**Qryptix - Scan Smart, Stay Safe**

**Documentation**

**1. Project Overview**

**App Name:** Qryptix  
**Version:** 1.0  
**Platform:** Flutter (iOS & Android)

**Purpose:**  
The app verifies UPI QR codes for secure payments. It checks **UPI ID** and **Payee Name** against a local database and allows users to proceed with payment safely if verification fails.

**Key Idea:**

* **Frontend:** Handles UI, QR scanning, dialogs, and launching Google Pay.
* **Backend:** A local CSV file acts as the “backend database,” storing verified UPI IDs and corresponding names.

**2. Features**

**2.1 Frontend Features**

* Home screen showing verification status.
* QR code scanner for UPI payments.
* Dialogs for unverified users or name mismatch.
* Automatic launch of Google Pay for verified payments.
* User-friendly interface with clear status messages and feedback.

**2.2 Backend Features**

* CSV file storing verified UPI IDs and names.
* Fast lookup for UPI ID verification.
* Supports basic local database operations: read, match, validate.
* Can be extended to a server or cloud database if needed.

**3. System Architecture**

User Device

|

v

[Flutter App Frontend]

| - UI screens (Home, Scanner, Dialogs)

| - QR Scanner (MobileScanner)

| - Status display & Google Pay launch

v

[Backend (CSV Database)]

| - Local CSV: verified\_upi\_list.csv

| - Columns: UPI\_ID, Name

| - Lookup & verification logic

**3.1 Frontend Components**

| **Component** | **Description** |
| --- | --- |
| HomeScreen | Displays status, Scan QR button, and feedback card |
| QRViewExample | Camera scanner for QR codes |
| verifyQr() | Handles UPI ID and name verification logic |
| askProceedPayment() | Shows dialog for unverified users |
| launchGPay() | Launches Google Pay via URI |

**3.2 Backend Components (CSV Database)**

| **Component** | **Description** |
| --- | --- |
| assets/verified\_upi\_list.csv | Stores verified UPI IDs and corresponding names |
| CsvToListConverter | Reads CSV into a Dart list |
| upiMap | Map of UPI ID → Name for fast verification |
| Verification Logic | Checks both UPI ID (pa) and Name (pn) |

**4. Database / CSV Structure**

**CSV File:** assets/verified\_upi\_list.csv

**Columns:**

| **UPI\_ID** | **Name** |
| --- | --- |
| merchant1@upi | John Doe |
| merchant2@upi | Jane Smith |
| merchant3@upi | Acme Store |

**Usage:**

* The CSV acts as a **local backend**.
* All verification logic reads from this file.
* Can later be replaced by an actual backend server if required.

**5. Workflow**

**5.1 Step-by-Step Process**

1. **App Launch:**
   * Loads CSV backend → displays “Tap Scan to Payment”.
2. **User Scans QR Code:**
   * MobileScanner captures QR.
   * QR text extracted → pa (UPI ID) and pn (Name).
3. **Backend Verification:**
   * CSV lookup for UPI ID.
   * Name comparison with CSV entry.
4. **Decision Branch:**
   * **Verified:** status shown → launch Google Pay automatically.
   * **Name Mismatch / Not Verified:** dialog shown → user chooses Yes/No.
   * **Yes:** launch Google Pay with UPI prefilled.
   * **No:** return to Home screen.

**5.2 Flow Diagram (Text)**

[Open App]

|

v

[Load CSV Database]

|

v

[Scan QR Code]

|

v

[Extract UPI ID (pa) & Name (pn)]

|

+--> [UPI ID in CSV?] -- No --> [Not Verified Dialog] --> [Launch GPay? Yes/No]

|

Yes

|

[Name Match?] -- No --> [Name Mismatch Dialog] --> [Launch GPay? Yes/No]

|

Yes

|

[Verified] --> [Launch Google Pay Automatically]

**6. User Guide**

**6.1 Installation**

* Install app via Play Store / App Store or run Flutter project.
* Grant camera and storage permissions.

**6.2 Usage**

1. Open app → Home screen shows status.
2. Tap **Scan QR** → Camera opens.
3. Point camera to QR code.
4. Verification occurs automatically.
5. Status is displayed:
   * **Verified** → Google Pay launches.
   * **Name Mismatch / Not Verified** → dialog appears, user chooses Yes/No.

**6.3 Troubleshooting**

* QR not scanning → check camera permissions and lighting.
* Google Pay not installed → payment cannot launch.
* CSV missing → app will fail to verify.

**7. Technical Documentation**

**7.1 Code Structure**

lib/

├─ main.dart

├─ screens/

│ ├─ home\_screen.dart

│ └─ qr\_scan\_screen.dart

├─ services/

│ └─ csv\_service.dart (optional for modularity)

assets/

└─ verified\_upi\_list.csv

**7.2 Dependencies**

* Flutter ≥ 3.x
* mobile\_scanner → QR scanning
* csv → CSV parsing
* url\_launcher → Google Pay integration

**7.3 Key Functions**

| **Function** | **Purpose** |
| --- | --- |
| loadCsv() | Reads CSV and populates upiMap |
| verifyQr(String qrText) | Verifies UPI ID & name |
| extractParam(String text, String key) | Extracts parameters from QR URI |
| askProceedPayment(pa, pn) | Shows dialog for unverified/mismatch |
| launchGPay(String uri) | Launches Google Pay app |

**8. Security Considerations**

* Local CSV only; no sensitive info sent externally.
* Name and UPI ID verification reduces fraud.
* Payment through official Google Pay URI → secure.
* Optional: encrypt CSV or migrate to cloud backend.

**9. Future Enhancements**

* Replace CSV with cloud backend for real-time verification.
* Flexible name matching (ignore minor typos).
* Transaction history logs.
* Multi-language support and accessibility improvements.

**10. Summary**

* **Frontend:** Flutter app for UI, QR scanning, and Google Pay launch.
* **Backend:** Local CSV file as lightweight database for verification.
* The app combines UPI ID + Name verification for safer transactions.