# PROJECT REPORT

## Online Shopping System

### INTRODUCTION

The Online Shopping System is a database-driven software application designed to simplify and automate the process of browsing products, placing orders, handling customer data, managing inventory, processing payments, and generating reports. This centralized system improves shopping experiences for customers while enabling administrators and suppliers to efficiently manage backend operations.

The system provides real-time updates on product availability and order status, supports secure transactions, and maintains detailed records for future use.

### Planning of the Project

Objective:
- To create an efficient system for customers to browse, order, and pay for products online.
- To automate inventory and order management.
- To support real-time updates for product stock, order processing, and delivery.
- To ensure secure payment processing and address management.

Scope:
The system will handle customer registration, product listings, inventory, orders, shipping, and payment details. It allows customers to make purchases, staff to manage products and inventory, and admins to monitor the system and generate reports.

### Target Users

- Customers: Can register, browse products, place orders, and view their order history.
- Suppliers: Provide and update product stock.
- Administrators: Oversee operations, manage categories, and generate analytics/reports.

### Working of the Project

Customer Interaction:
- Browse available products with categories and filters.
- View product details and add items to the cart.
- Place orders and make payments securely.
- Track order and delivery status.

Supplier Interaction:
- Upload product information.
- Manage stock quantities and product updates.

Administrator Interaction:
- Add or modify product categories and supplier info.
- Generate sales and inventory reports.
- Monitor user activity and system health.

## Functionalities

For Customers:
- Register and log in
- Browse/search products
- Place orders
- Manage addresses and view order history

For Suppliers:
- Add/update product info
- Track product stock levels

For Admins:
- Manage users, products, categories, and suppliers
- Generate performance reports
- Monitor order and payment statuses

## Entities and Attributes

CUSTOMER
- customer_id (Primary Key)
- name
- email
- password
- phone

PRODUCT
- product_id (Primary Key)
- name
- description
- price
- image
- stock_quantity
- category_id (Foreign Key)
- supplier_id (Foreign Key)

CATEGORY
- category_id (Primary Key)
- name

SUPPLIER
- supplier_id (Primary Key)
- name
- contact_info

ORDER_TABLE
- order_id (Primary Key)
- customer_id (Foreign Key)
- order_date
- status
- total_amount
- shipping_address_id (Foreign Key)
- payment_id (Foreign Key)

ORDER_ITEM
- order_item_id (Primary Key)
- order_id (Foreign Key)
- product_id (Foreign Key)
- quantity
- price

ADDRESS
- address_id (Primary Key)
- customer_id (Foreign Key)
- street
- city
- state
- zip

SHIPPING_ADDRESS
- shipping_address_id (Primary Key)
- address_id (Foreign Key)

PAYMENT
- payment_id (Primary Key)
- order_id (Foreign Key)
- payment_date
- amount
- payment_method_id (Foreign Key)

