

sportyshoes.com

Sports Shoes Portal Online

Prototype of the Application

Name: Amit Yadav

GitHub: <https://github.com/amityadav872699/Project-Phase-3-DevOps-sportyshoes>

The prototype of the application starts from the backend, and it can also directly start from the project folder. This portal allows us to do shoes management across administrator and provide CRUD methodologies across admin side. This prototype is built through various controllers which are auto wired with Repository, models, Interfaces, Exception and Services.

The implementation is done with the help of Hibernate, Maven, Spring Boot, MySql, JDK 8, IntelliJ and for testing API is done through Postman and Swagger-UI for Documentation.

Sprint Planning

The Implementation is done in five sprints which are mentioned below:

Sprint 1:

- Clarify the specification and requirements.
- Implement a model of "Product" which are available in stock with their respective attributes such as category, price, shoe size, company, origin.
- Implement a service and repository of same entity Product to perform CRUD operation on the database.
- Creating a controller through Rest API for testing the functions of all the java classes (repository, service, model, controller).

Sprint 2:

- Similarly, creating the same java classes for products which have been purchased by the customer named "PurchasedProduct".
- And the corresponding same structure is defined for customers who have bought the products using this portal. Java Class files such as Customer.java, ICustomerService.java, CustomerServiceImpl, CustomerRepository.java, CustomerController.
- Implementing a relationship between "Customer" and "PurchasedProduct" using One-To-Many associations.

Sprint 3:

- Implement a model "Admin" for registering and logging entries in the database.
- Implement functionality for changing password in admin section which consist of new password and new email.
- Implementing a functionality in which admin can fetch all the data of the customer and the available product.

- Implementing another function for procuring the records of all users (Discussed in Sprint 4) who have signed up in this portal.

Sprint 4:

- Developing the user package containing the same structure (Repository, model, controller, service).
- The user can select the product without inserting any of the user details for purchasing and being registered in the database.
- Implementing a functionality on user side where user can register, login or change its details in the database.

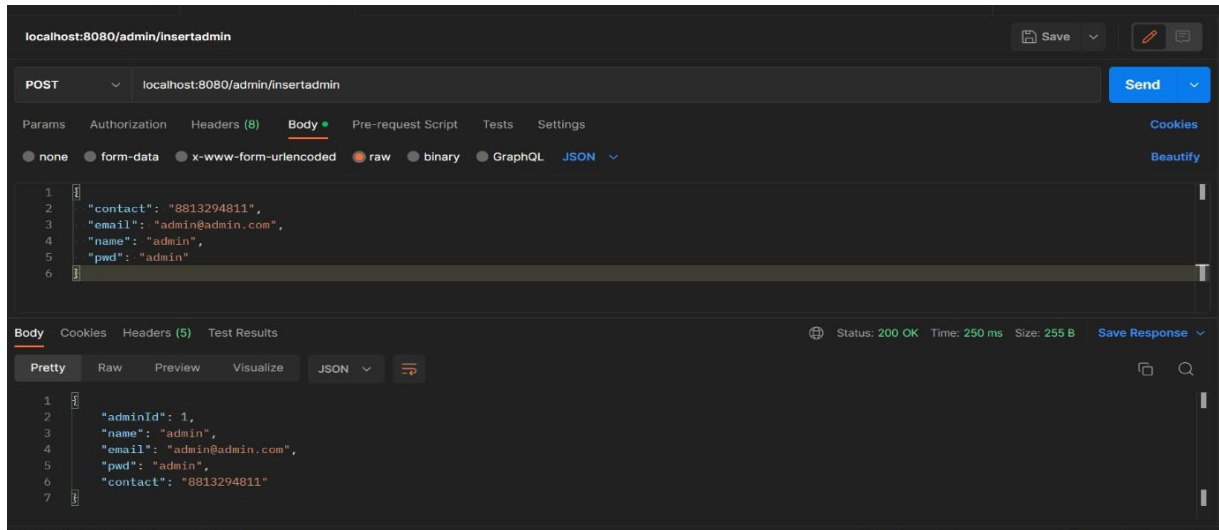
Sprint 5:

- Implementing a functionality on user and customer side where a person can purchase more than one product without the need of inserting its details and does not change the state of its content in the database, relationship.
- Implementing a functionality where admin can retrieve all the purchased products through the use of foreign key in custom queries.
- Documentation

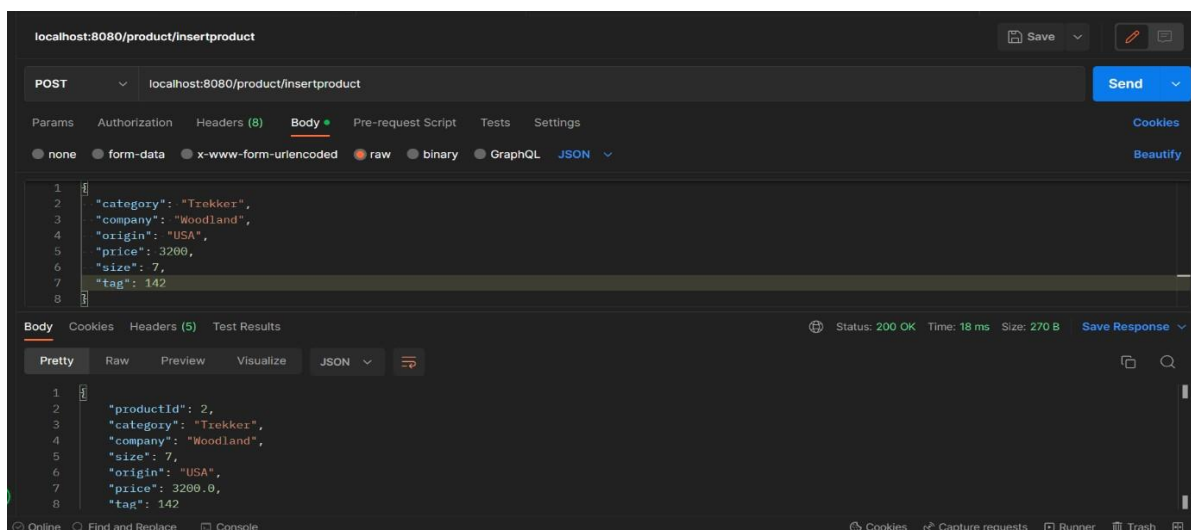
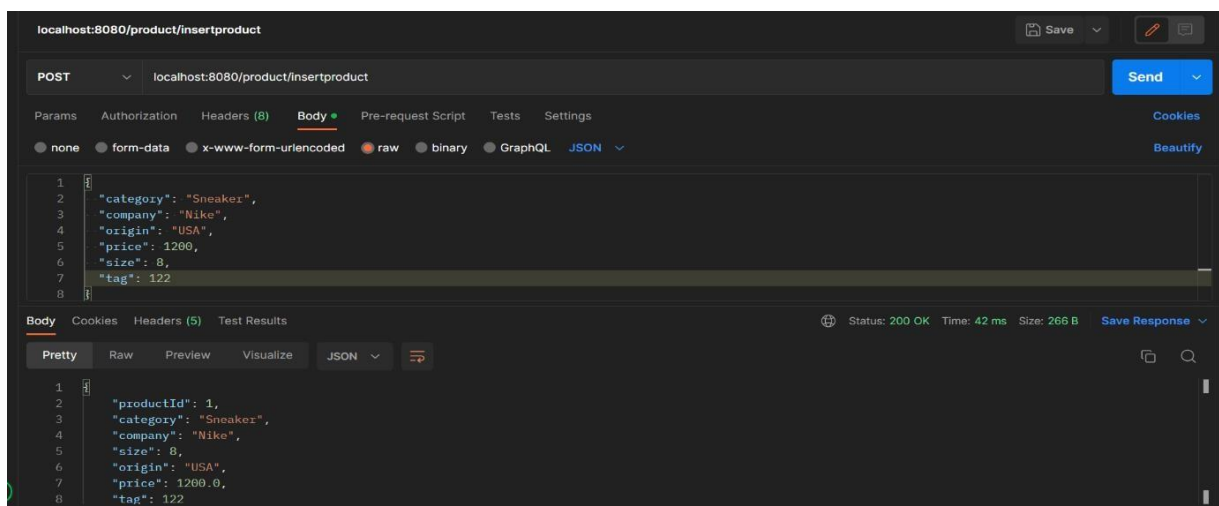
Documentation of the functionality:

