Warehouse Management System

Database Management System

MSCS 542L-256-24F

TechBoys

Marist College
School of Computer Science and Mathematics

Submitted To:
Dr. Reza Sadeghi

Sept 25, 2024

PROJECT REPORT OF WAREHOUSE MANAGEMENT SYSTEM

Team Name
TechBoys

Team Members
Sumanth Kumar Katapally SumanthKumar.Katapally1@marist.edu (Team Head)
Abhijeet Cherungottil Abhijeet.Cherungottil1@marist.edu (Team Member)
Sagar Shankaran Sagar.Shankaran1@marist.edu (Team Member)

TABLE OF CONTENTS

TABLE OF FIGURES

1. DESCRIPTION OF TEAM MEMBERS

Sumanth Kumar Katapally:

I attained my Bachelor's degree in Computer Science from Keshav Memorial Institute of Technology in 2022. I kickstarted my career as a Software Developer Intern at Virtusa, after which I took up a full-time role of a Software Developer at DBS TECH thereby gaining an overall experience of 2 years and 5 months. I am proficient in Java, Spring, Spring Boot, Angular, MongoDB and MySQL. I came to Marist College to pursue my MS in Computer Science concentrating in Artificial Intelligence.

Abhijeet Cherungottil:

I am a passionate computer science graduate currently pursuing an MS in Cloud Computing at Marist College, set to graduate in the 2026 batch. I also have 3 years of hands-on experience in iPhone app development, with expertise in Swift 5 and Swift UI. My current team is from India, which creates familiarity within the group and allows us to communicate easily with each other. I chose Sumanth because he mentioned having previous experience with GitHub and actively volunteered.

Sagar Shankaran:

I am a 2026 batch student of Masters in AI program at Marist College. I have experience in full stack development for 3 years. I have worked across various projects including MERN stack and Android Apps. I have chosen this team because all the members have a good understanding of each other and have profound knowledge of the project. Abhijeet and Sumanth have good relevant skills.

2. INTRODUCTION

In today's dynamic retail environment, efficient warehouse management is crucial for ensuring smooth operations, especially when multiple shopkeepers rely on a shared space to store and manage their inventory. The Warehouse Management System (WMS) project is designed to meet these needs by providing a secure, organized, and user-friendly platform that enhances the management of a warehouse where shopkeepers store and haul their cloth boxes under the oversight of an administrative user.

The WMS aims to streamline warehouse operations by offering a robust set of features tailored to both admin and shopkeeper needs. Admins will benefit from secure login capabilities, enabling them to manage access by adding or removing shopkeepers and adjusting credentials as needed. The system will also empower admins to oversee the entire inventory, allowing them to add, edit, or delete items with detailed information such as item type, storage time, ID, name, quantity, price, and storage location. Additionally, admins will have the authority to review and manage borrowing requests, ensuring that warehouse resources are allocated efficiently.

For shopkeepers, the WMS offers intuitive features designed to simplify their interactions with the warehouse. They will be able to search for items based on various attributes, save a list of favorite items, and request to borrow or purchase items for specific time periods. The system will also keep a detailed history of borrowed items, providing transparency and accountability.

User experience is at the forefront of the WMS design, with a welcoming interface, clear navigation, and tabular reports to present organized lists of requested items. Security is also a key focus, with options for password encryption and safeguards against potential errors like duplicate entries or exceeding borrowing limits. This WMS will revolutionize warehouse management, making it more efficient and secure for all users involved.

3. PROJECT OBJECTIVE

The objective of this project is to design and implement a robust Warehouse Management System (WMS) that provides an organized, secure, and user-friendly platform for managing a warehouse where multiple shopkeepers store and haul their cloth boxes, overseen by an admin user. The WMS will offer comprehensive functionalities, including:

1. **Admin Management:**
   - Secure login for the admin with the ability to change credentials.
   - Add and remove shopkeepers by creating or deleting their usernames and passwords.
   - Manage the warehouse inventory by adding, editing, and deleting items, including details such as item type, storage time, ID, name, quantities, price, and storage location.
   - Review and manage borrowing requests, with the ability to accept or reject them.
2. **Shopkeeper Features:**
   - Search for items in the warehouse based on various attributes like ID, name, and producer.
   - Save a list of favorite items for quick reference.
   - Request to borrow or purchase items for specific periods.
   - View the history of borrowed items.
3. **User Experience:**
   - Provide a welcoming page and a clear menu of functions on all pages.
   - Display reports in a tabular format, showing organized lists of requested items.
   - Include an exit function that thanks users for using the software.
   - Implement warnings for specific actions, such as duplicate item IDs, exceeding borrowing limits, or null search results.
4. **Security:**
   - Protect user information with optional encryption of passwords.

4. REVIEW

5.1 Zoho Inventory [1]:

- Helps with creating and tracking the inventory (stock flow) of Items and Item Groups.
- Integrates with popular e-commerce platforms and monitors your stock flow across multiple sales channels.
- Creates sales orders, raises invoices, gets paid instantly by integrating into a payment gateway, prints package slips and generates shipment labels.
- Helps in adding custom fields to your orders, invoices, bills, and payment receipts.

5.2 Fishbowl Inventory [2]:

- Provides visibility and control of inventory across all locations.
- Integrates with accounting systems like Intuit QuickBooks Online, Xero, and Reckon Accounts to provide real-time inventory and accounting reporting.
- Allows users to scan items by barcode, product number, UPC, or SKU to update inventory records, make accounting adjustments, and customize prices.
- Includes features like auto reorder points, in-depth reporting, and the ability to create purchase orders, sales orders, and pick tickets.
- Helps businesses track inventory spending, find ways to save money on purchasing, and avoid over-ordering.

5.3 Korber Warehouse Management Systems [3]:

- Integrates with other supply chain solutions, such as yard management and transportation, and ERP systems.
- Is flexible and adaptable, with configuration functionality that reduces the need for extensive customization and development.
- Is available as a software-as-a-service (SaaS), which includes provisioning, maintenance, and support.
- Has over 99% inventory accuracy and 80-85% less inventory loss.

5. MERITS

1. Add a notification to alert the user to order more items when the count reaches 'n'

- WMS proactively alerts users (admin or shopkeepers) when the inventory level of a particular item falls below a specified threshold (n), ensuring that the stock levels are replenished before they run out.
- In older or manual systems, stock monitoring is typically done manually. Users might only realize an item is out of stock when it's too late, leading to delays in fulfilling orders.

2. Tracks the location of items within the warehouse, including specific aisles, racks, and bins.

- WMS offers detailed location tracking for each item in the warehouse, down to the aisle, rack, and bin level, ensuring that every item can be quickly located.
- In older systems or warehouses, item locations may be tracked in a general sense, such as by section or area, or even on paper. This can lead to inefficiencies in locating items.

