Front Desk Ops APP - Project outline

https://meet.google.com/gjt-jajm-kon

https://github.com/odmustafa/mini_checkin_app-feature-confidence-based-matching/tree/windows-computer

#NEW-SECTION: Tribute Music Gallery Branding Information

App should follow the brand color scheme and use the same font for the app name as the Tribute Music Gallery logo.

- App Name: **TRiBE**
    - Notes: All capital letters, except lower-case “i”.
- Font: **Know Your Product**
  - Download: https://www.1001fonts.com/know-your-product-font.html
  - License: 1001Fonts Free For Commercial Use License (FFC)
- Color Pallet:

Black (primary)	
Hex	#000000
RGB	0, 0, 0
White	
Hex	#FFFFFF
RGB	255, 255, 255
Artichoke (accent)	
Hex	#949878
RGB	148, 152, 120

See: https://www.tribute.gallery/join for example on how color pallet is used.

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 New Members Onboarding process at Tribute Music Gallery (assuming new member has already created an account on the website (https://tribute.gallery) and have purchased a membership plan):

1. **User Input:** Government ID scanned by a USB Duplex ID scanner, processed by the **Scan-ID** software -> auto-export scanned data from ID into a CSV file.
2. **Middleman Software (the application):**
    - Reads the most recent scan from the CSV file.
    - Use First Name, Last Name, Birthday to query Wix Contacts via Wix JavaScript SDK (headless) and display matching accounts with confidence rating (calculated by how closely the account information matches the information from the scanned ID).
    - Allows staff to select the correct Wix Contact, which will display the current status of their membership plan (active, expired, or undefined), plan-related details, and recent membership history
    - (Wix JavaScript SDK pricing-plans order.managementListOrders( ) to get a Wix Contact's membership history maybe?).
    - There will be an "Enroll FP" button, which will put the **fingerprint scanner** into enrollment mode. Member will be instructed to place their finger on the scanner and go through the process to enroll their fingerprint.
    - Stores an association: (ID Scan Data + filename of the scanned photo of their ID (front and back are automatically combined into 1 photo by the duplex scanner) + Wix Contact ID + Relevant Fingerprint Data) in a local, free, secure database (e.g., SQLite or something).
3. **Fingerprint Scanner**
    - After they have completed New Member Onboarding on their initial visit to Tribute, members check-in on subsequent visits will be streamlined:
        i. Upon arrival, members will scan their finger
        ii. The application will retrieve and display account information associated with their fingerprint including
            1. Name
            2. Birthday (Age)
            3. Additional Notes (saved to local database, for internal purposes only, can be added by Front Desk staff or management).
            4. Current Membership Plan Info

a. Status (active, expired)
b. Plan Type (Week, Month, Year, Legacy, Staff)
c. Expiration date of most recent plan

5.

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 fingerprint scanner.
* Maintaining a local database to associate ID scan data, Wix contact information, and fingerprint 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.
- **Fingerprint Scanner Fingerprint Enrollment:**
  - Initiate fingerprint enrollment mode on the Fingerprint Scanner for the selected member.
  - Receive confirmation and/or fingerprint template/reference from the Fingerprint Scanner.

    - Store members associated localDB_id and fingerprint reference.
- **Local Data Management:**
    - Store associations between ID scan data, Wix Contact ID, and 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 Fingerprint Scanner.
    - Look up member in local DB via members associated localDB_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(Fingerprint Scanner);
    H -- Captures Fingerprint / Returns Status/Data --> F;

    F -- Stores (ID Data, Wix ID, members associated localDB_id, FP Ref) --> I[Local SQLite Database];

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

Use code with caution. 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 (SecurGen SDK) using `ffi-napi` or similar. (Documentation and SecurGen 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
- **fingerprint scan 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 SecurGen SDK) adds user info (e.g., from CSV or Wix) to Fingerprint Scanner using `CChex_ModifyPersonInfo` (this creates an `EmployeeId` on the device).
    6. App initiates fingerprint enrollment for that `EmployeeId` using `CCHex_AddFingerprintOnline`.
    7. Fingerprint Scanner 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, members localDB_id, FingerprintIndex, EnrollmentStatus] in SQLite.