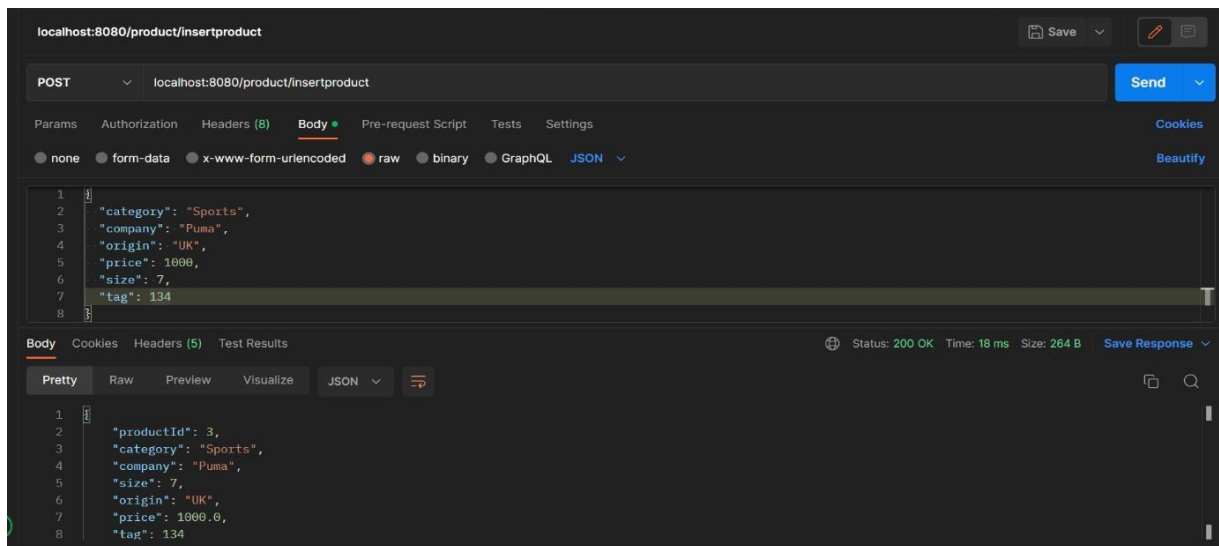
Here are some of the screenshots of testing API through various controllers.

1: Login And Register (Admin Section)

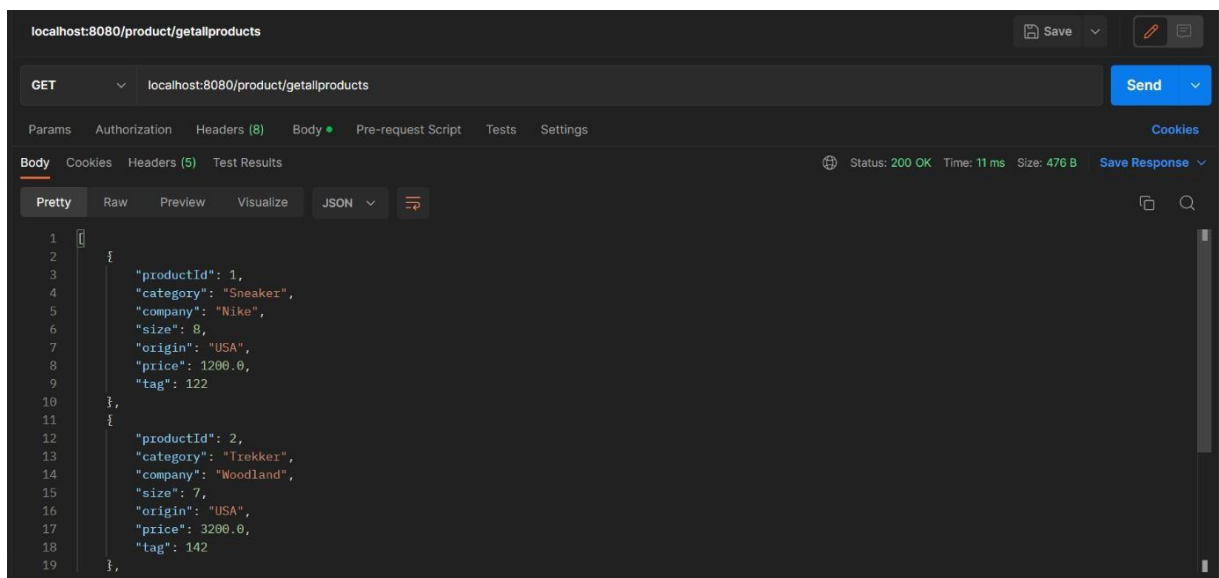


2: Inserting Product (Admin Section)





3: Display All Available Products (Admin Section)



4: Purchasing Product through productid (Customer Section)

POST /customer/insertcustomer/{id} insertCustomer

Parameters Cancel

Name	Description
id * required integer(\$int64) (path)	id <input type="text" value="1"/>
newUser * required object (body)	newUser Edit Value Model <pre> { "contact": 1391391312, "customerName": "John", "date": "20-01-2023", "email": "john@gmail.com", "list": [{ "category": "string", "company": "string", "origin": "string", "price": 0, "productId": 0, "size": 0, "tag": 0 }], "totalPrice": 0 } </pre>

http://localhost:9080/customer/insertcustomer/1

Server response

Code	Details
200	<div><p>Response body</p><pre>{ "customerId": 1, "customerName": "John", "email": "john@gmail.com", "contact": 2391391312, "date": "20-01-2023", "totalPrice": 1200, "list": [{ "productId": 1, "category": "Sneaker", "company": "Nike", "size": 8, "origin": "USA", "price": 1200, "tag": 122 }] }</pre>Download</div> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Sun29 Jan 2023 12:31:08 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

