# Front Desk Ops APP - Project outline

Skillset: check section “**5. Technology Stack”** below

Okay, this is a comprehensive request involving hardware integration, multiple SDKs, and data synchronization. Let's break this down into a project plan and a checklist.

First, a high-level understanding of the Check-In process at Tribute Music Gallery:

1. **User Input:** Government ID scanned by a USB Duplex ID scanner, processed by **Scan-ID** software -> auto-export scanned data from ID into a CSV file.
2. **Middleman Software (Your Application):**
   * Reads the most recent scan from the CSV file.
   * Uses First Name, Last Name, Birthday to query Wix Contacts via Wix JS SDK (headless).
   * Allows staff to select the correct Wix Contact.
   * Uses Anviz C2 Pro SDK/API to enroll the member's fingerprint.
   * Stores an association: (ID Scan Data + Wix Contact ID + Anviz User ID/Fingerprint Data) in a local, free, secure database (e.g., SQLite).
3. **Anviz C2 Pro Interaction:**
   * The "New SDK API 2021" document seems most relevant for direct device control from your application, especially functions like CCHex\_AddFingerprintOnline. This implies your application will likely communicate directly with the C2 Pro over the local network (or USB if the SDK supports it for this device).
   * The "Middleware Server API Interface" and "AnvizCloudKit" (PHP SOAP server) describe a cloud-centric or self-hosted server approach. While powerful, for a local front-desk application, direct SDK control might be simpler if your application can manage it (e.g., if it's a desktop app that can P/Invoke DLLs or use an FFI). Given the need for direct fingerprint enrollment initiation, the local SDK seems primary.

## 

## 

## Project Plan: Membership Management Software

**1. Project Title:** Front Desk Ops (working title)

**2. Introduction & Goals:**To develop a desktop application for the front desk of a private membership club. The application will streamline new member registration by:  
\* Ingesting member details from a scanned government ID.  
\* Querying and linking to existing member profiles in Wix Contacts.  
\* Enrolling new members' fingerprints using an Anviz C2 Pro device.  
\* Maintaining a local database to associate ID scan data, Wix contact information, and Anviz biometric data.  
The primary goal is to create an efficient, secure, and user-friendly registration and check-in process.

**3. Core Features:**

* **ID Scan & Data Ingestion (completed):**
  + Monitor a designated folder for new CSV files exported by Scan-ID.
  + Automatically parse CSV data (First Name, Last Name, Birthday).
* **Wix Contact Integration (Headless):**
  + Query Wix Contacts API using parsed ID data.
  + Display potential matches to the front desk staff. (**completed**)
  + Allow staff to select/confirm the correct Wix Contact.
  + Store the selected Wix Contact ID.
* **Anviz C2 Pro Fingerprint Enrollment:**
  + Initiate fingerprint enrollment mode on the C2 Pro for the selected member.
  + Receive confirmation and/or fingerprint template/reference from the C2 Pro.
  + Store Anviz User ID and fingerprint reference.
* **Local Data Management:**
  + Store associations between ID scan data, Wix Contact ID, and Anviz User ID/fingerprint data in a local SQLite database.
* **User Interface (UI):**
  + Clear, intuitive interface for front desk staff.
  + Display parsed ID information.
  + Show Wix Contact search results.
  + Provide feedback on fingerprint enrollment status.
* **Member Check-in:**
  + Listen for fingerprint scan events from Anviz C2 Pro.
  + Look up member in local DB via Anviz User ID.
  + Display associated Wix Contact information.

**4. System Architecture: (mermaid chart)**

graph TD

A[Member] -- Presents ID --> B(USB ID Scanner);

B -- Scans ID --> C(Scan-ID Software on Front Desk PC);

C -- Exports CSV --> D{Monitored CSV Folder};

E[Front Desk PC] --> F[FrontDeskOps Application (Electron/Python/C#)];

D -- Reads CSV --> F;

F -- Queries (FN, LN, DOB) --> G[Wix Contacts API (Headless)];

G -- Returns Potential Contacts --> F;

F -- Staff Selects Contact --> F;

F -- Initiates Enrollment / Sends User Info --> H(Anviz C2 Pro);

H -- Captures Fingerprint / Returns Status/Data --> F;

F -- Stores (ID Data, Wix ID, Anviz User ID, FP Ref) --> I[Local SQLite Database];

J[Staff] -- Interacts --> F;

