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

**Complete Project Documentation & Development Guide**

**📋 Table of Contents**

1. [App Concept Overview](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#app-concept-overview)
2. [Goals & Core Features](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#goals--core-features)
3. [App Pages & User Flow](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#app-pages--user-flow)
4. [Tech Stack](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#tech-stack)
5. [Database Design](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#database-design)
6. [Project Structure](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#project-structure)
7. [Color Palette & Design System](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#color-palette--design-system)
8. [Architecture & Implementation](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#architecture--implementation)
9. [Getting Started Checklist](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#getting-started-checklist)
10. [Future Upgrades](https://claude.ai/chat/2bd91d38-084d-4d0b-9e8a-a1e48fbbe2c8#future-upgrades)

**🎯 App Concept Overview**

**Jame** is 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

**🎯 Goals & Core Features**

**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.

**🔍 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 |

**🔄 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

**🧱 Tech Stack**

| **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 |

**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 Design (SQLite)**

**Database Schema**

**1. products table**

CREATE TABLE products (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

price REAL NOT NULL,

cost\_price REAL, -- for profit calculation

quantity INTEGER NOT NULL DEFAULT 0,

min\_stock\_level INTEGER DEFAULT 5, -- for low stock alerts

code TEXT UNIQUE, -- QR/barcode

category TEXT,

brand TEXT,

unit TEXT DEFAULT 'pcs', -- kg, liter, pcs, etc.

image\_path TEXT, -- local image storage

is\_active BOOLEAN DEFAULT 1,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP,

updated\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

**2. categories table**

CREATE TABLE categories (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

description TEXT,

color\_code TEXT DEFAULT '#FFC928',

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

**3. sales table**

CREATE TABLE sales (

id INTEGER PRIMARY KEY AUTOINCREMENT,

sale\_date TEXT NOT NULL DEFAULT CURRENT\_TIMESTAMP,

total\_amount REAL NOT NULL,

tax\_amount REAL DEFAULT 0,

discount\_amount REAL DEFAULT 0,

payment\_method TEXT DEFAULT 'QR', -- QR, Cash, Card

payment\_status TEXT DEFAULT 'Completed', -- Pending, Completed, Cancelled

customer\_name TEXT,

customer\_phone TEXT,

receipt\_number TEXT UNIQUE,

notes TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

**4. sale\_items table**

CREATE TABLE sale\_items (

id INTEGER PRIMARY KEY AUTOINCREMENT,

sale\_id INTEGER NOT NULL,

product\_id INTEGER NOT NULL,

quantity INTEGER NOT NULL,

unit\_price REAL NOT NULL,

total\_price REAL NOT NULL,

discount\_amount REAL DEFAULT 0,

FOREIGN KEY (sale\_id) REFERENCES sales (id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products (id)

);

**5. inventory\_logs table**

CREATE TABLE inventory\_logs (

id INTEGER PRIMARY KEY AUTOINCREMENT,

product\_id INTEGER NOT NULL,

change\_type TEXT NOT NULL, -- 'STOCK\_IN', 'STOCK\_OUT', 'ADJUSTMENT', 'SALE'

quantity\_change INTEGER NOT NULL, -- positive for in, negative for out

previous\_quantity INTEGER NOT NULL,

new\_quantity INTEGER NOT NULL,

reference\_id INTEGER, -- sale\_id if from sale, null for manual adjustment

notes TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products (id)

);

**6. app\_settings table**

CREATE TABLE app\_settings (

id INTEGER PRIMARY KEY AUTOINCREMENT,

setting\_key TEXT UNIQUE NOT NULL,

setting\_value TEXT,

updated\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

-- Default settings

INSERT INTO app\_settings (setting\_key, setting\_value) VALUES

('shop\_name', 'My Shop'),

('shop\_address', ''),

('shop\_phone', ''),

('tax\_rate', '0'),

('currency\_symbol', '$'),

('receipt\_footer', 'Thank you for shopping with us!'),

('low\_stock\_alert', '1'),

('auto\_backup', '1');

**Database Indexes (for performance)**

-- Improve query performance

CREATE INDEX idx\_products\_code ON products(code);

CREATE INDEX idx\_products\_category ON products(category);

CREATE INDEX idx\_sales\_date ON sales(sale\_date);

CREATE INDEX idx\_sale\_items\_sale\_id ON sale\_items(sale\_id);

CREATE INDEX idx\_sale\_items\_product\_id ON sale\_items(product\_id);

CREATE INDEX idx\_inventory\_logs\_product\_id ON inventory\_logs(product\_id);

