Based on the "Smart Helmet" Project Report provided, here is the detailed technical breakdown and testing plan.

1. Feature Definition

Application-Level Features (Web Dashboard / Cloud)

These features are user-facing, designed for fleet managers or safety supervisors to monitor the driver's status remotely.

- **Real-Time Dashboard:** Visualizes live driver status, active alerts, and connectivity status.

+2

- **Event Log & Replay:** Lists safety events (fatigue, distraction, crash) with timestamps and allows playback of the associated 10-20 second video clips.

+3

- **Driver Scoring System:** Calculates and displays safety scores and performance trends based on historical data.

+1

- **Data Visualization:** Graphs trending metrics for PERCLOS (fatigue) and distraction events over time.
- **User Management:** Administrative interface to manage driver profiles and device assignments.
- **Privacy Controls:** Tools to manage data retention, handle Data Subject Access Requests (DSAR), and toggle "events-only" upload modes.

On-Board Features (Embedded / Smart Helmet)

These features run locally on the Raspberry Pi Zero 2 W to ensure low-latency safety monitoring.

- **Fatigue Detection:** Real-time calculation of PERCLOS (Percentage of Eyelid Closure), Eye Aspect Ratio (EAR), and blink rate using the inward-facing camera.

+1

- **Distraction Detection:** Monitors head pose (pitch/yaw/roll) and gaze deviation to detect when the driver is looking away.

+1

- **Crash & Maneuver Detection:** Monitors the IMU for high-G spikes (>1g peaks) to detect accidents or harsh braking/cornering.

+1

- **Local Alerting:** Triggers immediate audio/visual warnings (buzzer or LED) when fatigue or distraction thresholds are breached (<200ms latency).

+1

- **Circular Video Buffering:** Continuously records video but only permanently saves and uploads the $\pm$10-20 seconds surrounding a triggered event.

+1

- **Smart Power Management:** Duty-cycling of IR LEDs and model quantization to maintain battery life for >8 hours.

+1

2. Hardware Architecture

Overall Architecture

The system is built around a **Raspberry Pi Zero 2 W** acting as the central processing unit. It interfaces with a **NoIR Camera** for vision, a **BMI160 IMU** for motion, and an **IR LED array** for night vision. The entire unit is powered by a portable 10,000mAh battery pack.

+1

Interfaces & Connectivity

Component	**Interface/Protocol**	**Connection Details**

Camera (NoIR V2)	**CSI-2** (Camera Serial Interface)	Connected directly to the Pi's CSI camera port via ribbon cable.
IMU (Bosch BMI160)	**I^2C** (Inter-Integrated Circuit)	Connected to GPIO pins (SDA/SCL). Uses interrupt pin (INT) to wake Pi on high-G events.
IR LED Array	**GPIO / PWM**	Controlled via a MOSFET driver connected to a GPIO pin. PWM is used for brightness control/duty-cycling. +1
Storage (MicroSD)	**SDIO**	Built-in slot; hosts the OS, local logs, and the circular video buffer.
Network	**Wi-Fi (802.11n)**	Built-in to Pi Zero 2 W; used for MQTT/HTTPS telemetry to the cloud.
Power Input	**Micro-USB**	Receives 5V DC from the battery bank.

Signal Direction & Control Logic

1. **Input:** Camera (Video Stream) $\rightarrow$ Pi (CSI Port).
2. **Input:** IMU (Accel/Gyro Data) $\rightarrow$ Pi (I^2C Bus).
3. **Output:** Pi (GPIO/PWM) $\rightarrow$ MOSFET $\rightarrow$ IR LEDs (Illumination).
4. **Output:** Pi (Wi-Fi) $\rightarrow$ Cloud Dashboard (MQTT/HTTPS).
5. **Output (Optional):** Pi (GPIO) $\rightarrow$ Buzzer/Status LED (Local Alert).

