Information system for Warehouse Management

Alja Eremić

Definition of the problem

Since their inception, big retail stores such as OBI, Harvey Norman and Big Bang have been known to have a constant need to efficiently track and manage merchandise-related information; What items does the store sell (inventory)? What is the position of each item in the store/warehouse? How many of a given item is there, should a restock be ordered or should some “overflowing” items be withdrawn from the store back into the warehouse. As well as numerous other details of a given item such as its ID, EAN/IAN (European/International Article number), picture, its dimensions, price, available colors/variants, power, usage etc.

We can observe this problem from more than one perspective;

Customer Janez, living two hours from the nearest Harvey Norman, is looking to buy a white drawer that fits between his bed and the wall. Without knowing whether the store has a suitable drawer in stock, he risks wasting time and resources. An effective information system would allow Janez to check availability, dimensions, and color options remotely - saving him a potential four-hour trip.

*In-Store-**Employee*** at Obi - Nina, is arranging products on display and needs 10 additional circular saws. Without knowing if more are in the warehouse, she may have to interrupt her task to search manually. Similarly, when encountering an unpriced item, she needs a quick way to access pricing and product details. A centralized system would streamline her workflow and increase productivity.

*Store **manager*** - Sasha oversees operations and inventory. She needs real-time insights into stock levels to reorder items when they run out, or to cancel restocking for items that aren't selling. Additionally, she may analyze historical data to identify sales trends. Furthermore she may adjust store layout, or reclassify inventory. A reliable information system is crucial for data-driven decision-making and inventory optimization.

WHM (WareHouse Manager) is an information management system designed for large-scale retailers. WHM enables real-time visibility and control over warehouse and store inventory, offering a unified solution for customers, employees, and managers. It is particularly suited for companies operating in the “Big-box” retail sector, including OBI, Harvey Norman, and Decathlon.

Functional and Non-functional requirements

Functional requirements

The system should enable the following functionalities:

1. Item overview display

- Users can view information about a specific item; its name, price, description, EAN …
- The information displayed is based on the user's role;
  Customers see limited item details (e.g., price, image, description, etc).
  Employees and **managers** can view all internal item details, such as Article Number and EAN codes.

2. Item management and Classification

- **Employees** update the quantity of the given item at the given inventory.
- **Managers** can create new or remove old items as well edit attributes of existing items.
- **Managers** can also update which class an item belongs to.

3. Class management

- **Managers** can create, read, update and delete classes. A typical example of updating a class would be either renaming it or repositioning it.

4. Inventory management

- **Managers** can create and delete Inventories.

5. Location Management

Managers can manage storage locations within the system:
- Add new location where inventory is stored (e.g. new warehouse)
- Remove old locations that are no longer in use
- Edit location details (name, website…)
- View all items and quantities stored at a specific location.

6. Browsing Items

- **Customers** can browse or search items using the item's class and name.
- **Employees** and **managers** can also perform advanced searches by Article numbers or EANs.

7. Create Restock list

- Users can create a **Restock List** that identifies which items need to be replenished based on current stock levels.

8. Operation History

- The system logs all significant item-related actions (e.g., sales, restocks, stock transfers, and reclassifications) and presents them in chronological order for visualisation. Employees and managers have access to this feature.

9. Managing user roles

- Users must register and be assigned a role: **customer**, **employee**, or **manager**.
- System functionality is restricted based on role:
  Customers can only browse and view limited item details.
  Employees can view all item information and update item quantity.
  Managers have full access to system functionalities.

Nonfunctional requirements

1. *Performance*
The system must respond to user requests (e.g., viewing item overview) within 2 seconds under defined normal load.
The system should support at least 100 concurrent users without noticeable performance degradation.

2. *Security*
All private user data must be encrypted (at rest and in transit).
User authentication is required to access all system functionality.

3. *Capacity / Information*
The system must store and enable access to at least 10,000 distinct items, with capacity to scale as inventory grows.

4. *Availability*
The system should be accessible 24/7; maintenance windows must occur outside of peak hours (e.g., when stores are closed).

5. *Consistency*
After any update, the system must reach a consistent state within 15 minutes to ensure accurate inventory visibility.

6. *Recoverability*
In the event of failure, the system should be recoverable to a consistent, operational state within 1 hour.

