SAMS DOCUMENTATION

1. Project Overview
2. Introduction
3. Requirements
4. Architecture and Design
5. Implementation Details
6. Development Approach
7. Database Design
8. API and Services
9. Testing and Quality Assurance
10. Deployment Plan
11. Security
12. Challenges and Risks
13. Team and Roles

SAMS (Smart Attendance Management System)

---

1. Project Overview

- **Project Title**: Smart Attendance Management System (SAMS)
- **Abstract/Summary**:
  SAMS is a QR code and facial recognition-based attendance system that ensures accurate and tamper-proof attendance tracking. It integrates geolocation, facial verification, and QR code scanning to eliminate proxy attendance and improve administrative efficiency.
- **Objectives**:
  - Automate the attendance process.
  - Prevent proxy attendance using facial recognition.
  - Validate attendance using geolocation.
  - Provide admins with real-time attendance data.
- **Scope**:
  - Used in educational institutions or workplaces.
  - Admins can manage attendance data.
  - Students can mark attendance using QR codes and facial verification.

---

2. Introduction

- **Background and Context**:
  Manual attendance systems are prone to errors and misuse, such as proxy attendance. To overcome these challenges, SAMS integrates QR code scanning, geolocation tracking, and facial recognition to ensure secure and efficient attendance management.
- **Problem Statement**:
  Traditional attendance systems are inefficient, time-consuming, and vulnerable to tampering.
- **Purpose and Significance**:
  SAMS streamlines attendance management, improves accuracy, and ensures real-time monitoring.

---

3. Requirements

Functional Requirements

1. Admin Login and Dashboard.
2. QR Code Generation for Attendance.
3. Student Login and Dashboard.
4. QR Code Scanning for Attendance.
5. Geolocation Validation.
6. Facial Recognition Verification.
7. Real-time Attendance Data for Admins.

Non-Functional Requirements

1. System should respond within 2 seconds.
2. Facial recognition accuracy > 95%.
3. Data stored securely in MongoDB.
4. Cross-platform compatibility (web browsers).

System Requirements

- **Software**:
  - OS: Windows, Linux, macOS
  - Backend: Django REST Framework
  - Frontend: React.js
  - Database: MongoDB Atlas
- **Hardware**:
  - Camera-enabled device (mobile/laptop).
  - Internet connection.

Stakeholder Requirements

- **Admins**: Manage attendance data.
- **Students**: Mark attendance securely.

---

4. Architecture and Design

System Architecture:

- **Admin Module**: Generates QR codes and views attendance records.
- **Student Module**: Scans QR codes, validates geolocation, and undergoes facial recognition.

Detailed Design:

![Flowchart](Insert the flowchart image here)

Technology Stack

- **Frontend**: React.js, html5-qrcode, Tailwind CSS, Material-UI.
- **Backend**: Django, REST Framework, qrcode.

- **Database**: MongoDB Atlas.
- **Facial Recognition**: OpenCV, DeepFace.
- **Geolocation**: Browser Geolocation API.

---

5. Implementation Details

Modules and Components

1. **Admin Dashboard**:
   - QR Code Generation
   - Attendance Reports
2. **Student Dashboard**:
   - Scan QR Code
   - Geolocation Validation
   - Facial Recognition Verification

Features and Functionalities

1. Generate time-sensitive QR codes.
2. Fetch and validate student location.
3. Real-time face verification.
4. Admin access to attendance data.

Integration Details

- QR Code Scanning: `` `html5-qrcode` ``
- Facial Recognition: OpenCV + DeepFace
- Location Validation: Geolocation API + Haversine

Key Algorithms

1. **Haversine Formula**: Validates geofencing (location within a radius).
2. **Facial Matching**: Compares live face data with stored embeddings.

---

6. Development Approach

- **Methodology**: Agile Development
- **Tools Used**:
  - Version Control: GitHub
  - Testing: Postman (API Testing)
- **Milestones**:
  1. UI/UX Development
  2. QR Code and Geolocation Implementation
  3. Facial Recognition Integration
  4. Testing and Deployment

---

7. Database Design

Database Schema

Collection/Table	Fields	Purpose
Users	Name, Email, Role, Face Embedding	Stores user data.
Attendance	UserID, Date, Time, Location	Stores attendance records.

---

8. API and Services

API Endpoints

Endpoint	Method	Purpose	Parameters
`` `/api/login` ``	POST	User Authentication	Email, Password
`` `/api/generate_qr` ``	GET	Generate QR Code	-
`` `/api/validate_attendance` ``	POST	Validate Attendance (QR + Face)	Location, Face Data
`` `/api/attendance_data` ``	GET	Fetch Attendance Records	Date, UserID

---

9. Testing and Quality Assurance

- **Tools Used**:
  - Postman for API Testing
  - Jest for Frontend Testing
- **Test Cases**:
  - Verify QR Code Expiry.
  - Validate Geolocation Radius.
  - Test Facial Recognition Accuracy.

---

10. Deployment Plan

- **Frontend**: Vercel (React Deployment).
- **Backend**: Render.com or Heroku (Django Deployment).
- **Database**: MongoDB Atlas.

---

11. Security

1. **Authentication**: JWT Authentication for secure user login.
2. **Data Encryption**: Encrypt attendance data in MongoDB.
3. **Proxy Prevention**: Facial recognition + geolocation ensures no proxy attendance.

---

12. Challenges and Risks

Challenge	Mitigation Strategy
Facial recognition accuracy	Use high-quality training data.
Geolocation spoofing	Validate multiple factors (QR + Face).

Challenge	Mitigation Strategy
Network issues	Implement offline QR code caching.

13. Team and Roles

Team Member	Role	Contact Information
Hari	Developer/Project Lead	hariharanpugazh@gmail.com
Ajay	Co-Developer	ajaychakaravathi2004@gmail.com

Credits To ME :)

Thanks For Reading