Sprint Three Report

PORTFOLIO TASK 7

Unit code: EAT40006
Unit Name: Engineering Technology Project B
Submission date: 12 / 10 / 2025

ACKNOWLEDGMENT OF COUNTRY

We Dale Bent, Mohammed Barsat Zulkarnine, Dang Khoa Le, and Sadman Tariq, acknowledge the Traditional Custodians of the lands on which we lived and worked while completing this project. We pay our respects to their Elders past and present and extend that respect to all Aboriginal and Torres Strait Islander peoples.

CONTRIBUTION SUMMARY (THIS SPRINT)

All team members should complete the following table together. You can provide a yes/no response or a brief comment where appropriate.

	Contribution	Initiatives	Communication					Respect
Team Member	Finished tasks in time or on time with an acceptable quality	Self-assigned tasks and proactively found solutions to problems	Attended all team meetings	Attended all supervisor meetings	Attended all client meetings	Replied every time within an acceptable timeframe	Kept the team updated about the status	The student respected others and listened to different opinions
Mohammed Barsat Zulkarnine	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dang Khoa Le	Contributed to RLHF, front-to-back APIs, and efficiency scoring	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dale Bent	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sadman Tariq	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

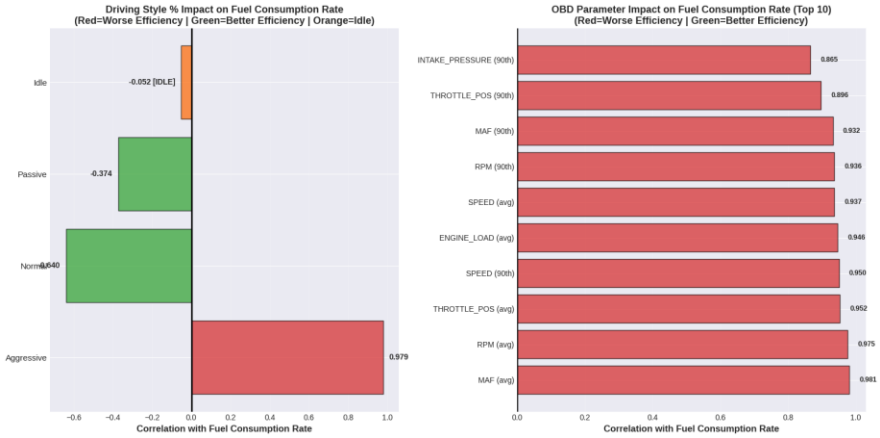
CONTRIBUTION DETAILS (THIS SPRINT)

In this section, every member of the team should write a summary of what they have contributed during this sprint. All contributions must be supported by evidence, such as the number of GitHub commits, a screenshot of developed screens, a summary/outcome of research, etc.

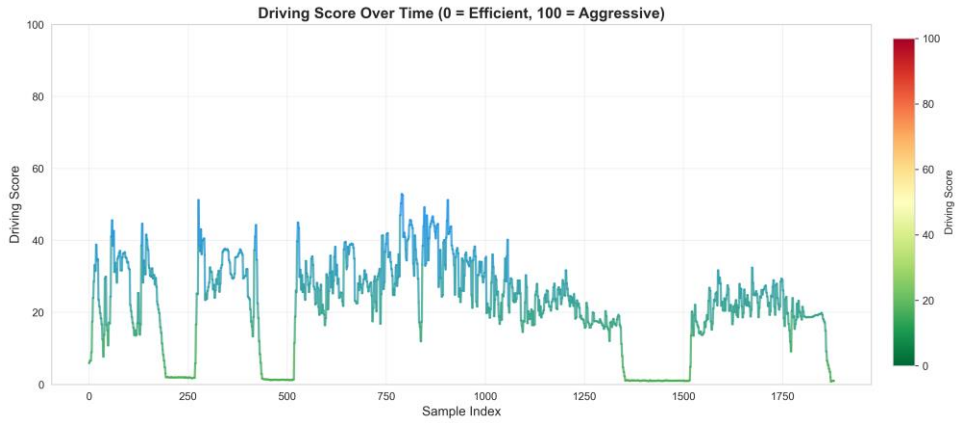
Dale:

During this sprint, I continued to collect logs and label driving data. In addition to this, I began some data analysis on this data, comparing the exact same drives, with varied driving styles to determine correlation between OBD parameters and fuel use, across these drives. Below is the collab for this and the results:

https://colab.research.google.com/drive/13Ahp5UIkSB79RZLEzGMliAwR_HqT6aQB#scrollTo=Zun36nMlPACX



After this, I began looking for a way to determine the driving style, using the parameters given, rather than relying on self labelling. We now use essentially a scoring system, based on the most highly correlated parameters and then give a score using the weight of the correlation, see:



Which, based on the average of this, determines a driving score for the whole drive. This in combination with ML based upon labelled driving gives us an explainable and robust labeling of the drive in terms of driving style.

Sadman:
During this sprint, I shifted the focus of the datalogger development away from battery life optimisation based on the facilitator's suggestion, instead focusing on using a cigarette lighter adapter as the main power supply, which simplified the power management requirements.

In the following weeks, I worked further on hardware integration for the datalogger and explored methods for stream translation and redirection using socat, which supported more flexible data routing between components. Although I was unwell with a cold and fever during part of this period, I kept the team informed of my situation. Toward the end of the sprint, I continued refining the datalogger and integrated it with the newly developed backend endpoints to ensure proper communication between the device and the server.

A large portion of work was also dedicated to research and reading technical documentation. Particular areas of research were asynchronous programming in python using the asyncio module (https://docs.python.org/3/library/asyncio.html), the socat command line utility for stream redirection and USB serial access (https://www.redhat.com/en/blog/getting-started-socat), and the NetworkManager suite for wifi and network management on the Raspberry Pi (https://networkmanager.dev/).

Barsat:
In this sprint, I started by finishing the integration of the reinforcement (RLHF) feature on the developer side of the website (Figure 1). This feature allows developers to interact with model outputs and apply reinforcement updates directly through the web interface.

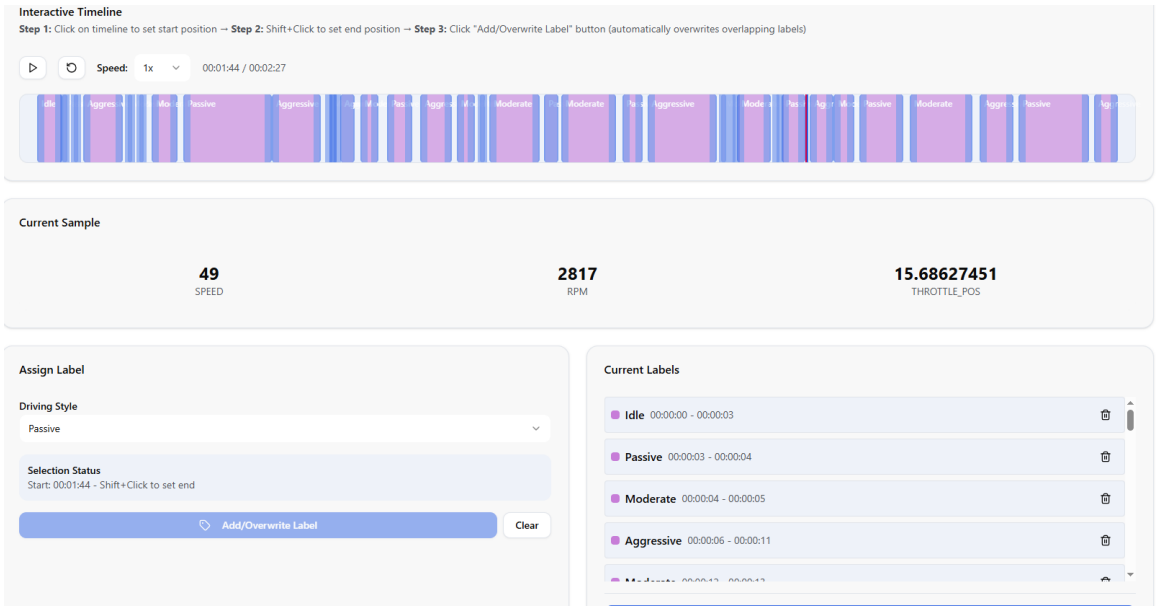


Figure 1. Driver behavior relabellign for RLHF integration. Where we segmented driver states into smaller chunks (from model predictions) and allowing driver/developer to relabel.

