CAPSTONE PROJECT

ON

ShopForHome

(Shopping Web Application for Home Décor Stuff)

WIPRO_FSD _JAVA

Company Name –: Wipro Limited
Training Partner –: Great Learning

Capstone Project_Group-A_C-VIII_G-1

Under the guidance of

Mr. Akash Kale

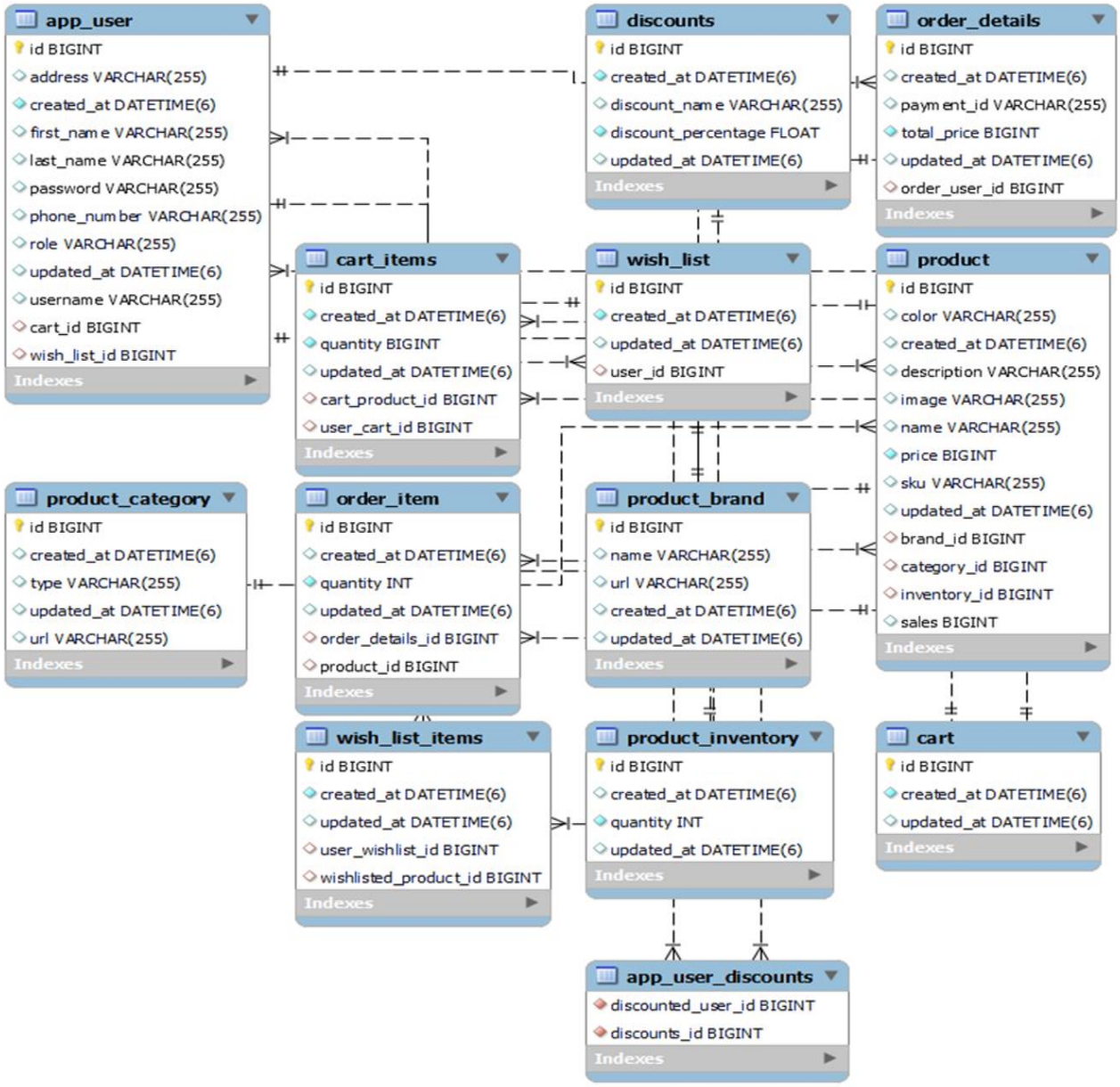
Submitted By

1. **Swastika Kundu**
2. **Vivek Kumar Singh**
3. **Shreya Rajesh Thalkar**
4. **Subhasis Khuntia**
5. **Sahaj Shrimal**

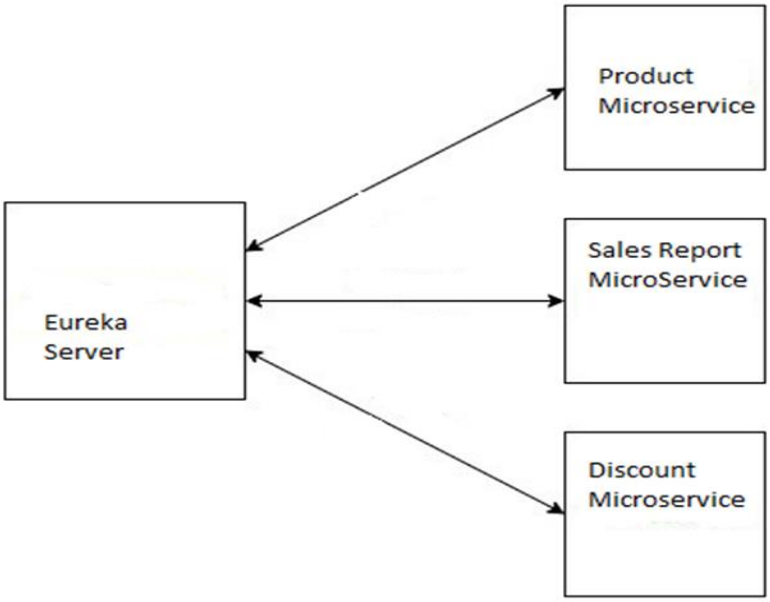
ABSTRACT

Online shopping plays a great importance in the modern business environment. E-commerce is process of doing business through computer networks. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products. Due to Covid 19, all the offline shopping stopped. So, the store wants to move to the online platforms and wants their own web application. ShopForHome has opened the door of opportunity and advantage to the firms. This project is a shopping web application for home décor stuffs. This project saves time for both the buyer and the retailer, reducing phone calls about availability, specifications, hours of operation or other information can be easily found on company and product pages.

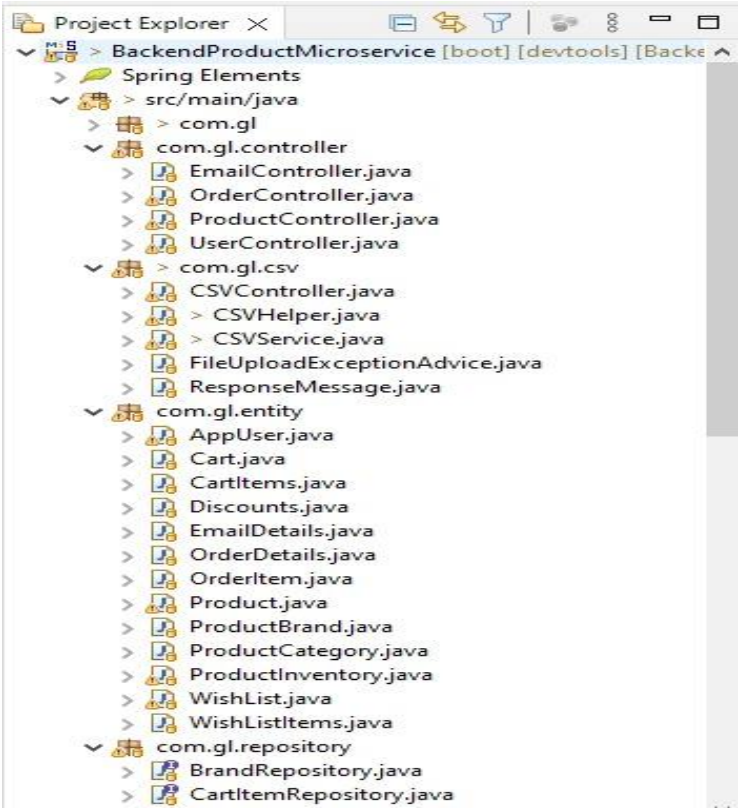
ER DIAGRAM

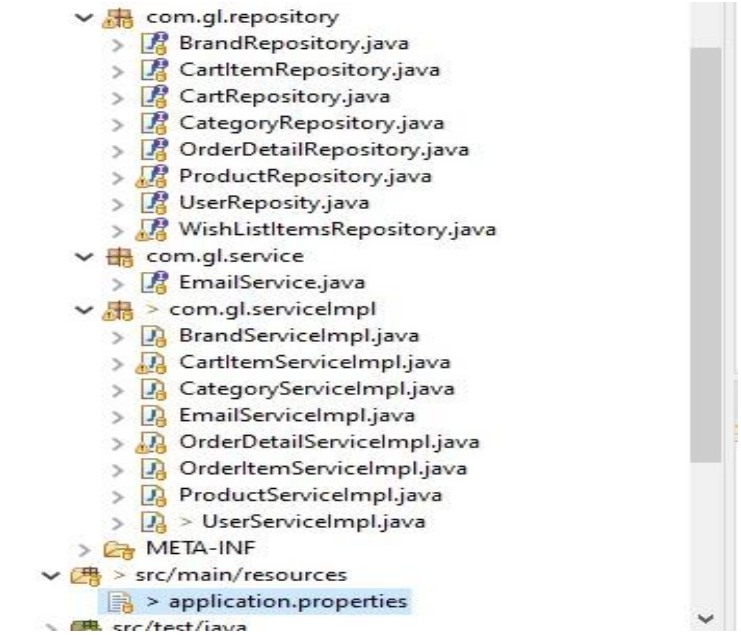

app_user
id BIGINT
address VARCHAR(255)
created_at DATETIME(6)
first_name VARCHAR(255)
last_name VARCHAR(255)
password VARCHAR(255)
phone_number VARCHAR(255)
role VARCHAR(255)
updated_at DATETIME(6)
username VARCHAR(255)
cart_id BIGINT
wish_list_id BIGINT
Indexes
discounts
id BIGINT
created_at DATETIME(6)
discount_name VARCHAR(255)
discount_percentage FLOAT
updated_at DATETIME(6)
Indexes
order_details
id BIGINT
created_at DATETIME(6)
payment_id VARCHAR(255)
total_price BIGINT
updated_at DATETIME(6)
order_user_id BIGINT
Indexes
cart_items
id BIGINT
created_at DATETIME(6)
quantity BIGINT
updated_at DATETIME(6)
cart_product_id BIGINT
user_cart_id BIGINT
Indexes
wish_list
id BIGINT
created_at DATETIME(6)
updated_at DATETIME(6)
user_id BIGINT
Indexes
product
id BIGINT
color VARCHAR(255)
created_at DATETIME(6)
description VARCHAR(255)
image VARCHAR(255)
name VARCHAR(255)
price BIGINT
sku VARCHAR(255)
updated_at DATETIME(6)
brand_id BIGINT
category_id BIGINT
inventory_id BIGINT
sales BIGINT
Indexes
product_category
id BIGINT
created_at DATETIME(6)
type VARCHAR(255)
updated_at DATETIME(6)
url VARCHAR(255)
Indexes
order_item
id BIGINT
created_at DATETIME(6)
quantity INT
updated_at DATETIME(6)
order_details_id BIGINT
product_id BIGINT
Indexes
product_brand
id BIGINT
name VARCHAR(255)
url VARCHAR(255)
created_at DATETIME(6)
updated_at DATETIME(6)
Indexes
wish_list_items
id BIGINT
created_at DATETIME(6)
updated_at DATETIME(6)
user_wishlist_id BIGINT
wishlisted_product_id BIGINT
Indexes
product_inventory
id BIGINT
created_at DATETIME(6)
quantity INT
updated_at DATETIME(6)
Indexes
cart
id BIGINT
created_at DATETIME(6)
updated_at DATETIME(6)
Indexes
app_user_discounts
discounted_user_id BIGINT
discounts_id BIGINT
Indexes

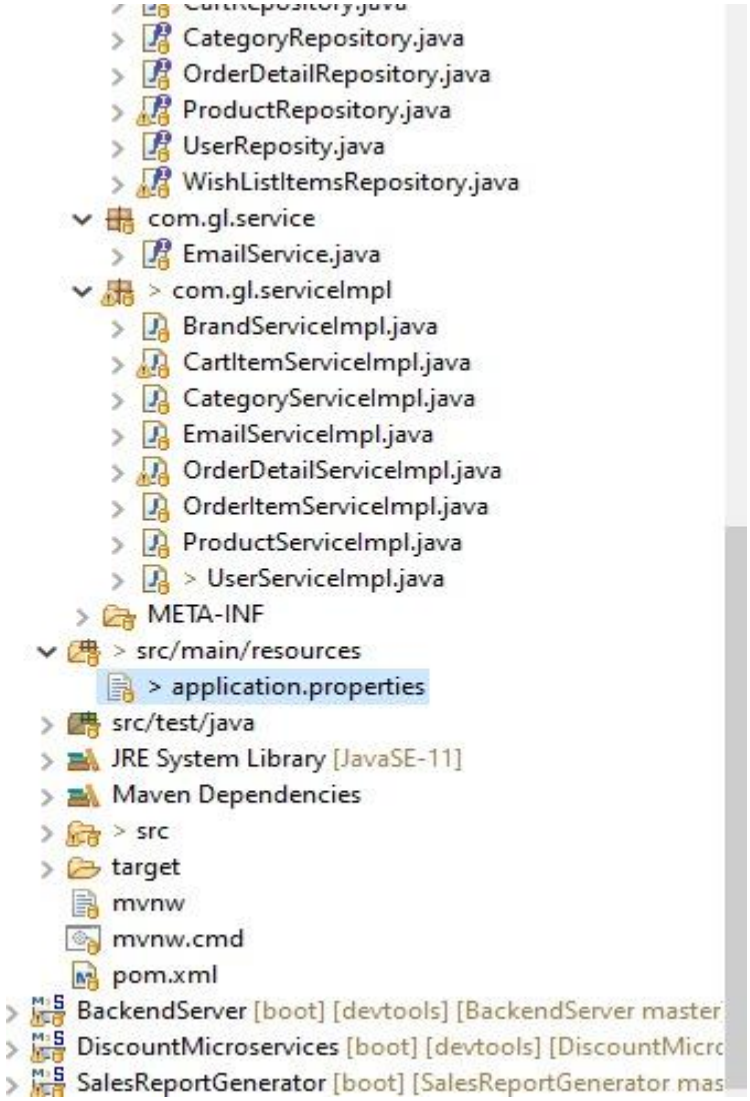
MICROSERVICE ARCHITECTURE

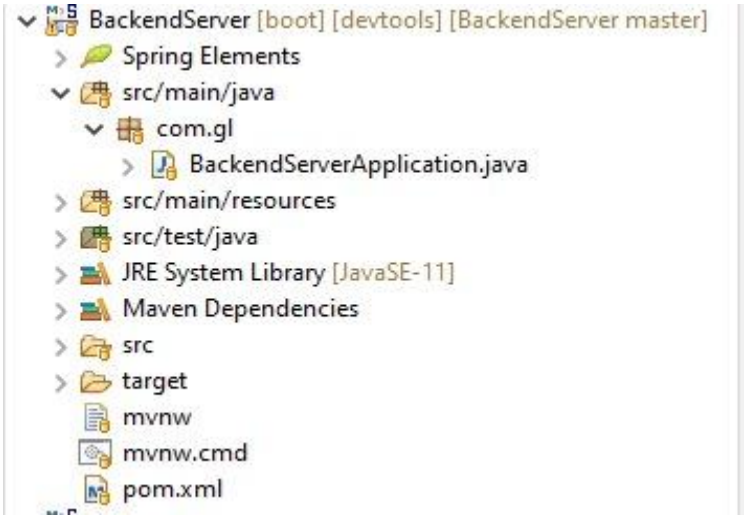


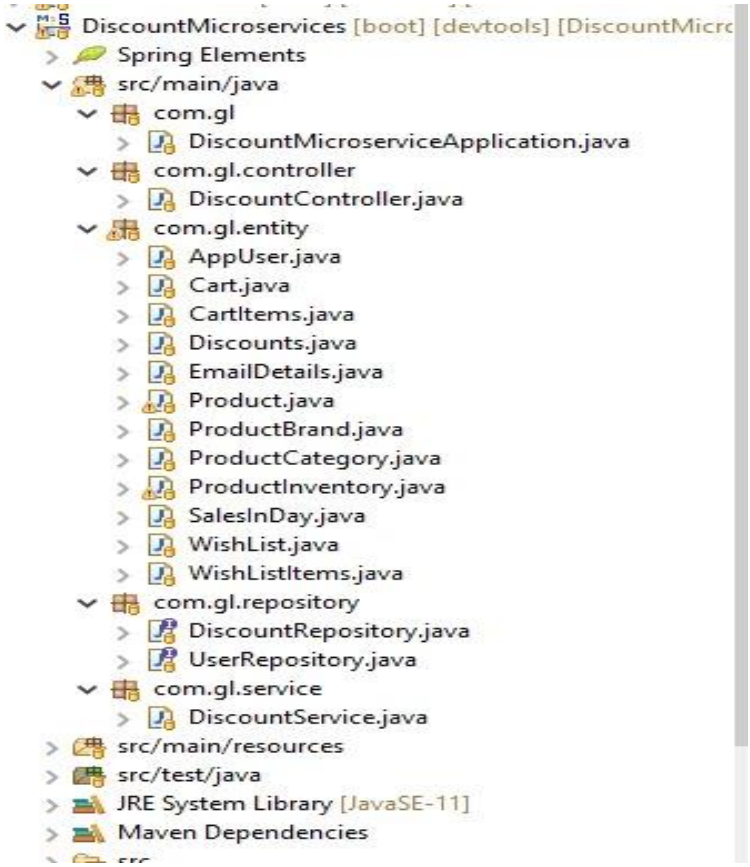
PROJECT STRUCTURE

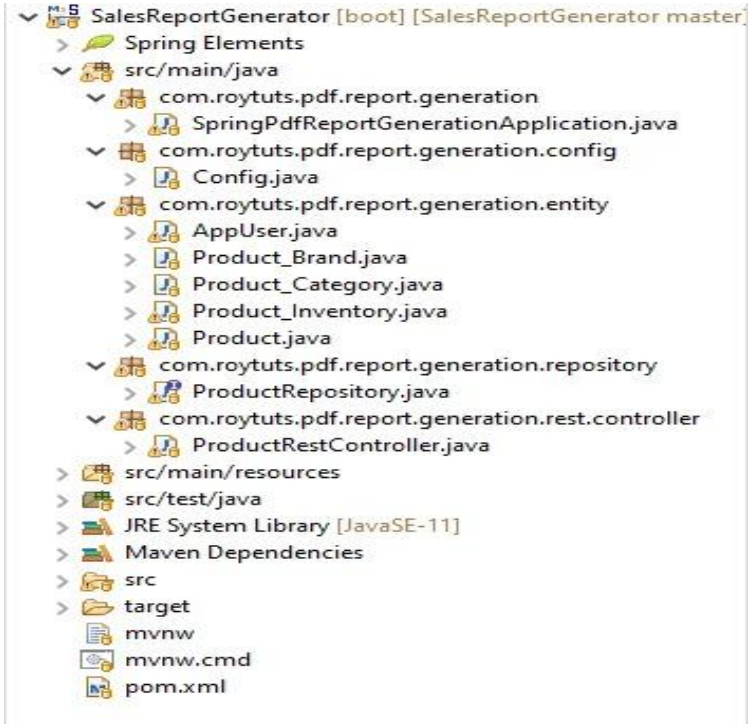





CategoryRepository.java
OrderDetailRepository.java
ProductRepository.java
UserReposity.java
WishListItemsRepository.java
com.gl.service
EmailService.java
com.gl.serviceImpl
BrandServiceImpl.java
CartItemServiceImpl.java
CategoryServiceImpl.java
EmailServiceImpl.java
OrderDetailServiceImpl.java
OrderItemServiceImpl.java
ProductServiceImpl.java
UserServiceImpl.java
META-INF
src/main/resources
application.properties
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml
BackendServer [boot] [devtools] [BackendServer master]
DiscountMicroservices [boot] [devtools]
SalesReportGenerator [boot] [SalesReportGenerator

