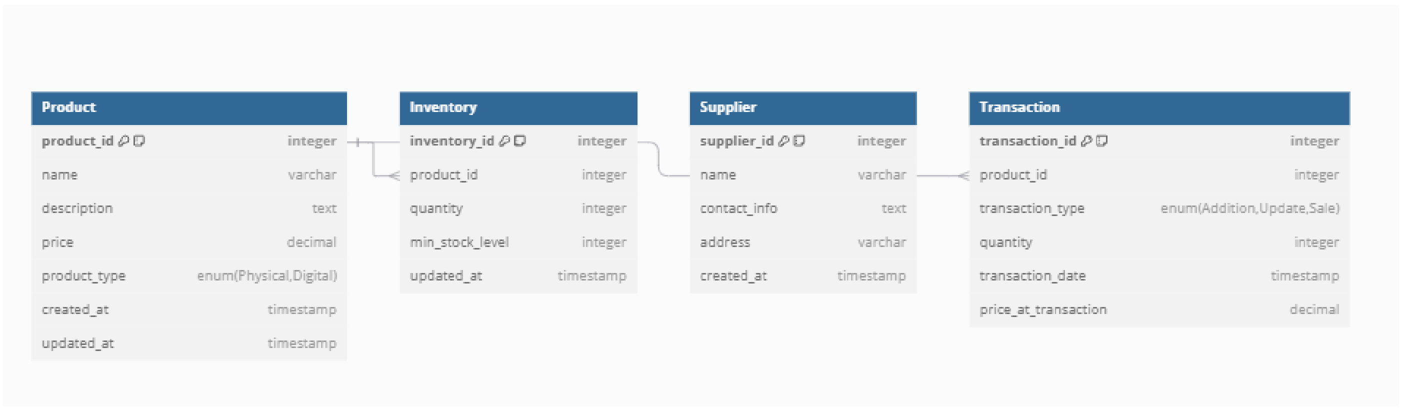
CONSOLE-BASED RESTAURANT RESERVATION SYSTEM

Abstract:

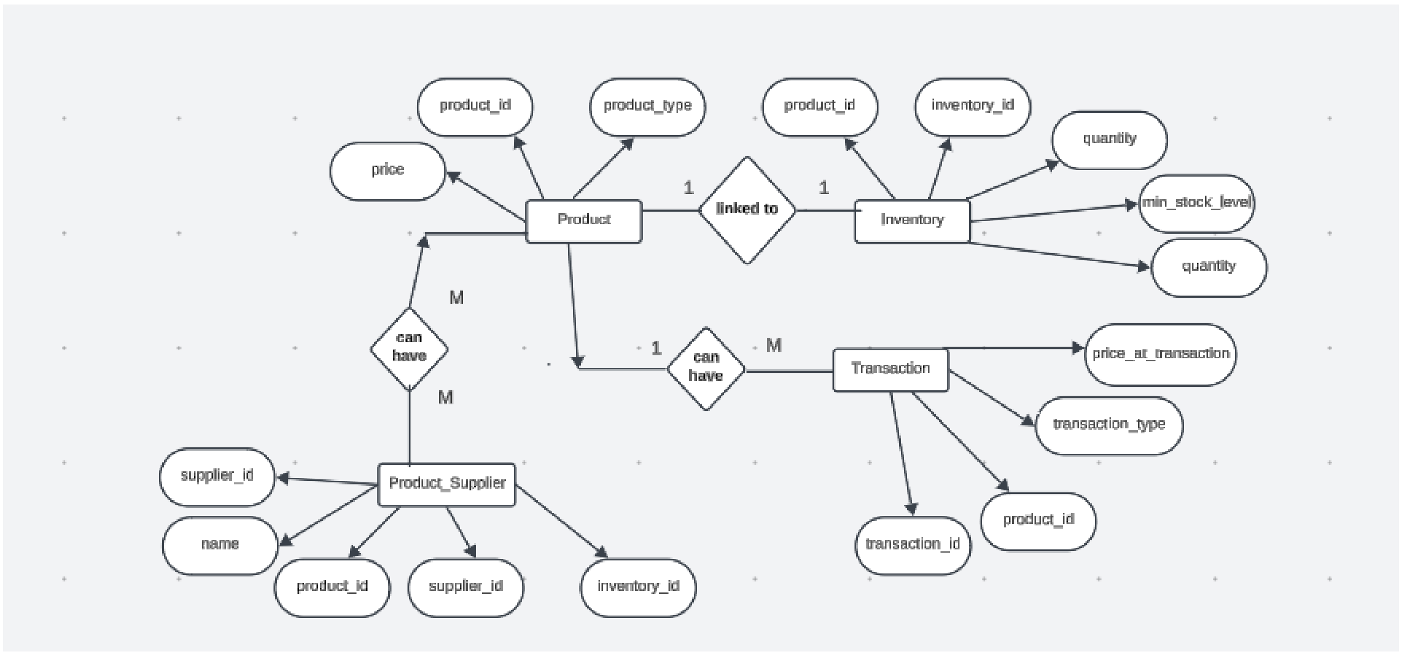
The Retail Management System is a Java-based application designed to streamline product and inventory management within a retail environment. The system leverages the Data Access Object (DAO) pattern to abstract database operations and provide a clear separation between business logic and data access. It enables efficient management of physical and digital products, supplier data, inventory updates, and transaction processing. Through the use of DAO, the application offers a robust framework for adding, updating, and tracking products and inventory in real-time. The system supports essential operations such as stock addition, product updates, and sales transactions, with seamless database interaction through JDBC.