7. *Data Storage & Backup*
Data must be stored securely for a minimum of 3 years.
Perform weekly backups to two geographically separate locations.

8. *Usability*
The UI should balance simplicity with functionality; it is acceptable to have a steep learning curve if it leads to greater efficiency for power users like managers and warehouse personnel.

9. *Implementation & Compatibility*
The system must be implemented as a web application, ensuring compatibility across modern browsers and devices.

10. *Maintainability*.
The system should require minimal weekly maintenance.
Major updates are limited to 1–2 per year, minimizing disruption and downtime.

Feasibility study

One proposed solution to the problem of inefficient warehouse and in-store inventory management is to develop a centralized web-based Information System - Warehouse Manager (WHM). The system would provide distinct user experiences for customers, employees, and managers by offering real-time visibility into stock levels, item attributes, and warehouse logistics. From a technical perspective, the most complex functionality involves role-based data access and the classification of inventory into categories and positions within the warehouse. However, these features can be implemented using long established technologies such as SQL databases and modern server-side frameworks like Node.js.

Other expected system functionalities - such as user authentication, role-based permissions, item browsing, restock list generation, and item operation history logging - are well-supported in current development ecosystems and can be achieved using standard components. For data storage and access, any relational database (PostgreSQL or MySQL) would be sufficient to handle the scale of operations, as the system is expected to store fewer than 100,000 records initially. The system can be deployed on a cloud platform or a local server depending on the retailer's infrastructure.

From a **legal standpoint**, user login credentials fall under personal data protection regulations such as the GDPR (General Data Protection Regulation). The system henceforth must enforce encrypted connections (TLS/HTTPS), restrict data access based on roles, and maintain audit logs without storing unnecessary personally identifiable information. Besides GDPR legal risk is low.

Economically, the solution is highly viable. WHM can be developed using open-source technologies, with the main cost being development time and optional hardware for on-premise hosting. Since the system introduces measurable improvements in stock control, customer satisfaction, and employee efficiency, the return on investment could be significant. For example, reducing manual inventory checks and unnecessary restocking can result in major time savings and cost reductions.

Organizationally and **socially**, the WHM system presents minimal disruption. Customers benefit from remote inventory browsing, reducing wasted store visits. Employees gain a streamlined interface to retrieve or update stock data without breaking workflow. Managers receive robust tools for strategic planning, reducing reliance on outdated paper-based methods or fragmented spreadsheets. Because the system is tailored to mirror current warehouse and retail workflows, no drastic behavioral shifts are needed from users—only a period of onboarding and training, particularly for warehouse personnel.

Warehouse Management system is technically feasible, legally compliant, economically justified, and socially acceptable. The project can be developed using common web technologies, hosted affordably, and rolled out in phases to limit operational disruption. Its ability to centralize and

enhance warehouse data access makes it a strong candidate for real-world deployment in “Big-box” retail environments.

Logical Design

Matrix user role/functions.

Table 1 - Matrix user role/functions.
Note CRUD - Create, Read, Update and delete*

Functions	Costumer	Employee	Manager
Register/Login	Yes	Yes	Yes
Item overview	Yes	Yes	Yes
Editing Quantity of Item	No	Yes	Yes
CRUD Item	No	No	Yes
CRUD Class	No	No	Yes
Creation/Deletion of Inventory	No	No	Yes
Location Management	No	No	Yes
Browsing Items	Yes	Yes	Yes
Creating Restock list	No	No	Yes
Operation History	No	Yes	Yes

Table 1

Data dictionary

Table 2 - Data dictionary

Entity	Description	Attribute	Type	Description of attribute
User	User of the system.	user_id	int	Identifies the user.
		user_name	varchar(255)	First name of the user.
		user_lastname	varchar(255)	Last name of the user.
		user_email	varchar(255)	Email of the user.
		user_password	password_hash	Allows user to login.

