# Waste Volume Estimator - Change Report

2025-10-07

## Dashboard Sync Reliability

### Problem:

* Dashboard relied on a fixed 3-second reload timer, so users still saw stale data until the next poll.

### Changes Applied:

1. Implemented a lightweight dashboard event broker on the Flask server to publish notifications whenever a session is saved.
2. Added a server-sent events endpoint (/dashboard/events) and wired the template to listen for real-time updates.
3. Removed the polling script in dashboard.html so the view reloads immediately after Firestore persistence succeeds.

## Firestore Persistence Failures

### Problem:

* Flask crashed with AttributeError because Firestore add returned a timestamp tuple, not a document reference, breaking the event notification code.

### Changes Applied:

1. Replaced .add with an explicit document().set(...) call so we always retain the generated document ID.
2. Reused the new doc\_ref.id in event payloads to keep the dashboard updates working.

## Webcam Start/Stop Stability

### Problem:

* Initial Start call failed silently when rear camera access was blocked or unavailable.
* Repeated Start/Stop cycles left the previous stream attached, causing the browser to think the webcam was still busy.

### Changes Applied:

1. Introduced progressive constraint fallbacks (environment, user-facing, any) with clearer error messaging so a workable camera is selected.
2. Disabled/enabled the Start/Stop buttons to mirror capture state and avoid double-start clicks.
3. Added a cleanupStream helper to stop tracks, pause the video element, and clear srcObject, ensuring fresh access on every restart.

## Interface Modernisation

### Problem:

* Existing capture and dashboard pages used basic inline styles and were not responsive on smaller screens.

### Changes Applied:

1. Rebuilt index.html and dashboard.html with Tailwind CSS, giving both pages a cohesive dark theme and responsive layouts.
2. Organised controls and metrics into card-based grids, improved typography, and added visual status indicators.
3. Preserved all element IDs and data bindings so the JavaScript capture flow and dashboard listeners continue to operate.

## Food Classification Integration

### Problem:

* Application lacked food-level insights; dashboard could not differentiate sessions by detected meal type.

### Changes Applied:

1. Added a TensorFlow-backed classifier wrapper that loads keras\_model.h5/labels.txt lazily and returns the most likely food along with confidence.
2. Extended the capture pipeline to classify each frame before volume estimation, surfacing the prediction in the live HUD and JSON responses.
3. Persisted detected food labels and confidence alongside each session, updated dashboard metrics to count classes, and exposed the breakdown in the UI.