Tribute Biometric System – Full Source & Integration Guide

# 🔐 Introduction

This document contains the full source code and integration instructions for the Tribute Gallery Biometric System.  
It includes backend and frontend code, the SecuGen SDK setup, and full Wix API integration.  
The system is built to identify members using their fingerprints and show their membership status at the door.

# ⚙️ Backend (Flask/Python)

## app.py

from flask import Flask, request, jsonify  
from flask\_sqlalchemy import SQLAlchemy  
from flask\_cors import CORS  
  
app = Flask(\_\_name\_\_)  
CORS(app)  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///tribute\_users.db'  
app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  
db = SQLAlchemy(app)  
  
class User(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 name = db.Column(db.String(50))  
 email = db.Column(db.String(120), unique=True)  
 membership\_type = db.Column(db.String(50))  
 fingerprint\_template = db.Column(db.LargeBinary)  
  
@app.before\_first\_request  
def create\_tables():  
 db.create\_all()  
  
def generate\_template\_from\_scan(scan\_data):  
 import hashlib  
 return hashlib.sha256(scan\_data.encode()).digest()  
  
def match\_templates(template1, template2):  
 return template1 == template2  
  
@app.route('/register', methods=['POST'])  
def register():  
 data = request.json  
 template = generate\_template\_from\_scan(data['fingerprint\_scan'])  
 new\_user = User(name=data['name'], email=data['email'], membership\_type=data['membership\_type'], fingerprint\_template=template)  
 db.session.add(new\_user)  
 db.session.commit()  
 return jsonify({"message": "User registered."})  
  
@app.route('/login', methods=['POST'])  
def login():  
 scan = request.json['fingerprint\_scan']  
 template = generate\_template\_from\_scan(scan)  
 users = User.query.all()  
 for user in users:  
 if match\_templates(user.fingerprint\_template, template):  
 return jsonify({"message": f"Welcome {user.name}", "membership": user.membership\_type})  
 return jsonify({"error": "No match found"}), 401  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

## requirements.txt

Flask==2.3.2  
Flask-SQLAlchemy==3.0.5  
Flask-Cors==3.1.1

## biometric\_sdk.py

import hashlib  
  
def generate\_template\_from\_scan(scan\_data):  
 return hashlib.sha256(scan\_data.encode()).digest()  
  
def match\_templates(template1, template2):  
 return template1 == template2

## models.py

from flask\_sqlalchemy import SQLAlchemy  
  
db = SQLAlchemy()  
  
class User(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 name = db.Column(db.String(50))  
 email = db.Column(db.String(120), unique=True)  
 membership\_type = db.Column(db.String(50))  
 fingerprint\_template = db.Column(db.LargeBinary)

# 💻 Frontend (React)

## BiometricLogin.js

import axios from 'axios';  
import React, { useState } from 'react';  
  
function BiometricLogin() {  
 const [status, setStatus] = useState("");  
  
 const handleLogin = async () => {  
 const scan = prompt("Enter fingerprint scan (test input):");  
 try {  
 const response = await axios.post('http://localhost:5000/login', { fingerprint\_scan: scan });  
 setStatus(`${response.data.message} - Membership: ${response.data.membership}`);  
 } catch (error) {  
 setStatus("Authentication failed");  
 }  
 };  
  
 return (  
 <div>  
 <h2>Biometric Login</h2>  
 <button onClick={handleLogin}>Scan & Login</button>  
 <p>{status}</p>  
 </div>  
 );  
}  
  
export default BiometricLogin;

## index.js

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import BiometricLogin from './BiometricLogin';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(<BiometricLogin />);

## package.json

{  
 "name": "tribute-bio-frontend",  
 "version": "1.0.0",  
 "dependencies": {  
 "axios": "^1.6.0",  
 "react": "^18.2.0",  
 "react-dom": "^18.2.0"  
 }  
}

# 🌐 Wix API Integration

## http-functions.js

import {ok, badRequest} from 'wix-http-functions';  
  
export function post\_login(request) {  
 return request.body.text()  
 .then(body => {  
 const data = JSON.parse(body);  
 return fetch("https://your-backend-url/login", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ fingerprint\_scan: data.fingerprint\_scan })  
 });  
 })  
 .then(res => res.json())  
 .then(json => ok({ body: json }))  
 .catch(err => badRequest({ error: err.message }));  
}

# 🧬 SecuGen SDK Setup (Windows – Hamster Pro 20)

1. Download the FDx SDK Pro for Windows from: https://secugen.com/download/  
2. Install the drivers and SDK.  
3. Use `sgfplib.dll` and access via Python using ctypes or integrate with your C++/C# app.  
4. Capture the fingerprint image and convert it to a byte stream or string.  
5. Send the scan string to your backend via fetch() or Axios.  
  
This can be used for live scans and transmitted to the Flask backend where the fingerprint template is validated.