Project Overview

The **Inventory Management System** (IMS) is a web-based application designed to help businesses and organizations manage their inventory, track stock levels, and handle product transactions (e.g., sales, purchases). The system allows users to add, edit, delete, and track products in the inventory. It supports **role-based authentication** (admin, manager, and staff) and provides features such as low-stock alerts, product categories, and reporting.

This project leverages **ASP.NET Core MVC**, **ASP.NET Core Web API**, **Entity Framework Core**, **ASP.NET Core Identity**, and **Role-based Authorization** to build a secure and scalable inventory management platform. The system will also allow administrators to manage users, assign roles, and define access levels based on user roles.

Key Features

1. **User Authentication and Role Management**:
    - Users can register, log in, and authenticate using **ASP.NET Core Identity**.
    - The system uses **role-based authentication**:
        - **Admins**: Full control over the system, including user management and access control.
        - **Managers**: Can manage products, view reports, and generate transactions.
        - **Staff**: Can view product information and handle stock transactions (e.g., sales, purchases).
    - **Role-based Access Control (RBAC)** ensures that only users with appropriate roles can access certain features.
2. **Product Management**:
    - **Admins** and **Managers** can add, update, and delete products in the inventory.
    - Each product has attributes such as **name**, **description**, **category**, **price**, **quantity in stock**, and **supplier**.
    - The system allows bulk import and export of product data via CSV files.
3. **Stock Transactions**:
    - The system tracks both **incoming** (purchases) and **outgoing** (sales) transactions for each product.
    - **Staff** can record stock movements when items are sold or purchased.
    - **Transaction History** is maintained for audit purposes, showing the date, quantity, and type of transaction (sale or purchase).

4. **Inventory Alerts**:
    - The system sends **low-stock alerts** when the quantity of a product falls below a predefined threshold.
    - Alerts can be sent via **email** or displayed on the dashboard.
5. **Category Management**:
    - Products can be assigned to categories (e.g., electronics, furniture, office supplies).
    - Admins can create, edit, and delete categories for better organization and product management.
6. **Reporting and Analytics**:
    - **Admins** and **Managers** can view **inventory reports**, including total stock value, total sales, total purchases, and product performance.
    - **Sales reports** track total sales, total revenue, and most popular products.
7. **API Endpoints**:
    - The system exposes RESTful APIs to perform CRUD operations on products, transactions, and categories.
    - APIs support retrieving inventory data, adding products, and managing transactions.
8. **Email Notifications**:
    - **Admin/Managers** receive **low-stock alerts** and notifications on pending stock purchases.
    - **Email notifications** for successful product addition, removal, and stock level changes.

Technologies Used

- **ASP.NET Core MVC**: For building the front-end of the inventory management system, allowing users to interact with products, transactions, and reports.
- **ASP.NET Core Web API**: To expose backend services for managing products, categories, and transactions through RESTful endpoints.
- **Entity Framework Core**: To manage the database and perform CRUD operations on products, users, and transactions.
- **ASP.NET Core Identity**: To manage user authentication, authorization, and role-based access control.
- **SignalR (Optional)**: For real-time updates on stock levels or low-stock alerts.

- **SMTP Service**: For sending email notifications (low-stock alerts, transaction confirmations).
- **CSV Import/Export**: To enable bulk import/export of product data and transaction history.
- **SQL Server**: For data storage (products, transactions, categories, user data).

System Architecture

1. **Frontend (ASP.NET Core MVC)**:
    - The MVC architecture handles the user interface, including product management, transaction logging, report generation, and user role-specific views.
    - It displays inventory data such as product lists, sales reports, and stock levels.
2. **Backend (ASP.NET Core Web API)**:
    - The API exposes endpoints to manage products, categories, users, and transactions.
    - It processes requests for adding products, recording sales and purchase transactions, and generating reports.
3. **Database (Entity Framework Core)**:
    - EF Core manages the database schema and relationships between products, transactions, categories, and users.
    - The database supports product data, transaction logs, stock movements, and user details.
4. **Email Notification System**:
    - When stock levels fall below a set threshold, the system sends low-stock alerts via email to managers or admins.
    - Email notifications also confirm successful transactions (sales/purchases) and product additions.
5. **CSV Import/Export**:
    - Users (admins and managers) can import products and transactions in bulk via CSV files and export product data for reporting purposes.