Next, I worked on building user UI. I began designing and implementing the first version (Figure 2), but midway through the sprint, the plan was updated based on supervisor feedback, so I scrapped that version and redesigned it from scratch.

Figure 2. Recent trip UI with metrics breakdown on user side.

The new user UI Figure 3 now displays key information and integrates with the backend to show available data and model outputs. At this stage, only partial data is being displayed since not all model outputs are finalized yet. In the final version, the UI will dynamically adapt to show only the verified data and relevant status features based on what we provide from the backend.

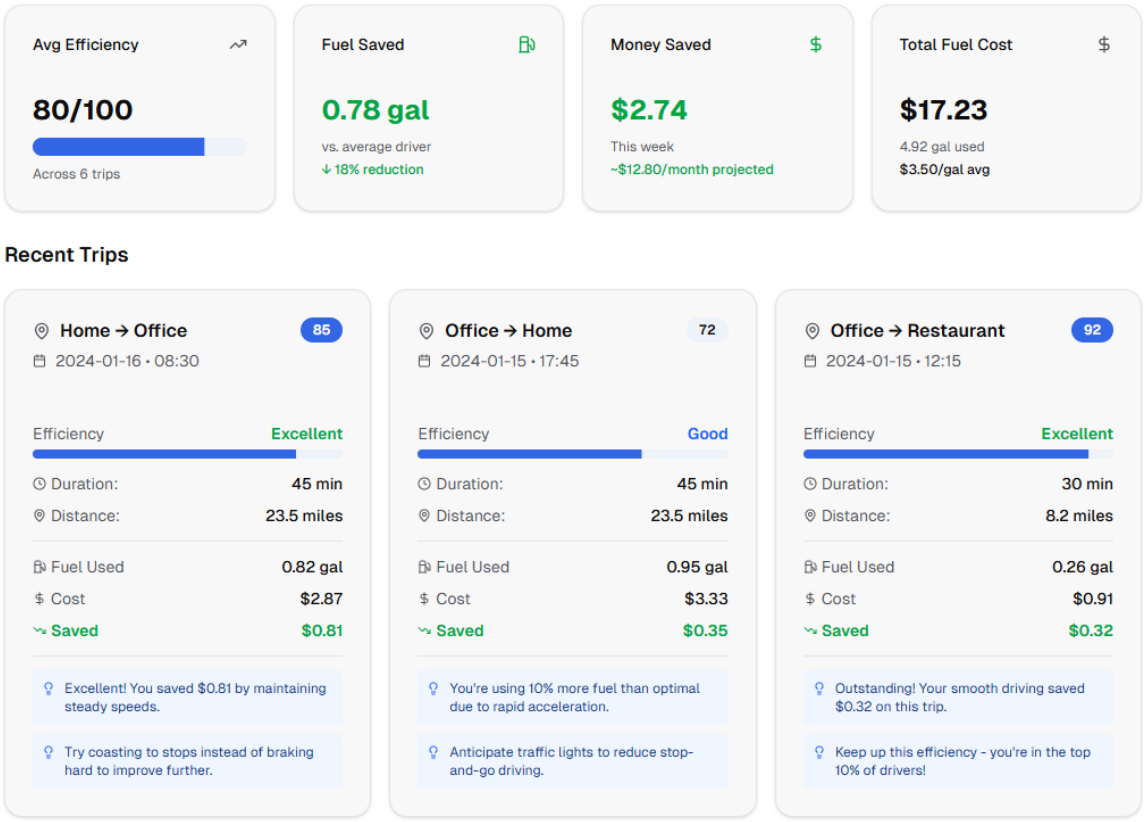


Figure 3. Recent trip UI with metrics on developer side with better breakdown.

Khoa:
In this sprint, I focus on RLHF implementation and optimize the RLHF workflow (model versioning and finetuning strategies), frontend-backend integration (via APIs and version control), data exploration and analysis, machine learning algorithms and architecture for driver behavior and fuel efficiency.

Here we have RLHF on the UI with full control, model redirection, versioning, and usage.

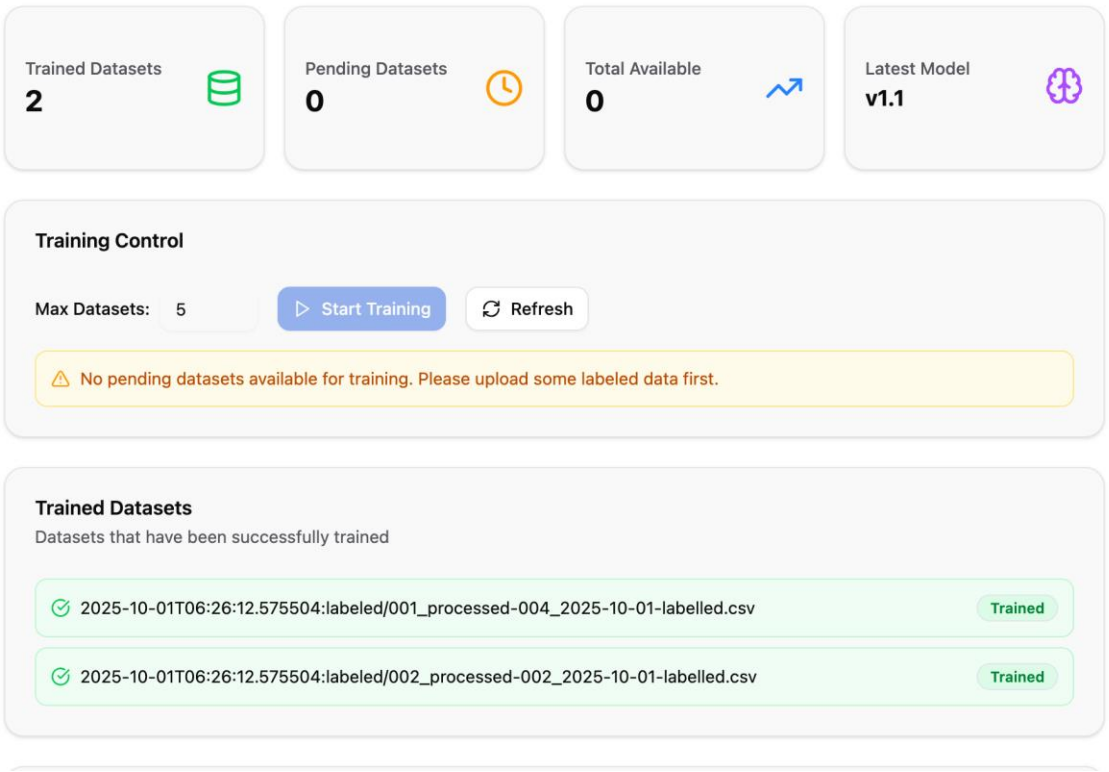


Figure 4. RLHF model control and training phase startup. All relabeled data that haven't been trained will be set to pending, and we can finetune the model based on these pending data (max 10 allowed at once). The new model version is incremented (started from v1.0). At default, we will use the latest model; however, developers can make changes to that. At the end of the page will be the hyperlink redirect developers to where the model is located in Hugging Face repo hub.

