Project Design Document

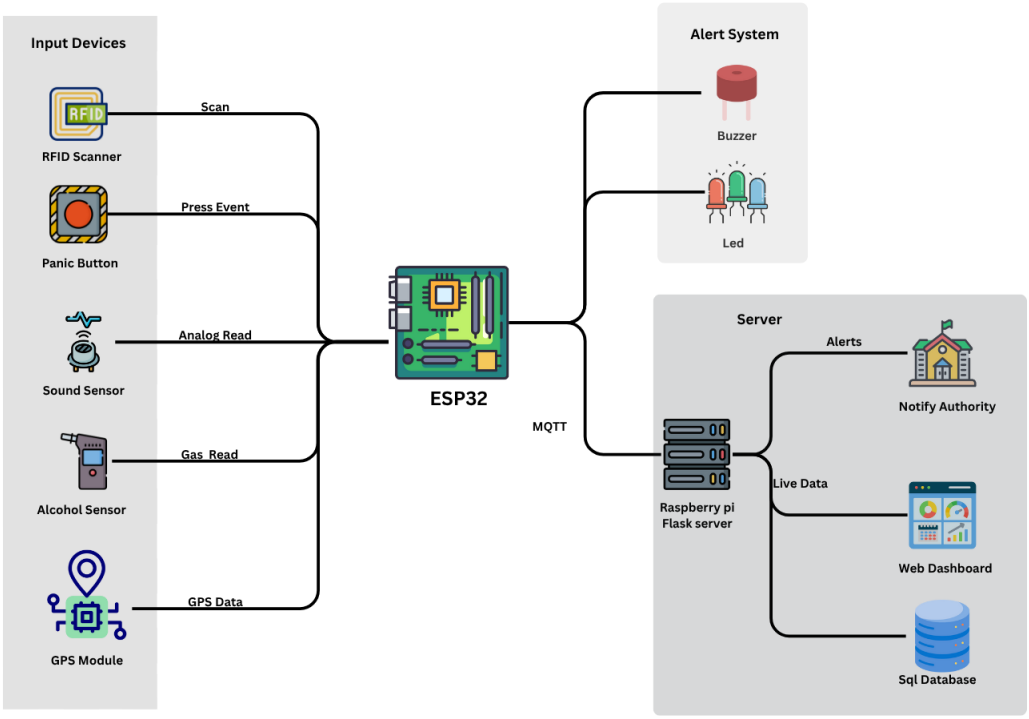
1. Title of the Project

SHE Secure (Safety, Help, Emergency)

2. Objective

- To design and implement an IoT-based system that enhances women's safety during college bus commutes.
- Restricted entry, ensuring only authorized individuals can board using RFID-based authentication or simulation.
- To ensure real-time detection and rapid response to emergencies through Panic buttons, Alcohol sensors, High-Pitch Noise Detection, and Route deviation monitoring, triggering alerts via a centralized Master Alert System.
- To implement a live web dashboard for authorities to monitor student entries, location updates, and safety alerts—empowering institutions with visibility and control over on-board safety in real time.

