**🛍️ App Concept: Jame – All-in-One Inventory & Sales Management App**

**👋 Overview**

You're building a **smart, offline-capable mobile app** to help manage your **shop’s inventory, sales, and payments**—all from your smartphone using **QR code scanning** and **receipt generation**. This is ideal for small businesses, grocery stores, or mini-marts.

The app will help you:

* Track products and stock
* Scan products at checkout
* Calculate total prices
* Accept payments via customer-scanned QR code
* Automatically generate receipts

## 🎯 Goal of the App

To make your shop:

* **Faster** in checkout
* **Smarter** in managing stock
* **More professional** with QR-based payments and digital receipts  
  All while being fully **offline-capable**, cost-efficient, and simple to use.

## 📱 Core Features

### 🔍 Product & Inventory Management

* Add/edit/delete product information
* Store product name, price, quantity, and QR/product code
* Scan product QR code to retrieve info instantly
* Track stock-in and stock-out
* Automatic stock reduction after sale
* Low stock alerts

### 🛒 POS & Cart System

* Scan multiple products to add to a virtual cart
* Auto-display product details (name, price, quantity)
* Modify quantity if needed
* Display real-time total price
* Option to remove or edit items from cart

### 💳 Payment via QR Code

* After finalizing the cart, app generates a **payment QR code** (e.g. PromptPay, PayPal, or mobile wallet)
* Customer scans the QR using their preferred payment app
* App confirms payment (manually or via API)
* Once paid, transaction is marked complete

### 🧾 Receipt & Sales Tracking

* Generate a receipt with:
  + Product list
  + Quantity
  + Price per item
  + Total price
  + Date & time
* Export or print the receipt (PDF or image)
* Share receipt via WhatsApp, email, etc.
* Store all sales data for future reporting

### 📊 Dashboard & Reports (Optional)

* View daily/weekly/monthly sales
* Track best-selling products
* Visual overview of stock levels
* Sales history with filtering

## 📋 App Pages & User Flow

| **Screen** | **Function** |
| --- | --- |
| **Login/Signup** (optional) | Secure login if you want multi-user support |
| **Dashboard** | Overview of sales, inventory, alerts |
| **Scan Product** | Use camera to scan QR or enter product code |
| **Cart / Checkout** | View items scanned, update quantity, view total price |
| **Payment Screen** | Show total and display dynamic payment QR code |
| **Receipt Page** | View/save/share printable receipt |
| **Products List** | Add/edit/remove products and stock |
| **Reports** (optional) | Track past sales and inventory performance |

## 🧱 Tech Stack (Your Choice: Flutter + Dart + SQLite)

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| **UI** | Flutter | Build responsive, cross-platform mobile app |
| **Language** | Dart | Used with Flutter to build logic and UI |
| **Database** | SQLite (sqflite) | Local database for storing products, sales, etc. |
| **QR Scanner** | mobile\_scanner or qr\_code\_scanner | Scan product QR codes |
| **QR Generator** | qr\_flutter | Generate payment QR codes |
| **PDF Generator** | pdf + printing | Create and print/share receipts |

## ✅ Your Flutter Tech Stack Overview

| **Feature / Requirement** | **Available in Flutter?** | **How to implement it** |
| --- | --- | --- |
| 📱 **User Interface (UI)** | ✅ Yes (built-in) | Use Flutter widgets |
| 💾 **Local Database (SQLite)** | ✅ Yes (via package) | Use [sqflite](https://pub.dev/packages/sqflite) or [drift](https://pub.dev/packages/drift) |
| 📷 **QR Code Scanner** | ✅ Yes (via package) | Use mobile\_scanner or qr\_code\_scanner |
| 🔳 **QR Code Generator** | ✅ Yes (via package) | Use [qr\_flutter](https://pub.dev/packages/qr_flutter) |
| 🧾 **PDF Receipt Generation** | ✅ Yes (via package) | Use [pdf](https://pub.dev/packages/pdf) + printing |
| 💲 **Calculate Total Prices** | ✅ Yes (custom logic) | Basic Dart logic, no extra library needed |
| 💳 **Payment QR Integration** | ✅ Yes (with logic) | Generate QR with payment link or info |
| 🔔 **Low Stock Alerts** | ✅ Yes (custom logic) | Compare quantity in stock and show UI alert |
| 📈 **Sales Reports** | ✅ Yes (custom logic/UI) | Use local data from SQLite and show graphs or lists |
| 🖨️ **Print Receipts (optional)** | ✅ Yes (with printing lib) | Use printing or native printer plugins |

