Inventory Management System API Documentation

Auth Controller

1. Login

- **Method**: POST
- **Endpoint**: /auth/login
- **Request Parameters**:
    - UserLoginDto (Request Body): Contains the user's login details, typically including the email and password.
- **Operation Summary**: Authenticates the user and generates an authentication token (JWT).
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A response body of type AuthResponse, which contains the generated token and role of the user.
- **Failure Response**:
    - Status Code: 500 Internal server error if the request is malformed or missing necessary fields

2. Register

- **Method**: POST
- **Endpoint**: /auth/register
- **Request Parameters**:
    - RegisterDto (Request Body): Contains the user's details for registration, including username, email, password.
- **Operation Summary**: Registers a new user by creating their account in the system.
- **Success Response**:
    - **Status Code**: 201 Created
    - **Response Body**: It return a success message.
- **Failure Response**:
    - **Status Code**: 500 Internal Server Error if there’s a server issue during registration

Product Controller

1. Create a New Product

- **Method**: POST
- **Endpoint**: /api/products/create
- **Request Body**: ProductRequestDTO
    - A JSON object containing details of the product we want to create (e.g., name, price, description).
- **Operation Summary**: Creates a new product in the inventory.
- **Success Response**:
    - Status Code: 201 Created
    - Response Body: "Product added successfully"
- **Failure Response**:
    - Status Code: 401 Unauthorized if user is not authorized.

2. Get All Products

- **Method**: GET
- **Endpoint**: /api/products
- **Request Parameters**: None
- **Operation Summary**: Retrieves a list of all products in the inventory.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A JSON array of Product objects.
- **Failure Response**:
    - Status Code: 500 Internal Server Error if there's an issue retrieving the products.

3. Get a Product by ID

- **Method**: GET
- **Endpoint**: /api/products/{productId}
- **Request Parameters**:
    - productId (Path Variable): The unique ID of the product we want to retrieve.
- **Operation Summary**: Retrieves details of a specific product by its ID.
- **Success Response**:
    - Status Code: 200 OK

    - Response Body: A JSON object representing the Product.
- **Failure Response**:
    - Status Code: 404 Not Found if the product is not found.

4. Update an Existing Product

- **Method**: PUT
- **Endpoint**: /api/products/{productId}
- **Request Parameters**:
    - productId (Path Variable): The ID of the product to update.
    - ProductRequestDTO (Request Body): The updated product data.
- **Operation Summary**: Updates an existing product.
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: The updated Product.
- **Failure Response**:
    - Status Code: 500 internal server error if the request body is invalid.
    - Status Code: 404 Not Found if the product with the specified ID does not exist.

5. Delete a Product

- **Method**: DELETE
- **Endpoint**: /api/products/{productId}
- **Request Parameters**:
    - productId (Path Variable): The ID of the product to delete.
- **Operation Summary**: Deletes a product from the inventory.
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: "Product deleted successfully."
- **Failure Response**:
    - Status Code: 404 Not Found if the product does not exist.
    - Status Code: 403 Forbidden if the user doesn't have the required role.

6. Get Products by Name

- **Method**: GET
- **Endpoint**: /api/products/productName
- **Request Parameters**:
    - productName (Query Parameter): The name of the product to search for.
- **Operation Summary**: Retrieves products from the inventory that match the provided name.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A JSON array of products matching the name.
- **Failure Response**:
    - Status Code: 404 Not Found if no products are found with the given name.

7. Import Products from CSV

- **Method**: POST
- **Endpoint**: /api/products/import
- **Request Parameters**:
    - file (Multipart File): The CSV file containing the products to be imported.
- **Operation Summary**: Imports products into the inventory from a CSV file.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A list of ProductRequestDTO objects representing the imported products.
- **Failure Response**:
    - Status Code: 400 Bad Request if the file is invalid or not a CSV file.

8. Export Products to CSV

- **Method**: GET
- **Endpoint**: /api/products/export

- **Request Parameters**: None
- **Operation Summary**: Exports all products in the inventory to a CSV file.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: The response will be a CSV file.
- **Failure Response**:
    - Status Code: 500 Internal Server Error if there's an issue during the export process.

Supplier Controller

1. Get All Suppliers

- **Method**: GET
- **Endpoint**: /api/supplier
- **Request Parameters**: None
- **Operation Summary**: Retrieves a list of all suppliers.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A JSON array of Supplier objects.
- **Failure Response**:
    - Status Code: 500 Internal Server Error if there's an issue retrieving the suppliers.

2. Get Supplier by ID

- **Method**: GET
- **Endpoint**: /api/supplier/{supplierId}
- **Request Parameters**:
    - supplierId (Path Variable): The unique ID of the supplier.
- **Operation Summary**: Retrieves details of a specific supplier by their ID.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: A JSON object representing the Supplier.
- **Failure Response**:

    - Status Code: 404 Not Found if the supplier with the specified ID is not found

3. Create a New Supplier

- **Method**: POST
- **Endpoint**: /api/supplier/create
- **Request Body**: SupplierDTO
    - A JSON object containing the details of the supplier to be created (e.g., name, contact, email).
- **Operation Summary**: Creates a new supplier in the database.
- **Success Response**:
    - Status Code: 201 Created
    - Response Body: The Supplier object that was created, including the generated id.
- **Failure Response**:
    - Status Code: 400 Bad Request if the request body is invalid.

