

# **Developer Guide**

SecurePass AI

QR & Face Recognition Door Access System

Version 1.1.0

## **Table of Contents**

1. Introduction
1.1 Project Overview
2. Technology Stack
2.1 Admin Web App
2.2 User Mobile App
2.3 External Services
3. System Architecture
4. Development Setup
4.1 Prerequisites
4.2 Github Repositories
4.3 Install Dependencies
4.4 Environment Variables
4.5 Run Locally
4.6 Setup SuperAdmin credentials 8
5. Project Structure 9
5.1 Admin Web App
5.2 User Mobile App9
6. API Documentation
6.1 External Face Recognition APIs
6.2 Internal APIs
7. Security Practices 11
8. Deployment 12
8.1 Server Setup
8.2 Caddyfile Configuration
8.3 Deployment Script
9. Testing & Debugging
10. Developers

## 1. Introduction

## 1.1 Project Overview

SecurePass AI is a browser-based door access system that combines QR code scanning and facial recognition to grant or deny entry.

- User Mobile App: Allows users to register faces, scan QR codes, and unlock doors.
- Admin Web App: Manages users, doors, access logs, and system configurations.
- Multi-Tenancy: Super Admins can create Admins, who manage specific tenants/doors.

### **Key Features**:

- Face registration and verification via external APIs.
- QR code generation and validation.
- Real-time door access logs using MQTT/Socket.io.
- Role-based access control (Super Admin, Admin, User).

## 2. Technology Stack

## 2.1 Admin Web App

Component	Tools/Libraries
Frontend	React, Redux, Tailwind CSS, Axios, react-toastify, Vite
Backend	Node.js, Express.js, MongoDB, Mongoose, Bcrypt, JWT, nodemon (dev)

## 2.2 User Mobile App

Component	Tools/Libraries
Frontend	React, face-api.js, react-qr-reader, MQTT, Tailwind CSS, Vite
Backend	Node.js, Express.js, MongoDB, Mongoose, qrcode-reader, moment-timezone

## 2.3 External Services

- Face Recognition: <u>iEntrada API</u> for face registration/verification.
- MQTT Broker: Real-time communication for door access events.
- **Database**: MongoDB Atlas (or self-hosted).

## 3. System Architecture

### 1. User App:

- $\circ$  Scans QR code or captures face  $\rightarrow$  Sends request to backend.
- Backend validates access → Triggers door unlock via MQTT.

## 2. Admin App:

- o Manages users/doors → Updates MongoDB.
- o Real-time logs via Socket.io.

### 3. **APIs**:

- o Internal APIs for user/door management.
- o External iEntrada APIs for face recognition.

## 4. Development Setup

## 4.1 Prerequisites

- Node.js  $\geq 14.x$
- MongoDB (local or Atlas URI).
- Git.

### 4.2 Github Repositories

### **Admin Web App**:

https://github.com/SLTDigitalLab/Secure-Pass-AI-Admin-App.git

#### **User Mobile App:**

https://github.com/SLTDigitalLab/Secure-Pass-AI-User-App.git

## **Clone Repositories**

#### **Admin Web App:**

```
git clone https://github.com/SLTDigitalLab/Secure-Pass-AI-Admin-App.git
```

### **User Mobile App:**

```
git clone <a href="https://github.com/SLTDigitalLab/Secure-Pass-AI-User-App.git">https://github.com/SLTDigitalLab/Secure-Pass-AI-User-App.git</a>
```

## 4.3 Install Dependencies

#### **Admin Frontend:**

```
cd Secure-Pass-AI-Admin-App /frontend && npm install
```

### Admin Backend:

cd Secure-Pass-AI-Admin-App /backend && npm install

#### **User Frontend:**

```
cd Secure-Pass-AI-User-App /frontend
npm install react-qr-reader --force && npm install
```

#### **User Backend:**

cd Secure-Pass-AI-User-App /backend && npm install

#### **4.4 Environment Variables**

#### **Admin Frontend (frontend/.env):**

VITE\_API\_URL=http://localhost:5000 # Admin backend URL

#### Admin Backend (backend/.env):

PORT=5000

MONGODB\_URI=mongodb://localhost:27017/admin\_db

JWT\_SECRET=your\_jwt\_secret

FRONTEND\_URL=http://localhost:3000

#### **User Frontend (frontend/.env):**

VITE\_API\_URL=http://localhost:8080

VITE\_MQTT\_URL=mqtt://your\_broker\_url

VITE\_API=ientrada\_api\_key

### User Backend (backend/.env):

MONGO\_URL=mongodb://localhost:27017/user\_db

JWT\_SECRET=your\_jwt\_secret

PORT=8080

## 4.5 Run Locally

#### **Admin Web App:**

# Frontend

cd Secure-Pass-AI-Admin-App /frontend && npm run dev # http://localhost:3000

# Backend

cd Secure-Pass-AI-Admin-App /backend && npm run dev # http://localhost:5000

### **User Mobile App:**

# Frontend

 $\verb|cd Secure-Pass-AI-User-App| \textit{frontend \&\& npm run dev} \# \verb| http://localhost:3001| \\$ 

