Endpoints Usage Scenario

We Know Checking All Endpoints for All These Projects Is BORING.

That's why we wrote this scenario.

This scenario to help you go on and test ALL developed Endpoints for our project.

Hope It Helps, Happy Marking!

Assuming you ran the App on your **localhost** on port **8080** ⇒ http://localhost:8080 will be the starting link path for all upcoming links.

- First let's **register** an account for you in our system, for future use: **[Register]**

  Note: from now on, your name is Ahmed 😊.

  1. **Method type:** POST
  2. **URL:** http://localhost:8080/**accounts/register**
  3. **Request Body:** Enter info of new account like this format:
     a. JSON Representation of an object of type **Account** is sent.
     b. "Language" property represents system language of user:
        - If you choose **"ENGLISH"**, **"GERMAN"** you'll see notifications with this language.
        - If you choose any other available language, you'll see notifications in English.

```
{
    "balance": 5000,
    "customerInfo": {
        "name": "Ahmed",
        "email": "Ahmed204@gmail.com",
        "phone": "01122957010",
        "language": "ENGLISH"
    },
    "accountCredentials": {
        "username": "ahmed204", // unique
        "password": "ar1789"
    }
}
```

  - **Response:** [hopefully you receive this response 😊]

```
{
    "accountCredentials": {
        "username": "ahmed204",
        "password": "ar1789"
    },
    "customerInfo": {
        "name": "Ahmed",
```

```
        "email": "Ahmed204@gmail.com",
        "phone": "01122957010",
        "language": "ENGLISH"
    },
    "balance": 5000.0
}
```

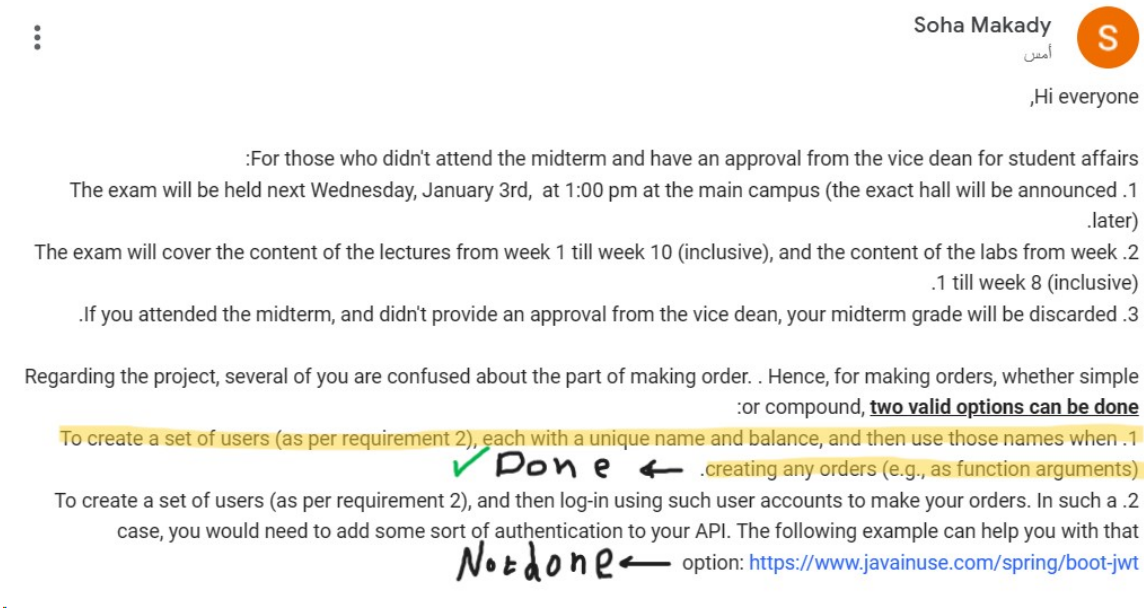

- To make sure your account is registered, you made a **Login** with your account: **[Login]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/**accounts/login**
  3. **Request Body:** Enter info for account like this format:


```
{
    "username": "ahmed204",
    "password": "ar1789"
}
```


  - **Response:** [hopefully you receive this response 😊]


```
{
    "accountCredentials": {
        "username": "ahmed204",
        "password": "ar1789"
    },
    "customerInfo": {
        "name": "Ahmed",
        "email": "Ahmed204@gmail.com",
        "phone": "01122957010",
        "language": "ENGLISH"
    },
    "balance": 5000.0
}
```


  - Note: This method returns the **Account object** that matched your input credentials. But you can still use the system functionalities without logging in, since **Dr. Soha** mentioned that the token-authentication part is optional (you're allowed to create orders by only giving usernames of the order's user, no need to login/authenticate).



