Inventory Control System API Documentation

Table of Contents

Authentication

Login

Authenticate user and generate JWT token.

Endpoint: `POST /api/auth/login`

Request Body:

```
{
    "identifier": "user@example.com",
    "password": "userPassword123"
}
```

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Login successful",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Error Response (401 Unauthorized):

```
{
    "status": 401,
    "message": "Invalid credentials"
}
```

Products

Create Product

Create a new product in the inventory.

Endpoint: `POST /api/products/create`

Request Body:

```
{
    "name": "Product Name",
    "description": "Product Description",
    "price": 99.99,
    "quantity": 100,
    "category": "Electronics",
    "supplierId": 1
}
```

Success Response (201 Created):

```
{
    "status": 201,
    "message": "Product created successfully",
    "product": {
        "id": 1,
        "name": "Product Name",
        "description": "Product Description",
        "price": 99.99,
        "quantity": 100,
        "category": "Electronics",
        "supplierId": 1
    }
}
```

Error Response (400 Bad Request):

```
{
    "status": 400,
    "errors": [
        "name: must not be blank",
        "price: must be greater than 0"
    ]
}
```

Update Product

Update an existing product's details.

Endpoint: `PUT /api/products/update/{id}`

Request Body:

```
{
    "name": "Updated Product Name",
    "description": "Updated Description",
    "price": 149.99,
    "quantity": 150,
    "category": "Electronics",
    "supplierId": 1
}
```

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Product updated successfully",
    "product": {
        "id": 1,
        "name": "Updated Product Name",
        "description": "Updated Description",
        "price": 149.99,
        "quantity": 150,
        "category": "Electronics",
        "supplierId": 1
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Product not found with ID: 1"
}
```

Get Product by ID

Retrieve product details by ID.

Endpoint: `GET /api/products/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "product": {
        "id": 1,
        "name": "Product Name",
        "description": "Product Description",
        "price": 99.99,
        "quantity": 100,
        "category": "Electronics",
        "supplierId": 1
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Product not found with ID: 1"
}
```

Get All Products

Retrieve all products in the inventory.

Endpoint: `GET /api/products/all`

Success Response (200 OK):

```
{
    "status": 200,
    "products": [
        {
            "id": 1,
            "name": "Product 1",
            "description": "Description 1",
            "price": 99.99,
            "quantity": 100,
            "category": "Electronics",
            "supplierId": 1
        },
        {
            "id": 2,
            "name": "Product 2",
            "description": "Description 2",
            "price": 149.99,
            "quantity": 150,
            "category": "Electronics",
            "supplierId": 1
        }
    ]
}
```

Delete Product

Delete a product from the inventory.

Endpoint: `DELETE /api/products/delete/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Product deleted successfully with ID: 1"
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Product not found with ID: 1"
}
```

Orders

Create Order

Create a new order.

Endpoint: `POST /api/orders/create`

Request Body:

```
{
    "customerId": 1,
    "products": [
        {
            "productId": 1,
            "quantity": 2
        },
        {
            "productId": 2,
            "quantity": 1
        }
    ],
    "shippingAddress": "123 Main St, City, Country",
    "orderStatus": "PENDING"
}
```

Success Response (201 Created):

```
{
    "status": 201,
    "message": "Order created successfully",
    "order": {
        "id": 1,
        "customerId": 1,
        "products": [
            {
                "productId": 1,
                "quantity": 2
            },
            {
                "productId": 2,
                "quantity": 1
            }
        ],
        "totalAmount": 349.97,
        "orderDate": "2024-11-22T10:30:00",
        "shippingAddress": "123 Main St, City, Country",
        "orderStatus": "PENDING"
    }
}
```

Error Response (400 Bad Request):

```
{
    "status": 400,
    "errors": [
        "customerId: must not be null",
        "products: must not be empty"
    ]
}
```

Update Order

Update an existing order's details.

Endpoint: `PUT /api/orders/update/{id}`

Request Body:

```
{
    "shippingAddress": "456 New St, City, Country",
    "orderStatus": "SHIPPED"
}
```

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Order updated successfully",
    "order": {
        "id": 1,
        "customerId": 1,
        "products": [
            {
                "productId": 1,
                "quantity": 2
            },
            {
                "productId": 2,
                "quantity": 1
            }
        ],
        "totalAmount": 349.97,
        "orderDate": "2024-11-22T10:30:00",
        "shippingAddress": "456 New St, City, Country",
        "orderStatus": "SHIPPED"
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Order not found with ID: 1"
}
```

Get Order by ID

Retrieve order details by ID.