3. Ensuring old stock is cleared before new stock is added.

- Many traditional systems don’t enforce stock rotation, meaning newer stock can be placed over older stock, increasing the risk of product damage

4. Manages items that are out of stock and ensures they are prioritized when they become available.

- Out-of-stock items may not be tracked effectively in less advanced systems, which can result in delays in fulfilling backorders or not promptly notifying users when the items become available

5. Adding sustainability tracking.

- Less advanced WMSs rarely integrate sustainability tracking. Most inventory systems focus purely on operational aspects like stock levels, ignoring environmental or sustainability impacts.

6. GITHUB REPOSITORY

https://github.com/Sumanthkatapally/DBMS_PROJECT.git

7. ENTITY RELATIONSHIP MODEL (ER MODEL)

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design debug relational databases in the fields of software engineering, business information systems, education, and research.

Entities	Attributes
ADMIN	AdminID, Username, Password, Role, Email
SHOPKEEPER	ShopkeeperID, Username, Password, FullName, Email
PRODUCT	productid, productName, CategoryId, Size, PricePerUnit, QuantityInStock
WAREHOUSE	WarehouseID, LocationId, MaxCapacity, CurrentCapacity, AvailableSpace
BORROWREQUEST	RequestID ,RequestDate,BorrowStatus,ProductID ,ShopkeeperID
CATEGORY	CategoryID ,CategoryName,Description,Createddate,Seasonality
SUPPLIER	SupplierID ,SupplierName,ContactPerson,Email,SuppliedCloths
ORDER	orderId, orderDate, ShopkeeperID, Orderstatus, Amount
LOCATION	locationId, Address, PhoneNumber, Manager, OpeningHours, TotalCapacity
TRANSACTION	TransactionID, OrderID, TransactionDate, PaymentMethod, Amount, DiscountApplied, FinalAmount, Invoice
STOCKAUDIT	AuditID, WarehouseID, AuditDate, AuditorName, Discrepancies
SHIPMENT	ShipmentID, OrderID, ShipmentDate, ShipmentStatus, TrackingNumber

7.1 IMPLEMENTATION OF ER DIAGRAM

Entities and Attributes [4][5]:

Entity:Admin

Description: administrator responsible for managing warehouses and overseeing operations.

Attributes:

AdminID (PK): Unique identifier for each admin

Username: Login name for the admin

Password: Encrypted password for account security

Role: Specifies the admin's role (e.g., Manager, Supervisor)

Email: Contact email for the admin

Entity: SHOPKEEPER

Description: Represents store managers who place orders and manage inventory.

Attributes:

ShopkeeperID (PK): Unique identifier for each shopkeeper

Username: Login name for the shopkeeper

Password: Encrypted password for account security

FullName: Composite attribute (FirstName, LastName)

Email: Contact email for the shopkeeper

Entity: PRODUCT

Description:. Represents items available for sale in the clothing retail system

Attributes:

productID (PK): Unique identifier for each product

productName: Name of the product

CategoryId (FK): References the category the product belongs to

Size: Size of the product (e.g., Small, Medium, Large)

PricePerUnit: Cost of a single unit of the product

QuantityInStock: Current available quantity of the product

Entity: WAREHOUSE

Description: Represents storage facilities for products.

Attributes:

WarehouseID (PK): Unique identifier for each warehouse

LocationId (FK): References the location of the warehouse

MaxCapacity: Maximum storage capacity of the warehouse

CurrentCapacity: Current occupied capacity (derived)

AvailableSpace: Remaining storage space (derived)

Entity: BORROWREQUEST

Description: Represents requests from shopkeepers to borrow products.

Attributes:

RequestID (PK): Unique identifier for each request

RequestDate: Date when the request was made

BorrowStatus: Current status of the request (e.g., Pending, Approved, Rejected)

ProductID (FK): References the product being requested

ShopkeeperID (FK): References the shopkeeper making the request

Entity: CATEGORY

Description: Represents different classifications of products.

Attributes:

CategoryID (PK): Unique identifier for each category

CategoryName: Name of the category (e.g., Casual, Formal, Sportswear)

Description: Detailed description of the category

Createddate: Date when the category was created

Seasonality: Applicable season(s) for the category

Entity: SUPPLIER

Description: Represents companies or individuals who supply products.

Attributes:

SupplierID (PK): Unique identifier for each supplier

SupplierName: Name of the supplier company or individual

ContactPerson: Composite attribute (FirstName, LastName) of the primary contact

Email: Contact email for the supplier

SuppliedCloths: Multivalued attribute listing supplied product IDs

Entity: ORDER

Description: Represents customer purchases or stock orders.

Attributes:

orderId (PK): Unique identifier for each order

orderDate: Date when the order was placed

ShopkeeperID (FK): References the shopkeeper who placed the order

Orderstatus: Current status of the order (e.g., Processing, Shipped, Delivered)

Amount: Total cost of the order

Entity: LOCATION

Description: Represents physical locations of warehouses or shops.

Attributes:

locationId (PK): Unique identifier for each location

Address: Composite attribute (Street, City, State, ZipCode)

PhoneNumber: Contact number for the location

Manager: Composite attribute (FirstName, LastName) of the location manager

OpeningHours: Multivalued attribute listing opening hours for each day

TotalCapacity: Sum of all warehouse capacities at this location (derived)

Entity: TRANSACTION

Description: Represents financial transactions related to orders.

Attributes:

TransactionID (PK): Unique identifier for each transaction

OrderID (FK): References the associated order

TransactionDate: Date when the transaction occurred

PaymentMethod: Method used for payment (e.g., Cash, Credit Card)

TotalAmount: Total amount of the transaction

Invoice: BLOB

Entity: STOCKAUDIT

Description: Represents periodic checks of warehouse inventory accuracy.

Attributes:

AuditID (Partial Key): Identifier for the audit, unique within a warehouse

WarehouseID (PK, FK): References the warehouse being audited, part of the primary key

AuditDate: Date when the audit was conducted

AuditorName: Composite attribute (FirstName, LastName) of the auditor

Discrepancies: Multivalued attribute listing product IDs and quantity differences

Entity: SHIPMENT

Description: Represents the process of delivering orders to customers.