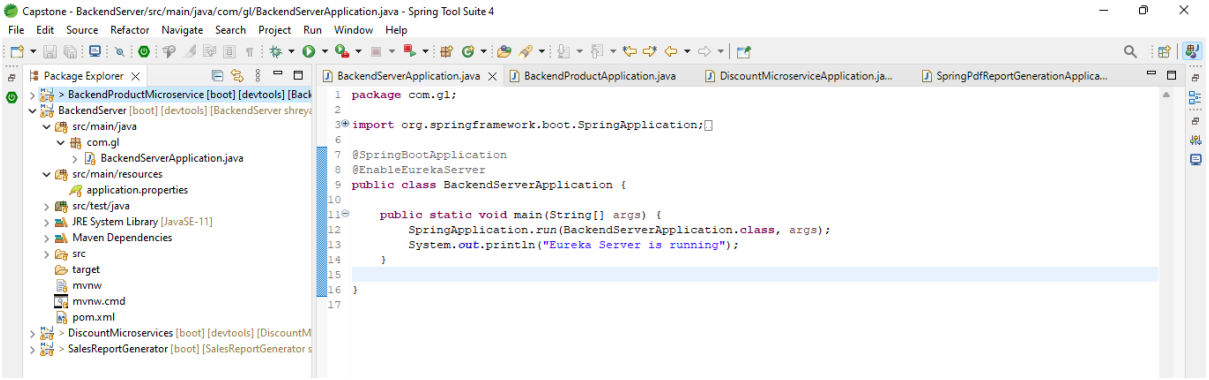

BackendServer [boot] [devtools] [BackendServer master]
Spring Elements
src/main/java
com.gl
BackendServerApplication.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml


DiscountMicroservices [boot] [devtools] [DiscountMicrc
Spring Elements
src/main/java
com.gl
DiscountMicroserviceApplication.java
com.gl.controller
DiscountController.java
com.gl.entity
AppUser.java
Cart.java
CartItems.java
Discounts.java
EmailDetails.java
Product.java
ProductBrand.java
ProductCategory.java
ProductInventory.java
SalesInDay.java
WishList.java
WishListItems.java
com.gl.repository
DiscountRepository.java
UserRepository.java
com.gl.service
DiscountService.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies

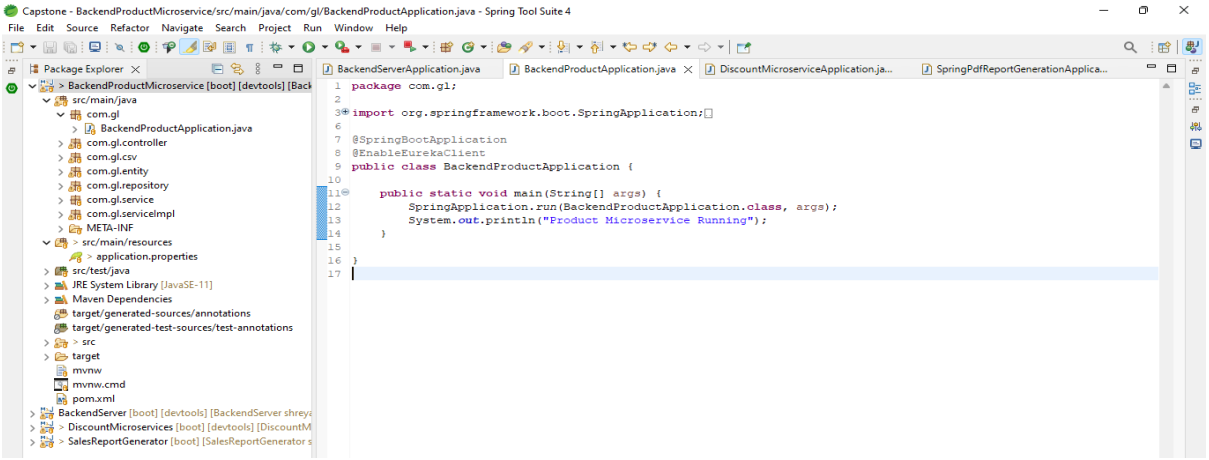

SalesReportGenerator [boot] [SalesReportGenerator master]
Spring Elements
src/main/java
com.roytuts.pdf.report.generation
SpringPdfReportGenerationApplication.java
com.roytuts.pdf.report.generation.config
Config.java
com.roytuts.pdf.report.generation.entity
AppUser.java
Product_Brand.java
Product_Category.java
Product_Inventory.java
Product.java
com.roytuts.pdf.report.generation.repository
ProductRepository.java
com.roytuts.pdf.report.generation.rest.controller
ProductRestController.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml

CODE SNIPPETS

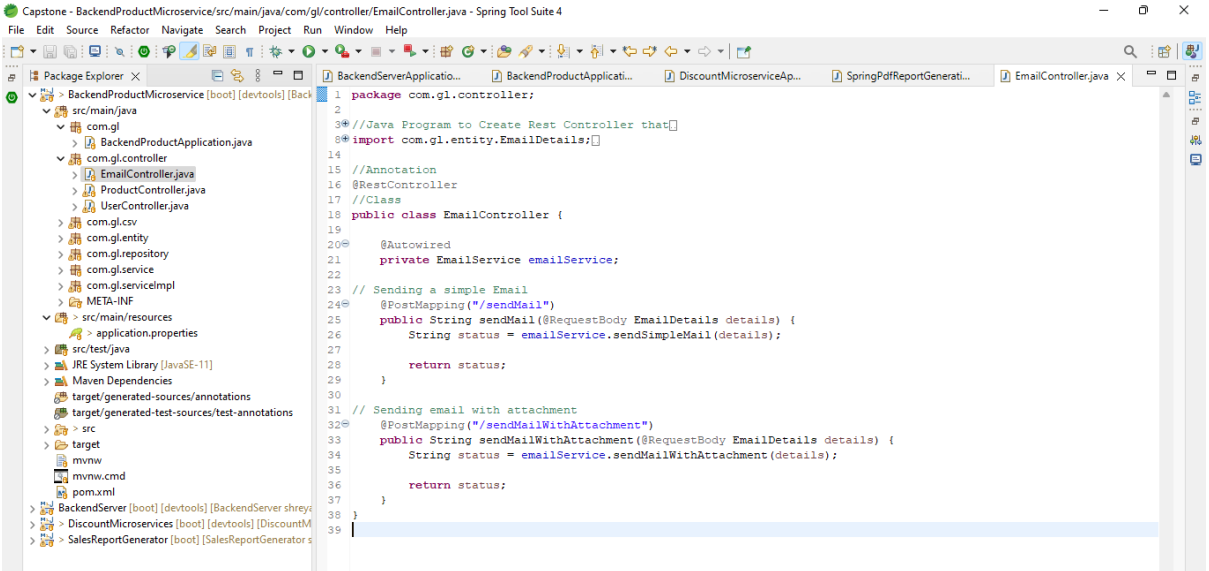
Eureka Server



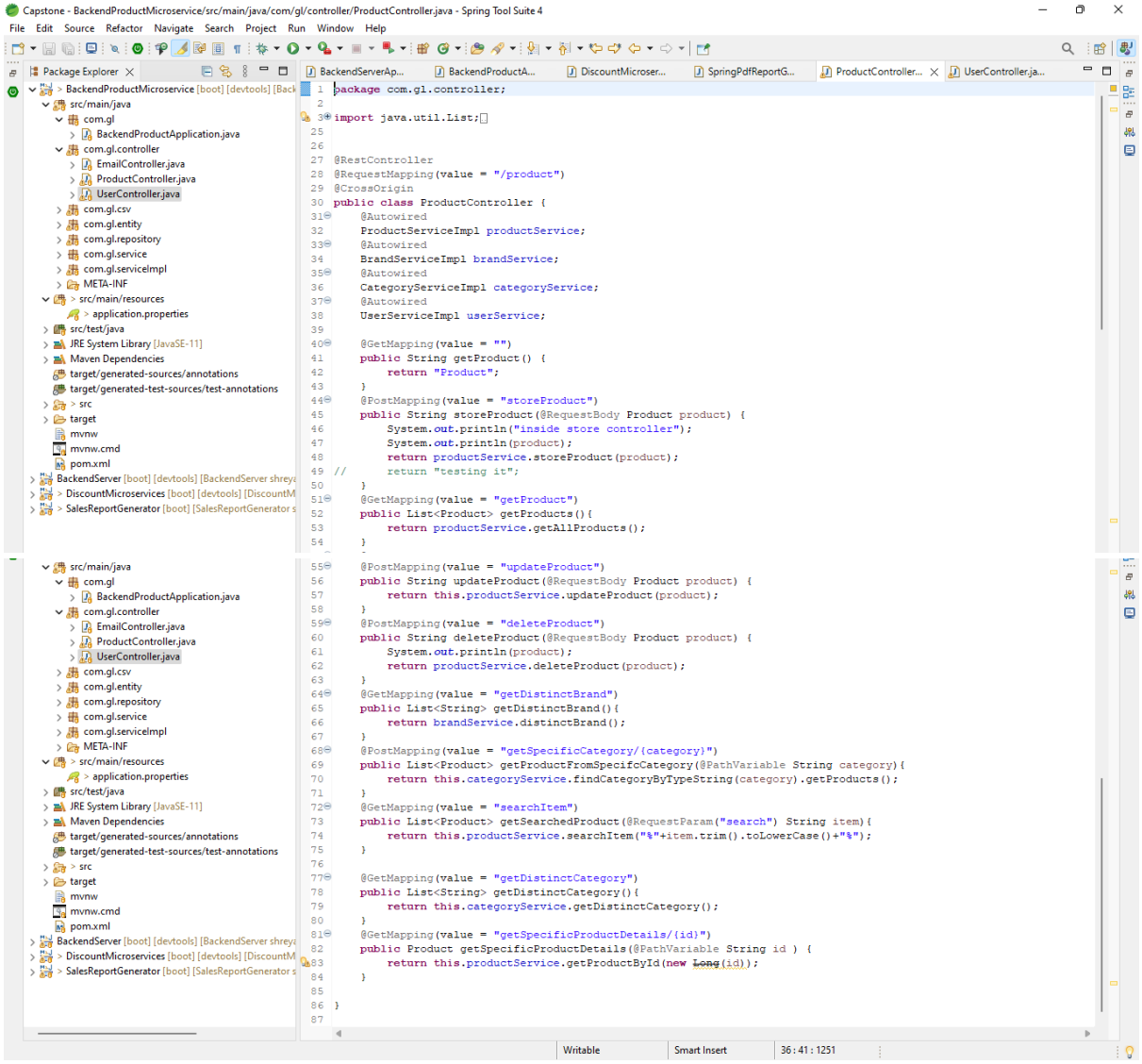
Product Microservice



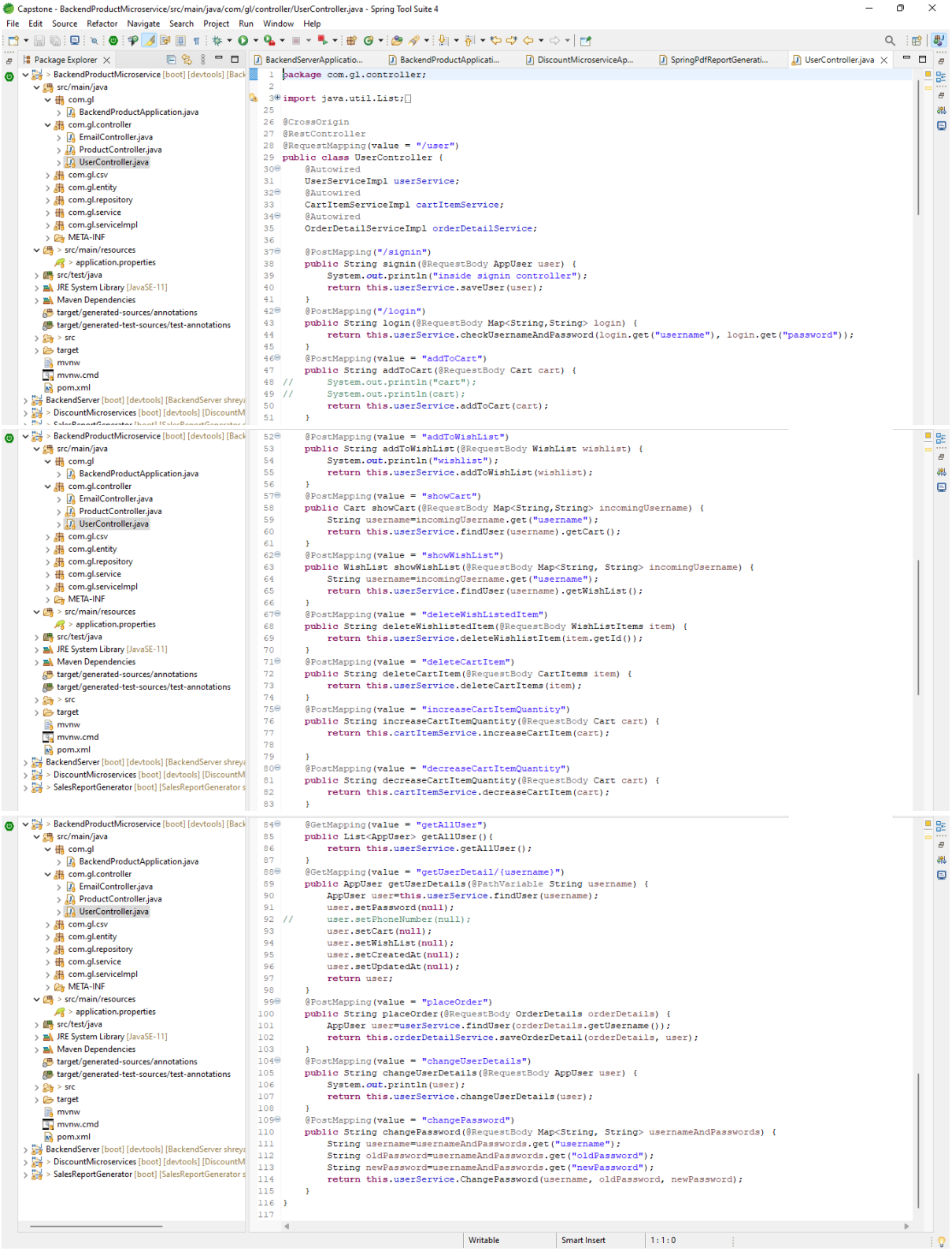
Email Controller



Product Controller

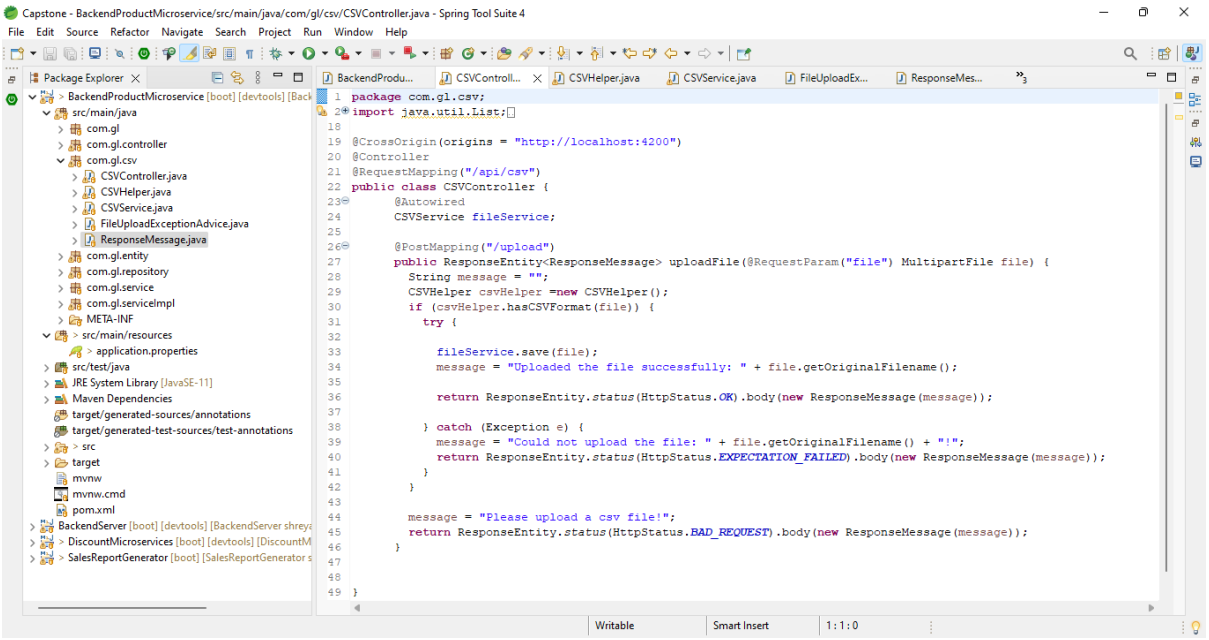


User Controller



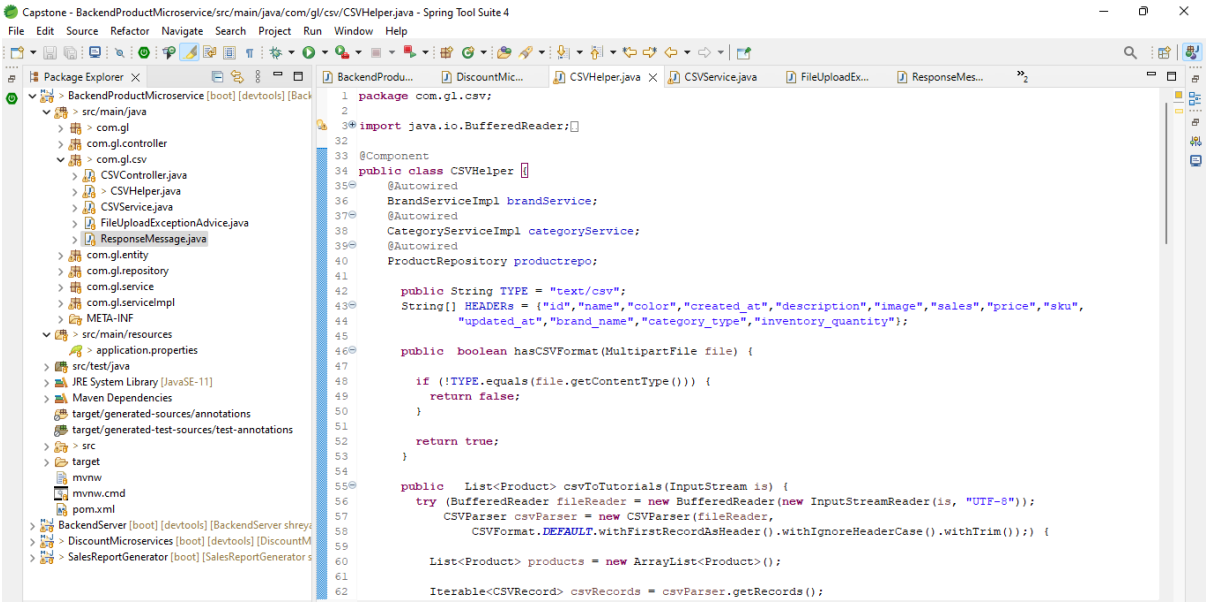

```java
package com.gl.controller;

import java.util.List;

@CrossOrigin
@RestController
@RequestMapping(value = "/user")
public class UserController {
    @Autowired
    UserServiceImpl userService;
    @Autowired
    CartItemServiceImpl cartItemService;
    @Autowired
    OrderDetailServiceImpl orderDetailService;

    @PostMapping("/signin")
    public String signin(@RequestBody AppUser user) {
        System.out.println("inside signin controller");
        return this.userService.saveUser(user);
    }
    @PostMapping("/login")
    public String login(@RequestBody Map<String,String> login) {
        return this.userService.checkUsernameAndPassword(login.get("username"), login.get("password"));
    }
    @PostMapping(value = "addToCart")
    public String addToCart(@RequestBody Cart cart) {
//      System.out.println("cart");
//      System.out.println(cart);
        return this.userService.addToCart(cart);
    }
    @PostMapping(value = "addToWishList")
    public String addToWishList(@RequestBody WishList wishlist) {
        System.out.println("wishlist");
        return this.userService.addToWishList(wishlist);
    }
    @PostMapping(value = "showCart")
    public Cart showCart(@RequestBody Map<String,String> incomingUsername) {
        String username=incomingUsername.get("username");
        return this.userService.findUser(username).getCart();
    }
    @PostMapping(value = "showWishList")
    public WishList showWishList(@RequestBody Map<String, String> incomingUsername) {
        String username=incomingUsername.get("username");
        return this.userService.findUser(username).getWishList();
    }
    @PostMapping(value = "deleteWishListedItem")
    public String deleteWishlistedItem(@RequestBody WishListItems item) {
        return this.userService.deleteWishlistItem(item.getId());
    }
    @PostMapping(value = "deleteCartItem")
    public String deleteCartItem(@RequestBody CartItems item) {
        return this.userService.deleteCartItems(item);
    }
    @PostMapping(value = "increaseCartItemQuantity")
    public String increaseCartItemQuantity(@RequestBody Cart cart) {
        return this.cartItemService.increaseCartItem(cart);

    }
    @PostMapping(value = "decreaseCartItemQuantity")
    public String decreaseCartItemQuantity(@RequestBody Cart cart) {
        return this.cartItemService.decreaseCartItem(cart);
    }
    @GetMapping(value = "getAllUser")
    public List<AppUser> getAllUser(){
        return this.userService.getAllUser();
    }
    @GetMapping(value = "getUserDetail/{username}")
    public AppUser getUserDetails(@PathVariable String username) {
        AppUser user=this.userService.findUser(username);
        user.setPassword(null);
//      user.setPhoneNumber(null);
        user.setCart(null);
        user.setWishList(null);
        user.setCreatedAt(null);
        user.setUpdatedAt(null);
        return user;
    }
    @PostMapping(value = "placeOrder")
    public String placeOrder(@RequestBody OrderDetails orderDetails) {
        AppUser user=userService.findUser(orderDetails.getUsername());
        return this.orderDetailService.saveOrderDetail(orderDetails, user);
    }
    @PostMapping(value = "changeUserDetails")
    public String changeUserDetails(@RequestBody AppUser user) {
        System.out.println(user);
        return this.userService.changeUserDetails(user);
    }
    @PostMapping(value = "changePassword")
    public String changePassword(@RequestBody Map<String, String> usernameAndPasswords) {
        String username=usernameAndPasswords.get("username");
        String oldPassword=usernameAndPasswords.get("oldPassword");
        String newPassword=usernameAndPasswords.get("newPassword");
        return this.userService.ChangePassword(username, oldPassword, newPassword);
    }
}
```

CSV Controller

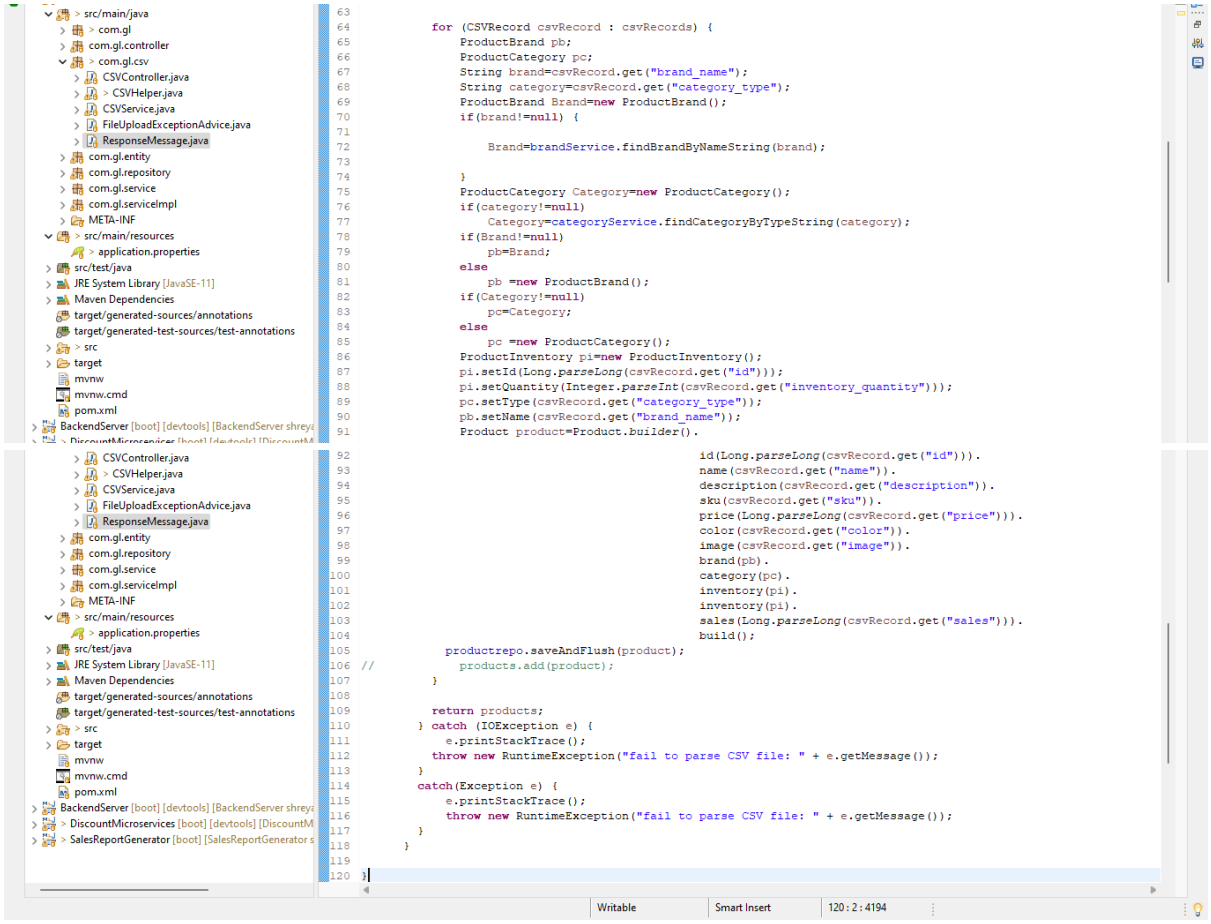



```java
package com.gl.csv;
import java.util.List;

@CrossOrigin(origins = "http://localhost:4200")
@Controller
@RequestMapping("/api/csv")
public class CSVController {
      @Autowired
      CSVService fileService;

      @PostMapping("/upload")
      public ResponseEntity<ResponseMessage> uploadFile(@RequestParam("file") MultipartFile file) {
        String message = "";
        CSVHelper csvHelper =new CSVHelper();
        if (csvHelper.hasCSVFormat(file)) {
          try {

            fileService.save(file);
            message = "Uploaded the file successfully: " + file.getOriginalFilename();

            return ResponseEntity.status(HttpStatus.OK).body(new ResponseMessage(message));

          } catch (Exception e) {
            message = "Could not upload the file: " + file.getOriginalFilename() + "!";
            return ResponseEntity.status(HttpStatus.EXPECTATION_FAILED).body(new ResponseMessage(message));
          }
        }

        message = "Please upload a csv file!";
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new ResponseMessage(message));
      }


}
```

CSV Helper

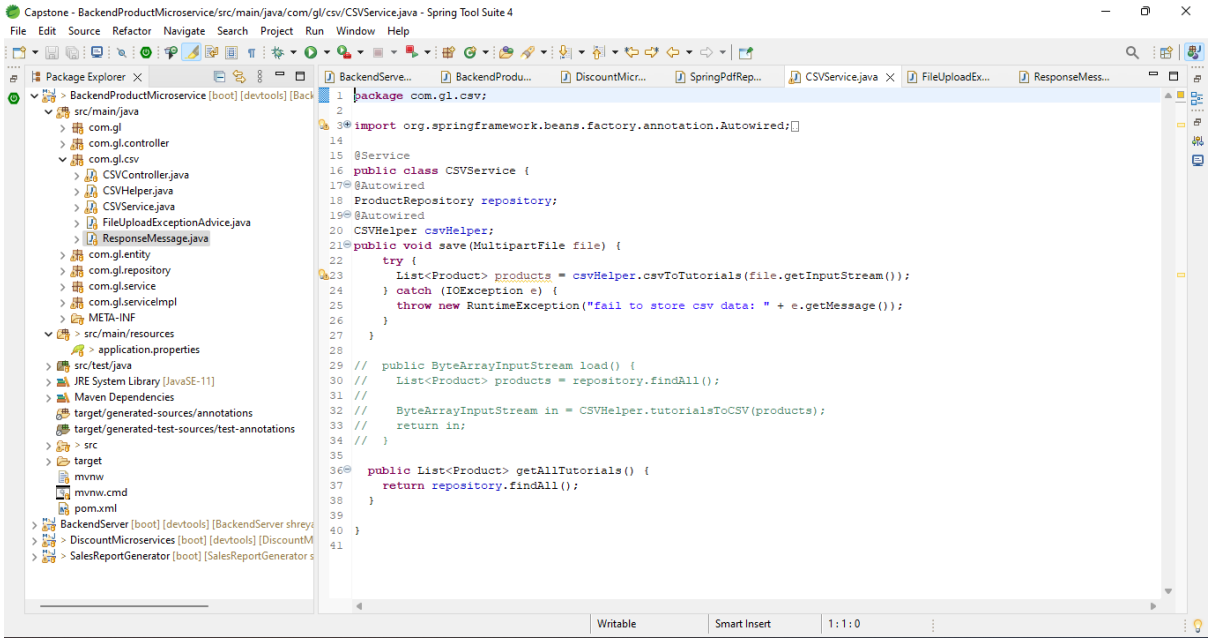



```java
package com.gl.csv;

import java.io.BufferedReader;

@Component
public class CSVHelper {
    @Autowired
    BrandServiceImpl brandService;
    @Autowired
    CategoryServiceImpl categoryService;
    @Autowired
    ProductRepository productrepo;

      public String TYPE = "text/csv";
      String[] HEADERs = {"id","name","color","created_at","description","image","sales","price","sku",
              "updated_at","brand_name","category_type","inventory_quantity"};

      public  boolean hasCSVFormat(MultipartFile file) {

        if (!TYPE.equals(file.getContentType())) {
          return false;
        }

        return true;
      }

      public   List<Product> csvToTutorials(InputStream is) {
        try (BufferedReader fileReader = new BufferedReader(new InputStreamReader(is, "UTF-8"));
            CSVParser csvParser = new CSVParser(fileReader,
                CSVFormat.DEFAULT.withFirstRecordAsHeader().withIgnoreHeaderCase().withTrim());) {

          List<Product> products = new ArrayList<Product>();

          Iterable<CSVRecord> csvRecords = csvParser.getRecords();
```



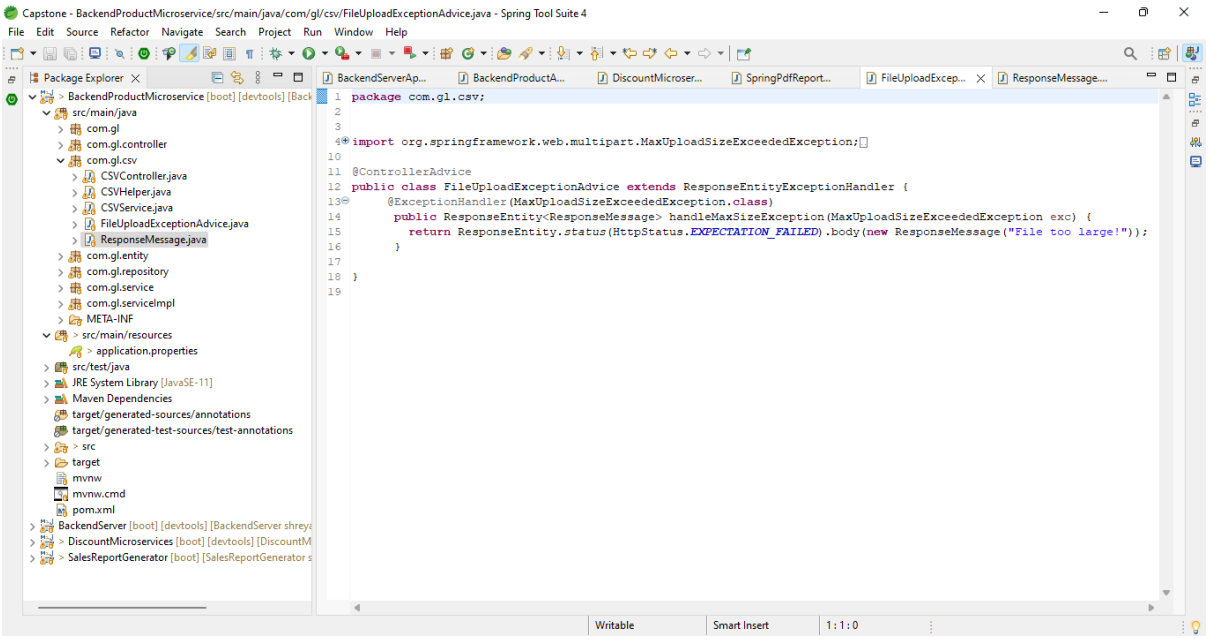
```
        for (CSVRecord csvRecord : csvRecords) {
            ProductBrand pb;
            ProductCategory pc;
            String brand=csvRecord.get("brand_name");
            String category=csvRecord.get("category_type");
            ProductBrand Brand=new ProductBrand();
            if(brand!=null) {

                Brand=brandService.findBrandByNameString(brand);

            }
            ProductCategory Category=new ProductCategory();
            if(category!=null)
                Category=categoryService.findCategoryByTypeString(category);
            if(Brand!=null)
                pb=Brand;
            else
                pb =new ProductBrand();
            if(Category!=null)
                pc=Category;
            else
                pc =new ProductCategory();
            ProductInventory pi=new ProductInventory();
            pi.setId(Long.parseLong(csvRecord.get("id")));
            pi.setQuantity(Integer.parseInt(csvRecord.get("inventory_quantity")));
            pc.setType(csvRecord.get("category_type"));
            pb.setName(csvRecord.get("brand_name"));
            Product product=Product.builder().

                                                id(Long.parseLong(csvRecord.get("id"))).
                                                name(csvRecord.get("name")).
                                                description(csvRecord.get("description")).
                                                sku(csvRecord.get("sku")).
                                                price(Long.parseLong(csvRecord.get("price"))).
                                                color(csvRecord.get("color")).
                                                image(csvRecord.get("image")).
                                                brand(pb).
                                                category(pc).
                                                inventory(pi).
                                                inventory(pi).
                                                sales(Long.parseLong(csvRecord.get("sales"))).
                                                build();
              productrepo.saveAndFlush(product);
//              products.add(product);
            }

            return products;
        } catch (IOException e) {
            e.printStackTrace();
            throw new RuntimeException("fail to parse CSV file: " + e.getMessage());
        }
        catch(Exception e) {
            e.printStackTrace();
            throw new RuntimeException("fail to parse CSV file: " + e.getMessage());
        }
    }

}
```

CSV Service

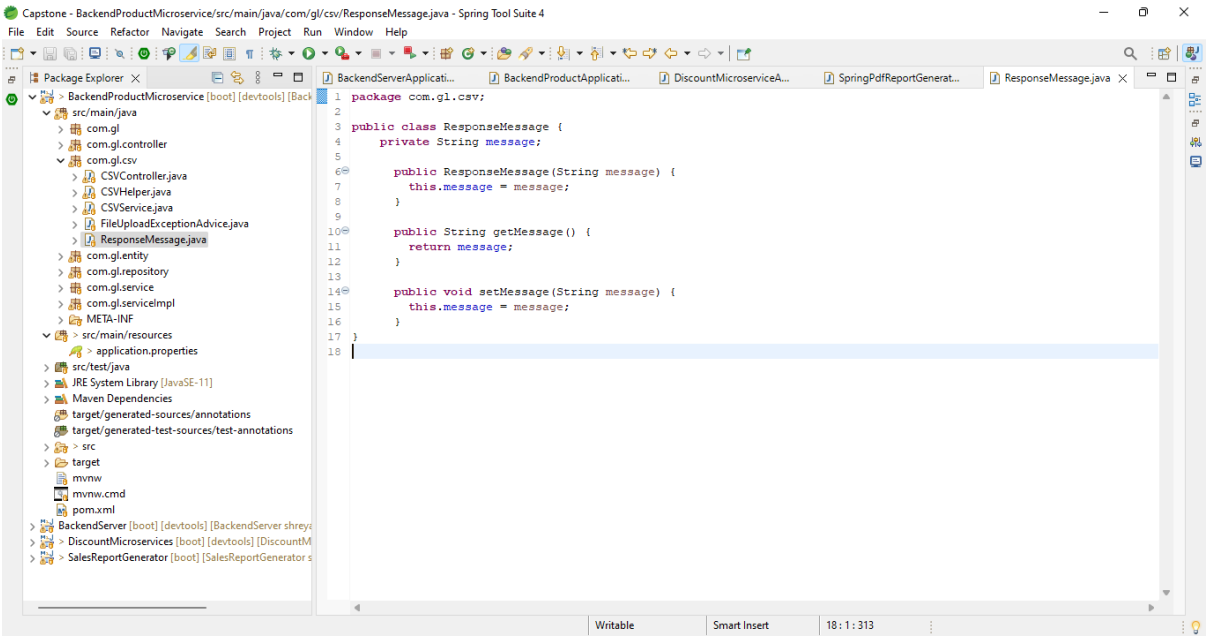


```
package com.gl.csv;

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

@Service
public class CSVService {
@Autowired
ProductRepository repository;
@Autowired
CSVHelper csvHelper;
public void save(MultipartFile file) {
    try {
      List<Product> products = csvHelper.csvToTutorials(file.getInputStream());
    } catch (IOException e) {
      throw new RuntimeException("fail to store csv data: " + e.getMessage());
    }
  }

//  public ByteArrayInputStream load() {
//    List<Product> products = repository.findAll();
//
//    ByteArrayInputStream in = CSVHelper.tutorialsToCSV(products);
//    return in;
//  }

  public List<Product> getAllTutorials() {
    return repository.findAll();
  }

}
```