User Stories

1. User Registration and Role Management

- **As a new user**, I want to register for an account to access the inventory management system.
- **As a registered user**, I want to log in securely to access my dashboard.
- **As an admin**, I want to manage user roles (admin, manager, staff) and ensure the appropriate access for each user.

2. Product Management

- **As an admin or manager**, I want to add, edit, and delete products in the inventory.
- **As a manager**, I want to see a list of all products and track their quantity, price, and category.
- **As a staff member**, I want to view product details (name, price, and available stock) when processing sales.

3. Stock Transactions

- **As a staff member**, I want to record a sale when a customer buys a product, reducing the available stock.
- **As a staff member**, I want to record a purchase when new stock is received from suppliers.
- **As an admin or manager**, I want to view a history of all stock transactions for auditing and reporting purposes.

4. Inventory Alerts

- **As an admin or manager**, I want to receive low-stock alerts when a product's quantity falls below the predefined threshold.
- **As a staff member**, I want to be notified about products that are running low on stock to assist in restocking decisions.

5. Reporting and Analytics

- **As an admin**, I want to generate reports on total stock value, sales performance, and revenue.
- **As a manager**, I want to analyze the sales performance of individual products and identify top-selling items.

6. Category Management

- **As an admin**, I want to create, edit, and delete product categories for better organization.
- **As a manager**, I want to filter products based on categories to quickly find the items I need.

Database Design

Entities:

1. **User**:
    - Id (Primary Key)
    - Username
    - Email
    - PasswordHash
    - Role (Admin, Manager, Staff)
2. **Product**:
    - Id (Primary Key)
    - Name
    - Description
    - Price
    - QuantityInStock
    - CategoryId (Foreign Key referencing Category)
    - Supplier
    - CreatedAt
    - UpdatedAt
3. **Transaction**:
    - Id (Primary Key)
    - ProductId (Foreign Key referencing Product)
    - Quantity
    - Type (Sale or Purchase)
    - Date
    - UserId (Foreign Key referencing User)
    - TotalAmount
4. **Category**:
    - Id (Primary Key)
    - Name
    - Description

5. **LowStockAlert**:
    - Id (Primary Key)
    - ProductId (Foreign Key referencing Product)
    - Threshold
    - AlertSent (Boolean)
    - Date
6. **Payment**:
    - Id (Primary Key)
    - TransactionId (Foreign Key referencing Transaction)
    - PaymentMethod (Credit Card, Cash, etc.)
    - Amount
    - PaymentStatus (Completed, Pending, Failed)

APIs and Endpoints

- **POST** /api/auth/register: Register a new user.
- **POST** /api/auth/login: Log in a user.
- **GET** /api/products: Retrieve a list of all products.
- **GET** /api/products/{id}: Retrieve details of a specific product.
- **POST** /api/products: Add a new product (for admin and manager).
- **PUT** /api/products/{id}: Update product details (for admin and manager).
- **DELETE** /api/products/{id}: Delete a product (for admin).
- **POST** /api/transactions: Record a transaction (sale or purchase).
- **GET** /api/transactions: Retrieve transaction history.
- **POST** /api/lowstockalerts: Trigger low-stock alert for products.
- **GET** /api/reports: Generate inventory reports (sales, stock value, etc.).

CSV Import/Export

- **CSV Import**: Allow admins and managers to import product data and transaction history in bulk from CSV files.

- **CSV Export**: Allow exporting product inventory, transaction logs, and stock reports to CSV for analysis.