PAYMENT_METHOD
- payment_method_id (Primary Key)
- name

## SQL Code

```
-- Create CUSTOMER Table
CREATE TABLE CUSTOMER (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(100),
    phone VARCHAR(20)
);

-- Create CATEGORY Table
CREATE TABLE CATEGORY (
    category_id INT PRIMARY KEY,
    name VARCHAR(100)
);

-- Create SUPPLIER Table
CREATE TABLE SUPPLIER (
    supplier_id INT PRIMARY KEY,
    name VARCHAR(100),
    contact_info TEXT
);

-- Create PRODUCT Table
CREATE TABLE PRODUCT (
    product_id INT PRIMARY KEY,
```

```
    name VARCHAR(100),
    description TEXT,
    price DECIMAL(10, 2),
    image VARCHAR(255),
    stock_quantity INT,
    category_id INT,
    supplier_id INT,
    FOREIGN KEY (category_id) REFERENCES CATEGORY(category_id),
    FOREIGN KEY (supplier_id) REFERENCES SUPPLIER(supplier_id)
);

-- Create ADDRESS Table
CREATE TABLE ADDRESS (
    address_id INT PRIMARY KEY,
    customer_id INT,
    street VARCHAR(255),
    city VARCHAR(100),
    state VARCHAR(100),
    zip VARCHAR(20),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);

-- Create SHIPPING_ADDRESS Table
CREATE TABLE SHIPPING_ADDRESS (
    shipping_address_id INT PRIMARY KEY,
    address_id INT,
```

```
    FOREIGN KEY (address_id) REFERENCES ADDRESS(address_id)
);

-- Create PAYMENT_METHOD Table
CREATE TABLE PAYMENT_METHOD (
    payment_method_id INT PRIMARY KEY,
    name VARCHAR(50)
);

-- Create ORDER_TABLE Table
CREATE TABLE ORDER_TABLE (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    status VARCHAR(50),
    total_amount DECIMAL(10, 2),
    shipping_address_id INT,
    payment_id INT,
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (shipping_address_id) REFERENCES
SHIPPING_ADDRESS(shipping_address_id),
    FOREIGN KEY (payment_id) REFERENCES PAYMENT(payment_id)
);

-- Create PAYMENT Table
CREATE TABLE PAYMENT (
```

```
    payment_id INT PRIMARY KEY,
    order_id INT,
    payment_date DATE,
    amount DECIMAL(10, 2),
    payment_method_id INT,
    FOREIGN KEY (order_id) REFERENCES ORDER_TABLE(order_id),
    FOREIGN KEY (payment_method_id) REFERENCES
PAYMENT_METHOD(payment_method_id)
);
```

**-- Create ORDER_ITEM Table**

```
CREATE TABLE ORDER_ITEM (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES ORDER_TABLE(order_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);
```

## Insert Sample data

**CUSTOMER--**

```
INSERT INTO CUSTOMER (customer_id, name, email, password, phone) VALUES
(1, 'Alice Smith', 'alice@example.com', 'password123', '123-456-7890'),
(2, 'Bob Johnson', 'bob@example.com', 'securepass', '234-567-8901');
```

**CATEGORY--**

INSERT INTO CATEGORY (category_id, name) VALUES

(1, 'Electronics'),

(2, 'Books');

**SUPPLIER--**

INSERT INTO SUPPLIER (supplier_id, name, contact_info) VALUES

(1, 'Tech Supplier Co.', 'tech@example.com'),

(2, 'Book Wholesalers Inc.', 'books@example.com');

**PRODUCT--**

INSERT INTO PRODUCT (product_id, name, description, price, image, stock_quantity, category_id, supplier_id) VALUES

(1, 'Smartphone', 'Latest model smartphone', 699.99, 'smartphone.jpg', 50, 1, 1),

(2, 'Science Fiction Novel', 'A thrilling sci-fi book', 14.99, 'scifi.jpg', 200, 2, 2);

**ADDRESS--**

INSERT INTO ADDRESS (address_id, customer_id, street, city, state, zip) VALUES

(1, 1, '123 Maple St', 'New York', 'NY', '10001'),

(2, 2, '456 Oak Ave', 'Los Angeles', 'CA', '90001');

**SHIPPING_ADDRESS--**

INSERT INTO SHIPPING_ADDRESS (shipping_address_id, address_id) VALUES

(1, 1),

(2, 2);

**PAYMENT_METHOD--**

```
INSERT INTO PAYMENT_METHOD (payment_method_id, name) VALUES

(1, 'Credit Card'),

(2, 'PayPal');
```

**ORDER_TABLE--**

```
INSERT INTO ORDER_TABLE (order_id, customer_id, order_date, status, total_amount,
shipping_address_id, payment_id) VALUES

(1, 1, '2025-04-01', 'Shipped', 714.98, 1, 1),

(2, 2, '2025-04-03', 'Processing', 14.99, 2, 2);
```