4. Update an Existing Supplier

- **Method**: PUT
- **Endpoint**: /api/supplier/{supplierId}
- **Request Parameters**:
    - supplierId (Path Variable): The ID of the supplier to update.
    - SupplierDTO (Request Body): The updated supplier data.
- **Operation Summary**: Updates the details of an existing supplier by their ID.
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - Status Code: 200 OK
    - Response Body: The updated Supplier object.
- **Failure Response**:
    - Status Code: 404 Not Found if the supplier with the specified ID does not exist

5. Delete a Supplier

- **Method**: DELETE
- **Endpoint**: /api/supplier/{supplierId}

- **Request Parameters**:
    - supplierId (Path Variable): The ID of the supplier to delete.
- **Operation Summary**: Deletes a specific supplier from the database.
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - Status Code: 204 No Content
    - Response Body: "Supplier deleted successfully."
- **Failure Response**:
    - Status Code: 404 Not Found if the supplier does not exist.
    - Status Code: 403 Forbidden if the user doesn't have the required role

Order Controller

1. Get All Orders

- **Method**: GET
- **Endpoint**: /api/orders
- **Request Parameters**: None
- **Operation Summary**: Retrieve a list of all orders in the inventory.
- **Success Response**:
    - **Status Code**: 200 OK
    - **Response Body**: A list of OrderResponseDTO objects containing order details.
- **Failure Response**:
    - **Status Code**: 500 Internal Server Error if there’s a failure in fetching

2. Get Order by ID

- **Method**: GET
- **Endpoint**: /api/orders/{orderId}
- **Request Parameters**:
    - orderId (Path Variable): The ID of the order to retrieve.
- **Operation Summary**: Retrieve details of a specific order by its ID.

- **Success Response**:
    - **Status Code**: 200 OK
    - **Response Body**: An OrderResponseDTO object representing the order.
- **Failure Response**:
    - **Status Code**: 404 Not Found if the order with the given ID doesn't exist.

3. Create an Order

- **Method**: POST
- **Endpoint**: /api/orders/create
- **Request Body**: Orders
    - A JSON object containing the details of the order to create.
- **Operation Summary**: Creates a new order and saves it in the inventory.
- **Success Response**:
    - **Status Code**: 201 Created
    - **Response Body**: A success message "Order created successfully".
- **Failure Response**:
    - **Status Code**: 400 Bad Request if the request body is invalid.

4. Update Order Status

- **Method**: PUT
- **Endpoint**: /api/orders/{orderId}
- **Request Parameters**:
    - orderId (Path Variable): The ID of the order to update.
    - OrderStatus (Request Body): The new status of the order (e.g., PENDING, DELIVERED, CANCELED).
- **Operation Summary**: Updates the status of an order (e.g., change to "DELIVERED").
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - **Status Code**: 200 OK

    - **Response Body**: A success message "Order status updated successfully."
- **Failure Response**:
    - **Status Code**: 404 Not Found if the order does not exist.
    - **Status Code**: 403 Forbidden if the user does not have the required role.

5. Delete an Order

- **Method**: DELETE
- **Endpoint**: /api/orders/{orderId}
- **Request Parameters**:
    - orderId (Path Variable): The ID of the order to delete.
- **Operation Summary**: Deletes a specific order from the inventory.
- **Authorization**: Requires ROLE_ADMIN to access.
- **Success Response**:
    - **Status Code**: 204 No Content
    - **Response Body**: Empty response body (successful deletion).
- **Failure Response**:
    - **Status Code**: 404 Not Found if the order does not exist.
    - **Status Code**: 403 Forbidden if the user does not have the required role.

6. Find Pending Orders

- **Method**: GET
- **Endpoint**: /api/orders/pending
- **Request Parameters**: None
- **Operation Summary**: Retrieves a list of orders with status PENDING.
- **Success Response**:
    - **Status Code**: 200 OK
    - **Response Body**: A list of OrderResponseDTO objects where status is "PENDING".
- **Failure Response**:
    - **Status Code**: 500 Internal Server Error if there’s a failure in fetching the orders.

7. Import Orders from CSV

- **Method**: POST
- **Endpoint**: /api/orders/import
- **Request Parameters**:
    - file (MultipartFile): The CSV file containing orders to import.
- **Operation Summary**: Imports orders from a CSV file into the inventory.
- **Success Response**:
    - **Status Code**: 200 OK
    - **Response Body**: A list of OrderRequestDTO objects representing the imported orders.
- **Failure Response**:
    - **Status Code**: 400 Bad Request if the file format is incorrect or there is an issue with the data.

8. Export Orders to CSV

- **Method**: GET
- **Endpoint**: /api/orders/export
- **Request Parameters**: None
- **Operation Summary**: Exports all orders to a CSV file.
- **Success Response**:
    - **Status Code**: 200 OK
    - **Response Body**: A CSV file containing all orders.
- **Failure Response**:
    - **Status Code**: 500 Internal Server Error if there’s a failure in generating the CSV.