**AI POWERED MULTIFACTOR AUTHENTICATION SYSTEM**

**TEAM NAME: QUANTUM CODERS**

**MEMBER 1: VEDANATHAN PON ABIRAM**

**MEMBER 2: SARA SAKEENA**

**MEMBER 3: SHANMUGA PRIYA**

**MEMBER 4: SRI RANJINI**

**SOLUTION OVERVIEW:**

In the current digital landscape, **traditional authentication methods** such as passwords and OTPs are increasingly proving to be **inadequate and vulnerable**. They are highly susceptible to a range of **cyber threats**, including brute force attacks, credential stuffing, phishing, and social engineering. Moreover, these methods often introduce **user friction** due to lengthy or repetitive processes, ultimately compromising both **security and user experience**.

To address these challenges, we have developed an **AI/ML-powered multi-layered authentication solution** that **reimagines identity verification** through intelligent, adaptive mechanisms. Our system is designed to offer a **seamless, secure, and scalable** authentication framework suitable for a variety of platforms—web, mobile, and enterprise applications.

**Core Features and Capabilities:**

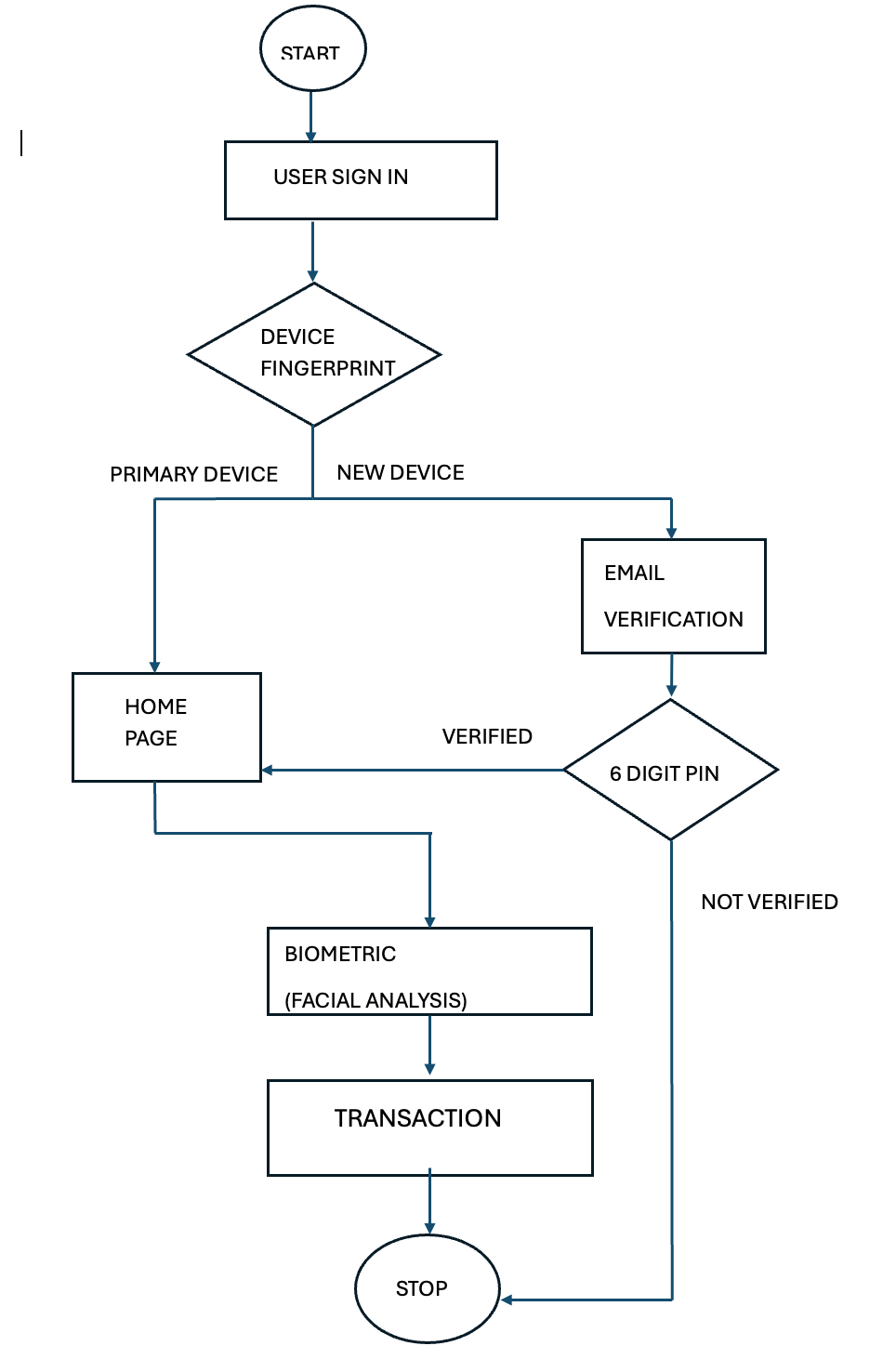
* **Multi-layered Security Architecture**: Combines traditional credentials with advanced biometric methods and AI-based analysis to create a robust authentication process.
* **Device Fingerprinting**: Each device used for login is uniquely identified through hardware and software metadata, helping detect and block suspicious devices even before login attempts are made.
* **Biometric Face Recognition**: Leveraging deep learning models, we offer secure and fast facial recognition that can identify users with high accuracy, reducing dependency on passwords.
* **AI-Powered Anomaly Detection**: The system continuously learns user behavior patterns (e.g., login time, location, device) to detect anomalies and flag or block suspicious activity in real time.
* **Phishing & Credential Attack Prevention**: Our ML models are trained to detect phishing attempts and prevent users from falling prey to malicious links or suspicious inputs.