		user_role	varchar(255)	What role user is.
Item	Represents a single product.	item_id	int	Identifies the item.
		item_article_number	int	The article number of the item usually issued by producer
		item_barcode	varchar(13)	EAN or barcode - allows for quick item identification by scanning.
		item_name	varchar(255)	Name of the item.
		item_price	decimal(10,2)	Price of the item in eur.
		item_picture	varchar(255)	Picture of the item.
		item_dimensions	varchar(255)	Dimensions of the item in cm.
		item_description	text	Description of the item.
Class	Category or class of similar items that are closely positioned in store.	class_id	int	Identifies the class.
		class_name	varchar(255)	Name of the class.
		class_position	varchar(255)	Position of the class in the store.
Inventory	Represents the stock of a specific item at a particular location.	inventory_id	int	Identifies inventory.
		inventory_item_quantity	int	Quantity of a given item in this inventory.
		inventory_alert_point	int	Minimal quantity before system alerts for restocking.
Location	Represents a physical storage place where inventory items are kept.	location_id	int	Identifies the Location.
		location_name	varchar(255)	Name of the Location.
		location_address	varchar(255)	Address of the Location.
		location_type	varchar(255)	Type of the location store/warehouse.
Restock List	Represents a list created by users (usually managers) that identifies items needing	restock_id	int	Identifies the restock List.
		restock_date	timestamp	The date that restock list was created
		restock_status	varchar(255)	Status of the Restocking

	replenishment based on current inventory levels.			List - issued/completed.
Item Operation Log	Records all significant actions performed to and items, such as sales, restocks, transfers, and reclassification.	operation_id	int	Identifies the operation
		operation_type	varchar(255)	Type of operation reclassified/moved/sold
		operation_date	timestamp	Date that operation was carried out.
		operation_quantity	int	How many items were moved.

