Quick Mart System

self_checkout supermarket

Contents

1.Feasibility Study

1. Executive Summary

This feasibility study evaluates the viability of developing a mobile self-checkout application that enables supermarket customers to scan items and pay via mobile money (M-Pesa). The goal is to reduce checkout queue times and enhance the shopping experience. The study examines technical, economic, operational, legal, and schedule feasibility aspects to determine if this solution can be successfully implemented.

---

2. Description of the Product/Service

The proposed product is a mobile app that allows users to scan products in a supermarket using their smartphones and make secure payments through the M-Pesa mobile payment platform. This app reduces reliance on traditional cashier checkouts, helping supermarkets improve customer flow and reduce staffing needs during peak hours.

---

3. Technology Considerations

The app is developed using Java and integrates Firebase Authentication to securely manage users. Payment integration is accomplished via the Daraja M-Pesa API, which enables real-time transaction processing. The system relies on reliable mobile internet connectivity and smartphone capabilities like barcode scanning, which are widely available in the target market.

---

4. Market Analysis

The target market consists primarily of urban supermarkets where customer wait times at checkout are high. Mobile money usage, especially M-Pesa, is common in the region, ensuring familiarity and trust in the payment method. Competitors include traditional cashier systems and other self-checkout solutions, but few provide seamless mobile payment integration. The increasing smartphone penetration supports strong adoption potential.

---

5. Marketing Strategy

Marketing efforts will focus on partnerships with supermarket chains, demonstrations of time-saving benefits, and targeted campaigns to tech-savvy shoppers. The app will differentiate itself by offering a simple user interface combined with secure and fast mobile payments, catering specifically to regions with high M-Pesa usage.

6. Organization and Staffing

A small development and support team will be required initially, including mobile developers, backend engineers, and customer service representatives. Training will be provided to supermarket staff to assist customers in the initial adoption phase. Ongoing maintenance and updates will be managed by the core team.

---

7. Financial Projections

Initial development costs are estimated at $100,000, covering app development and backend integration. Ongoing annual costs for hosting, support, and marketing are projected at $50,000. Revenue is expected from subscription fees charged to supermarkets and transaction fees on payments processed. Profitability is expected within 3 years as adoption scales.

---

8. Schedule Feasibility

The project development timeline is estimated at 6 months for the initial launch, including design, development, and pilot testing. Full rollout to multiple supermarkets could occur within 18 months. The schedule is realistic given the project scope and resource availability.

---

9. Legal Feasibility

Compliance with data protection laws and mobile payment regulations will be strictly maintained. Contracts with payment providers and supermarkets will define roles and responsibilities, ensuring secure handling of customer data and transactions.

2.System Requirements Specification (SRS)

2.1 Functional Requirements

These are the core functions your self-checkout system must perform:

- Item Scanning: The system shall allow users to scan product barcodes to add items to their virtual cart.
- Quantity Adjustment: Users shall be able to increase or decrease the quantity of each scanned item.
- Price Calculation: The system shall calculate and display the total price, including discounts and taxes, in real-time.
- Payment Processing: The system shall support multiple payment methods (e.g., credit/debit card, mobile payment, cash).
- Receipt Generation: After payment, the system shall generate a digital or printed receipt for the customer.
- Cart Management: Users shall be able to remove items from the cart before checkout.
- Security Measures: The system shall verify payment and prevent fraud during checkout.
- Admin Functions: The system shall allow store administrators to update product prices, manage inventory, and view sales reports via a backend portal.

2.2 Non-Functional Requirements

These describe the system's qualities and constraints:

- Performance: The system shall respond to scanning actions within 1 second.

- Availability: The system shall be available for use 24/7 during store operating hours with 99.9% uptime.

- Usability: The user interface shall be intuitive and easy to use for customers of all ages.

- Security: Sensitive data such as payment information must be encrypted.

- Scalability: The system shall handle simultaneous checkouts of up to 100 users during peak hours.

