Sales and Stock System

This application is designed to serve as a back-end system that provides warehouse operations for sales business. The system is developed with a focus on managing the stock control and sales from data fetched from the external API , and is meant to be interfaced with a front-end designed by a separate UX design company.

At the backbone of the application is the SQL Workbench database which stores and manages all relevant information relating to the inventory and sales operations.

The system implements CRUD (Create, Read, Update, Delete) operations through most of the endpoints for all database tables.

Authentication are implemented to ensure only authorized access to the system and to maintain the integrity of the system.

In terms of stock management, the system uses clear definitions for 'in-stock' and 'out-of-stock' items. 'In-stock' refers to items with a stock quantity greater than zero, indicating their availability for purchase, while 'out-of-stock' refers to items with a stock quantity equal to zero, indicating that they are not available for purchase at the moment.

AUTHORIZATION Bearer Token

Token	<token>

Utilities

Setup and Search endpoints

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

POST Initial Setup

```
http://localhost:3000/setup
```

POST_/_setup

Notes:

This endpoint must be used only once to set up the system. It is not meant for multiple usage.

Description:

This endpoint is used for initial system setup. It completes a series of tasks to prepare the system for operation. These tasks include:

1. Checking if the setup has been previously completed
2. Fetching items from an external API and populating the Items and Categories in the database
3. Creating roles in the database (Admin and User)
4. Creating an Admin user

URL: http://localhost:3000/setup

Method: POST

Authentication required: No

Data Parameters:

This endpoint doesn't require any data parameters in the request body and Headers.

Success Response:

- Code: 201
- Content: { message: "Database populated successfully" }

Error Responses:

- If the setup operation has already been completed
  - Code: 409
  - Content: { message: "Setup operation has already been completed" }
- If there's an error fetching items from the external API
  - Code: 500
  - Content: { message: "Error fetching items from external API: " + error.message }
- If there's an error populating items and categories
  - Code: 500
  - Content: { message: "Error populating items and categories: " + error.message }
- If there's an error creating roles
  - Code: 500
  - Content: { message: "Error creating roles: " + error.message }
- If there's an error creating the Admin user
  - Code: 500
  - Content: { message: "Error creating admin user: " + error.message }
- If the external API is not reachable
  - Code: 500
  - Content: { message: "External API is not reachable" }

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

POST Search

```
http://localhost:3000/search
```

POST /search

Description:

This endpoint is used for searching items in the catalog. It allows to search by item name, SKU (Stock Keeping Unit), and/or category name.

URL: `http://localhost:3000/search`

Method: `POST`

Authentication required: No

Data Parameters:

The request body can contain one or more of the following parameters:

- item_name (string, optional)
- category_name (string, optional)
- SKU (string, optional)

The search process will vary based on which parameters are provided:

- If both item_name and category_name are provided, it will search for items with the given name within the specified category.
- If only item_name is provided, it will search for items with the given name.
- If only category_name is provided, it will return all items within the specified category.
- If SKU is provided, it will search for the item with the given SKU.

Success Response:

- **Code:** 200
- **Content:** An array of items matching the search criteria.

Error Responses:

- If the request body contains invalid data
  - **Code:** 400
  - **Content:** { errors: errors.array() }
- If no items are found matching the search criteria
  - **Code:** 404
  - **Content:** { message: "No items found matching your criteria" }
- If there's a server error while processing the request
  - **Code:** 500
  - **Content:** { message: error.message }

Note:

This endpoint supports partial matching for the item_name and category_name parameters. For instance, searching for item_name "mart" will return items like "Smartwatch ", "Smart Watch", and "Smartphone"

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

Body raw (json)

json

```
{
    "item_name" : "mart"
}
```

Authentication(s)

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

POST Signup new user x1

```
http://localhost:3000/signup
```

POST /signup

Description:

This endpoint is used to register a new user.

URL: `http://localhost:3000/signup`

Method: `POST`

Authentication required: No

Data Parameters:

The request body should contain the following parameters:

- fullname (string, required) - It must be at least 5 characters long.
- username (string, required) - It must be at least 5 characters long.
- password (string, required) - It must be at least 6 characters long.
- email (string, required) - It should be in valid email format.

Success Response:

- **Code:** 201
- **Content:** The created user's data.

