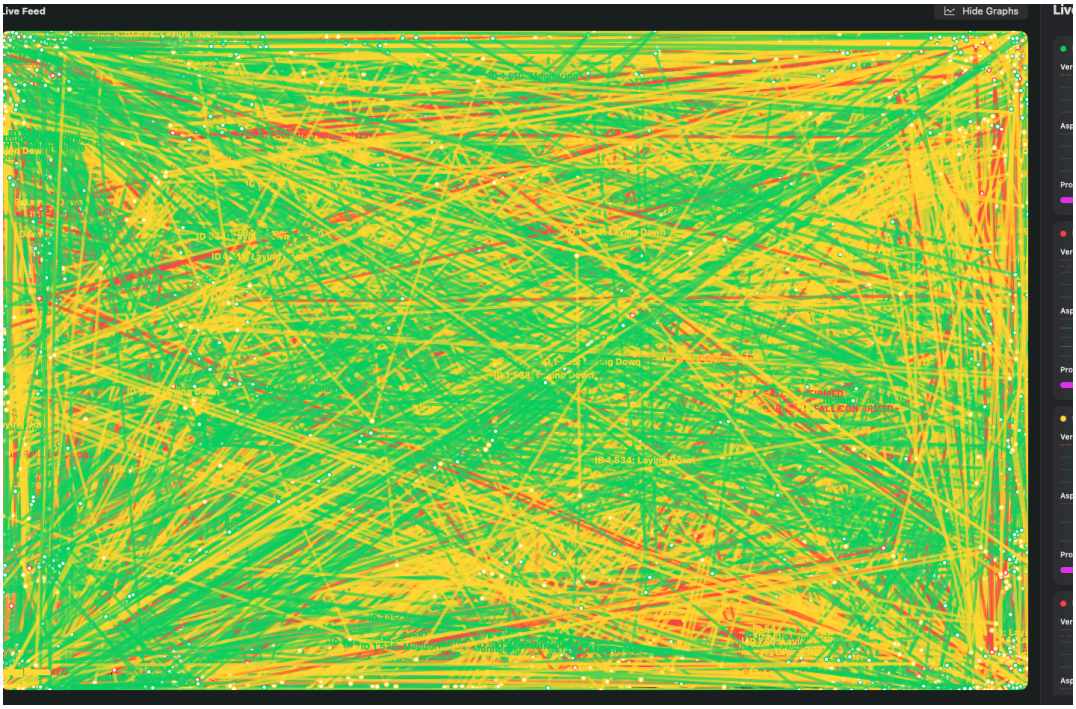
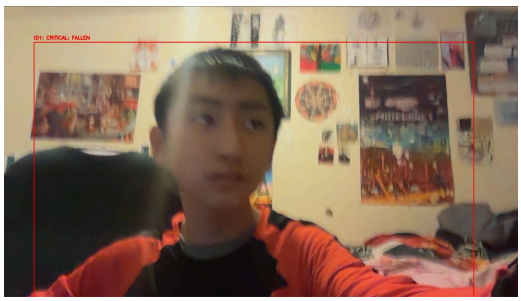
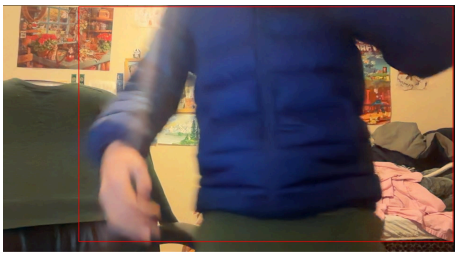
2026-01-15

- Found that Apple's built-in vision framework sacrifices accuracy for efficiency; decided to switch to the YOLO11n-pose model by converting it to a Create ML model.
- Encountered difficulties in changing the model to YOLO26n-pose
    - 
    - I switched to the newest version of yolo (26) that was released this month.
- Decided to switch to a Python implementation, as I am more familiar with Python, and it is easier to work with
    - Decided to slightly change the fall detection algorithm
        - Descent Expiry: If "High Descent" is triggered but the person doesn't enter a "Lying" posture within 1.5 seconds, the state resets to "Normal".
        - Recovery Detection: If the state is "Fallen" but the aspect ratio returns to a "Standing" value (Aspect < 0.8) for a sustained period, the alarm clears automatically.
        - Hysteresis Buffer: Added a small buffer to the aspect ratio to prevent the label from flickering when you are at the threshold.
        - Professional UI Colours: Updated the BGR values to be more vibrant for presentation.
    - Added real-time email functionality
- Encountered issues with the algorithm flagging people very close to the camera