All data is listed with status and vitals, including the raw, processed or labeled data.

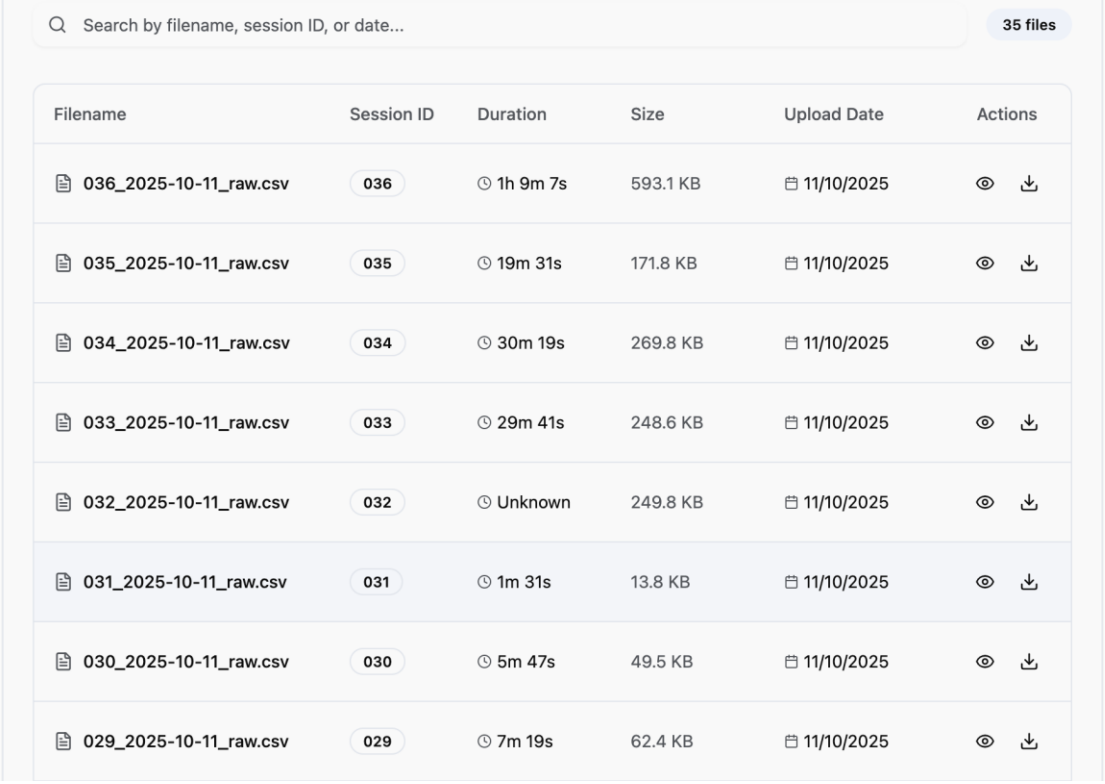


Figure 5. Data listing (including duration, file size, uploaded data), and actions (view and download).

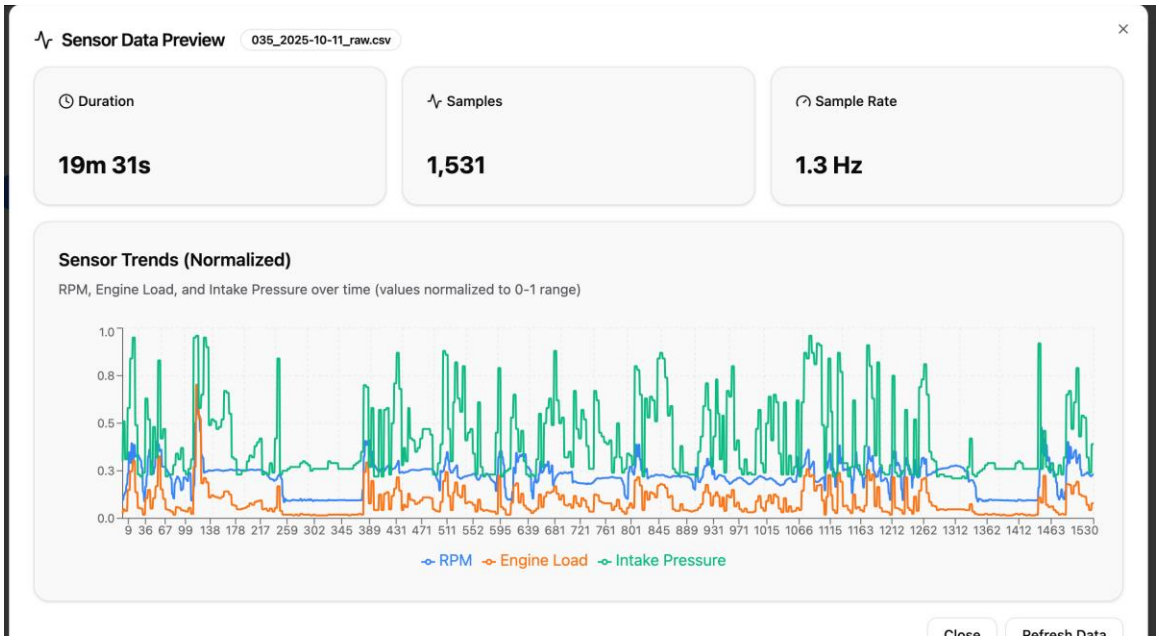


Figure 6. Data inspection.

RLHF and backend pipelines and full services can be accessed from https://huggingface.co/spaces/BinKhoaLe1812/OBD_Logger

For fuel efficiency, our ideologies include spike (constant acceleration and deceleration) variants of frequency / duration / magnitude (with hysteresis), plus idling penalties. Here we visualize an example of a drive (scored 65.8 in efficiency scoring), where least to most efficient chunks was shaded in color grading (green to red).

Figure 7. Color-graded fuel efficiency shadings.

1. SPRINT PLAN

For this sprint, we didn't nominate a formal Scrum Master as each of us was working in our own areas of expertise. This setup made sense for where the project was at as everyone knew what needed to be done and could work more efficiently without waiting on others. Khoa focused on the model training and RLHF (Reinforcement Learning from Human Feedback) side, handling backend integration and versioning of models. Dale continued collecting OBD driving logs and analyzing them to build reliable data for our rule-based scoring system. Sadman worked on improving the Raspberry Pi setup, ensuring it could automatically upload drive data to Firebase once connected to Wi-Fi. Meanwhile, Barsat concentrated on the front-end development, building both the developer dashboard and user interface that link everything together.

The main goal for this sprint was to get the system fully integrated, from the Raspberry Pi and OBD data collection, through the backend and data pipelines, all the way to the frontend dashboards. We also aimed to replace our earlier machine-learning-only approach with a more explainable, rule-based efficiency scoring method. This shift allows us to clearly show users how their driving behavior (like idling or acceleration patterns) affects their score. Alongside that, we started designing the user-facing dashboard, which will visualize trip summaries, driving scores, and improvement tips.