Error Responses:

- If the request body contains invalid data
  - **Code:** 400
  - **Content:** { errors: errors.array() }
- If the username is already taken
  - **Code:** 400
  - **Content:** { message: "Username is already taken" }
- If the provided email is already registered with 4 users

- **Code:** 400
- **Content:** { message: "This email is already registered with 4 users" }

- If there's a server error while processing the request
  - **Code:** 500
  - **Content:** { message: "An error occurred during signup" }

Note:

Each user is also assigned a role ("User" in this case). A new shopping cart is created for the user during signup. A discount is calculated and updated for all users registered with the same email address. The maximum allowed number of users with the same email address is 4.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

Body raw (json)

json

```json
{
    "fullname" : "Lilly Doe",
    "username": "Lilly",
    "email": "Lilly@example.com",
    "password": "123456"
}
```

POST Signup user same email

http://localhost:3000/signup

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

Body raw (json)

json

```json
{
    "fullname" : "Kelly Doe",
    "username": "Kelly",
    "email": "Lilly@example.com",
    "password": "147852"
}
```

POST Login

```
http://localhost:3000/login
```

POST /login

Description:

This endpoint is used to log in an existing user.

URL: `http://localhost:3000/login`

Method: `POST`

Authentication required: No

Data Parameters:

The request body should contain the following parameters:

- username (string, required)
- password (string, required)

Success Response:

- **Code:** 200
- **Content:** An object containing the JWT token and the user's data.

Error Responses:

- If the request body does not contain the username or password
  - **Code:** 400
  - **Content:** { errors: errors.array() }
- If the provided username or password is invalid
  - **Code:** 400
  - **Content:** { message: "Invalid username or password" }
- If there's a server error while processing the request
  - **Code:** 500
  - **Content:** { message: "An error occurred during login" }

Note:

The JWT token provided upon successful login has a validity of 2 hours. The user should use this token to authenticate subsequent requests.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

Body raw (json)

json

```
{
    "username": "Admin",


    "password": "P@ssword2023"
}
```

GET user

http://localhost:3000/3

GET /:id

Description:

This endpoint is used to get details of a user by their id.

URL: `http://localhost:3000/:id`

Method: `GET`

Authentication required: Yes

URL Parameters:

- `id` (integer, required)

Success Response:

- **Code:** 200
- **Content:** An object containing the user's data.

Error Responses:

- If the authenticated user is not an admin and is trying to get details of another user
  - **Code:** 403
  - **Content:** { message: "Unauthorized Access" }
- If the user with the provided id does not exist
  - **Code:** 500
  - **Content:** { message: "User not found" }
- If there's a server error while processing the request
  - **Code:** 500
  - **Content:** { message: error.message }

Note:

Only admins or the users themselves can retrieve their own details.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

PUT update user

```
http://localhost:3000/3
```

PUT /:id

Description:

This endpoint is used to update the details of a user.

URL: `http://localhost:3000/:id`

Method: `PUT`

Authentication required: Yes

URL Parameters:

- id (integer, required).

Data Parameters:

- fullname (string, optional) - The new full name of the user.
- username (string, optional) - The new username for the user.
- password (string, optional) - The new password for the user.
- email (string, optional) - The new email for the user.

Success Response:

- **Code:** 200
- **Content:** An object containing the updated user's data.

Error Responses:

- If the authenticated user is not an admin and is trying to update another user's details:
  - **Code:** 403
  - **Content:** { message: "Unauthorized Access" }
- If the user with the provided id does not exist:
  - **Code:** 500
  - **Content:** { message: "User not found" }
- If the username or email already exists:
  - **Code:** 500
  - **Content:** { message: "Username is already taken" } or
  - { message: "This email is already registered with 4 users" }
- If there's a server error while processing the request:
  - **Code:** 500
  - **Content:** { message: error.message }

Note:

Only admins or the users themselves can update their own details.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```
{
  "username": "Kylie"
}
```

DELETE Delete user

http://localhost:3000/7

DELETE /:id

Description:This endpoint is used to delete a user account.

URL: `http://localhost:3000/:id`

Method: `DELETE`

Authentication required: Yes

URL Parameters:

- id (integer, required) - The id of the user to be deleted.

Data Parameters: None

Success Response:

- **Code:** 204 - No Content

Error Responses:

- If the authenticated user is not an admin and is trying to delete another user's account:
  - **Code:** 403
  - **Content:** { message: "Unauthorized Access" }
- If the user with the provided id does not exist:
  - **Code:** 500
  - **Content:** { message: "User not found" }
- If there's a server error while processing the request:
  - **Code:** 500
  - **Content:** { message: error.message }

- **Content:** { message: error.message }

Note:

Only admins or the users themselves can delete their own accounts. When a user account is deleted, the user discount based on the email associated with the account will also be updated.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Item(s)

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

GET items(all users)

```
http://localhost:3000/items
```

GET /items

Description:

This endpoint retrieves all items. The retrieved items differ depending on the role of the authenticated user.

URL: `http://localhost:3000/items`

Method: `GET`

Authentication required: No

URL Parameters: None

Data Parameters: None

Success Response:

- **Code:** 200 - OK
- **Content:** [{ id: 1, name: "Item 1", category: "Category 1", stock_quantity: 10 }, ...]

Error Responses:

- If the role is not supported:
  - **Code:** 500
  - **Content:** { message: "Unsupported role: {role}" }

- If there's a server error while processing the request:
  - **Code:** 500
  - **Content:** { message: error.message }

Notes:

- The items returned differ depending on the role of the authenticated user. Admins and regular users will see all items, while guests will only see items that have a stock quantity greater than 0.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

POST item(Admin)

http://localhost:3000/item

POST /item

Description:

This endpoint allows an admin to create a new item.

URL: `http://localhost:3000/item`

Method: `POST`

Authentication required: Yes, admin only

URL Parameters: None

Data Parameters:

`{ "name": [string], Required "price": [number], Required "stock_quantity": [number], Required "sku": [string], Required "categoryId": [integer], Required "img_url": [string], Required }`

Success Response:

- **Code:** 201 - Created
- **Content:** Created Item JSON Object

Error Responses:

- If fields are missing or incorrect:
  - **Code:** 400 - Bad Request
  - **Content:** { errors: [{ value, msg, param, location }] }
- If the item already exists:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Item is already created" }
- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: error.message }

Notes:

- Only admins can create new items.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```json
{
    "categoryId": 4,
    "sku": "BT152",
    "name": "Foundation",
    "price": 450.00,
    "stock_quantity": 10,
    "img_url": "http://example.com/image/foundation.jpg"
}
```

PUT item/id(Admin)

http://localhost:3000/item/33

PUT /item/:id

Description:

This endpoint allows an admin to update an existing item.

URL: `http://localhost:3000/item/:id`

Method: `PUT`

Authentication required: Yes, admin only

URL Parameters:

- id=[integer] - Required

Data Parameters:

`{ "name": [string], Optional "price": [number], Optional "stock_quantity": [number], Optional "sku": [string], Optional "categoryId": [integer], Optional "img_url": [string], Optional }`

Success Response:

Success Response:

- **Code:** 200 - OK

- **Content:** Updated Item JSON Object

Error Responses:

- If fields are incorrect:
  - **Code:** 400 - Bad Request
  - **Content:** { errors: [{ value, msg, param, location }] }
- If the item does not exist:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Item not found" }
- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred during item update" }

Notes:

- Only admins can update items.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

DELETE item/id(Admin)

```
http://localhost:3000/item/33
```

DELETE /item/:id

Description:

This endpoint allows an admin to delete an existing item.

URL: `http://localhost:3000/item/:id`

Method: `DELETE`

Authentication required: Yes, admin only

URL Parameters:

- id=[integer] - Required

Success Response:

- **Code:** 200 - OK
- **Content:** item deleted successfully

- **Content:** Item deleted successfully

Error Responses:

- If the item does not exist:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Item not found" }
- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred during item deletion" }

Notes:

- Only admins can delete items.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Category(ies)

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

GET categories(all users)

```
http://localhost:3000/categories
```

GET /categories

Description:

This endpoint retrieves all categories in the system.

URL: `http://localhost:3000/categories`

Method: `GET`

Authentication required: No, but the user's role (admin, user, or guest) will have an effect on the results.