3. System Diagram/Architecture

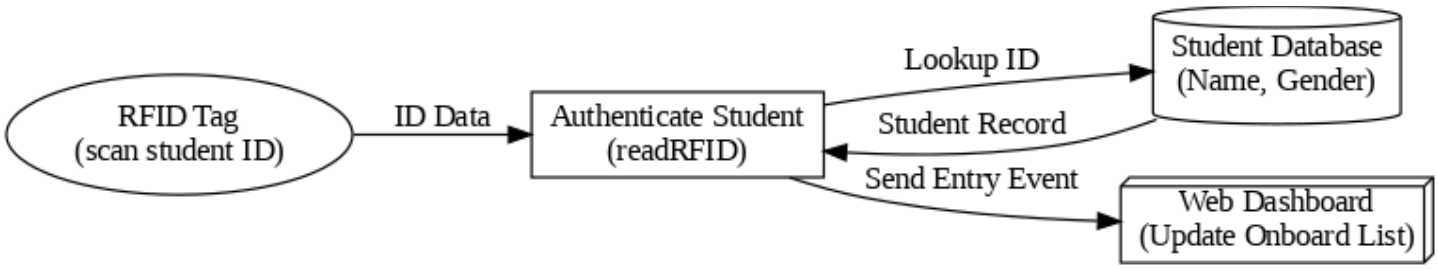


4. Data Flow Diagram (DFD) for Each Function

The following diagrams illustrate how data flows through various modules of the SHE Secure system. Each DFD shows the interaction between inputs (sensors or users), processing units (ESP32s), data stores (server/database), and outputs (dashboard or alert systems).

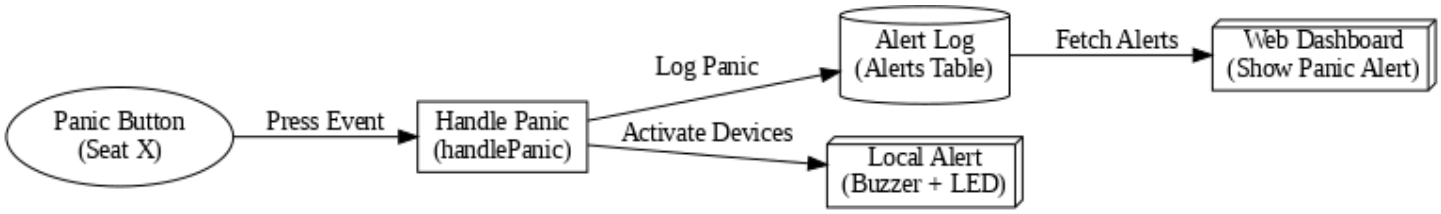
a) Student Authentication

Description: When a student scans their RFID card, the ESP32 controller reads the input and verifies it through the Flask server. The student's data is fetched from the database, and their entry is logged and shown on the web dashboard.



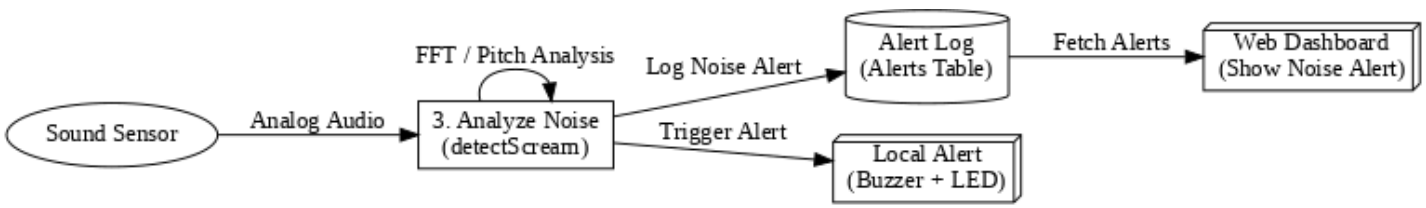
b) Panic Button Alert

Description: When a panic button is pressed by any student, the ESP32 detects the input and triggers a buzzer and alert LED. It also sends the alert to the server, where it's logged and displayed in real time on the dashboard.



