Extra Features :

Features

1. Admin Deactivate User
2. Admin Can Get Users by Status (Active / Not Active)
3. Get Product Sales
4. Get Product Income
5. Merchants Selling a Product
6. Filter Products (Advanced Search)

Admin Deactivate User

🔹 Description:

•Allows an Admin to deactivate a user account.

•User can’t Buy any more

🔹 Logic Steps:

•Verify admin role

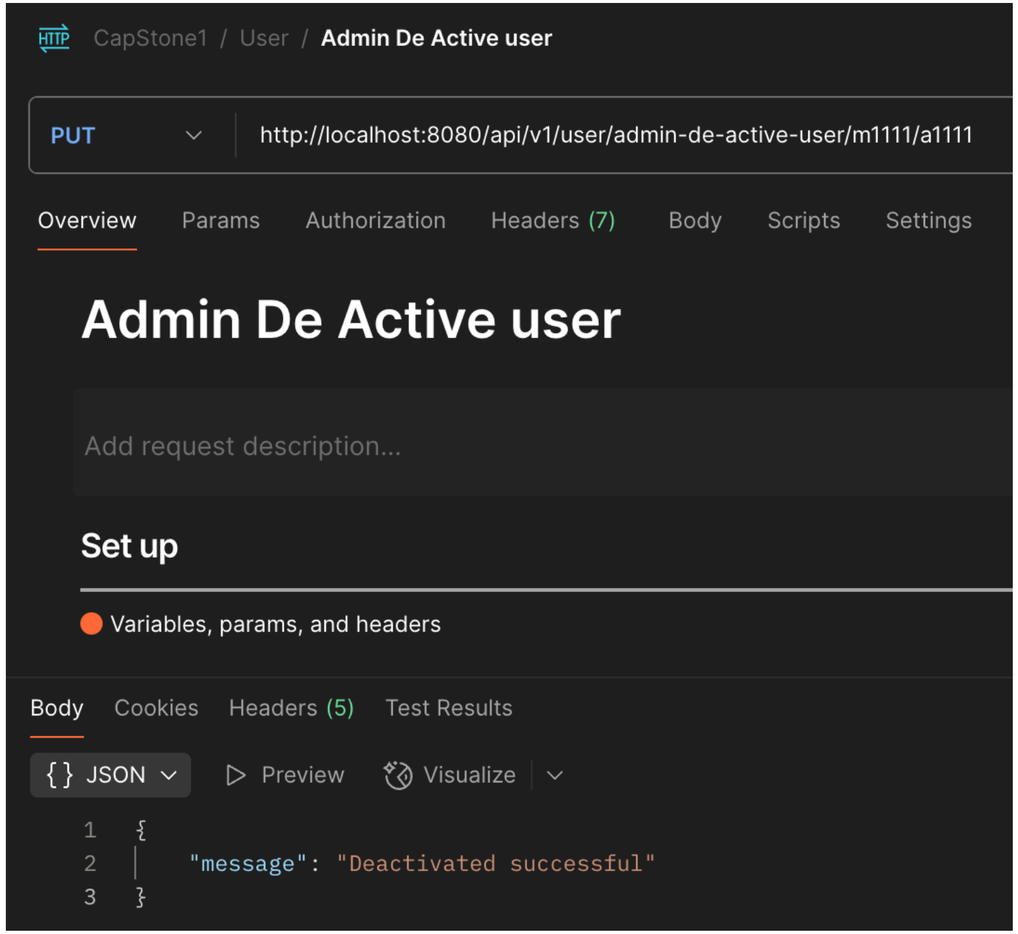
• Check if the user exists

•Set user isActive to false

🔹 Output :