**📁 Project Structure**

**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).

jame\_inventory\_app/

├── lib/

│ ├── main.dart # App entry point

│ │

│ ├── core/

│ │ ├── constants/

│ │ │ ├── app\_colors.dart # Color palette

│ │ │ ├── app\_strings.dart # Text constants

│ │ │ ├── app\_sizes.dart # Spacing, sizes

│ │ │ └── app\_routes.dart # Route names

│ │ │

│ │ ├── utils/

│ │ │ ├── date\_formatter.dart # Date utilities

│ │ │ ├── currency\_formatter.dart # Price formatting

│ │ │ ├── qr\_helper.dart # QR generation/validation

│ │ │ ├── receipt\_generator.dart # Receipt PDF generation

│ │ │ └── validators.dart # Form validation

│ │ │

│ │ ├── services/

│ │ │ ├── database\_service.dart # SQLite service

│ │ │ ├── pdf\_service.dart # PDF generation

│ │ │ ├── payment\_service.dart # Payment QR logic

│ │ │ ├── backup\_service.dart # Data backup/restore

│ │ │ └── notification\_service.dart # Local notifications

│ │ │

│ │ └── theme/

│ │ ├── app\_theme.dart # Light/dark themes

│ │ └── text\_styles.dart # Typography

│ │

│ ├── data/

│ │ ├── models/

│ │ │ ├── product.dart # Product model

│ │ │ ├── category.dart # Category model

│ │ │ ├── sale.dart # Sale model

│ │ │ ├── sale\_item.dart # Sale item model

│ │ │ ├── inventory\_log.dart # Inventory log model

│ │ │ ├── cart\_item.dart # Cart item model

│ │ │ └── app\_settings.dart # Settings model

│ │ │

│ │ ├── database/

│ │ │ ├── database\_helper.dart # SQLite initialization

│ │ │ ├── migrations.dart # DB version migrations

│ │ │ └── dao/

│ │ │ ├── product\_dao.dart # Product queries

│ │ │ ├── category\_dao.dart # Category queries

│ │ │ ├── sale\_dao.dart # Sale queries

│ │ │ ├── inventory\_dao.dart # Inventory queries

│ │ │ └── settings\_dao.dart # Settings queries

│ │ │

│ │ └── repositories/

│ │ ├── product\_repository.dart # Product business logic

│ │ ├── sale\_repository.dart # Sale business logic

│ │ ├── inventory\_repository.dart # Inventory business logic

│ │ └── settings\_repository.dart # Settings business logic

│ │

│ ├── presentation/

│ │ ├── pages/

│ │ │ ├── splash/

│ │ │ │ └── splash\_screen.dart

│ │ │ │

│ │ │ ├── dashboard/

│ │ │ │ ├── dashboard\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── sales\_summary\_card.dart

│ │ │ │ ├── quick\_actions\_grid.dart

│ │ │ │ └── low\_stock\_alerts.dart

│ │ │ │

│ │ │ ├── products/

│ │ │ │ ├── product\_list\_screen.dart

│ │ │ │ ├── product\_detail\_screen.dart

│ │ │ │ ├── add\_edit\_product\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── product\_card.dart

│ │ │ │ ├── product\_search\_bar.dart

│ │ │ │ └── category\_filter\_chips.dart

│ │ │ │

│ │ │ ├── categories/

│ │ │ │ ├── category\_list\_screen.dart

│ │ │ │ ├── add\_edit\_category\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ └── category\_card.dart

│ │ │ │

│ │ │ ├── scanner/

│ │ │ │ ├── qr\_scanner\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── scanner\_overlay.dart

│ │ │ │ └── manual\_code\_input.dart

│ │ │ │

│ │ │ ├── cart/

│ │ │ │ ├── cart\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── cart\_item\_tile.dart

│ │ │ │ ├── cart\_summary.dart

│ │ │ │ └── quantity\_selector.dart

│ │ │ │

│ │ │ ├── payment/

│ │ │ │ ├── payment\_screen.dart

│ │ │ │ ├── payment\_success\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── payment\_qr\_display.dart

│ │ │ │ ├── payment\_method\_selector.dart

│ │ │ │ └── amount\_display.dart

│ │ │ │

│ │ │ ├── receipt/

│ │ │ │ ├── receipt\_screen.dart

│ │ │ │ ├── receipt\_preview\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── receipt\_header.dart

│ │ │ │ ├── receipt\_items\_list.dart

│ │ │ │ └── receipt\_footer.dart

│ │ │ │

│ │ │ ├── reports/

│ │ │ │ ├── reports\_screen.dart

│ │ │ │ ├── sales\_report\_screen.dart

│ │ │ │ ├── inventory\_report\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── report\_chart.dart