5: Adding more Products through customerId and Product Id (Customer Section)

custid * required
integer(int64)
(path)

prodid * required
integer(int64)
(path)

user * required
object
(body)

1

2

User

Edit Value | Model

```
{
  "contact": 0,
  "customerId": 0,
  "customerName": "string",
  "date": "20-01-2023",
  "email": "string",
  "list": [
    {
      "category": "string",
      "company": "string",
      "origin": "string",
      "price": 0,
      "productId": 0,
      "size": 0,
      "tag": 0
    }
  ],
  "totalPrice": 0
}
```

200

Response body

```
[
  {
    "customerId": 2,
    "customerName": "John",
    "email": "john@gmail.com",
    "contact": 2391391312,
    "date": "20-01-2023",
    "totalPrice": 4400,
    "list": [
      {
        "productId": 3,
        "category": "Trekker",
        "company": "Woodland",
        "size": 7,
        "origin": "USA",
        "price": 3200,
        "tag": 142
      },
      {
        "productId": 2,
        "category": "Sneaker",
        "company": "Nike",
        "size": 8,
        "origin": "USA",
        "price": 1200,
        "tag": 122
      }
    ]
  }
]
```

[Download](#)

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun29 Jan 2023 12:32:42 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

6: Register And Login (User Section)

POST /user/signup InsertUser

Parameters Cancel

Name	Description
newUser * required object (body)	newUser Edit Value Model

```
{
  "userContact": 2345910910,
  "userEmail": "jack@gmail.com",
  "userName": "Jack",
  "userPwd": "12345"
}
```

Cancel

7: Inserting Product through userId and productId (User Section)

http://localhost:8080/user/insertproductuser/1/1

Server response

Code	Details
200	<p>Response body</p> <pre>{ "customerId": 3, "customerName": "Jack", "email": "jack@gmail.com", "contact": 2345910910, "date": "string", "totalPrice": 1200, "list": [{ "productId": 4, "category": "Sneaker", "company": "Nike", "size": 8, "origin": "USA", "price": 1200, "tag": 122 }] }</pre> <p>Download</p> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Sun29 Jan 2023 12:34:33 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

8: Adding more products in same userId/customerId (User Section)

custid * required
integer(\$int64)
(path)

prodid * required
integer(\$int64)
(path)

user * required
object
(body)

custid

3

prodid

2

user

Edit Value | Model

```
{
  "contact": 0,
  "customerId": 0,
  "customerName": "string",
  "date": "22-01-2023",
  "email": "string",
  "list": [
    {
      "category": "string",
      "company": "string",
      "origin": "string",
      "price": 0,
      "productId": 0,
      "size": 0,
      "tag": 0
    }
  ],
  "totalPrice": 0
}
```

Cancel

9: Other Functions in Admin Section

admin-controller Admin Controller			▼
GET	/admin/getallcustomerbydate/{date}	getAllCustomersByDateAndCategory	
GET	/admin/getallcustomerbyname/{custname}	findByCustomerName	
GET	/admin/getallproductsbycategory/{category}	getAllProductsByCategory	
GET	/admin/getallusers	getAllUsers1	
GET	/admin/getallvalidcustomers	getAllUsers	
POST	/admin/insertadmin	insertAdmin	
PUT	/admin/updateadminbyid/{adminid}	updateAdmin	

product-controller Product Controller			▼
DELETE	/product/deleteproduct/{productid}	deleteProduct	
GET	/product/getallproducts	getAllProducts	
GET	/product/getproduct/{productid}	getProduct	
POST	/product/insertproduct	insertProduct	
PUT	/product/updateproductbyid/{productid}	updateProduct	

10: Other Functions in Customer Section

customer-controller Customer Controller			▼
PUT	/customer/addcustomerproducts/{custid}/{prodid}	updateProductInCustomer	
DELETE	/customer/deletecustomer/{userid}	deleteCustomer	
GET	/customer/getallcustomers	getAllCustomers	
GET	/customer/getcustomer/{userid}	getCustomer	
POST	/customer/insertcustomer/{id}	insertCustomer	
PUT	/customer/updatecustomerbyid/{userid}	updateCustomer	

purchased-product-controller Purchased Product Controller			▼
DELETE	/purchased/deletepurchasedproduct/{productid}	deletePurchasedProduct	
GET	/purchased/getallpurchasedproducts	getAllPurchasedProducts	
GET	/purchased/getpurchasedproduct/{productid}	getProduct	
POST	/purchased/insertpurchasedproduct	insertProduct	
PUT	/purchased/updatepurchasedproductbyid/{productid}	updatePurchasedProduct	

11: Other Functions in User Section

user-controller User Controller			▼
PUT	/user/addmoreuserproducts/{custid}/{prodid}	updateProductInCustomer	
GET	/user/getproducts/{userid}	getProducts	
POST	/user/insertproductinuser/{userid}/{prodid}	insertProductUserInDB	
POST	/user/signup	insertUser	
PUT	/user/updateuserbyid/{userid}	updateUser	

12: Database (MySQL)

```
mysql> select * from product;
```

product_id	product_category	product_company	product_origin	product_price	product_size	product_tag
1	Sneaker	Nike	USA	1200	8	122
2	Trekker	Woodland	USA	3200	7	142
3	Sports	Puma	UK	1000	7	134

```
3 rows in set (0.01 sec)
```

```
mysql>
```

```
mysql> select * from customer;
```

customer_id	customer_contact	customer_name	purchased_date	customer_email	total_price
2	2391391312	John	20-01-2023	john@gmail.com	4400
4	2345910910	Jack	22-01-2023	jack@gmail.com	4400

```
2 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> select * from customer;
```

customer_id	customer_contact	customer_name	purchased_date	customer_email	total_price
2	2391391312	John	20-01-2023	john@gmail.com	4400
4	2345910910	Jack	22-01-2023	jack@gmail.com	4400

```
2 rows in set (0.00 sec)
```

```
mysql> select * from user;
```

user_id	user_contact	user_email	user_name	user_pwd
1	2345910910	jack@gmail.com	Jack	12345

