Sprint End Report

Sprint: S4
Sprint Dates: 12/11/2025-24/11/2025
Sprint Goal: Prototype the user interface and implement the first version of the backend to expose machine-state data.
Product Owner: Joaquin Ordieres
Scrum Master: G.Muller
Team Members: Sandini Suraweera, Léo Marquant, Auxence Letellier, Jairo Paez Leal, Lorenzo Niola, Hawazen Hawalah, Tim Haubner, Anna Gyllerup, Tora Fredheim, Gabrielle Muller

1. Sprint Review Meeting Details

Date of Meeting	24/11/2025
Start Time	16:20
End Time	16:55
Duration	00:35
Location / Mode	In-person / ~~Online~~
Participants	All team members

2. Sprint Review (Demonstration of Increment & Feedback)

Increment / Feature	Description	Demo Status (Done/Partial)	Stakeholder Feedback	Action Items/Owner
Feature 1 – Real Data Integration & UI Controls	Connected the Streamlit interface to the real database (2020-2022), replacing all mock data. Added dynamic date range pickers to filter visualizations effectively.	**Done**	The connection is stable, and the timeline selection works well with the real dataset.	Continue refining the UI based on specific user needs. (Owner: Frontend Team)
Feature 2 – Alarm Backend & Optimization	Optimized the SQL queries based on the logic finalized by the Data Team .	**Done**	The query is faster, and the filtered results are relevant.	Confirm the filtered data are appropriate

Feature 3 – Machine State Detection (Data Layer)	New Logic: Created an activity score based on distinct sensor counts per second. Sampled 5 average non-consecutive days and applied a 15-second moving average. Determined statistical thresholds (14 and 20). **Final Classification:** -Idle: 0 (No activity) - Low: 1–14 - Intermediate: 15–20 - High: >20	**Done**	Reflects actual machine behavior better than previous methods, need to associate them to concrete and understandable states.	
Feature 4 – Energy Monitoring	Initiated the development of energy consumption endpoints and basic visualization. Integrated the only relevant energy variable identified so far: variable 630 originally expressed in percentage. Converted the raw % values into kWh and computed consumption for the previously identified "relevant days". Implemented a first analysis method to extract and aggregate real spindle energy consumption	**Partial**	A parallel validation method is being developed to compare results.	Finalize correlation between spindle consumption and machine states + include them in the main program (backend and interface) Owner: data team/UI/backend

3. Sprint Retrospective (Team Reflection & Improvements)

Category	Observations	What Went Well	What Can Be Improved	Action Items / Owner
Process	A wide variety of tasks were executed.	Backend Velocity: SQL optimization resolved the slowness issues from S3. Integration: The connection between data_service and interface is functional with real data.	Some coordination gaps within and between teams led to accumulated and unnecessary workload	Maintain app.py as a technical testing tool for future developments. (Backend Team)
Communication	Determination of "Critical Alarms".	The Data Team provided clear filtering rules for the Backend.	Check if the determined “relevant” alarms are adapted to the client’s needs	/
Tools / Environment	Config.yaml created and hidden from public		Need to share the definitive version to the whole team	Share and explain how to use the file to the rest of the team (backend)

4. Summary & Next Steps

- **Key Takeaways:**
    - The interface has transitioned from a static mockup to a realistic, interactive MVP connected to real data for end-users.
    - The backend now satisfies the first two requirements: Machine State hours aggregation and Alarm visualization.
- **Improvements for Next Sprint (S5):**
    - Energy Monitoring: Finalize the logic by correlating consumption data with specific machine states and integrate it into the final application structure.

- Testing: Conduct End-to-End testing to ensure stability across Data, Backend, and Frontend layers before the final polish.
- Contextualization: map the new activity levels (Idle/Low/Int/High) to concrete operational contexts.
- Validation: Confirm the final list of critical alarms to display and define precisely which metadata (duration, code, description) is most valuable to the user.

- **Additional Notes / Decisions:**
  - app.py will be maintained as an internal diagnostic tool for developers, while main.py remains the user-facing interface.