Table 2

Entity relationship diagram

Figure 1 - Entity relationship diagram

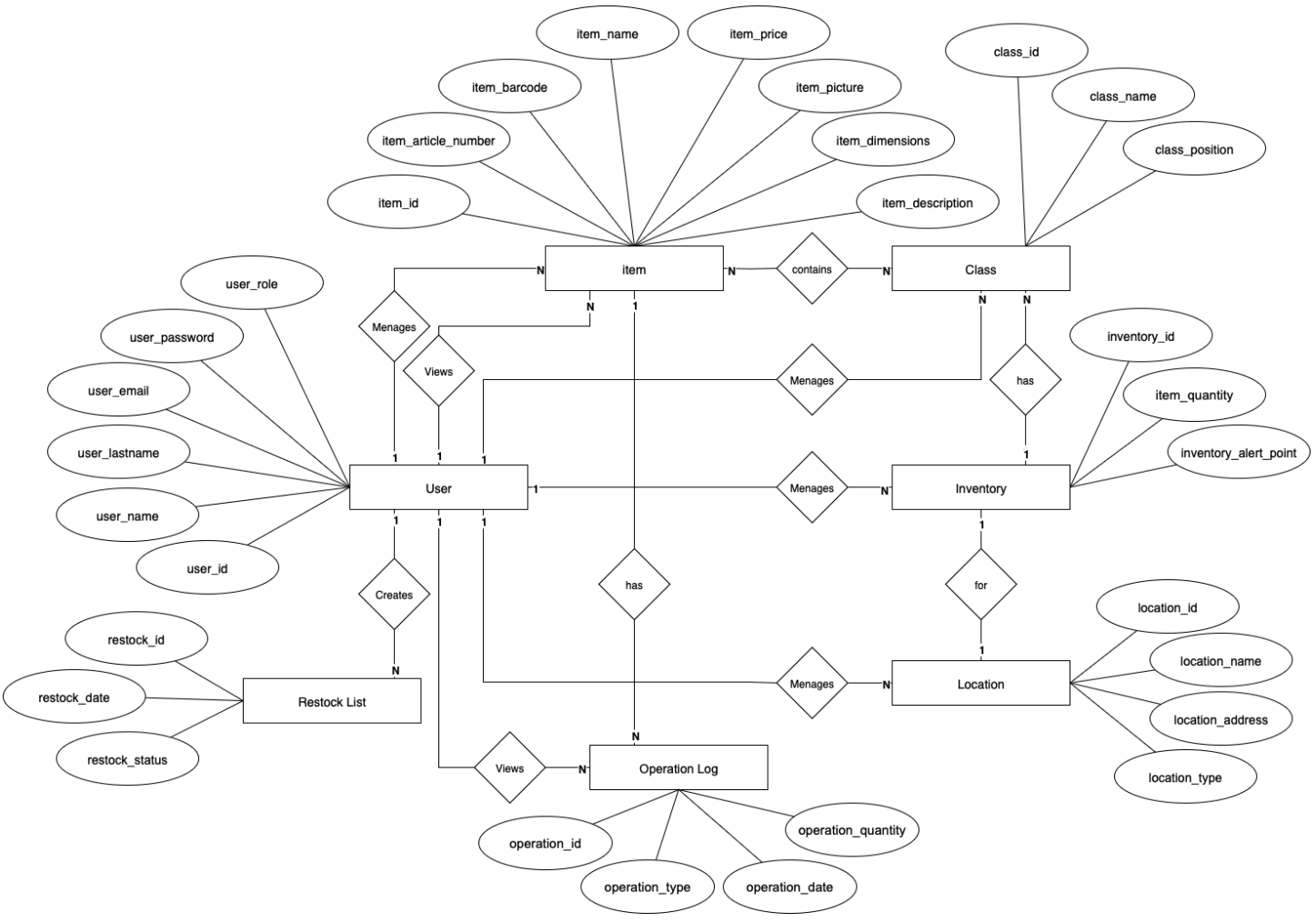

Figure 1

Relation diagram

Figure 2 - Relation diagram

