BIRZEIT UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
COMP333

Project Description Report

Bookstore management system

Prepared by:

Amjad Adi 1230800

Waseem Yahya 1231052

Supervised by:

Dr. Bassem Sayrafi

18 JAN 2025

Table of Contents

Table Of Figures

1 Project Description:

The Bookstore Management System is a platform designed to manage the transactions on books, stationery items to the general customers or to the university and school students. Customers can search, purchase, rent, or exchange items conveniently online or on store. By centralizing inventory and providing real-time stock updates

Customers can interact with the platform by browsing products, completing transactions, and submitting reviews and ratings, while administrators can manage the customers, products, warehouses, staff monitor stock levels, and track all transactions, including rental and return activities along with system statistics.

2 Project Scope:

The scope of the project includes user management with role-based access, admins could fully control product management (adding, updating, deleting, and categorizing items), inventory tracking, transaction management (sales, rentals, and exchanges) and reporting. While customers can view products, make purchases, exchanges or rentals, and submit reviews. The system will be developed using “JAVA” for the backend logic, “MySQL” for the database system and “JAVA-FX-Scene Builder” for the user interface.

3 Client Information:

We have chosen the Ramallah Public Library as our client, as they can provide valuable assistance in defining the project requirements and supplying the necessary data for development. The proposed system aims to extend the library's existing services to include resources for university and school students, enabling access to academic materials alongside general collections. The Ramallah Public Library will serve as our primary reference and partner institution throughout the design and implementation phases, ensuring that the system meets real world needs and supports the library's mission of promoting knowledge and learning.

Contact Information [1]:

- Name: Al-Bireh Public Library (Ramallah & Al-Bireh)
- Address: Municipality Building, behind Al-Ein Mosque, Al-Ein, Ramallah & Al-Bireh
- Phone: 02-2404549
- Email: librarians2palestine@gmail.com
- Website: https://librarianswithpalestine.org/featured-projects-members/public-libraries/el-bireh-public-library-ramallah/

4 Entities description:

The Customer table stores information about people who use the system, such as the customer ID, full name, email, password, profession, birth date (with age), registration date, activation and expiration dates, budget in dollars, and whether the account is disabled.

The Activated Customer table represents customers who are currently active in the system, and it keeps activation-related information such as the activation date linked to the same customer record.

The Inactivated Customer table represents customers who are currently inactive, and it keeps deactivation-related information such as the last activation time linked to the same customer record.

The Staff table stores details about bookstore employees who manage orders, including staff ID, name, email, password, salary, birth date (with age), hire date, and current status.

The Product table holds general information about all items available in the store, including product ID, name, description, category, company, and price, while stock is tracked through warehouses rather than stored directly here.

The Book table is a specialized part of the Product data that stores book details such as ISBN, title, author information, publication year, genre, language, edition, and number of pages.

The Stationery Item table is also a specialized part of the Product data and stores stationery details such as color, material, dimensions, and production year.

The Review table stores customer feedback on products, including the review ID, rating, comment, and review date, and each review is linked to one customer and one product.

The Orders table records customer orders with details like order ID, order date, channel, and order information, and it is linked to the customer who made it, the staff member who manages it, and the payment method and plan used.

The Order Items relation (between Orders and Product) represents the products included in each order, and it stores important order-time details such as quantity and the price at the time of purchase or rental.

The Payment Method table lists the available ways to pay (such as cash or card), including the payment method ID, method name, and a short description used when processing orders.

The Payment Plan table stores installment plan options, including the plan ID, payment period in months, months before legal trial, and a description, and an order may use one plan if needed.

The Rental table represents rental orders and stores rental-specific information such as rental start date, rental end date, return date, and rental status.

The Sale table represents purchase orders and stores sale-related status information used to track the completion or condition of the sale.

The Warehouse table stores information about storage locations, including warehouse ID, name, address, date of establishment, maximum capacity, current storage, and available storage as a calculated value.

The Store relation (between Warehouse and Product) shows which products are stored in each warehouse, and it records the quantity of each product available in a specific warehouse.

5 Relationships:

•Customer-Orders(1-M):

Each customer can place many orders, while each order is associated with exactly one customer.

•Staff-Orders(1-M):

Each staff member can manage many orders, while each order is managed by at most one staff member.

•Orders-Product(M-M)

Each order can contain many products, and each product can appear in many orders. (Association attributes: Quantity, Price_at_Time in the order-Product relation.)

•Warehouse-Product(M-M)

Each warehouse can store many products, and each product can be stored in many warehouses. (Association attribute: **Quantity** in the warehouse-product relation.)

