

Web Development Repair assessment

Inventory management system

Description

Stockify is an internal web application developed for a logistics company to digitalize their inventory flow. The system addresses the lack of clarity regarding product locations and supplier responsibilities. The application enables administrators to manage products, assign warehouse locations, and monitor current stock levels.

The project emphasizes a robust ASP.NET backend and a functional React (TypeScript) frontend. Note that the styling/CSS of the project is outside the scope of the assessment. The use of CSS libraries or UI frameworks is permitted, and you are free to suggest alternative styling solutions. Your task is to implement the workflow as stated in the project requirements.

There is already a template given so you can focus on building the features. The template consists of:

- A backend in .NET with EF Core and a Database Context
- A main controller and service for you to complete
- A Frontend in React with a basic file structure and basic markup.

Create a new GitHub Repository and share a link with your teacher before starting on the assignment. During the assessment you will be asked to present and run your project.

Database ERD

The application consists of the following entities:

- **Admin:** Has access to the dashboard and can add products to the warehouse or manage inventory.
- **Product:** The core of the inventory. The application must display an overview of products currently in stock.
- **Warehouse:** Each product has a physical storage location. A warehouse also has a manager who can log in to manage that specific location.
- **Inventory:** The junction table (link table) that tracks stock levels per location.
- **Supplier:** The external party providing products. Every product is linked to a supplier.

Relationships:

- Product [N-N] Warehouse: Inventory serves as the junction table between Product and Warehouse.
 - Product [N-1] Inventory
 - Inventory [N-1] Warehouse
- Supplier [1-N] Product: One product has one supplier; one supplier can provide multiple products.

Requirements

Frontend

- 1.1. Develop a React component that represents the login screen. There should be a logical reasoning behind error messages given to the end-user when the login fails.
- 1.2. Develop an administrative dashboard from where the admin can do the following:
 - Create products
 - Update products
 - Delete products
 - View product details
- 1.3. Homepage of the application should show a list of available products, which is after the admin is logged in.
- 1.4. The admin should be able to add a Product to a Warehouse, as in from the list of given products the admin can create a relationship from Product to Warehouse.
- 1.5. Warehouse managers can login to view the details of their own holding.
- 1.6. Multipage page navigation with React Router, with a minimum of these routes:
 - Home page
 - Login page
 - Product detail page
- 1.7. A warehouse manager is able to also update the stock of its own inventory

Backend

- 2.1. Develop the backend for the login system for the admin user.
- 2.2. Create a CRUD API Controller for the Product entity.
- 2.3. Extend the login service in such a way that it can handle multiple roles, so that a Warehouse manager can also login into the system.
- 2.4. There needs to be an endpoint where the warehouse manager can update the inventory.
- 2.5. There needs to be an endpoint where the warehouse manager can modify products.
- 2.6. Make an endpoint where a warehouse manager can add existing products to its own inventory.