Success Response:

- **Code:** 200 - OK
- **Content:** `[ { "id": 1, "name": "category1", "createdAt": "2021-12-17T14:36:28.000Z", "updatedAt": "2021-12-17T14:36:28.000Z" }, { "id": 2, "name": "category2", "createdAt": "2021-12-17T14:36:28.000Z", "updatedAt": "2021-12-17T14:36:28.000Z" } ]`

`2021-12-17T14:36:28.000Z", "updatedAt": "2021-12-17T14:36:28.000Z" } ]`

Error Responses:

- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred during categories retrieval" }

Notes:

- All users, regardless of their role (admin, user, or guest), receive the same category list.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

POST category(Admin)

```
http://localhost:3000/category
```

POST /category

Description:

This endpoint is used for creating new categories in the system.

URL: `http://localhost:3000/category`

Method: `POST`

Authentication required: Yes, only Admin

Payload: { "category": "CategoryName" }

Data Constraints:

`{ "category": "string" }`

Success Response:

- **Code:** 201 - Created
- **Content:** `{ "id": 3, "category": "CategoryName", "updatedAt": "2023-06-05T14:36:28.000Z", "createdAt": "2023-06-05T14:36:28.000Z" }`

Error Responses:

- If category name is not provided or is not a string:
  - **Code:** 400 - Bad Request
  - **Content:** { errors: [{ msg: "Name must be a string" }] }
- If category already exists:

- Code: 400 - Bad Request
  - **Content:** { message: "Category already exists" }

- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** `{` message: "An error occurred during category creation" }

Notes:

- Only admin users can create a new category.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```json
{
    "category" : "Tools"
}
```

PUT category(Admin)

http://localhost:3000/category/9

PUT /category/:id

Description:

This endpoint is used for updating the name of an existing category in the system.

URL: `http://localhost:3000/category/:id`

Method: `PUT`

Authentication required: Yes, only Admin

Payload: { "category": "NewCategoryName" }

Data Constraints:

`{ "category": "string" }`

Success Response:

- **Code:** 200 - OK
- **Content:** "Category updated successfully"

Error Responses:

- If category name is not provided or is not a string:
  - **Code:** 400 - Bad Request
  - **Content:** { errors: [{ msg: "Name must be a string" }] }
- If category is not found:
  - **Code:** 404 - Not Found
  - **Content:** { message: "Category not found" }
- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred during category update" }

Notes:

- Only admin users can update a category.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```json
{
    "category" : "Updated tools"
}
```

DELETE category(Admin)

http://localhost:3000/category/9

DELETE /category/:id

Description:

This endpoint is used for deleting an existing category in the system. It will check if there are any items associated with this category before deletion.

URL: `http://localhost:3000/category/:id`

Method: `DELETE`

Authentication required: Yes, only Admin

Success Response:

- **Code:** 200 - OK
- **Content:** "Category deleted successfully"

Error Responses:

- If category is not found:
  - **Code:** 404 - Not Found
  - **Content:** { message: "Category not found" }
- If category has associated items:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Category has associated items" }
- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred during category deletion" }

Notes:

- Only admin users can delete a category.
- Category cannot be deleted if it has associated items. Remove the items from the category first before trying to delete it.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Cart(s)

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

GET cart

```
http://localhost:3000/cart
```

GET /cart

Description:

This endpoint retrieves the current user's shopping cart.

URL: `http://localhost:3000/cart`

Method: `GET`

Authentication required: Yes, registered users only.

Success Response:

- **Code:** 200 - OK
- **Content:** The user's cart object including items in the cart or "Your cart is empty" if there are no items in the cart.

Error Responses:

- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- This endpoint excludes some attributes like status, createdAt, and updatedAt from the response.
- The items in the cart exclude stock_quantity, createdAt, and updatedAt.
- Only registered users can view their cart.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

GET all carts

```
http://localhost:3000/allcarts
```

GET /allcarts

Description:

This endpoint retrieves all the carts of all users.

URL: `http://localhost:3000/allcarts`

Method: `GET`

Authentication required: Yes, admin users only.

Success Response:

- **Code:** 200 - OK
- **Content:** Array of all carts with their associated user details and items in the cart.

Error Responses:

- If there's a server error while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- This endpoint can only be accessed by admin users.
- Each cart in the response includes user details (id, username, fullname, email), cart id, and an array of items in the cart (id, name, quantity).
- This endpoint uses a raw SQL query to fetch data, which then is processed into a JavaScript object.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

POST cart_item

```
http://localhost:3000/cart_item
```

POST /cart_item

Description:

This endpoint allows a registered user to add an item to their cart.