File Upload Exception Advice

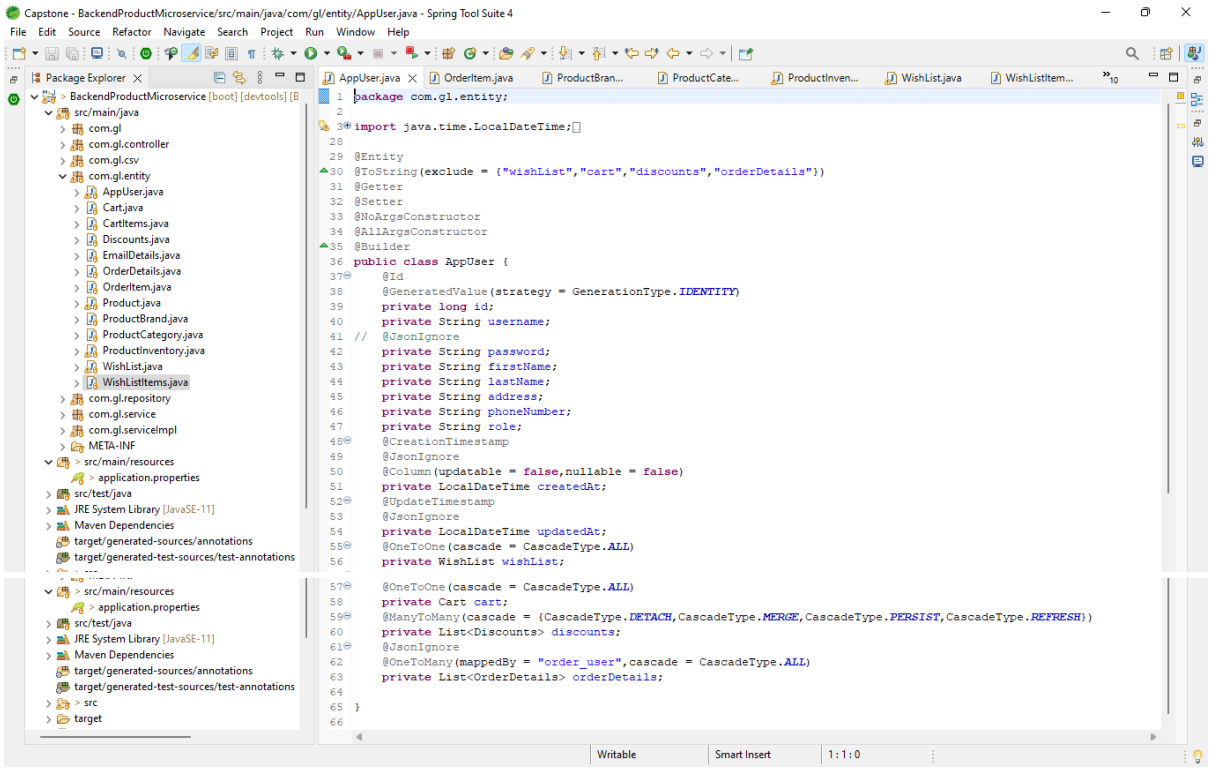


Response Message

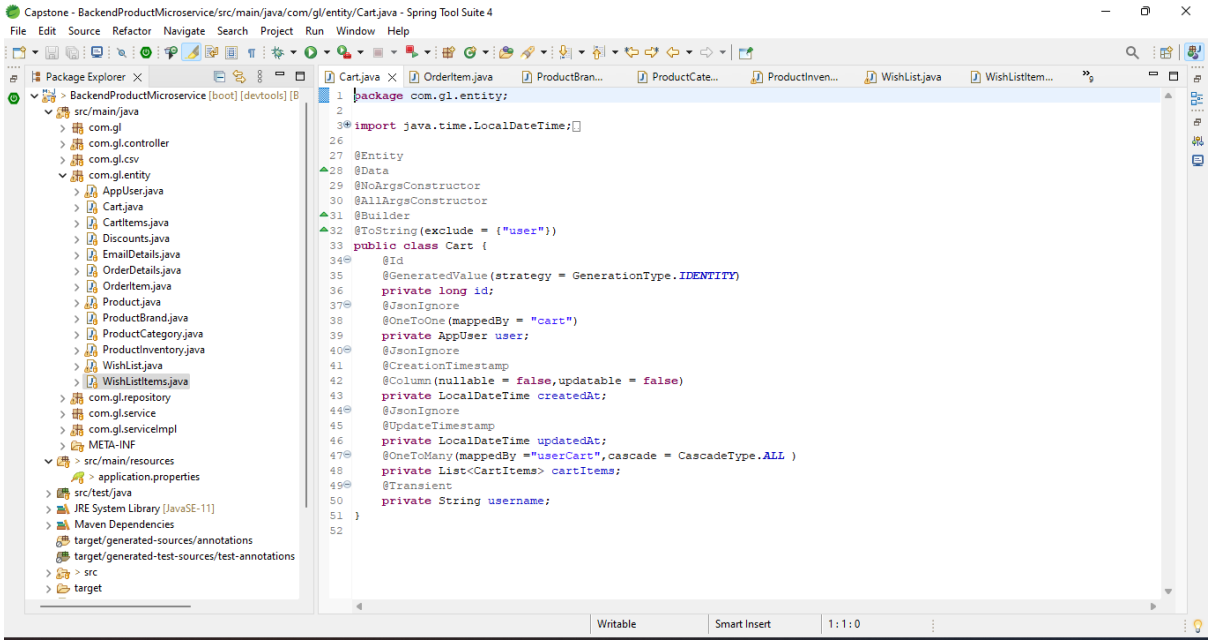


Entities

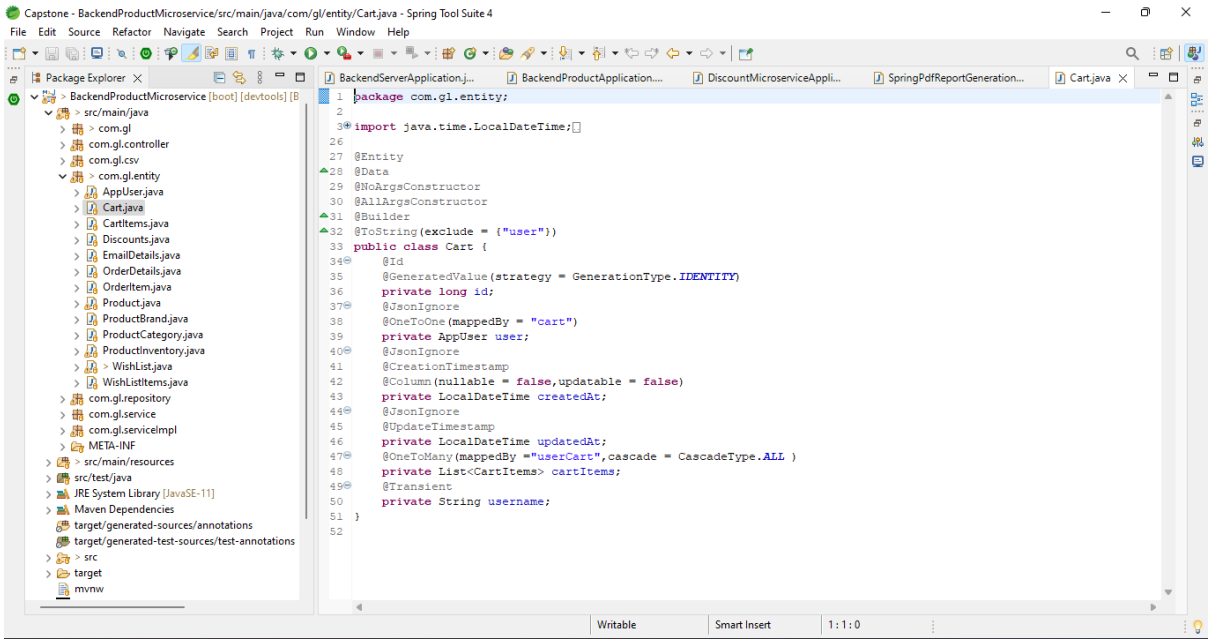
App User



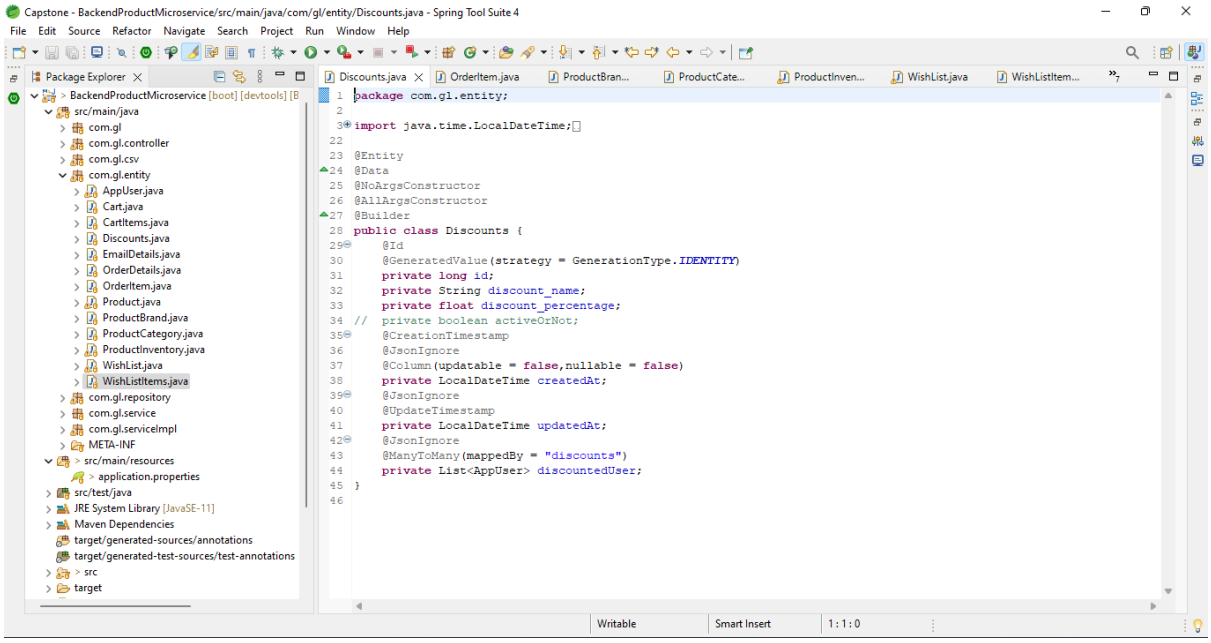
Cart



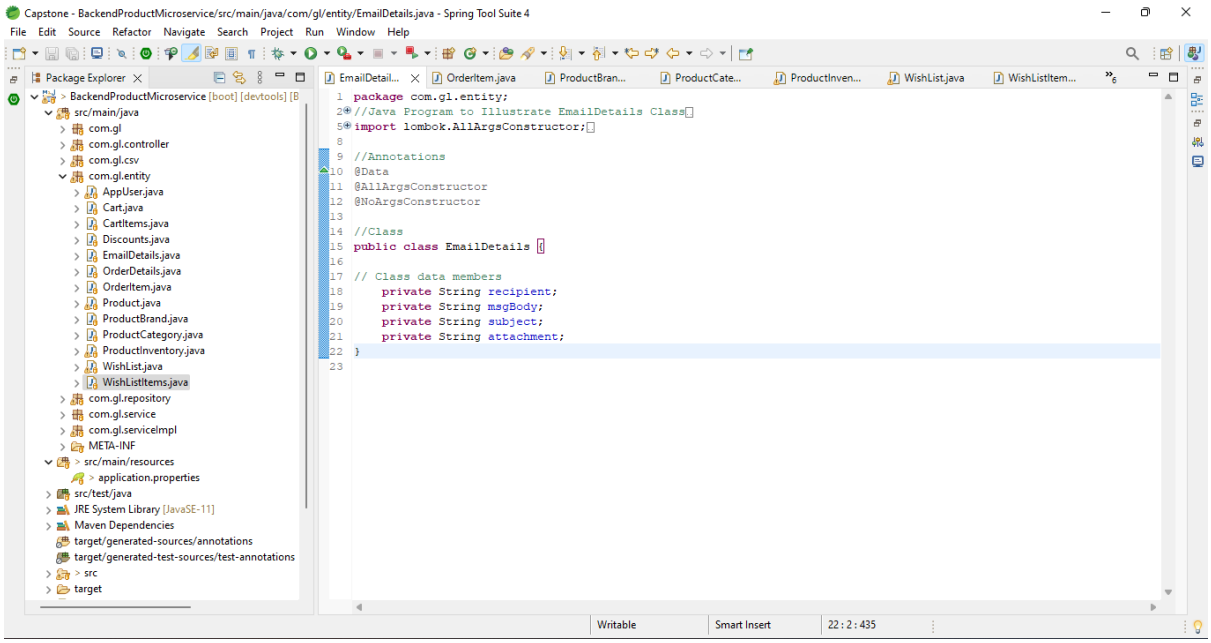
Cart Items



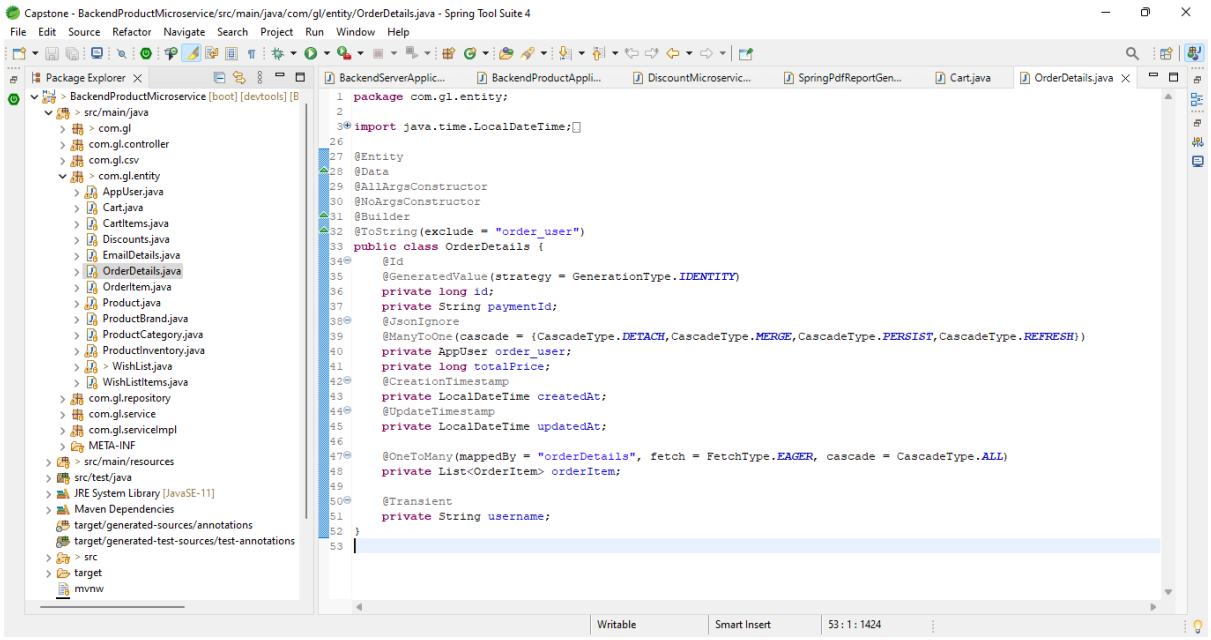
Discounts



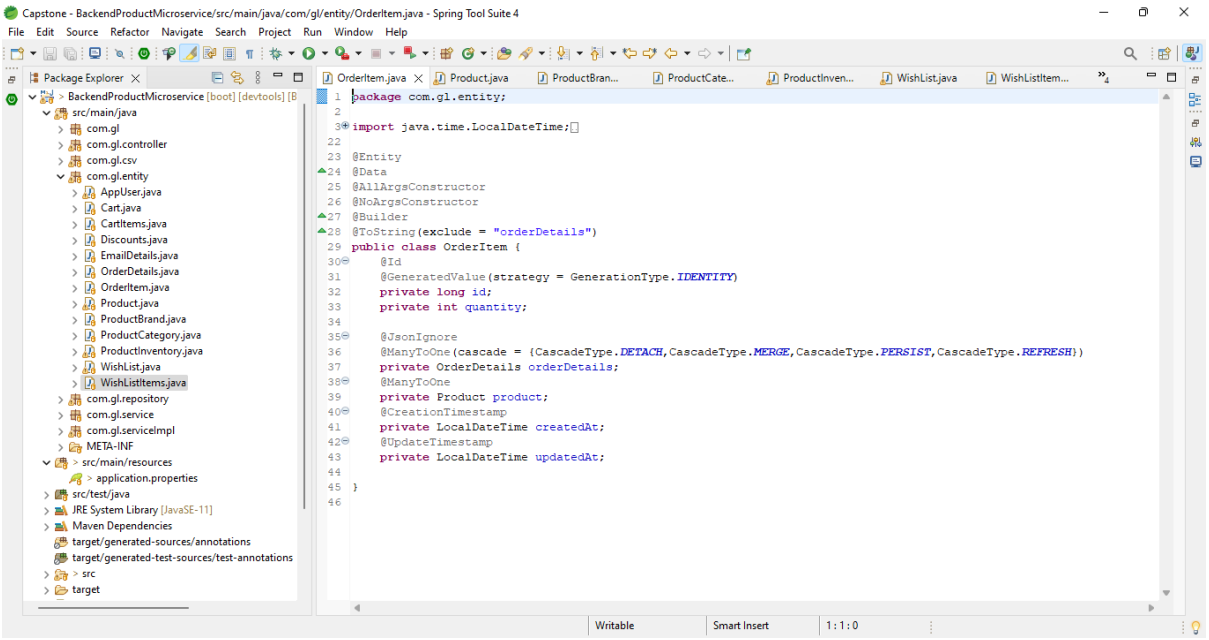
Email Details



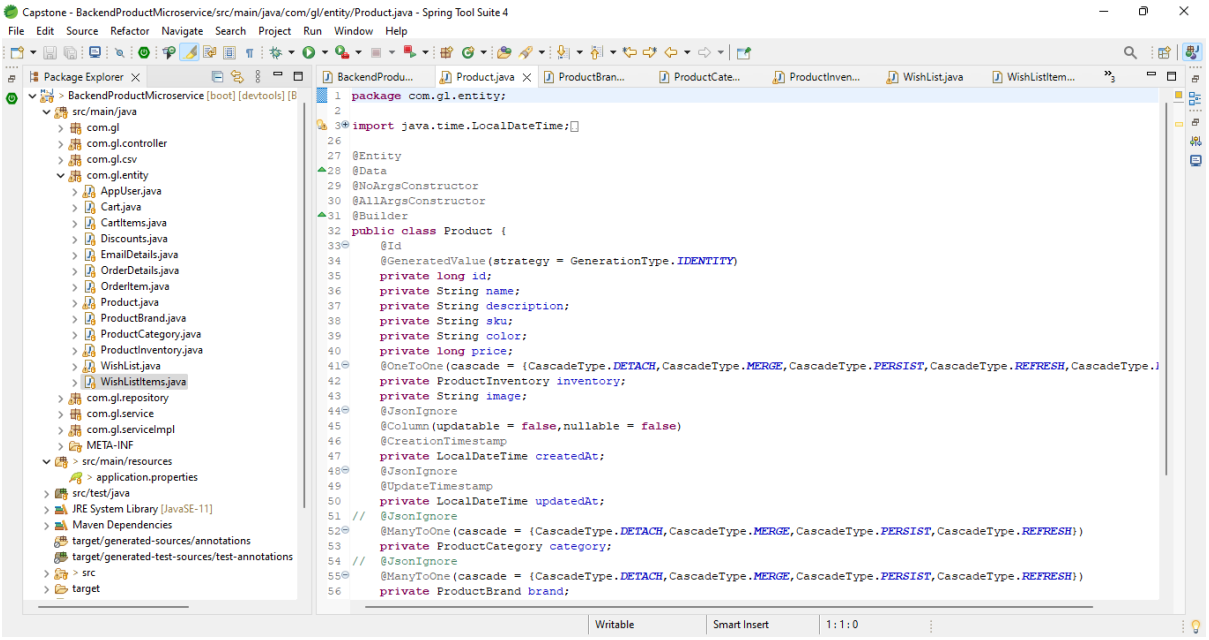
Order Details



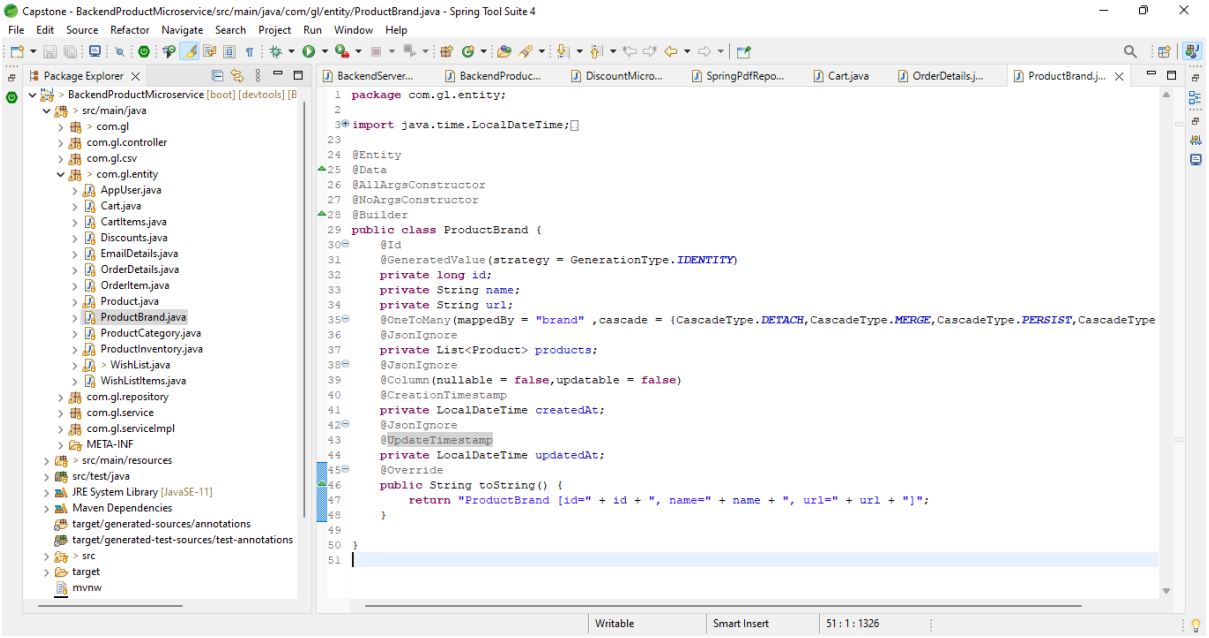
Order Item



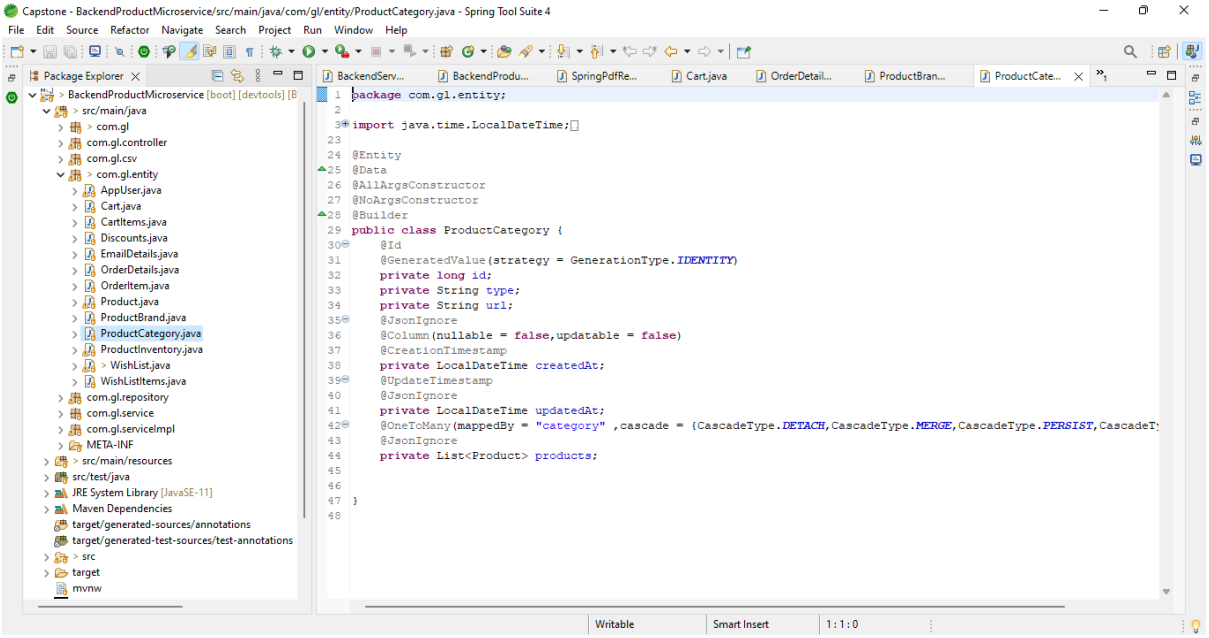
Product



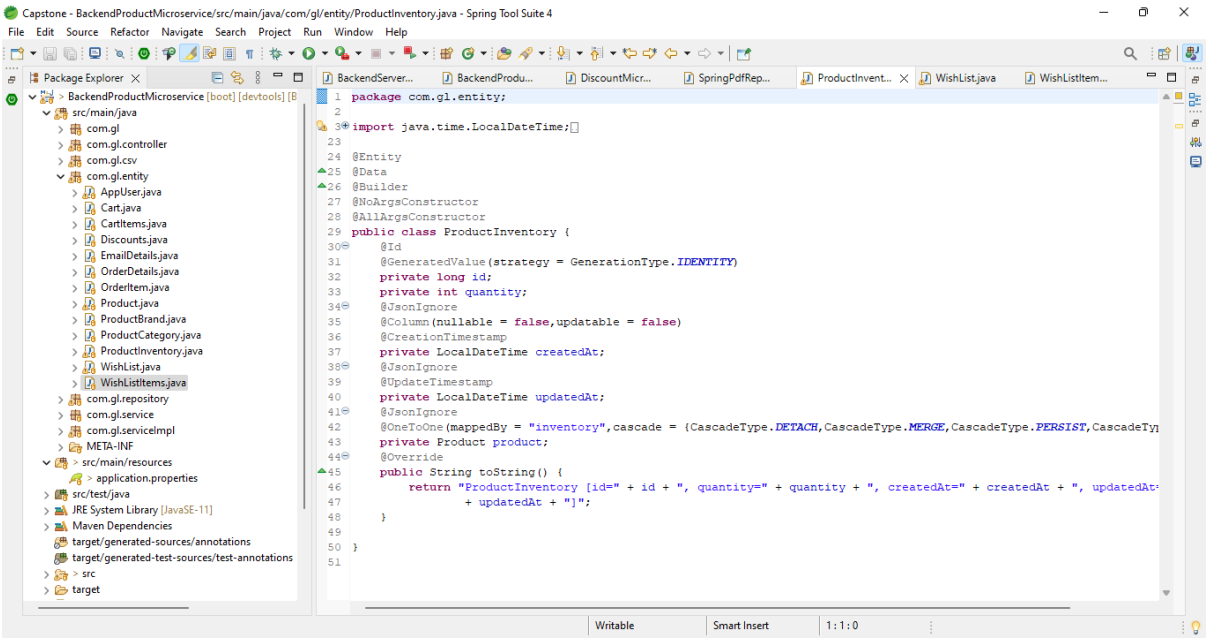
Product Brand



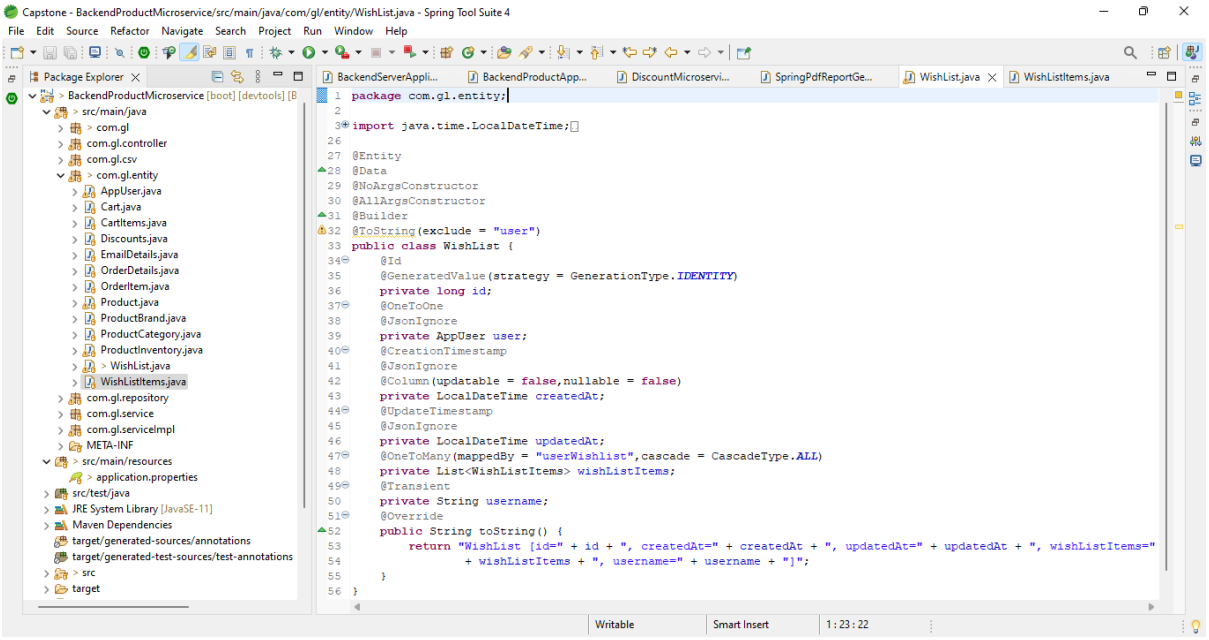
Product Category



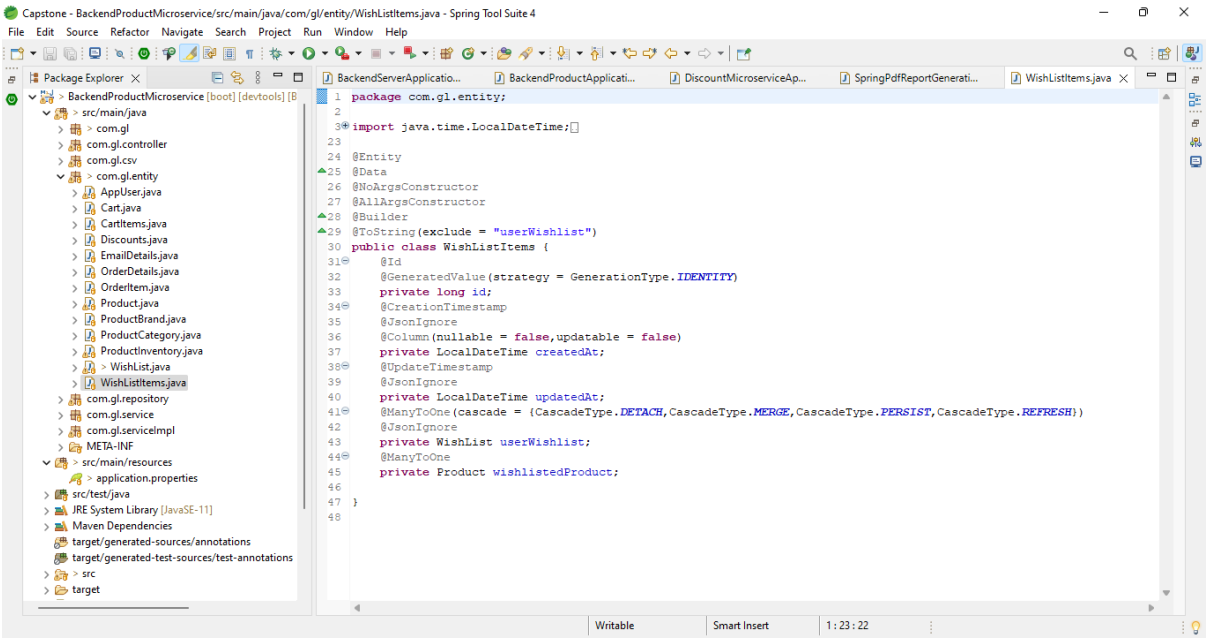
Product Inventory



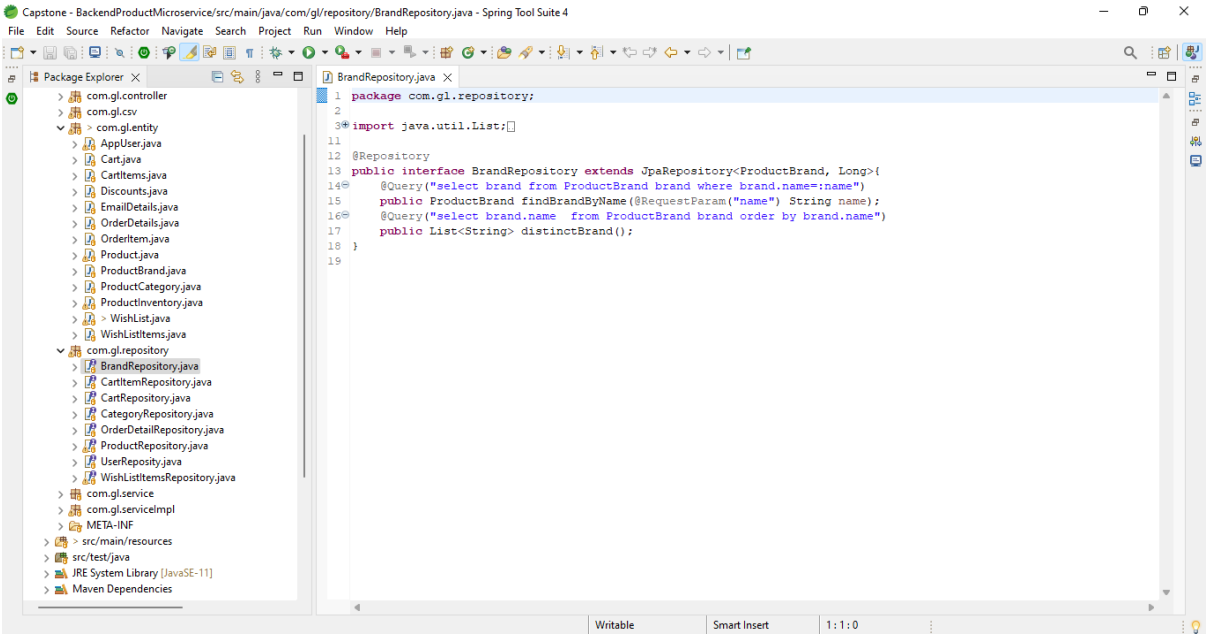
Wishlist