- Now come have a look on **all our available products**. **[Get all products]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/products/all
     - **Response:** [hopefully you receive this response 😊]

```
[
    {
        "id": "1",
        "name": "Clean Code Book",
        "price": 100,
        "category": "BOOKS",
        "vendor": "AMAZON",
        "remainingItemsCount": 4
    },
    {
        "id": "2",
        "name": "Sun Block cream",
        "price": 200,
        "category": "BEAUTY",
        "vendor": "ALIBABA",
        "remainingItemsCount": 3
    },
    {
        "id": "3",
        "name": "Clean Architecture Book",
        "price": 300,
        "category": "BOOKS",
        "vendor": "AMAZON",
        "remainingItemsCount": 3
```

```
    }
]
```

- As a hard-working **TA**, you want to check the requirement asking for **how many products remaining per category**.
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/products/categories/get_products_count/BOOKS
  - **Response:** [hopefully you receive this response 😊]

```
2
```

  You received `2` since we only have 2 products in the `BOOKS` category.

- You really love clean code book, so let's **place a new order** of it. **[Place Simple order]**
  1. **Method type**: POST
  2. **URL**: http://localhost:8080/orders/place
  3. **Request Body:** Enter your order input like this format:
     a. JSON Representation of an object of type **OrderInput** is sent.
     b. **"type"** property represents order type:
        - If you choose "simple" your order will be simple (one order).
        - If you choose "compound" your order will be compound order (more than one order).

```
{
    "type": "simple",
    "userName": "ahmed204",
    "location": "dokki",
    "productsIDs": [
        "1"
    ]
}
```

  - **Response:** [hopefully you receive this response 😊]

```
{
    "orderID": "1",
    "shippingFee": 10.0,
    "setTime": "2023-12-31T17:40:09.264+00:00",
    "itemsTotalPrice": 100.0,
    "location": "dokki",
    "products": [
        {
            "product": {
                "id": "1",
                "name": "Clean Code Book",
                "price": 100.0,
```

```
                "category": "BOOKS",
                "vendor": "AMAZON",
                "remainingItemsCount": 3
            },
            "serialNumber": "1"
        }
    ],
    "username": "ahmed204",
    "shipping": false
}
```


- Now use the **[Login]** method to check your new balance after placing the order, it became *4890.0 pounds (it decreased by 110 pounds)*, right?

- You want a prove that the order is placed, so you get your orders details **[Get Order]**

  1. **Method type:** GET
  2. **URL:** http://localhost:8080/orders/get/1 *[1 here represents the order id]*
  - **Response:** [hopefully you receive this response 😊]


```
{
    "orderID": "1",
    "shippingFee": 10.0,
    "setTime": "2023-12-30T13:36:54.314+00:00",
    "location": "dokki",
    "totalPrice": 100.0,
    "products": [
        {
            "product": {
                "id": "1",
                "name": "Clean Code Book",
                "price": 100.0,
                "category": "BOOKS",
                "vendor": "AMAZON",
                "remainingItemsCount": 3
            },
            "serialNumber": "1"
        }
    ],
    "username": "ahmed204",
    "shipping": false
}
```


- You got excited after seeing the order details, you want to **SHIP it** RIGHT NOW! **[Ship Order]**

  1. **Method type:** PUT
  2. **URL: http://localhost:8080/orders/ship/1** *[1 here represents the order id]*
  - **Response:** [hopefully you receive this response 😊]

```
{
    "orderID": "1",
    "shippingFee": 10.0,
    "setTime": "2023-12-31T17:40:09.264+00:00",
    "itemsTotalPrice": 100.0,
    "location": "dokki",
    "products": [
        {
            "product": {
                "id": "1",
                "name": "Clean Code Book",
                "price": 100.0,
                "category": "BOOKS",
                "vendor": "AMAZON",
                "remainingItemsCount": 3
            },
            "serialNumber": "1"
        }
    ],
    "username": "ahmed204",
    "shipping": true
}
```


- Still want a prove of shipping. You know your way go use **[Get Order],** and check the **“shipping”** property ⇒ It’s **true** now, right?