• Customer – Review (1-M):

Each customer can submit many reviews, while each review is written by exactly one customer.

• Product – Review (1-M) [Rate]:

Each product can receive many reviews, while each review is associated with exactly one product.

•Payment Method - Orders (1-M):

Each payment method can be used in many orders, while each order uses exactly one payment method.

• Payment Plan - Orders (1-M):

Each payment plan can be used for many orders, while each order is paid using at most one payment plan (optional if paying without a plan).

• **Product - Book (specialization):**

Book is a specialized subtype of Product that adds book-specific attributes.

• **Product - Stationery_Item (specialization):**

Stationery_Item is a specialized subtype of Product that adds stationery-specific attributes.

• **Staff - Admin (specialization):**

Admin is a specialized subtype of Staff that adds admin-specific attributes (e.g., promotion date).

• **Staff - Worker (specialization):**

Worker is a specialized subtype of Staff that adds worker-specific attributes (e.g., job title, shift time).

• **Customer -Activated_Customer (specialization):**

Activated_Customer is a specialized subtype of Customer that stores activation-related attributes.

• **Customer - Inactivated_Customer (specialization):**

Inactivated_Customer is a specialized subtype of Customer that stores deactivation-related attributes.

• **Orders - Sale (specialization):**

Sale is a specialized subtype of Orders representing purchase orders.

• **Orders - Rental (specialization):**

Rental is a specialized subtype of Orders that stores additional rental data (start/end/return dates, rental status).

6 Queries:

1. Retrieves all records from Payment_Plan sorted by Payment_Plan_ID to display them in the table view.
2. Adds a new payment plan into Payment_Plan including period, optional legal-trial months, Retrieve the title of all books by a specific author.
3. Updates an existing plan's period, legal-trial months, description, and disabled status based Retrieve all transactions made by a specific user.
4. Retrieve the number of orders in a specific date
5. Retrieve all reviews for a specific product.
6. Retrieve all warehouses sorted by any attribute.
7. Retrieve the salary of all staff members.
8. Retrieve the orders and their order items.
9. Retrieve the average rating review for a specific product.
10. Retrieve all products and its categories in charts.
11. Retrieve all books orders revenue.
12. Retrieve the products names with price equal to a specific price.
13. Retrieve the salary of all staff members hired after a specified date.
14. Retrieve the popularity of a product.
15. Retrieve the date of inserting a product.
16. Retrieve the budget of each customer.
17. Retrieve the product review and by which customer it was done.
18. Retrieve the contact number of warehouses.
19. Retrieve the orders and their type.
20. Retrieve Retrieve product names with a specific company.