**SECURITY**

Our system is designed not just for **maximum security**, but also to deliver a **smooth and frictionless experience** to the end-user. By eliminating repetitive logins and password resets, while simultaneously elevating the security through biometrics and intelligent analysis, we create a future-ready authentication system that is:

* **User-Friendly**
* **Highly Secure**
* **Easy to Integrate**
* **Scalable across platforms and devices**

**SOLUTION DESIGN**



| **Component** | **Purpose** |
| --- | --- |
| **Frontend (Web/Mobile)** | Provides user interface for login and registration; captures biometric data (face/fingerprint) and communicates with the backend securely. |
| **Backend Server (API)** | Core authentication logic; handles user login requests, processes biometric verification, issues tokens, and communicates with AI/ML services. |
| **Device Fingerprinting Module** | Uniquely identifies the user’s device using hardware/software metadata; helps detect unauthorized device access. |
| **Biometric Authentication Module** | Performs face/fingerprint recognition using AI models; verifies identity securely and conveniently. |
| **AI/ML Engine** | Analyzes login behavior and patterns; detects anomalies, unusual login attempts, and potential threats using trained models. |
| **Database** | Stores user credentials, biometric templates (encrypted), device fingerprints, and authentication logs. |
| **Anomaly Detection Module** | Monitors user activity and flags suspicious behavior in real-time; triggers adaptive authentication if needed. |
| **Security Layer (SHA-512)** | Ensures all data transmission is encrypted; protects user data during communication between frontend, backend, and cloud. |

**SOURCE CODE FILE**

**GITHUB:** [**https://github.com/sarasakeena/iobproject**](https://github.com/sarasakeena/iobproject)

**DEPENDENT LIBRARIES**

#### ****Security & Authentication****

* flask – Web framework to build the backend server..
* dotenv – Loads environment variables from .env file securely.
* psycopg2 – PostgreSQL database connector.

#### ****AI/ML & Biometrics****

* deepface – High-level API for facial analysis and verification.
* face\_recognition – For generating facial embeddings and comparison.
* dlib – Used under the hood by face\_recognition for facial landmark detection.
* tensorflow.keras – For loading and running custom trained deep learning models.
* mtcnn – For face detection (Multi-task Cascaded Convolutional Networks).
* mediapipe – Used for additional facial landmark detection or gesture recognition.

#### ****Image & Video Processing****

* cv2 (OpenCV) – For real-time face capture and image processing.
* numpy – Numerical computations and image data handling.
* base64 – For encoding/decoding images to/from base64 format.

#### ****Device & Browser Tracking****

* device\_fingerprint – Custom/local module used for device fingerprinting.

#### ****Utilities & System****

* os, time, datetime, subprocess, threading, webbrowser – Built-in Python modules used for file handling, process control, threading, and browser control.

#### ****Database Abstraction****

* database, database.crud, database.db\_utils – Custom modules for DB operations like saving embeddings, hashes, and user info.

### ****Steps to Deploy and Execute the Solution (For Testers/Users)****

#### ****1. Clone the Repository****

Get the project files:

git clone https://github.com/your-username/your-repo-name.git

cd your-repo-name

#### ****2. Create & Activate Virtual Environment****

To manage dependencies:

python -m venv venv

source venv/bin/activate # Windows: venv\Scripts\activate

#### ****3. Install All Dependencies****

Install required Python packages:

pip install -r requirements.txt

#### ****4. Set Up Environment Variables****

Create a .env file in the root directory and add your own credentials:

SECRET\_KEY=your\_own\_secret\_key

DATABASE\_URL=your\_own\_database\_connection\_url

MAIL\_USERNAME=your\_email@example.com

MAIL\_PASSWORD=your\_email\_password

**Note:** You must use your own values. The developer has not shared live credentials for security reasons.

#### ****5. Initialize the Database****

Make sure the PostgreSQL database is created and then run:

python -m database.db\_utils

#### ****6. Run the Flask App****

Start the server:

python main.py

#### ****7. Open the App in a Browser****

Visit:

http://127.0.0.1:5000/

Or it may open automatically depending on your system.

#### ****8. Test the Authentication Flow****

* Register a user with facial or fingerprint data.
* Log in using the registered biometric and device.
* If email verification is enabled, check your inbox.
* Observe anomaly detection and user tracking.

### Requirements Before Running:

* PostgreSQL installed and running
* Python 3.10+ recommended
* Basic internet access for email functions (if using Gmail/SMTP)