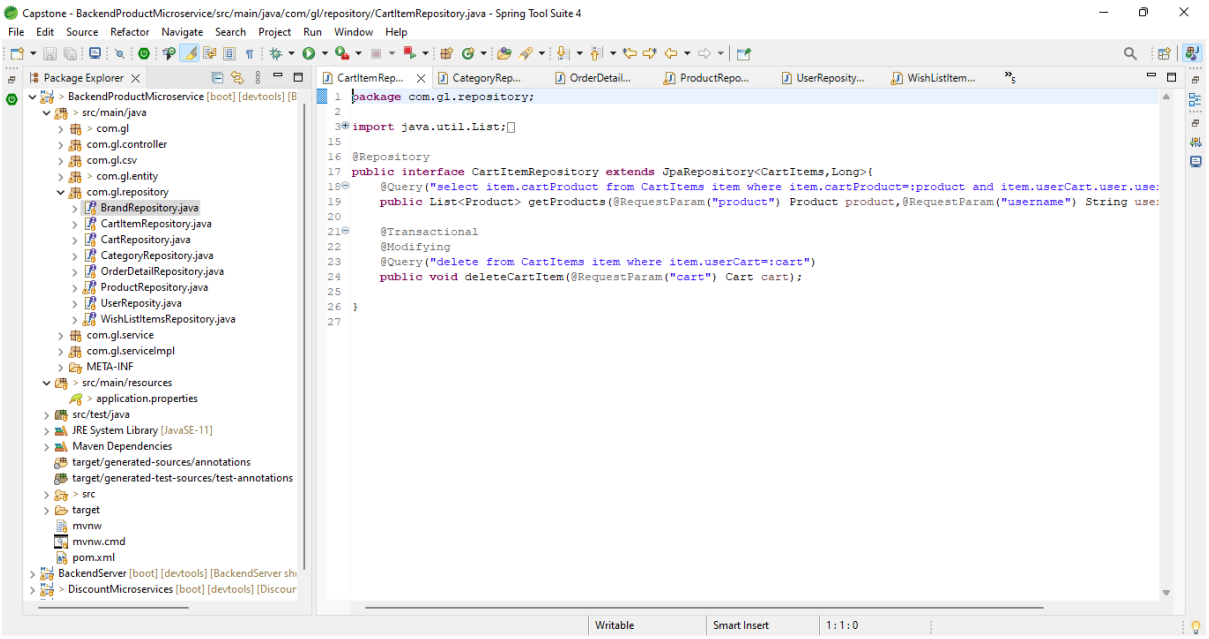
Wishlist Items



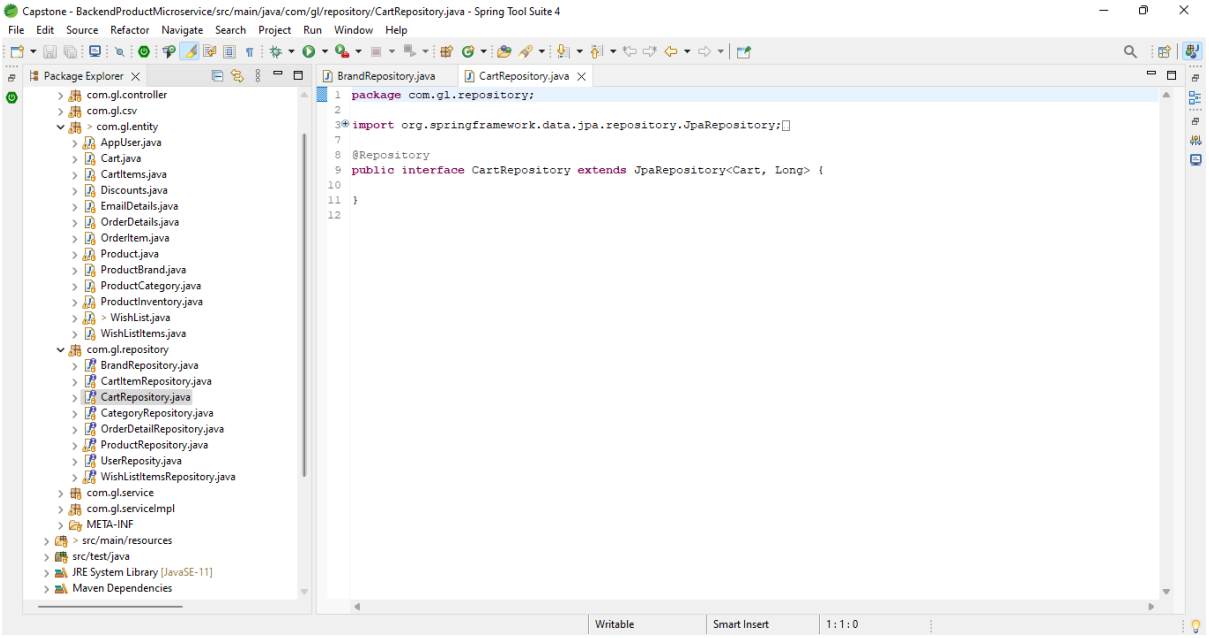
Brand Repository



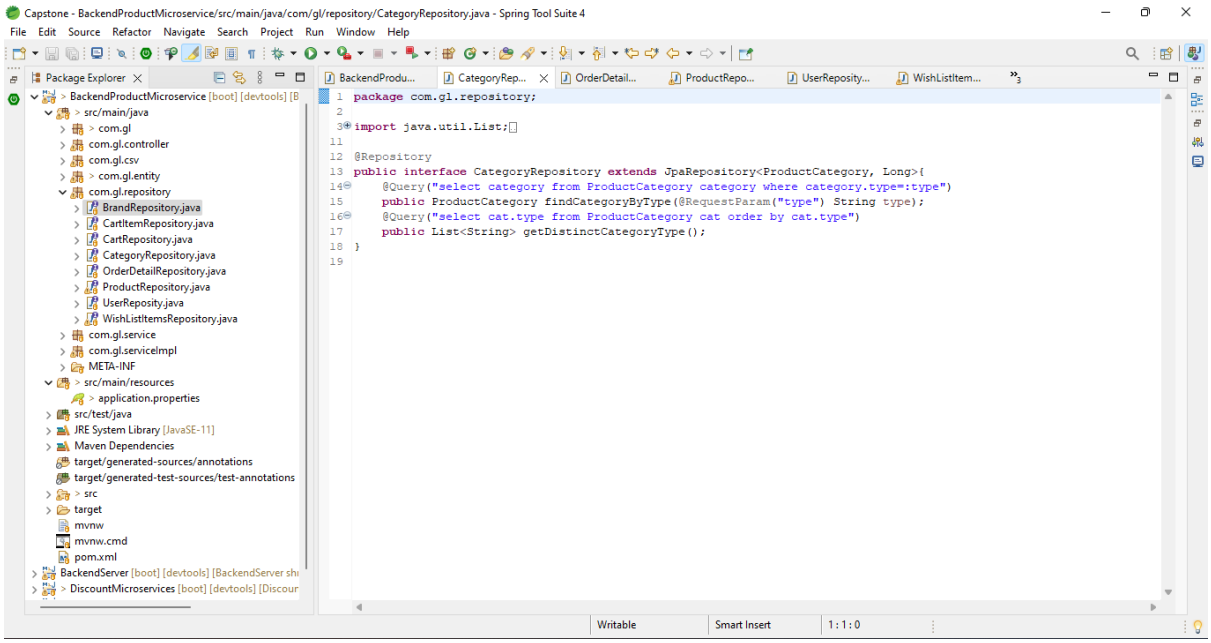
Cart Item Repository



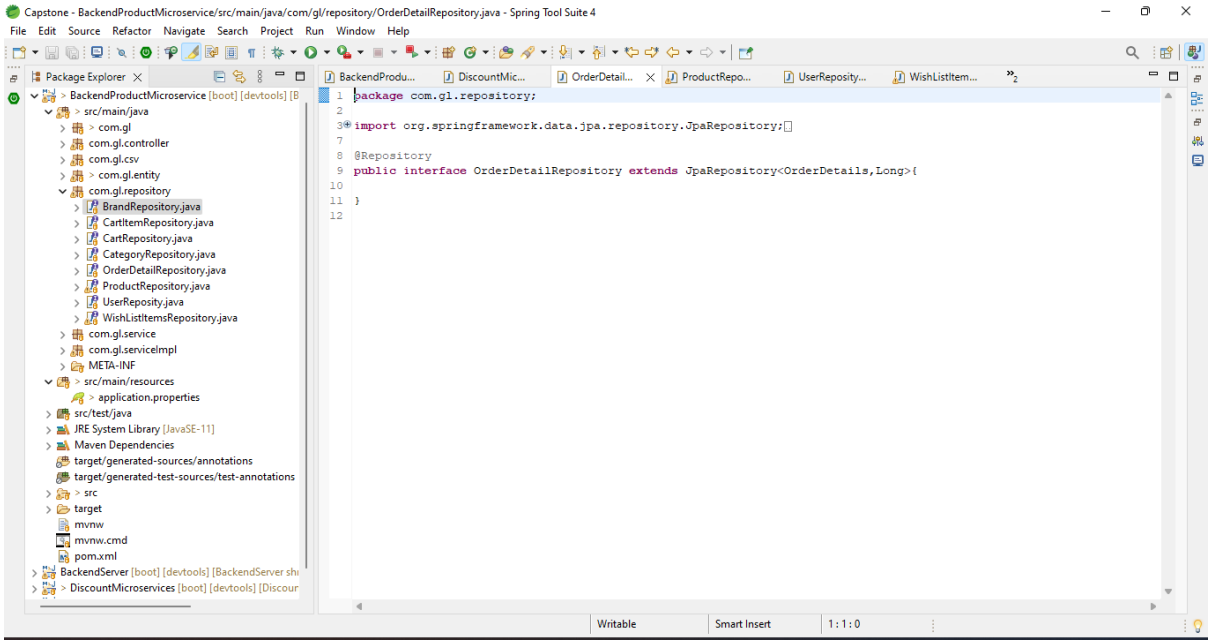
Cart Repository



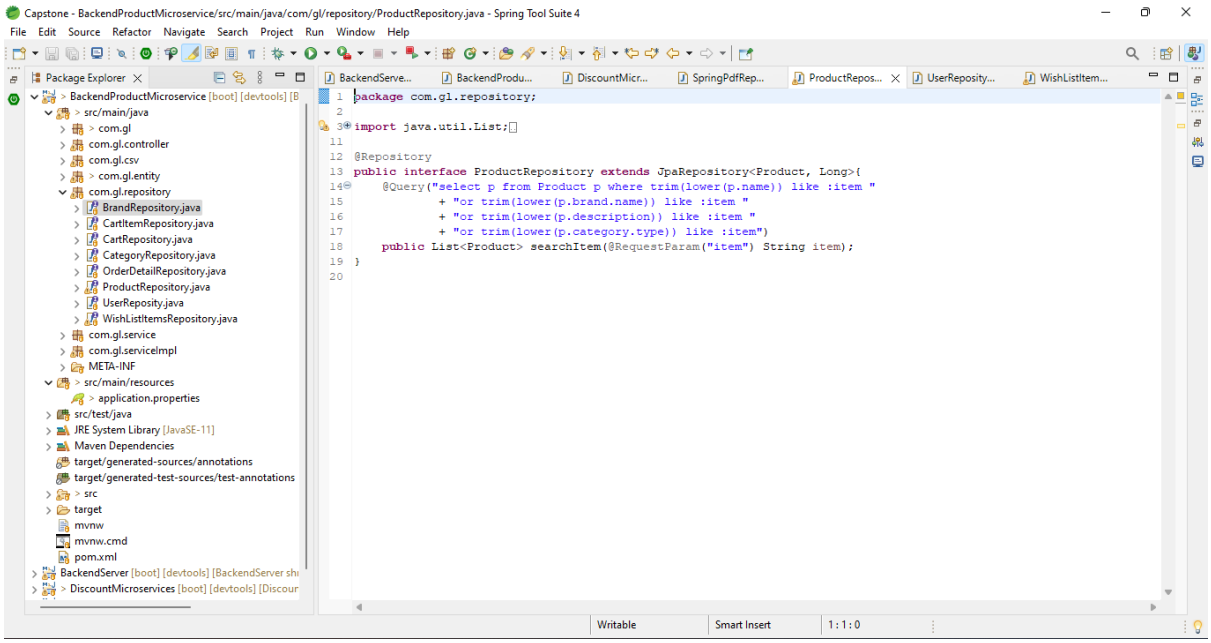
Category Repository



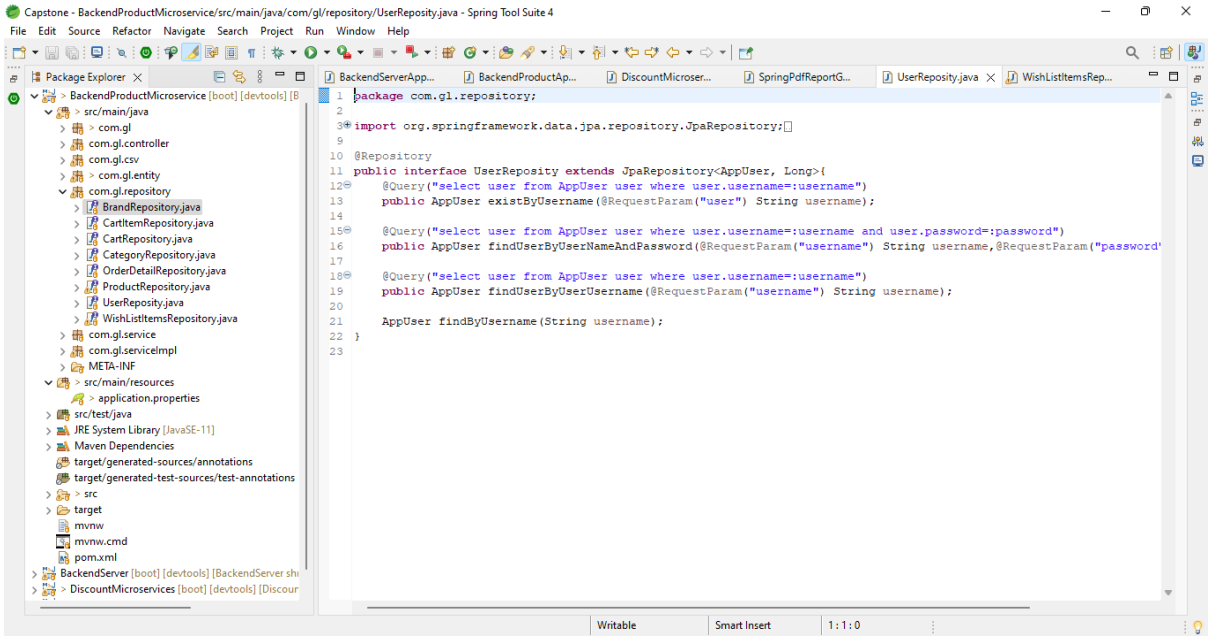
Order Detail Repository



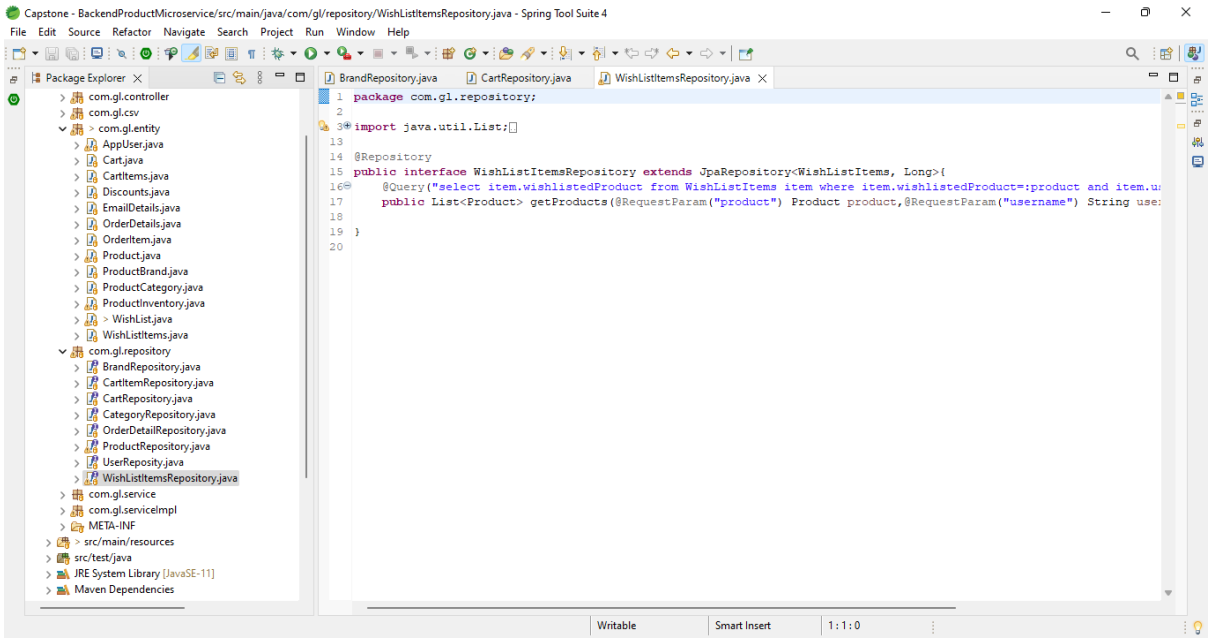
Product Repository



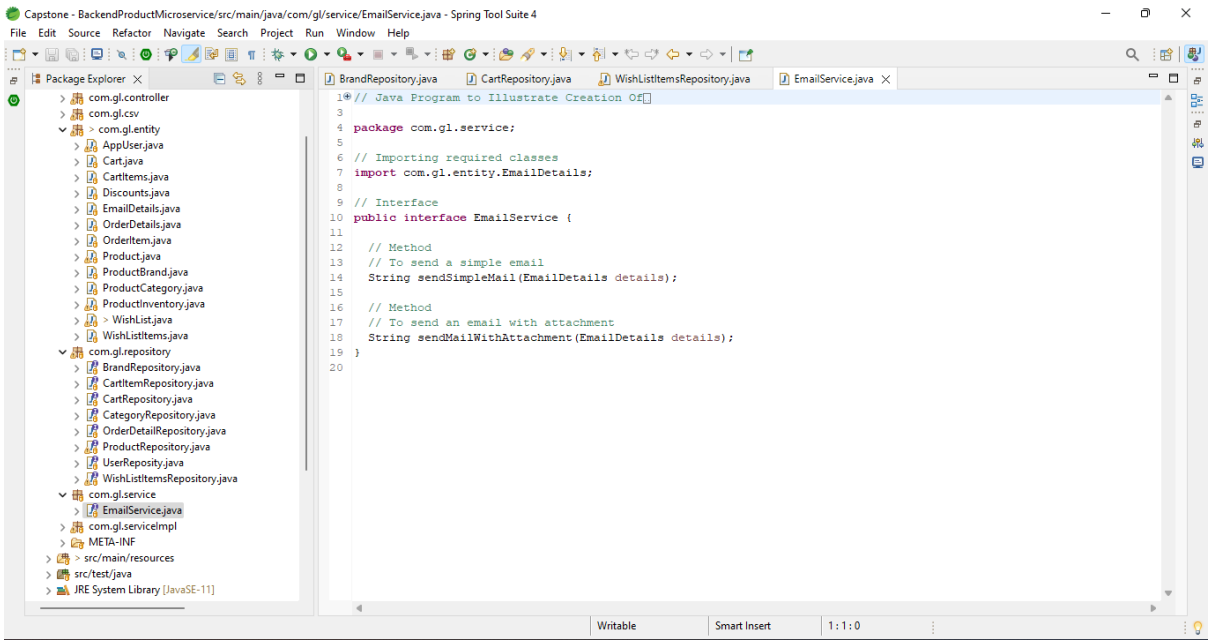
User Repository



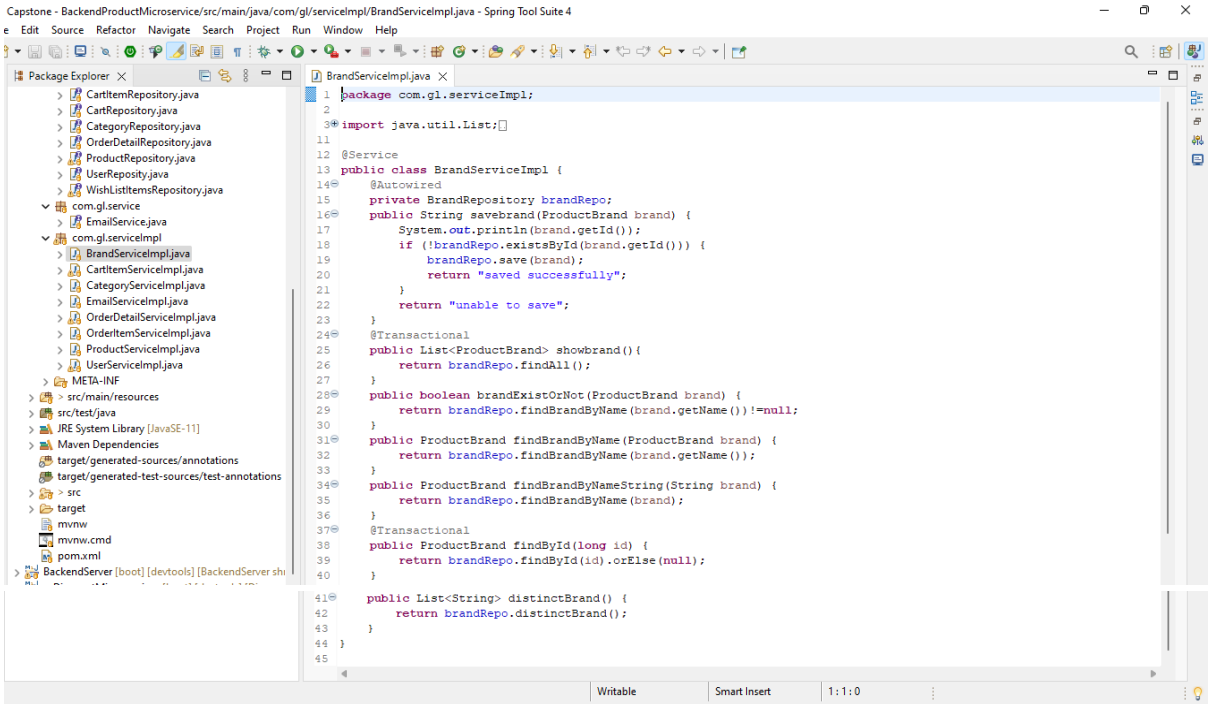
Wishlist Items Repository



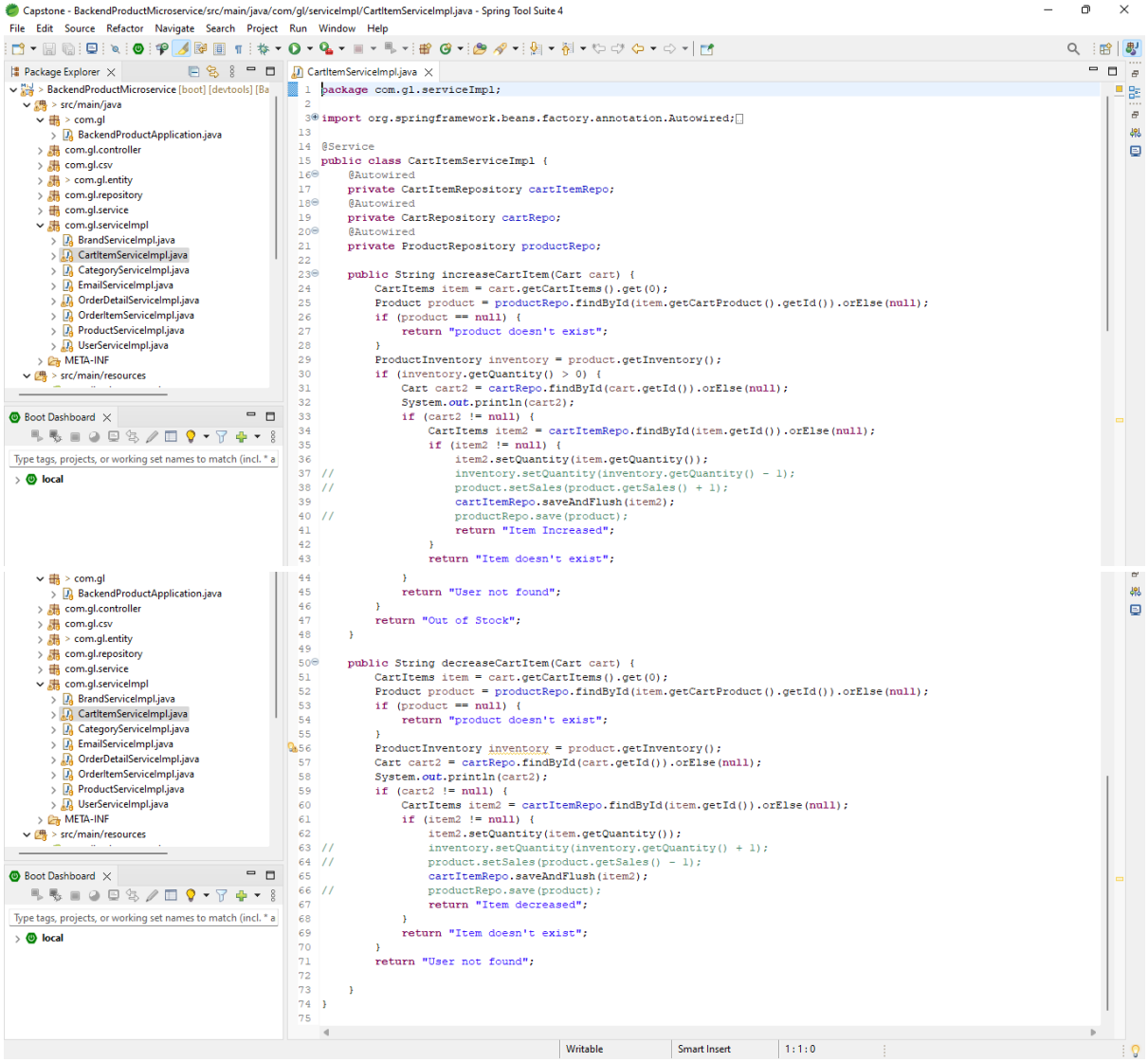
Email Service



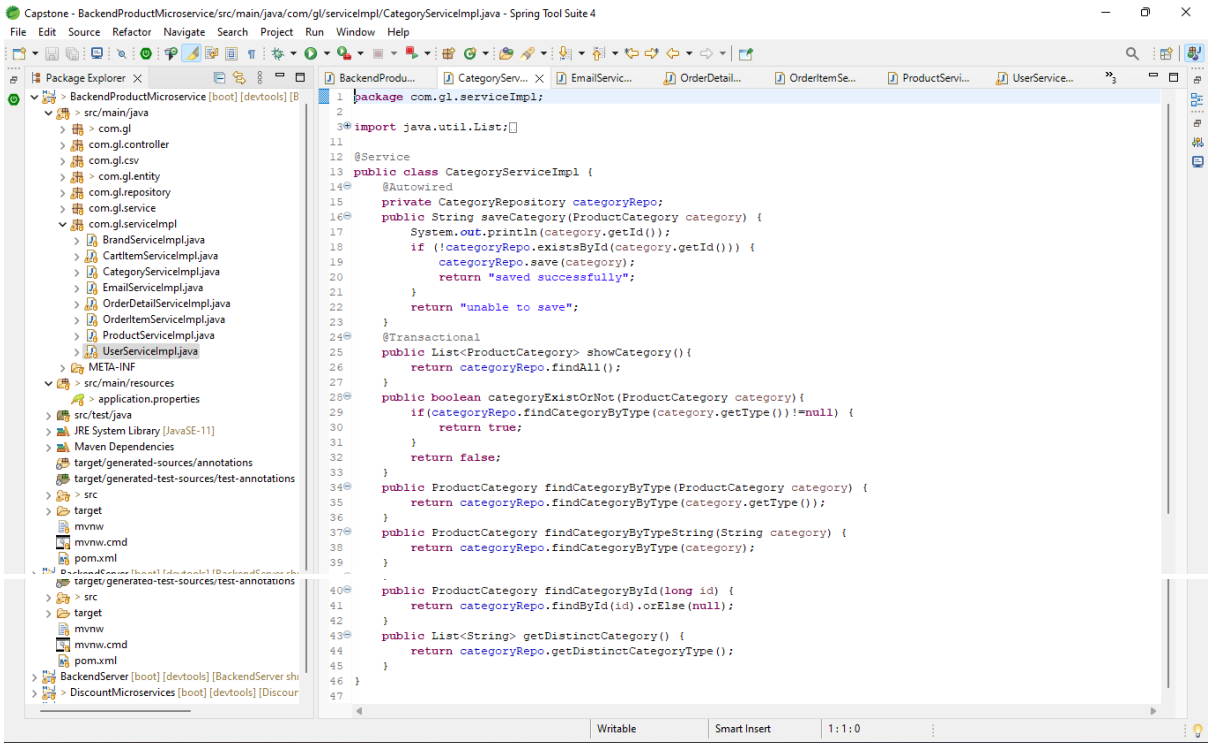
Brand Service Implementation



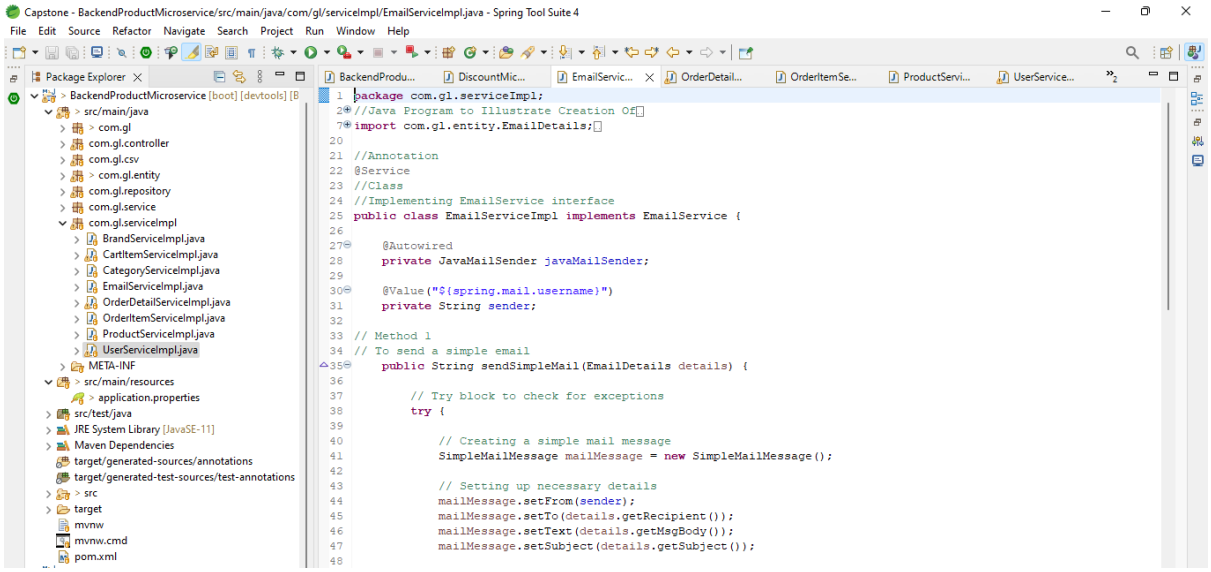
Cart Items Service Implementation

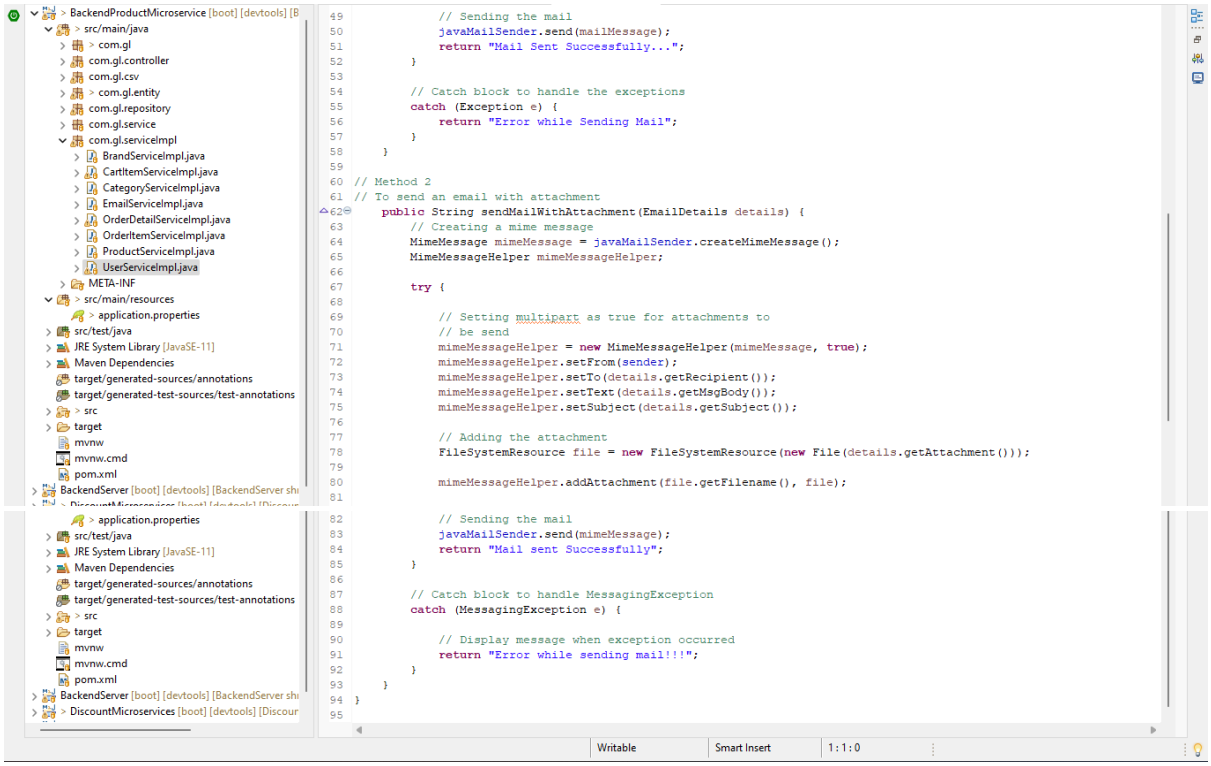