By the end of the sprint, most of the developer-side system was working and connected to the backend, while the user dashboard was still being finalized. The two main active versions of the web app can be accessed here:

- Developer dashboard: https://skyledge-ten.vercel.app/dev
- User dashboard (early version): https://skyledge-ten.vercel.app/developer

Overall, this sprint focused on connecting all parts of the system and building a proper foundation for testing and validation. While some backend and data integration work still needs refining, we now have a much clearer picture of how the final deliverable will look and function, and the system is finally coming together as one complete pipelin

2. FINAL DOCUMENTATION DELIVERABLES

The final version of our system isn't fully ready yet and will need a bit more time to complete. Right now, we've decided to focus on building and testing a rule-based scoring system to measure the overall driving score for each trip. This approach is easier to explain and verify since it's based on clear factors such as acceleration, braking, and idling time.

Once we've confirmed that the rule-based system works as expected, we'll bring in our machine learning model as a secondary option to see if it can produce better or more consistent results. This way, we can compare both methods and decide which one performs best before the final release.

Apart from that, the content and layout for the final user deliverable have been decided. The figure below shows a mock-up of the planned dashboard, which currently uses dummy data to demonstrate what users will see in the final product.

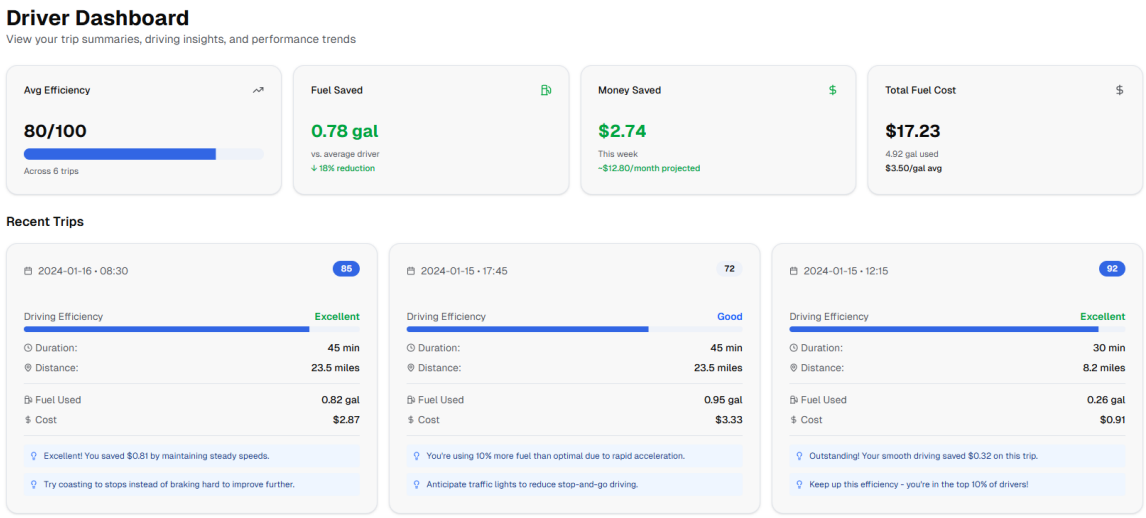


Figure 8. Driver dashboard

Each time a user completes a drive, the system will automatically calculate their driving score and show useful information such as:

- Trip duration
- Distance travelled
- Estimated fuel used and fuel efficiency

Based on their driving score, the user will also receive tips and suggestions on how to improve their driving habits and increase their score over time. These tips are linked to key behaviors, for example: reducing long idle times, avoiding sudden accelerations, or maintaining smoother throttle control.

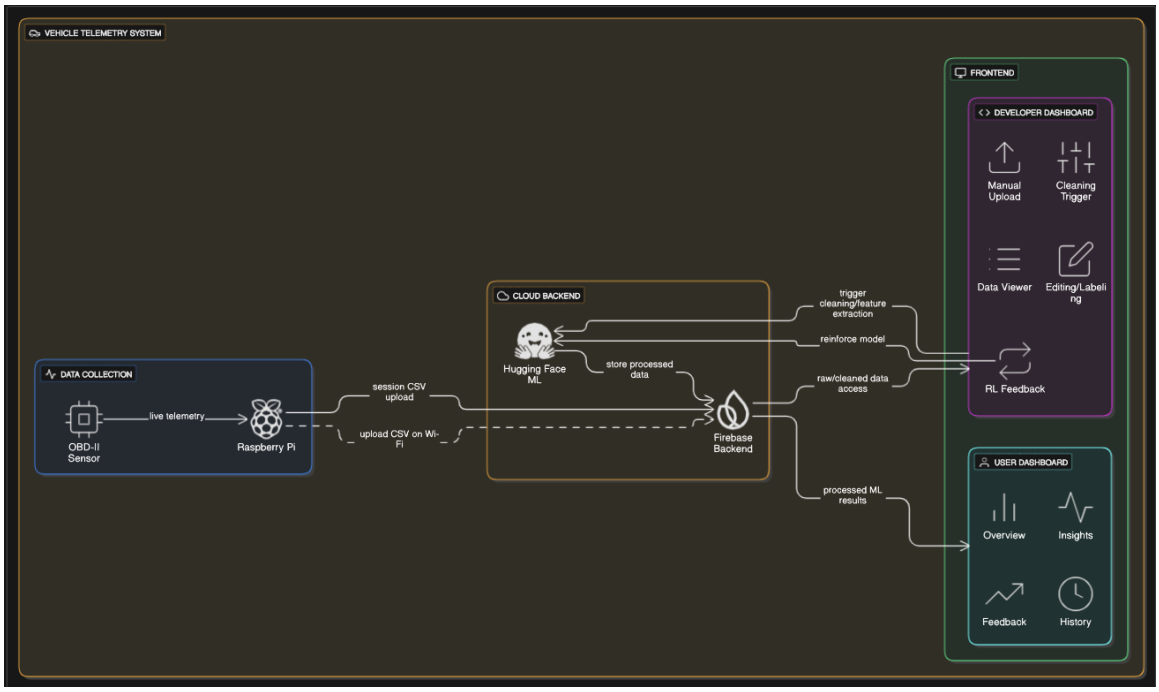


Figure 9. System summary – A brief introduction to multi-systems integration

This diagram the summary breaking down our system by 3 main components:

- Data collection (Vehicle, OBD, Raspberry Pi)
- Frontend (NextJS)
- Backend (FastAPI)
- ML models (will be fetched and served by the backend).