Use code [with caution](https://support.google.com/legal/answer/13505487). Mermaid

**5. Technology Stack:** (red = need to find help)

* **Frontend/Application Framework:**
  + **: Electron (Node.js + Chromium):**
    - Allows use of Wix JavaScript SDK directly. **(in progress)**
    - Node.js for CSV parsing, file system operations. **(complete)**
    - Can interface with C/C++ DLLs (Anviz SDK) using ffi-napi or similar. (Documentation and Anviz SDK
    - This application will be used on a Windows 11 desktop computer
  + Alternatives: Python (with Tkinter/PyQt/Kivy) + ctypes for DLLs, C# .NET (Windows-specific) + P/Invoke.
* **ID Scanning:** Existing Scan-ID software and USB duplex scanner with auto-export of CSV. **(complete)**
* **Wix Integration:** Wix JavaScript SDK (for Headless sites). **(in progress)**
  + Parse newest entry from CSV file, query Wix contacts (first name, last name, birthday) and displays matching results (confidence-based
  + Having issues
* **Anviz Integration:** "New SDK API 2021" (Dynamic Link Library - DLL).
  + Focus on functions like CChex\_Start, CChex\_Stop, CChex\_Update (for async responses), CChex\_ListPersonInfo, CChex\_ModifyPersonInfo (to add user to device before FP enroll), CCHex\_AddFingerprintOnline.
* **Database:** SQLite (local, file-based, free, secure).

**6. Data Flow:**

* **New Member Registration:**
  1. Staff scans ID -> Scan-ID -> CSV.
  2. FrontDeskOps App ingests CSV.
  3. App queries Wix (FN, LN, DOB) -> Wix returns matches.
  4. Staff selects Wix Contact.
  5. App (optionally, if required by Anviz SDK) adds user info (e.g., from CSV or Wix) to Anviz C2 Pro using CChex\_ModifyPersonInfo (this creates an EmployeeId on the device).
  6. App initiates fingerprint enrollment for that EmployeeId using CCHex\_AddFingerprintOnline.
  7. Anviz C2 Pro guides user; enrollment result returned via CChex\_Update (MsgType CCHEX\_RET\_ADD\_FINGERPRINT\_ONLINE\_TYPE)
  8. App stores [ScanID\_FN, ScanID\_LN, ScanID\_DOB, WixContactID, Anviz\_EmployeeID, FingerprintIndex, EnrollmentStatus] in SQLite.
* **Member Check-in:**
  1. Member scans fingerprint on C2 Pro.
  2. Anviz C2 Pro sends attendance event (via CChex\_Update, MsgType CCHEX\_RET\_RECORD\_INFO\_TYPE or CCHEX\_RET\_LIVE\_SEND\_ATTENDANCE\_TYPE).
  3. App receives Anviz\_EmployeeID.
  4. App queries local SQLite for Anviz\_EmployeeID -> retrieves WixContactID.
  5. App (optionally) queries Wix API for full contact details using WixContactID.
  6. App displays member info/check-in confirmation.

**7. Database Schema (SQLite):**

* **Members Table:**
  + MemberID (INTEGER, PRIMARY KEY, AUTOINCREMENT)
  + WixContactID (TEXT, UNIQUE) - Foreign key or reference to Wix.
  + AnvizEmployeeID (TEXT, UNIQUE) - ID on the Anviz device.
  + ScanFirstName (TEXT)
  + ScanLastName (TEXT)
  + ScanDOB (TEXT) - Store as YYYY-MM-DD.
  + RegistrationTimestamp (DATETIME, DEFAULT CURRENT\_TIMESTAMP)
* **Fingerprints Table:**
  + FingerprintID (INTEGER, PRIMARY KEY, AUTOINCREMENT)
  + MemberID\_FK (INTEGER, FOREIGN KEY REFERENCES Members(MemberID))
  + AnvizFingerIndex (INTEGER) - e.g., 0-9 for different fingers.
  + AnvizFingerprintData (BLOB or TEXT) - If SDK provides template, otherwise just status.
  + EnrollmentTimestamp (DATETIME, DEFAULT CURRENT\_TIMESTAMP)
* **AttendanceLog Table:**
  + LogID (INTEGER, PRIMARY KEY, AUTOINCREMENT)
  + MemberID\_FK (INTEGER, FOREIGN KEY REFERENCES Members(MemberID))
  + AnvizEmployeeID\_FK (TEXT) - For quick lookup from device event.
  + CheckinTimestamp (DATETIME, DEFAULT CURRENT\_TIMESTAMP)
  + DeviceID (TEXT) - If multiple C2 Pros are used in future.