Endpoint: `GET /api/orders/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "order": {
        "id": 1,
        "customerId": 1,
        "products": [
            {
                "productId": 1,
                "quantity": 2
            },
            {
                "productId": 2,
                "quantity": 1
            }
        ],
        "totalAmount": 349.97,
        "orderDate": "2024-11-22T10:30:00",
        "shippingAddress": "123 Main St, City, Country",
        "orderStatus": "PENDING"
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Order not found with ID: 1"
}
```

Get All Orders

Retrieve all orders.

Endpoint: `GET /api/orders/all`

Success Response (200 OK):

```
{
    "status": 200,
    "orders": [
        {
            "id": 1,
            "customerId": 1,
            "products": [
                {
                    "productId": 1,
                    "quantity": 2
                }
            ],
            "totalAmount": 199.98,
            "orderDate": "2024-11-22T10:30:00",
            "shippingAddress": "123 Main St, City, Country",
            "orderStatus": "PENDING"
        }
    ]
}
```

Delete Order

Delete an order.

Endpoint: `DELETE /api/orders/delete/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Order deleted successfully"
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Order not found with ID: 1"
}
```

Suppliers

Create Supplier

Create a new supplier.

Endpoint: `POST /api/suppliers/create`

Request Body:

```
{
    "name": "Supplier Name",
    "email": "supplier@example.com",
    "phone": "+1234567890",
    "address": "Supplier Address",
    "contactPerson": "John Doe"
}
```

Success Response (201 Created):

```
{
    "status": 201,
    "message": "Supplier created successfully",
    "supplier": {
        "id": 1,
        "name": "Supplier Name",
        "email": "supplier@example.com",
        "phone": "+1234567890",
        "address": "Supplier Address",
        "contactPerson": "John Doe"
    }
}
```

Error Response (400 Bad Request):

```
{
    "status": 400,
    "errors": [
        "name: must not be blank",
        "email: must be a valid email"
    ]
}
```

Update Supplier

Update an existing supplier's details.

Endpoint: `PUT /api/suppliers/update/{id}`

Request Body:

```
{
    "name": "Updated Supplier Name",
    "email": "updated@example.com",
    "phone": "+1987654321",
    "address": "Updated Address",
    "contactPerson": "Jane Doe"
}
```

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Supplier updated successfully",
    "supplier": {
        "id": 1,
        "name": "Updated Supplier Name",
        "email": "updated@example.com",
        "phone": "+1987654321",
        "address": "Updated Address",
        "contactPerson": "Jane Doe"
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Supplier not found"
}
```

Get Supplier by ID

Retrieve supplier details by ID.

Endpoint: `GET /api/suppliers/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "supplier": {
        "id": 1,
        "name": "Supplier Name",
        "email": "supplier@example.com",
        "phone": "+1234567890",
        "address": "Supplier Address",
        "contactPerson": "John Doe"
    }
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Supplier not found"
}
```

Get All Suppliers

Retrieve all suppliers.

Endpoint: `GET /api/suppliers/all`

Success Response (200 OK):

```
{
    "status": 200,
    "suppliers": [
        {
            "id": 1,
            "name": "Supplier 1",
            "email": "supplier1@example.com",
            "phone": "+1234567890",
            "address": "Address 1",
            "contactPerson": "John Doe"
        },
        {
            "id": 2,
            "name": "Supplier 2",
            "email": "supplier2@example.com",
            "phone": "+1987654321",
            "address": "Address 2",
            "contactPerson": "Jane Doe"
        }
    ]
}
```

Delete Supplier

Delete a supplier.

Endpoint: `DELETE /api/suppliers/delete/{id}`

Success Response (200 OK):

```
{
    "status": 200,
    "message": "Supplier deleted successfully with ID: 1"
}
```

Error Response (404 Not Found):

```
{
    "status": 404,
    "message": "Supplier not found with ID: 1"
}
```

Users

Register User

Creates a new user account in the system.

Endpoint: `POST /api/users/register`

Request Body:

```
{
    "username": "john_doe",
    "email": "john.doe@example.com",
    "password": "securePassword123",
    "firstName": "John",
    "lastName": "Doe",
    "role": "ADMIN"
}
```

Successful Response (201 Created):

```
{
    "status": 201,
    "message": "User registered successfully",
    "user": {
        "id": 1,
        "username": "john_doe",
        "email": "john.doe@example.com",
        "firstName": "John",
        "lastName": "Doe",
        "role": "ADMIN"
    }
}
```

Failed Response (400 Bad Request):

```
{
    "status": 400,
    "errors": [
        "username: must not be blank",
        "email: must be a valid email address",
        "password: must be at least 8 characters long"
    ]
}
```

Update User

Updates existing user information.