- You suddenly remembered that you need to check on the **Notifications Queue Requirement**! don’t worry I got your back! **[Get Notifications]**

  1. **Method type:** GET
  2. **URL:** http://localhost:8080/notifications/get_in_queue
  3. Each 1 minute a notification is popped from the queue, so if you found the queue empty or not having all required notifications, that means some of them were popped.

     You can create another order and quickly check the queue 😁.
  4. Response: [If you requested this method the most early.]

     some of them might be missing depending on when you request the method.


```
[
    {
        "subject": "Order #1 has been set",
        "message": "Dear Ahmed, your order #1 has been set.",
        "channel": "SMS",
        "language": "ENGLISH",
        "userName": "ahmed204"
    },
    {
```

```
        "subject": "Order #1 has been set",
        "message": "Dear Ahmed, your order #1 has been set.",
        "channel": "EMAIL",
        "language": "ENGLISH",
        "userName": "ahmed204"
    },
    {
        "subject": "Order #1 started shipping",
        "message": "Dear Ahmed, your order #1 is on the way. The destination is
        "channel": "EMAIL",
        "language": "ENGLISH",
        "userName": "ahmed204"
    }
]
```


- Surprisingly, you decided you don't need clean code book, because your code is actually clean! You want to **cancel the shipping** 😔 **[Cancel Shipping]**
    1. **Method type:** PUT
    2. **URL:** http://localhost:8080/orders/cancel_shipping/1 *[1 here represents the order id]*
    - **Response:** [hopefully you receive this response 😊]


```
{
    "orderID": "1",
    "shippingFee": 10.0,
    "setTime": "2023-12-31T17:44:09.453+00:00",
    "itemsTotalPrice": 100.0,
    "location": "dokki",
    "products": [
        {
            "product": {
                "id": "1",
                "name": "Clean Code Book",
                "price": 100.0,
                "category": "BOOKS",
                "vendor": "AMAZON",
                "remainingItemsCount": 3
            },
            "serialNumber": "1"
        }
    ],
    "username": "ahmed204",
    "shipping": false
}
```


    - use **[Get Order],** and check the **"shipping"** property ⇒ It's **false** now, right?

- After canceling the shipping, you **cancel the order** itself. **[Cancel Order]**

1. **Method type**: DELETE
2. **URL:** http://localhost:8080/orders/cancel/1

- **Response:** [hopefully you receive this response 😊]

```
Order is canceled successfully (balance is refunded)
```

- Check your refunded balance using **[Login],**
  - It's *$4990* now, product price *($100)* is refunded, but the shipping money isn't refunded, of course!

- A **German** friend of yours named **Youssef** (it's actually me), convinced you to make **compound order**, he wants to buy a "sun block product" and "clean architecture book", and you only want a "sun block product". **[Place Compound Order]**
  - First, you registered a new account for him. **[Register]**
  - Here's the request body you sent to the `POST` `http://localhost:8080` `/accounts/register`.

```
{
    "balance": 6000,
    "customerInfo": {
        "name": "Youssef",
        "email": "Youssef21@gmail.com",
        "phone": "01122958876",
        "language": "ENGLISH"
    },
    "accountCredentials": {
        "username": "youssef1",
        "password": "you123"
    }
}
```

