Database Systems

CS06301/ CS-1117

Project Documentation

Grocery Store Management System

Instructor

Sir Babar Imran

Submitted by

Hafiz Muhammad Haris

{70143269}

Department of Software Engineering

The University of Lahore

Group:
Individual Project

Submission Date:14-01-2026

1. Introduction

The Grocery Store Management System is a web-based application designed to digitize and centralize the operations of a small-to-medium grocery store. The system replaces manual record keeping and spreadsheet-based management with a structured relational database.

The system manages products, categories, inventory, suppliers, customers, orders, and payments. It enforces database constraints, maintains data integrity, and provides structured order and inventory tracking through a three-tier architecture consisting of a React frontend, Express/Node.js backend, and a MySQL database.

2. System Users

• Admin: Full access

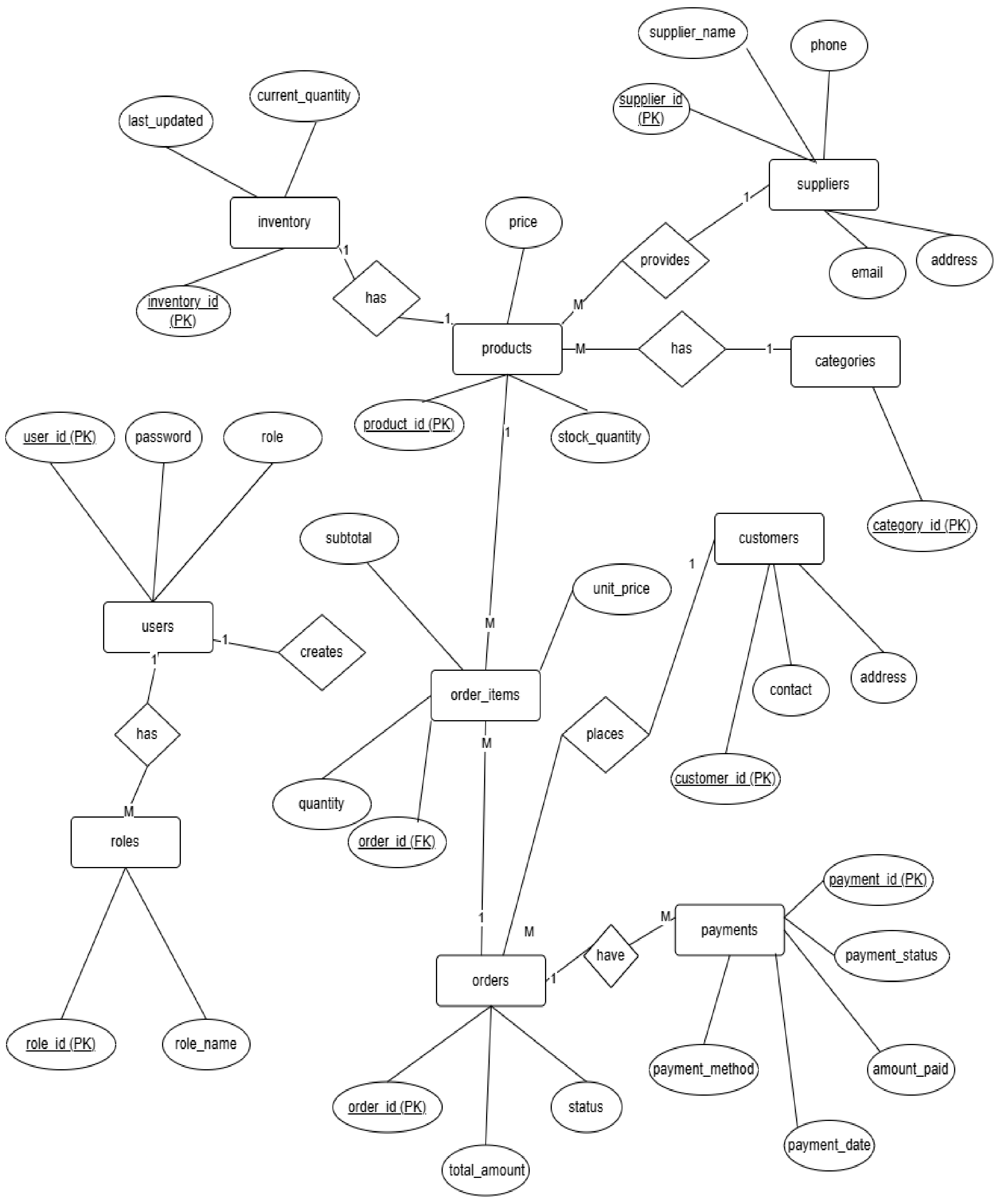
Manage products, categories, view orders, manage inventory, run reports.

• Store Staff: Create and view orders, search products, update order status; limited product-edit permission.

