EP

Items

GET Get/Items

http://localhost:3000/items

- The GET Items page displays items for sale with their name, price, and stock levels.
- Guest Users can view in-stock items only.
- Authentication: Not required (accessible to Guest Users).

AUTHORIZATION Bearer Token

Token	<token>

PUT PUT/item/:id

http://localhost:3000/item/1

- The /item/:id endpoint allows Admin Users to update an existing item.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the item to update.
- The request body must include the updated information for the item, such as name, SKU, price, stock, image URL, and category ID.
- If the user is not an Admin User or the item/category is not found, appropriate responses are returned.
- After successful update, a 200 OK response is returned with the updated item details

Please see example for correct JSON format

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "name": "Updated Item Name",
  "sku": "ABC12345",
  "price": 19.99,
  "stock": 0,
  "img_url": "https://example.com/updated_image.jpg",
  "categoryId": 1
}
```

POST POST/Item

http://localhost:3000/item

StartFragment

- The /item endpoint allows Admin Users to create a new item.
- The user's information is extracted from the JWT for authentication.
- The request body must include the necessary information for the new item, such as name, SKU, price, stock, image URL, and category ID.
- If the user is not an Admin User, a 403 Forbidden response is returned.
- If the category is not found, a 404 Not Found response is returned.
- After successful creation, a 201 Created response is returned with the created item details.

EndFragment

Please see example for correct JSON format

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "name": "New Item",
  "sku": "NI001",
  "price": 50.99,
  "stock": 10,
  "categoryId": 10000
}
```

DELETE DEL/item/

http://localhost:3000/item/35

StartFragment

- The /item/:id endpoint allows Admin Users to delete an existing item.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the item to delete.
- If the user is not an Admin User or the item is not found, appropriate responses are returned.
- After successful deletion, a 200 OK response is returned.

EndFragment

AUTHORIZATION Bearer Token

Token	<token>

POST http://localhost:3000/item

http://localhost:3000/item

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "name": "New Item",
  "sku": "NI001",
  "price": 50.99,
  "stock": 10,
  "categoryId": 10000
}
```

Category

GET GETcategories

http://localhost:3000/categories

- Description: Returns all categories in the database.
- Authentication: Not required (accessible to Guest Users).
- Response:
- 200 OK: Returns an array of categories.
- 500 Internal Server Error: An error occurred while processing the request

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "name": "Updated Category"
}
```

POST POST/category

http://localhost:3000/category

- The /category endpoint allows Admin Users to create a new category.
- The user's information is extracted from the JWT for authentication.
- The request body must include the name of the category to create.
- If the user is not an Admin User, a 403 Forbidden response is returned.
- If the category name is missing, a 400 Bad Request response is returned.
- After successful creation, a 201 Created response is returned with the created category details

Please see example for JSON format.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
    "name": ""
}
```

PUT PUT/category/

http://localhost:3000/category/1

- The /category/:id endpoint allows Admin Users to update an existing category.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the category to update.
- The request body must include the name of the updated category.
- If the user is not an Admin User or the category is not found, appropriate responses are returned.
- After successful update, a 200 OK response is returned with the updated category details.

Please see example for JSON format

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "name": ""
}
```

DELETE DEL/category

http://localhost:3000/category/10

- The /category/:id endpoint allows Admin Users to delete an existing category.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the category to delete.
- If the user is not an Admin User, a 403 Forbidden response is returned.

- If the category is not found or has associated items, appropriate responses are returned.
- After successful deletion, a 200 OK response is returned.

AUTHORIZATION Bearer Token

Token	<token>

Cart

GET GET/cart

http://localhost:3000/cart

- The /cart endpoint retrieves the cart of the logged-in user.
- The user's information is extracted from the JWT for authentication.
- If the user is not logged in, a 401 Unauthorized response is returned.
- If the cart is found, a 200 OK response is returned with the cart details.
- If the cart is not found, a 404 Not Found response is returned.

AUTHORIZATION Bearer Token

Token	<token>

POST Po/cart_item

http://localhost:3000/cart_item

- The /cart_item endpoint allows registered users to add items to their cart.
- The user's information is extracted from the JWT for authentication.
- The request body must include the itemId of the item to add.
- An optional quantity can be specified, defaulting to 1 if not provided.
- If the user does not have a cart, a new cart is created and the item is added to the cart as a new cart item.
- If the item is out of stock, a 400 Bad Request response is returned.
- If the item already exists in the cart, the quantity is updated accordingly.
- After successful addition, a 201 Created response is returned with the cart item details.