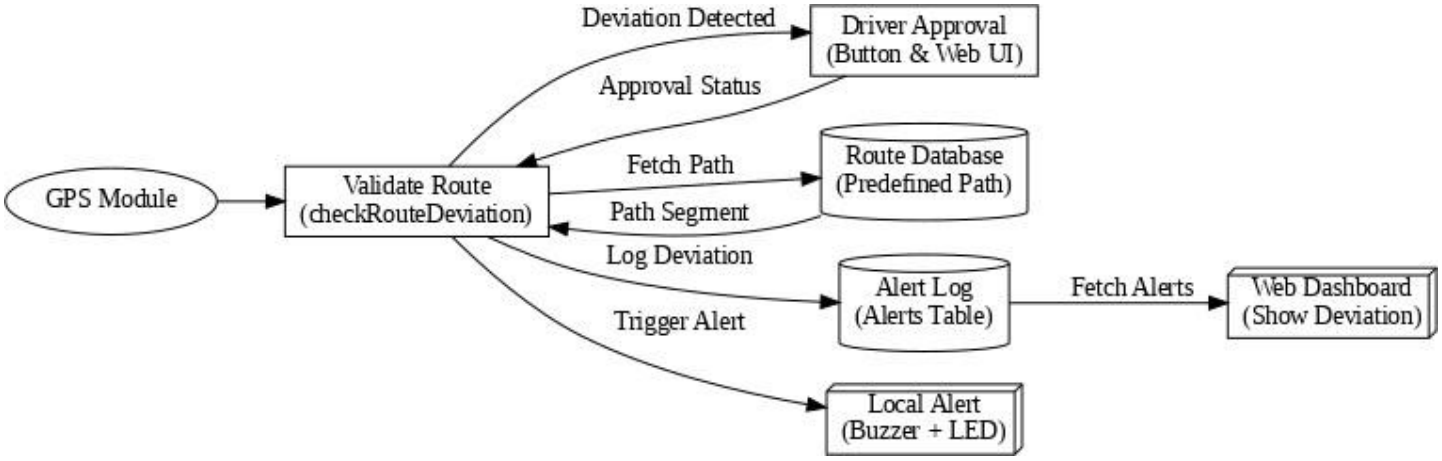
c) Noice (Scream) Detection

Description: The sound sensor continuously monitors for high-pitch noise using FFT on the ESP32. If a scream is detected, the system triggers a local alert and logs the event on the server, which is reflected on the monitoring dashboard.



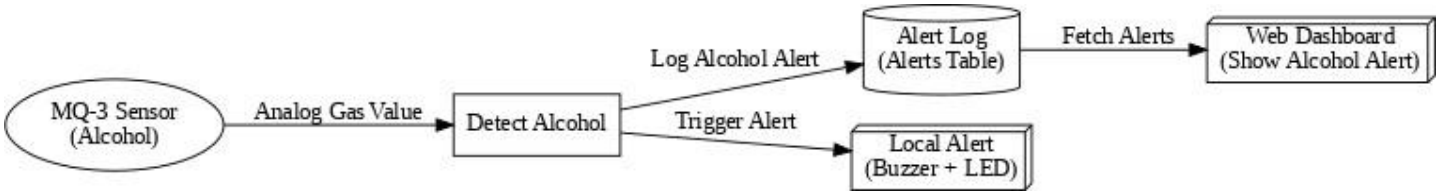
d) Route Deviation Detection

Description: GPS data from the bus is monitored and compared to a predefined path. If the bus deviates without approved confirmation, an alert is logged and shown on the dashboard, and optionally triggers emergency actions.