- Reliability: The system shall prevent transaction loss even in case of network interruptions by queuing transactions locally.

- Compatibility: The mobile app shall support both Android and iOS devices.

- Maintainability: The system shall allow easy updates and bug fixes with minimal downtime.

2.3 User Requirements

This section outlines what the users expect from the system:

- Customers want a fast, seamless checkout experience without waiting in long lines.

- Customers want clear instructions and feedback during the scanning and payment process.

- Customers expect secure handling of their payment information.

- Store employees want to quickly manage inventory and monitor system status.

- Store managers want access to sales analytics and reports for business decisions.

2.4 System Constraints

These are the limitations or restrictions on the system:

- The system must operate reliably in the store’s existing network environment with limited bandwidth.

- The self-checkout app must run on smartphones with minimum Android 8.0 or iOS 12.

- Payment gateways used must be compliant with local regulations and support multiple currencies.

- The system must integrate with the store’s existing inventory database.

- Hardware used for scanning must be compatible with the system software.

- Privacy laws require that customer data be stored only for a limited time and protected from unauthorized access.

3.Use Case Modeling

3. Use Case Diagrams

- **What it is:**

Shows the different interactions between users (like customers, admins) and the system.

- **Importance:**

Helps define system requirements clearly and understand user needs.

3.1 Use Cases:

1. Use Case: Scan Item

- **Actor**: Customer
- **Description**: The customer scans a product's barcode using the camera or scanner to add it to the cart.
- **Precondition**: Customer is logged in or has started a session.
- **Postcondition**: The item appears in the cart with default quantity = 1.
- **Main Flow**:
    1. Customer selects "Scan Item".
    2. System opens the camera/scanner.
    3. Barcode is recognized.
    4. Items are fetched from the database and displayed in the cart.
- **Use Case Scenario:**

A customer launches the app and selects "Scan Item." The camera opens, and they scan a shampoo bottle's barcode. The system fetches the product details and adds it to the cart with quantity set to 1.

---

2. Use Case: Enter Quantity

- **Actor**: Customer
- **Description**: Customer selects an item and updates the desired quantity.
- **Precondition**: Item has been added to the cart.
- **Postcondition**: The item's quantity is updated and total recalculated.
- **Main Flow**:
    1. Customer taps on an item in the cart.
    2. Enter the quantity.
    3. System recalculates the price and updates the cart.
- **Use Case Scenario:**

After scanning a chocolate bar, the customer taps on it in the cart and changes the quantity to 3. The system updates the total cost accordingly.

---

3. Use Case: View Cart

- **Actor**: Customer
- **Description**: View all items scanned with quantities and prices.
- **Precondition**: At least one item is scanned.
- **Postcondition**: Cart is shown with updated list and totals.
- **Main Flow**:
    1. Customer clicks "Cart".
    2. System shows a list of all items, prices, and totals.

- **Use Case Scenario:**

The customer clicks the cart icon and sees a list of items including a soda and bread, with prices and quantities, and a subtotal at the bottom.

---

4. Use Case: Apply Discount

- **Actor**: Customer
- **Description**: Apply promo code, loyalty points, or vouchers.
- **Precondition**: Customer has code or eligible loyalty points.
- **Postcondition**: Total amount is reduced accordingly.
- **Main Flow**:
    1. Customer clicks "Apply Discount".
    2. Enter code or select loyalty option.
    3. System validates and applies discounts.

- **Use Case Scenario:**

The customer has a 10% discount code. They tap “Apply Discount,” enter the code, and the system updates the total amount after applying the discount.

---

5. Use Case: Remove Item

- **Actor**: Customer
- **Description**: Remove unwanted items from the cart.
- **Precondition**: Item exists in cart.
- **Postcondition**: Item is deleted and totally recalculated.
- **Main Flow**:
    1. Customer selects the item to remove.
    2. Confirms removal.
    3. System updates cart.

- **Use Case Scenario:**

The customer changes their mind about buying milk. They select it in the cart and confirm to remove it. The cart updates without that item.