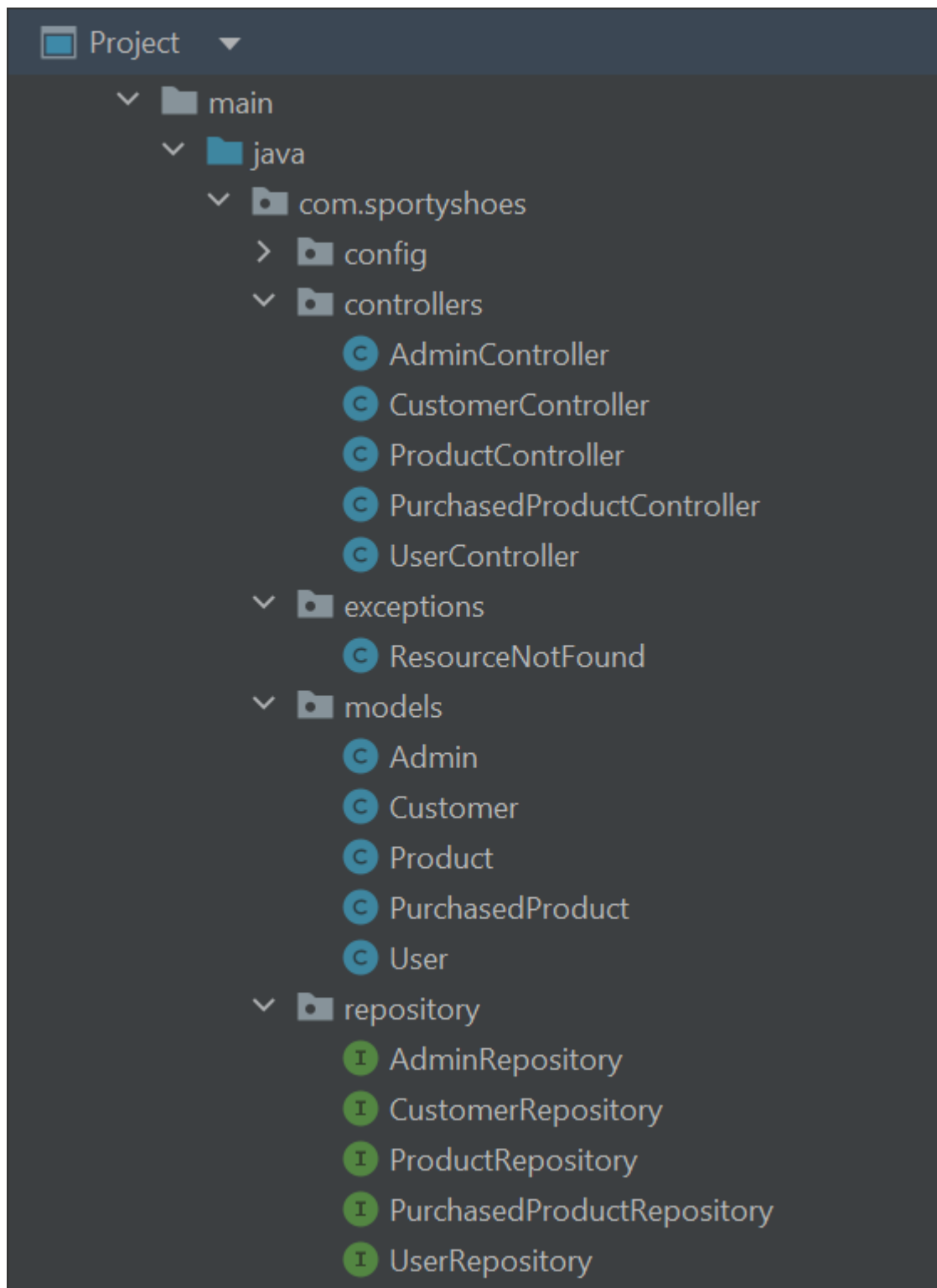
```
1 row in set (0.00 sec)
```

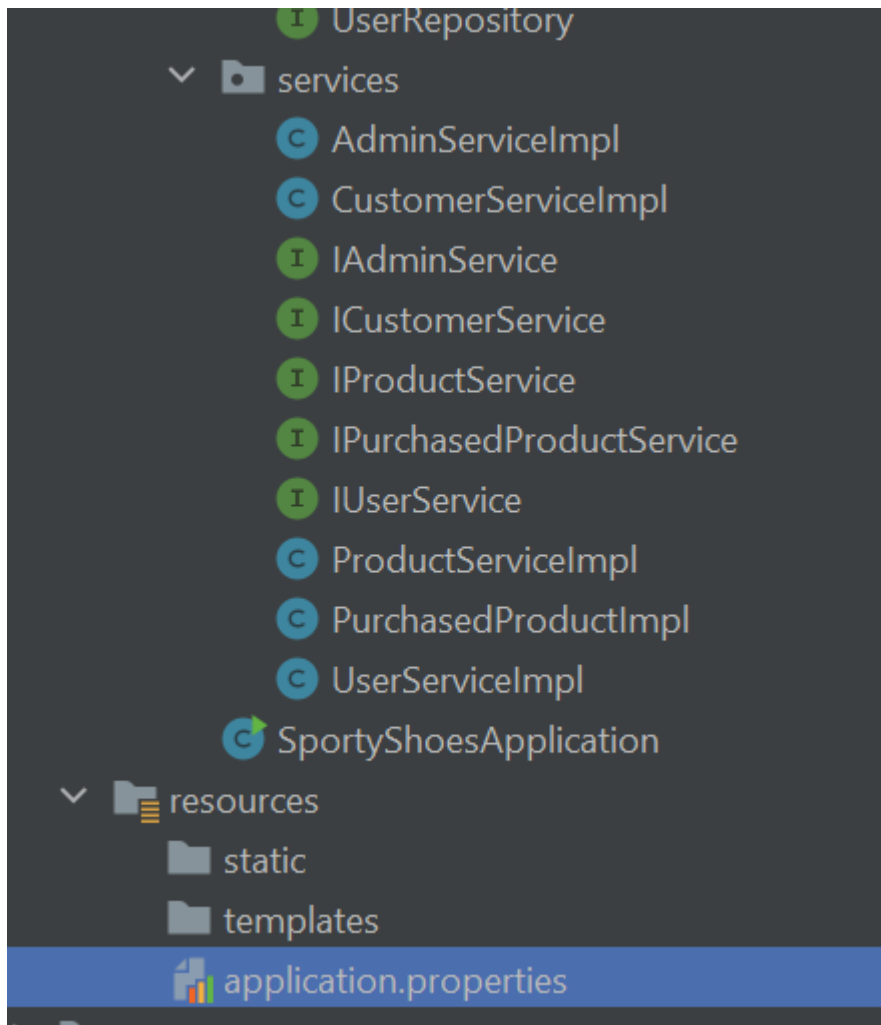
```
mysql>
```


Source Code:

Here is some of the source code.

