StockDB - Inventory Management System

Mahmoud Mohamed Abdelgelil (221001313)
Abderhman Ehab Rahal (221001443)
Nour Sharkawy (221001458)

June 1, 2025

Contents

1 Project Description

StockDB is a comprehensive inventory management system designed to help businesses efficiently manage their products, suppliers, orders, and payments. The system provides role-based access control for different user types (Admin, Staff, Supplier, Customer) and includes features for inventory tracking, order management, and payment processing.

1.1 Key Features

- User authentication and authorization with JWT
- Product management with categorization
- Supplier relationship management

- Order processing workflow
- Payment handling and tracking
- Real-time inventory tracking with alerts
- Role-based access control (RBAC)
- Comprehensive reporting system
- Responsive web interface

2 Entity Relationship Diagram (ERD)

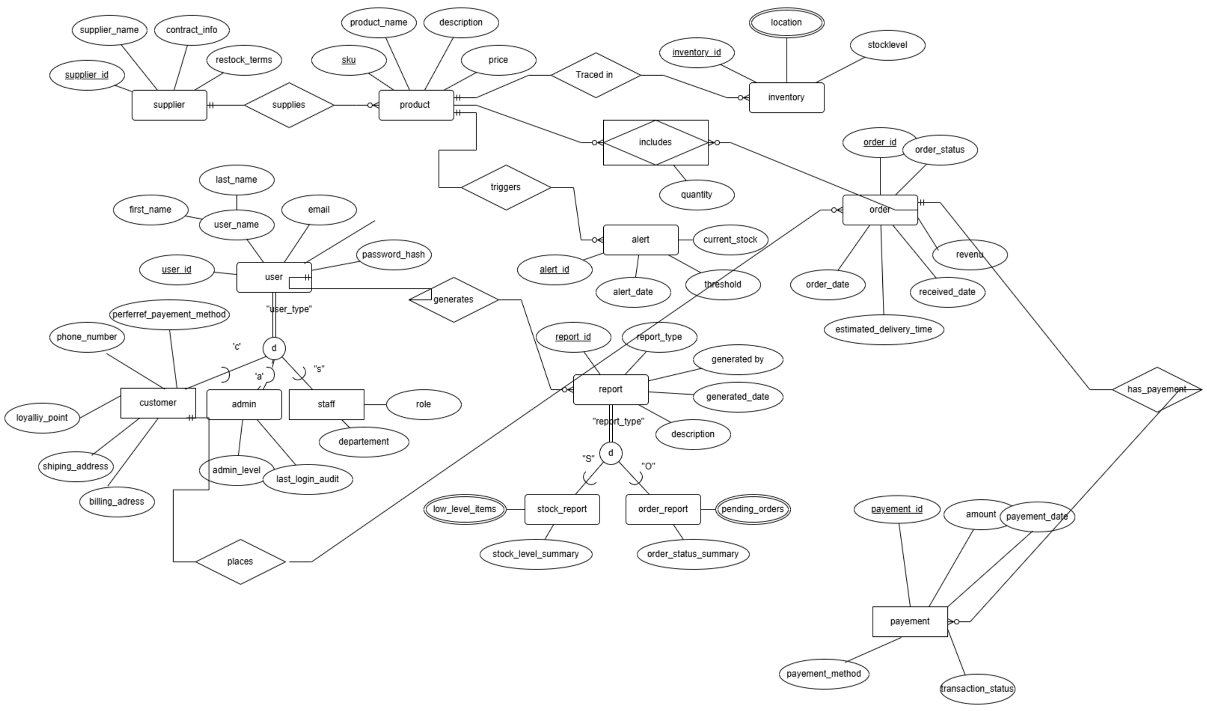
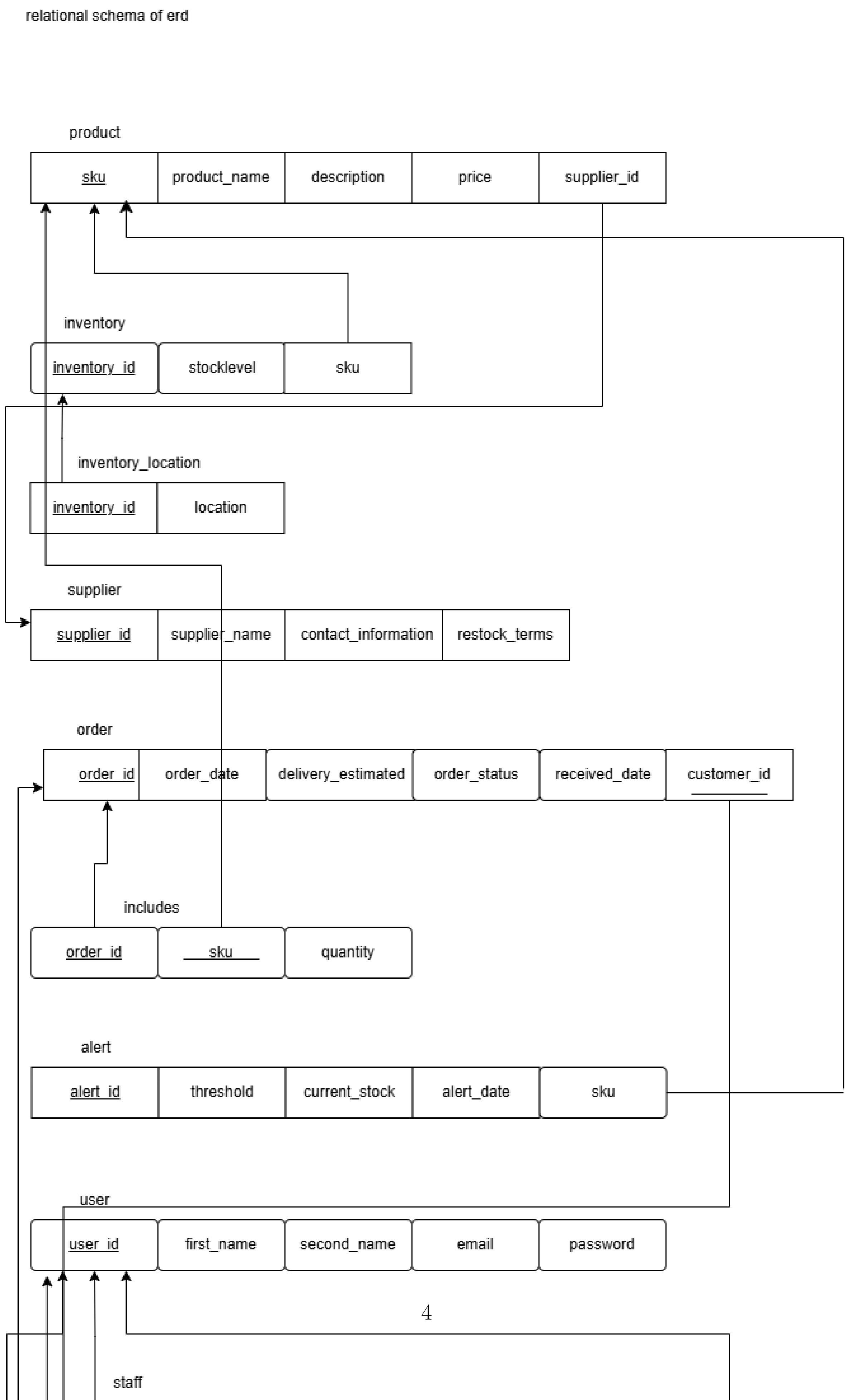