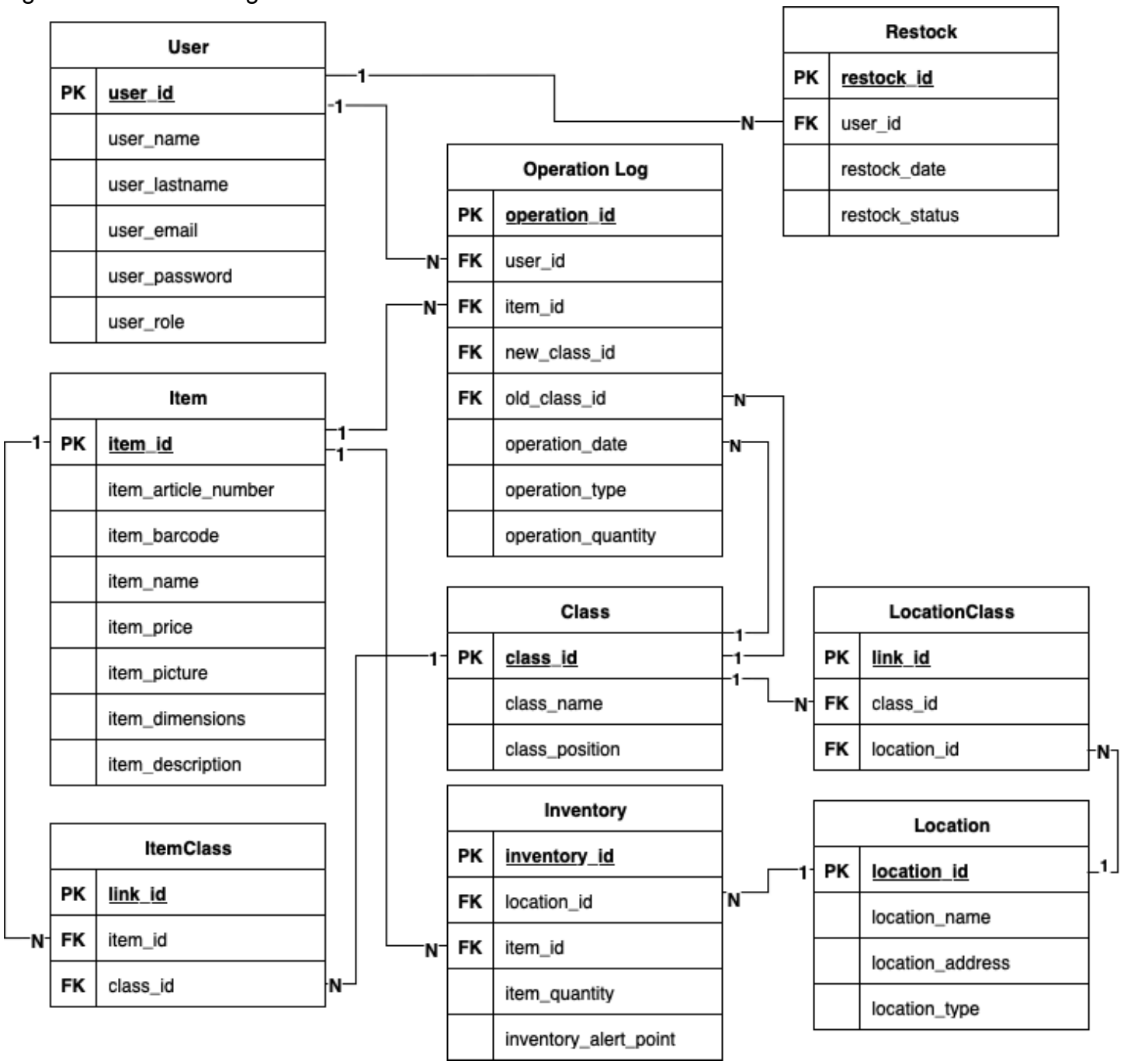

Figure 2

Data analysis and optimisation

The data model is in the third normal form. No further optimization is needed.

