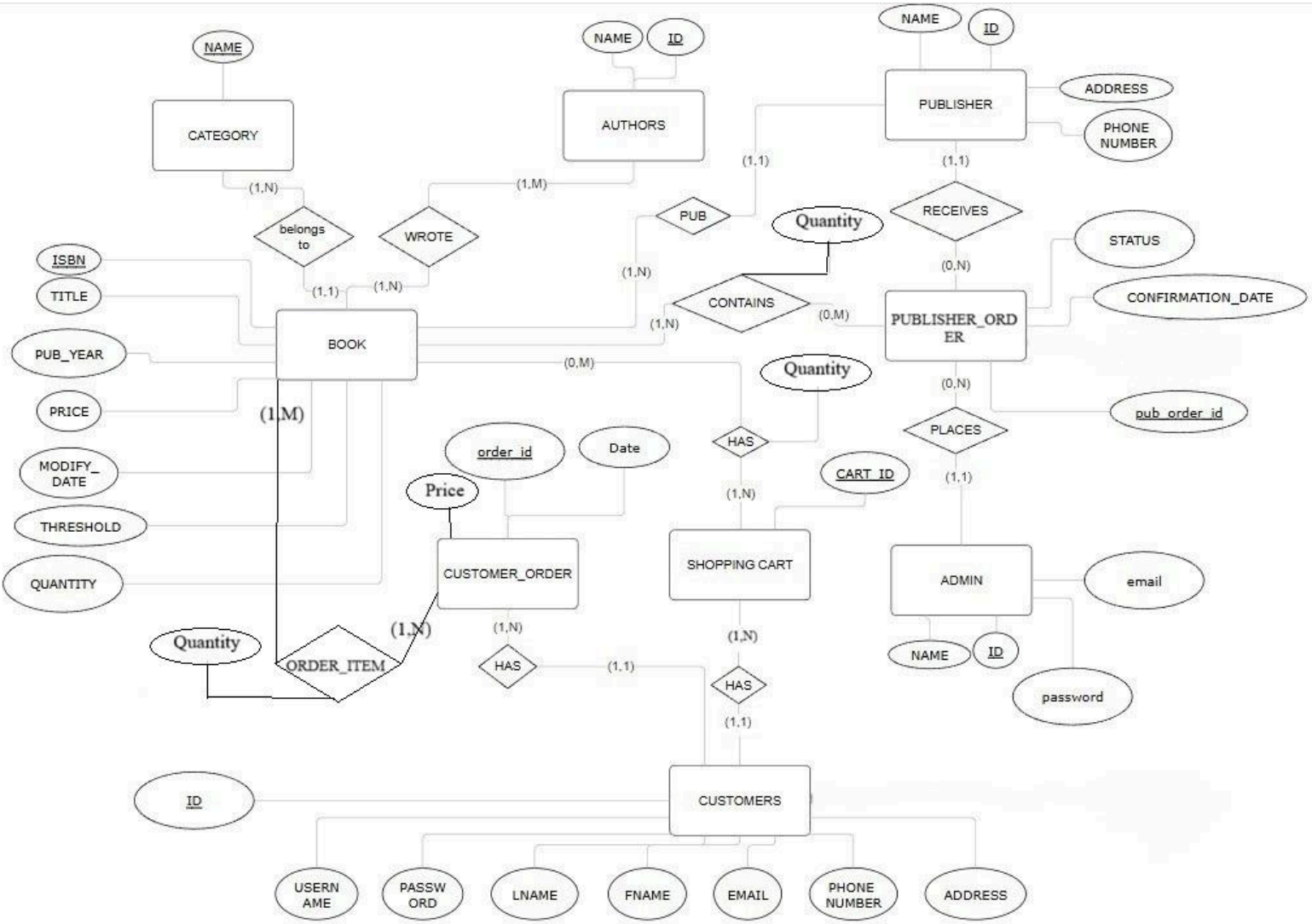
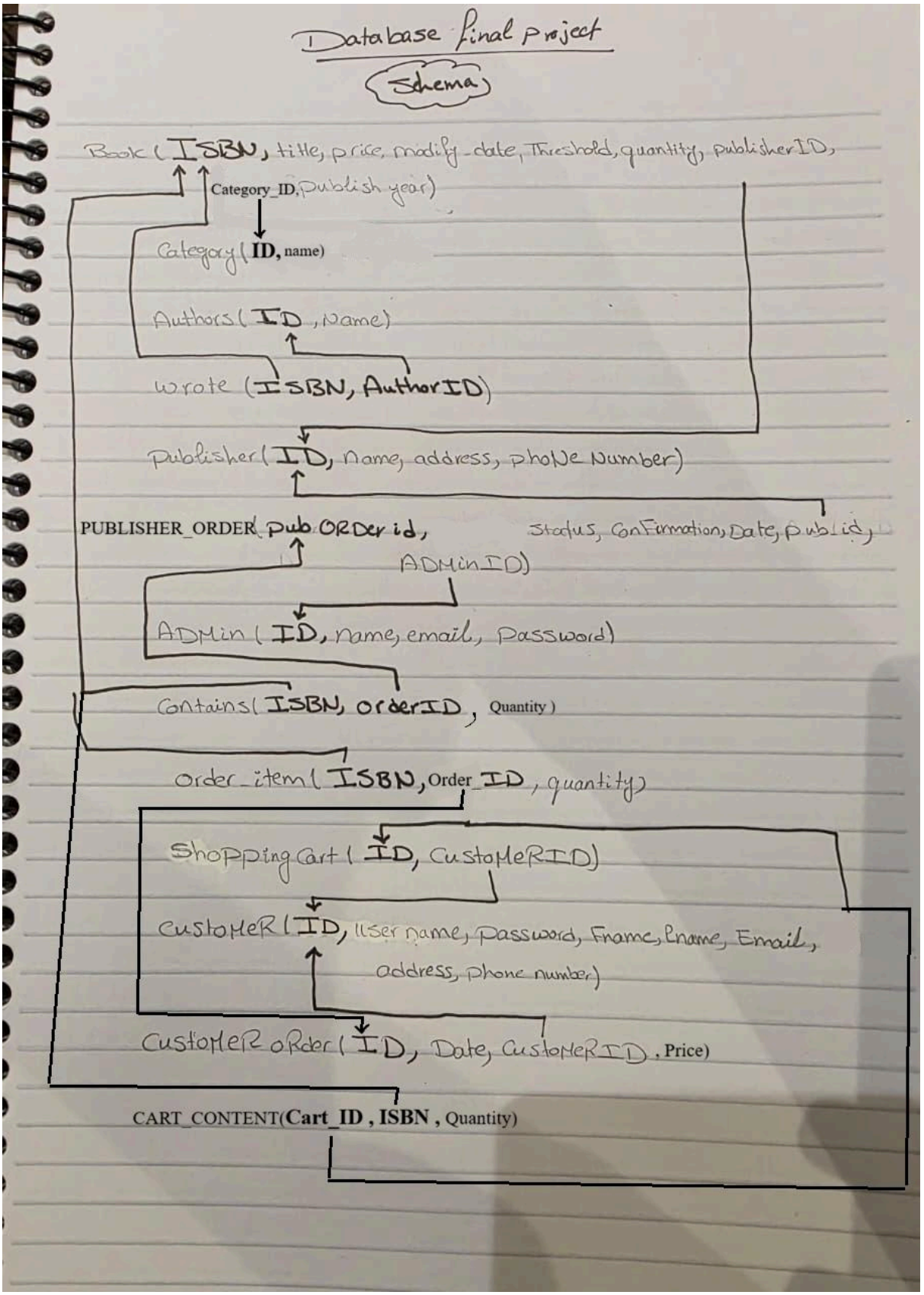
Order Processing System