1> Project Structure





2> Controller

a) Admin Controller

```
AdminController.java
29  @Autowired
30  private IPurchasedProductService prodService;
31
32  @Autowired
33  private IUserService userService;
34
35  @PostMapping("/insertadmin")
36  public Admin insertAdmin(@RequestBody Admin admin) { return adminService.insertAdminInDB(admin); }
37
38  @PostMapping("/updateadminbyid/{adminid}")
39  public void updateAdmin(@PathVariable("adminid") Long adminId, @RequestBody Admin admin) throws ResourceNotFoundException {
40      adminService.updateAdminInDB(admin, adminId);
41  }
42
43  @GetMapping("/getallvalidcustomers")
44  public List<Customer> getAllUsers() { return custService.getAllCustomers(); }
45
46  @GetMapping("/getallcustomerbyname/{custname}")
47  public List<Customer> findByCustomerName(@PathVariable("custname") String customerName){
48      return custService.findByCustomerName(customerName);
49  }
50
51  }
```

b) Customer Controller

```
AdminController.java x CustomerController.java x
35
no usages
36 @GetMapping("/getcustomer/{userid}")
37 public Customer getCustomer(@PathVariable("userid") Long customerId) throws ResourceNotFound {
38     return customerService.getCustomerInDB(customerId);
39 }
40
no usages
41 @DeleteMapping("/deletecustomer/{userid}")
42 public void deleteCustomer(@PathVariable("userid") Long customerId) {
43     customerService.deleteCustomerInDB(customerId);
44 }
45
no usages
46 @PutMapping("/updatecustomerbyid/{userid}")
47 public void updateCustomer(@PathVariable("userid") Long userId, @RequestBody Customer user) throws ResourceNotFound {
48     customerService.updateCustomerInDB(user, userId);
49 }
50
no usages
51 @PutMapping("/addcustomerproducts/{custid}/{prodid}")
52 public void updateProductInCustomer(@PathVariable("custid") Long userId, @PathVariable("prodid") Long prodid, @RequestBody Product product) {
53     customerService.insertProductInExistingCustomerInDB(user, userId, prodid, product);
54 }
```

c) Product Controller

```
AdminController.java x CustomerController.java x ProductController.java x
19 @RestController
20 @RequestMapping("/product")
21 public class ProductController {
22
23     5 usages
24     @Autowired
25     private IProductService productService;
26
27     no usages
28     @PostMapping("/insertproduct")
29     public Product insertProduct(@RequestBody Product newProduct) {
30         return productService.insertProductInDB(newProduct);
31     }
32
33     no usages
34     @GetMapping("/getallproducts")
35     public List<Product> getAllProducts() { return productService.getAllProducts(); }
36
37     no usages
38     @GetMapping("/getproduct/{productid}")
39     public Product getProduct(@PathVariable("productid") Long productId) throws ResourceNotFound {
40         return productService.getProductInDB(productId);
41     }
42 }
```

3> Resource Not Found Exception

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help demo - ResourceNotFound.java
demo src main java com demo exceptions ResourceNotFound
Project
  ProductController
  PurchasedProductController
  UserController
  exceptions
    ResourceNotFound
  models
    Admin
    Customer
    Product
    PurchasedProduct
    User
  repository
    AdminRepository
    CustomerRepository
    ProductRepository
    PurchasedProductRepository
3
package com.demo.exceptions;
4
5 public class ResourceNotFound extends Exception {
6
7     public ResourceNotFound(String exMsg) { super(exMsg); }
8 }
9
Maven
Notifications
```

4> Models

a) Admin

```
AdminController.java x CustomerController.java x ProductController.java x ResourceNotFound.java x Admin.java x
11 @Table(name = "admin")
12 public class Admin {
13
14     2 usages
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name = "admin_id")
18     private Long adminId;
19
20     3 usages
21     @Column(name = "admin_name")
22     private String name;
23
24     3 usages
25     @Column(name = "admin_email")
26     private String email;
27
28     3 usages
29     @Column(name = "admin_password")
30     private String pwd;
31
32     3 usages
33     @Column(name = "admin_contact")
34     private String contact;
35
36     no usages
37     public Admin() {
38
39     }
40 }
```

b) Customer

```
AdminController.java x CustomerController.java x ProductController.java x ResourceNotFound.java x Admin.java x Customer.java x
25 private Float totalPrice;
26
27     2 usages
28     @OneToMany(cascade = CascadeType.ALL)
29     @JoinColumn(name = "customer_id")
30     private Set<PurchasedProduct> list = new HashSet<>();
31
32     public Customer() { totalPrice = (float) 0; }
33
34     public Customer(String customerName, String email, Long contact, String date, Float totalPrice) {
35         this.customerName = customerName;
36         this.email = email;
37         this.contact = contact;
38         this.date = date;
39         this.totalPrice = totalPrice;
40     }
41
42     no usages
43     public Long getCustomerId() { return customerId; }
44
45     no usages
46     public void setCustomerId(Long customerId) { this.customerId = customerId; }
47
48     1 usage
49     public String getCustomerName() { return customerName; }
50
51 }
```

c) Product

```
AdminController.java x CustomerController.java x ProductController.java x ResourceNotFound.java x Admin.java x Customer.java x Product.java x
62
63     3 usages
64     public int getSize() { return size; }
65
66     no usages
67     public void setSize(int size) { this.size = size; }
68
69     4 usages
70     public String getOrigin() { return origin; }
71
72     1 usage
73     public void setOrigin(String origin) { this.origin = origin; }
74
75     7 usages
76     public Float getPrice() { return price; }
77
78     2 usages
79     public void setPrice(Float price) { this.price = price; }
80
81     4 usages
82     public int getTag() {
83         return tag;
84     }
85
86     1 usage
87     public void setTag(int tag) { this.tag = tag; }
88
89 }
```

d) PurchasedProduct

```
11  @Table(name = "purchased_product")
12  public class PurchasedProduct {
13
14      no usages
15      public int getSize() { return size; }
16
17      3 usages
18      public void setSize(int size) { this.size = size; }
19
20
21      1 usage
22      public String getOrigin() { return origin; }
23
24      4 usages
25      public void setOrigin(String origin) { this.origin = origin; }
26
27
28      3 usages
29      public Float getPrice() { return price; }
30
31      5 usages
32      public void setPrice(Float price) { this.price = price; }
33
34
35      1 usage
36      public int getTag() {
37          return tag;
38      }
39
40      4 usages
41      public void setTag(int tag) { this.tag = tag; }
42  }
```

e) User