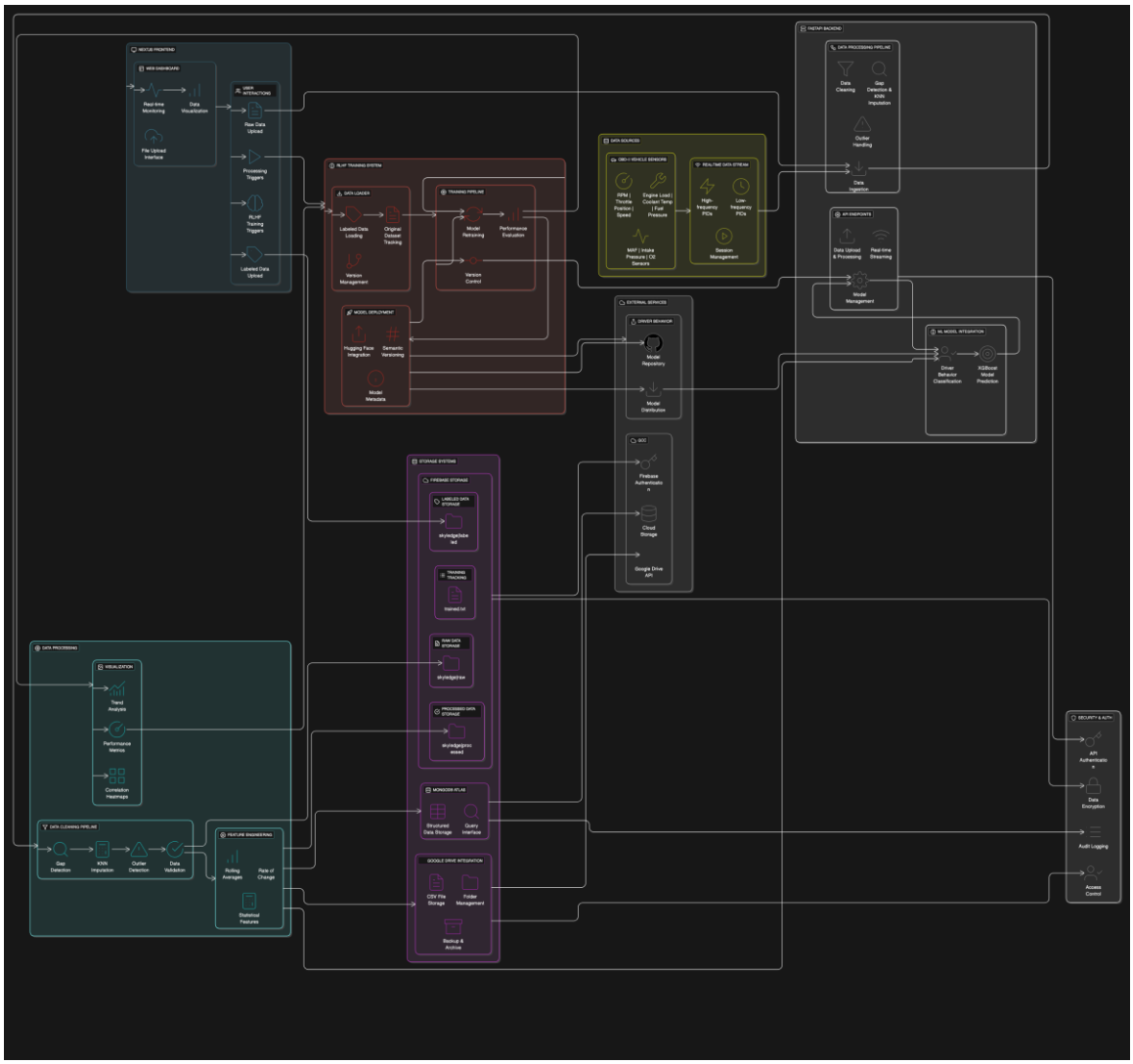


Figure 10. A detailed system breakdown

Here, we have the system being explained explicitly and comprehensively, where each sub-system and its services are demonstrated.

SYSTEM OVERVIEW

1) Data Collection (OBD-II + Raspberry Pi)

- Reads high/low-frequency PIDs (RPM, SPEED, THROTTLE_POS, ENGINE_LOAD, MAF, O2, pressures).
- Sessioned logging with timestamping; parallelised to minimise dropouts; LED for live status; power optimised for field runs.
- Streams to backend (or batches as CSV) with lightweight edge validation.

2) Frontend (Next.js Dashboard)

- Uploads: raw → skyledge/raw/, label-first timeline UI → skyledge/labeled/.
- Clean→Predict path: sends to cleaner + model → skyledge/processed/.
- Version governance: pick/pin model, trigger RLHF finetune, view per-version metrics.
- Visuals: segment colour-grading (efficiency 0–100), shaded Idle, behaviour mix, tips.

3) Backend (FastAPI Core)

- Data logging can usually be missed or corrupted; an intelligent cleaning and backfilling pipeline is necessitated.
- Ingestion → gap detection → KNN+LR joint imputation → corruption fixes while preserving valid extremes → feature engineering → validation.
- Model serving (versioned); artefact parity; MongoDB for run/session metadata; Firebase Storage for files; optional Google Drive archive.
- RLHF endpoints to train/promote/rollback with preferences and dataset tracking.

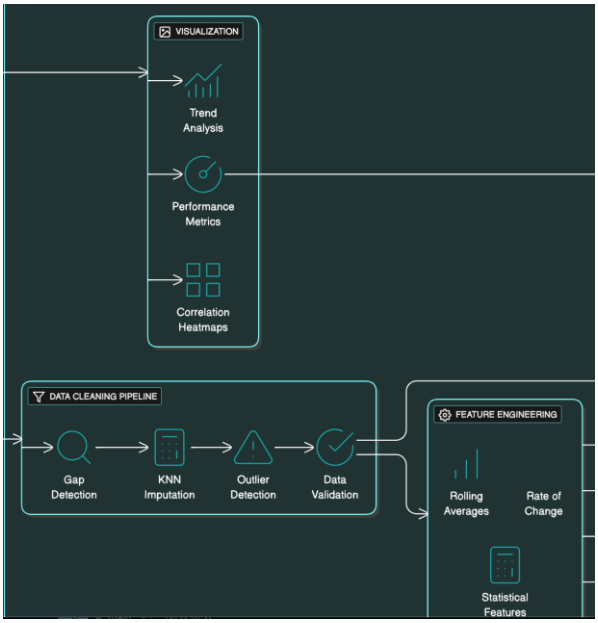


Figure 11. Cleaning pipelines

4) Model (Behaviour & Efficiency + RLHF)

- **Behaviour:** Rule-based Idle (cut-offs + smoothing), BGMM Idle confirmation & driving stratification, supervised XGBoost.
- **Efficiency (0–100):** Algorithmic bounded score (frequency/duration>magnitude + idle) + monotone GBM with isotonic calibration; length-invariant chunking (60–120s).
- **RLHF:** Compare-and-train on unused labeled sets; warm-start; capped penalties; semantic versioning (e.g., 1.1→2.0) with full export kit (thresholds, scalers, GBM, calibrator, schema, seeds).

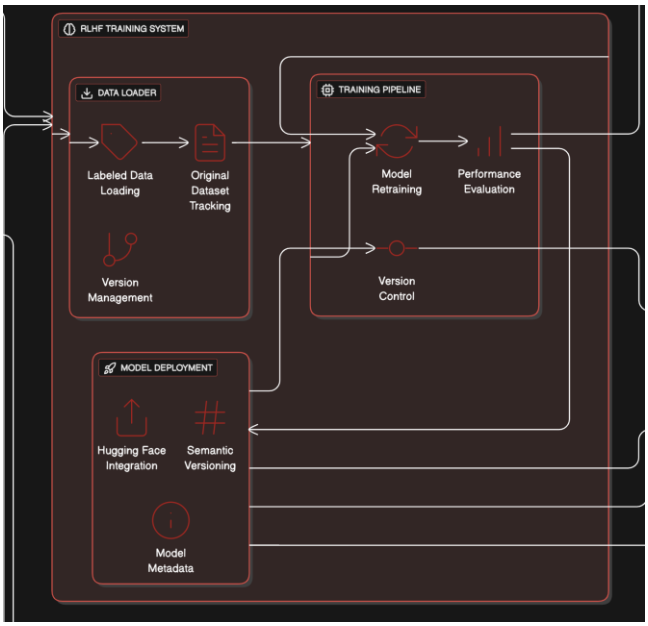


Figure 12. RLHF service components

5) Data Storage

Our processed data will be stored in Firebase for UI and backend control, while we also store them to Google Drive for quick developer access (for EDA, data analysis, and model training) on Google Colab.

Drive access at:

https://drive.google.com/drive/folders/1r-wefqKbK9k9BeYDW1hXRbx4B-0Fvj5P?usp=drive_link

Primary Links

- App: https://skyledge-ten.vercel.app
- Cleaner/Space: https://huggingface.co/spaces/BinKhoaLe1812/OBD_Logger
- Model Hub: https://huggingface.co/BinKhoaLe1812/Driver_Behavior_OBD

API Manuals

Below is a practical, concise guide to call the APIs grouped by purpose. All paths are relative to your deployed domain (e.g., https://skyledge-ten.vercel.app). Unless stated, responses are JSON with 200 on success; standard errors: 400/404/500.