---

6. Use Case: Proceed to Payment

- **Actor**: Customer
- **Description**: Begin the payment process.
- **Precondition**: Cart has items.
- **Postcondition**: Customer moves to payment screen.
- **Main Flow**:
    1. Customer taps "Checkout".
    2. System displays payment options.
- **Use Case Scenario:**

The customer taps "Checkout" after reviewing their cart. The system shows available payment methods like card, wallet, or QR.

---

7. Use Case: Make Payment

- **Actor**: Customer
- **Description**: Finalize the purchase via card, digital wallet, or QR payment.
- **Precondition**: Customer has chosen payment method.
- **Postcondition**: Payment is confirmed and logged.
- **Main Flow**:
    1. Customer selects payment type.
    2. Enters details or scans QR.
    3. System processes payment.

- **Use Case Scenario:**

The customer chooses to pay via QR code. They scan it with their banking app, approve the transaction, and payment is processed successfully.

---

8. Use Case: Generate Bill

- **Actor**: System
- **Description**: System issues a receipt after successful payment.
- **Precondition**: Payment is confirmed.
- **Postcondition**: Customer receives receipt (printed, emailed, or in-app).
- **Main Flow**:
    1. System compiles purchase info.
    2. Generates and displays/sends receipts.

- **Use Case Scenario:**

After payment, the system shows a digital receipt and also sends a copy to the customer's email.

---

9. Use Case: Monitor Transactions

- **Actor**: Admin
- **Description**: Admin monitors all sales in real-time.
- **Precondition**: Admin is authenticated.
- **Postcondition**: Admin sees transaction logs and patterns.
- **Main Flow**:
    1. Admin logs in.
    2. System displays a sales dashboard.

- **Use Case Scenario:**

An admin logs into the web portal and views a live dashboard showing hourly sales, peak periods, and popular items.

---

10. Use Case: Manage Inventory

- **Actor**: Admin
- **Description**: Admin updates stock levels, prices, and product info.
- **Precondition**: Admin has access rights.
- **Postcondition**: Inventory is updated in the database.
- **Main Flow**:
    1. Admin navigates to inventory.
    2. Edits item info or adds/removes products.

- **Use Case Scenario:**

The admin notices the price of sugar has changed. They log in, update the price from $2.00 to $2.50, and the system reflects the change.

---

11. Use Case: Handle Refunds

- **Actor**: Admin
- **Description**: Process customer refund or billing corrections.
- **Precondition**: Admin is logged in, customer presents transaction proof.
- **Postcondition**: Refund is approved and processed.
- **Main Flow**:
    1. Admin verifies transactions.
    2. Approves or declines refund.
    3. System updates records and adjusts inventory.