```
52  public void setUserEmail(String userEmail) {
53      this.userEmail = userEmail;
54  }
55
56      1 usage
57      public String getUserPwd() {
58          return userPwd;
59      }
60
61      1 usage
62      public void setUserPwd(String userPwd) {
63          this.userPwd = userPwd;
64      }
65
66      2 usages
67      public Long getUserContact() {
68          return userContact;
69      }
70
71      1 usage
72      public void setUserContact(Long userContact) {
73          this.userContact = userContact;
74      }
75  }
```

5> Repository

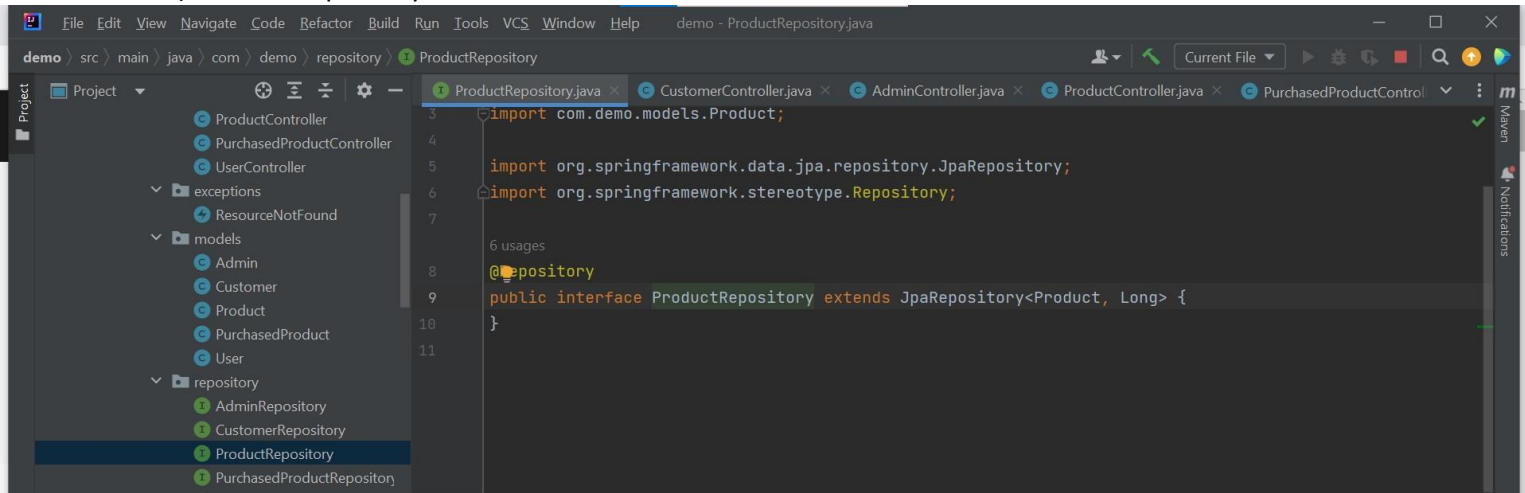
a) Admin Repository

```
demo - AdminRepository.java
demo > src > main > java > com > demo > repository > AdminRepository

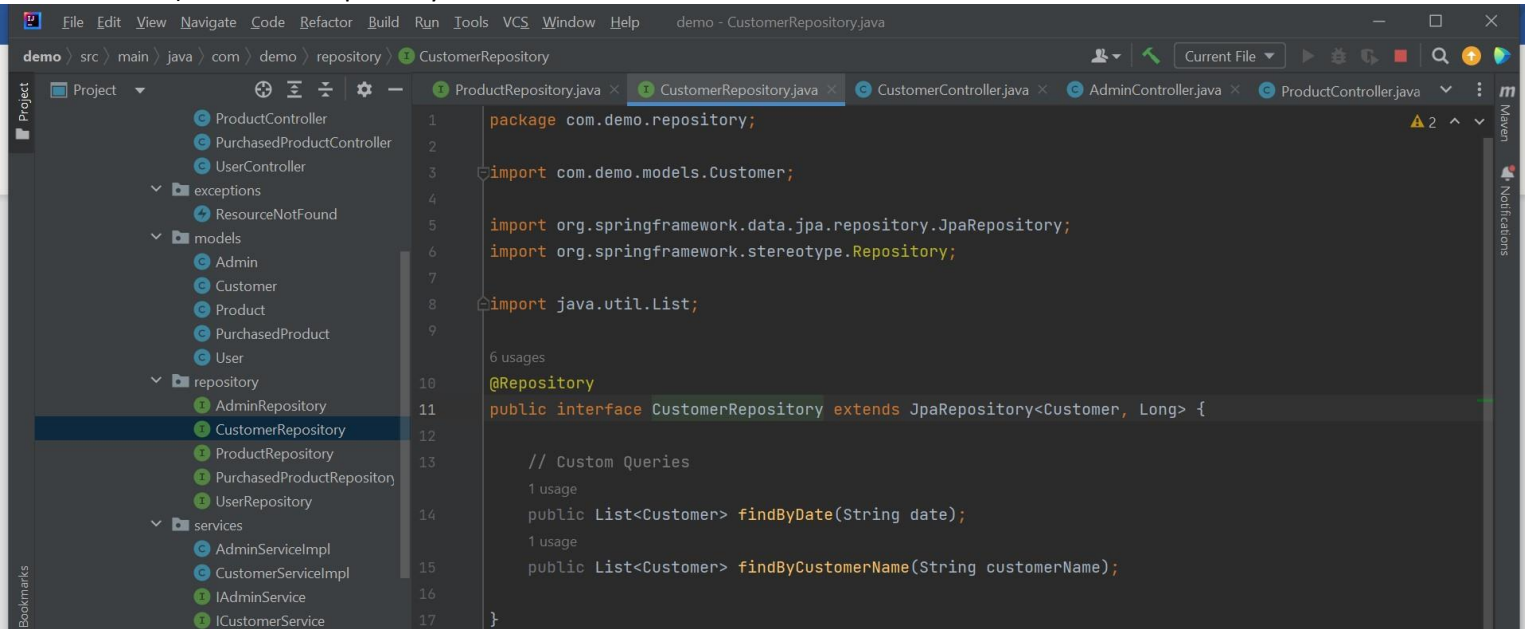
Project
├── Project
│   ├── ProductController
│   ├── PurchasedProductController
│   ├── UserController
│   ├── exceptions
│   │   ├── ResourceNotFound
│   ├── models
│   │   ├── Admin
│   │   ├── Customer
│   │   ├── Product
│   │   ├── PurchasedProduct
│   │   ├── User
│   └── repository
│       ├── AdminRepository
│       ├── CustomerRepository
│       ├── ProductRepository
│       └── PurchasedProductRepository
└── Maven