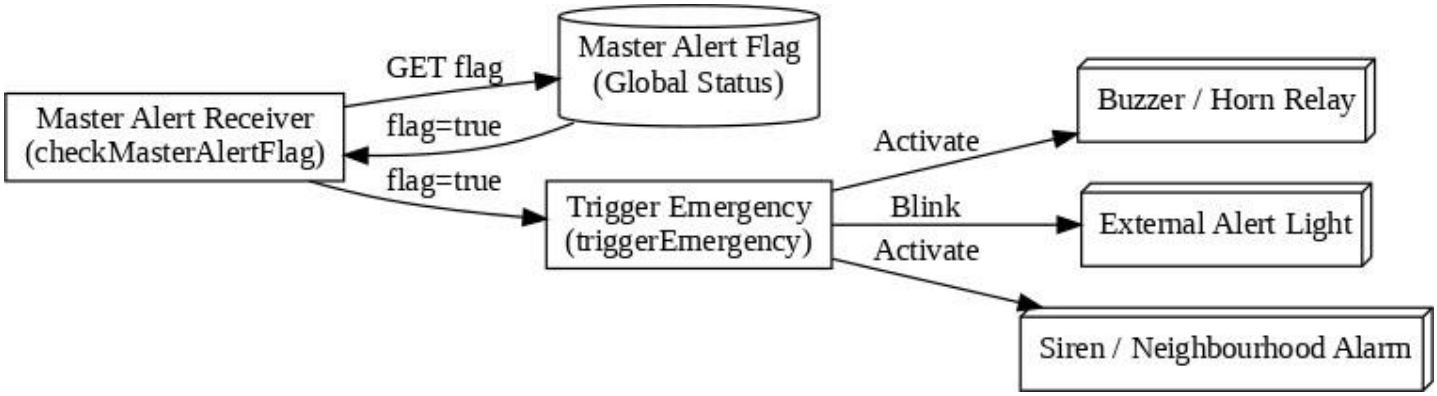
e) Alcohol Detection

Description: The MQ-3 sensor detects alcohol levels in the driver's and student's breath. If the concentration exceeds a safe threshold, the ESP32 triggers local alerts and notifies the server to log the event and inform authorities via the dashboard.



f) Master Alert System

Description: If any emergency is detected (panic, noise, alcohol, or deviation), the Flask server sets a master alert flag. An ESP32 polls this flag and activates external devices like the siren, buzzer, and alert light, ensuring nearby people and authorities are aware.



5. Programming Functions and Complete Algorithm

I. **Function 1**: readRFID()

Purpose: Reads and verifies student ID using RFID module.

Input: RFID Tag.

Output: Student Details sent to server.

II. **Function 2**: sendToServer(data)

Purpose: Sends structured JSON payload to Flask/Raspberry pie server.

Input: Event type and data.

Output: MQTT Response.

III. **Function 3**: triggerLocalAlert()

Purpose: Activates buzzer, LED, and optional siren.

Input: Alert Condition met.

Output: Audible (Buzzer) / Visible (Led) Signals.

IV. **Function 4**: detectScream()

Purpose: Uses sound sensor + FFT to detect high-pitch scream.

Input: Microphone analog data.

Output: Trigger for scream alert.

V. **Function 5**: checkRouteDiversion()

Purpose: Compares GPS data to predefined route.

Input: Current latitude/longitude from GPS module.

Output: Triggers alert if off-route and not approved.

VI. **Function 6**: detectAlcohol()

Purpose: Checks MQ-135 analog value against threshold.

Input: Concentration of alcohol/ethanol vapors.

Output: Trigger alert if alcohol level above threshold detected.

VII. **Function 7**: cehckMasterAlertFlag()

Purpose: Periodically polls server for master alert status.

Input: MQTT, GET

Output: Trigger buzzer/led if alert is True.

Algorithm:

1. Initialize microcontroller and sensor.
2. Connect to Wi-Fi network.
3. Continuously read data from input sensors.
4. On trigger Condition:
    - Send structured alert/event to server.
    - Activate local alert devices (buzzer, LEDs).
    - It will notify the authorities.
5. Dashboard updates in real-time using server endpoints.
6. Master Alert ESP32 polls server to activate external alarm systems if necessary.
7. Data being updated on the database.

7. Equipment Required

S. No.	Component	Specification	Quantity	Availability in Lab (Yes/No)
1	ESP32-C6	Wi-Fi Microcontroller	2	Yes
2	RFID Reader	RFID scanning	1	No
3	Push Button	2 Pin Tactile Micro Switch	2	Yes
4	Sound Sensor	MAX9814	1	No
5	MQ-3 Sensor	Alcohol sensor	1	No
6	GPS Module	Neo-6M	1	No
7	16x2 Display	Display with I2C module	1	Yes
8	Led	Red, Green, Yellow	4	Yes
9	Buzzer Module		2	Yes
10	Raspberry Pi 5	Model B 8GB RAM	1	Yes
11	MicroSD Card	32GB Class10	1	Yes

12	USB Cable + Power Source	Power supply to ESP8266	1	Yes
13	Jumper wire	Male to Female Female to Female Male to Male	12 each	Yes
14	Breadboard		2-3	Yes
15	Resistor/potentiometer	1k, 10k	5 each	Yes