URL: `http://localhost:3000/cart_item`

Method: `POST`

Authentication required: Yes, registered users only.

Data constraints:

`{ "itemId": "integer", "quantity": "integer" }`

Success Response:

- **Code:** 201 - Created
- **Content:** "Item added to the cart"

Error Responses:

- If the provided data does not pass validation:
  - **Code:** 400 - Bad Request
  - **Content:** Array of validation errors
- If the item doesn't exist:
  - **Code:** 400 - Bad Request

    - **Code:** 400 - Bad Request
    - **Content:** { message: "Item not found" }
- If there's not enough stock of the item to fulfill the requested quantity:
    - **Code:** 400 - Bad Request
    - **Content:** { message: "Not enough stock for this item" }
- If there's an error on the server while processing the request:
    - **Code:** 500 - Internal Server Error
    - **Content:** { message: "An error occurred on the server" }

Notes:

- The itemId and quantity are both required for this endpoint and should be included in the request body.
- The itemId must be an integer and quantity must be an integer greater than 0.
- If the item already exists in the user's cart, the quantity will be updated instead of creating a new cart item.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```json
{
    "itemId" : 2,
    "quantity" : 1
}
```

PUT cart_item/id

http://localhost:3000/cart_item/1

PUT /cart_item/:id

Description:

This endpoint allows a registered user to update the quantity of an item in their cart.

URL: `http://localhost:3000/cart_item/:id`

Method: `PUT`

Authentication required: Yes, registered users only.

Data constraints:

`{ "quantity": "integer greater than 0" }`

Success Response:

- **Code:** 200 - OK
- **Content:** "Your item has been updated in the cart"

Error Responses:

- If the provided data does not pass validation:
  - **Code:** 400 - Bad Request
  - **Content:** Array of validation errors
- If the cart item doesn't exist:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Cart item not found" }
- If the cart doesn't exist for the cart item:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Cart not found for this cart item" }
- If the user is not authorized to modify the cart item:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "User not authorized to modify this cart item" }
- If there's not enough stock of the item to fulfill the requested quantity:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Not enough stock for this item" }
- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The id parameter in the URL should be the id of the cart item to be updated and should be an integer.
- The quantity is required for this endpoint and should be included in the request body, and it must be an integer greater than 0.
- This endpoint will check that the cart item exists, the cart exists for the cart item, and that the user is authorized to modify the cart item before making any changes.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Body raw (json)

json

```json
{
    "itemId" : 2,
    "quantity" : 4
}
```

DELETE cart_item/id

```
http://localhost:3000/cart/2
```

DELETE /cart_item/:id

Description:

This endpoint allows a registered user to remove an item from their cart.

URL: `http://localhost:3000/cart_item/:id`

Method: `DELETE`

Authentication required: Yes, registered users only.

Data constraints: None.

Success Response:

- **Code:** 200 - OK
- **Content:** `"item has been removed from the cart"`

Error Responses:

- If the cart item doesn't exist:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "Cannot find this cart item" }
- If the user is not authorized to modify the cart item:
  - **Code:** 400 - Bad Request
  - **Content:** { message: "User not authorized to modify this cart item" }
- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The id parameter in the URL should be the id of the cart item to be removed and should be an integer.
- This endpoint will check that the cart item exists and that the user is authorized to modify the cart item before deleting it.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

DELETE cart/id