- **Use Case Scenario:**

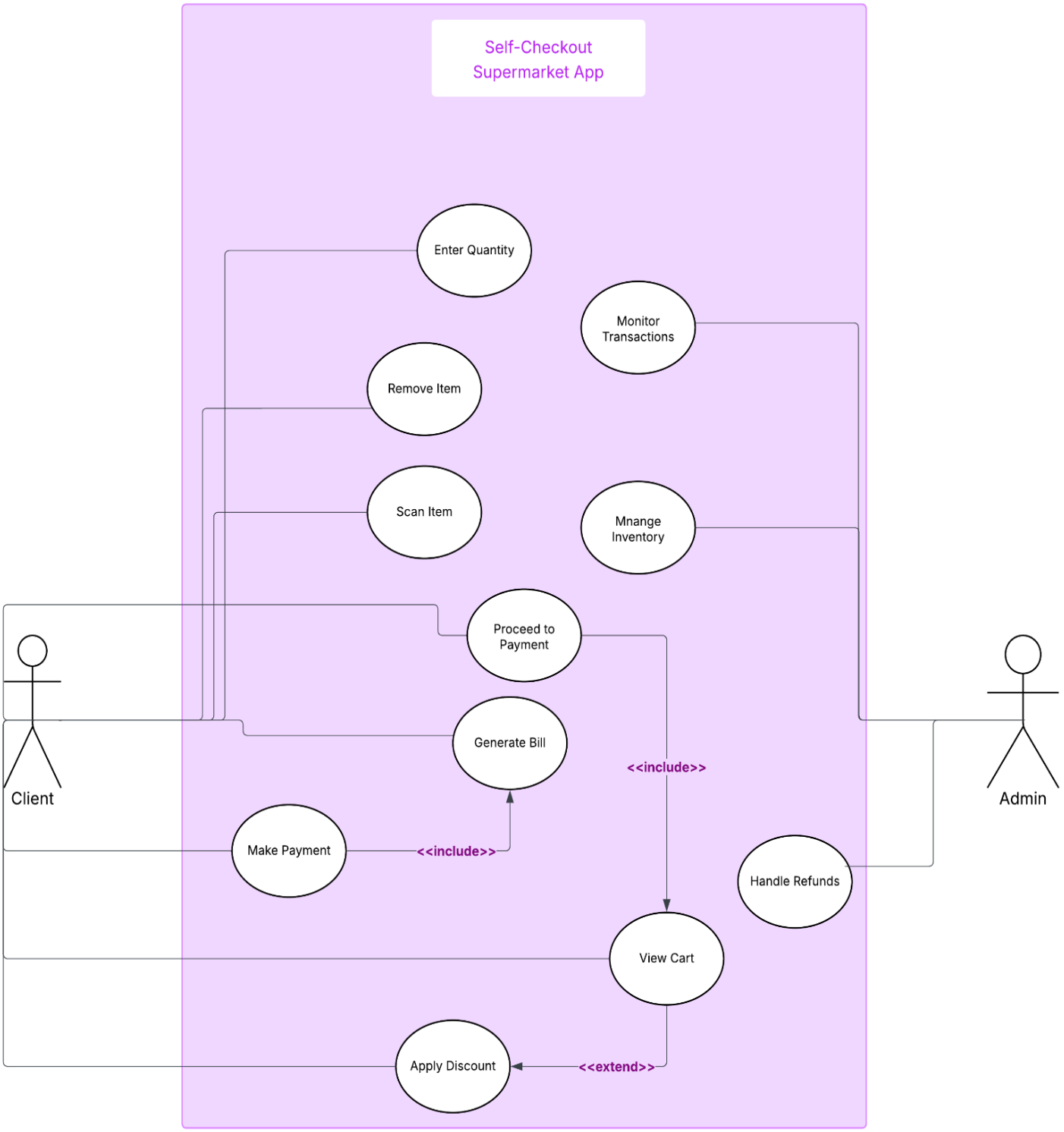
A customer returns with a faulty product and shows the receipt. The admin checks the transaction, approves the refund, and updates the inventory.

3.2 Use Case Table

Use Case	Actor	Description	Precondition	Postcondition	Main Flow
Scan Item	**Customer**	**Scans a product's barcode to add it to cart**	**Customer has started a session**	**Item is added to cart with quantity = 1**	**1. Select scan → 2. Open scanner → 3. Scan barcode → 4. Display in cart**
Enter Quantity	**Customer**	**Updates quantity of an item in the cart**	**Item is in the cart**	**Quantity and total updated**	**1. Select item → 2. Enter quantity → 3. Update total**
View Cart	**Customer**	**Displays scanned items with price and quantity**	**At least one item scanned**	**Cart is shown with totals**	**1. Click "Cart" → 2. System displays items & prices**

