductController.java

```java
package com.sportshoes.ecom.controllers;

import com.fasterxml.jackson.core.
JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.sportshoes.ecom.entity.Cart;
import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.entity.Purchase;
import com.sportshoes.ecom.exceptions.
ProductNotFoundException;
import com.sportshoes.ecom.security.
ApplicationUserDetails;
import com.sportshoes.ecom.services.ProductService;
import com.sportshoes.ecom.services.PurchaseService;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.
AuthenticationPrincipal;
import org.springframework.security.core.context.
SecurityContextHolder;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpSession;
import java.util.*;

@RestController
@RequestMapping("api/customer/products")
public class ProductController {
    private final PurchaseService purchaseService;
    private final ProductService productService;
    @Autowired
    public ProductController(PurchaseService
purchaseService, ProductService productService) {
        this.purchaseService = purchaseService;
        this.productService =  productService;
    }

    @PostMapping("add-to-cart")
    public ResponseEntity<String> buyProduct(@
RequestBody Cart newCart,

```

```
36 HttpSession session) {
37         Map<Long, Integer> cart = (Map<Long, Integer
   >) session.getAttribute("cart");
38         if(cart==null)
39             cart = new HashMap<>();
40         Long price = (Long) session.getAttribute("
   price");
41         if(price == null)
42             price = 0L;
43         if(!this.productService.findProduct(newCart.
   getProductId()))
44             throw new ProductNotFoundException("
   Product " + newCart.getProductId() + " not found");
45         price +=this.productService.getProductPrice(
   newCart.getProductId());
46         cart.merge(newCart.getProductId(),newCart.
   getQuantity(),
47                 (a,b)->b+newCart.getQuantity());
48         session.setAttribute("cart", cart);
49         session.setAttribute("price", price);
50         return ResponseEntity.ok("product ID" +
   newCart.getProductId() + " has been added to cart");
51    }
52
53    @GetMapping("view-cart")
54    public ResponseEntity viewCart(HttpSession session
   ) throws JsonProcessingException {
55        Map<Long, Integer> cart = (Map<Long, Integer
   >) session.getAttribute("cart");
56        System.out.println("cart  " + cart);
57        if(cart == null)
58            return ResponseEntity.ok("Cart is empty");
59        Long price = (Long) session.getAttribute("
   price");
60        String json = new ObjectMapper().
   writeValueAsString(cart);
61        return ResponseEntity.ok(json + "\n\n" + "
   total Price " + price );
62     }
63
64    @PostMapping("buy-product")
65    public ResponseEntity buyProduct(@
   AuthenticationPrincipal ApplicationUserDetails user,
   HttpSession session) {
```

```
66          Map<Products, Integer> purchaseBag =new
    HashMap<> ();
67          Map<Long, Integer> cart = (Map<Long, Integer
    >) session.getAttribute("cart");
68          if(cart == null)
69              return ResponseEntity.status(HttpStatus.
    BAD_REQUEST).body("cart is empty");
70          for (Map.Entry<Long, Integer> entry : cart.
    entrySet()) {
71              purchaseBag.put(new Products(entry.getKey
    ()),entry.getValue());
72          }
73
74          Purchase purchase = new Purchase();
75
76          Long price = (Long) session.getAttribute("
    price");
77           purchase.setTotalPrice(price);
78          purchase.setCustomer(new Customers(user.
    getUserID()));
79           purchase.setProduct(purchaseBag);
80          System.out.println(purchase);
81          this.purchaseService.savePurchase(purchase);
82          session.setAttribute("cart", null);
83          session.setAttribute("price", null);
84          return ResponseEntity.ok("success");
85      }
86
87      @GetMapping("/purchase-history")
88       public List<Purchase> getMyPurchases() {
89          Object principal = SecurityContextHolder.
    getContext().getAuthentication().getPrincipal();
90          Long userId = ((ApplicationUserDetails)
    principal).getUserID();
91          return purchaseService.getAllMyPurchases(
    userId);
92      }
93
94
95  }
96
```

File - F:\SportsShoesEcomApp\src\main\java\com\sportshoes\ecom\controllers\CustomerController.java

```java
package com.sportshoes.ecom.controllers;

import com.sportshoes.ecom.entity.Customers;
import com.sportshoes.ecom.entity.JSONMappers.customerMapper;
import com.sportshoes.ecom.entity.Products;
import com.sportshoes.ecom.exceptions.ProductNotFoundException;
import com.sportshoes.ecom.security.ApplicationUserDetails;
import com.sportshoes.ecom.services.CustomerService;
import com.sportshoes.ecom.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;
import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("api/customer")
@Validated
public class CustomerController {
    private final CustomerService customerService;
    private final ProductService productService;
    @Autowired
    public CustomerController(CustomerService customerService, ProductService productService) {
        this.customerService = customerService;
        this.productService = productService;
    }

    @PostMapping("/change-password")
    public ResponseEntity<?> changeMyPassword(@RequestBody String newPassword) {
        String password = this.customerService.changePassword(newPassword, 1L);
        return ResponseEntity.ok("Password changed: " + password);
    }
```

```
35
36
37      /*@PostMapping("register")
38      public ResponseEntity<String> AddCustomer(@Valid
     @RequestBody customerMapper customer) {
39          Customers customerEntity = this.
   customerService.addNewCustomer(customer.mapToCustomer
   ());
40          return ResponseEntity.ok("Registration
   success\n " + customerEntity);
41
42      }*/
43      /*@GetMapping(value = "product/{id}")
44      public Products getProductById(@PathVariable("id
   ") Long id) {
45          Products product = productService.
   findProductById(id);
46          if(product == null)
47              throw new ProductNotFoundException("
   Product with " + id + " Not found" );
48          return product;
49      }*/
50 /*
51      @GetMapping("products")
52      public List<Products> getAllProducts() {
53          return productService.getAllProducts();
54      }
55
56      @GetMapping("products/search-by-category/{id}")
57      public List<Products> filterProductsByCategory(@
   PathVariable("id") long id) {
58          return productService.getProductsByCategory(
   id);
59      }*/
60
61
62 }
63
```

File - F:\SportsShoesEcomApp\src\main\resources\application.properties

```
1 spring.datasource.url=jdbc:postgresql://localhost:
  5432/sports_prod
2 spring.datasource.username=postgres
3 spring.datasource.password=root
4 spring.jpa.hibernate.ddl-auto=create-drop
5 spring.jpa.generate-ddl=true
6 spring.jpa.show-sql=true
7
```