Physical Design Phase

Physical Data Model

Figure 3 - Physical Data Model

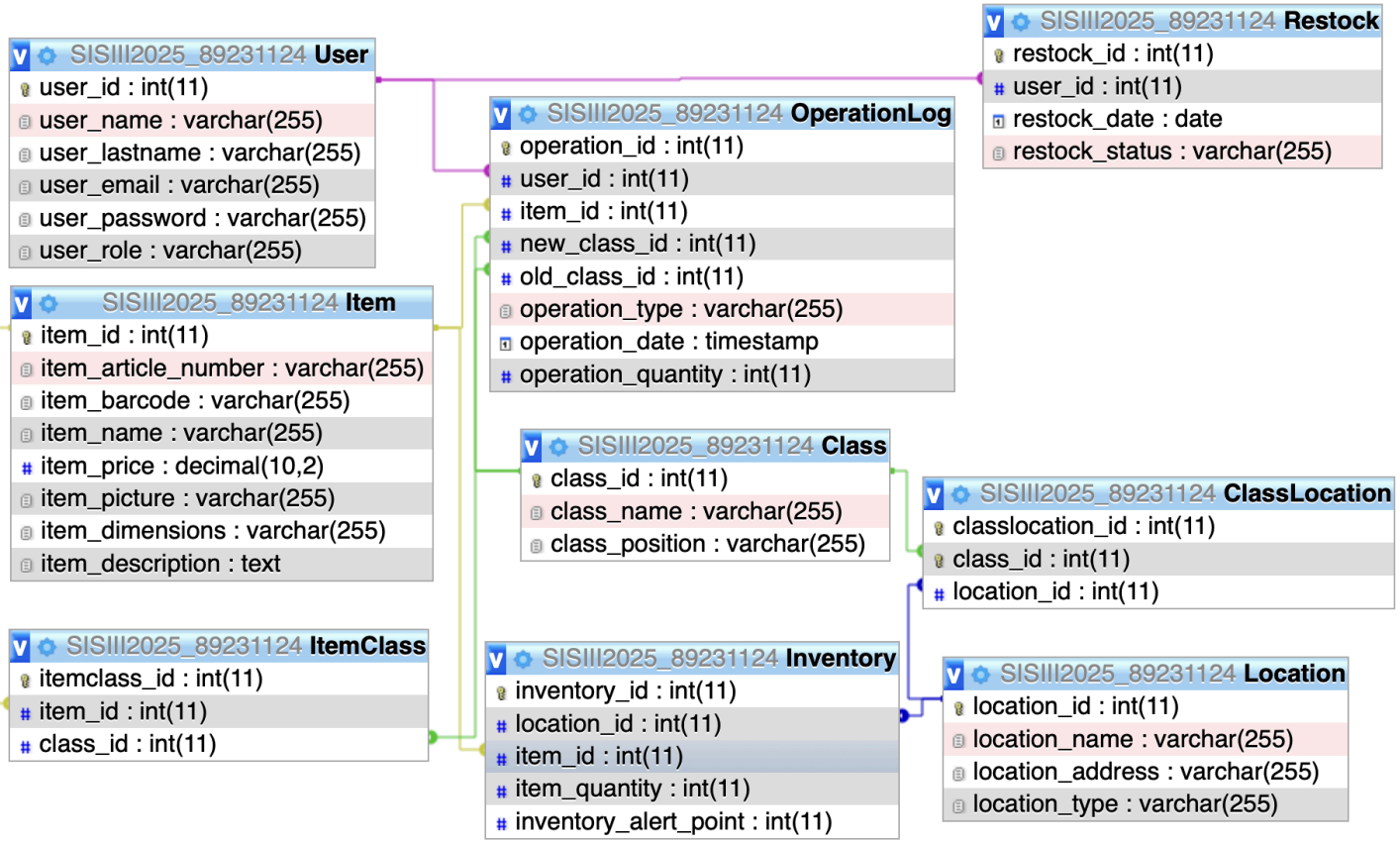

Figure 3

Object-oriented Analysis