- SOLUTION: Edge Filtering and Maximum Glitch Suppression





  - 1. Edge Margin Exclusion
    - Objects entering the frame from the sides or bottom often appear as "explosive" motion because the AI first detects only a head or shoulder, then suddenly the whole body.
    - The Fix: Ignore any HIGH DESCENT triggers if the person's bounding box is within the outer 5-10% of the screen edge.
  - 2. Max Velocity "Glitch" Cap
    - Real human falls have a physical limit to how fast they can accelerate downward. Moving a limb directly toward the camera lens can create a pixel-velocity that is physically impossible for a falling body.
    - The Fix: Add a MAX_GLITCH_VELOCITY constant (e.g., 0.8). If the calculated velocity exceeds this, it is treated as a sensor glitch or "teleportation" rather than a fall.
  - 3. Spatial Consistency Check (Y-Axis)
    - When you lean into a camera, your vertical center (y_pos) might shift, but you are usually still in the upper or middle part of the frame.
    - The Fix: Require that a FALLEN state can only be confirmed if the person's final resting Y-coordinate is in the bottom half of the screen (e.g., y_norm > 0.55).
- Thought of other features that are important and would be great for the final version
  - Privacy Mode. When active, the video stream shows only the person's skeleton (pose keypoints) on a black background, but no actual video of the person. Protects the dignity of seniors and adds a layer of reassurance
  - Add a chart showing latency and inference speed.

    - Add a physical “floor calibration”, where the user can click four points on the floor in the video. The AI then calculates the person’s height relative to the floor.
- Thought of and encountered several false positives
    - The "Yawn" / Arm Reach: Raising arms stretches the bounding box upward, then dropping them causes a massive drop in the "box center" velocity.
        - SOLUTION: Anatomical Centroid Tracking: Velocity is now calculated using the mid-point of the shoulders (kpts[5] and kpts[6]) rather than the box center. Shoulder height remains stable even if arms reach toward the ceiling.
    - Proximity Glitch: Moving very close to the lens causes the person to "jump" across many pixels, creating impossible velocity spikes.
        - SOLUTION: Max Glitch Filter: Implemented a MAX_GLITCH_VELOCITY constant (0.8). Any movement faster than this is ignored as a sensor "teleportation" error rather than a physical fall.
    - Leaning into the Camera: A person sitting on a bed and leaning toward the camera can trigger a "lying" aspect ratio while still safely on the bed.
        - SOLUTION: Floor Proximity Check: A fall is only confirmed if the final resting vertical coordinate (y_norm) is in the bottom half of the frame (e.g., y_norm > 0.55).
    - Edge Clipping: A person walking past the side of the camera might have their bounding box cut off, making them look "flat" (lying).
        - Edge Margin Suppression: Added an EDGE_MARGIN (0.05). If the person's center is within 5% of the screen edge, detection triggers are suppressed to prevent "partial person" errors.
    - Rapid Crouching: Dropping quickly to pick something up creates high velocity, followed by a low height.
        - SOLUTION: Stillness Window: The system now requires the person to remain in a lying position for a specific duration (STILLNESS_WINDOW) before the alert is sent, allowing for quick "down and up" motions.

2026-01-16

- Started by creating the project board online on Canva.
- Fixed the issue of having arms extended, resulting in lying down and also lying down causing falls.