Amr Ahmed Zeina **9029**
Abdelrahman Youssef **8711**
Salma Abdelhameed **8754**
Reem Ibrahim **9030**

1. ERD


NAME
CATEGORY
(1,N)
belongs to
(1,1)
NAME
ID
AUTHORS
(1,M)
WROTE
(1,N)
NAME
ID
PUBLISHER
ADDRESS
PHONE NUMBER
(1,1)
PUB
(1,1)
(1,1)
RECEIVES
(0,N)
Quantity
(1,N)
CONTAINS
(0,M)
STATUS
CONFIRMATION_DATE
PUBLISHER_ORDER
ISBN
TITLE
PUB_YEAR
PRICE
MODIFY_DATE
THRESHOLD
QUANTITY
BOOK
(1,N)
(0,M)
(1,M)
Quantity
(0,N)
pub_order_id
PLACES
(1,1)
HAS
(1,N)
order_id
Date
Price
CUSTOMER_ORDER
(1,N)
CART_ID
SHOPPING CART
(1,N)
ADMIN
email
NAME
ID
password
Quantity
ORDER_ITEM
(1,N)
(1,N)
HAS
(1,1)
(1,N)
HAS
(1,1)
ID
CUSTOMERS
USERNAME
PASSWORD
LNAME
FNAME
EMAIL
PHONE NUMBER
ADDRESS

2. Schema


Database final project
Schema
Book (ISBN, title, price, modify_date, Threshold, quantity, publisherID,
Category_ID, Publish year)
Category (ID, name)
Authors (ID , Name)
wrote (ISBN, AuthorID)
Publisher (ID, name, address, phone Number)
PUBLISHER_ORDER (pubORDerid, Status, ConFirmation, Date, pubid,
ADMinID)
ADMin (ID, name, email, Password)
Contains (ISBN, orderID , Quantity)
order_item (ISBN, Order_ID , quantity)
ShoppingCart (ID, CustomeRID)
CustomeR (ID, User name, Password, Fname, Lname, Email,
address, phone number)
CustomeR oRder (ID, Date, CustomeRID , Price)
CART_CONTENT(Cart_ID , ISBN , Quantity)