```
http://localhost:3000/cart/2
```

DELETE /cart/:id

Description:

This endpoint allows a registered user to empty their cart.

URL: `http://localhost:3000/cart/:id`

Method: `DELETE`

Authentication required: Yes, registered users only.

Data constraints: None.

Success Response:

- **Code:** 200 - OK
- **Content:** "Your cart is emptied"

Error Responses:

- If the cart doesn't exist:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "Cart not found" }
- If the user is not authorized to modify the cart:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "Unauthorized to empty this cart" }
- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The id parameter in the URL should be the id of the cart to be emptied and should be an integer.
- This endpoint will check that the cart exists and that the user is authorized to modify the cart before emptying it.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

Order(s)

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection Sales and Stock System

POST order/id

http://localhost:3000/order/2

POST /order/:id

Description:

This endpoint allows a registered user to place an order from their cart.

URL: `http://localhost:3000/order/:id`

Method: `POST`

Authentication required: Yes, registered users only.

Data constraints: None.

Success Response:

- **Code:** 201 - CREATED
- **Content:**

`{ message: "Congratulations! You've successfully placed your order", orderId: [integer], status: [string], discount: [float], YourTotalBeforeDiscount: [float], YourTotalAfterDiscount: [float], items: [ { itemId: [integer], name: [string], quantity: [integer], price: [float] }, ] }`

Error Responses:

- If the order has already been placed or the cart is empty:
  - **Code:** 400 - BAD REQUEST
  - **Content:** { message: "This order has already been placed" } or { message: "Your cart is currently empty" }
- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The id parameter in the URL should be the id of the cart to be converted into an order and should be an integer.
- The server will check if the cart exists, if the order has not been placed yet, and if there are enough items in stock before placing the order.
- After placing the order, the cart will be emptied.
- The total price of the order will be calculated based on the items in the cart and any discounts applicable to the user.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

GET orders(Admin)

```
http://localhost:3000/allorders
```

GET /allorders

Description:

This endpoint retrieves all orders in the system. This endpoint is restricted to admin users only.

URL: `http://localhost:3000/allorders`

Method: `GET`

Authentication required: Yes, admin users only.

Data constraints: None.

Success Response:

- **Code:** 200 - OK
  `[{ "orderId": [integer], "total": [float], "status": [string], "createdAt": [datetime], "updatedAt": [datetime], "fullname": [string], "userName": [string], "userEmail": [string], "userDiscount": [float], "items": [{ "itemId": [integer], "name": [string], "description": [string], "price": [float], "SKU": [string], "quantity": [integer] }, ...] }, ...]`

Error Responses:

- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The endpoint will return all orders in the system, along with the associated user information and the items in each order.
- The items field in the response is an array that contains all the items in the order. Each item has its id, name, description, price, SKU, and the quantity of the item in the order.
- The fullname, userName, userEmail, and userDiscount fields contain information about the user who placed the order.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

PUT order/id(Admin)

```
http://localhost:3000/order/2
```

PUT /order/:id

Description:

This endpoint updates the status of an order. This endpoint is restricted to admin users only.

URL: `http://localhost:3000/order/:id`

Method: `PUT`

Authentication required: Yes, admin users only.

Data constraints:

`{ "status": "In Progress | Completed | Cancelled" }`

Success Response:

- **Code:** 200 - OK
- **Content:** Order status has been updated successfully

Error Responses:

- If the status provided is not in the list of valid statuses ["In Progress", "Completed", "Cancelled"]:
  - **Code:** 400 - Bad Request
  - **Content:** { errors: [{ "msg": "Invalid status" }] }
- If there's an error on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

- The status field is a string that indicates the current status of the order. The valid statuses are "In Progress", "Completed", and "Cancelled".

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token

GET orders(Registered User)

```
http://localhost:3000/orders
```

GET /orders

Description:

This endpoint retrieves all the orders for a user. If the authenticated user is an admin, it will return all orders from all users.

URL: `http://localhost:3000/orders`

Method: `GET`

Authentication required: Yes.

Query Parameters:

None.

Success Response:

- **Code:** 200 - OK
- **Content:** An array of orders each containing the following fields:
  - orderId - The ID of the order.
  - status - The current status of the order.
  - discount - The discount applied to the order.
  - total - The total cost of the order after the discount.
  - items - An array of items included in the order, each with the following fields:
    - itemId - The ID of the item.
    - name - The name of the item.
    - category - The category ID of the item.
    - quantity - The quantity of the item in the order.
    - price - The price of the item.
    - imageUrl - The URL of the item's image.

Error Responses:

- If an error occurs on the server while processing the request:
  - **Code:** 500 - Internal Server Error
  - **Content:** { message: "An error occurred on the server" }

Notes:

This endpoint behaves differently depending on the role of the authenticated user. If the user is an admin, all orders from all users are returned. Otherwise, only the orders belonging to the authenticated user are returned.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection Sales and Stock System

HEADERS

Authorization	Bearer User_Token