1  package com.demo.repository;
2
3  import com.demo.models.Admin;
4
5  import org.springframework.data.jpa.repository.JpaRepository;
6  import org.springframework.stereotype.Repository;
7
8      2 usages
9  @Repository
10 public interface AdminRepository extends JpaRepository<Admin, Long> {
11 }
```


b) Product Repository



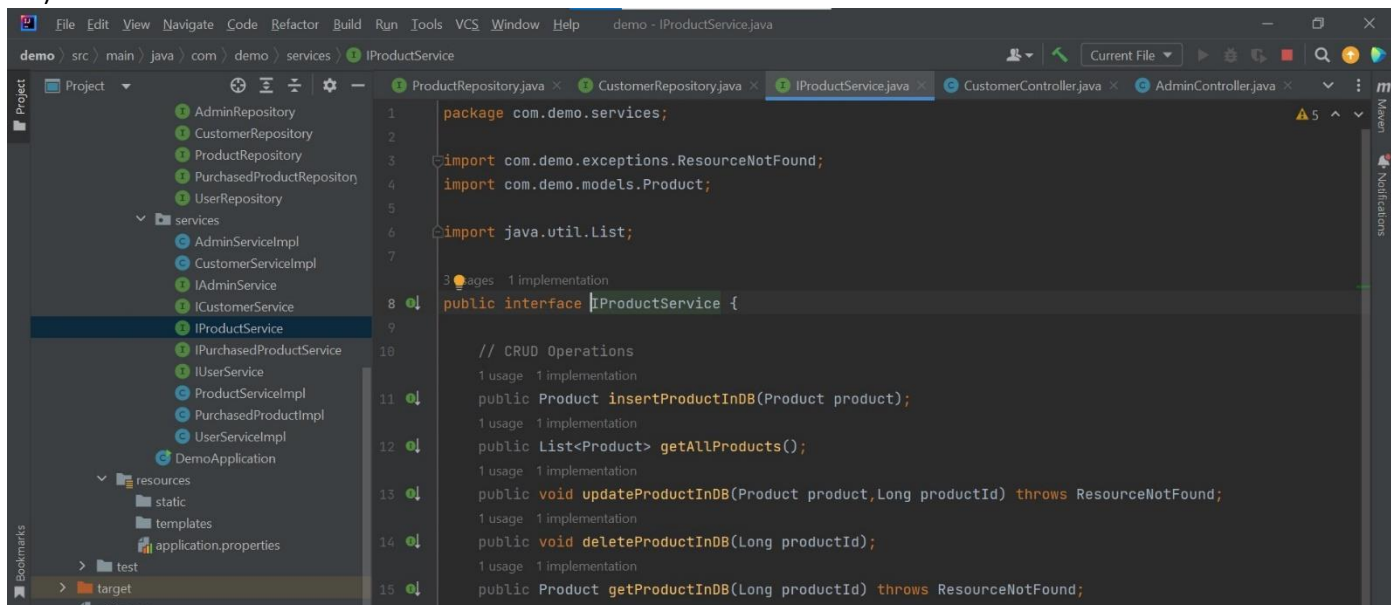
c) Customer Repository



6> Services

=> Interfaces

a) IProduct Interface



=> Classes

a) ProductServiceImpl

```
ProductRepository.java x ProductRepository.java x CustomerRepository.java x IAdminService.java x IProductService.java x ProductServiceImpl.java x
16 private ProductRepository productRepo;
17
18 1 usage
19 @Override
20 public Product insertProductInDB(Product product) { return productRepo.save(product); }
21
22 1 usage
23 @Override
24 public List<Product> getAllProducts() { return productRepo.findAll(); }
25
26 1 usage
27 @Override
28 public void updateProductInDB(Product product, Long productId) throws ResourceNotFound {
29     Product existingProduct = productRepo.findById(productId).get();
30     if(product != null) {
31         // Update existing Product Details with new Details.
32         existingProduct.setCategory(product.getCategory());
33         existingProduct.setCompany(product.getCompany());
34         existingProduct.setOrigin(product.getOrigin());
35         existingProduct.setPrice(product.getPrice());
36         existingProduct.setTag(product.getTag());
37         productRepo.save(existingProduct);
38     } else {
39         throw new ResourceNotFound("Product not found");
40     }
41 }
```

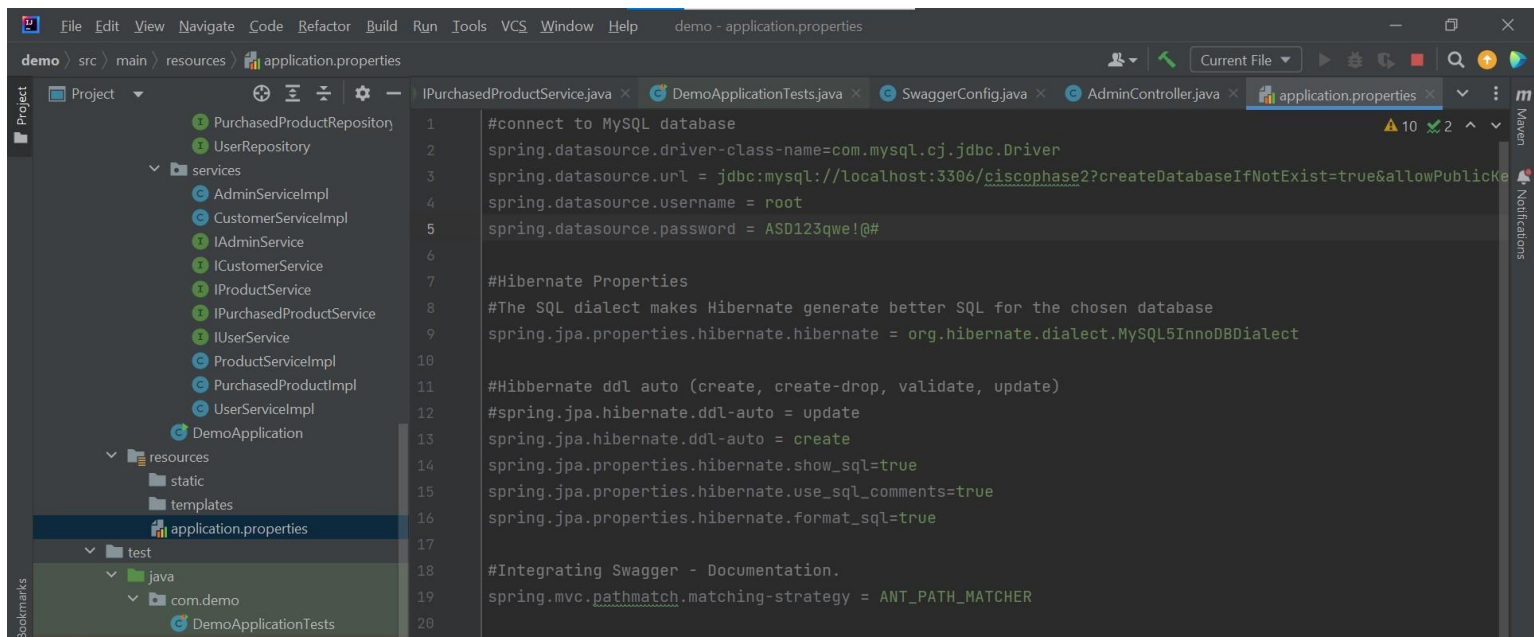
b) AdminServiceImpl