3. SQL Queries:

- DDL

```
CREATE DATABASE Order_Processing_System;
USE Order_Processing_System;

CREATE TABLE Publisher(
    publisher_id INT PRIMARY KEY AUTO_INCREMENT,
    publisher_name VARCHAR(25) NOT NULL,
    publisher_address VARCHAR(100),
    phone_number VARCHAR(11)
);

CREATE TABLE Category (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    category_name
ENUM('Science','Art','Religion','History','Geography') UNIQUE NOT NULL
);

CREATE TABLE Author(
    author_id INT PRIMARY KEY AUTO_INCREMENT,
    author_name VARCHAR(50) NOT NULL
);

CREATE TABLE Book (
    ISBN VARCHAR(13) PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    modify_date DATETIME,
    Threshold INT CHECK (Threshold >= 0),
    quantity INT CHECK (quantity >= 0),
    publish_year YEAR,
    publisher_id INT,
    category_id INT,
    FOREIGN KEY (publisher_id) REFERENCES Publisher(publisher_id),
    FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

CREATE TABLE Written_By(
    ISBN VARCHAR(13),
    author_id INT,
    PRIMARY KEY (ISBN , author_id),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
```

```
    FOREIGN KEY (author_id) REFERENCES Author(author_id)
);

CREATE TABLE Admins(
    admin_id INT PRIMARY KEY AUTO_INCREMENT,
    admin_name VARCHAR(25),
    email VARCHAR(40) UNIQUE,
    admin_password VARCHAR(255) NOT NULL


);

CREATE TABLE Publisher_Order (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    admin_id INT,
    order_status ENUM('Pending','Confirmed') DEFAULT 'Pending',
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (admin_id) REFERENCES Admins(admin_id)
);

CREATE TABLE Pub_Order_Contains(
    ISBN VARCHAR(13),
    quantity INT NOT NULL,
    order_id INT,
    PRIMARY KEY(ISBN, order_id),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
    FOREIGN KEY (order_id) REFERENCES Publisher_Order(order_id)
);

CREATE TABLE Customer(
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(20),
    Lname VARCHAR(20),
    user_name VARCHAR(40) UNIQUE,
    email VARCHAR(40),
    customer_password VARCHAR(255) NOT NULL,
    customer_address VARCHAR(100),
    phone_number VARCHAR(11)
);

CREATE TABLE Shopping_Cart(
    cart_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
```

```
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)

);

CREATE TABLE Cart_Content (
    cart_id INT,
    ISBN VARCHAR(13),
    quantity INT CHECK (quantity > 0),
    PRIMARY KEY (cart_id, ISBN),
    FOREIGN KEY (cart_id) REFERENCES Shopping_Cart(cart_id),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);


CREATE TABLE Customer_Order (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    total_price DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Order_Item (
    order_id INT,
    ISBN VARCHAR(13),
    quantity INT,
    PRIMARY KEY (order_id, ISBN),
    FOREIGN KEY (order_id) REFERENCES Customer_Order(order_id),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);
```

- Triggers:

1)Auto Buy

```
DELIMITER //

CREATE TRIGGER auto_reorder
BEFORE UPDATE ON Book
FOR EACH ROW
BEGIN
    DECLARE new_order_id INT;

    -- Trigger only if stock drops below threshold
    IF OLD.quantity >= OLD.Threshold AND NEW.quantity < NEW.Threshold
THEN

        -- Insert a new publisher order (ensure admin_id=1 exists)
        INSERT INTO Publisher_Order (admin_id)
        VALUES (1);

        SET new_order_id = LAST_INSERT_ID();

        -- Insert the book in the new order with fixed quantity (e.g.,
20)
        INSERT INTO Pub_Order_Contains (order_id, ISBN, quantity)
        VALUES (new_order_id, NEW.ISBN, 20);
    END IF;
END;
//
DELIMITER ;
```

2)Auto Addition

```
DELIMITER //

CREATE TRIGGER confirm_publisher_order
AFTER UPDATE ON Publisher_Order
FOR EACH ROW
BEGIN
    -- Only act when order changes from Pending → Confirmed
    IF OLD.order_status = 'Pending' AND NEW.order_status = 'Confirmed'
THEN
        UPDATE Book b
        JOIN Pub_Order_Contains p
        ON b.ISBN = p.ISBN
```

```
        SET b.quantity = b.quantity + p.quantity
        WHERE p.order_id = NEW.order_id;
    END IF;
END;
//
DELIMITER ;
```

3)Deduct

```
DELIMITER //
CREATE TRIGGER deduct_stock_after_sale
BEFORE INSERT ON Order_Item
FOR EACH ROW
BEGIN
    -- Check for enough stock first
    IF (SELECT quantity FROM Book WHERE ISBN = NEW.ISBN) < NEW.quantity
THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Not enough stock for this order item';
    END IF;

    -- Stock will be deducted automatically after this insert
    UPDATE Book
    SET quantity = quantity - NEW.quantity
    WHERE ISBN = NEW.ISBN;
END;
//
DELIMITER ;
```

4)Modify Book

```
DELIMITER //

CREATE TRIGGER prevent_negative_stock
BEFORE UPDATE ON Book
FOR EACH ROW
BEGIN
    IF NEW.quantity < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Book quantity cannot be negative';
    END IF;
END;
//
DELIMITER ;
```