Endpoint: `PUT /api/users/update/{userId}`

Request Body:

```
{
    "email": "john.updated@example.com",
    "firstName": "Johnny",
    "lastName": "Doe",
    "role": "USER"
}
```

Successful Response (200 OK):

```
{
    "status": 200,
    "message": "User updated successfully",
    "user": {
        "id": 1,
        "username": "john_doe",
        "email": "john.updated@example.com",
        "firstName": "Johnny",
        "lastName": "Doe",
        "role": "USER"
    }
}
```

Failed Response (404 Not Found):

```
{
    "status": 404,
    "message": "User not found with ID: 1"
}
```

Get User by ID

Retrieves user information by their ID.

Endpoint: `GET /api/users/{userId}`

Successful Response (200 OK):

```
{
    "status": 200,
    "user": {
        "id": 1,
        "username": "john_doe",
        "email": "john.doe@example.com",
        "firstName": "John",
        "lastName": "Doe",
        "role": "ADMIN"
    }
}
```

Failed Response (404 Not Found):

```
{
    "status": 404,
    "message": "User not found with ID: 1"
}
```

Get All Users

Retrieves a list of all users in the system.

Endpoint: `GET /api/users/all`

Successful Response (200 OK):

```
{
    "status": 200,
    "users": [
        {
            "id": 1,
            "username": "john_doe",
            "email": "john.doe@example.com",
            "firstName": "John",
            "lastName": "Doe",
            "role": "ADMIN"
        },
        {
            "id": 2,
            "username": "jane_smith",
            "email": "jane.smith@example.com",
            "firstName": "Jane",
            "lastName": "Smith",
            "role": "USER"
        }
    ]
}
```

Delete User

Removes a user from the system.

Endpoint: `DELETE /api/users/delete/{userId}`

Successful Response (200 OK):

```
{
    "status": 200,
    "message": "User with ID: 1 deleted successfully."
}
```

Failed Response (404 Not Found):

```
{
    "status": 404,
    "message": "User not found with ID: 1. Unable to delete non-existent user."
}
```

CSV Operations

Import Products

Imports product data from a CSV file.

Endpoint: `POST /api/csv/import/products`

Request:

- Content-Type: multipart/form-data
- Parameter: file (CSV file)

CSV Format:

```
id,name,description,price,quantity,category
1,Product A,Description A,29.99,100,Electronics
2,Product B,Description B,19.99,50,Books
```

Successful Response (200 OK):

```
{
    "status": "success",
    "message": "Products imported successfully.",
    "data": [
        {
            "id": 1,
            "name": "Product A",
            "description": "Description A",
            "price": 29.99,
            "quantity": 100,
            "category": "Electronics"
        },
        {
            "id": 2,
            "name": "Product B",
            "description": "Description B",
            "price": 19.99,
            "quantity": 50,
            "category": "Books"
        }
    ]
}
```

Failed Response (400 Bad Request):

```
{
    "status": "error",
    "message": "The uploaded file is empty."
}
```

Failed Response (500 Internal Server Error):

```
{
    "status": "error",
    "message": "Failed to import products."
}
```

Export Products

Exports product data to a CSV file.

Endpoint: `POST /api/csv/export/products`

Request Body (optional):

```
"D:\exports\products.csv"
```

Successful Response (200 OK):

```
{
    "status": "success",
    "message": "Products exported successfully.",
    "filePath": "D:\JAVA\inventory-control-system-exports\products-1637012345678.csv"
}
```

Failed Response (500 Internal Server Error):

```
{
    "status": "error",
    "message": "Failed to export products."
}
```

Import Orders

Imports order data from a CSV file.

Endpoint: `POST /api/csv/import/orders`

Request:

- Content-Type: multipart/form-data
- Parameter: file (CSV file)

CSV Format:

```
id,orderDate,customerId,status,totalAmount
1,2024-03-21,100,PENDING,149.99
2,2024-03-22,101,COMPLETED,299.99
```

Successful Response (200 OK):

```
{
    "status": "success",
    "message": "Orders imported successfully.",
    "data": [
        {
            "id": 1,
            "orderDate": "2024-03-21",
            "customerId": 100,
            "status": "PENDING",
            "totalAmount": 149.99
        },
        {
            "id": 2,
            "orderDate": "2024-03-22",
            "customerId": 101,
            "status": "COMPLETED",
            "totalAmount": 299.99
        }
    ]
}
```

Failed Response (400 Bad Request):

```
{
    "status": "error",
    "message": "The uploaded file is empty."
}
```

Failed Response (500 Internal Server Error):

```
{
    "status": "error",
    "message": "Failed to import orders."
}
```

Export Orders

Exports order data to a CSV file.

Endpoint: `POST /api/csv/export/orders`

Request Body (optional):

```
"D:\exports\orders.csv"
```

Successful Response (200 OK):

```
{
    "status": "success",
    "message": "Orders exported successfully.",
    "filePath": "D:\JAVA\inventory-control-system-exports\orders-1637012345678.csv"
}
```

Failed Response (500 Internal Server Error):

```
{
    "status": "error",
    "message": "Failed to export orders."
}
```