Category Service Implementation



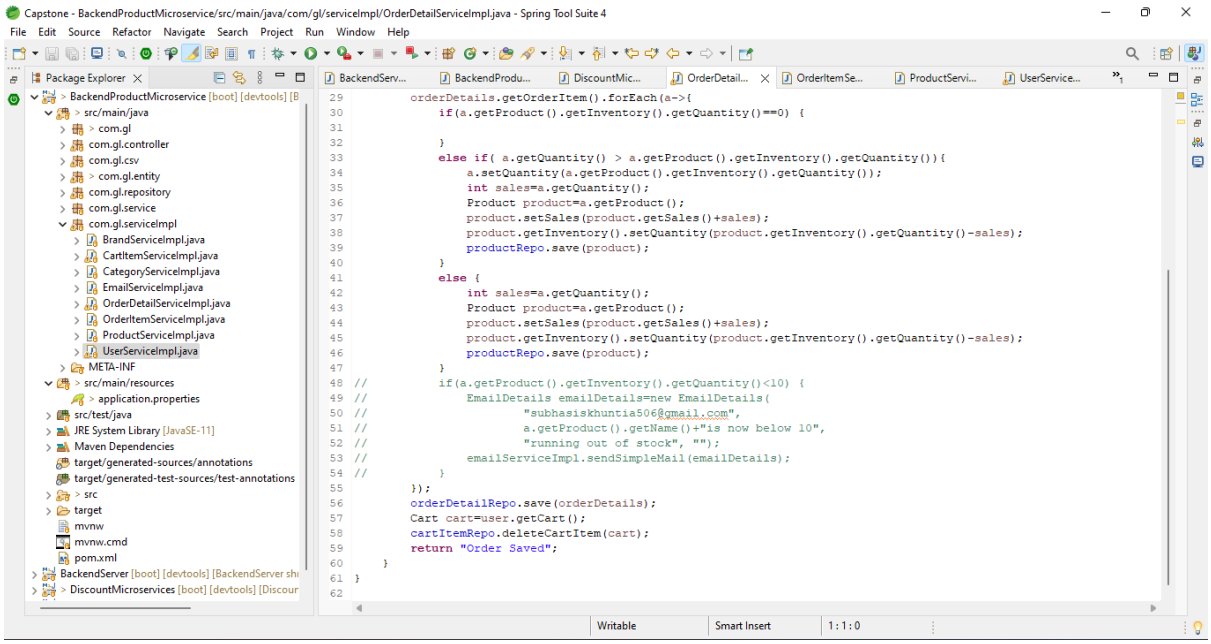
Email Service Implementation





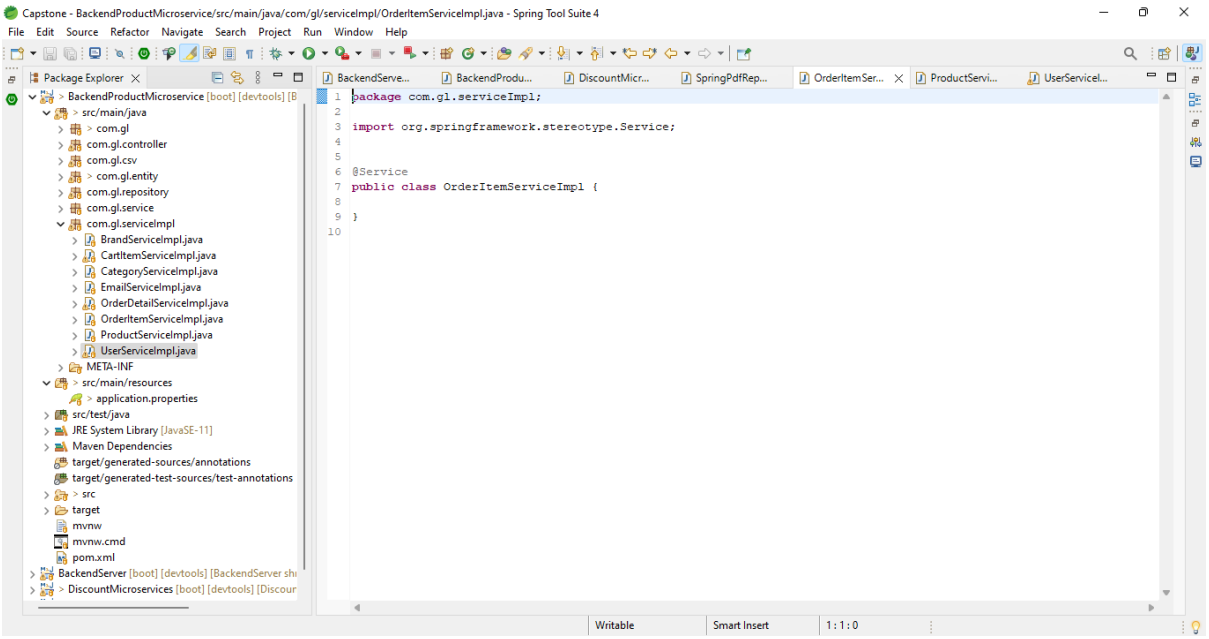
```
            // Sending the mail
            javaMailSender.send(mailMessage);
            return "Mail Sent Successfully...";
        }

        // Catch block to handle the exceptions
        catch (Exception e) {
            return "Error while Sending Mail";
        }
    }

// Method 2
// To send an email with attachment
    public String sendMailWithAttachment(EmailDetails details) {
        // Creating a mime message
        MimeMessage mimeMessage = javaMailSender.createMimeMessage();
        MimeMessageHelper mimeMessageHelper;

        try {

            // Setting multipart as true for attachments to
            // be send
            mimeMessageHelper = new MimeMessageHelper(mimeMessage, true);
            mimeMessageHelper.setFrom(sender);
            mimeMessageHelper.setTo(details.getRecipient());
            mimeMessageHelper.setText(details.getMsgBody());
            mimeMessageHelper.setSubject(details.getSubject());

            // Adding the attachment
            FileSystemResource file = new FileSystemResource(new File(details.getAttachment()));

            mimeMessageHelper.addAttachment(file.getFilename(), file);


            // Sending the mail
            javaMailSender.send(mimeMessage);
            return "Mail sent Successfully";
        }

        // Catch block to handle MessagingException
        catch (MessagingException e) {

            // Display message when exception occurred
            return "Error while sending mail!!!";
        }
    }
}
```