- DML

1) For Admins:

Login:

```
SELECT admin_id, admin_password
            FROM Admins
            WHERE admin_name = ? OR email = ?
```

Insert Book:

```
INSERT INTO Book
        (ISBN, title, price, Threshold, quantity, publish_year,
publisher_id, category_id)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
```

Display Orders to be Confirmed:

```
SELECT order_id, order_date, order_status
    FROM Publisher_Order
    WHERE order_status = 'Pending'
```

Confirm Order:

```
UPDATE Publisher_Order
        SET order_status = 'Confirmed'
        WHERE order_id = ?
```

Updating Book:

```
        UPDATE Book
        SET quantity = ?
        WHERE ISBN = ?
```

Display Books:

```
SELECT ISBN, title, quantity, Threshold FROM Book
```

Display Total Sales Last Month:

```
SELECT SUM(total_price) AS total
FROM Customer_Order
WHERE order_date <= CURRENT_DATE - INTERVAL 1 MONTH;
```

Display Total Sales on Specific Day:

```
SELECT SUM(total_price) AS total
        FROM Customer_Order
        WHERE DATE(order_date) = ?
```

Display Top 5 Customers in the Last 3 Month:

```
    SELECT c.user_name, SUM(o.total_price) AS spent
    FROM Customer_Order o
    JOIN Customer c ON c.customer_id = o.customer_id
    WHERE o.order_date >= CURRENT_DATE - INTERVAL 3 MONTH
    GROUP BY c.user_name
```

```
ORDER BY spent DESC
LIMIT 5
```

Display Top 10 Selling Books in the Last 3 Month:

```
SELECT b.title, SUM(oi.quantity) AS sold
FROM Order_Item oi
JOIN Book b ON b.ISBN = oi.ISBN
JOIN Customer_Order co ON co.order_id = oi.order_id
WHERE co.order_date >= CURRENT_DATE - INTERVAL 3 MONTH
GROUP BY b.title
ORDER BY sold DESC
LIMIT 10
```

Display Restock Times for Specific Book:

```
SELECT COUNT(*) AS times
FROM Pub_Order_Contains
WHERE ISBN = ?
```

Searching for a Book:

```
SELECT DISTINCT b.ISBN, b.title, b.price, b.quantity
FROM Book b
LEFT JOIN Written_By wb ON b.ISBN = wb.ISBN
LEFT JOIN Author a ON wb.author_id = a.author_id
LEFT JOIN Publisher p ON b.publisher_id = p.publisher_id
LEFT JOIN Category c ON b.category_id = c.category_id
WHERE b.ISBN LIKE ?
   OR b.title LIKE ?
   OR a.author_name LIKE ?
   OR p.publisher_name LIKE ?
   OR c.category_name LIKE ?
```

2) For Customers:

Login:

```
SELECT customer_id, customer_password FROM Customer
            WHERE user_name = ? OR email = ?
```

Register:

```
INSERT INTO Customer
        (Fname, Lname, user_name, email, customer_password,
customer_address, phone_number)
        VALUES (?, ?, ?, ?, ?, ?, ?)
```

Ensure Cart Exist:

```
 INSERT IGNORE INTO Shopping_Cart (customer_id)
    VALUES ($cid)
```

Insert Items Into Shopping Cart:

```
INSERT INTO Cart_Content (cart_id, ISBN, quantity)
        VALUES (?, ?, ?)
        ON DUPLICATE KEY UPDATE quantity = quantity + VALUES(quantity)
```

Getting Same Cart for Same Customer:

```
SELECT cart_id FROM Shopping_Cart WHERE customer_id = $cid
```

Delete Item From Cart:

```
DELETE FROM Cart_Content WHERE cart_id=? AND ISBN=?
```

Display Shopping Cart:

```
SELECT b.title, b.price, c.quantity, c.ISBN
    FROM Cart_Content c
    JOIN Book b ON b.ISBN = c.ISBN
    WHERE c.cart_id = $cart_id
```

Checking Out (TRIGGER deducts stock):

```
INSERT INTO Customer_Order (customer_id, total_price)
VALUES ($cid, $total)
INSERT INTO Order_Item (order_id, ISBN, quantity)
VALUES ($order_id, '{$i['ISBN']}', {$i['quantity']})
```

Clear Cart:

```
DELETE FROM Cart_Content WHERE cart_id = $cart_id
```

Home Page Display All Books:

```
    SELECT b.ISBN, b.title, b.price, b.quantity, c.category_name
    FROM Book b
    JOIN Category c ON b.category_id = c.category_id
```

Display Past Customer's Orders:

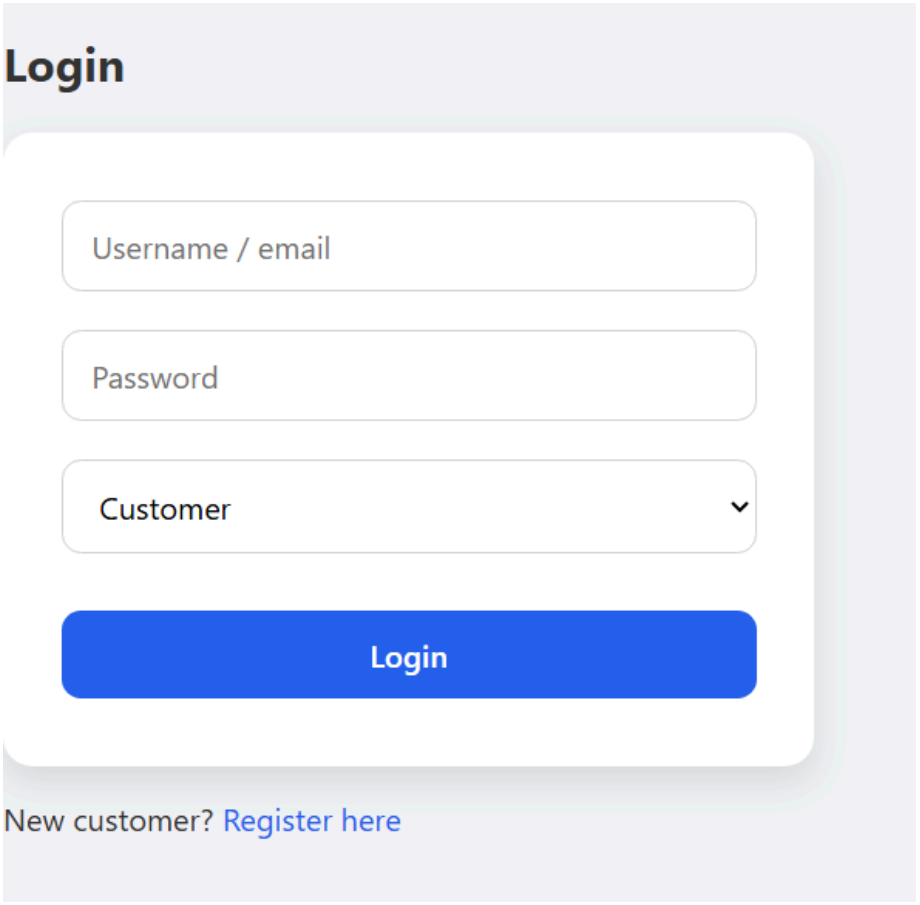
```
    SELECT order_id, order_date, total_price
    FROM Customer_Order
    WHERE customer_id = $cid
    ORDER BY order_date DESC

    SELECT b.title, oi.quantity
    FROM Order_Item oi
    JOIN Book b ON b.ISBN = oi.ISBN
    WHERE oi.order_id = {$o['order_id']}
```

Search Specific Book:

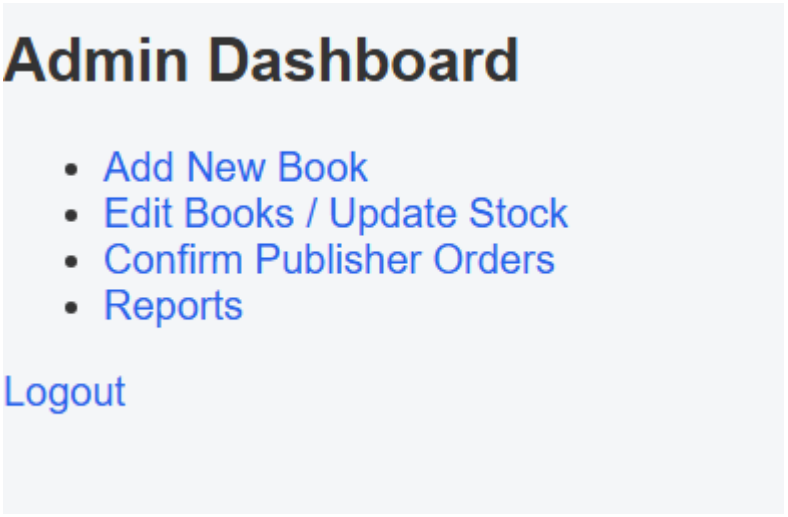
```
        SELECT DISTINCT b.ISBN, b.title, b.price, b.quantity
        FROM Book b
        LEFT JOIN Written_By wb ON b.ISBN = wb.ISBN
        LEFT JOIN Author a ON wb.author_id = a.author_id
        LEFT JOIN Publisher p ON b.publisher_id = p.publisher_id
        LEFT JOIN Category c ON b.category_id = c.category_id
```