A) FILE MANAGEMENT

Our web backend BASE_URL is at https://binkhoale1812-obd-logger.hf.space/

1. **Download File** — GET /api/download-file
   Query: fileName (required, full Firebase path)

```
curl -L "<BASE_URL>/api/download-file?fileName=file.csv" -o 001.csv
```

2. **Upload to Firebase** — POST /api/upload-firebase (multipart)
   Form fields: file (binary), fileName (target path)

```
curl -X POST "<BASE_URL>/api/upload-firebase" \
 -F file=@file.csv \
 -F fileName=skyledge/raw/id_datetime_raw.csv
```

B) FILE LISTING

4. **Raw Files** — GET /api/raw-files
5. **Processed Files (detailed)** — GET /api/processed-files
6. **Labeled Files** — GET /api/labeled-files
7. **Labeled Files (detailed)** — GET /api/labeled-files-detailed
8. **Predicted Files (with UI render)** — GET /api/predicted-files
9. **Raw Logs (detailed)** — GET /api/raw-logs

C) TRIP MANAGEMENT

10. **Eligible Trips:** Trips that can be processed — GET /api/eligible-trips
11. **Label Trip: R**eturn trip segments for UI rendering — POST /api/label-trip
    Body:

```
{
 "rawPath": "skyledge/raw/001_2024-01-15_raw.csv",
 "timestampColumn": "timestamp",
 "sourceType": "processed",
 "ranges": [
  {"startTs":"2024-01-15T10:00:00.000Z","endTs":"2024-01-15T10:05:00.000Z","label":"idle"}
 ]
}
```

Valid labels: "idle"|"passive"|"moderate"|"aggressive"

12. **Reinforce Trip — POST /**api/reinforce-trip
    Body (similar to Label Trip) writes corrected JSON to a target folder.

D) SENSOR DATA

13. **Sensor Data** — GET /api/sensor-data
    Query: fileName (required). Parses CSV and returns selected sensor columns with file stats.
    Required CSV columns (case-insensitive aliases supported):

- RPM (rpm|engine_rpm|engine_speed)

- Engine Load (engine_load|load|engine_load_percent)
- Intake Pressure (intake_pressure|manifold_pressure|map)

E) SESSION MANAGEMENT

14. **Session ID** — GET /api/session-id
    Returns next available sessionId (3-digits) based on existing files.

F) UPLOAD RECORDS (FIRESTORE)

15. **List Uploads** — GET /api/uploads?limit=20
    Create Upload — POST /api/uploads (body: filename, size, status, etc.)
    Delete Uploads — DELETE /api/uploads?filename=... or ?purgeInvalid=true

G) RLHF (REINFORCEMENT LEARNING FROM HUMAN FEEDBACK)

16. **Start Training** — POST /api/rlhf/train
    Body:

{"max_datasets": x, "force_retrain": false}

17. **Training Status:** Real-time training status and outcome update — GET /api/rlhf/status
18. **Latest Model Info:** Check the path to latest model storage — GET /api/rlhf/latest-model
19. **Pending Datasets**: Labeled datasets that haven't been trained — GET /api/rlhf/pending-datasets
20. **Trained Datasets:** Labeled datasets that have been trained — GET /api/rlhf/trained-datasets

Notes: RLHF endpoints proxy to the HF Space backend (https://binkhoale1812-obd-logger.hf.space) under the hood; training skips datasets listed in trained.txt.

3. SPRINT PROGRESS

WEEK 7

Overview:
We completed **end-to-end integration** from the **UI** to **backend** to **Firebase** and the **Hugging Face cleaning + model stack**, and added a **human-in-the-loop RLHF** training workflow. In parallel, we began the **fuel-efficiency** study on repeated routes and delivered several **Raspberry Pi** reliabilities and UX improvements.

Applications & Links:
UI: https://skyledge-ten.vercel.app

Pipeline: https://huggingface.co/spaces/BinKhoaLe1812/OBD_Logger
Model hub: https://huggingface.co/BinKhoaLe1812/Driver_Behavior_OBD/

Activities:

1. Labeling & data flow (UI ⇄ backend ⇄ Firebase ⇄ HF models).

- **Two ingestion paths** now supported from the UI:
  (i) Clean→Predict first: raw upload → skyledge/raw/ → backend sends to HF cleaner (gap-aware + model-in-the-loop) → **XGBoost** prediction → persist to skyledge/processed/.
  (ii) Label first: raw upload → in-UI streaming timeline labelling → save **human-corrected spans** to skyledge/labeled/ with **provenance** (user, time, source file).
- Both paths preserve **traceability** (file ids, model version used, timestamps) for audit and retraining.

2. **RLHF-style improvement loop (compare-and-train).**

- **Trigger:** From UI "Finetune" or backend schedule.
- **Batch builder:** Diffs **model predictions vs human labels**, selects **unused** labeled sets, and composes a balanced training batch (up-weighting rare classes and disagreements).
- **Objective shaping:**
  - **Sample weights / margin loss** to emphasise mispredicted spans.
  - **Pairwise preferences** ("prefer human over model") for close-call regions.
- **Versioning:** Checkpoints tagged **1.1 → … → 1.9 → 2.0**.
  - **Default**: pipeline uses latest.
  - **Override**: per-vehicle/config **pinning** supported.
- **Safety:** dry-run evaluation before promotion; rollback to any prior tag.

3. **Fuel efficiency — early findings (repeated-route EDA).**

- Collected **10×** runs on a **fixed route** with fuel readings/estimates and route ids.
- EDA: **correlation heatmap** of features vs. fuel rate (direct PID where available, else **MAF-derived** proxy), **timeline overlays** of dynamics bursts.
- Drafted a **0–100 score** seeded by **spike frequency / duration / magnitude** (with hysteresis), plus **idling penalties**. Considered a **tiny temporal encoder** for pattern context while keeping **interpretable spike metrics** for coaching copy.

4. **Raspberry Pi developments.**

- **Parallelised logger** to reduce dropouts at higher cadences.
- **LED indicator** for on-device state feedback (recording / streaming / error).
- **Hardware integration tests** for cable/power/sensor stability.
- **Power optimisation** (process scheduling & sampling duty-cycle) to lengthen field runtimes.

Client Feedback:
Positive on end-to-end integration; encouraged further **Pi integration enhancements** and asked us to **pursue fuel-efficiency scoring** rigorously, with potential to develop an **academic/industrial paper** linking driver behaviour to cost efficiency.

Team Action:
Prioritised **RLHF finetuning** of behaviour models and continued **fuel-efficiency** experiments; documented the paper outline (methods, datasets, baselines, ablations).

WEEK 8

Overview:
We advanced the **fuel-efficiency analysis** and formalised a **hybrid scoring** approach (bounded algorithmic + monotone ML), then **hardened RLHF** (weight-preserving finetunes, API/controls, artifact tracking) and **surfaced model governance in the UI**.

Notebooks:
i) EDA:
https://colab.research.google.com/drive/13Ahp5UIkSB79RZLEzGMliAwR_HqT6aQB?usp=sharing
ii) Scoring Colab notebook:
https://colab.research.google.com/drive/1zy1FSJwN2MAYFWe32F71B1ABHfdfJEmE?usp=sharing
Iii) Comprehensive research on hybrid scoring and conclusion on ideologies, methodologies, and outcomes are documented: https://hackmd.io/@ngFNmXW1RVOfNb7b3NYBJg/efficiency

Activities:

1. **Fuel-efficiency — later experiments (multi-trip aggregation).**

- Aggregated sessions to correlate **standardised fuel consumption (L/km)** vs **driving styles** and key PIDs.
- **Findings:**
    - **Idle time** strongly reduces efficiency (fuel burn without distance).
    - Elevated **mean RPM** and **engine load** are primary predictors of higher consumption.
    - Frequency/duration of sharp dynamics (spikes) are more predictive than pure magnitude.