    - Instead of tracking the jittery bounding box, the algorithm now tracks the Mid-Shoulder/Mid-Hip point. This is much more stable when people are moving around in bed.
- Added a feature where users can input the floor proximity threshold, so that the algorithm is accurate for different camera angles and room types.
- Decided to lower the normalization value for far people

2026-01-17

- Experienced problems with the algorithm thinking people are standing towards the camera, as it looks at the bounding box and torso angle.
- Problems:
    - **Occlusion Jitter** (e.g., chair blocking a hip)
        - Persistence of State: Created a last_valid memory system that freezes the torso angle if keypoints disappear.
        - Prevents "false-positives" where the system thinks someone stood up because it lost sight of a joint.
    - **Perspective Distortion** (Falling toward the camera)
        - **Hybrid Posture Check**: Added Bounding Box Aspect Ratio (Aspect > 1.2) as a backup to the Torso Angle check.
        - Correctly identifies "forward falls" where the torso still looks vertical but the person is on the floor.
    - **Velocity Jitter** (Shrinking/Growing person)
        - **Ruler Lock Logic**: The normalization "ruler" (torso length) only updates while standing; it locks during a fall.
        - Ensures velocity remains accurate even as the person's distance from the camera changes during impact.

2026-01-18

- Researched sound classification models. Decided on the following sound classification labels for the final project:
    - Thud / Impact: The primary trigger for a potential fall.
    - Distress (Screams/Shouts): High-decibel vocalization that confirms a crisis.
    - Normal (TV, Conversation, Footsteps): To prevent false positives from everyday life.
    - Silence / Background: The baseline.
- Decided on additionally running audio through Speech-to-Text (STT) to detect "Are you okay?" responses, as it is the professional approach. Here is why:

    - Why STT is better: Audio classification is great at recognizing a "sound" (like a clap), but it struggles with the nuances of human speech. STT (using Apple's SFSpeechRecognizer) allows the system to understand the intent of the answer.
- **The Logic:**
    - **Detection:** Vision + "Thud" sound = Fall Detected.
    - **Interaction:** The system triggers a Text-to-Speech (TTS) voice: *"A fall was detected. Are you okay?"*
    - **Confirmation (STT):**
        - If the user says **"Yes"** → False Alarm, the system resets.
        - If the user says **"No"** or "Help," → Immediate Alert.
        - If **Silence** (10 seconds) → **Critical Emergency** (User may be unconscious). This is the most dangerous scenario.
- Found several audio datasets for model training. Will create own dataset too
    - Audio Dataset of Scream and Non-Scream
    - https://www.kaggle.com/datasets/sanzidaakterarusha/primary-data-scream?resource=download
    - https://www.kaggle.com/datasets/whats2000/human-screaming-detection-dataset
    - All together, the model has 6934 data points for 2 classes. The validation accuracy is 87%
- Experienced dependency and security issues with Apple's speech recognition on Python; decided to switch to "Whisper," optimized for CoreML
    - The speech framework requires strict app-like permissions that crash Python. Decided to treat speech-to-text just like a YOLO model, loading the model and running it directly on the NPU

Finalizing Project - Poster Board, Visualization, Datasets, Testing & Getting Results

2026-01-19

- Found more fall detection dataset videos to test
    - https://www.iro.umontreal.ca/~labimage/Dataset/#/
    - https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/75QPKK
- Tried connecting to home cameras from school; realized it didn't work
    - Researched and found out connection requires using a VPN to "simulate" the MacBook and Windows server to be in the same wifi network.

    - Decided on using “Tailscale” to bridge the Mac and Windows connections together.

2026-01-19

- Found another fall dataset
    - https://figshare.com/articles/dataset/Sensor-Based_Fall_Detection_Dataset_with_2017_Activities_from_29_Subjects/28596332?file=52990358

2026-01-21

- Experienced audio jitters and lag through connections. Currently, the code tries to play audio as soon as a packet arrives. If the internet lags for even 0.01 seconds, the speakers "starve" of data, causing that robotic, crackling jitter.
    - SOLUTIONS:
        - **Jitter Buffer:** The code will store a small amount of audio (0.2 seconds) before starting playback. This acts as a "safety net."
        - **Chunk Size:** The logic will have an increased chunk size of `2048`.
        - **Mono vs Stereo:** The code will force 1-channel (Mono) to cut the data sent over the network in half.
- Started incorporating Whisper STT and the scream classifier model into the dashboard script
    - The code now uses `coremltools` to load the `scream_classifier.mlmodel`. It runs parallel to Whisper. If it hears a scream, it triggers an immediate alert (bypassing the countdown).
    - **The "Grace Period" Logic:**
        - **Fall Detected:** System plays audio: *"Fall detected. The system will notify emergency contacts in 10 seconds unless the system hears, 'I am okay, fine, or yes'. Say “Help, No” to trigger notifications.*
        - **Whisper Listens:** If it hears "Okay/Fine/Yes," it resets. If it hears "Help/No," it triggers immediately.
        - **Timeout:** If silence for 8-10s, it triggers.