**8. Development Phases & Modules: [ not started | some complete | mostly complete ]**

* **Phase 1: Setup & Core Libraries [45% complete]**
  + **Set up development environment (Electron/Node.js or chosen alternative).**
  + **Basic UI shell.**
  + SQLite database setup and wrapper/ORM.
  + Integrate Anviz SDK (load DLL, basic init/start/stop calls).
* **Phase 2: ID Scan & CSV Processing [95% complete]**
  + **Implement CSV file monitoring and parsing.**
  + **Display parsed data in UI.**
* **Phase 3: Wix Integration [90% complete]**
  + **Implement Wix JS SDK authentication (headless).**
  + **Develop Wix Contact query functionality.**
  + **UI for displaying Wix search results and selection.**
* **Phase 4: Anviz C2 Pro Integration - Enrollment [0% complete]**
  + Implement logic to add a new user to the C2 Pro (CChex\_ModifyPersonInfo).
  + Implement CCHex\_AddFingerprintOnline to initiate enrollment.
  + Handle asynchronous responses via CChex\_Update for enrollment status.
* **Phase 5: Anviz C2 Pro Integration - Check-In [0% complete]**
  + Real-time member check-in using fingerprint scans.
  + Scan fingerprint -> Displays matching Wix membership data and current plan status along with relevant op
* **Phase 6: Data Persistence & Linking [0% complete]**
  + Save combined data (Scan, Wix, Anviz) to SQLite.
  + Ensure data integrity and associations.
* **Phase 7: Testing & Refinement [0% complete]**
  + Unit tests for individual modules.
  + Integration testing with actual hardware (ID Scanner, C2 Pro).
  + User acceptance testing (UAT) with front desk staff.
  + Bug fixing and performance optimization.
* **Phase 8: Deployment & Documentation [15% complete]**
  + Create installer/package for the front desk PC.
  + User manual for front desk staff.
  + Technical documentation.

**9. Key Challenges & Risks:**

* **Anviz SDK Integration:** Interfacing with a C/C++ DLL from the chosen application framework (e.g., Electron/Node.js via ffi-napi) can be complex. Understanding the exact data structures, memory management, and asynchronous callback mechanisms of the Anviz SDK is critical. The "New SDK API 2021" document has many C-style structs that will need careful mapping.
* **Asynchronous Operations:** Both Wix API calls and Anviz SDK operations will likely be asynchronous. Managing callbacks, promises, or async/await patterns correctly is crucial for a responsive UI.
* **Wix Headless Authentication:** Ensuring secure and robust authentication with the Wix platform for headless operations.
* **Hardware Reliability:** Dependence on the ID scanner and Anviz C2 Pro functioning correctly. Error handling for hardware issues is needed.
* **Error Handling & Edge Cases:** Robust error handling for API failures, device communication issues, invalid CSV data, no Wix match, fingerprint enrollment failures, etc.
* **Security:** Protecting Wix API keys/tokens, securing the local SQLite database (if it contains sensitive data beyond references).
* **Scan-ID CSV Format:** Changes to the Scan-ID CSV output format could break parsing logic.

**10. Deployment:**The application will be a desktop application deployed to the front desk PC(s) at the club.

**11. Next part:**  
\* Reporting features (e.g., new member registrations per day).  
\* Updating Wix Contact information from the app (e.g., marking as "biometric enrolled").  
\* Cloud backup of the local SQLite database.

## Feature Checklist

**I. Project Setup & Environment**[✅] 1. Choose primary application framework (e.g., Electron).  
[✅] 2. Set up development environment for the chosen framework.  
[✅] 3. Initialize Git repository for version control.  
[ ] 4. Basic UI scaffolding/layout designed.  
[ ] 5. SQLite database file created and connection library integrated.  
[ ] 6. Create initial database schema.

**II. ID Scan & CSV Processing Module**[✅] 1. Implement logic to monitor a specific folder for new .csv files.  
[✅] 2. Implement CSV parsing logic for First Name, Last Name, Birthday.  
[✅] 3. UI component to display parsed ID information.  
[✅] 4. Error handling for invalid CSV format or missing files.  
[✅] 5. Test with sample CSV files from Scan-ID.