```
ProductRepository.java x CustomerRepository.java x IAdminService.java x IProductService.java x ProductServiceImpl.java x AdminServiceImpl.java x
12 public class AdminServiceImpl implements IAdminService {
13
14     3 usages
15     @Autowired
16     private AdminRepository adminRepo;
17
18     no usages
19     @Autowired
20     private CustomerRepository custRepo;
21
22     1 usage
23     @Override
24     public Admin insertAdminInDB(Admin admin) { return adminRepo.save(admin); }
25
26     1 usage
27     @Override
28     public void updateAdminInDB(Admin admin, Long adminId) throws ResourceNotFound {
29         Admin existingAdmin = adminRepo.findById(adminId).get();
30         if(admin != null) {
31             // Update existing Admin with Password.
32             existingAdmin.setPwd(admin.getPwd());
33             adminRepo.save(existingAdmin);
34         } else {
35             throw new ResourceNotFound("Customer not found");
36         }
37     }
38 }
```

c) UserServiceImpl

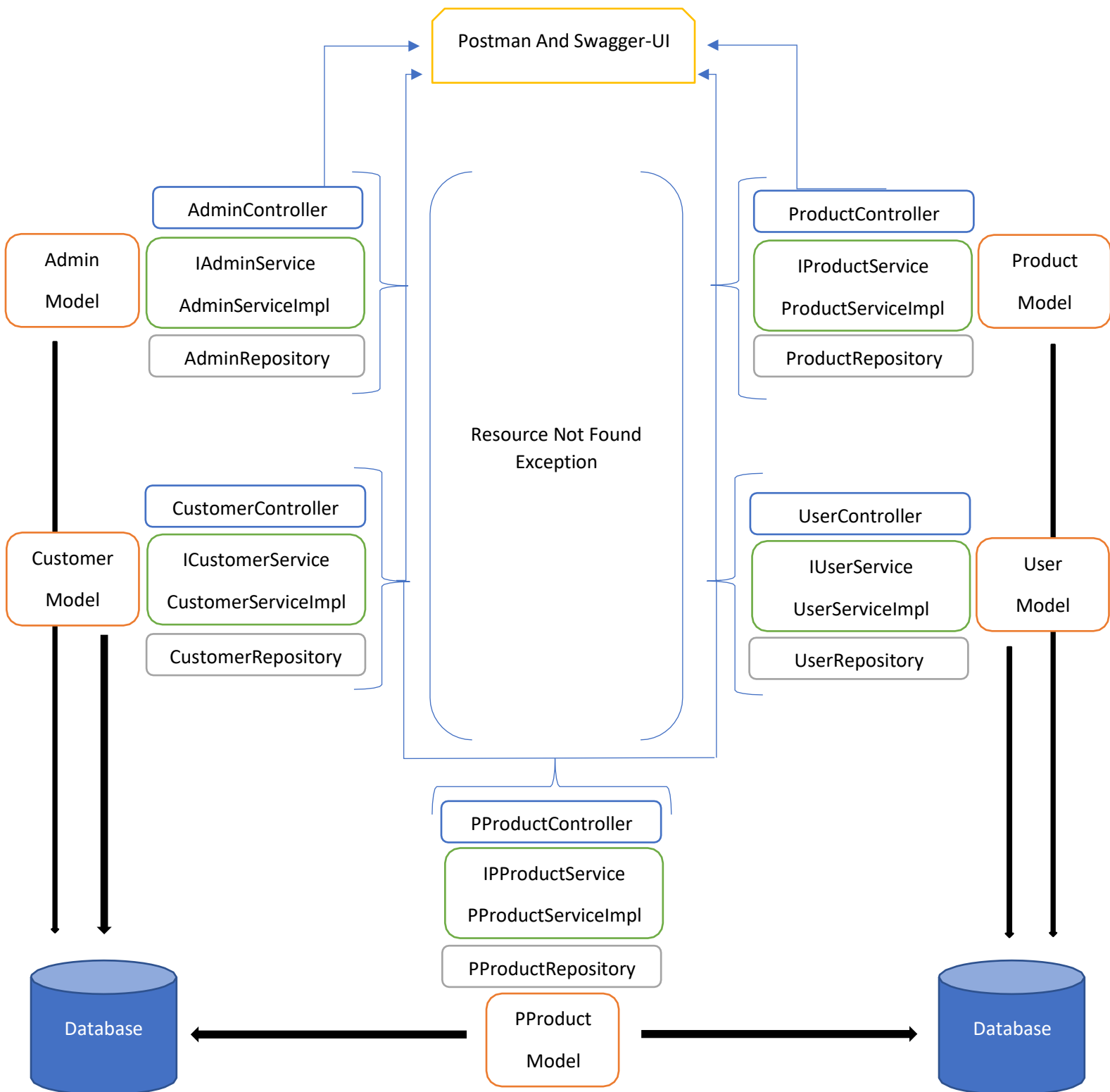
```
CustomerRepository.java x IAdminService.java x IProductService.java x ProductServiceImpl.java x AdminServiceImpl.java x UserServiceImpl.java x
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.stereotype.Service;
14
15 import java.util.List;
16 import java.util.Set;
17
18 no usages
19 @Service
20 public class UserServiceImpl implements IUserService {
21
22     5 usages
23     @Autowired
24     UserRepository userRepo;
25
26     2 usages
27     @Autowired
28     ProductRepository productRepo;
29
30     1 usage
31     @Autowired
32     CustomerRepository customerRepo;
33
34     3 usages
35     Float total = (float) 0;
36
37     1 usage
38     @Override
39 }
```


7> application.properties



```
1 #connect to MySQL database
2 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
3 spring.datasource.url = jdbc:mysql://localhost:3306/ciscophase2?createDatabaseIfNotExist=true&allowPublicKeyRetrieval=true&useSSL=false
4 spring.datasource.username = root
5 spring.datasource.password = ASD123qwe!@#
6
7 #Hibernate Properties
8 #The SQL dialect makes Hibernate generate better SQL for the chosen database
9 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
10
11 #Hibernate ddl auto (create, create-drop, validate, update)
12 spring.jpa.hibernate.ddl-auto = update
13 spring.jpa.hibernate.ddl-auto = create
14 spring.jpa.properties.hibernate.show_sql=true
15 spring.jpa.properties.hibernate.use_sql_comments=true
16 spring.jpa.properties.hibernate.format_sql=true
17
18 #Integrating Swagger - Documentation.
19 spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER
20
```

Flow Diagram



❖ Core Concepts used in this project are mostly maven, hibernate, MySQL, spring boot, associations, java, CRUD operations in a database, Postman, and Swagger-UI.

Algorithm

Step 1> Start.

Step 2> The user has the options to go through three sections.

Case 1: If user select “admin” section then go to step 3 & 4.

Case 2: If user select “user” section then go to step 5 & 6.

Case 3: If user select “customer” section, then go to step 7.

Step 3> Once a user select admin, it will show all the admin details along with showing users details that are signed up in this portal.

Step 4> An admin main page will also display the list of customers along with their purchased products. Along with this the admin can also change his credentials.

Step 5> In the “user” section, the user can register and login with their details such as email and password. And the user can purchase the product without the need of entering user details.

Step 6> And user can also check his purchased product history along with its details such as price, purchase date. And user can also change their details along with email and password.

Step 7> If user is not “admin” or “user”, they can still purchase the product with the need of fulfilling their details along with getting their receipt, but they cannot access any database of their past product history.

Step 8> Stop

Conclusion

1: The prototype is robust and platform independent.

2: User can easily use the prototype and safely exit out of it.

3: As a developer, we can enhance it by introducing several new features such as dynamic web pages, guards and return once admin has been logout, routing, custom validators and can have more user-friendly by adding styling (CSS, Bootstrap), custom loaders.

4: Though this prototype is tightly connected, the data will only persist in database until server is running and gets reset with “CREATE” option of hibernate in “application.properties”.

5: This prototype can also be implemented with multithreading to enable better performance.

6: And lastly, this prototype can be upgraded by making every service as standalone and will be created whenever it is invoked.