- **Member Check-in:**
  1. Member scans fingerprint on Fingerprint Scanner.
  2. Fingerprint Scanner sends attendance event (via `CChex_Update`, MsgType `CCHEX_RET_RECORD_INFO_TYPE` or `CCHEX_RET_LIVE_SEND_ATTENDANCE_TYPE`).
  3. App receives members localDB_id.
  4. App queries local SQLite for members localDB_id -> 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.
  - `members localDB_id` (TEXT, UNIQUE) - local database id associated with each member.
  - `ScanFirstName` (TEXT)
  - `ScanLastName` (TEXT)
  - `ScanDOB` (TEXT) - Store as YYYY-MM-DD.
  - `RegistrationTimestamp` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **`Fingerprints` Table (theoretical, based on Anviz hardware documentation - no longer using in this project)**
  - `FingerprintID` (INTEGER, PRIMARY KEY, AUTOINCREMENT)
  - `MemberID_FK` (INTEGER, FOREIGN KEY REFERENCES Members(MemberID))
  - `fingerprint scanFingerIndex` (INTEGER) - e.g., 0-9 for different fingers.
  - `fingerprint scanFingerprintData` (BLOB or TEXT) - If SDK provides template, otherwise just status.
  - `EnrollmentTimestamp` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **`AttendanceLog` Table (based on Anviz hardware documentation - no longer using in this project):**
  - `LogID` (INTEGER, PRIMARY KEY, AUTOINCREMENT)
  - `MemberID_FK` (INTEGER, FOREIGN KEY REFERENCES Members(MemberID))
  - `members localDB_id_FK` (TEXT) - For quick lookup from device event.

- `CheckinTimestamp` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `DeviceID` (TEXT) - If multiple Fingerprint Scanners 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 SecurGen 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: Fingerprint Scanner Integration - Enrollment [0% complete]**
  - Need outside help on this.

- **Phase 5: Fingerprint Scanner Integration - Check-In [0% complete]**
  - And this
- **Phase 6: Data Persistence & Linking [0% complete]**
  - And this, too
- **Phase 7: Testing & Refinement [0% complete]**
  - Unit tests for individual modules.
  - Integration testing with actual hardware (ID Scanner, Fingerprint Scanner).
  - User acceptance testing (UAT) with front desk staff *(Need to get Kendee and Amber to test for feedback on process flow compared to soft opening night)*
  - Bug fixing and performance optimization.
- **Phase 8: Deployment & Documentation [15% complete]**
  - Halpppp!
  - User manual for front desk staff.
  - Technical documentation.

9. Key Challenges & Risks:

- **Im not a programmer, this ain't in my scope of knowledge. Ahhhh!!!**
- **Asynchronous Operations:** Both Wix API calls and SecurGen 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 Fingerprint Scanner 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). Dang hackers n data thieves snooping n stuff.
- **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 computer..

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

[✅] 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 (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. Fingerprint Scanner SDK Integration

[ ] 1. Add Fingerprint Enrollment function

[ ] 2. Implement function to trigger fingerprint enrollment.

[ ] 3. UI feedback for enrollment process (e.g., "Place finger," "Success," "Try again").

[ ] 4. Store fingerprint data with associated member's information

[ ] 5. Add Check-In feature to recognize fingerprint when scanned and pull associated account information up.

V. Local Database (SQLite?)

[ ] 1. Implement functions to insert new member records (associating Scan ID, Wix ID, Fingerprint data).

[ ] 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 and scanned ID image file folder path, secret keys for wix.config.json, connection status showing if if wix api is working properly, if fingerprint scanner is connected and recognized by app, csv file is being watched for updates (which would indicate a new ID was scanned and its information was auto-exported)).
[ ] 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, Fingerprint Scanner, 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.

Project Repo

- This is the most recent version of what I've been trying to put together
- **NOTE:** keep in mind, this was all created using AI. Also, I was focusing on getting the various backend functions working and the UI was an afterthought. Do not use the UI for this as an example): https://github.com/odmustafa/mini_checkin_app-feature-confidence-based-matching/tree/windows-computer