UML Class Diagram

Figure 4 - UML Class Diagram

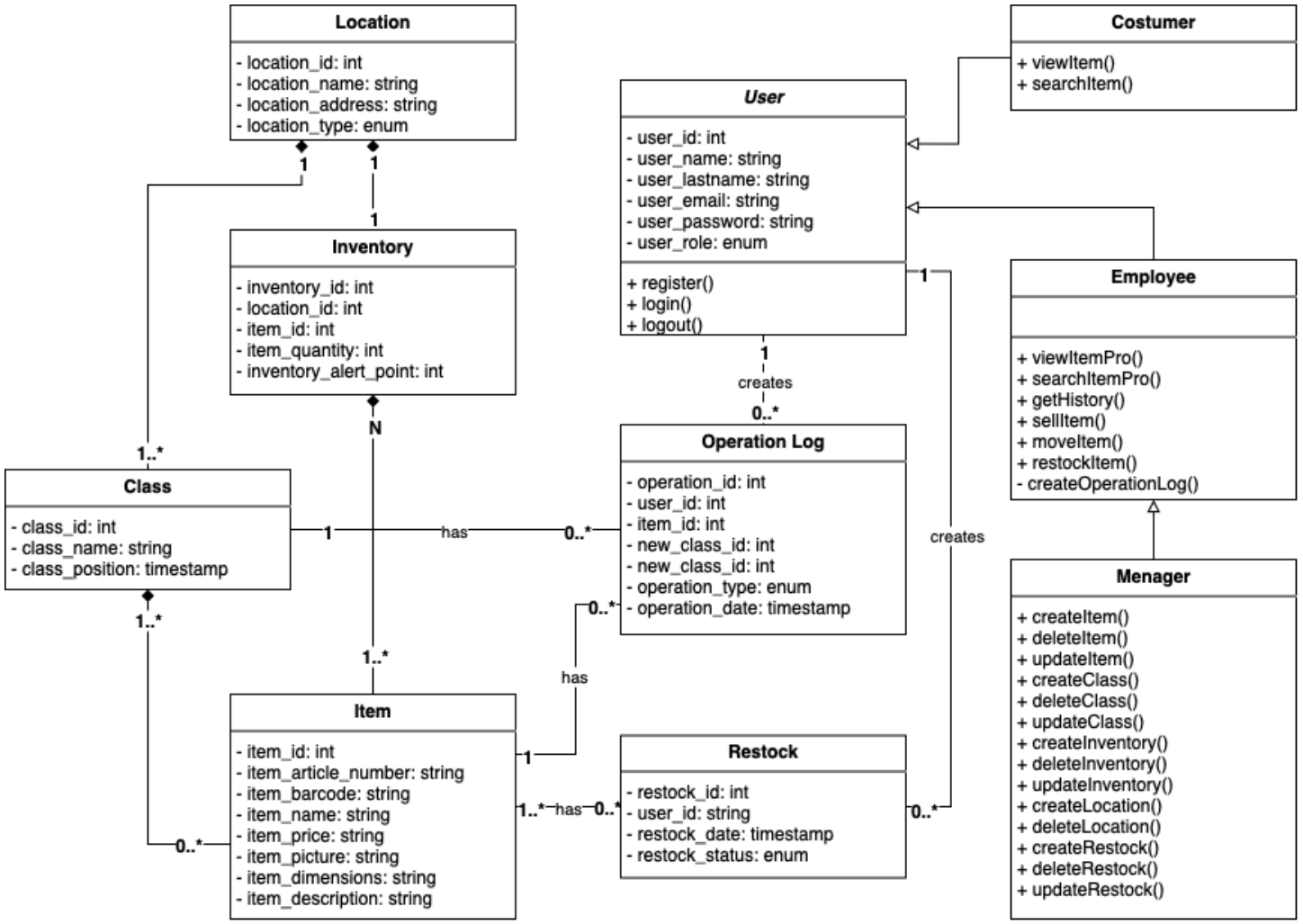

Figure 4

UML Sequence Diagram

Figure 5 - UML Diagram representing new Item creation