Figure 1: Entity Relationship Diagram for StockDB

The ERD shows the relationships between the core entities in the system:

- Users (with different roles)
- Products and their inventory
- Suppliers and their products
- Orders and order items
- Payments
- Reports and alerts

3 Relational Database Schema



The database consists of the following tables with their relationships:

Table 1: Database Tables

Table	Description
user	Stores user information including authentication details
supplier	Contains supplier information
product	Product catalog with descriptions and pricing
inventory	Tracks stock levels for products
admin	Admin-specific attributes
staff	Staff-specific attributes
customer	Customer-specific attributes
order	Order records
order_item	Items within each order
payment	Payment records
alert	System alerts and notifications
report	Generated reports
stock_report	Stock-specific report details
low_level_items_stock_report	Low stock alerts
order_report	Order-related reports
pending_orders_order_report	Pending order reports

4 Database Implementation

4.1 Complete Schema Definition

```sql
-- Main tables creation
-- Create users table
CREATE TABLE IF NOT EXISTS "user" (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    user_type VARCHAR(20) NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create suppliers table
CREATE TABLE IF NOT EXISTS supplier (
    id SERIAL PRIMARY KEY,
    supplier_name VARCHAR(255) NOT NULL,
    contact_information VARCHAR(255) NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create products table
CREATE TABLE IF NOT EXISTS product (
    sku VARCHAR(20) PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    supplier_id INTEGER REFERENCES supplier(id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create inventory table
CREATE TABLE IF NOT EXISTS inventory (
    inventory_id SERIAL PRIMARY KEY,
    stock_level INTEGER NOT NULL,
    sku VARCHAR(20) REFERENCES product(sku),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create admin table
CREATE TABLE IF NOT EXISTS admin (
    user_id INTEGER PRIMARY KEY REFERENCES "user"(id),
    admin_level VARCHAR(50) NOT NULL,
    last_login_audit TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create staff table
CREATE TABLE IF NOT EXISTS staff (
    user_id INTEGER PRIMARY KEY REFERENCES "user"(id),
    department VARCHAR(100) NOT NULL,
    role VARCHAR(100) NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create customer table
CREATE TABLE IF NOT EXISTS customer (
    user_id INTEGER PRIMARY KEY REFERENCES "user"(id),
    shipping_address TEXT NOT NULL,
    billing_address TEXT NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    loyalty_points INTEGER DEFAULT 0,
    preferred_payment_method VARCHAR(50),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create orders table
CREATE TABLE IF NOT EXISTS "order" (
    id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES customer(user_id),
    order_date TIMESTAMP WITH TIME ZONE NOT NULL,
    delivery_estimated TIMESTAMP WITH TIME ZONE,
    received_date TIMESTAMP WITH TIME ZONE,
    order_status VARCHAR(50) NOT NULL,
    revenue DECIMAL(10,2),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create order items table
CREATE TABLE IF NOT EXISTS order_item (
    order_id INTEGER REFERENCES "order"(id),
    sku VARCHAR(20) REFERENCES product(sku),
    quantity INTEGER NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (order_id, sku)
);

-- Create payments table
CREATE TABLE IF NOT EXISTS payment (
    id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES "order"(id),
    amount DECIMAL(10,2) NOT NULL,
    payment_date TIMESTAMP WITH TIME ZONE NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    transaction_status VARCHAR(50),
    customer_id INTEGER REFERENCES customer(user_id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create alerts table
CREATE TABLE IF NOT EXISTS alert (
    id SERIAL PRIMARY KEY,
    threshold INTEGER NOT NULL,
    current_stock INTEGER NOT NULL,
    alert_date TIMESTAMP WITH TIME ZONE NOT NULL,
    inventory_id INTEGER REFERENCES inventory(inventory_id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create reports table
CREATE TABLE IF NOT EXISTS report (
    id SERIAL PRIMARY KEY,
    report_type VARCHAR(50) NOT NULL,
    generated_date TIMESTAMP WITH TIME ZONE NOT NULL,
    description TEXT,
    user_id INTEGER REFERENCES "user"(id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create stock reports table
CREATE TABLE IF NOT EXISTS stock_report (
    report_id INTEGER PRIMARY KEY REFERENCES report(id),
    stock_level_summary TEXT NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create low level items stock report table
CREATE TABLE IF NOT EXISTS low_level_items_stock_report (
    report_id INTEGER PRIMARY KEY REFERENCES report(id),
    low_level_items TEXT NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create order reports table
CREATE TABLE IF NOT EXISTS order_report (
    report_id INTEGER PRIMARY KEY REFERENCES report(id),
    order_status_summary TEXT NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create pending orders order report table
CREATE TABLE IF NOT EXISTS pending_orders_order_report (
    report_id INTEGER PRIMARY KEY REFERENCES report(id),
    pending_orders TEXT NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

4.2 queries