Key features include automatic inventory updates triggered by transactions, detailed product reports, and transaction logging. Additionally, a stored procedure is used to handle new product additions and inventory updates, ensuring consistency across the database. The application also supports the generation of reports to visualize product stock levels, supplier information, and transaction history. This project demonstrates how the DAO pattern improves code modularity, maintainability, and scalability while ensuring that the retail system operates efficiently, handles large volumes of data, and can easily be extended for future needs.

Schema Diagram:



Entity Relationship Diagram:



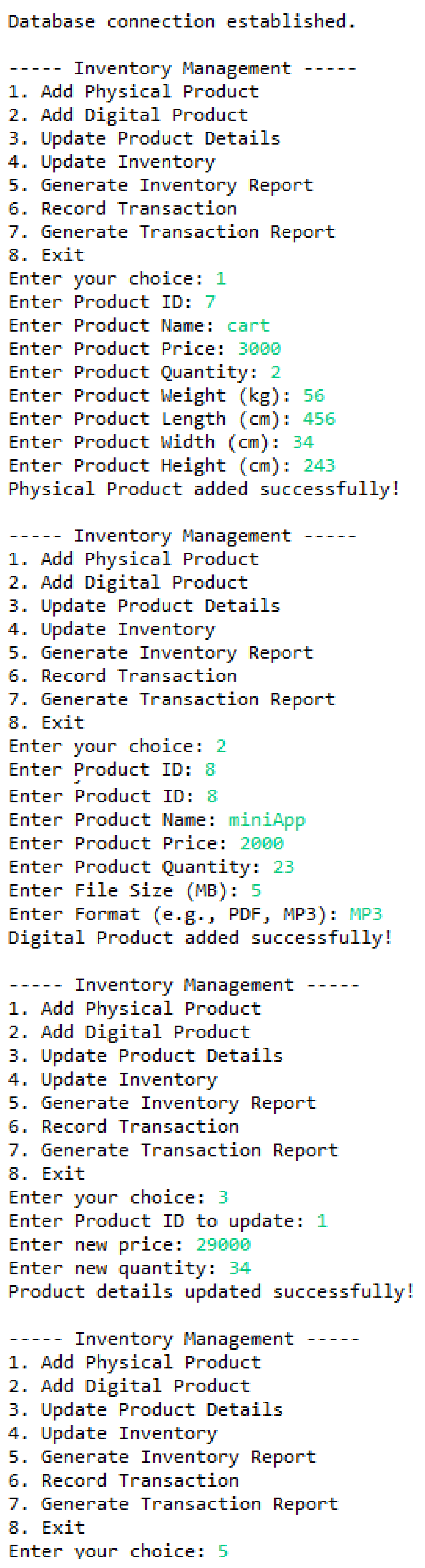
Project Structure:

```
RestaurantReservationSystem/
│
├── src/                    # Source code folder
│   ├── config/              # Configuration and utilities
│   │   ├── DatabaseConnection.java      # Database connection setup
│   │   └── InventoryLogger.java         # Logger for file handling (logs inventory updates)
│   │
│   ├── dao/                 # Data Access Objects for database operations
│   │   ├── ProductDAO.java              # CRUD operations for Product
│   │   ├── InventoryDAO.java            # CRUD operations for Inventory
│   │   └── TransactionDAO.java          # CRUD operations for Transaction logs
│   │
│   ├── exceptions/          # Custom exception handling
│   │   ├── ProductNotFoundException.java
│   │   └── InvalidStockQuantityException.java
│   │
│   ├── interfaces/          # Interface definitions
│   │   └── InventoryManageable.java     # Interface for inventory management
│   │
│   ├── models/              # Core Java classes representing entities
│   │   ├── Product.java                 # Base Product class
│   │   ├── PhysicalProduct.java         # Physical product subclass
│   │   ├── DigitalProduct.java          # Digital product subclass
│   │   ├── Inventory.java               # Inventory management class
│   │   ├── Supplier.java                # Supplier entity
│   │   └── Transaction.java             # Transaction entity
│   │
│   ├── services/            # Service layer for business logic
│   │   ├── InventoryService.java        # Inventory services (add, update, delete, report generation)
│   │   └── ProductService.java          # Product services (price update, product information)
│   │
│   ├── threads/             # Multithreading for concurrent operations
│   │   ├── ProductAdderThread.java      # Thread for adding products