7 ER Diagram

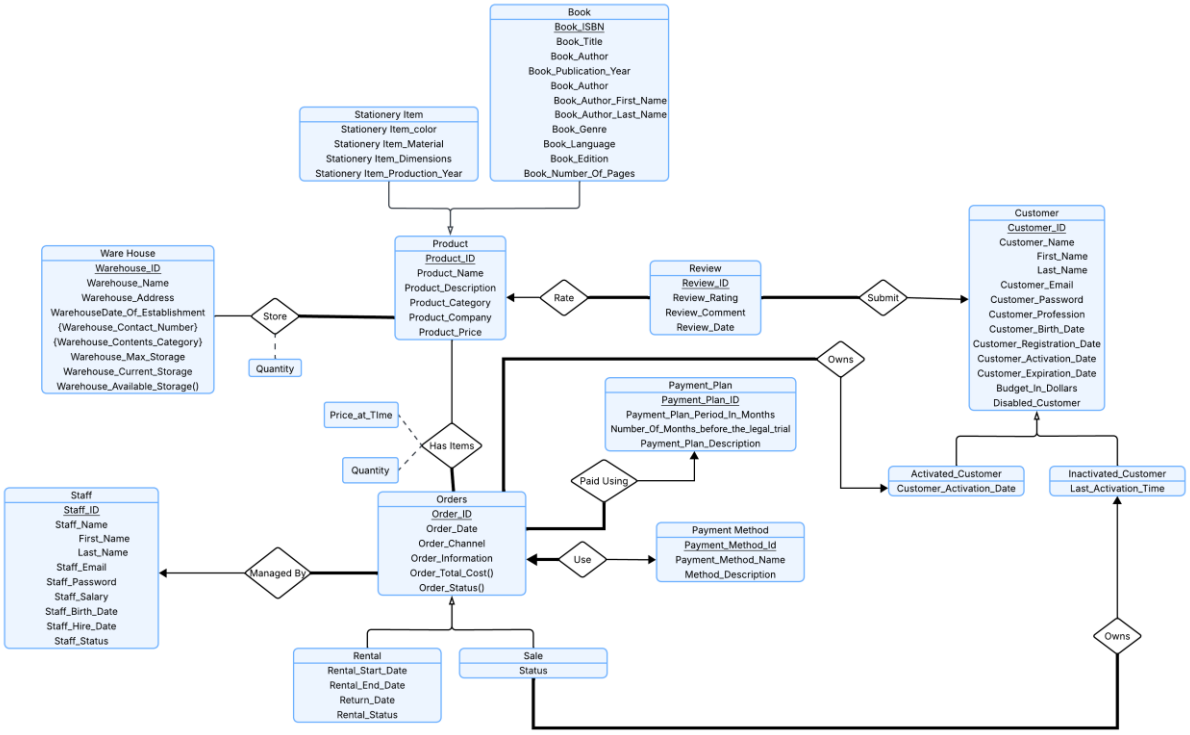

Figure 7-1Bookstore managment system ER diagram.

As shown in Figure 1, the data model connects the system's main entities Supplier, Purchase, Warehouse, Product, Book, Customer, Transaction, Book Transaction, Staff, and Review through a set of relationships that describe how items enter the store, how they are stored and sold, and how users interact with them. These relationships make it possible to trace every product from the moment it is supplied, through storage and availability, until it is purchased or rented by a customer. By linking entities using keys and recording important details such as quantities, dates, and prices, the database supports accurate inventory control, consistent transaction records, and reliable customer activity tracking.

On the supply and inventory side, the Supplier and Purchase entities are connected through the Bought relationship, which represents that a supplier provides purchases to the bookstore. Each purchase record identifies the specific supplier involved, the product acquired, the date of the purchase, and the quantity received, making it possible to track restocking events and supplier contribution over time. Purchases are then connected to Warehouse through the Saved In relationship, which indicates where the purchased stock is stored; this helps the system manage storage locations and relate incoming quantities to warehouse capacity, current storage, and available space. The Purchase entity is also connected to Product through the Purchased

relationship, showing which product is being restocked in each purchase and ensuring that every incoming quantity is tied to a known product in the catalog.

The Product entity acts as the central catalog for all items in the system, and the Book entity is linked to it as a specialization relationship, meaning every book is a product but with additional book-specific attributes such as ISBN, title, author information, publisher, publication year, genre, language, edition, and number of pages. This design allows the system to treat all items consistently as products for pricing and transactions, while still supporting detailed book metadata for searching and display. On the customer operations side, the Customer and Transaction entities are linked through the Done By relationship, which expresses that each transaction is performed by exactly one customer, while a customer may perform many transactions. This relationship is essential for tracking purchase history, rentals, exchanges, and overall customer activity.

Transactions are connected to Product through the Has Items relationship, which captures the fact that a single transaction can include multiple products and a single product can appear in many transactions. The quantity attribute attached to this relationship is crucial because it records how many units of each product were included in each transaction, enabling shopping-cart behavior and accurate stock deduction. Transactions are also connected to Staff through the Managed By relationship, meaning staff members handle and supervise transactions; this supports accountability and allows administrators to track which staff member managed each order or rental. For rentals and exchanges, the Book Transaction entity is linked as a specialization of Transaction, storing extra information such as rental dates, return dates, and transaction notes, which are not needed for normal purchases but are essential for managing borrowed items and due dates.

Customer feedback is modeled through the Review entity, which is connected to Customer through the Submit relationship and to Product through the Rate relationship. This structure ensures that every review is clearly tied to the customer who wrote it and the product being reviewed, while allowing customers to write many reviews and products to receive many reviews. Reviews store ratings, comments, and review dates, which helps other users evaluate items and helps the bookstore monitor satisfaction and product performance. Overall, these relationships create a complete and connected view of the bookstore workflow, linking supply, storage, catalog management, customer transactions, staff oversight, and product reviews into one consistent database structure.

Staff is linked to the system through the “Managed By” relationship with Transaction, which means each transaction is handled and supervised by a staff member for control and accountability, while one staff member can manage many transactions over time. This connection allows the bookstore to track who processed each purchase, rental, or exchange, making it easier to audit actions, resolve issues, and evaluate staff workload and performance.