Apply Discount	**Customer**	**Applies voucher, promo code, or loyalty points**	**Code or points available**	**Discount applied to total**	**1. Select apply discount → 2. Enter/select code → 3. Apply**
Remove Item	**Customer**	**Removes item from the cart**	**Item exists in cart**	**Item removed and total updated**	**1. Select item → 2. Confirm remove → 3. Update cart**
Proceed to Payment	**Customer**	**Begins the checkout process**	**Cart is not empty**	**Payment screen is shown**	**1. Tap "Checkout" → 2. System shows payment options**
Make Payment	**Customer**	**Completes payment process**	**Payment method selected**	**Payment successful and logged**	**1. Select payment → 2. Provide info → 3. Confirm payment**
Generate Bill	**System**	**Issues digital or printed receipt**	**Payment is successful**	**Receipt generated and delivered**	**1. Compile purchase info → 2. Send or**

					show receipt
Monitor Transactions	Admin	Views real-time sales data	Admin is logged in	Dashboard shows transaction data	1. Admin logs in → 2. View dashboard
Manage Inventory	Admin	Updates product stock and information	Admin has access rights	Inventory updated in system	1. Access inventory → 2. Edit or update product
Handle Refunds	Admin	Processes customer refund requests	Admin is logged in, proof provided	Refund approved and processed	1. Verify transaction → 2. Approve refund → 3. Update system


Self-Checkout
Supermarket App
Enter Quantity
Monitor
Transactions
Remove Item
Scan Item
Mnange
Inventory
Proceed to
Payment
Client
Generate Bill
<<include>>
Admin
Make Payment
<<include>>
Handle Refunds
View Cart
Apply Discount
<<extend>>

4.system design

1. Data Flow Diagrams (DFD)

- **What it is:**

Shows how data moves through the system.

- **Importance:**

Helps identify key processes (like customer orders or inventory updates) and how information flows between them.

2. Entity-Relationship Diagram (ERD)

- **What it is:**

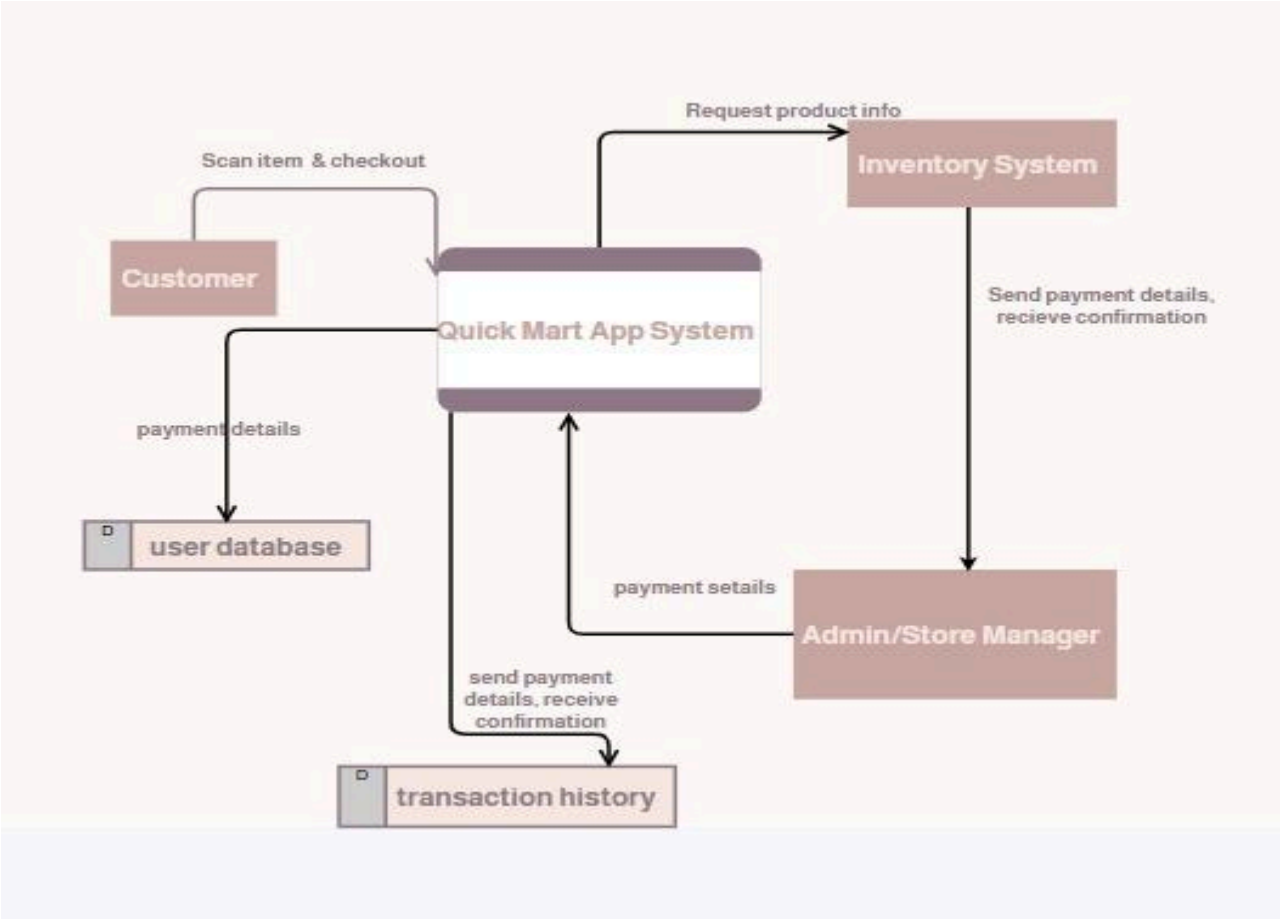
Visualizes the database structure.