│   │   └── InventoryUpdaterThread.java  # Thread for updating inventory concurrently
│   │
│   ├── utils/               # Utility classes for additional functionality
│   │   └── InputValidator.java          # Input validation utilities
│   │
│   ├── Main.java            # Main class for running the application
│   │
│   └── tests/               # Test cases and testing framework
│       └── RetailManagementSystemTest.java # Main test class
│
├── db/                     # SQL scripts and DB setup files
│   └── restaurant_reservation_system.sql        # SQL file for inserting test data
│
├── inventory_log.txt       # Log file to track inventory changes (for file handling)
│
├── README.md               # Documentation for the project setup and usage
└── .gitignore              # Files and directories to ignore in version control
```

Code Files:

GITHUB LINK : RESTAURANT RESERVATION SYSTEM

Sample Output(Screenshots):



```
Database connection established.

----- Inventory Management -----
1. Add Physical Product
2. Add Digital Product
3. Update Product Details
4. Update Inventory
5. Generate Inventory Report
6. Record Transaction
7. Generate Transaction Report
8. Exit
Enter your choice: 1
Enter Product ID: 7
Enter Product Name: cart
Enter Product Price: 3000
Enter Product Quantity: 2
Enter Product Weight (kg): 56
Enter Product Length (cm): 456
Enter Product Width (cm): 34
Enter Product Height (cm): 243
Physical Product added successfully!

----- Inventory Management -----
1. Add Physical Product
2. Add Digital Product
3. Update Product Details
4. Update Inventory
5. Generate Inventory Report
6. Record Transaction
7. Generate Transaction Report
8. Exit
Enter your choice: 2
Enter Product ID: 8
Enter Product ID: 8
Enter Product Name: miniApp
Enter Product Price: 2000
Enter Product Quantity: 23
Enter File Size (MB): 5
Enter Format (e.g., PDF, MP3): MP3
Digital Product added successfully!

----- Inventory Management -----
1. Add Physical Product
2. Add Digital Product
3. Update Product Details
4. Update Inventory
5. Generate Inventory Report
6. Record Transaction
7. Generate Transaction Report
8. Exit
Enter your choice: 3
Enter Product ID to update: 1
Enter new price: 29000
Enter new quantity: 34
Product details updated successfully!

----- Inventory Management -----
1. Add Physical Product
2. Add Digital Product
3. Update Product Details
4. Update Inventory
5. Generate Inventory Report
6. Record Transaction
7. Generate Transaction Report
8. Exit
Enter your choice: 5
```