Project: Firefighter Activity Recognition Training Pipeline

Objective: We need to build an end-to-end system to train an AI model that can recognize specific physical actions performed by a firefighter based **solely** on wearable sensor data.

To train this sensor model, we first need to build a **Data Annotation & Validation Tool** (the UI shown in the image). This tool will use video and pose estimation to help us create accurate labels for the sensor data.

Part 1: The Hardware & Data Inputs (Features)

The final AI model will only see data from two sources. We need to ensure our recording system captures these synchronously:

1. **Neck Accelerometer:** A 3-axis accelerometer placed just below the neck. (Data: Time-series X, Y, Z acceleration).
2. **Foot Pressure Insoles:** Sensor pads that collect pressure distribution across the foot. (Data: Time-series heatmaps/pressure array values).

Part 2: The Target Outputs (Classes)

The AI model needs to classify the current state of the user into one of these categories:

- Walking
- Running
- Crawling
- Sitting
- Turn Left
- Turn Right

Part 3: The Training Workflow (The Core Requirement)

This is the most crucial part for the developer to understand. We are not just training a model; we are building the pipeline to generate the training data.

We need to implement a 3-step "Ground Truth" generation process:

Step 1: Synchronized Data Collection We will record a session where a person performs actions. We must simultaneously record:

- Video of the person.
- Sensor data (Neck Accel + Foot Pressure) exactly synced to the video timecodes.

Step 2: Automatic Visual Pre-Labeling (The "Draft" Timeline) Once the data is uploaded to our tool:

- The backend should process the video using a visual pose estimation library (like **ml5.js** / PoseNet / MediaPipe).
- Based on the visual skeleton movement, the system should generate a "best guess" timeline of actions (e.g., the video AI sees legs moving quickly and labels a segment as "Running"). This is the **"Auto-Detected" track**.

Step 3: Manual Annotation & Correction (The UI Tool) We need to build the web interface shown in the image to allow a human to finalize the training data.

- **Visual Visualization:** The UI must play the video (with the stick-figure pose overlay) side-by-side with visualizations of the sensor data (foot heatmaps and accelerometer graphs), all perfectly synced in real-time.
- **The Timeline Editor (Bottom of image):** This is the heart of the tool. It needs multiple timeline tracks:
    - **Track A (Reference):** Displays the "Auto-Detected" labels generated in Step 2 by ml5.js.
    - **Track B (Ground Truth):** An editable track where a human annotator can override the auto-labels. For example, if ml5.js thought the person was "Walking" but they were actually "Turning Left," the human corrects it here.

The data from **Track B (Ground Truth)**, combined with the raw sensor data, is what will be used to train the final model.

Part 4: The Final AI Model Goal

Once we have used the tool above to generate clean, human-verified data segments:

- We will train a new machine learning model (likely an LSTM or CNN-based network suitable for time-series data).
- **Model Input during Training:** Raw Sensor Data + The Corrected Ground Truth Labels from the timeline tool.
- **Model Input during Deployment:** ONLY the Raw Sensor Data (no video).
- **Model Output:** Real-time prediction of the action (e.g., "Walking").