# Backend

cd Secure-Pass-AI-User-App /backend && npm start # http://localhost:8080

## 4.6 Setup SuperAdmin credentials

• Go to scripts folder

## cd /backend/scripts

After setup the mongodb first need to create the super admin account

• Run the script

### node createSuperAdmin.js

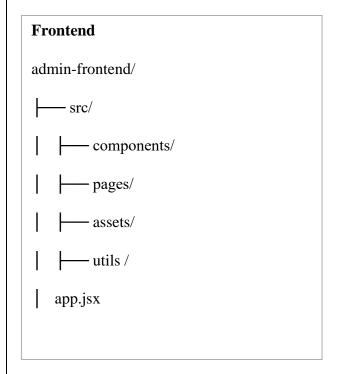
• SuperAdmin Initial Credentials

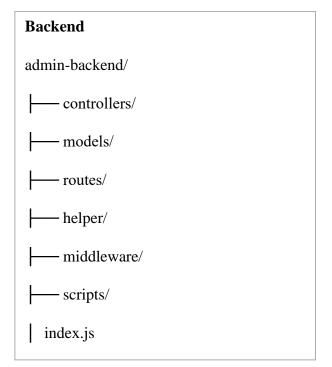
Email: superadmin@gmail.com

Password:

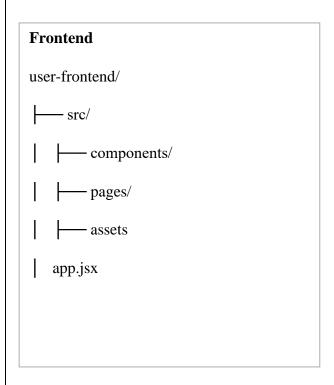
## 5. Project Structure

## 5.1 Admin Web App





## **5.2 User Mobile App**



Backend
user-backend/
— controllers/
— models/
helper/
— middleware/
Lroutes/
index.js

## 6. API Documentation

## **6.1 External Face Recognition APIs**

### 1. Face Registration

```
POST https://ientrada.raccoon-ai.io/api/register_face

Headers: { Authorization: "Bearer <API_KEY>" }

Body: { image: "base64_encoded_image", user_id: "123" }

Response: { success: true, face_id: "abc" }
```

#### 2. Face Verification

```
POST https://ientrada.raccoon-ai.io/api/verify_face

Headers: { Authorization: "Bearer <API_KEY>" }

Body: { image: "base64_encoded_image", face_id: "abc" }

Response: { match: true, confidence: 0.95 }
```

### **6.2 Internal APIs**

#### **User Login (Admin Backend):**

```
POST /api/auth/login

Body: { email: "admin@securepass.com", password: "..." }

Response: { token: "jwt_token", role: "superadmin" }
```

#### **Door Access Logs (User Backend):**

```
GET /api/access/logs

Headers: { Authorization: "Bearer <JWT>" }

Response: [ { door_id: "1", timestamp: "2024-01-01T12:00:00Z" } ]
```

/ · Decuilly illactices	7.	<b>Security</b>	<b>Practices</b>
-------------------------	----	-----------------	------------------

- 1. Authentication: JWT tokens with role-based access (Super Admin, Admin, User).
- 2. **Password Hashing**: Bcrypt (salt rounds: 10).
- 3. **CORS**: Whitelisted domains (Admin: http://localhost:3000, User: http://localhost:3001).
- 4. **Secrets Management**: .env files excluded from Git.

## 8. Deployment

## 8.1 Server Setup

1. VM Access:

```
ssh root@178.128.20.26
# Password: Use provided credentials
```

### 2. Directory Structure:

```
/SecurePass

— Secure-Pass-AI-User-App

— Secure-Pass-AI-Admin-App
```

## 8.2 Caddyfile Configuration

Go to the Directory/etc/caddy/Caddyfile

Edit Caddyfile
 nano Caddyfile

• Caddyfile Code

```
securepass.sltdigitallab.lk {
    reverse_proxy localhost:3001 #User App
}
admin.securepass.sltdigitallab.lk {
    reverse_proxy localhost:3000 #Admin App
}
```

## 8.3 Deployment Script

## 1. **Update Code**:

Update from Github

git pull origin main

Username : SLTDigitalLab

Password: Use PAT for authentication

## 2. Run Deployment:

Dockerization and Contanerization Process, Build Process and Deployment process are scheduled in deploy.sh file. For the deployment run in the root folder of the Application

./deploy.sh

## 9. Testing & Debugging

#### **Common Issues:**

- Face API Errors: Ensure api key is valid and images are base64-encoded.
- MQTT Connection Failed: Check broker URL in .env.
- **CORS Errors**: Verify backend cors() middleware allows frontend URLs.

#### Logs:

- Admin Backend: npm run dev (nodemon logs).
- User Frontend: Browser DevTools console.

## 10. Developers

- Chathura Dissanayake
- Mohamed Yoosuf Aathil
- Shavindu Rajapaksha
- Mohamed Afraar