Attributes:

ShipmentID (PK): Unique identifier for each shipment

OrderID (FK): References the associated order

ShipmentDate: Date when the shipment was sent

ShipmentStatus: Current status of the shipment (e.g., Preparing, In Transit, Delivered)

TrackingNumber: Unique number for tracking the shipment

Multivalued Attributes:

- SUPPLIER: SuppliedCloths (list of supplied product IDs)
- LOCATION: OpeningHours (list of opening hours for each day)
- STOCKAUDIT: Discrepancies (list of product IDs and quantity differences)

Composite Attributes:

- SHOPKEEPER: FullName (FirstName, LastName)
- SUPPLIER: ContactPerson (FirstName, LastName)
- LOCATION: Address (Street, City, State, ZipCode)
- LOCATION: Manager (FirstName, LastName)
- STOCKAUDIT: AuditorName (FirstName, LastName)

Derived Attributes:

- WAREHOUSE: CurrentCapacity (derived from sum of stored products)
- WAREHOUSE: AvailableSpace (derived from MaxCapacity - CurrentCapacity)
- LOCATION: TotalCapacity (derived from sum of all warehouse capacities at this location)
- TRANSACTION: AMOUNT(derived from quantity, price per unit from order table)

Weak Entity:

- STOCKAUDIT (its existence depends on WAREHOUSE)

Strong Entities:

ADMIN, SHOPKEEPER, PRODUCT, WAREHOUSE, BORROWREQUEST, CATEGORY, SUPPLIER, ORDER, LOCATION, TRANSACTION, SHIPMENT

Total Participations:

- ORDER to TRANSACTION (every order must have at least one transaction)
- ORDER to SHIPMENT (every order must have at least one shipment)
- WAREHOUSE to STOCKAUDIT (every warehouse must undergo stock audits)

Partial Participations:

- PRODUCT to BORROWREQUEST (not all products may be involved in borrow requests)
- SHOPKEEPER to BORROWREQUEST (not all shopkeepers may make borrow requests)

Cardinality Ratios:

a. One-to-One (1:1):

- WAREHOUSE to STOCKAUDIT (one warehouse undergoes one stock audit)

b. One-to-Many (1:N):

- ADMIN to WAREHOUSE (one admin can manage many warehouse)
- SHOPKEEPER to ORDER (one shopkeeper can place many orders)
- CATEGORY to PRODUCT (one category can have many products)
- SUPPLIER to PRODUCT (one supplier can supply many products)
- ORDER to TRANSACTION (one order can have many transactions)
- LOCATION to WAREHOUSE (one location can house many warehouses)
- ORDER to SHIPMENT (one order can have many shipments)
- WAREHOUSE to SHIPMENT (one warehouse can process many shipments)

c. **Many-to-Many (M:N):**

- PRODUCT to WAREHOUSE (many products can be stored in many warehouses)

The entity-relationship conceptual model of a database can be represented graphically, using a diagram. These diagrams are simple and intuitive, and their basic components are:

• Rectangles - specifying entity types

• Ellipses - specifying attributes

• Rhomboids - specifying relationship types

• Lines - connections between entities and attributes, or entities and relationship types.

In the picture, you can see a more detailed overview of the notation:

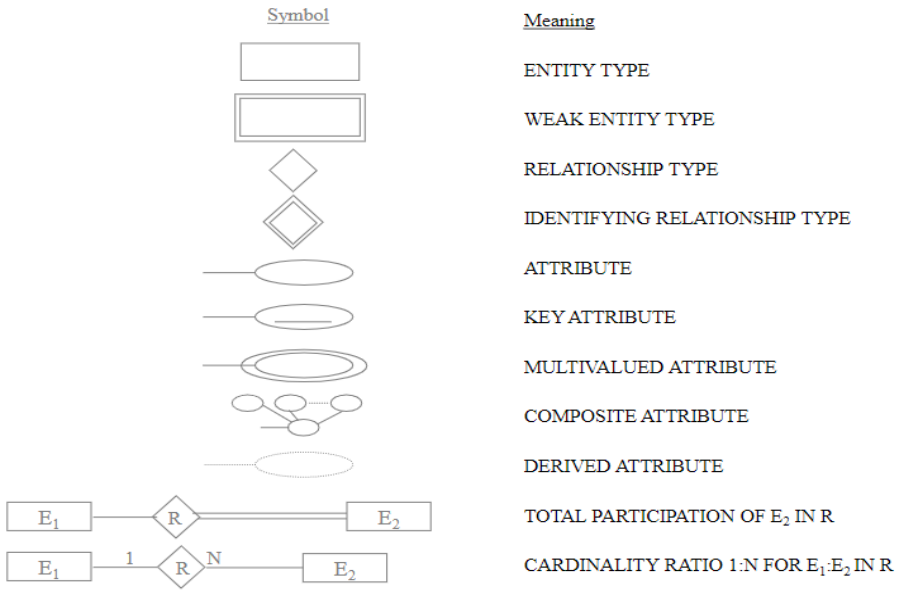

Figure 7. 1 Relationship of Entity Relationship Model

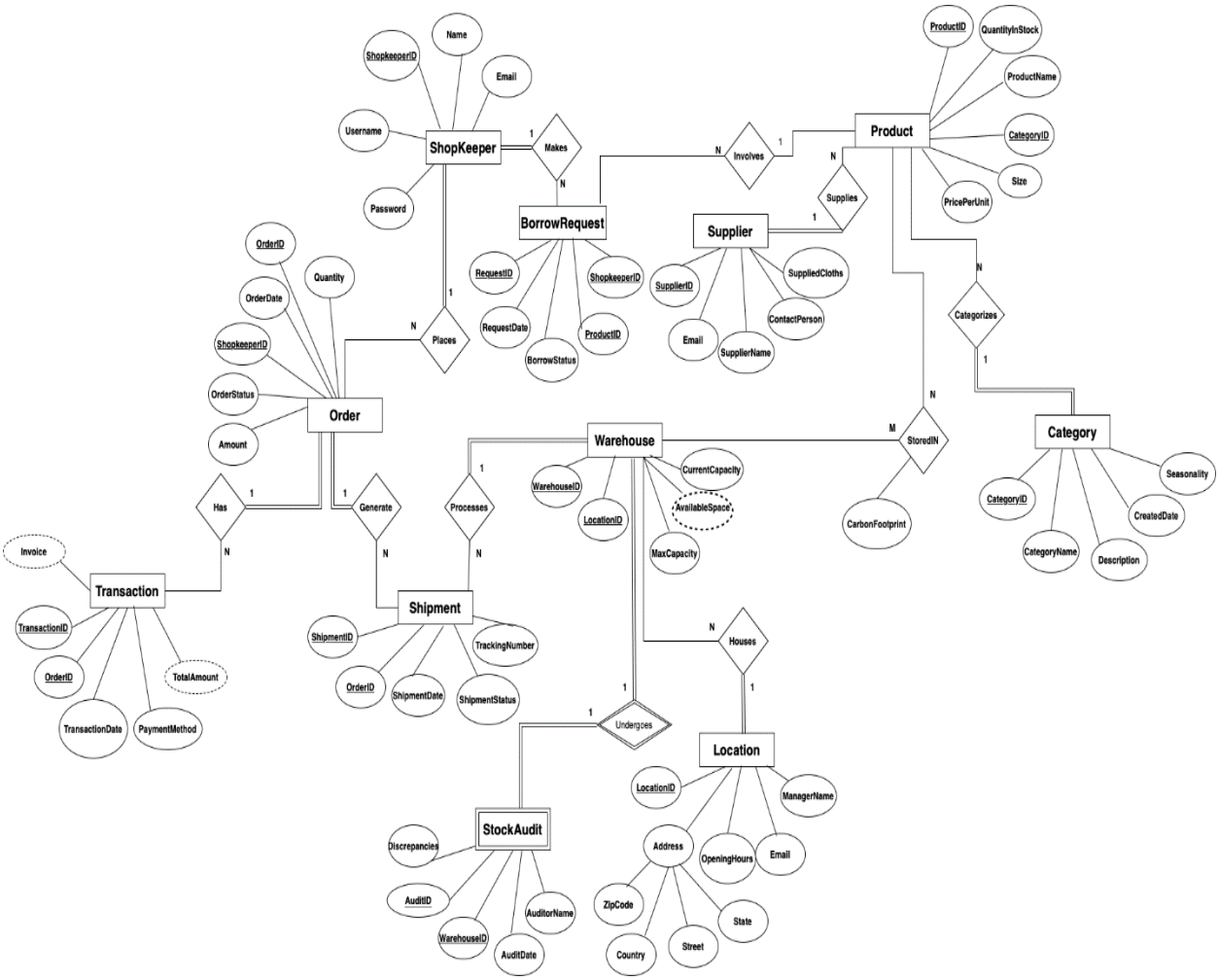

Figure 7.2 External ER SHOPKEEPER POV [6]

As an external ER user, as a customer, I might interact with the Product and Category entities when browsing items. The Order and Shipment entities would be relevant when making a purchase and tracking my delivery.

The Warehouse and Product entities suggest that I might be able to check item availability across different locations, which could be useful and the presence of the Shipment entity with attributes like TrackingNumber and ShipmentStatus indicates I could likely track my orders easily.

The Product entity seems to contain detailed information (like Size, PricePerUnit), which would be helpful for making purchasing decisions.

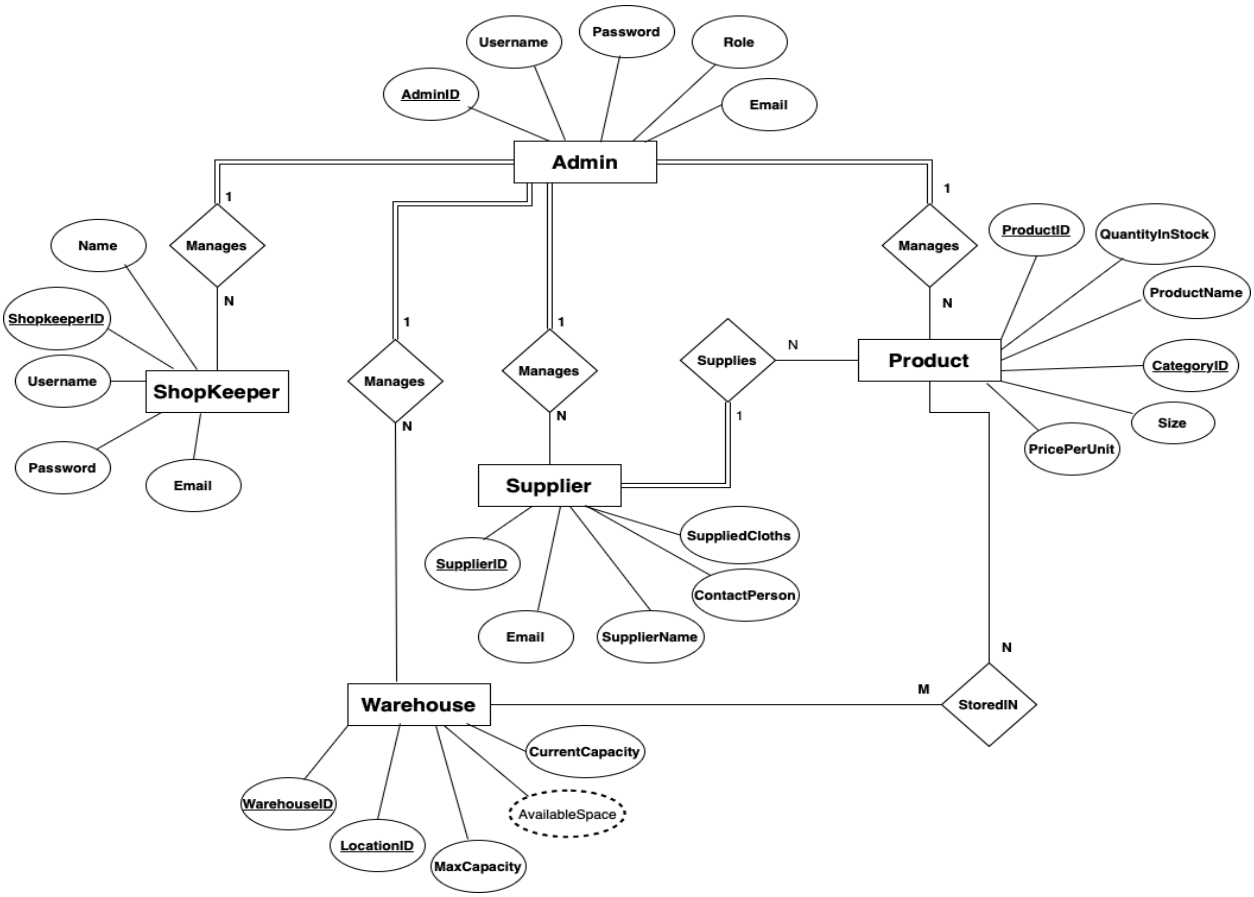

Figure 7. 3 External ER ADMIN POV [6]

The diagram places the Admin entity at the center, indicating a centralized management approach. The Admin has oversight of ShopKeepers, Suppliers, and Products, implying broad control over the entire system. This centralized structure allows for efficient top-down management and decision-making processes.

Product management appears to be a core focus of the system. Products are linked to Suppliers who provide them, Warehouses where they are stored, and are managed by both Admins and ShopKeepers. This comprehensive approach to product tracking enables detailed inventory control and supply chain visibility.

