EAT40005 Engineering Technology Projects A

Individual Work Log

PROJECT NAME:	OBD-II Based Predictive Maintenance System		
STUDENT NAME:	**Dang Khoa Le**		
STUDENT ID:	**103844421**	**WEEK # (& dates covered):**	**#11 (20–24 May 2025)**

TASKS	STATUS	TIME SPENT	ACTION ITEM/NOTE
Developed dashboard /ui route using Jinja2 and static HTML/JS/CSS	Completed	3 hours	Integrated dashboard into FastAPI backend to visualize data pipeline status with card-based updates using polling /events API.
Dockerize the app environment	Completed	1 hour	Instantiate Python and FastAPI environment with caching to store log data
Extended /ingest to support status=start/end signals for visual tracking	Completed	1 hours	Allowed dashboard to track real-time start, processing, and completion of incoming logs.
Implemented chart visualizations for correlation heatmaps and trend plots	Completed	2 hours	Automatically generated and rendered with each data session, exposed in expandable dashboard cards.
Discussed and designed drive_style labeling methods with client	Completed	1 hour	Finalized hybrid method: driver-initiated UI tagging + manual labeling post-drive; unsupervised clustering considered as backup.
Raspberry Pi upload planning with intermittent network solution	Pending	2 hours	Since Raspberry Pi can only connect with 1 Wifi port at the time, I proposed collecting data every 5 seconds, then batch uploading to server, supporting streaming-like logging in limited WiFi environment. Design is not supported by Sadman; however, we will still work on the optimal solution for this.
TOTAL WEEKLY TIME SPENT		**10 hours**	

TASKS PLANNED FOR NEXT WEEK	EXPECTED COMPLETION
Implement solution for drive_style label	Week 12
Finalize Raspberry Pi timed upload loop (5s rolling log) - once approved	Week 12
Explore unsupervised clustering (e.g., KMeans) for backup drive_style labels	Week 12

Summary/weekly reflection for Week 11:

- **Key Tasks Done:**

This week focused on creating a real-time **dashboard service** that interfaces with our FastAPI backend. Using /events, the frontend dynamically updates visualization cards for each data session, displaying statuses (started → processed → done) along with automatically rendered charts (heatmaps and sensor trends). This gives clients and developers a clear view of data flow and pipeline health.

The /ingest route now supports status=start/end signals to log sessions more clearly, allowing visualization even for Raspberry Pi sessions that batch every 5–10 seconds (interval configurable).

We discussed and finalized the **hybrid approach to drive_style labeling**:

- Real-time tagging via dashboard (radio buttons per session).
- Post-drive manual review.
- Optional backup via unsupervised ML (clustering based on acceleration).

Due to Raspberry Pi's WiFi limitations (cannot connect to both OBD and internet simultaneously), I designed a **timed upload loop**: collect continuously for 5 seconds, then reconnect to the server and push logs. This strategy preserves near real-time feel while working within hardware constraints. Although it hasn't been widely approved, this is a good design that once proven the work, could be integrated to the system for a fully automated workflow.

- **Key Learning:**

- Real-time data dashboards enhance observability and debugging during development.
- Hybrid labeling (manual + in-drive UI) balances accuracy and practicality in resource-constrained setups.
- Interval-based data upload is a realistic compromise when hardware limits full-time connectivity.

- **Literature/Resources Reviewed:**

- Chart.js for data visualization.
- Jinja2 templating with FastAPI.
- FastAPI static file routing.
- Raspberry Pi WiFi polling strategies.
- Discussions on drive behavior labeling in automotive telemetry papers.

- **Issues Faced:**

- Raspberry Pi cannot maintain simultaneous connections to both OBD-II and internet.
- Live drive_style toggling requires stable frontend-backend linkage — not trivial over intermittent networks.
- Chart responsiveness and caching on dashboard required fixes for real-time display.