San Jose State University

Computer Science Department

CS 157A Fall 2024

Project Proposal

Project - Grocery Store Inventory Manager

Submitted By:

Alejandro Garcia (015143349, alejandro.a.garcia@sjsu.edu)

Arpita Misal (015385565, arpita.misal@sjsu.edu)

Bhavya Preethika Vatsavayi (016142009, bhavyapreethika.vatsavayi@sjsu.edu)

Jasleen Narang (016466970, jasleen.narang@sjsu.edu)

Blake Mongtomery (016565991, blake.montgomery@sjsu.edu)

Instructor:

Ramin Moazeni

19 September 2024

Proposal

Project Overview:

Our team is developing a web-based grocery store inventory management application that will use MySQL as the database to manage all the data. It focuses on efficient tracking of products, inventory levels, customer orders, and supplier relationships, ensuring real-time updates and decision-making support for grocery store staff and management.
The system will manage important data including **Products**, **Categories**, **Suppliers**, **Customers**, **Orders**, **Order Items**, **Inventory**, **Employees**, **Discounts**, and **Store Locations**. The application will allow grocery store employees to perform crucial tasks such as covering the full scope of inventory control, product management, customer order processing, employee management, and handling discounts and promotions.

Grocery Store database entities:

1. **Product**: Stores information about products (product_id, product_name, category_id, price, stock_quantity, expiration_date).
2. **Category**: Organizes products into categories (category_id, category_name, description).
3. **Supplier**: Tracks suppliers for products (supplier_id, supplier_name, contact_info, address).
4. **Customer**: Stores customer information (customer_id, first_name, last_name, email, phone, loyalty_points).
5. **Order**: Logs customer orders (order_id, customer_id, order_date, total_amount, order_status).
6. **Order_Item**: Contains individual items within an order (order_item_id, order_id, product_id, quantity, price).
7. **Inventory**: Tracks inventory levels for products (inventory_id, product_id, stock_level, reorder_level).
8. **Employee**: Stores information about store employees (employee_id, first_name, last_name, position, salary, hire_date).
9. **Discount**: Applies discounts to products or orders (discount_id, product_id, discount_percentage, start_date, end_date).
10. **Store_Location**: Manages multiple store locations (location_id, address, city, state, zip_code, store_manager_id).

Functional Requirements:

1. **Product Management**: The system should allow staff to add, update, delete, and view products.
2. **Inventory Management**: The system should track stock levels and notify staff when stock reaches a reorder level.
3. **Order Management**: The system should allow customers to place orders and track order status.
4. **Customer Management**: The system should store and manage customer information, including loyalty points and order history.
5. **Supplier Management**: The system should track supplier details and allow ordering/restocking from suppliers.
6. **Employee Management**: The system should manage employee records, roles, and permissions.
7. **Payment Processing**: The system should support multiple payment methods (cash, credit card, etc.) and generate receipts.
8. **Discounts and Promotions**: The system should apply discounts automatically based on predefined rules (e.g., coupons, special offers).
9. **Report Generation**: The system should generate sales, inventory, and order reports for management to review.
10. **Store Location Management**: The system should allow tracking of multiple store locations and their respective inventory levels.

Non-Functional Requirements:

1. **Data Storage Capacity**: The DBMS can support up to 1 TB of data storage for product inventories, customer records, orders, and transaction history.
2. **Query Performance**: The DBMS should handle complex queries (e.g., sales reports, inventory restocking levels).
3. **Transaction Processing**: The DBMS must support ACID properties to ensure data integrity during simultaneous transactions like product purchases and inventory updates.
4. **Backup and Recovery**: The DBMS can support automatic backups daily and have a recovery point. Full backups should occur weekly, with incremental backups happening hourly.
5. **Data Consistency**: The DBMS must ensure that inventory data is always accurate across multiple locations so that if an item is sold in one store, it reflects across the system immediately.
6. **Concurrency Control**: The DBMS must handle concurrent transactions from multiple users without locking issues or data corruption, particularly for inventory updates and customer orders.
7. **Scalability**: The DBMS should be able to scale vertically or horizontally to support growth in data (up to 10 million records) and user access
8. **Security**: The DBMS has role-based access control, with separate permissions for customers, employees, and administrators.