Warehouse management is given particular attention, with attributes like CurrentCapacity, MaxCapacity, and AvailableSpace. The inclusion of AvailableSpace as a potentially derived attribute (shown in a dashed oval) suggests dynamic capacity management, crucial for efficient inventory control and storage optimization.

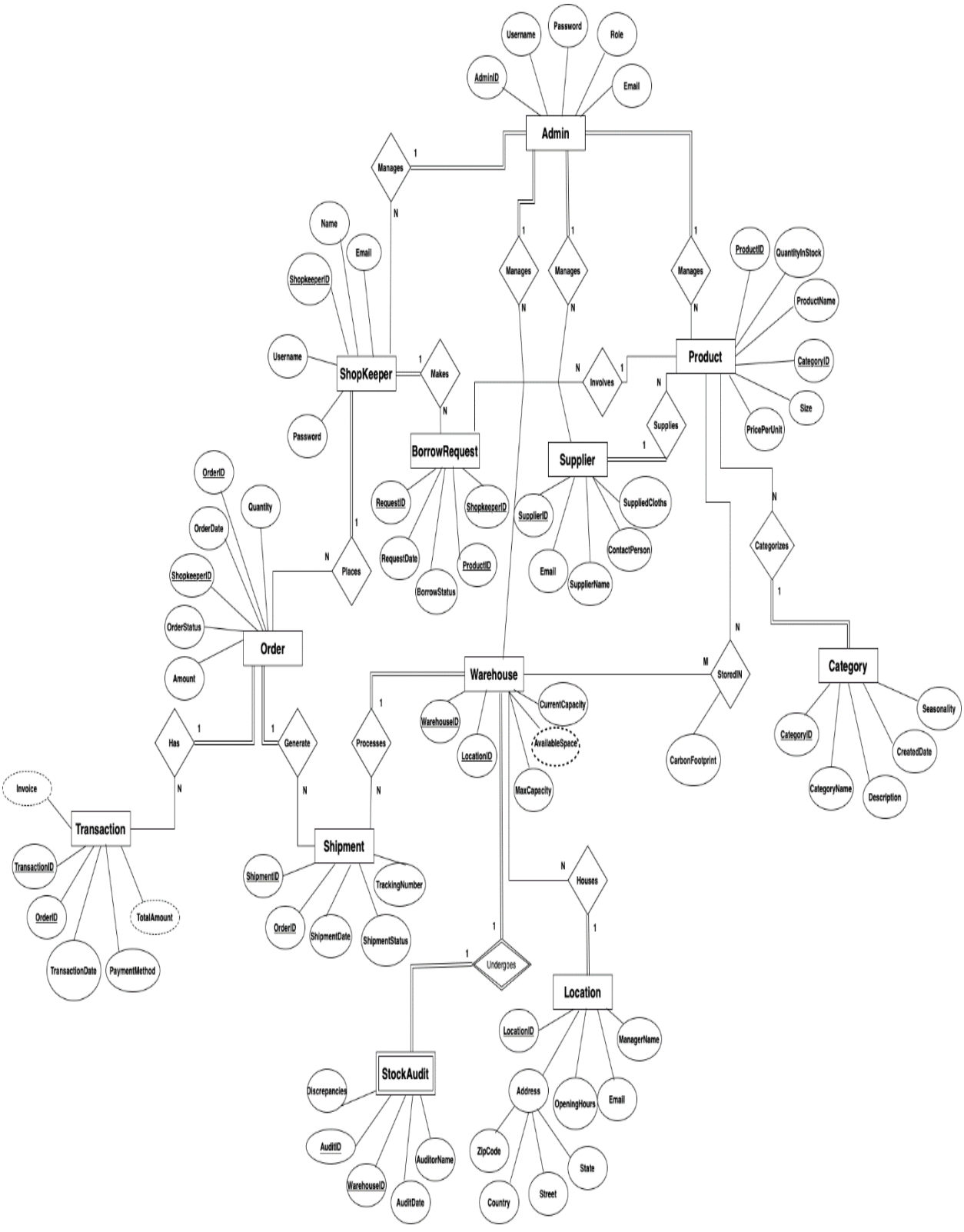

Figure 7. 4 Entity Relationship Diagram [6]

7.2 ENHANCED ENTITY RELATIONSHIP MODEL (EER MODEL)

EER Diagram, also abbreviated as Enhanced Entity-relationship diagram, helps us create and maintain detailed databases through high-level models and tools. In addition, they are developed on the basic ER diagrams and are its extended version.

We have used MySQL Workbench to create the EER diagram. Enhanced Entity Relationship (EER) diagrams are an essential part of the modeling interface in MySQL Workbench. EER diagrams provide a visual representation of the relationships among the tables in our model.

EER diagrams extend ER diagrams by introducing the concepts of **superclasses** and **subclasses** to depict inheritance between entities. Specialization and generalization allow for categorizing entities into more specific subtypes or abstracting them into a generalized form. **Aggregation** represents a higher-level abstraction where relationships themselves are treated as entities.

EER Diagrams basically help in creating and maintaining excellent databases with the help of smart and efficient techniques. It is a visual representation of the plan or the overall outlook of the database you intend to create.