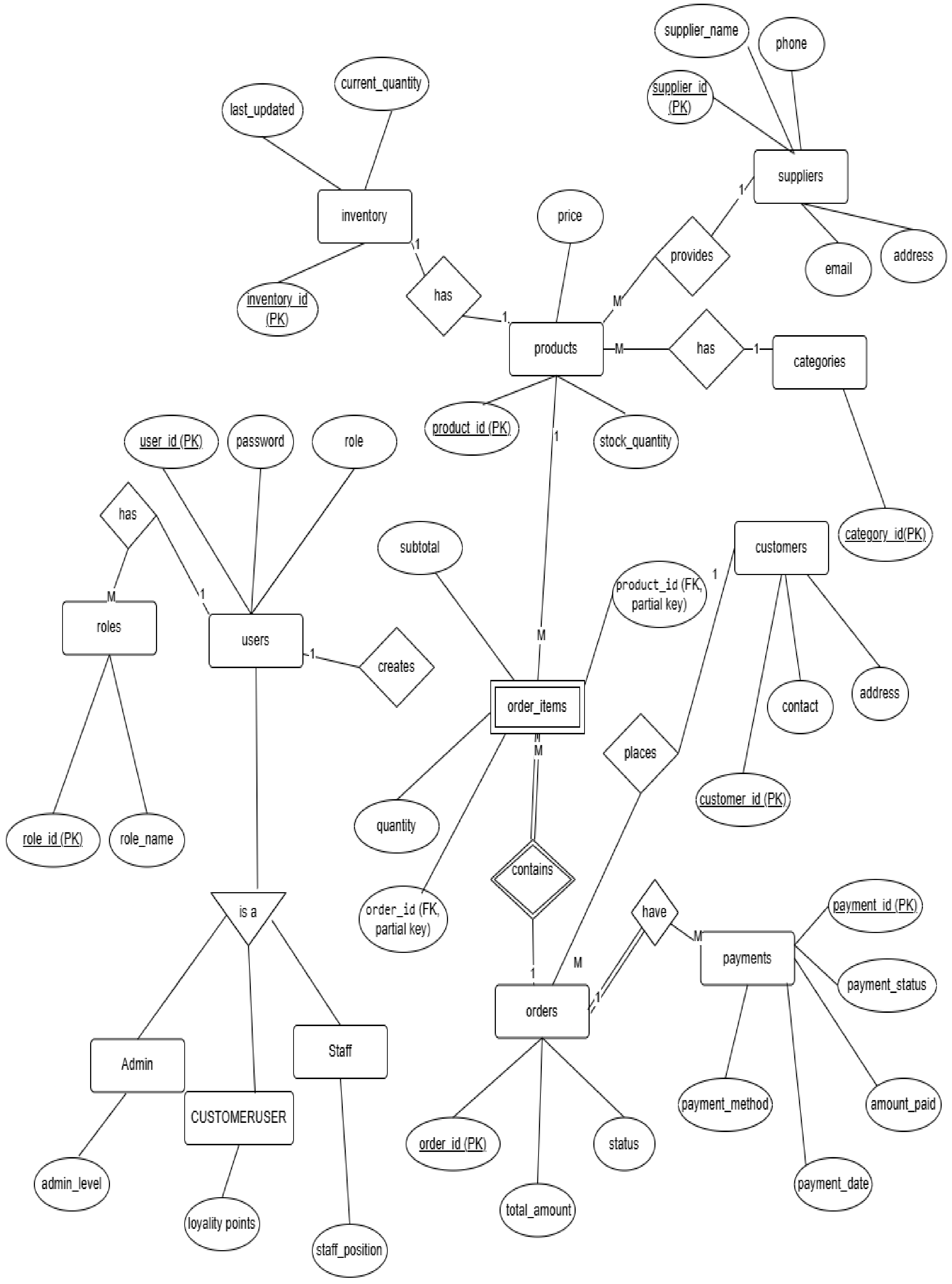
• Database / Maintenance Role (DBA): Load sample data, run backups, execute SQL scripts.

3. Database Design

Entity Relationship Diagram (ERD)



Enhanced Entity Relationship Diagram (EERD)



Entity Descriptions

Entity: Users

Description: Stores login and role-based access information.
Attributes:

- user_id (PK)
- password
- role

Entity: Roles

Description: Defines system roles.
Attributes:

- role_id (PK)
- role_name

Entity: Admin

Description: Specialized user with administrative privileges.
Attributes:

- admin_level

Entity: Staff

Description: Specialized user responsible for store operations.
Attributes:

- staff_position

Entity: CustomerUser

Description: Specialized user with purchasing capability.
Attributes:

- loyalty_points

Entity: Categories

Description: Groups products by type.
Attributes:

- category_id (PK)

Entity: Products

Description: Stores product details.
Attributes:

- product_id (PK)
- price
- stock_quantity

Entity: Inventory

Description: Tracks current stock state of products.
Attributes:

- inventory_id (PK)
- current_quantity
- last_updated

Entity: Suppliers

Description: Stores supplier details.
Attributes:

- supplier_id (PK)
- supplier_name
- phone
- email
- address

Entity: Customers

Description: Stores customer profile information.
Attributes:

- customer_id (PK)
- contact
- address

Entity: Orders

Description: Represents customer orders.
Attributes:

- order_id (PK)
- total_amount
- status

Entity: Order_Items

Description: Weak entity storing products within orders.
Attributes:

- order_id (FK, partial key)
- product_id (FK, partial key)
- quantity

- subtotal

Entity: Payments

Description: Stores payment details for orders.
Attributes:

- payment_id (PK)
- payment_method
- payment_date
- payment_status
- amount_paid

ERD Relationships

- User **has** Role (M:1)
- User **is a** Admin / Staff / CustomerUser (Specialization)
- Product **belongs to** Category (M:1)
- Supplier **provides** Product (1:M)
- Inventory **has** Product (1:1)
- Customer **places** Order (1:M)
- Order **contains** Order_Items (1:M)
- Product **appears in** Order_Items (1:M)
- Order **has** Payment (1:M)

4. Database Schema Design

Table: roles

• role_id INT AUTO_INCREMENT PRIMARY KEY
• role_name VARCHAR(50) NOT NULL UNIQUE

Table: users

• user_id INT AUTO_INCREMENT PRIMARY KEY
• password VARCHAR(255) NOT NULL
• role_id INT NOT NULL
• FOREIGN KEY (role_id) REFERENCES roles(role_id) ON DELETE RESTRICT

Table: admin

• user_id INT PRIMARY KEY
• admin_level VARCHAR(50)
• FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE

Table: staff

• user_id INT PRIMARY KEY
• staff_position VARCHAR(50)
• FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE

Table: dba

• user_id INT PRIMARY KEY
• access_level VARCHAR(50) DEFAULT 'Full'
• FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE

Table: categories

• category_id INT AUTO_INCREMENT PRIMARY KEY
• category_name VARCHAR(100) NOT NULL UNIQUE

Table: suppliers

• supplier_id INT AUTO_INCREMENT PRIMARY KEY
• supplier_name VARCHAR(100) NOT NULL
• phone VARCHAR(20)
• email VARCHAR(100)
• address VARCHAR(255)

Table: products

• product_id INT AUTO_INCREMENT PRIMARY KEY
• product_name VARCHAR(150) NOT NULL
• category_id INT NOT NULL
• supplier_id INT NOT NULL
• price DECIMAL(10,2) NOT NULL CHECK (price >= 0)
• stock_quantity INT DEFAULT 0 CHECK (stock_quantity >= 0)
• description TEXT
• FOREIGN KEY (category_id) REFERENCES categories(category_id) ON DELETE CASCADE
• FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id) ON DELETE CASCADE

Table: inventory

• inventory_id INT AUTO_INCREMENT PRIMARY KEY
• product_id INT NOT NULL UNIQUE
• current_quantity INT NOT NULL CHECK (current_quantity >= 0)
• last_updated DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
• FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE

Table: customers

• customer_id INT AUTO_INCREMENT PRIMARY KEY
• customer_name VARCHAR(100) NOT NULL
• contact VARCHAR(20)
• address VARCHAR(255)

Table: orders

• order_id INT AUTO_INCREMENT PRIMARY KEY
• customer_id INT NOT NULL
• total_amount DECIMAL(10,2) DEFAULT 0.00
• order_date DATETIME DEFAULT CURRENT_TIMESTAMP
• FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE

Table: order_items

• order_id INT NOT NULL
• product_id INT NOT NULL
• quantity INT NOT NULL CHECK (quantity > 0)
• subtotal DECIMAL(10,2) NOT NULL CHECK (subtotal >= 0)
• PRIMARY KEY (order_id, product_id)
• FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE
• FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE

Table: payments

• payment_id INT AUTO_INCREMENT PRIMARY KEY
• order_id INT NOT NULL
• payment_method VARCHAR(50) NOT NULL
• payment_date DATE NOT NULL
• payment_status VARCHAR(50) DEFAULT 'Completed'
• amount_paid DECIMAL(10,2) NOT NULL
• FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE

5. SQL Implementation

Sql Scripts file uploaded separately along with this documentation pdf.

6. Frontend Interface (Basic)

Purpose

The frontend provides interfaces to insert, view, and manage data, and demonstrate database usage.

Frontend Technology

- React.js
- HTML / CSS
- Node.js (Express)
- MySQL Connector

Frontend Snippets

- Product list view
- Add/Edit product form
- Order creation page
- Orders listing page

Code Base:

7. Conclusion

This project successfully implements a normalized grocery store database system with a complete web-based interface. It ensures data consistency, supports core store operations, and provides a scalable foundation for future enhancements such as analytics and advanced reporting.