Warehouse Manager

Implementation Report

2025 Alja Eremic

Implementing a large information system inevitably comes with significant challenges. Starting with the back end, I found it to be more difficult than I initially anticipated. One reason was that I greatly overestimated my ability to implement the front end, which resulted in developing several complex back-end functions that ultimately went unused.

Additionally, I chose to include two many-to-many relationships in my database: one between classes and items (classItem), and another for inventory, linking items to multiple locations. This design was intended to allow an item to exist in multiple locations while restricting it to only one class per location. However, this requirement made writing queries considerably more complicated.

A prime example of this complexity is the query that takes a location_id and returns all items (along with their attributes) and the associated inventory data for that location - Picture 1.

```
data.getAllItemsInLocation = (location_id, class_id, nameLike) => {

    if (class_id && nameLike && nameLike.trim() !== "") {
        return new Promise((resolve, reject) => {
            const query = `
                SELECT
                    Item.*,
                    ItemClass.*,
                    Class.*,
                    Inventory.item_quantity
                FROM Inventory
                JOIN Item ON Item.item_id = Inventory.item_id
                JOIN ItemClass ON ItemClass.item_id = Item.item_id
                JOIN Class ON Class.class_id = ItemClass.class_id
                WHERE Inventory.location_id = ? AND Class.class_id = ?
                AND Item.item_name LIKE ?
            `;
```

Picture 1.

On the front end, the implementation was significantly impacted by the limited time allocated to the back end. As a result, not all planned features could be completed. One particular challenge that stood out was loading images efficiently. This issue was eventually resolved by exposing a static upload folder via Express, which allowed for easier access to the images. Additionally, supporting multiple file extensions for saved files helped address related problems.

Implementation Overview by Functionalities

Note: The points are listed in the order they appear in my seminar report.

1. Item Overview Display
This core functionality displays detailed information about any item in the system—either showing all items at once for users or individual items for employees. It was the first and most difficult feature to implement, primarily because of the complex SQL query (referenced in Picture 1) that underpins it. Coordinating multiple React components to work seamlessly together was also a challenge.

2. Item Management
This feature allows employees to adjust item quantities and add new items to the system. Implementing sell/restock functionality was relatively straightforward. However, item creation proved more challenging due to the large number of required parameters (such as item name, article number, EAN, price, etc.). Tracking and passing all these variables to the back end was a complex task.

Note; item deletion, class creation and class deletion are also implemented even though they are not shown in the video so that video could be kept relatively short. Implementation of these features was relatively straight forward, the hardest part was making sure that class doesn't have any items before deleting it.

6. Item Browsing
Item browsing enables users to quickly find items by either selecting a class (for customers) or searching by name. Employees can also search by EAN (European Article Number), which is typically scanned from the item barcode for swift input. The main difficulty here was the backend query, which built upon the complex query mentioned earlier (Picture 1).

8. History Overview
This functionality allows employees to view past operations such as sales, restocks, and item creation performed by colleagues. One of the hardest aspects was designing the interface to be at least “usable,” as styling took considerable time. Additionally, joining data from multiple tables based solely on entries in the OperationLog table was challenging.

9. Role Management
This feature restricts system functionalities based on user roles. While overall not difficult to implement, the critical part was reliably verifying user roles from the database.

Github Repository : https://github.com/aljagaming/Warehouse-Manager

Youtube Video : https://youtu.be/orLmqsadEbI

Link to the app : http://88.200.63.148:4073/