Smart Multi-Layered Attendance Verification System (SMAVS)

Development Breakdown Document

1. Overview

This document outlines all development tasks and modules for the Smart Multi-Layered Attendance Verification System (SMAVS). The project will use FastAPI for the backend and React Native for the mobile frontend. The system is designed to prevent impersonation and proxy attendance through biometric, device, and code-based verification.

2. Backend Development (FastAPI)

The backend forms the system's core, responsible for user authentication, verification logic, data management, and API communication.

Tasks:
• Setup FastAPI project with PostgreSQL integration.
• Design database models: User, Course, AttendanceSession, AttendanceRecord, FlaggedRecord, Device, VerificationLog.
• Implement JWT-based authentication and role-based access control.
• Create REST APIs:
- Auth APIs: Register, Login, Logout, Refresh Token.
- Attendance APIs: Generate code, Submit attendance, Verify data, Retrieve reports.
- Verification APIs: Handle selfie, back-camera capture, and IMEI verification.
- Admin APIs: Manage users, approve IMEI resets, monitor flagged sessions.
• Integrate image and biometric verification using OpenCV and TensorFlow Lite.
• Implement code generation and expiration logic.
• Secure all endpoints with HTTPS and data encryption (AES-256).
• Deploy backend using Docker on AWS or GCP.

3. Frontend Mobile App (React Native)

The unified mobile app will serve both students and lecturers, with role-based dashboards and real-time API communication.

Tasks:
• Setup React Native project and connect it to FastAPI backend.
• Implement JWT authentication flow (login, signup, token refresh).
• Role-Based UI:
- Student Dashboard: Code entry, selfie capture, IMEI check, attendance confirmation.
- Lecturer Dashboard: Generate codes, view attendance status, handle flagged cases.
• Integrate device camera APIs for front and back captures.
• Retrieve device IMEI and bind it to user profile.
• Implement offline mode with background sync.
• Add push notifications for mid-lecture verifications.
• Ensure responsive and user-friendly interface.

4. Biometric Verification Module

Handles face recognition and liveness detection to ensure identity verification integrity.

Tasks:
• Implement server-side face matching using OpenCV.
• Use TensorFlow Lite model for facial feature extraction.
• Optionally integrate AWS Rekognition for enhanced accuracy.
• Add liveness detection (blink/smile recognition).
• Create /verify-face endpoint for mobile app.
• Return JSON result with match status and confidence score.

5. Admin Web Dashboard

A web-based platform for university administrators to manage users, view logs, and oversee flagged sessions.

Tasks:
• Build with React.js or Next.js.
• Implement login and JWT-based authorization.
• Features:
- Manage students and lecturers.
- Approve IMEI reset requests.
- View attendance analytics and export reports.
- Review and clear flagged attendance sessions.
• Add visual charts for attendance trends.
• Host on Vercel or Netlify.

6. Testing & Deployment

Ensures all components function as expected and integrates smoothly.

Tasks:
• Backend: Unit, integration, and performance testing.
• Frontend: UI and functional testing (Jest / Detox).
• Biometric: Accuracy and false-positive testing.
• End-to-end testing for full attendance flow.
• Continuous integration and delivery pipeline setup.
• Deploy FastAPI backend on AWS (EC2 or App Runner) and PostgreSQL on RDS.
• Publish React Native app on Play Store and TestFlight.

7. Documentation & Maintenance

• Generate API documentation using Swagger/OpenAPI.
• Create setup and usage guides for developers.
• Write user manuals for students, lecturers, and admins.
• Set up monitoring and maintenance routines for uptime and logs.