## 💡 Database Structure (SQLite)

### products

id INTEGER PRIMARY KEY

name TEXT

price REAL

quantity INTEGER

code TEXT -- (QR code or product code)

### sales

id INTEGER PRIMARY KEY

date TEXT

total\_price REAL

### sale\_items

id INTEGER PRIMARY KEY

sale\_id INTEGER

product\_id INTEGER

quantity INTEGER

price\_per\_item REAL

## 🔄 Example User Flow: A Customer Buys Products

1. Open the app
2. Scan QR code on each product
3. Each scanned product is added to the cart
4. Total price is auto-calculated
5. Tap **Checkout**
6. App generates **payment QR code**
7. Customer scans the QR and pays
8. App confirms payment
9. Inventory is updated
10. Receipt is generated → print or share

## 🛠 Future Upgrades (Optional)

* ✅ Barcode support
* 🌍 Multi-language interface
* 🖨️ Bluetooth printer integration
* ☁️ Cloud sync with Firebase or Supabase
* 📤 Export to Excel or CSV
* 📈 Sales analytics dashboard
* 👥 Multi-user with role-based access

## ✅ 1. Project Folder Structure (Flutter + SQLite)

This structure is **scalable**, **clean**, and built for **maintainability**. It separates concerns and prepares your project for future upgrades (e.g., Firebase sync or multi-user).

## lib/

## ├── main.dart

## ├── core/

## │ ├── constants/ # App-wide constants (colors, styles, strings)

## │ ├── utils/ # Utilities (formatters, QR helpers, etc.)

## │ └── services/ # Services (e.g. PDFService, PaymentService)

## │

## ├── data/

## │ ├── models/ # Data models (Product, Sale, SaleItem)

## │ ├── database/ # SQLite DB init and queries

## │ │ ├── db\_helper.dart

## │ │ ├── product\_dao.dart

## │ │ └── sale\_dao.dart

## │ └── repositories/ # Repositories to access DB easily

## │ ├── product\_repository.dart

## │ └── sale\_repository.dart

## │

## ├── features/

## │ ├── product/

## │ │ ├── screens/

## │ │ │ └── product\_list\_screen.dart

## │ │ ├── controllers/

## │ │ │ └── product\_controller.dart

## │ │ └── widgets/

## │ │ └── product\_card.dart

## │ │

## │ ├── cart/

## │ │ ├── screens/

## │ │ │ └── cart\_screen.dart

## │ │ ├── controllers/

## │ │ │ └── cart\_controller.dart

## │ │ └── widgets/

## │ │ └── cart\_item\_tile.dart

## │ │

## │ ├── scan/

## │ │ ├── screens/

## │ │ │ └── scan\_screen.dart

## │ │ └── controllers/

## │ │ └── scan\_controller.dart

## │ │

## │ ├── payment/

## │ │ ├── screens/

## │ │ │ └── payment\_screen.dart

## │ │ └── controllers/

## │ │ └── payment\_controller.dart

## │ │

## │ ├── receipt/

## │ │ ├── screens/

## │ │ │ └── receipt\_screen.dart

## │ │ └── controllers/

## │ │ └── receipt\_controller.dart

## │

## ├── widgets/ # Reusable widgets (buttons, app bars, etc.)

## │

## └── providers/ # State management (using Provider or Riverpod)

## ├── cart\_provider.dart

## └── product\_provider.dart

🏗️ 2. App Architecture (Layered MVC + Provider Pattern)

The architecture is **clean**, based on 3 key layers:

### 1. **Presentation Layer (UI)**

* Flutter screens and widgets
* Displays data from providers or controllers
* Listens to changes in state (e.g., products, cart items)

### 2. **Business Logic Layer (Controllers / Providers)**

* Logic to scan QR, calculate totals, update stock
* Interacts with data layer through repositories
* Example: CartController manages cart operations

### 3. **Data Layer (Repositories + SQLite DAOs)**

* Handles local storage, DB reads/writes
* Abstracted via repositories to separate logic from DB calls

+----------------------------+

| Presentation Layer | ← UI screens & widgets

| - CartScreen |

| - ProductListScreen |

+----------------------------+

↓

+----------------------------+

| Business Logic Layer | ← Controllers / Providers

| - CartController |

| - ProductProvider |

+----------------------------+

↓

+----------------------------+

| Data Layer | ← SQLite + Repositories

| - ProductDAO |

| - SaleRepository |

+----------------------------+

🧠 You can use **Provider**, **Riverpod**, or **Bloc** depending on what you're most comfortable with. For local apps, **Provider** or **Riverpod** is lightweight and easy.