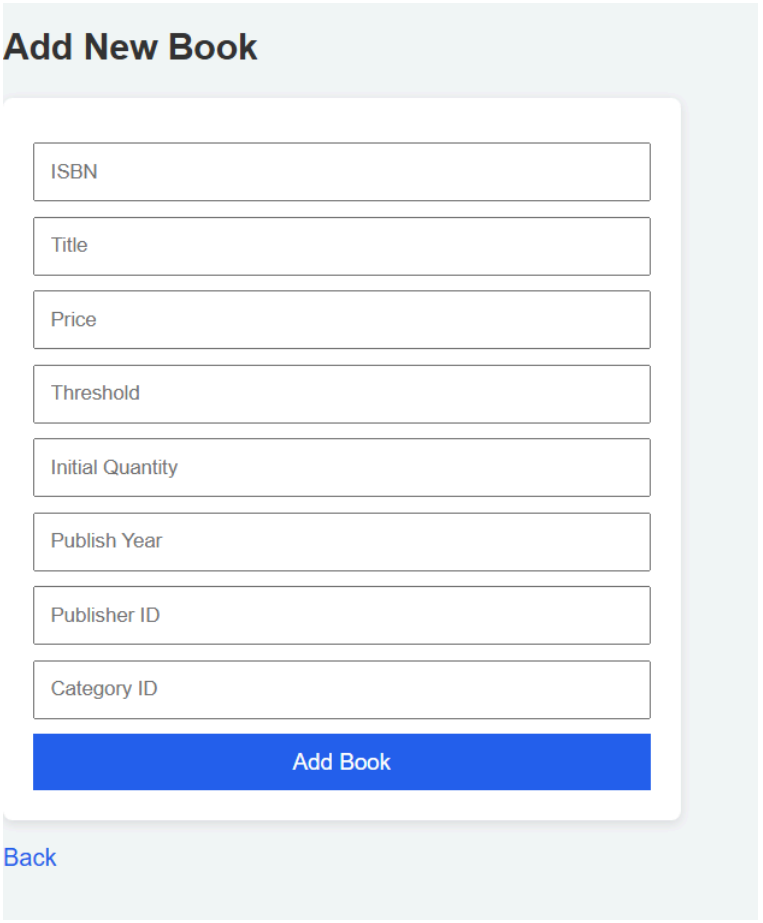
```
        WHERE b.ISBN LIKE ?
           OR b.title LIKE ?
           OR a.author_name LIKE ?
           OR p.publisher_name LIKE ?
           OR c.category_name LIKE ?
```

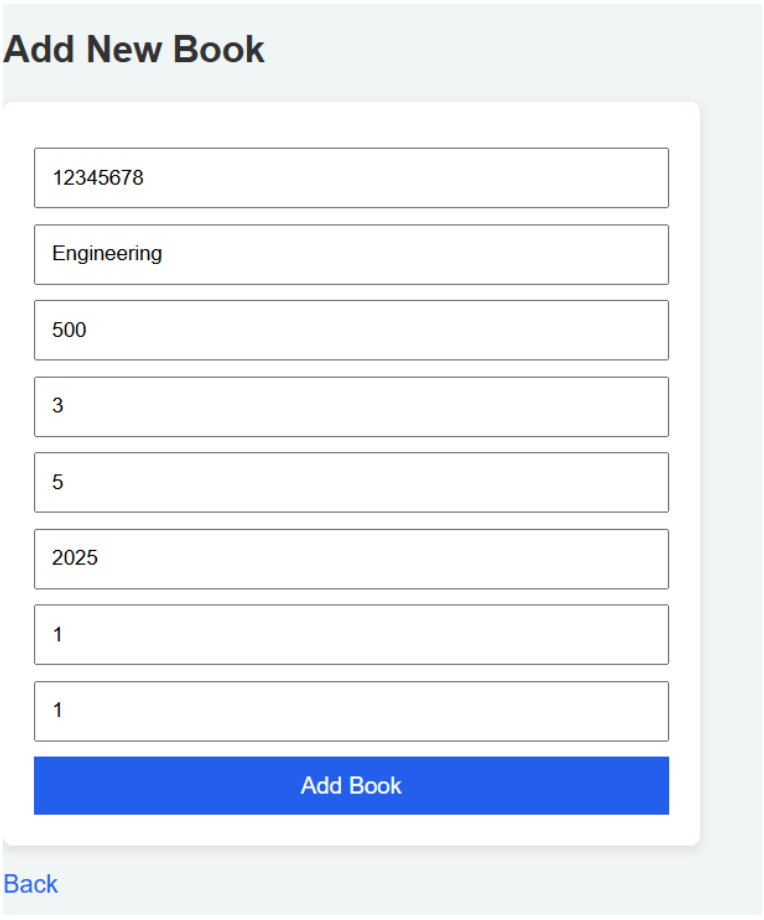
4. Sample Runs



Only one admin for the system and no new admins registration is acceptable.




Add New Book
ISBN
Title
Price
Threshold
Initial Quantity
Publish Year
Publisher ID
Category ID
Add Book
Back


Add New Book
12345678
Engineering
500
3
5
2025
1
1
Add Book
Back

Edit Books

Title	Quantity	Threshold	Update
Engineering	5	3	Update
Physics 101	52	5	Update
Art for Beginners	40	3	Update
History of Egypt	30	2	Update
Advanced Chemistry	20	5	Update
Modern Art	25	3	Update
Religious Studies	15	2	Update
World Geography	30	5	Update
Medieval History	28	2	Update
Biology Basics	35	5	Update
Sculpture 101	20	3	Update
Islamic Studies	18	2	Update
European History	22	2	Update
Geography of Asia	25	3	Update