2026-01-22

- Worked on the poster and researched falling-related statistics
- Successfully implemented WAN communication for remote connection with different IP addresses
- Fixed Raspberry Pi Zero 2W camera lag and blurriness by switching to Putty’s SSH connection instead of VNC viewer, which dramatically reduced power consumption and chip load.

2026-01-22

- Experienced several errors happening simultaneously after trying to implement sound models.
    - Threading
        - The Mac is trying to run YOLO (CoreML) and Whisper (PyTorch MPS) on the Neural Engine at the same time, and they are fighting over memory, causing the contracting dimensions to abort.

2026-01-27

- Had several audio overlap issues.
    - The "infinite loop" is caused by the system's microphone picking up its own speakers. When it plays "System Cancelled," the word **"cancel"** is heard by Whisper, which triggers the cancellation logic again, and when it says **"notify,"** it can accidentally trigger the alert sequence if "notify" is near a keyword.
- Finished making all the audio files with QuillBot's TTS functionality.

2026-01-28

- Had audio lag and jitters. Solutions:
    - **Implement a Deque Buffer:** Instead of playing audio immediately, put it into a thread-safe queue. Let a separate dedicated audio thread pull from that queue at a steady rate.
    - **Pre-buffering:** Wait until you have 3–5 chunks in the queue before you start playing. This acts as a "safety cushion" against network spikes.
    - **Switch to a non-blocking callback:** PyAudio can be configured in "callback mode," which is much more efficient for real-time streaming than the current `.write()` method.

2026-01-29

- Fixed several errors
- Thought of using the "py2app" Python library to turn it into an app
    - `pip install py2app`
    - Create a `setup.py` file.
    - Run `python setup.py py2app`.

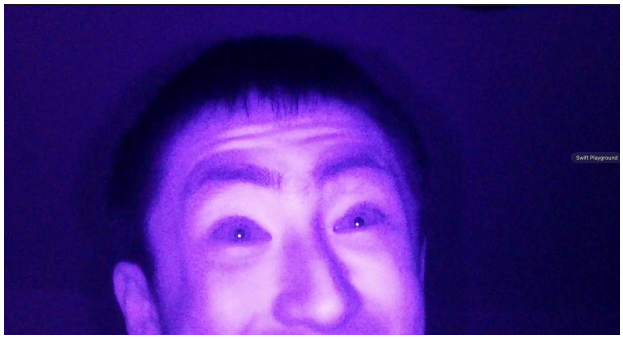
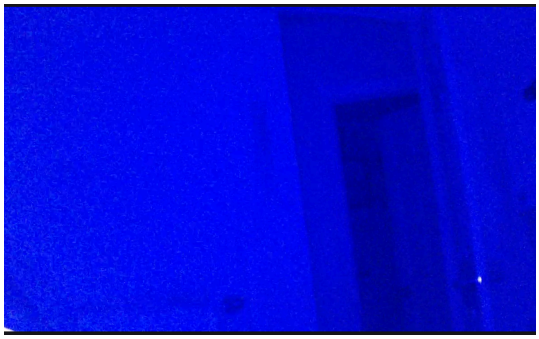
2026-02-01

- Switched to Mac's built-in speech recognition---it is more power efficient and accurate.

2026-02-06

- Started learning how to solder
- Got the audio and speakers

    - For the amplifier:
        - Open the boot configuration file: `sudo nano /boot/config.txt`
        - Scroll down and look for `dtparam=audio=on`. Change it to: `#dtparam=audio=on` (This disables the default driver).
        - Add the following line at the bottom of the file: `dtoverlay=max98357a`
        - Press `Ctrl+O`, `Enter`, and `Ctrl+X` to save and exit.