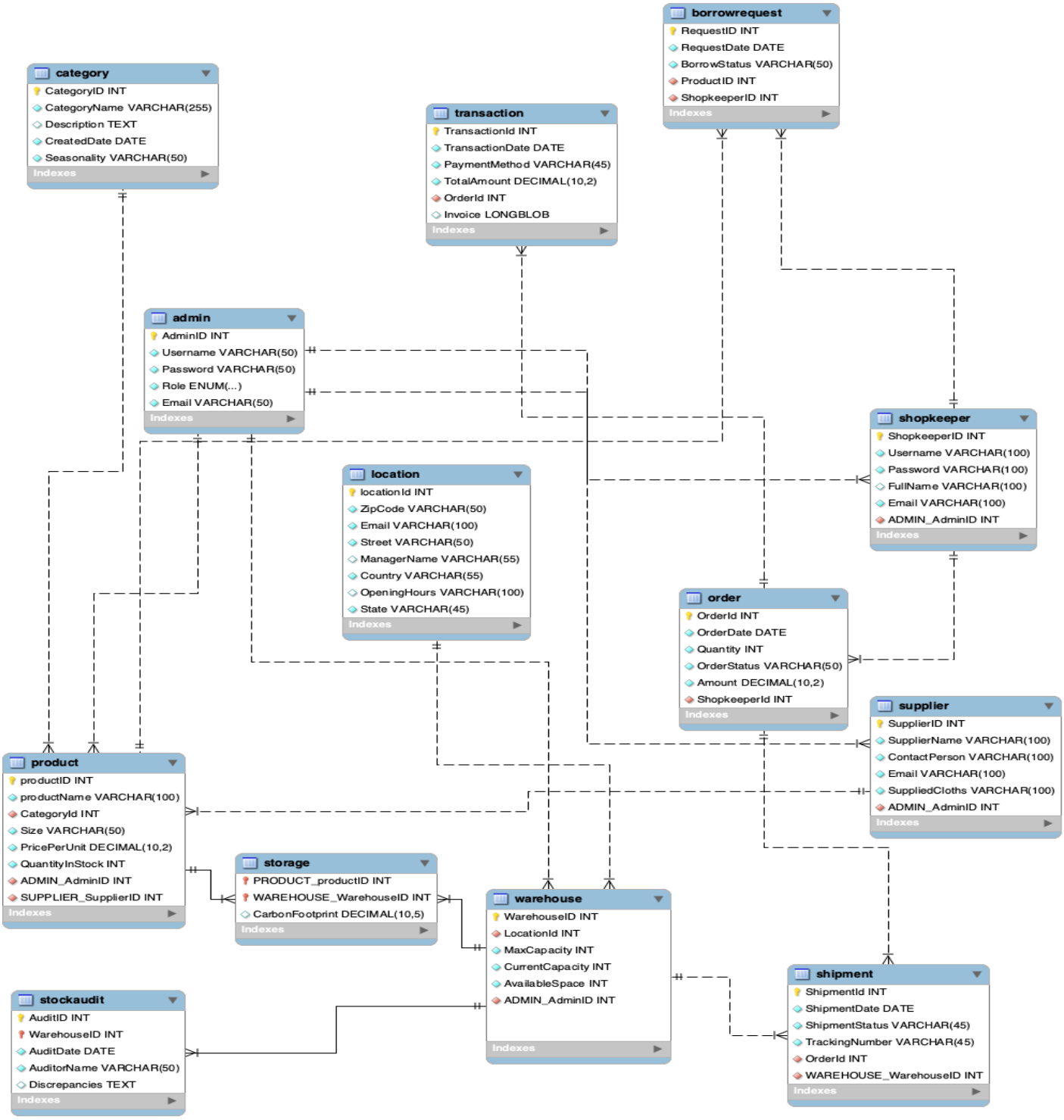

Figure 8. 1 Enhanced Entity Relationship

8. DATABASE DEVELOPMENT

Database development plays a crucial role in managing and leveraging data efficiently, securely, and at scale. It is used to create various applications and is essential for businesses seeking to show the importance of data for decision making and innovation.

The below schema has been created from the Enhanced Entity Diagram, that involves all the tables, meets all the primary and foreign key constraints.

8.1 DESCRIPTION:

Table Name	Description
Admin	**Usage:** Represents administrators responsible for managing warehouses, products, shopkeepers, and suppliers. **Data attributes:** 1. AdminID (Integer, Primary Key, Not Null) 2. Username (Varchar (50),Not Null) 3. Password (Varchar (50),Not Null) 4. Email (Varchar (50),Not Null) 5. Role (Enum (Manager, Supervisor),Not Null) **Constraints:** 1. AdminID must be unique and cannot be null. PRIMARY KEY 2. Email must follow a valid email format. 3. Role is an Enum (Manager, Supervisor)

	Relationships : 1. Admin manages multiple Warehouses 2. Admin manages multiple Products, 3. Admin manages multiple Shopkeepers 4. Admin manages multiple Suppliers.
Product	**Usage:** Represents items that are stored in warehouses, supplied by suppliers, and categorized into various product categories. **Data Attributes:** 1. ProductID (Integer, Primary Key, Not Null) 2. ProductName (Varchar (50), Not Null) 3. Size (Varchar(50), Not Null) 4. PricePerunit (Decimal(10,2),, Not Null) 5. QuantityInStock (Integer, Not Null) 6. CategoryID (Integer, Foreign Key, Not Null) 7. SupplierID (Integer, Foreign Key, Not Null) 8. AdminID(Integer, Foreign Key, Not Null) **Constraints:** 1. ProductID must be unique and cannot be null. PRIMARY KEY

	2. CategoryID references the Category table. Foreign Key, 3. SupplierID references the Supplier table. Foreign Key, 4. AdminID references the admin table. Foreign Key **Relationships:** 1. Product can be involved in multiple BorrowRequests . 2. Product Belongs to one Category. 3. Product Supplied by one Supplier. 4. Each product has an admin

Warehouse	**Usage:** Represents storage locations for products and handles stock audits and shipments. **Data Attributes:** 1. WarehouseID (Integer, Primary Key, Not Null) 2. LocationID (Integer, Foreign Key, Not Null) 3. MaxCapacity (Integer,Not Null) 4. AvailableSpace(Integer,Not Null) 5. CurrentCapacity (Integer,Not Null) 6. AdminID (references the admin table. Foreign Key, Not Null) **Constraints :** 1. WarehouseID must be unique and cannot be null.

	2. LocationID references the Location table. **Relationships:** 1. Warehouse Stores multiple Storage. 2. Warehouse Processes multiple Shipments. 3. Warehouse Undergoes one StockAudit .
Supplier	**Usage:** Represents entities that supply products to the warehouses. **Data Attributes:** 1. SupplierID (Integer, Primary Key, Not Null) 2. SupplierName (Varchar (100), Not Null) 3. ContactPerson (Varchar (100), Not Null) 4. Email (Varchar (100), Not Null) 5. SuppliedCloths (Varchar (100), Not Null) 6. AdminID (references the admin table. Foreign Key, Not Null) **Constraints:** 1. SupplierID must be unique and cannot be null. 2. AdminID references the admin table. Foreign Key

	Relationships: 1. Supplier supplies multiple Products . 2. Each Supplier has a admin
StockAudit	**Usage:** Represents audits performed on the stock of a warehouse. **Data Attributes:** 1. AuditID (Integer, Primary Key, Not Null) 2. WarehouseID (Integer, Foreign Key, Not Null) 3. AuditDate (Date, Not Null) 4. AuditorName (Varchar(50),Not Null) 5. Discrepancies TEXT **Constraints:** 1. StockAuditID must be unique and cannot be null. 2. WarehouseID references the Warehouse table.

	Relationships: 1. StockAuditID Performed on one Warehouse.