•User successfully deactivated.

```java
public String adminDeActiveUser(String adminId , String userId) {  1 usage  new *
        boolean isAdmin = false;
        boolean isUserExist = false;
        UserModel user = null;
        for (int i = 0; i < users.size(); i++) {
            if (users.get(i).getId().equals(adminId) && users.get(i).getRole().equals("Admin")) {
                isAdmin = true;
            }
        }
        if (!isAdmin) {
            return "No admin found for id: " + adminId;
        }
        for (int i = 0; i < users.size(); i++) {
            if (users.get(i).getId().equals(userId)) {
                user = users.get(i);
                isUserExist = true;
            }
        }
        if (!isUserExist) {
            return "No User found for id: " + userId;
        }
        user.setActive(false);
        return "Deactivated successful";
```



Get Users by Status (Active / Not Active)

Description:

•Allows Admin to view users based on activation status.

Logic Steps:

•Verify admin role

Validate the search term (active / notActive)

•Set user isActive to false

Output :

•List of users filtered by active/notActive status.

```java
//Admin can get Active users and not Active users
public ArrayList<UserModel> getUsersByStatus(String adminId , String search) {  3 usages  new *
    ArrayList<UserModel> usersFound = new ArrayList<>();

    boolean isAdmin = false;
    for (UserModel user : users) {
        if (user.getId().equals(adminId) && user.getRole().equals("Admin")) {
            isAdmin = true;
            break;
        }
    }

    if (!isAdmin) {
        return null;
    }

    if (!search.equalsIgnoreCase( anotherString: "active") && !search.equalsIgnoreCase( anotherString: "notActive")) {
        return null;
    }


    for (UserModel user : users) {
        if (search.equalsIgnoreCase( anotherString: "active") && user.isActive()) {
            usersFound.add(user);
        } else if (search.equalsIgnoreCase( anotherString: "notActive") && !user.isActive()) {
            usersFound.add(user);
        }
    }

    return usersFound;
}
```

CapStone1 / User / **Admin Search by status users**

GET http://localhost:8080/api/v1/user/admin/get-users-by-status/m1111/notActive

Overview Params Authorization Headers (6) Body Scripts Settings

Admin Search by status users

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize


```json
[
    {
        "id": "a1111",
        "username": "Abdullah",
        "password": "pass123",
        "email": "Abdullah@example.com",
        "role": "Customer",
        "balance": 500.75,
        "active": false
    },
    {
        "id": "me1111",
        "username": "Meshary",
        "password": "pass123",
        "email": "Meshary@example.com",
        "role": "Customer",
        "balance": 1599.75,
        "active": false
    },
    {
        "id": "l1111",
        "username": "Lama123",
        "password": "pass123",
        "email": "lama@example.com",
        "role": "Customer",
        "balance": 1599.75,
        "active": false
    }
```

Product Sales & Income

- 🔹 **Description:**

•Displays how many times a product was sold and the total income from it.