3. Power Supply Plan (Electrical Specifications)

Total Power Budget: ~2.0W - 3.5W (Average) **Target Runtime:** 8 Hours on 10,000mAh Battery

+1

Voltage & Current Requirements

Component	Voltage	Max Current	Source
Raspberry Pi Zero 2 W	5V DC	~0.8A (Peak) / 0.4A (Idle)	Battery Bank (USB)
NoIR Camera Module	3.3V (Internal)	~250mA	Powered via Pi CSI port
BMI160 IMU	3.3V	< 1mA (Low Power)	Powered via Pi 3.3V GPIO rail
IR LED Array (850nm)	5V DC	~500mA - 1A (Peak)	Direct from Battery (Switched by MOSFET)

Power Distribution Architecture

- **Primary Source:** 10,000mAh @ 5V Power Bank.
- **5V Rail:** Direct connection to Pi USB Power and the Source (Drain) of the IR LED MOSFET.
- **3.3V Rail:** Regulated locally by the Pi; supplies the IMU and logic levels for the MOSFET Gate.
- **Ground:** Common ground between Battery, Pi, IMU, and MOSFET.

Consumption Estimation

- **Pi + Camera (Inference Running):** ~2.3 Watts
- **IMU:** Negligible (~0.01 Watts)
- **IR LEDs (50% Duty Cycle):** ~1.0 Watts
- **Total Average: 3.3 Watts**
- **Capacity Check:** 10,000mAh @ 5V = 50Wh.
    - $50Wh / 3.3W \approx 15.1 \text{ hours}$ (Theoretical).
    - Accounting for regulator loss/efficiency (70%): $\approx 10.5 \text{ hours}$. This meets the 8-hour requirement safely.

4. Unit Testing Plan

These tests are designed to be run as Python scripts on the Raspberry Pi.

Part A: Individual Hardware Unit Tests

Test 1: IMU Sensor Verification (I2C)

- **Goal:** Verify the Pi can communicate with the BMI160 and read acceleration data.
- **Procedure:**
  - Initialize I2C bus at address 0x68 (default for BMI160).
  - Read the CHIP_ID register (should return 0xD1).
  - Read Accelerometer X, Y, Z registers.
  - **Pass Criteria:** Chip ID matches; Z-axis shows ~$9.8m/s^2$ (gravity) while stationary.

Test 2: Camera & IR System

- **Goal:** Verify camera stream works and IR LEDs illuminate.
- **Procedure:**
  - Turn on IR LED GPIO pin (High).
  - Capture 5 frames using libcamera or OpenCV.
  - Turn off IR LED GPIO pin (Low).
  - **Pass Criteria:** Images are saved successfully; images taken with LEDs ON are significantly brighter than LEDs OFF (in a dark room).

Test 3: Network Telemetry (MQTT)

- **Goal:** Verify connectivity to the cloud broker.
- **Procedure:**
  - Connect to MQTT broker (e.g., test.mosquitto.org or private IP).
  - Publish a JSON payload {"type": "test", "status": "ok"} to topic helmet/test.
  - **Pass Criteria:** Client receives PUBACK (publish acknowledgement) from broker.

Part B: Integrated System Tests

Test 4: The "Shake to Upload" Test (Crash Simulation)

- **Goal:** Test the full pipeline: Sensor $\rightarrow$ Logic $\rightarrow$ Storage $\rightarrow$ Cloud.
- **Procedure:**

    - Start the main helmet_monitor.py service.
    - Violently shake the sensor/helmet to simulate a >1G force.
    - Wait 30 seconds.
    - **Pass Criteria:**
        - System logs "CRASH DETECTED" in local logs.
        - A video file (e.g., crash_<timestamp>.mp4) appears in the events/ folder.
        - Dashboard receives an alert with "Severity: High".

Test 5: Fatigue Logic Simulation (Software Injection)

- **Goal:** Verify the AI logic triggers correctly without needing a sleepy driver.
- **Procedure:**
    - Feed a pre-recorded video file of a person closing their eyes into the CV pipeline instead of the live camera.
    - Monitor the PERCLOS variable in the debug log.
    - **Pass Criteria:**
        - PERCLOS value rises above 0.12 (12%).
        - "FATIGUE ALERT" is triggered.
        - Latency between eye closure in video and alert log is <200ms.

Test 6: Endurance Power Test

- **Goal:** Validate thermal and battery performance.
- **Procedure:**
    - Charge battery to 100%.
    - Run the system in full operational mode (Camera + AI + Wi-Fi) inside a room at 25°C.
    - Log CPU temperature and uptime every minute.
    - **Pass Criteria:**
        - System runs for >8 hours before shutdown.
        - CPU temperature does not exceed 80°C (thermal throttling limit).