1. **Method type**: POST
2. **URL**: http://localhost:8080/orders/place
3. **Request Body:** Enter your order input like this format:
   a. JSON Representation of an object of type **OrderInput** is sent.
   b. **"type"** property represents order type:
      - If you choose "simple" your order will be simple (one order).
      - If you choose "compound" your order will be compound order (more than one order).

```
{
    "type": "compound",
    "commonLocation": "dokki",
    "orders": [
        {
            "type": "simple",
            "userName": "youssef1",
            "location": "dokki1",
```

```
            "productsIDs": [
                "2",
                "3"
            ]
        },
        {
            "type": "simple",
            "userName": "ahmed204",
            "location": "dokki2",
            "productsIDs": [
                "2"
            ]
        }
    ]
}
```

- **Response:** [hopefully you receive similar response 😊]

```
{
    "orderID": "4",
    "shippingFee": 50.0,
    "setTime": "2023-12-31T15:10:23.141+00:00",
    "commonLocation": "dokki",
    "simpleOrders": [
        {
            "orderID": "2",
            "shippingFee": 50.0,
            "setTime": "2023-12-31T15:10:23.141+00:00",
            "location": "dokki1",
            "products": [
                {
                    "product": {
                        "id": "2",
                        "name": "Sun Block cream",
                        "price": 200.0,
                        "category": "BEAUTY",
                        "vendor": "ALIBABA",
                        "remainingItemsCount": 2
                    },
                    "serialNumber": "1"
                },
                {
                    "product": {
                        "id": "3",
                        "name": "Clean Architecture Book",
                        "price": 300.0,
                        "category": "BOOKS",
```

```
                        "vendor": "AMAZON",
                        "remainingItemsCount": 2
                    },
                    "serialNumber": "1"
                }
            ],
            "username": "youssef1",
            "itemsTotalPrice": 500.0,
            "shipping": false
        },
        {
            "orderID": "3",
            "shippingFee": 20.0,
            "setTime": "2023-12-31T15:10:23.141+00:00",
            "location": "dokki2",
            "products": [
                {
                    "product": {
                        "id": "2",
                        "name": "Sun Block cream",
                        "price": 200.0,
                        "category": "BEAUTY",
                        "vendor": "ALIBABA",
                        "remainingItemsCount": 2
                    },
                    "serialNumber": "1"
                }
            ],
            "username":"ahmed204",
            "itemsTotalPrice": 200.0,
            "shipping": false
        }
    ],
    "shipping": false
}
```


- You can now check the **[Notifications Queue]**, to see notifications in German, sent to your German friend 😮.
- The Order id for this compound order is 4, since we created simple order at the start, and now created a compound order having two orders inside, so the inner orders take ids 2 and 3. Then, the compound order takes the id 4.

You liked our web API and wanted to give us a **full mark** with **full bonus marks**. But you haven't checked the **Notification Statistics** yet, so you did the following:

- You viewed **the most notified mail**: **[Get Most Notified Mail]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/notifications/statistics/get_most_notified_mail
     - **Response Example:** [hopefully you receive similar response 😊]

```
Ahmed204@gmail.com
```

- Since Ahmed placed two orders, the most notified mail is **Ahmed's mail**.

- You viewed **the most notified phone**: **[Get Most Notified Phone]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/notifications/statistics/get_most_notified_phone
  - **Response Example:** [hopefully you receive similar response 😊]

```
01122957010
```

  - Since Ahmed placed two orders, the most notified phone is **Ahmed's phone number**.

- You viewed **the most notified phone/mail**: **[Get Most Notified Phone or Mail]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/notifications/statistics/get_most_notified_mail_or_phone
  - **Response Example:** [hopefully you receive similar response 😊]

```
Ahmed204@gmail.com
```

  - Since The **SetOrderTemplate** allows sending on Mail and SMS, but **ShipOrderTemplate** allows sending on Mail only (Mail is used more than phone), and since Ahmed placed two orders, the most notified mail/phone is **Ahmed's mail**.

- You viewed **the most used notification template: [Get Most Used Template]**
  1. **Method type**: GET
  2. **URL**: http://localhost:8080/notifications/statistics/get_most_notified_template
  - Response: [hopefully you receive this response 😊]

```
SetOrderMessageTemplate
```

  - Since the number of placed orders is greater than or equal to the number shipped orders, the most used template is **SetOrderMessageTemplate**.