Offline LAN-Based Attendance System

1. Purpose of This Document

This document proposes a **clear implementation workflow and system architecture** for an offline / local-network-based school attendance system. It is intended for **technical experts or developers** who may take over, extend, or productionize the system in the future. The focus is on **continuity, scalability, and clarity of intent**, not just the MVP.

The system is designed to:

- Work **without internet access**
- Operate within a **shared local network (LAN)**
- Enforce **one student → one device → one attendance record per session**
- Support future upgrades (cloud sync, biometrics, analytics, etc.)

---

2. Problem Context & Motivation

Traditional attendance methods (manual roll-call, paper sheets, centralized online systems) suffer from:

- Time inefficiency
- Proxy attendance (impersonation)
- Dependence on internet connectivity
- Poor auditability

This project explores a **network-based presence verification model**, inspired by basic networking principles (shared subnet, device discovery, message passing), adapted into a structured and authenticated attendance workflow.

---

3. Core Design Principles

1. **Offline-first**: The system must work fully on a local network.
2. **Device-bound identity**: Attendance is tied to a registered device.
3. **Session-based validation**: Attendance is valid only during an active class session.
4. **Low friction for students**: Minimal steps during class time.
5. **Instructor authority**: Lecturer controls session lifecycle and overrides.

6. **Extensibility**: Architecture must allow future modules without redesign.

---

4. System Actors

4.1 Students

- Use a **phone or laptop** connected to the classroom network
- Authenticate once (enrollment)
- Perform a single tap/click per class to mark attendance

4.2 Lecturer / Host

- Runs the **host server** (laptop recommended)
- Starts and ends attendance sessions
- Views real-time attendance status

4.3 System (Software Components)

- Local server (attendance authority)
- Client applications (student devices)
- Local storage (buffer + database)

---

5. High-Level Architecture

5.1 Network Layer

- All devices connect to the **same router / hotspot**
- Same subnet (e.g. 192.168.x.x)
- LAN communication only (no WAN dependency)

5.2 Communication Model

Two viable methods are supported:

Method A — WebSocket (Recommended)

- Persistent connection between client and host
- Reliable delivery
- Real-time feedback

- Supports authentication and acknowledgements

Method B — UDP Broadcast (Experimental / MVP)

- Clients broadcast attendance packets
- Host listens and buffers
- Lower reliability, minimal setup

The architecture favors **WebSocket for production**, UDP for early demos.

---

6. Identity & Authentication Model

Authentication is **layered**, not singular.

6.1 Student Identity

- Student ID / matric number
- Optional PIN or password

6.2 Device Binding (Critical)

- On first enrollment, client generates a **Device UUID**
- UUID is stored locally and sent to server
- Server binds:
  `student_id ↔ device_uuid`
- A student can only check in using the bound device

6.3 Session Authentication

- Each class creates a **session token**
- Token is time-bound and class-specific
- Distributed via:
  - QR code
  - LAN discovery

---

7. Proposed Workflow

7.1 Initial Enrollment (Once per Semester)

1. Student connects to school network

2. Opens client app / web page
3. Enters student ID (+ optional PIN)
4. Client generates device UUID
5. Server validates and stores binding
6. Enrollment completed

This phase is controlled and supervised.

7.2 Attendance Session Workflow (Per Class)

Lecturer Actions

1. Launch host server
2. Start new attendance session
3. Session token becomes active
4. QR code or session info displayed

Student Actions

1. Connect to same network
2. Open client app
3. Tap "Check In"
4. Client sends:
    - student_id
    - device_uuid
    - session_token

Server Validation

- Session active?
- Token valid?
- Device matches student?
- Student not already marked?

If all pass → attendance recorded

7.3 Buffering & Storage

- Incoming attendance records are stored in:
    - In-memory buffer (fast)
- On session end:
    - Records are committed to local database

This ensures resilience against crashes or sudden power loss.