System Reports

Total Sales – Previous Month

1435.00 EGP

Total Sales on Specific Day

mm/dd/yyyy

Show

Total: 440.00 EGP

Top 5 Customers (Last 3 Months)

alice_w – 330.00 EGP

reem – 110.00 EGP

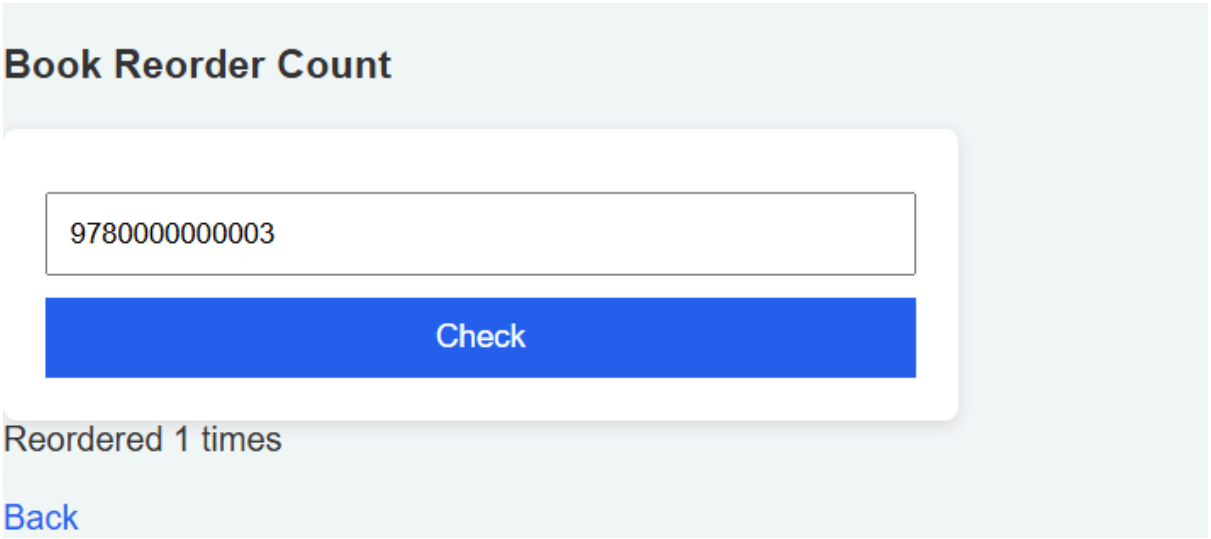
Top 10 Selling Books (Last 3 Months)

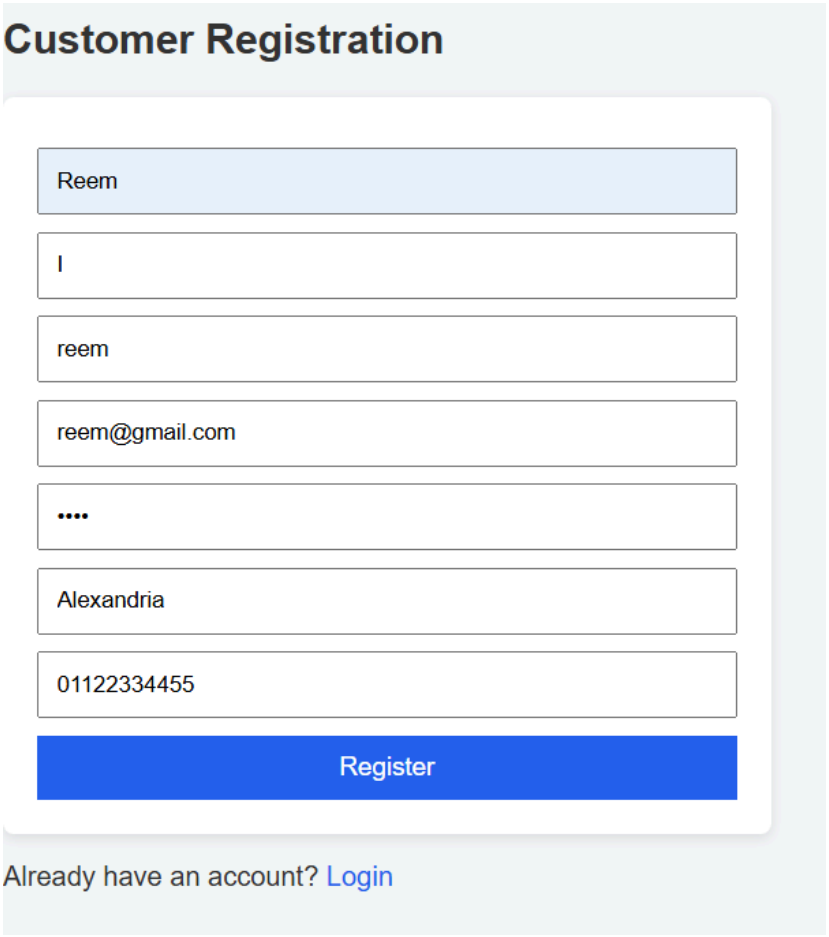
Physics 101 – 7 copies

Art for Beginners – 1 copies

Advanced Chemistry – 1 copies


Pending Publisher Orders
Order ID
Date
Action
1
2025-12-27 04:15:46
Confirm
Back


Book Reorder Count
9780000000003
Check
Reordered 1 times
Back


Customer Registration
Reem
I
reem
reem@gmail.com
••••
Alexandria
01122334455
Register
Already have an account? Login