🔹 **Logic Steps:**

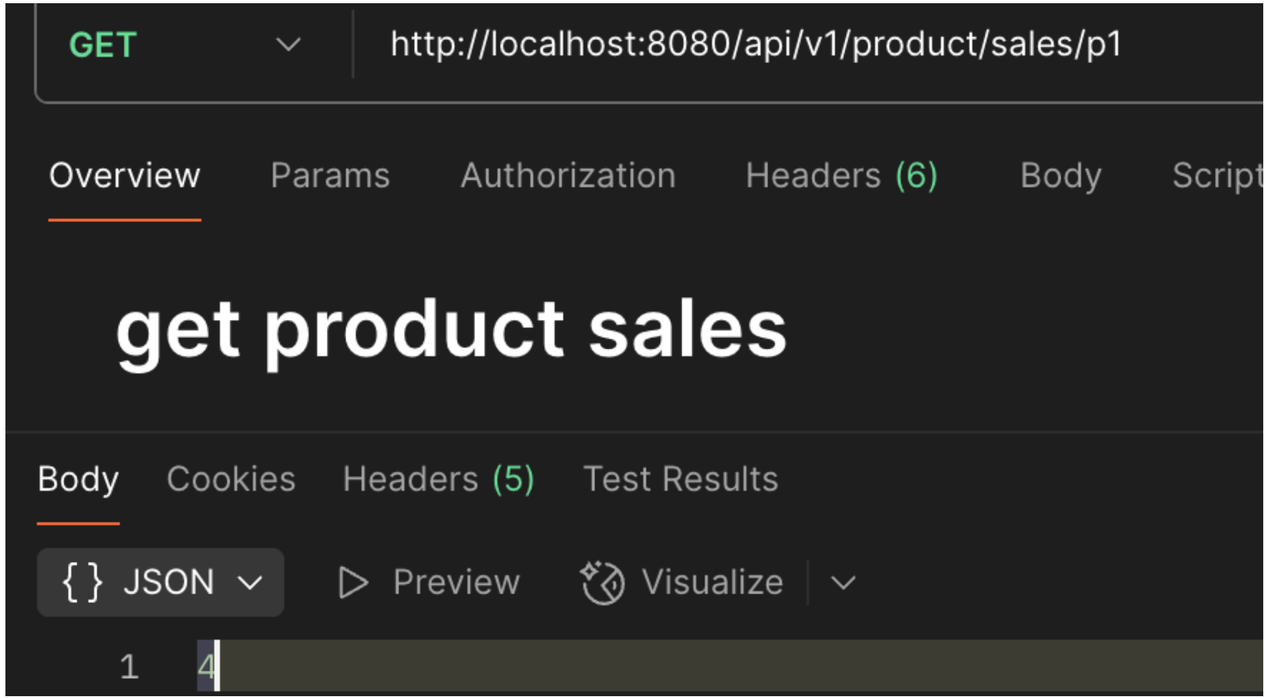
•Iterate through products

•Use soldCount and price to calculate income

🔹 **Output :**

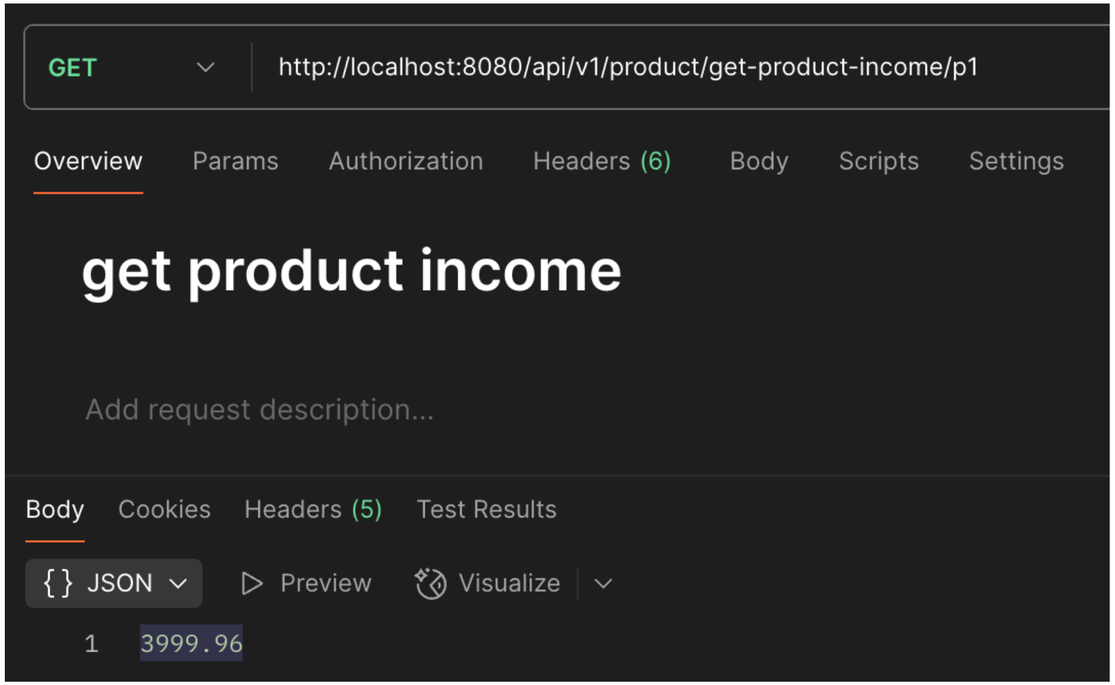
•Returns number of sales and total income.

```java
public int getProductCount(String id) {  1 usage  new *
    for (int i = 0; i < products.size(); i++) {
        if (products.get(i).getId().equals(id)) {
            return products.get(i).getSoldCount();
        }
    }
    return 0;
```



```java
    public double getProductIncome(String id) {  1 usage  new *
        for (int i = 0; i < products.size(); i++) {
            if (products.get(i).getId().equals(id)) {
                return products.get(i).getPrice() * products.get(i).getSoldCount();
            }
        }
        return 0;
    }


}
```