2. **Hybrid scoring (bounded + length-invariant).**

- **Algorithmic score (0–100)**, no batch min–max: spike **frequency/duration** heavily weighted; **magnitude** capped; **idle** integrated; smooth **sigmoid penalties** and **exponential combiner** ensure boundedness.
- **ML score: HistGradientBoostingRegressor** with **monotonic constraints** (more spikes/idle must not raise the score) + **isotonic calibration** learned once; **group CV** by trip to avoid leakage; features are **length-normalised**.

- **Validation visuals:** distribution alignment (algo vs ML), parity plot, correlation plots (efficiency vs idle/spike rates), and **timeline shading.**

3. **RLHF re-design (to prevent over-penalising / catastrophic forgetting).**

- **Weight-preserving continuation** for XGBoost via **warm-start** to keep useful splits/trees.
- **Reward/Penalty reweighting**: up-weight **human–model disagreements** and **rare classes**, but **cap penalties**.
- **Full API surface:**
    - /train, /predict?version=…, /models, /promote, /preferences

4. **Dashboard UX.**

- **Cards** (mocked and live where available) show post-services state: upload/raw, processing, labelling; driver metrics; and **tips/instructions** placeholders driven by behaviour/efficiency.
- Toggle **preference weights** (e.g., Idle FN penalty, Aggressive recall weight), **cap max datasets**, pick a **model version**, and view per-version **accuracy/CV**.
- **Artifacts & storage:**
    - trained.txt to skip already-used labeled sets; per-version metadata & rollback pinning; storage under skyledge/labeled/ (HITL) and skyledge/processed/.
- **Surfaced in UI**: RLHF phase/status, last train time, active/pinned version, and training configuration.

Client Feedback. Methodology is sound, but **presentation clarity and visuals** underwhelmed. Requested **stronger visual communication** and **more impactful, actionable tips** that drivers can easily internalise.

Team Action:

- Reworked the scoring notebook visuals (clear legends, units, shaded timelines).
- Drafted **tips templates** parameterised by behaviour/efficiency scores; rehearsed **presentation** with narrative structure and “why it matters” framing.

WEEK 9

Overview:
We implemented **segment-level color grading** for timeline insight, added **length-invariant chunking** for streaming/long trips, finalised the **export artifact set** for reproducible backend inference, and **mapped behaviour/efficiency to coaching tips**.

Notebook: Updated from week 8

Activities:

1. **Color grading (segment-level).**

- Compute **inefficiency segment scores** on rolling **10–15 s** windows using the same spike thresholds (τa ,τj)and **rule-based Idle**.
- **Color ramp** (HSL/HEX): **Green ≥ 80**, **Yellow 60–80**, **Orange 40–60**, **Red < 40**.
- **Frontend overlay:** polyline of **SPEED vs time** with per-point color; **background shade** when IDLE_RULE=True. This makes high-waste episodes self-evident.

2. Length-invariant chunking (streaming-friendly).

- Slice trips into **60–120 s** windows (50% overlap). For each window:
  i) derive ACCEL, JERK, IDLE_RULE;
  ii) form **length-normalised** features (freq/min, duration fraction, idle metrics, **p90 accel/jerk**, etc.);
  iii) predict with **sc_ml → gbm → iso_cal**;
- Aggregate with **distance-weighted mean** to produce a **trip score** robust to length.

3. **Export set (versioned, backend-ready).**
Persist together for **reproducibility**:

- **Thresholds** (τa, τj), **Idle-rule quantiles**, and **algorithmic params** (penalty weights *w*, centers/slopes from med/IQR).
- **ML objects:** sc_ml (**StandardScaler**), gbm (**HistGBR** with **monotone constraints**), iso_cal (**IsotonicRegression**).
- **Feature schema:** exact names & order.
- **Version info & seed:** This enables **stateless deployment** across environments and consistent scoring in the backend and UI.

4. Tips templates (score-to-copy mapping).

- **Inputs:** behaviour proportions (**Passive/Moderate/Aggressive** of **driving time**), **Idle fraction**, and **efficiency score** (0–100).
- **Rule:** apply **Idle penalty** first (deduct points if idle high), then map to **coaching tier**:
  - **90–100 (Green):** "Excellent stability. Keep steady throttle; your idle time is minimal."
  - **75–89 (Yellow):** "Good overall. Trim short bursts: ease into speed changes; plan lights to reduce idle."
  - **50–74 (Orange):** "Moderate inefficiency. Reduce rapid throttle swings; watch p90 jerk. Consider eco-cruise where safe."
  - **<50 (Red):** "High fuel waste. Long idles and frequent spikes detected. Avoid hard starts, limit engine brake surges; switch off during long waits."
- **Behaviour-aware inserts:**
  - **Passive↑, Idle↑:** "You're calm on the move, but idling is costly. Shut engine during long stops (>30 s) where safe."
  - **Moderate↑:** "Most trips are smooth; spikes cluster near merges. Anticipate ramp traffic; gentle throttle reduces waste."

    - **Aggressive↑:** “Frequent sharp accelerations. Practice progressive pedal: target < 2 spikes/min for this route.”
- **Optional cost framing:** show **estimated fuel cost delta** from baseline (route-wise median) to make tips tangible.

Team action:
Added **distance-weighted score** verification tests using the exported artifacts; wired the color-grading overlay into the UI timeline.

4. SPRINT REVIEW (CRITICAL REVIEW OF THE PRODUCT)

What Was Demonstrated to SkyLedge (Weeks 7–9)

- **End-to-end data loop:** UI upload → Firebase (raw/, labeled/, processed/) → HF cleaner/model → backend → **UI review & labeling**.
- **RLHF loop:** diff vs human, **weighted finetunes**, **pairwise preferences**, **versioning** (1.x→2.0), **pinning**, **rollback**, and **UI governance**.
- **Fuel-efficiency study:** repeated-route EDA (idle, RPM, load), **hybrid scoring v3** (bounded algorithmic + monotone ML), **calibration & visuals**.
- **Raspberry Pi reliability:** parallel logging, LED indicator, power optimisation, integration tests and validations.
- **Color-graded timelines & chunking:** segment-level overlays (green→red), **length-invariant windowing**, **distance-weighted trip score**.
- **Export kit:** thresholds, idle quantiles, algorithmic params, **scaler/GBM/iso_cal**, schema, version/seed.
- **Tips templates:** behaviour/idle/score → actionable copy (driver-facing).

FEEDBACK PROVIDED BY CLIENT

- Week 7: Strong progress; **double-down on fuel efficiency** and integration polish.
- Week 8: **Improve presentation clarity** and **make tips more impactful**; visuals should tell the story.

CRITICAL ANALYSIS OF PROGRESS & FEEDBACK

Progress against objectives:
We moved from working pieces to a **cohesive, governed system**: data flows are tracked; models are versioned and recoverable; and scoring has a **principled foundation** (bounded algorithmic + monotone ML). Hardware reliability improvements enable more consistent data capture. The **driver-facing layer** (color overlays + tips) now closes the loop toward **behaviour change**.

Why this matters:

- **Trust & auditability:** RLHF with versioning/pinning/rollback and artifact exports makes the system **operable** in production.
- **Scientific grounding:** Efficiency framing emphasises **consistency**, not speed; **idle**, **frequency**, and **duration** drive waste—aligned with domain intuition and EDA.
- **Actionability:** Segment colouring + concise tips transform analytics into **coachable moments**.