2026-02-12

- Collected fall, lie, stand, and walk statistic data by building a script
    - Researched and studied how to use matplotlib to create complex graphs
- Worked on the poster

2026-02-13

- Worked on poster---finalized the first 2 poster boards
- Found “Rasberbator” website to divide posters into printable A4-A3 chunks
- Finalized the thresholds for fall detection

2026-02-14

- Created a Python script with matplotlib to visualize the data (velocity, y-pose, shoulder to knee angle)
    - Changed the angle calculation to shoulder to knee (from shoulder to hip before) as it is more accurate in bending down events (it doesn’t flag)

2026-02-15

- Finished poster board
    - Decided on only having 2 poster boards
- Created a 3d visualization script
- Created a latency/fps visualization script & tested it up to 35 instances on 640p & 320p
- Transferred many data files from the MacBook to Windows

2026-02-20

- Finalized summary & submitted registration form
- Simplified poster board to increase font size & reduce the amount of text
- Decided not to finish the dashboard as it was too difficult to do.

2026-02-22

- Printed the poster using Rasterbator

2026-02-23

- Assembled the poster
- Researched current alternatives for watches

Watch Type	**Initial Cost**	**Monthly Subscription**	**Hidden Cost**
Apple Watch (SE/Series 10)	$350 - $550	$10 - $15 (Cellular plan)	Requires an iPhone ($400+)
Google Pixel Watch 3	$450+	$10	Requires an Android Phone
Medical Alert Watches	$50 - $150	~$30	High lifetime cost
Cheapest “Generic” Watch	$40 - $80 (Amazon)	$0	Very high false alarm rate (unreliable)

2026-02-24

- Created bibliography page
- Created a short ~60-second elevator pitch (will not memorize)
- Prepared simplified visualizations of the Spectrograms and 3D Clusters (Figures 5 and 10) to explain the difference between a 'thud' and a 'scream' to judges.

2026-02-27

- Printed logbook & the entire project is basically complete! Just need to review and practice the presentation.

Sources

Apple Inc. (n.d.). MLSoundClassifier. Apple Developer Documentation. Retrieved from https://developer.apple.com/documentation/createml/mlsoundclassifier

Arusha, S. A. (2023). Primary data scream [Data set]. Kaggle. https://www.kaggle.com/datasets/sanzidaakterarusha/primary-data-scream

Charfi, I., Miteran, J., Dubois, J., Atri, M., & Tourki, R. (2013). Optimised spatio-temporal descriptors for real-time fall detection: Comparison of SVM and Adaboost-based classification. Journal of Electronic Imaging, 22(4), 17. https://doi.org/10.1117/1.JEI.22.4.041006

Jocher, G., & Qiu, J. (2026). Ultralytics YOLO26 (Version 26.0.0) [Computer software]. Ultralytics. https://github.com/ultralytics/ultralytics

Kwolek, B., & Kepski, M. (2014). Human fall detection on an embedded platform using depth maps and wireless accelerometers. Computer Methods and Programs in Biomedicine, 117(3), 489–501. https://doi.org/10.1016/j.cmpb.2014.09.005

Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA). (n.d.). Multiple cameras fall on the dataset [Data set]. University of Montreal. http://www.iro.umontreal.ca/~labimage/Dataset/

Our World in Data. (n.d.). Death rate from falls. Retrieved from https://ourworldindata.org/grapher/death-rate-from-falls

Seeed Studio. (n.d.). Getting Started with Seeed Studio XIAO ESP32C3. Seeed Studio Wiki. Retrieved from https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/

Seth Adams. (2018, October 24). DSP Background - Deep Learning for Audio Classification p.1 [Video]. YouTube. https://www.youtube.com/watch?v=Z7YM-HAz-IY

Siam, A. A. (2023). Audio dataset of scream and non-scream [Data set]. Kaggle. https://www.kaggle.com/datasets/aananehsansiam/audio-dataset-of-scream-and-non-scream

Ultralytics. (n.d.). Pose Estimation - Ultralytics YOLO Docs. Retrieved from https://docs.ultralytics.com/tasks/pose/

Whats2000. (2021). Human screaming detection dataset [Data set]. Kaggle. https://www.kaggle.com/datasets/whats2000/human-screaming-detection-dataset