---

8. Data Model (Conceptual)

Student Table

- student_id
- name
- device_uuid
- enrollment_timestamp

Session Table

- session_id
- course_code
- lecturer_id
- start_time
- end_time

Attendance Table

- attendance_id
- student_id
- session_id
- timestamp
- status (present / late / flagged)

---

9. Security & Anti-Abuse Measures

- One device per student enforcement
- Session expiration
- Duplicate submission rejection
- Optional heartbeat to ensure presence
- Manual override by lecturer

Limitations are acknowledged (e.g. physical phone sharing), mitigated by supervision and device binding. The system prevents digital impersonation but cannot independently verify the physical identity of the device holder. Institutional supervision and optional biometric extensions mitigate this.

---

10. MVP Technology Stack (Suggested)

- **Backend:** Flask + Flask-SocketIO
- **Frontend:** Mobile-first web app (HTML/CSS/JS)
- **Database:** SQLite (local) → PostgreSQL (future)
- **Networking:** LAN WebSocket

---

11. Microcontroller (MCU) Integration

11.1 Rationale for MCU Integration

To strengthen system reliability, physical presence validation, and long-term scalability, this system incorporates a **microcontroller-based classroom node** as part of the core architecture.

The microcontroller does **not replace the server**. Instead, it acts as a **hardware-backed authority and gateway**, ensuring that attendance sessions are tied to a real physical classroom environment.

This mirrors established IoT and industrial system designs, where:

- Hardware enforces trust and presence
- Software handles logic, data, and extensibility

11.2 Role of the Microcontroller

The MCU is responsible for:

- Acting as the **classroom identity anchor**
- Initiating and terminating attendance sessions
- Providing hardware-level confirmation that a class is active
- Interfacing with peripherals (buttons, LEDs, RFID/NFC, buzzers)
- Maintaining a lightweight fallback buffer during server downtime

The MCU explicitly does **not** handle:

- Student authentication logic
- Device-to-student binding

- Database storage
- Complex cryptographic operations

These remain the responsibility of the server.

11.3 Recommended MCU Platform

While early prototypes may use basic microcontrollers, production deployment favors Wi-Fi–enabled MCUs:

- **ESP32 (Recommended)**
  - Built-in Wi-Fi
  - Sufficient RAM for HTTP/WebSocket communication
  - Stable ecosystem and low cost
- **Raspberry Pi Pico W (Alternative)**
  - Wi-Fi capable
  - Slightly higher complexity

Basic microcontrollers such as Arduino Uno are suitable only for **peripheral control** in early MVPs.

11.4 MCU–Server Communication Model

The MCU communicates with the local server over the same LAN using:

- HTTP (REST) for simplicity
- WebSocket for real-time session synchronization

Example responsibilities include:

- Notifying the server when a session starts or ends
- Confirming session validity when queried
- Uploading buffered attendance data after recovery

11.5 Session Authority Workflow with MCU

1. Lecturer presses a physical button on the MCU
2. MCU generates a new `session_id`
3. MCU signals the server that a session is active
4. Server accepts attendance submissions only for that session

5. MCU ends session → server locks attendance records

This ensures attendance cannot be started or manipulated remotely.

11.6 Offline Resilience & Fallback Mode

If the server becomes temporarily unavailable:

- MCU continues to accept basic attendance signals
- Stores them in a limited local buffer
- Flushes buffered records to the server once connectivity is restored

This guarantees session continuity even under partial system failure.

---

12. Future Continuity & Extensions

This system is intentionally modular to allow:

- Cloud synchronization (post-session)
- Multi-classroom support
- Biometric confirmation
- Bluetooth proximity verification
- Analytics & reporting dashboards
- University-wide deployment

The LAN-based MVP acts as the **foundation**, not the final form.

---

12. Summary

This proposed system translates basic networking principles into a structured, authenticated attendance platform. By combining **local networking**, **device binding**, and **session-based validation**, it achieves a balance between practicality, security, and scalability.

The design prioritizes continuity — enabling future experts to extend the system without re-architecting its core.