│ │ │ │ ├── report\_filter.dart

│ │ │ │ └── report\_summary\_card.dart

│ │ │ │

│ │ │ ├── inventory/

│ │ │ │ ├── inventory\_screen.dart

│ │ │ │ ├── stock\_adjustment\_screen.dart

│ │ │ │ ├── inventory\_logs\_screen.dart

│ │ │ │ └── widgets/

│ │ │ │ ├── inventory\_item\_card.dart

│ │ │ │ ├── stock\_level\_indicator.dart

│ │ │ │ └── adjustment\_form.dart

│ │ │ │

│ │ │ └── settings/

│ │ │ ├── settings\_screen.dart

│ │ │ ├── shop\_settings\_screen.dart

│ │ │ ├── backup\_restore\_screen.dart

│ │ │ └── widgets/

│ │ │ ├── setting\_tile.dart

│ │ │ └── backup\_options.dart

│ │ │

│ │ ├── widgets/

│ │ │ ├── common/

│ │ │ │ ├── custom\_app\_bar.dart

│ │ │ │ ├── loading\_widget.dart

│ │ │ │ ├── error\_widget.dart

│ │ │ │ ├── empty\_state\_widget.dart

│ │ │ │ ├── custom\_button.dart

│ │ │ │ ├── custom\_text\_field.dart

│ │ │ │ └── confirmation\_dialog.dart

│ │ │ │

│ │ │ └── navigation/

│ │ │ ├── bottom\_nav\_bar.dart

│ │ │ └── drawer\_menu.dart

│ │ │

│ │ └── providers/

│ │ ├── cart\_provider.dart # Cart state management

│ │ ├── product\_provider.dart # Product state

│ │ ├── sale\_provider.dart # Sale state

│ │ ├── inventory\_provider.dart # Inventory state

│ │ ├── theme\_provider.dart # Theme state

│ │ └── settings\_provider.dart # Settings state

│ │

│ └── routes/

│ └── app\_router.dart # Navigation routing

│

├── assets/

│ ├── images/

│ │ ├── logo.png

│ │ ├── placeholder\_product.png

│ │ └── icons/

│ │

│ ├── fonts/

│ │ └── (custom fonts if needed)

│ │

│ └── data/

│ └── sample\_products.json # For demo data

│

├── test/

│ ├── unit\_tests/

│ │ ├── models/

│ │ ├── repositories/

│ │ └── services/

│ │

│ ├── widget\_tests/

│ │ └── widgets/

│ │

│ └── integration\_tests/

│ └── app\_test.dart

│

├── android/

├── ios/

├── pubspec.yaml

├── README.md

└── .gitignore

**🎨 Color Palette & Design System**

**🛍️ Grocery Shop UI Kit – Full Color Palette**

**✅ Primary Colors**

| **Color Name** | **Hex Code** | **Usage** |
| --- | --- | --- |
| Bright Yellow | #FFC928 | Main background, discounts, highlights |
| Dark Blue | #1E3A8A | Header, titles, primary CTA background |
| White | #FFFFFF | Card backgrounds, screen background |

**🌈 Secondary / Accent Colors**

| **Color Name** | **Hex Code** | **Usage** |
| --- | --- | --- |
| Light Gray | #F5F5F5 | Card background, borders, input backgrounds |
| Orange | #FFA500 | Tags, highlight buttons, discount banners |
| Green | #90C659 | Success status, organic tags |
| Soft Blue | #3B82F6 | Hyperlinks, small price tags |
| Black / Dark Gray | #1F2937 | Primary text, labels |

**🍊 Product-Related Colors**

*(Used in images or to match product type – these can be sampled or adjusted based on real product images)*

| **Product** | **Suggested Color (Hex)** | **Notes** |
| --- | --- | --- |
| Lemon Yellow | #FDE047 | Used for lemons and yellow tags |
| Lime Green | #A3E635 | Used for limes, freshness |
| Apple Red | #EF4444 | Used for apples |
| Banana Yellow | #FACC15 | Used for bananas |

**🟡 Optional Supporting Colors**

| **Color Name** | **Hex Code** | **Usage** |
| --- | --- | --- |
| Light Yellow | #FFF8DC | Backgrounds for product cards (optional) |
| Light Orange | #FFE4B5 | Discount banners, card hover (optional) |

**Flutter Color Constants Implementation**

// lib/core/constants/app\_colors.dart

class AppColors {

// Primary Colors

static const Color primaryYellow = Color(0xFFFFC928);

static const Color primaryDarkBlue = Color(0xFF1E3A8A);

static const Color primaryWhite = Color(0xFFFFFFFF);

// Secondary/Accent Colors

static const Color lightGray = Color(0xFFF5F5F5);

static const Color accentOrange = Color(0xFFFFA500);

static const Color successGreen = Color(0xFF90C659);

static const Color softBlue = Color(0xFF3B82F6);

static const Color textDarkGray = Color(0xFF1F2937);

// Product Colors

static const Color lemonYellow = Color(0xFFFDE047);

static const Color limeGreen = Color(0xFFA3E635);

static const Color appleRed = Color(0xFFEF4444);

static const Color bananaYellow = Color(0xFFFACC15);