Location	**Usage:** Represents physical locations of warehouses or shops. **Data Attributes:** 2. LocationID (Integer, Primary Key, Not Null) 3. Zipcode(Varchar(55),Not Null) 4. Email (Varchar (100),Not Null) 5. Street (Varchar (55),Not Null) 6. State (Varchar (55),Not Null) 7. Country (Varchar (55)Not Null) 8. OpeningHours(Varchar(55),) 9. ManagerName(Varchar(55),) **Constraints:** 1. LocationID must be unique and cannot be null. **Relationships:**

	1. Location Contains multiple Warehouses .
Storage	**Usage :** Represents the many-to-many relationship between Products and Warehouses, indicating which products are stored in which warehouses. **Data Attributes**: 1. ProductID (Integer, Foreign Key, Not Null) 2. WarehouseID (Integer, Foreign Key, Not Null) 3. CarbonfootPrint (Double(10,5)) **Constraints**: 1. ProductID references the Product table and cannot be null. 2. WarehouseID references the Warehouse table and cannot be null. 3. Together, ProductID and WarehouseID form a composite primary key.

	Relationships: 1. Storage References for one Product. 2. Storage References one Warehouse.
Category	**Usage:** Represents the category of products that are stored in warehouses and supplied by suppliers. **Data Attributes:** 1. CategoryID (Integer, Primary Key, Not Null) 2. CategoryName (Varchar(255), Not Null) 3. Description (Text, Nullable) 4. CreatedDate (Date, Not Null) 5. Seasonality (Varchar(50), Nullable) **Constraints:** 1. CategoryID must be unique and cannot be null. PRIMARY KEY. **Relationships:** 1. CategoryID is referenced by the Product table. 2. Each Category can have multiple products.

BorrowRequest	**Usage:** Represents the request made by shopkeepers to borrow products from warehouses. **Data Attributes:** 1. RequestID (Integer, Primary Key, Not Null) 2. RequestDate (Date, Not Null) 3. BorrowStatus (Varchar(50), Not Null) 4. ProductID (Integer, Foreign Key, Not Null) 5. ShopkeeperID (Integer, Foreign Key, Not Null) **Constraints:** 1. RequestID must be unique and cannot be null. PRIMARY KEY. 2. ProductID references the Product table. 3. ShopkeeperID references the Shopkeeper table. **Relationships:** 1. Each BorrowRequest is associated with one Product. 2. Each BorrowRequest is associated with one Shopkeeper.
Transaction	**Usage:** Represents financial transactions related to orders placed by shopkeepers. **Data Attributes:** 1. TransactionId (Integer, Primary Key, Not Null) 2. TransactionDate (Date, Nullable) 3. PaymentMethod (Varchar(45), Nullable) 4. TotalAmount (Decimal(10,2), Nullable) 5. OrderId (Integer, Foreign Key, Nullable) 6. Invoice (Long BLOB, Nullable) **Constraints:** 1. TransactionId must be unique and cannot be null. PRIMARY KEY. 2. OrderId references the Order table. **Relationships:** 1. Each Transaction is associated with one Order.

Order	**Usage**: Represents orders placed by shopkeepers for products from warehouses. **Data Attributes:** 1. OrderId (Integer, Primary Key, Not Null) 2. OrderDate (Date, Not Null) 3. Quantity (Integer, Not Null) 4. OrderStatus (Varchar(50), Not Null) 5. Amount (Decimal(10,2), Not Null) 6. ShopkeeperId (Integer, Foreign Key, Not Null) **Constraints:** 1. OrderId must be unique and cannot be null. PRIMARY KEY. 2. ShopkeeperId references the Shopkeeper table. **Relationships:** 1. Each Order is associated with one Shopkeeper. 2. Each Order can have multiple Shipments.

Shipment	**Usage:** Represents shipments for the orders placed by shopkeepers, tracking the shipment process. **Data Attributes:** 1. ShipmentId (Integer, Primary Key, Not Null) 2. ShipmentDate (Date, Not Null) 3. ShipmentStatus (Varchar(45), Not Null) 4. TrackingNumber (Varchar(45), Not Null) 5. OrderId (Integer, Foreign Key, Not Null) 6. WAREHOUSE_WarehouseID (Integer, Foreign Key, Not Null) **Constraints:** 1. ShipmentId must be unique and cannot be null. PRIMARY KEY. 2. OrderId references the Order table. 3. WAREHOUSE_WarehouseID references the Warehouse table. **Relationships:** 1. Each Shipment is linked to one Order. 2. Each Shipment is processed by one Warehouse.
Shopkeeper	**Usage:** Represents shopkeepers who place orders and borrow products from warehouses. **Data Attributes:** 1. ShopkeeperID (Integer, Primary Key, Not Null) 2. Username (Varchar(100), Not Null) 3. Password (Varchar(100), Not Null) 4. FullName (Varchar(100)) 5. Email (Varchar(100), Not Null) 6. ADMIN_AdminID (Integer, Foreign Key, Not Null) **Constraints:** 1. ShopkeeperID must be unique and cannot be null. PRIMARY KEY. 2. ADMIN_AdminID references the Admin table. **Relationships:** 1. Each Shopkeeper is managed by one Admin.

	2. Each Shopkeeper can place multiple Orders and submit multiple BorrowRequests.

8.2 CREATE STATEMENTS

1. CREATE Warehouse_Management_System_DBMS_Project DATABASE:

```
CREATE DATABASE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`;
USE `Warehouse_Management_System_DBMS_Project` ;
```

2. CREATE ADMIN TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`ADMIN` (
  `AdminID` INT NOT NULL AUTO_INCREMENT,
  `Username` VARCHAR(50) NOT NULL,
  `Password` VARCHAR(50) NOT NULL,
  `Role` ENUM('Manager', 'Supervisor') NOT NULL,
  `Email` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`AdminID`));
```

3. CREATE SHOPKEEPER TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`SHOPKEEPER` (
  `ShopkeeperID` INT NOT NULL AUTO_INCREMENT,
```

```
  `Username` VARCHAR(100) NOT NULL,
  `Password` VARCHAR(100) NOT NULL,
  `FullName` VARCHAR(100) NULL DEFAULT NULL,
  `Email` VARCHAR(100) NOT NULL,
  `ADMIN_AdminID` INT NOT NULL,
  PRIMARY KEY (`ShopkeeperID`),
  CONSTRAINT `fk_SHOPKEEPER_ADMIN1`
    FOREIGN KEY (`ADMIN_AdminID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`ADMIN` (`AdminID`));
```

4. CREATE CATEGORY TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`CATEGORY` (
  `CategoryID` INT NOT NULL AUTO_INCREMENT,
  `CategoryName` VARCHAR(255) NOT NULL,
  `Description` TEXT NULL ,
  `CreatedDate` DATE NOT NULL,
  `Seasonality` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`CategoryID`));
```

5. CREATE SUPPLIER TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`SUPPLIER` (
  `SupplierID` INT NOT NULL AUTO_INCREMENT,
  `SupplierName` VARCHAR(100) NOT NULL,
  `ContactPerson` VARCHAR(100) NOT NULL,
```

```
  `Email` VARCHAR(100) NOT NULL,
  `SuppliedCloths` VARCHAR(100) NOT NULL,
  `ADMIN_AdminID` INT NOT NULL,
  PRIMARY KEY (`SupplierID`),
  CONSTRAINT `fk_SUPPLIER_ADMIN1`
    FOREIGN KEY (`ADMIN_AdminID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`ADMIN` (`AdminID`));
```