Payment Plan is connected to Transaction through the “Paid Using” relationship, which represents installment or deferred payment options used when completing a transaction. A single payment plan can be used in many transactions, while each transaction may use at most one plan, allowing the system to store plan definitions once (such as the payment period in months and months before legal trial) and reuse them consistently without repeating the same plan details in every transaction record.

Payment Method is attached to Transaction as the field that describes how the customer paid for the transaction, such as cash, card, or any other supported method. This ensures each transaction clearly records the chosen payment channel for financial tracking, reporting, and validation, while allowing the system to standardize payment methods so that many transactions can share the same method without storing duplicate descriptive information.

8 Normalization:

In the earlier diagram (the "Transaction–Purchase–Supplier" ERD), several attributes are still modeled in a way that causes redundancy and update anomalies, so the design is not cleanly in 3NF. The later ERD (the one with Orders, Payment Method, Payment Plan, Warehouse–Product stock, and the subtypes) reflects the normalization process by separating "things" (entities like payment methods and plans) from "events" (orders/transactions), and by moving repeating or relationship-dependent data (like quantities and item lists) into proper relationship tables. The result is that each table represents one concept, and most attributes depend only on the key of that table.

First Normal Form (1NF) was achieved by eliminating repeating groups and forcing all attributes to be atomic (single value per cell). In the earlier ERD, attributes shown in braces such as customer contact number, supplier contact number, and supplier supply category indicate multi-valued fields (a customer may have multiple phone numbers, a supplier may supply multiple categories). Storing these lists inside one row violates 1NF because it creates repeating values and makes searching and updating inconsistent. In the normalized ERD, this problem is conceptually removed by not keeping "lists inside a single attribute"; multi-valued information is instead represented through proper relationships (and in a physical schema you would implement them as separate tables like Customer Phones or Supplier Categories). Similarly, "an order/transaction contains many products" is handled in 1NF by representing each product-in-order as its own row in an Order Items table (your Has Items relation with Quantity and Price at Time), instead of trying to store multiple products in one transaction record.

Second Normal Form (2NF) was addressed by ensuring that any table with a composite key has attributes that depend on the whole key, not just part of it. The main composite-key situation in both designs is the "transaction/order items" relationship: (Transaction ID, Product ID) uniquely identifies a line item, and line-item attributes such as Quantity (and later Price at Time) depend on both the transaction and the product together. In the normalized ERD, this is made explicit by attaching Quantity and Price at Time to the Has Items relationship rather than to Transaction or Product alone, which prevents partial dependency problems. This also avoids incorrect designs where, for example, Quantity would depend only on Product ID (which would

be wrong because quantity differs by order) or only on Transaction ID (which would be wrong because an order has multiple products).

Third Normal Form (3NF) was the biggest improvement when moving from the earlier ERD to the later ERD, because it removed transitive dependencies and reduced repeated descriptive data. In the earlier ERD, Transaction Payment Method appears as an attribute inside Transaction; if this stores a method name/description, then Transaction ID determines Payment Method, and Payment Method determines method details this is a classic transitive dependency and causes update anomalies (renaming a payment method would require updating many transaction rows). The normalized ERD fixes this by creating a separate Payment Method table and letting Orders reference it by ID. The same principle appears with Payment Plan: instead of repeating plan details with each order, a separate Payment Plan table holds the plan definition, and Orders reference it when needed. Another major 3NF improvement is stock handling: the earlier ERD stores Product Quantity directly inside Product while also having Warehouse and Purchase flows, which can lead to inconsistent totals (the same stock is effectively represented in multiple places). The normalized ERD ties stock quantity to the Warehouse–Product relationship (Store) so quantity depends on the correct key (Warehouse ID + Product ID), eliminating redundant totals and preventing anomalies when stock moves between warehouses. Finally, derived values such as age, available storage, total cost, and status are treated as computed/derived (functions) in the normalized ERD, reducing redundancy and avoiding inconsistencies that occur when derived values are stored and then forgotten during updates.

Overall, the earlier ERD mainly violates normalization through (1) multi-valued attributes stored inside entity tables (1NF violation), (2) relationship-dependent attributes placed in the wrong table (risking partial dependencies in line-item data), and (3) transitive dependencies and redundant storage of descriptive and aggregate information (3NF issues like payment method details repeated and stock quantity represented in multiple ways). The later ERD shows how normalization resolves these issues: it separates lookup entities (payment method/plan), uses junction tables for many-to-many relationships (order items and warehouse stock), and keeps each non-key attribute dependent only on its table's key, which is the goal of 3NF.

9 References:

[1]: Yellow Pages. [online]. Al-Bireh Public Library | Yellow Pages.