Available Books

Title	Category	Price	Stock	Add
Physics 101	Science	50.00 EGP	52	1 Add
Advanced Chemistry	Science	60.00 EGP	20	1 Add
Biology Basics	Science	55.00 EGP	35	1 Add
Organic Chemistry	Science	60.00 EGP	20	1 Add
Physics Experiments	Science	55.00 EGP	40	1 Add
Art for Beginners	Art	30.00 EGP	40	1 Add
Modern Art	Art	35.00 EGP	25	1 Add
Sculpture 101	Art	33.00 EGP	20	1 Add
Renaissance Art	Art	37.00 EGP	18	1 Add
Abstract Art	Art	36.00 EGP	22	1 Add
Religious Studies	Religion	45.00 EGP	15	1 Add
Islamic Studies	Religion	48.00 EGP	18	1 Add
Christianity 101	Religion	50.00 EGP	15	1 Add
History of Egypt	History	40.00 EGP	30	1 Add

Your Cart

Book	Price	Quantity	Remove
Physics 101	50.00	1	Remove
Advanced Chemistry	60.00	1	Remove

Total: 110 EGP

Checkout | Continue Shopping

Checkout

Total Amount: $110.00

Credit Card Number:

1234567891011111

Expiry Date:

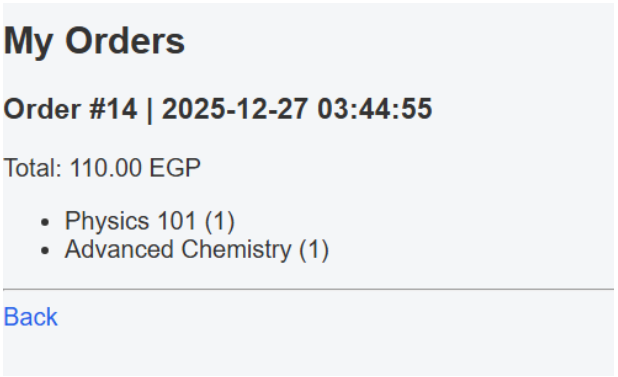
February 2026

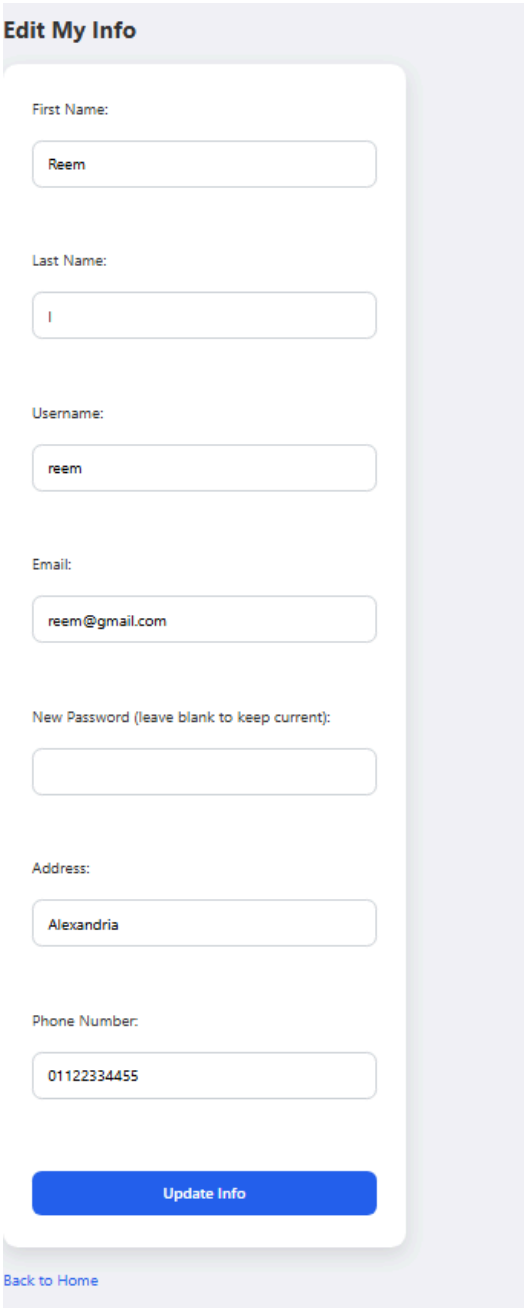
Place Order

Checkout Successful

Your order has been placed successfully.

View Orders | Home


My Orders
Order #14 | 2025-12-27 03:44:55
Total: 110.00 EGP
Physics 101 (1)
Advanced Chemistry (1)
Back


Edit My Info
First Name:
Reem
Last Name:
I
Username:
reem
Email:
reem@gmail.com
New Password (leave blank to keep current):
Address:
Alexandria
Phone Number:
01122334455
Update Info
Back to Home