6. CREATE LOCATION TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`LOCATION` (
  `locationId` INT NOT NULL AUTO_INCREMENT,
  `ZipCode` VARCHAR(50) NOT NULL,
  `Email` VARCHAR(100) NOT NULL,
  `Street` VARCHAR(50) NOT NULL,
  `ManagerName` VARCHAR(55) NULL,
  `Country` VARCHAR(55) NOT NULL,
  `OpeningHours` VARCHAR(100) NULL,
  `State` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`locationId`));
```

7. CREATE PRODUCT TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`PRODUCT` (
  `productID` INT NOT NULL AUTO_INCREMENT,
  `productName` VARCHAR(100) NOT NULL,
```

```
  `CategoryId` INT NOT NULL,
  `Size` VARCHAR(50) NOT NULL,
  `PricePerUnit` DECIMAL(10,2) NOT NULL,
  `QuantityInStock` INT NOT NULL,
  `ADMIN_AdminID` INT NOT NULL,
  `SUPPLIER_SupplierID` INT NOT NULL,
  PRIMARY KEY (`productID`),
  CONSTRAINT ``
    FOREIGN KEY (`CategoryId`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`CATEGORY`
(`CategoryID`),
  CONSTRAINT `fk_PRODUCT_ADMIN1`
    FOREIGN KEY (`ADMIN_AdminID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`ADMIN` (`AdminID`),
  CONSTRAINT `fk_PRODUCT_SUPPLIER1`
    FOREIGN KEY (`SUPPLIER_SupplierID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`SUPPLIER`
(`SupplierID`));
```

8. CREATE WAREHOUSE TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`WAREHOUSE` (
  `WarehouseID` INT NOT NULL AUTO_INCREMENT,
  `LocationId` INT NOT NULL,
  `MaxCapacity` INT NOT NULL,
```

```
  `CurrentCapacity` INT NOT NULL,
  `AvailableSpace` INT NOT NULL,
  `ADMIN_AdminID` INT NOT NULL,
  PRIMARY KEY (`WarehouseID`),
  CONSTRAINT ``
    FOREIGN KEY (`LocationId`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`LOCATION`
(`locationId`),
  CONSTRAINT `fk_WAREHOUSE_ADMIN1`
    FOREIGN KEY (`ADMIN_AdminID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`ADMIN` (`AdminID`));
```

9. CREATE BORROWREQUEST TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`BORROWREQUEST` (
  `RequestID` INT NOT NULL AUTO_INCREMENT,
  `RequestDate` DATE NOT NULL,
  `BorrowStatus` VARCHAR(50) NOT NULL,
  `ProductID` INT NOT NULL,
  `ShopkeeperID` INT NOT NULL,
  PRIMARY KEY (`RequestID`),
  CONSTRAINT ``
    FOREIGN KEY (`ProductID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`PRODUCT`
(`productID`),
```

```
  CONSTRAINT ``
    FOREIGN KEY (`ShopkeeperID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`SHOPKEEPER`
(`ShopkeeperID`));
```

10. CREATE STOCKAUDIT TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`STOCKAUDIT` (
  `AuditID` INT NOT NULL AUTO_INCREMENT,
  `WarehouseID` INT NOT NULL,
  `AuditDate` DATE NOT NULL,
  `AuditorName` VARCHAR(50) NOT NULL,
  `Discrepancies` TEXT NULL,
  PRIMARY KEY (`AuditID`, `WarehouseID`),
  CONSTRAINT `WarehouseID`
    FOREIGN KEY (`WarehouseID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`WAREHOUSE`
(`WarehouseID`));
```

11. CREATE ORDER TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`ORDER` (
  `OrderId` INT NOT NULL AUTO_INCREMENT,
  `OrderDate` DATE NOT NULL,
  `Quantity` INT NOT NULL,
  `OrderStatus` VARCHAR(50) NOT NULL,
```

```
  `Amount` DECIMAL(10,2) NOT NULL,
  `ShopkeeperId` INT NOT NULL,
  PRIMARY KEY (`OrderId`),
  CONSTRAINT `ShopkeeperId`
    FOREIGN KEY (`ShopkeeperId`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`SHOPKEEPER`
(`ShopkeeperID`));
```

12. CREATE SHIPMENT TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`SHIPMENT` (
  `ShipmentId` INT NOT NULL AUTO_INCREMENT,
  `ShipmentDate` DATE NOT NULL,
  `ShipmentStatus` VARCHAR(45) NOT NULL,
  `TrackingNumber` VARCHAR(45) NOT NULL,
  `OrderId` INT NOT NULL,
  `WAREHOUSE_WarehouseID` INT NOT NULL,
  PRIMARY KEY (`ShipmentId`),
  CONSTRAINT `OrderId`
    FOREIGN KEY (`OrderId`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`ORDER` (`OrderId`),
  CONSTRAINT `fk_Shipment_WAREHOUSE1`
    FOREIGN KEY (`WAREHOUSE_WarehouseID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`WAREHOUSE`
(`WarehouseID`));
```

.

13. CREATE STORAGE TABLE:

```
CREATE TABLE IF NOT EXISTS
`Warehouse_Management_System_DBMS_Project`.`STORAGE` (
  `PRODUCT_productID` INT NOT NULL AUTO_INCREMENT,
  `WAREHOUSE_WarehouseID` INT NOT NULL,
  `CarbonFootprint` DECIMAL(10,5) NULL,
  PRIMARY KEY (`PRODUCT_productID`, `WAREHOUSE_WarehouseID`),
  CONSTRAINT `fk_PRODUCT_has_WAREHOUSE_PRODUCT1`
    FOREIGN KEY (`PRODUCT_productID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`PRODUCT`
(`productID`),
  CONSTRAINT `fk_PRODUCT_has_WAREHOUSE_WAREHOUSE1`
    FOREIGN KEY (`WAREHOUSE_WarehouseID`)
    REFERENCES
`Warehouse_Management_System_DBMS_Project`.`WAREHOUSE`
(`WarehouseID`));
```

9. REFERENCES

[1] Zoho Inventory

[2] Fishbowl Inventory

[3] Korber Warehouse Management Systems

[4] Warehouse management system - ER Diagram

[5] Design ER Diagrams for Supply Chain Management

[6] Draw.io