- **Importance:**

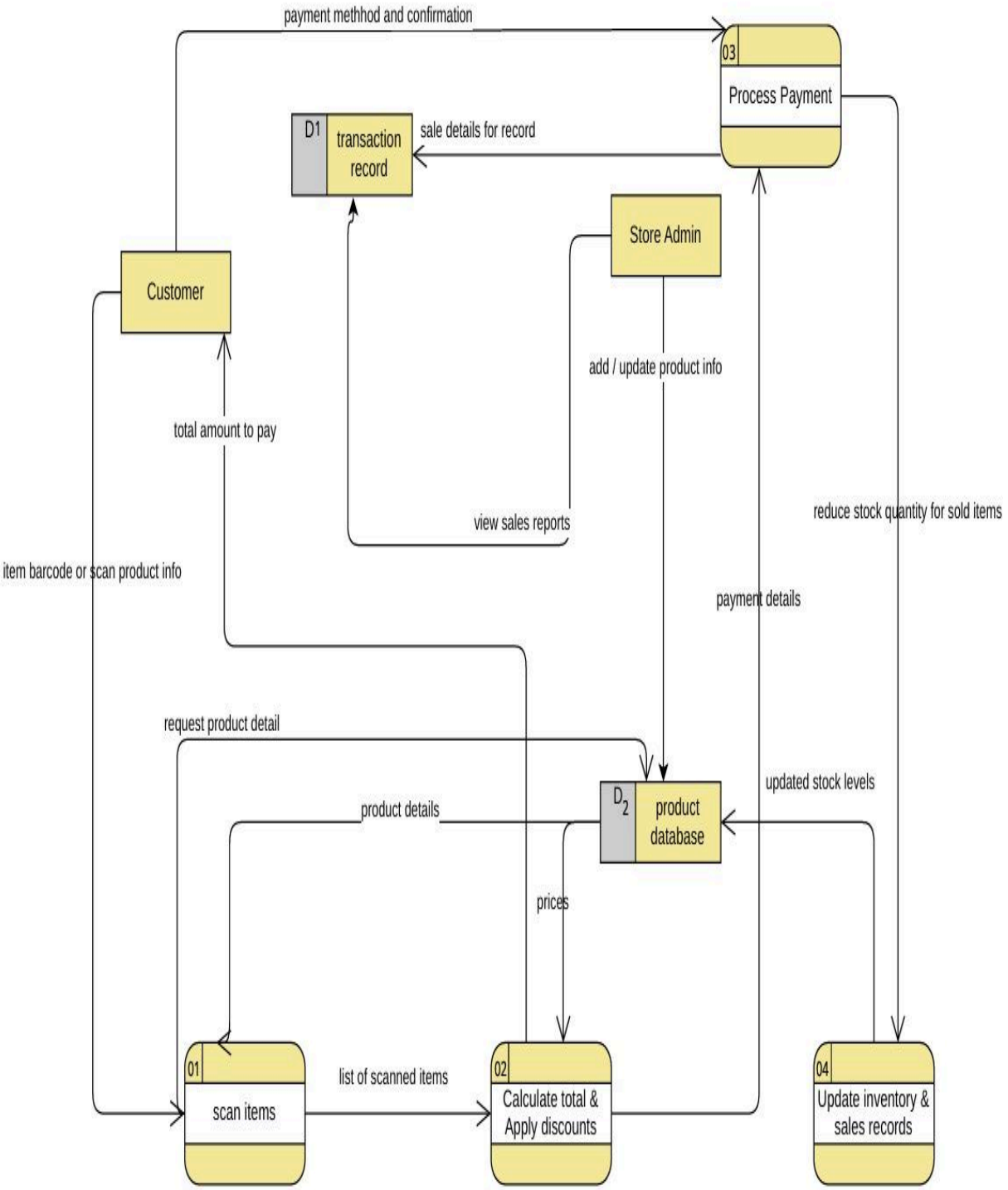
Helps you design the database of Quick Mart—like managing products, customers, orders, and payments.