**III. Wix Contacts Integration Module (Headless)**[✅] 1. Integrate Wix JS SDK into the application.  
[✅] 2. Implement secure authentication/token management for Wix Headless API.  
[✅] 3. Develop function to query Wix Contacts by First Name, Last Name.  
[✅] 4. Develop logic to further filter Wix results by Birthday.  
[✅] 5. UI component to display Wix contact search results.  
[✅] 6. UI mechanism for staff to select the correct Wix contact.  
[ ] 7. Store selected Wix Contact ID.  
[ ] 8. Test Wix API queries and authentication.

**IV. Anviz C2 Pro SDK Integration Module**[ ] 1. Load Anviz SDK DLL (New SDK API 2021) into the application.  
[ ] 2. Implement CChex\_Init() and CChex\_Start()successfully.  
[ ] 3. Implement CChex\_Stop() for clean shutdown.  
[ ] 4. Implement CChex\_Update() loop to process asynchronous messages.  
[ ] 5. **User Management on Device:**[ ] 5.1. Function to add/modify person info on C2 Pro (e.g., CChex\_ModifyPersonInfo) to create an EmployeeIdfor the new member. (Data from CSV/Wix).  
[ ] 5.2. Handle response for person modification.  
[ ] 6. **Fingerprint Enrollment:**[ ] 6.1. Implement function to trigger online fingerprint enrollment (CCHex\_AddFingerprintOnline) for a given EmployeeId and finger index.  
[ ] 6.2. Handle asynchronous response for enrollment success/failure/timeout (via CChex\_Update, check MsgType CCHEX\_RET\_ADD\_FINGERPRINT\_ONLINE\_TYPE).  
[ ] 6.3. UI feedback for enrollment process (e.g., "Place finger," "Success," "Try again").  
[ ] 7. Store Anviz EmployeeId and enrolled finger index/status.  
[ ] 8. Test basic communication with C2 Pro.  
[ ] 9. Test adding a user to C2 Pro.  
[ ] 10. Test fingerprint enrollment flow with C2 Pro.

**V. Local Database Module (SQLite)**[ ] 1. Implement functions to insert new member records (associating Scan ID, Wix ID, Anviz ID).  
[ ] 2. Implement functions to insert fingerprint records.  
[ ] 3. Implement queries to retrieve member data.  
[ ] 4. Ensure data relationships (foreign keys) are correctly established.  
[ ] 5. Test database CRUD operations.

**VI. Application UI & Workflow**[ ] 1. Main application window/dashboard.  
[ ] 2. Step-by-step workflow for new member registration.  
[ ] 3. Clear visual feedback for each step (ID scan, Wix search, FP enrollment).  
[ ] 4. Settings/Configuration section (e.g., CSV folder path, Anviz device IP if needed).  
[ ] 5. Logging mechanism within the app for diagnostics.

**VII. Testing & Quality Assurance**[ ] 1. Unit tests for CSV parsing, Wix API calls, DB operations.  
[ ] 2. Integration test: Full new member registration flow with ID scanner, C2 Pro, and Wix.  
[ ] 3. Test various scenarios:  
[ ] 3.1. New CSV arrival.  
[ ] 3.2. Wix contact found / not found / multiple matches.  
[ ] 3.3. Fingerprint enrollment success/failure.  
[ ] 3.4. Hardware disconnection/errors.  
[ ] 4. User Acceptance Testing (UAT) with front desk staff.  
[ ] 5. Performance testing (if applicable).  
[ ] 6. Security review (API key storage, data handling).

**VIII. Documentation & Deployment**[ ] 1. User manual for front desk staff.  
[ ] 2. Technical documentation (architecture, setup, SDK usage notes).  
[ ] 3. Create an installer or deployment package for the front desk PC.  
[ ] 4. Backup strategy for the local SQLite database.

This plan prioritizes using the "New SDK API 2021" for direct, local control of the Anviz C2 Pro, which seems most appropriate for a front-desk application.

The AnvizCloudKit (PHP SOAP server) would be an alternative if the application itself were web-based and needed a backend to communicate with devices, or if direct SDK integration proved too challenging.

AnvizCloudKit: <https://github.com/AnvizJacobs/AnvizCloudKit>

Project Repo: <https://github.com/odmustafa/mini_checkin_app/tree/feature/confidence-based-matching>