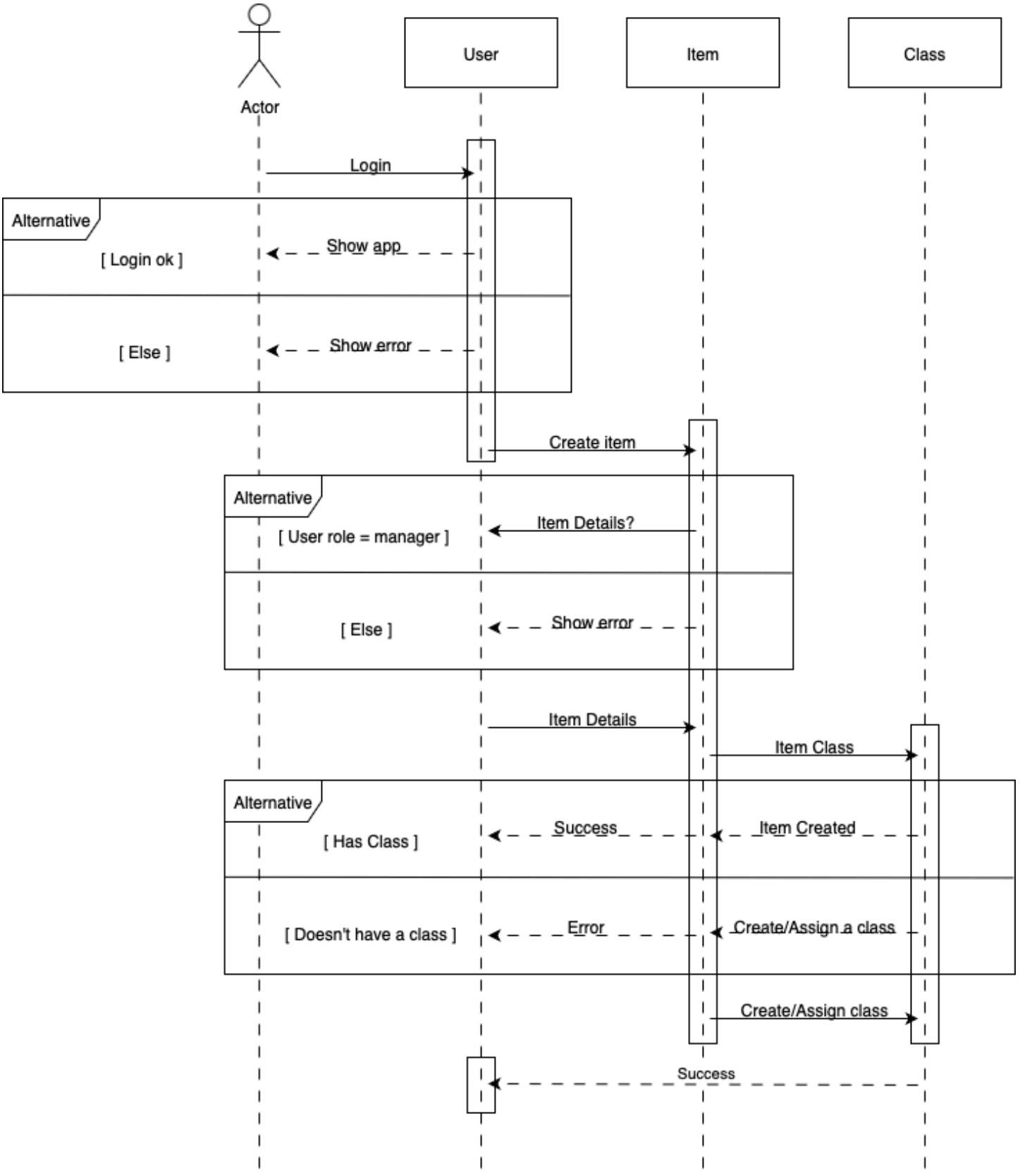

Figure 5

UML Use-case Diagram

Figure 6 - Use case diagram

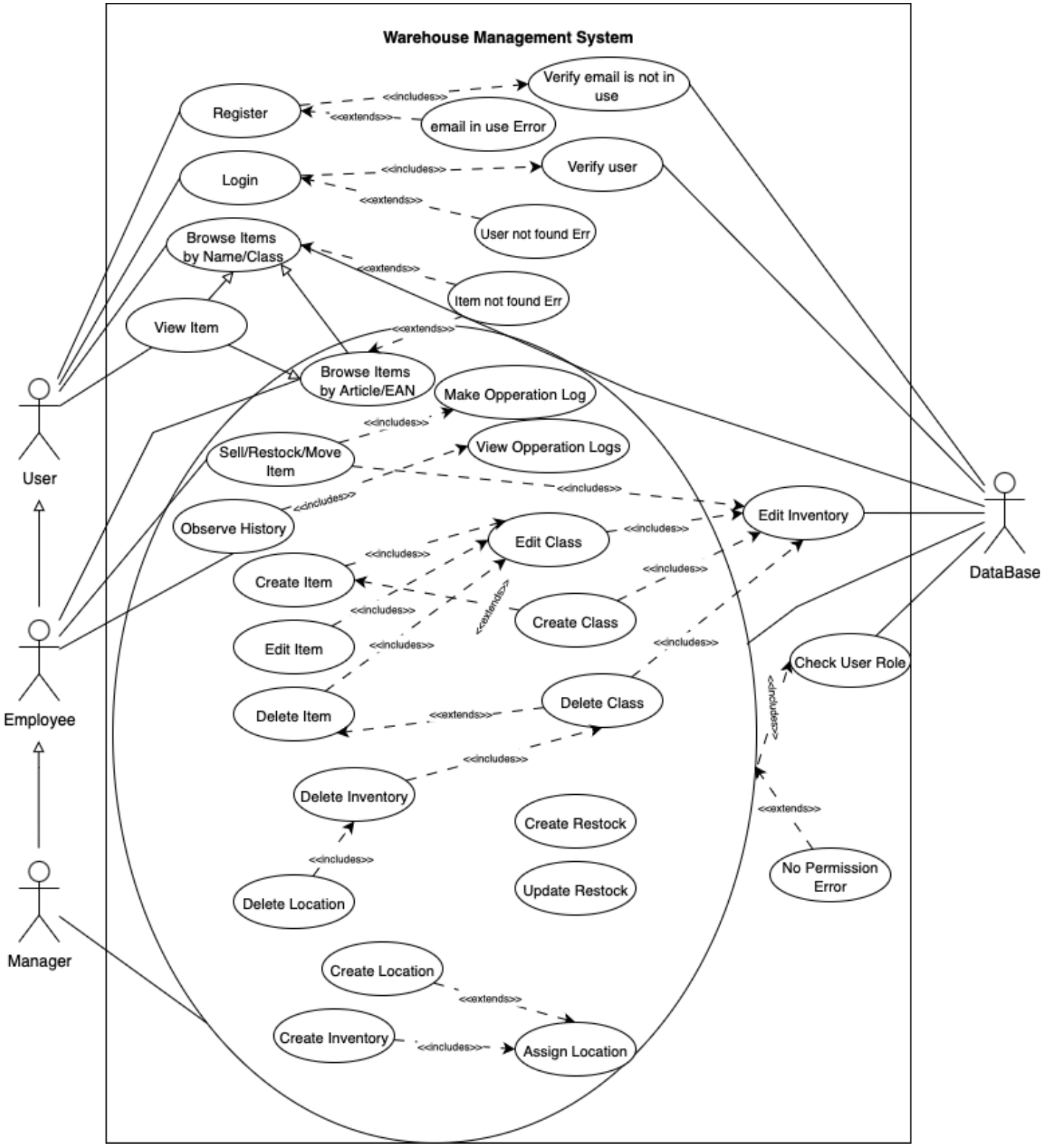

Figure 6