Merchants Selling a Product

- 🔹 **Description:**

•Shows merchants that sell a specific product and their available stock.

🔹 **Logic Steps:**

•Iterate through merchant stock

•Match product ID

•Get merchant name and stock count

🔹 **Output :**

•List of merchants and stock per merchant.

```java
public ArrayList<String> getMerchantsByProduct(String productId) {  2 usages  new *
    ArrayList<String> result = new ArrayList<>();

    for (MerchantStockModel stock : merchantStockModels) {
        if (stock.getProductId().equals(productId)) {
            String merchantName = " ";

            for (MerchantModel merchant : merchantService.merchants) {
                if (merchant.getId().equals(stock.getMerchantId())) {
                    merchantName = merchant.getName();
                    break;
                }
            }

            String found = "Merchant: " + merchantName + ", Stock: " + stock.getStock();
            result.add(found);
        }
    }

    return result;
}
```

GET http://localhost:8080/api/v1/product/merchants-stock/p1

Overview Params Authorization Headers (6) Body Scripts

Search for product

Body Cookies Headers (5) Test Results

{ } JSON Preview Visualize

```
1  [
2      "Merchant: Jarir, Stock: 13",
3      "Merchant: Extra, Stock: 8"
4  ]
```

Filter Products (Advanced Search)

- 🔹 **Description:**

•Filter available products based on keyword, price range, category, and merchant.

🔹 **Logic Steps:**

•Check stock availability

•Match name, price, category, merchent

•Add product if not already added

🔹 **Output :**

•List of filtered products based on search criteria.

```java
public ArrayList<ProductModel> filterAvailableProducts(String keyword, Double minPrice, Double maxPrice, String categoryId,
    ArrayList<ProductModel> result = new ArrayList<>();

    for (MerchantStockModel stock : merchantStockModels) {
        if (stock.getStock() > 0) {
            if (stock.getMerchantId().equals(merchantId)) {

                ProductModel product = null;
                for (ProductModel p : productService.products) {
                    if (p.getId().equals(stock.getProductId())) {
                        product = p;
                        break;
                    }
                }

                if (product != null) {
                    if (product.getName().toLowerCase().contains(keyword.toLowerCase())) {
                        if (product.getPrice() >= minPrice && product.getPrice() <= maxPrice) {
                            if (product.getCategoryID().equals(categoryId)) {
                                if (!result.contains(product)) {
                                    result.add(product);
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    return result;
```

GET http://localhost:8080/api/v1/product/filter-products/bo/0/5000/c2/m

Overview Params Authorization Headers (6) Body Scripts Settings

filter products

ody Cookies Headers (5) Test Results

{ } JSON Preview Visualize


```
[
    {
        "id": "p1",
        "name": "Java Programming Book",
        "price": 999.99,
        "categoryID": "c2",
        "soldCount": 4,
        "totalIncome": 0.0
    }
]
```