API Flow for Inventory Management System

This document outlines the flow of the Inventory Management System API, designed and implemented to manage inventory for the company's operations. The system includes endpoints for CRUD operations, security features, performance optimization, and extensive documentation for easy usage.

# 1. Authentication and Authorization

1.1 Login and Token Generation  
The first step in using the API is logging in to the system. After a successful login, the system returns a JSON Web Token (JWT) that contains the user's ID and other necessary information. This token is used for subsequent API requests to authenticate the user.  
The following is the flow for login and token generation:  
 - User sends a POST request to the '/login' endpoint with credentials (username and password).  
 - If the credentials are valid, the system responds with a JWT token.  
 - The user stores this token and includes it in the Authorization header for subsequent requests.  
 - If the token is invalid or expired, the user will need to log in again to generate a new token.

1.2 Role-Based Authorization  
Role-based authorization ensures that users can only access specific resources based on their roles. In the system, we define roles such as 'Admin' and 'User'. Depending on the user's role, they will have permissions to perform specific operations.  
For example, an Admin can perform all CRUD operations, while a User may only have 'Read' access.

# 2. Inventory Management CRUD Operations

The Inventory Management System provides endpoints for managing inventory items. The CRUD operations allow users to create, read, update, and delete inventory records. The following endpoints are available:

2.1 Create Inventory Item (POST /api/inventory)  
 - This endpoint allows the creation of a new inventory item.  
 - The user sends a POST request with the inventory item details (name, quantity, etc.).  
 - The system will create a new record and return the created inventory item along with a success status.

2.2 Read Inventory Items (GET /api/inventory)  
 - This endpoint retrieves a list of inventory items.  
 - It supports pagination, sorting, and filtering, allowing users to specify parameters like page number, sorting criteria, and filters for specific inventory properties.  
 - The system returns a paginated list of inventory items based on the request parameters.

2.3 Update Inventory Item (PUT /api/inventory/{id})  
 - This endpoint allows updating an existing inventory item.  
 - The user sends a PUT request with the updated details of the inventory item.  
 - The system will update the item and return the updated item in the response.

2.4 Delete Inventory Item (DELETE /api/inventory/{id})  
 - This endpoint deletes an inventory item.  
 - The user sends a DELETE request with the inventory item ID.  
 - The system will delete the item and return a success status.

# 3. Stock Level Alerts

3.1 Low Inventory Alerts  
The system tracks inventory stock levels and generates alerts for low inventory. Whenever an inventory item's quantity falls below a predefined threshold, an alert is triggered.  
These alerts are logged in the system and can be retrieved using the appropriate endpoint.

# 4. Security and Token Validation

4.1 Token-Based Authentication (JWT)  
The system uses JSON Web Tokens (JWT) for authentication. Upon login, a valid token is generated and provided to the user. This token must be included in the Authorization header for all protected API endpoints.  
4.2 Role-Based Authorization  
Role-based access control is implemented to restrict access to certain endpoints based on the user's role. The role of the user determines their ability to access certain resources or perform specific actions.

# 5. Performance Optimizations

5.1 Database Query Optimization  
The system uses Entity Framework Core to interact with the database. Query performance has been optimized by ensuring that only necessary data is fetched, and by implementing efficient querying techniques to minimize response times.  
5.2 Caching with Redis  
Frequently accessed data, such as inventory items, is cached using Redis to reduce database load and improve response times for repeated queries.

# 6. API Documentation

6.1 Swagger/OpenAPI Documentation  
The API is fully documented using Swagger/OpenAPI. This documentation provides detailed information about all available endpoints, request/response formats, and examples of how to interact with the API.  
Users can access the Swagger UI at the '/swagger' endpoint to explore and test the API interactively.

# 7. Conclusion

The API for the Inventory Management System has been designed and implemented with the necessary security, performance, and functionality requirements. The system is fully documented and ensure it performs as expected.