```sql
-- first query : This query shows which products have been ordered the most by total quantity. It helps identify the most popular products, which can be useful for inventory management and marketing focus.
-- Top 5 Best-Selling Products by Quantity Ordered
SELECT p.product_name, SUM(oi.quantity) AS total_quantity_sold
FROM order_item oi
JOIN Product p ON oi.sku = p.sku
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 5;
-- second query : This shows the total sales revenue generated from each suppliers products. Suppliers with higher revenue might indicate stronger partnerships or more popular product lines.
--Total Revenue Generated by Each Supplier
SELECT s.supplier_name, SUM(o.revenue) AS total_revenue
FROM "Order" o
JOIN Supplier s ON o.supplier_id = s.supplier_id
GROUP BY s.supplier_name
ORDER BY total_revenue DESC;
-- third query: This breaks down how many users belong to each user type (Customer, Staff, Admin). Useful to understand the composition of your user base.
SELECT user_type, COUNT(*) AS count
FROM "User"
GROUP BY user_type;
-- fourth query : Lists all orders that have not been delivered yet, sorted by estimated delivery date. Helps logistics teams prioritize shipments and monitor pending orders.
--Orders Pending Delivery
SELECT
    o.order_id,
    u.first_name,
    u.last_name,
    o.order_date,
    o.delivery_estimated
FROM "Order" o
JOIN Customer c ON o.customer_id = c.user_id
JOIN "User" u ON c.user_id = u.user_id
WHERE o.order_status = 'Pending'
ORDER BY o.delivery_estimated ASC;
-- fifth query : Shows current stock levels of all products and flags
   those with low stock (arbitrarily set here as less than 50 units).
   Useful for restocking decisions.
SELECT p.product_name, i.stock_level,
       CASE
         WHEN i.stock_level < 50 THEN 'Low Stock'
         ELSE 'Stock OK'
       END AS stock_status
FROM Inventory i
JOIN Product p ON i.sku = p.sku
ORDER BY i.stock_level ASC;
```

5 Tools Used

5.1 Backend

- Node.js - JavaScript runtime
- Express.js - Web application framework
- PostgreSQL - Relational database
- JWT - Authentication tokens
- bcrypt - Password hashing

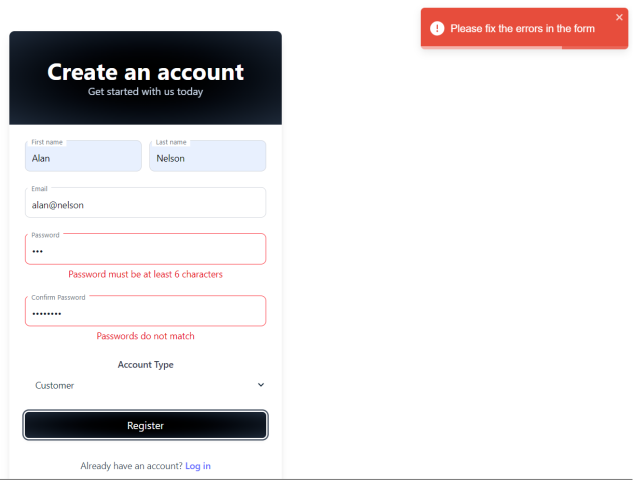
5.2 Frontend

- React.js - JavaScript library for UI