Please see example for JSON format

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```
json

{
  "itemId": 10,
  "quantity": 3
}
```

PUT Put/cart_item

http://localhost:3000/cart_item/1

- The /cart_item/:id endpoint allows users to update the quantity of a specific item in their cart.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the cart item to update.
- The request body must include the itemId and quantity of the item in the cart.
- The user must have permission to update the cart item, and the item must exist in the database.

Please see example for JSON format

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```
json

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

DELETE Del/cart_item/

http://localhost:3000/cart_item/3

- The /cart_item/:id endpoint allows registred user to delete a specific item from their cart.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the cart item to delete.

- The user must have permission to delete the cart item, and the cart item must exist in the database.
- The user must also have a cart, and the cart must exist in the database.
- After successful deletion, a 200 OK response is returned.

AUTHORIZATION Bearer Token

Token	<token>

DELETE Del/cart/

http://localhost:3000/cart/1

- The /cart_item/:id endpoint allows registered users to delete all items from their cart.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the cart to delete.
- The user must have permission to delete the cart items , and the cart item must exist in the database.
- After successful deletion, a 200 OK response is returned.

AUTHORIZATION Bearer Token

Token	<token>

GET GET/allcarts

http://localhost:3000/allcarts

- Returns all carts that exist, including the items in those carts and the full names of the users to whom those carts belong.
- Authentication: Required (accessible to Admin Users only).
- Response:
- 200 OK: Returns an array of carts with user and item information.
- 403 Forbidden: Access denied for non-Admin Users.
- 500 Internal Server Error: An error occurred while processing the request

AUTHORIZATION Bearer Token

Token	<token>

Orders

GET GET/orders

http://localhost:3000/orders

- Registered Users with completed orders can view their orders.
- Previously fulfilled orders are also viewable.
- All orders have a status displayed to the User (e.g., In Progress, Completed, Cancelled, etc).

AUTHORIZATION Bearer Token

Token	<token>

POST POST/order:id

http://localhost:3000/order/10

- The /order/:id endpoint allows users to place an order for a specific item in their cart.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the item for which the order is being placed.
- The endpoint checks if the user has a cart and if the item exists in the user's cart.
- It verifies if the item is in stock and if the requested quantity is available.
- The order and order item are created in the database, and the cart item is deleted.
- The stock of the item is updated in the items table.
- The response includes the order item details, the total price after discount, and the quantity.

AUTHORIZATION Bearer Token

Token	<token>

GET GET/allorders

http://localhost:3000/allorders

- Description: Returns all orders that exist, including the items in those orders and the full names of the users to whom those orders belong, regardless of order status.
- Authentication: Required (accessible to Admin Users only).
- Response:
- 200 OK: Returns an array of all orders with item and user information.
- 403 Forbidden: Access denied for non-Admin Users.
- 500 Internal Server Error: An error occurred while processing the request.

AUTHORIZATION Bearer Token

Token <token>

PUT PUT/order/

http://localhost:3000/order/3

- The /order/:id endpoint allows Admin Users to update the status of an order.
- The user's information is extracted from the JWT for authentication.
- The id parameter represents the ID of the order to update.
- The request body must include the updated status for the order.
- Only Admin Users have access to this endpoint.
- The endpoint checks if the provided status is valid.
- The accepted statuses are "Completed", "Cancelled" and "In Process"
- It finds the order by ID and updates the status accordingly.
- If the order status is updated to "Cancelled," the items in the order are returned to stock.
- If the order is already Cancelled, it will not let you cancel again and will return a message.
- The response includes the updated order details.

AUTHORIZATION Bearer Token

Token <token>

Body raw (json)

json

```
{ "status": "In Process" }
```

Auth

POST AdmLogin

http://localhost:3000/login

AUTHORIZATION Bearer Token

Token <token>

Body raw (json)

```
json

{
  "username": "admin",
  "password": "P@ssword2023"
}
```

POST Login

http://localhost:3000/login

AUTHORIZATION Basic Auth

Username	<username>
Password	<password>

Body raw (json)

```
json

{
  "username": "john_doe4",
  "password": "P@ssword123"
}
```

POST signup

http://localhost:3000/signup

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

json

```
{
  "username": "john_doe4",
  "password": "P@ssword123",
  "email": "john.doe@example.com",
  "firstName": "John",
  "lastName": "Doe"
}
```

DELETE Delete User

http://localhost:3000/user/3

AUTHORIZATION Bearer Token

Token	<token>

Utility

POST Setup

http://localhost:3000/setup

- The /setup endpoint allows the database to be populated with initial data if the database is empty.
- The endpoint accepts a POST request without any request body.
- When a request is received, the endpoint checks if any items exist in the database.
- If items already exist, indicating that the database is already populated, a 409 Conflict response is returned.
- If no items exist, the endpoint makes an Axios GET request to an external API to fetch the initial data.
- The endpoint utilizes the axios library to make the GET request.
- The received data from the external API is processed and validated.
- If the data is valid, the items are populated in the database.
- Additionally, the roles are populated in the db.Role table, and an admin user is created if it doesn't already exist.
- The admin user's password is hashed using bcrypt for security.
- Finally, a response is sent indicating the successful population of the database.

AUTHORIZATION Bearer Token

Token	<token>

POST search

http://localhost:3000/search

- The /search endpoint allows users to search for items based on specified criteria.
- The endpoint accepts a POST request with a JSON payload containing search parameters.
- The request body can include one or more of the following parameters:
- itemName: Search query to match item names. Items with names containing the search query will be retrieved.
- categoryName: Search query to match category names. Items belonging to categories with names matching the search query will be retrieved.
- sku: Search query to match item SKU. Items with SKUs matching the search query will be retrieved.
- The endpoint performs the search based on the provided search options.
- It utilizes the Sequelize library and its Op (Operator) object to construct the search conditions.
- The items that match the search criteria are retrieved from the database.
- The response includes the retrieved items, along with their associated category information.
- If no items match the search criteria, an empty array is returned.

Body raw (json)

json

```
{
  "itemName": "nove",
  "categoryName": "books"
}
```