Challenges & mitigations:

- **Presentation gaps →** Addressed with improved **visual hierarchy**, **color-graded timelines**, and concise **tips**.
- **Risk of overfitting in RLHF →** Introduced **warm-start** + **capped penalties**; added UI **preference controls** and **artifact tracking**.
- **Trip-length variability →** Solved via **length-invariant features** and **chunking** with distance-weighted aggregation.
- **Field reproducibility →** Finalised **export kit**; added pipeline tests to confirm equivalence (backend vs notebook).

Key achievements:

- **Production-grade loop:** upload ↔ label ↔ finetune ↔ deploy with **governance**.
- **Hybrid scoring v3** with **boundedness, monotonicity, and calibration**.
- **Segment-level insight** (color grading) and **tips templates** tied to scores.
- **Raspberry Pi** reliability path for sustained data capture.

Reflection & Next Steps:

- **Validate scoring at scale:** run A/B on two route types; report **parity**, **R^2/MAE**, and **stability vs length**; instrument disagreement heatmaps (rule vs ML).
- **Harden coaching UX:** integrate **dynamic tips** in UI cards; add **cost deltas** vs baseline; localise tone & length.
- **Operationalise RLHF:** ship **/promote** gate with acceptance checklist (CV, confusion, edge clips); surface **per-class metrics** and **rare-class recall**.
- **Backend parity tests:** CI step that loads **export kit** and verifies equality of trip scores vs notebook on fixtures.
- **Raspberry Pi pilot:** deploy LED-enabled logger to 1–2 vehicles; collect **multi-day** logs to broaden route/context coverage.
- **Paper/report outline:** document method (hybrid scoring, RLHF), datasets, baselines, and ablations for a **workshop or industry whitepaper**.
- **Deliverables handover:** The team will work on comprehensive documentation and product packaging handover to the client since this is a concluding phase of the project.

This sprint converted promising methods into a **traceable, reproducible, and driver-actionable** system, positioning us to measure real-world impact in the next iteration.

5. RETROSPECT (CRITICAL REVIEW OF THE PROCESS)

Building upon our technical foundations from Sprint 5, we entered Sprint 6 with established methodologies for driver behaviour classification and fuel efficiency prediction, yet this sprint revealed fundamental challenges in team coordination and product definition that challenged our progress. While we achieved significant improvements in data collection quality and model performance, the sprint was characterised by increasing team fragmentation and uncertainty around our final deliverable.

Positively, we maintained our core project management practices through task tracking and continued our stand up meetings, though the effectiveness of these communications diminished as the sprint progressed. Sprint 6 has exposed gaps in our shared understanding of the final product vision and highlighted the consequences of non-collaborative working practices on team cohesion and delivery confidence.

DATA COLLECTION AND BASELINE ESTABLISHMENT

A major achievement in this sprint was the substantial improvement in our logging accuracy and data quality through systematic data collection. We successfully captured separate runs of the identical driving route, each representing distinct driving styles, providing us with a valuable baseline dataset for comparative analysis. This controlled approach to data collection represents a significant step forward from our previous challenges with data quality and labelling accuracy, as it enables us to isolate the impact of driving behaviour on fuel consumption and efficiency with much greater confidence. The consistency of having the same route traversed multiple times under varying driving conditions provides the foundation for robust validation of our prediction models and offers clear, empirical evidence of how different behaviours translate to measurable efficiency outcomes.

MODEL PERFORMANCE AND TRAINING PROGRESS

On the technical front, we made encouraging progress with our model training pipeline, reaching a stage where we are nearly satisfied with the outputs generated from newly ingested data. The improvements to our training process, built on the unsupervised learning approaches established in Sprint 5, have begun to yield more consistent and reliable predictions for driver behaviour classification. While we haven't yet achieved complete confidence in all edge cases, the current model performance suggests we are approaching a level of accuracy that could be considered production ready, with the addition of RLHF, we see this only getting better, provided we can maintain this trajectory and resolve remaining validation concerns and reliability.

PRODUCT DEFINITION AND DELIVERY UNCERTAINTY

Despite our technical progress, Sprint 6 was marked by significant confusion around what our final product should look like and how it should be delivered to the client. As a team, we have struggled to maintain alignment on the scope and format of our deliverable, with the team seemingly holding slightly divergent views on what the final product is. This misalignment has created an environment where, while we are objectively near completion of the development work, there remains substantial uncertainty about whether what we are building delivers valuable insights.

TEAM FRAGMENTATION AND SILOED WORKING

Perhaps the most challenging development of this sprint has been the breakdown in team collaboration and the emergence of siloed working practices. Unlike earlier sprints where team members worked in close coordination, Sprint 6 saw individuals increasingly working independently on separate components without meaningful integration of ideas. This fragmentation has resulted in a situation where we often lack clarity about what each other are working on, undermining our ability to identify dependencies, avoid duplication of effort, or leverage each other's insights. The collaborative spirit that characterised our earlier sprints has given way to isolated development work, which has caused some confusion and velocity slowdown.

COMMUNICATION BREAKDOWN AND MEETING EFFECTIVENESS

Contributing to our misalignment in parts has been a notable deterioration in our communication and meeting effectiveness. We have, in many ways, lost our way regarding structured team interactions beyond the basic check ins. The ad hoc team meetings that previously helped us navigate technical decisions and maintain alignment have become less frequent, largely due to the impact of other coursework and uncertainty on final delivery, along with fatigue of working on and having to adapt due to feasablity of the final product.

PROXIMITY TO COMPLETION

Despite the challenges outlined above, we find ourselves in the somewhat odd position of being near completion of our development work while simultaneously feeling unsure about what completion of our development actually means. The core technical components are largely in place; our models are performing at satisfactory levels, and we have established solid data collection and validation methodologies.

Overall, Sprint 6 has been a technically productive but organisationally challenging period. While we have achieved meaningful progress in data quality and model performance, these advances have been coupled by team misalignment, communication breakdowns, and fundamental uncertainty about our final deliverable. As we move into our final push, it is critical that we address

these coordination and alignment issues through dedicated team planning and re-alignment sessions.

6. LESSONS LEARNED (CRITICAL REVIEW OF SPRINT ONE EXPERIENCE AND FUTURE PLAN)

Key Lessons from Sprint Three

Role specialization continued to be effective this sprint, allowing each member to focus on their domain and make significant progress in parallel. However, as we refined our final deliverable and adjusted some of the project features, we found ourselves revisiting earlier design choices. This led to certain redundant work and sections of code or workflow that became outdated. While our productivity remained high, it highlighted the need for clearer scope definition before each sprint.

We also noticed that while our communication remained positive and trust-based, the high level of independence meant that team members were not always aware of the deeper technical details of each other's work. This created small gaps in system integration and understanding. The team also received feedback that our presentation and reporting could be improved to better communicate technical progress, especially to non-technical audiences.

Despite these challenges, Sprint 3 marked the point where our project direction and scope finally solidified. The core deliverables—data automation, unsupervised driving behavior detection, and the web application—are now well-defined, and development is aligned across all members.

Recommendations for Future Sprints

- Finalize system scope and technical requirements before starting implementation to avoid redundant or outdated work.
- Maintain the current specialized work structure but introduce short weekly "sync sessions" to ensure all members understand major updates across domains.
- Dedicate more time to refining presentation materials and documentation, ensuring clarity for both clients and assessors.
- Begin structured integration testing earlier in the sprint to catch incompatibilities between independently developed components.
- Preserve the current momentum and focus on delivering a cohesive, user-ready product in the next phase.