// Status Colors

static const Color errorRed = Color(0xFFDC2626);

static const Color warningYellow = Color(0xFFFBBF24);

static const Color infoBlue = Color(0xFF3B82F6);

// Supporting Colors

static const Color lightYellow = Color(0xFFFFF8DC);

static const Color lightOrange = Color(0xFFFFE4B5);

}

**✅ Figma Style Guide – Grocery Shop App Colors**

**🎨 Color Styles to Create in Figma**

| **Style Name** | **Hex Code** | **Description** |
| --- | --- | --- |
| Primary / Yellow | #FFC928 | Main highlight, discount sections |
| Primary / Dark Blue | #1E3A8A | Headers, top bar, CTA buttons |
| Base / White | #FFFFFF | Backgrounds, card containers |
| Neutral / Light Gray | #F5F5F5 | Card backgrounds, borders, input fields |
| Accent / Orange | #FFA500 | Offer banners, callouts |
| Status / Green | #90C659 | Success indicators, organic tags |
| Accent / Soft Blue | #3B82F6 | Price tags, interactive links |
| Text / Dark Gray | #1F2937 | Primary text, labels |

**🍏 Product Colors (Optional, For Visual Branding)**

| **Style Name** | **Hex Code** | **Usage** |
| --- | --- | --- |
| Product / Lemon Yellow | #FDE047 | Lemon visuals, highlights |
| Product / Lime Green | #A3E635 | Lime, freshness tones |
| Product / Apple Red | #EF4444 | Apple visual tags |
| Product / Banana Yellow | #FACC15 | Banana items, subtle yellow highlights |

**📁 Suggested Figma Structure**

Inside your **Figma Design System**, organize the styles like this:

Color Styles

├── Primary

│ ├── Yellow

│ └── Dark Blue

├── Base

│ └── White

├── Neutral

│ └── Light Gray

├── Accent

│ ├── Orange

│ └── Soft Blue

├── Status

│ └── Green

├── Text

│ └── Dark Gray

└── Product

├── Lemon Yellow

├── Lime Green

├── Apple Red

└── Banana Yellow

**🏗️ Architecture & Implementation**

**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.

**Dependencies (pubspec.yaml)**

dependencies:

flutter:

sdk: flutter

# State Management

provider: ^6.0.5

# Database

sqflite: ^2.3.0

path: ^1.8.3

# QR Code

mobile\_scanner: ^3.5.2

qr\_flutter: ^4.1.0

# PDF Generation

pdf: ^3.10.4

printing: ^5.11.0

# File handling

path\_provider: ^2.1.1

permission\_handler: ^11.0.1

# UI Components

cupertino\_icons: ^1.0.2

flutter\_svg: ^2.0.7

cached\_network\_image: ^3.3.0

# Utilities

intl: ^0.19.0

uuid: ^4.1.0

share\_plus: ^7.2.1

url\_launcher: ^6.2.1

# Charts (for reports)

fl\_chart: ^0.65.0

dev\_dependencies:

flutter\_test:

sdk: flutter

flutter\_lints: ^3.0.0

build\_runner: ^2.4.7

**✅ Getting Started Checklist**

**Phase 1: Project Setup**

* [ ] Create new Flutter project
* [ ] Add dependencies to pubspec.yaml
* [ ] Set up project folder structure
* [ ] Create color constants and theme files
* [ ] Initialize SQLite database with tables

**Phase 2: Core Features**

* [ ] Implement database models and DAOs
* [ ] Create product management (CRUD operations)
* [ ] Implement QR code scanning
* [ ] Build cart functionality
* [ ] Create payment QR generation
* [ ] Implement receipt generation

**Phase 3: UI Development**

* [ ] Design and implement dashboard
* [ ] Create product listing and detail screens
* [ ] Build cart and checkout flow
* [ ] Design payment and receipt screens
* [ ] Implement inventory management UI

**Phase 4: Advanced Features**

* [ ] Add sales reports and analytics
* [ ] Implement inventory tracking and logs
* [ ] Create backup/restore functionality
* [ ] Add low stock notifications
* [ ] Implement settings management

**Phase 5: Testing & Polish**

* [ ] Write unit tests for business logic
* [ ] Create widget tests for UI components
* [ ] Test on different screen sizes
* [ ] Optimize performance
* [ ] Add error handling and edge cases

**🛠 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

**💡 Next Steps**

1. **Start with Phase 1** - Set up the project structure and database
2. **Focus on MVP** - Product management, cart, and basic sales
3. **Iterate and improve** - Add advanced features gradually
4. **Test thoroughly** - Ensure reliability for business use
5. **Consider future enhancements** - Cloud sync, multi-store support, etc.

This comprehensive documentation provides everything you need to build your Jame inventory management app, from initial concept to production-ready implementation. The structure is designed to be scalable, maintainable, and ready for future enhancements.