**PAYMENT--**

```
INSERT INTO PAYMENT (payment_id, order_id, payment_date, amount,
payment_method_id) VALUES

(1, 1, '2025-04-01', 714.98, 1),

(2, 2, '2025-04-03', 14.99, 2);
```

**ORDER_ITEM--**

```
INSERT INTO ORDER_ITEM (order_item_id, order_id, product_id, quantity, price) VALUES

(1, 1, 1, 1, 699.99),

(2, 1, 2, 1, 14.99),

(3, 2, 2, 1, 14.99);
```

## SQL Queries (Samples)

### 1. List All Products with Category and Supplier Names

```
SELECT
  p.product_id,
```

```
  p.name AS product_name,
  p.price,
  c.name AS category_name,
  s.name AS supplier_name
FROM PRODUCT p
JOIN CATEGORY c ON p.category_id = c.category_id
JOIN SUPPLIER s ON p.supplier_id = s.supplier_id;
```

---

## 📜 2. Get All Orders with Customer Name and Total Amount

```
SELECT
  o.order_id,
  c.name AS customer_name,
  o.order_date,
  o.status,
  o.total_amount
FROM ORDER_TABLE o
JOIN  CUSTOMER c ON o.customer_id = c.customer_id;
```

---

## 📦 3. Get Order Items for a Specific Order (e.g., Order ID = 1)

```
SELECT
  oi.order_item_id,
  p.name AS product_name,
  oi.quantity,
  oi.price
FROM ORDER_ITEM oi
JOIN PRODUCT p ON oi.product_id = p.product_id
WHERE oi.order_id = 1;
```

---

## 🛒 4. Find All Orders Placed by a Specific Customer (e.g., customer_id = 1)

```
SELECT
  o.order_id,
  o.order_date,
  o.status,
  o.total_amount
FROM ORDER_TABLE o
WHERE o.customer_id = 1;
```

---

## 💳 5. Get All Payments with Method Name

```
SELECT
  p.payment_id,
  p.order_id,
```

```
  p.payment_date,
  p.amount,
  pm.name AS payment_method
FROM PAYMENT p
JOIN PAYMENT_METHOD pm ON p.payment_method_id = pm.payment_method_id;
```

---

## 📬 6. Get Shipping Address Details for Each Order

```
SELECT
  o.order_id,
  a.street,
  a.city,
  a.state,
  a.zip
FROM ORDER_TABLE o
JOIN SHIPPING_ADDRESS sa ON o.shipping_address_id = sa.shipping_address_id
JOIN ADDRESS a ON sa.address_id = a.address_id;
```

---

## 📈 7. Total Revenue from All Orders

```
SELECT SUM(total_amount) AS total_revenue FROM ORDER_TABLE;
```

---

## 📊 8. Best-Selling Product by Quantity

```
SELECT
  p.name,
  SUM(oi.quantity) AS total_sold
FROM ORDER_ITEM oi
JOIN PRODUCT p ON oi.product_id = p.product_id
GROUP BY p.name
ORDER BY total_sold DESC
LIMIT 1;
```

---

## 🖨️ 9. Show All Customers and Their Saved Addresses

```
SELECT
  c.name AS customer_name,
  a.street,
  a.city,
  a.state,
  a.zip
FROM CUSTOMER c
JOIN ADDRESS a ON c.customer_id = a.customer_id;
```

---

## 📦 10. Check Inventory: Products with Low Stock (e.g., < 10 units)

```
SELECT
   name,
   stock_quantity
FROM  PRODUCT
WHERE  stock_quantity < 10;
```

### Reports & Analytics

- Total revenue
- Top-selling products
- Most active customers
- Payments by method
- Inventory overview

### Conclusion

The Online Shopping System delivers a robust solution for managing online retail operations. By automating key processes such as order placement, payment handling, and inventory tracking, the system enhances user satisfaction and operational efficiency.

This project serves modern e-commerce needs with a clean structure, clear entity relationships, and scalable design for future enhancements.