Order Detail Service Implementation

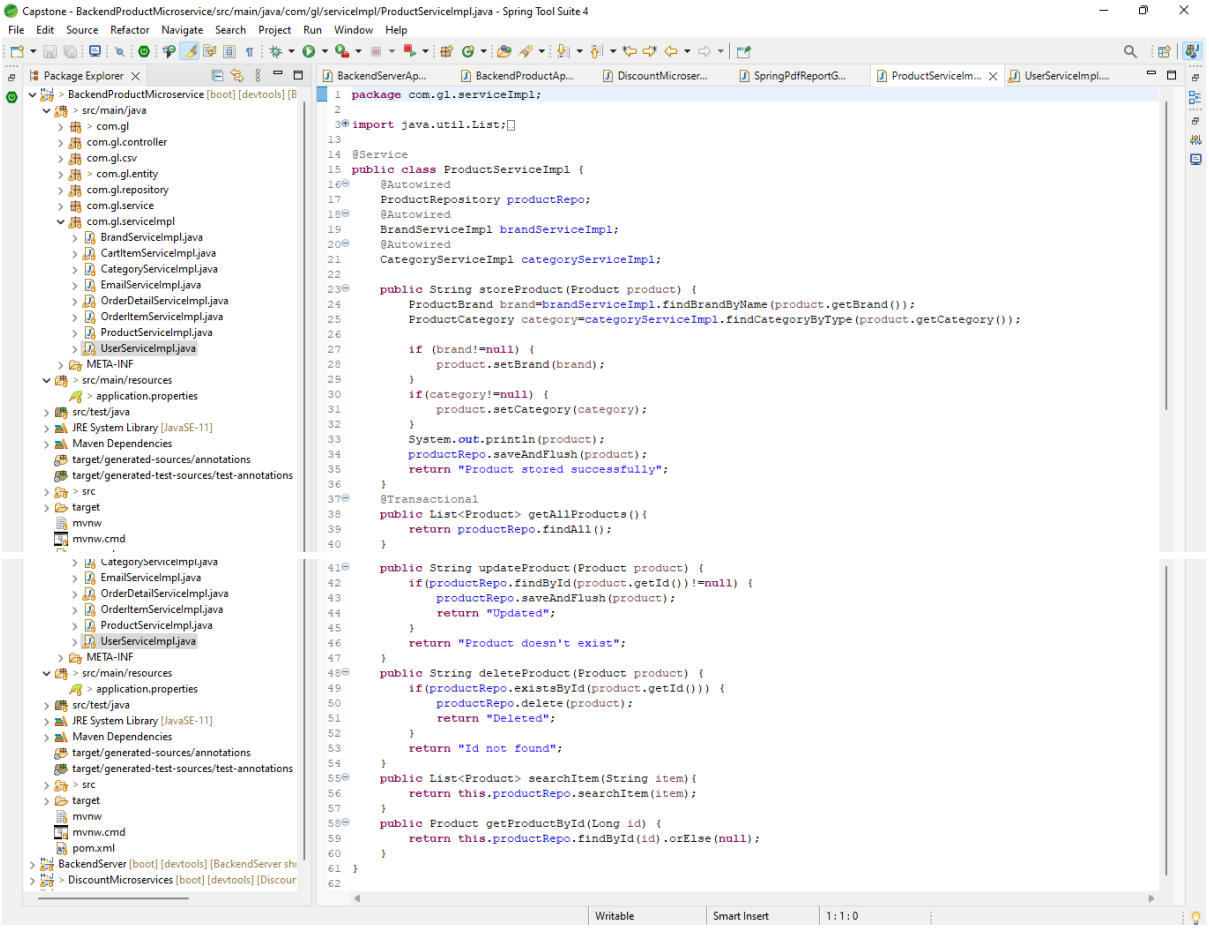


```
        orderDetails.getOrderItem().forEach(a->{
            if(a.getProduct().getInventory().getQuantity()==0) {

            }
            else if( a.getQuantity() > a.getProduct().getInventory().getQuantity()){
                a.setQuantity(a.getProduct().getInventory().getQuantity());
                int sales=a.getQuantity();
                Product product=a.getProduct();
                product.setSales(product.getSales()+sales);
                product.getInventory().setQuantity(product.getInventory().getQuantity()-sales);
                productRepo.save(product);
            }
            else {
                int sales=a.getQuantity();
                Product product=a.getProduct();
                product.setSales(product.getSales()+sales);
                product.getInventory().setQuantity(product.getInventory().getQuantity()-sales);
                productRepo.save(product);
            }
//          if(a.getProduct().getInventory().getQuantity()<10) {
//              EmailDetails emailDetails=new EmailDetails(
//                      "subhasiskhuntia506@gmail.com",
//                      a.getProduct().getName()+"is now below 10",
//                      "running out of stock", "");
//              emailServiceImpl.sendSimpleMail(emailDetails);
//          }
        });
        orderDetailRepo.save(orderDetails);
        Cart cart=user.getCart();
        cartItemRepo.deleteCartItem(cart);
        return "Order Saved";
    }
}
```

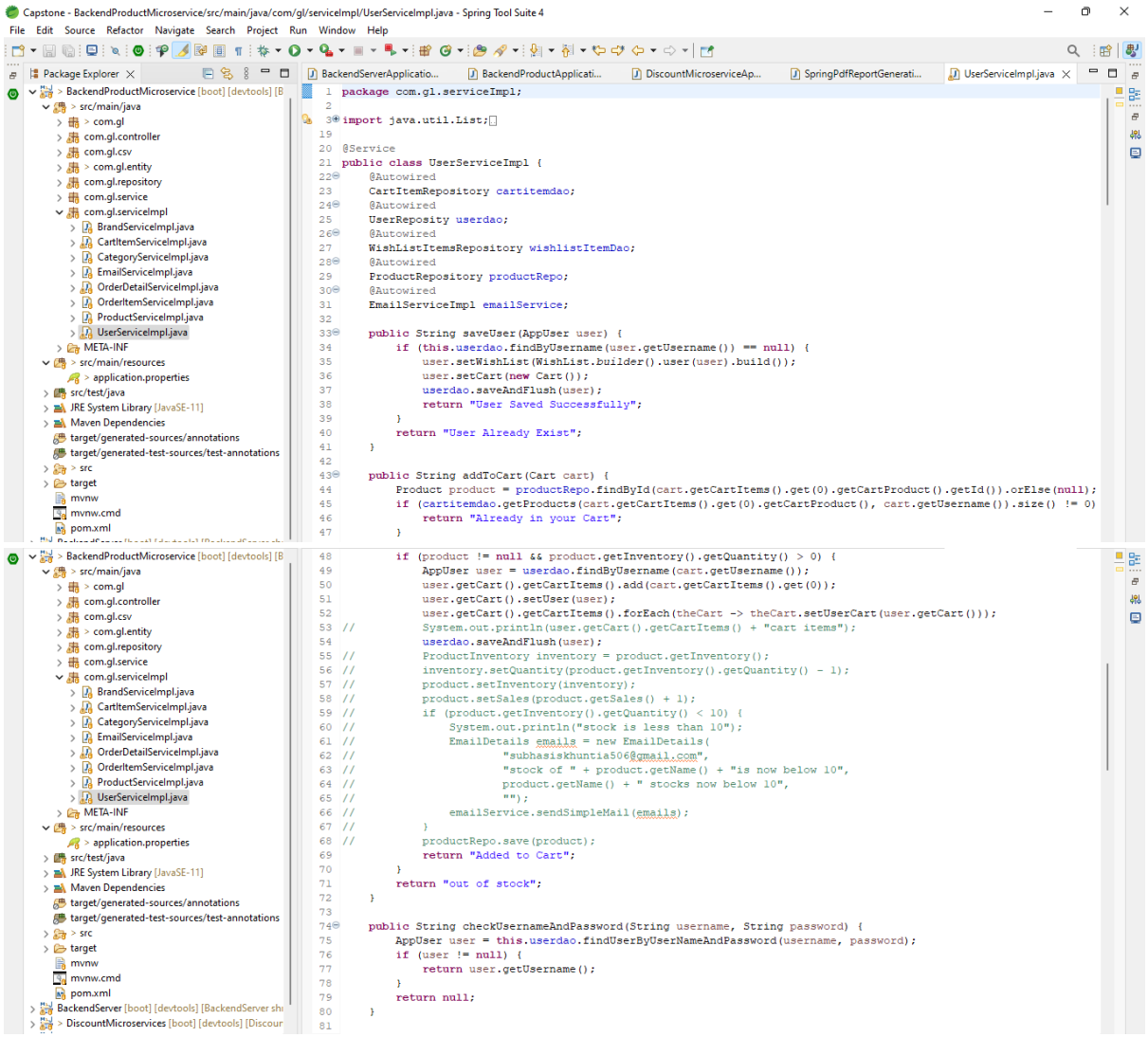
Order Item Service Implementation

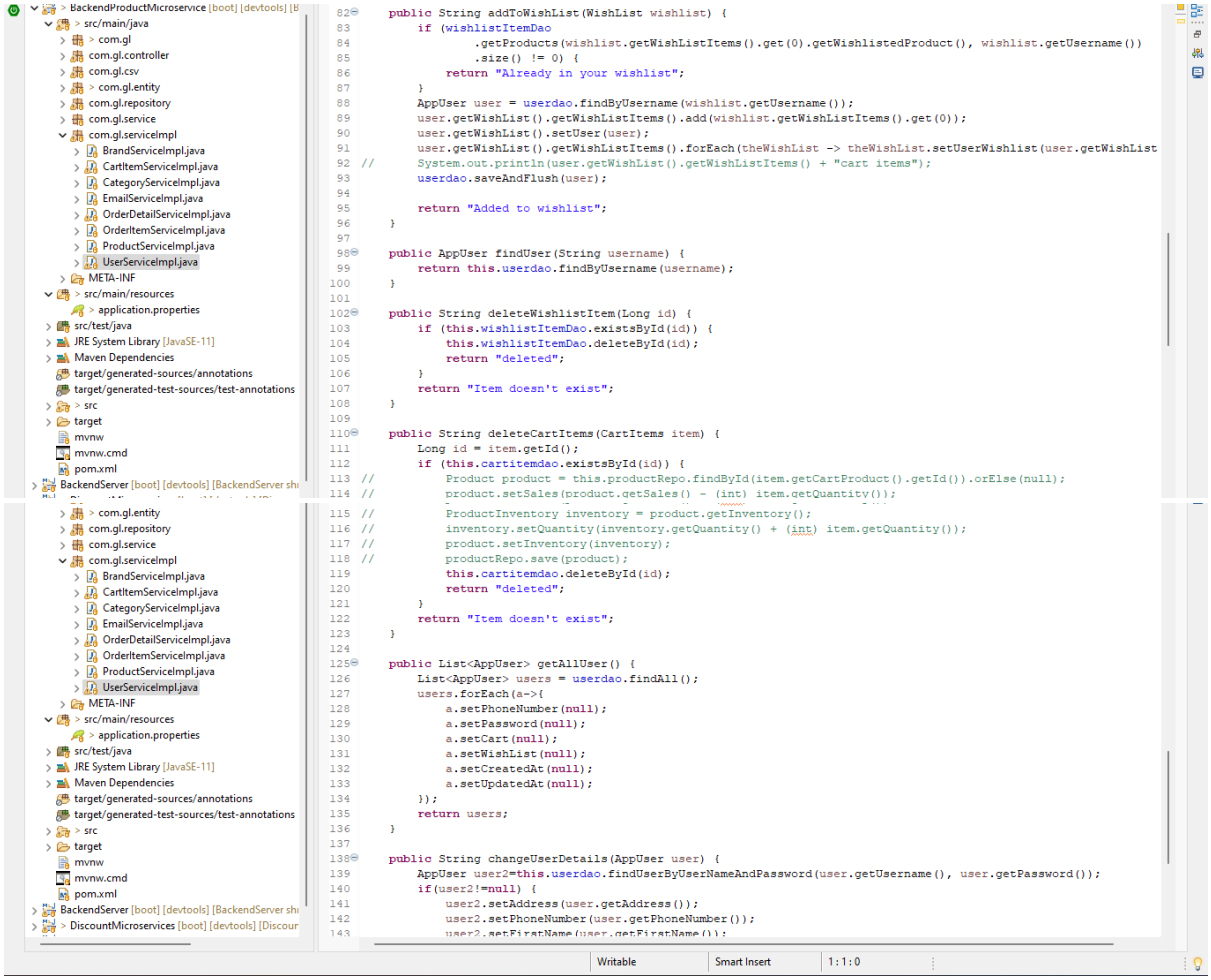


Product Service Implementation



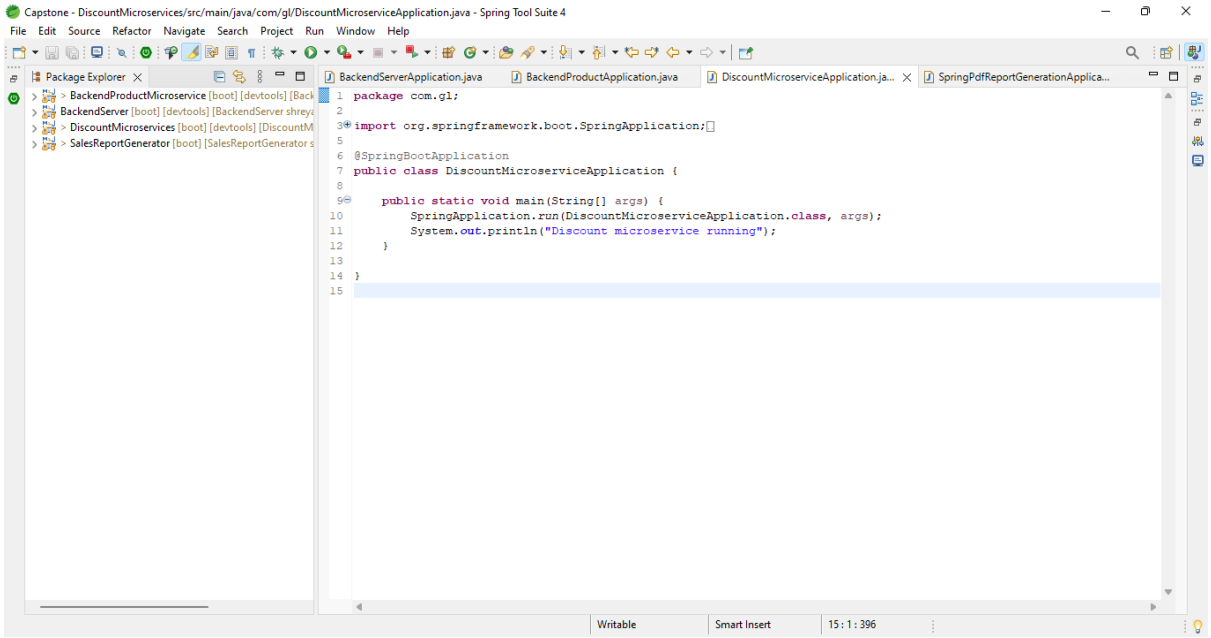
User Service Implementation






```
    public String addToWishList(WishList wishlist) {
        if (wishlistItemDao
                .getProducts(wishlist.getWishListItems().get(0).getWishlistedProduct(), wishlist.getUsername())
                .size() != 0) {
            return "Already in your wishlist";
        }
        AppUser user = userdao.findByUsername(wishlist.getUsername());
        user.getWishList().getWishListItems().add(wishlist.getWishListItems().get(0));
        user.getWishList().setUser(user);
        user.getWishList().getWishListItems().forEach(theWishList -> theWishList.setUserWishlist(user.getWishList
//      System.out.println(user.getWishList().getWishListItems() + "cart items");
        userdao.saveAndFlush(user);

        return "Added to wishlist";
    }

    public AppUser findUser(String username) {
        return this.userdao.findByUsername(username);
    }

    public String deleteWishlistItem(Long id) {
        if (this.wishlistItemDao.existsById(id)) {
            this.wishlistItemDao.deleteById(id);
            return "deleted";
        }
        return "Item doesn't exist";
    }

    public String deleteCartItems(CartItems item) {
        Long id = item.getId();
        if (this.cartitemdao.existsById(id)) {
//          Product product = this.productRepo.findById(item.getCartProduct().getId()).orElse(null);
//          product.setSales(product.getSales() - (int) item.getQuantity());

//          ProductInventory inventory = product.getInventory();
//          inventory.setQuantity(inventory.getQuantity() + (int) item.getQuantity());
//          product.setInventory(inventory);
//          productRepo.save(product);
            this.cartitemdao.deleteById(id);
            return "deleted";
        }
        return "Item doesn't exist";
    }

    public List<AppUser> getAllUser() {
        List<AppUser> users = userdao.findAll();
        users.forEach(a->{
            a.setPhoneNumber(null);
            a.setPassword(null);
            a.setCart(null);
            a.setWishList(null);
            a.setCreatedAt(null);
            a.setUpdatedAt(null);
        });
        return users;
    }

    public String changeUserDetails(AppUser user) {
        AppUser user2=this.userdao.findUserByUserNameAndPassword(user.getUsername(), user.getPassword());
        if(user2!=null) {
            user2.setAddress(user.getAddress());
            user2.setPhoneNumber(user.getPhoneNumber());
```

Discount Microservice




```
package com.gl;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class DiscountMicroserviceApplication {

    public static void main(String[] args) {
        SpringApplication.run(DiscountMicroserviceApplication.class, args);
        System.out.println("Discount microservice running");
    }

}
```

Sales Report Generator