- Material-UI - UI component library
- Redux - State management
- Axios - HTTP client

5.3 Development Tools

- Git - Version control
- VS Code - Integrated development environment
- Postman - API testing
- pgAdmin - Database management

6 GUI Screenshots


Please fix the errors in the form

Create an account

Get started with us today

First name: Alan

Last name: Nelson

Email: alan@nelson

Password: •••

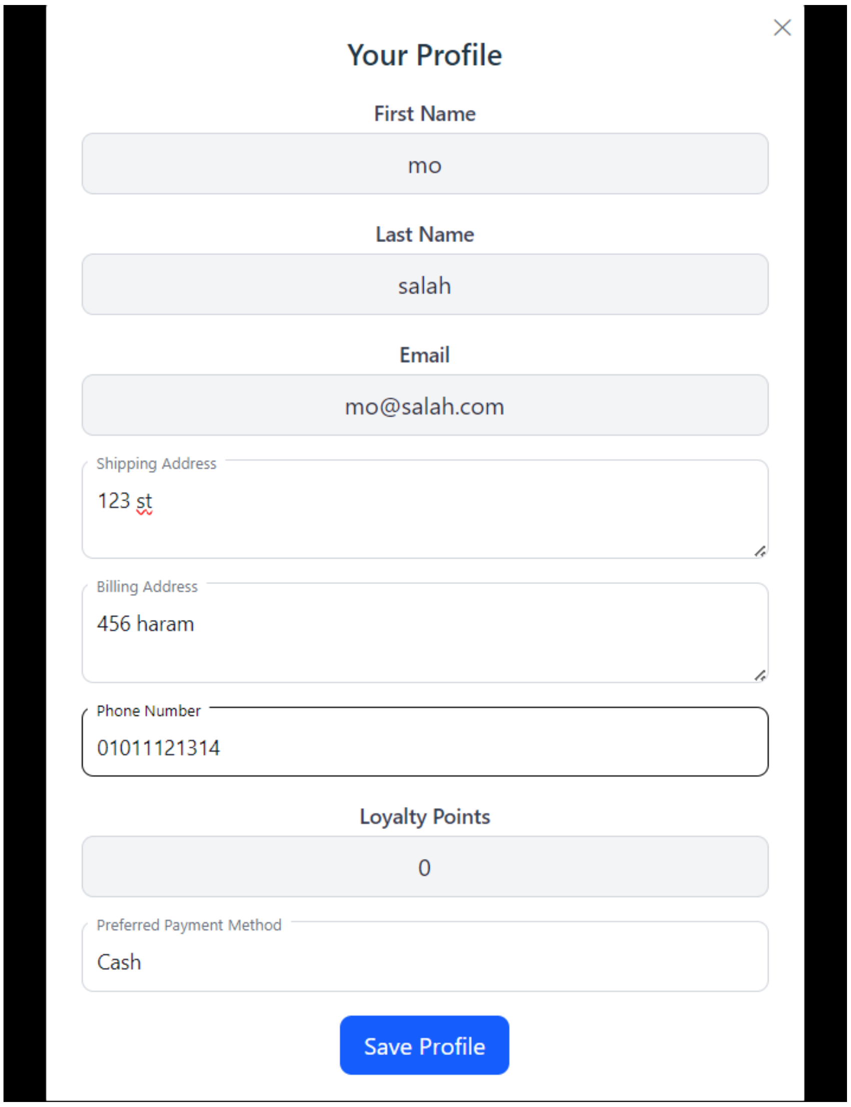
Password must be at least 6 characters

Confirm Password: ••••••••

Passwords do not match

Account Type: Customer

Figure 3: User registration interface with form validation, showing: (1) required fields for account creation (name, email, password), (2) password complexity enforcement (minimum 6 characters), and (3) password confirmation matching validation



Your Profile

First Name: mo

Last Name: salah

Email: mo@salah.com

Shipping Address: 123 st

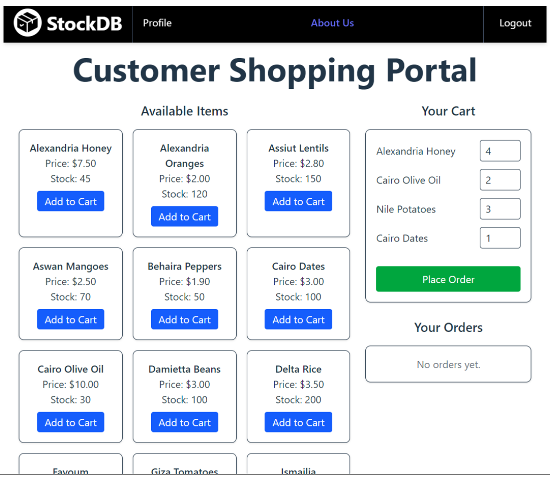
Billing Address: 456 haram

Phone Number: 01011121314

Loyalty Points: 0

Preferred Payment Method: Cash

Figure 4: Customer profile management interface showing personal information, contact details, shipping/billing addresses, and loyalty program status



Customer Shopping Portal

Available Items

Alexandria Honey
Price: $7.50
Stock: 45

Alexandria Oranges
Price: $2.00
Stock: 120

Assiut Lentils
Price: $2.80
Stock: 150

Aswan Mangoes
Price: $2.50
Stock: 70

Behaira Peppers
Price: $1.90
Stock: 50

Cairo Dates
Price: $3.00
Stock: 100

Cairo Olive Oil
Price: $10.00
Stock: 30

Damietta Beans
Price: $3.00
Stock: 100

Delta Rice
Price: $3.50
Stock: 200

Fayoum

Giza Tomatoes

Ismailia

Your Cart

Alexandria Honey	4
Cairo Olive Oil	2
Nile Potatoes	3
Cairo Dates	1

Your Orders

No orders yet.

Figure 5: Process workflow diagram showing the order fulfillment lifecycle from order creation to delivery and payment processing

7 Conclusion

StockDB provides a comprehensive solution for inventory management with:

- Secure role-based access
- Real-time inventory tracking
- Order and payment processing
- Comprehensive reporting
- Intuitive user interface

The system is built with modern technologies following best practices in database design and application architecture.