4.1Data Flow Diagram:

4.1.1 Context Data Flow:



4.1.2 Level1 Data Flow:



4.2 Entity Relationship Diagram(ERD):

❖ Entities and Their Attributes

1. DEPARTMENT

- Department_id
- Department_name
- Manager_id
- Manager_start_date

2. EMPLOYEE

- id
- First_name
- Middle_name
- Last_name
- Name
- salary
- phone_number
- address
- Perks
- End_date
- Join_date

3. LOGIN

- id
- username
- Password

4. PROMOTION

- Promo_code
- Discount
- Valid_upto

5. SUPPLIER

- Sid
- Sname
- Sdealer
- Semail
- Saddress
- Sphone

6. PRODUCT

- Product_id
- Product_name
- Product_type
- Supplier

- Quantity
- Cost_price
- Market_price
- Profit

7. CUSTOMER

- Cid
- Cname
- first_name
- last_name
- Cphone
- Caddress
- Cmoney_spent
- Cmoney_spent_res
- AtCjoin_date_birthdate
- Count

8. BUY (Associative Entity)

- Purchase_id
- Purchase_date
- Total_amount

❖ The relationship between entities is

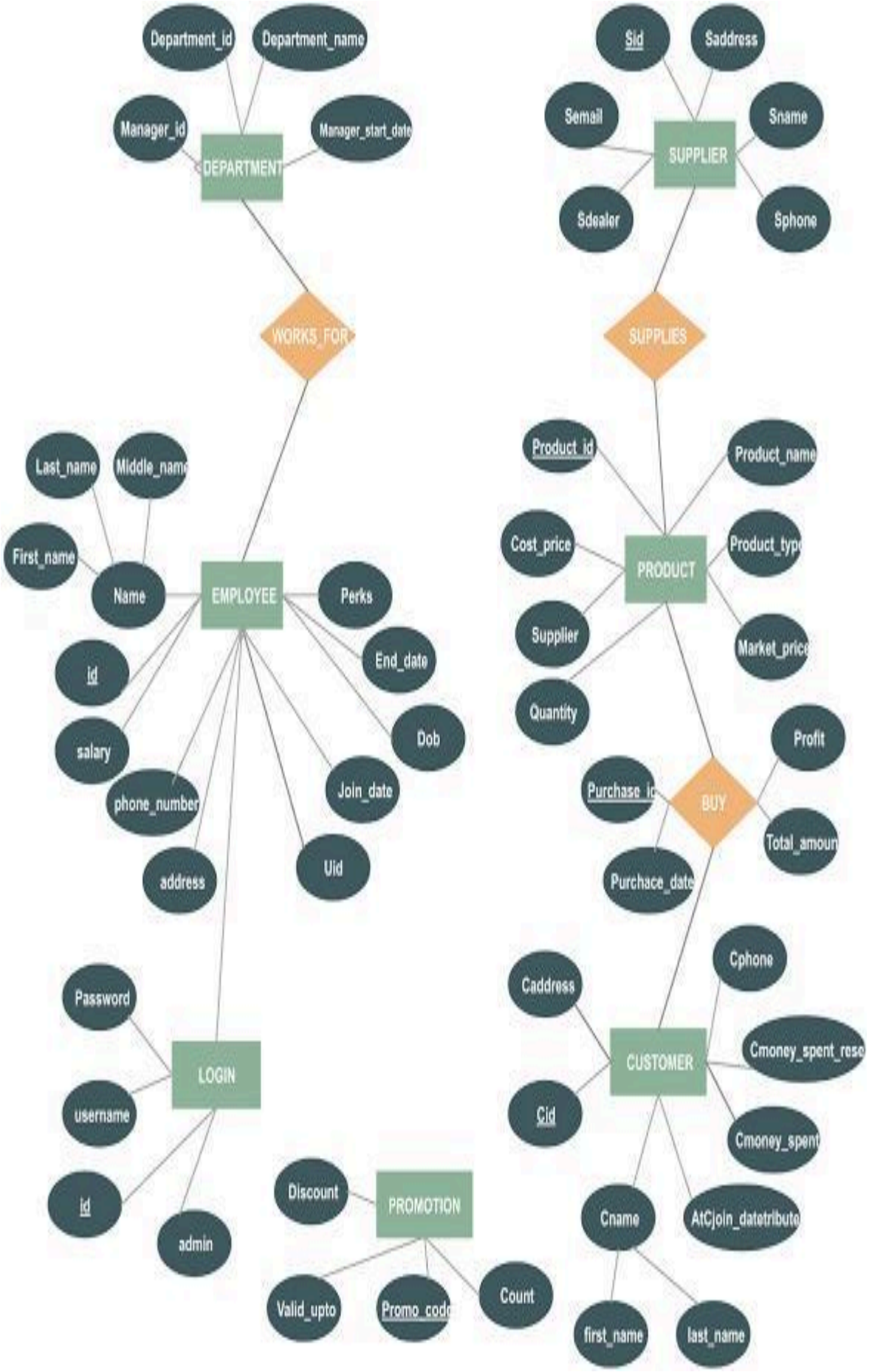
- EMPLOYEE — WORKS_FOR — DEPARTMENT
  (Many employees can work for one department.)

- EMPLOYEE — LOGIN
  Each Employee can have a Login (with username and password).

- PROMOTION — (linked to) — LOGIN
  Promotions are linked to Logins (maybe offering discounts to employees or customers through their login accounts).

- SUPPLIER — SUPPLIES — PRODUCT
  (One supplier can supply many products.)

- CUSTOMER — BUY — PRODUCT
  (Customers can purchase many products, and products can be purchased by many customers.)


Department_id
Department_name
Manager_id
Manager_start_date
DEPARTMENT
WORKS_FOR
Last_name
Middle_name
First_name
Name
EMPLOYEE
Perks
End_date
Dob
Join_date
Uid
id
salary
phone_number
address
Password
LOGIN
username
id
admin
Discount
PROMOTION
Valid_upto
Promo_code
Count
Sid
Saddress
Semail
Sname
SUPPLIER
Sdealer
Sphone
SUPPLIES
Product_id
Product_name
Cost_price
Product_type
PRODUCT
Supplier
Market_price
Quantity
Profit
Purchase_id
BUY
Total_amount
Purchace_date
Caddress
Cphone
CUSTOMER
Cmoney_spent_rese
Cid
Cmoney_spent
Cname
AtCjoin_datetribute
first_name
last_name

Thank Tou