**Final Project Report**

# Title:

Smart Real-Time Vehicle Monitoring System

# Submitted by:

Victoria Grand — ID: 211856208  
Noa Rofe— ID: 208328286

# Instructor:

Dr. Guy Tel-Zur

## Executive Summary

This project presents a smart real-time monitoring system for vehicles that simulates various car sensor data such as engine temperature, speed, and tire pressure. The system utilizes IoT protocols and cloud services to collect, process, analyze, and visualize data, enabling proactive detection of anomalies and real-time alerts both in the cloud and at the vehicle's edge. The architecture incorporates edge devices, cloud storage, live dashboards, and alert mechanisms.

## Project Body

# Objectives:

- Simulate and transmit real-time vehicle sensor data.  
- Detect anomalies in critical car parameters.  
- Send alerts to the cloud and back to the edge system.  
- Visualize data in real-time using dashboards.  
- Store and analyze data securely in the cloud.  
- Demonstrate integration with multiple cloud services.

# Implementation:

# Architecture Overview:

The project architecture is built as a modular pipeline that reflects an IoT-to-cloud flow. Below is the system diagram:  
  
📡 Simulator (main.py)  
⬇️  
🟦 MQTT Broker (Mosquitto)  
⬇️  
🛠 Telegraf  
⬇️  
📦 InfluxDB  
⬇️  
📊 Grafana Dashboard

⬇️

☁️ AWS Lambda (processing)

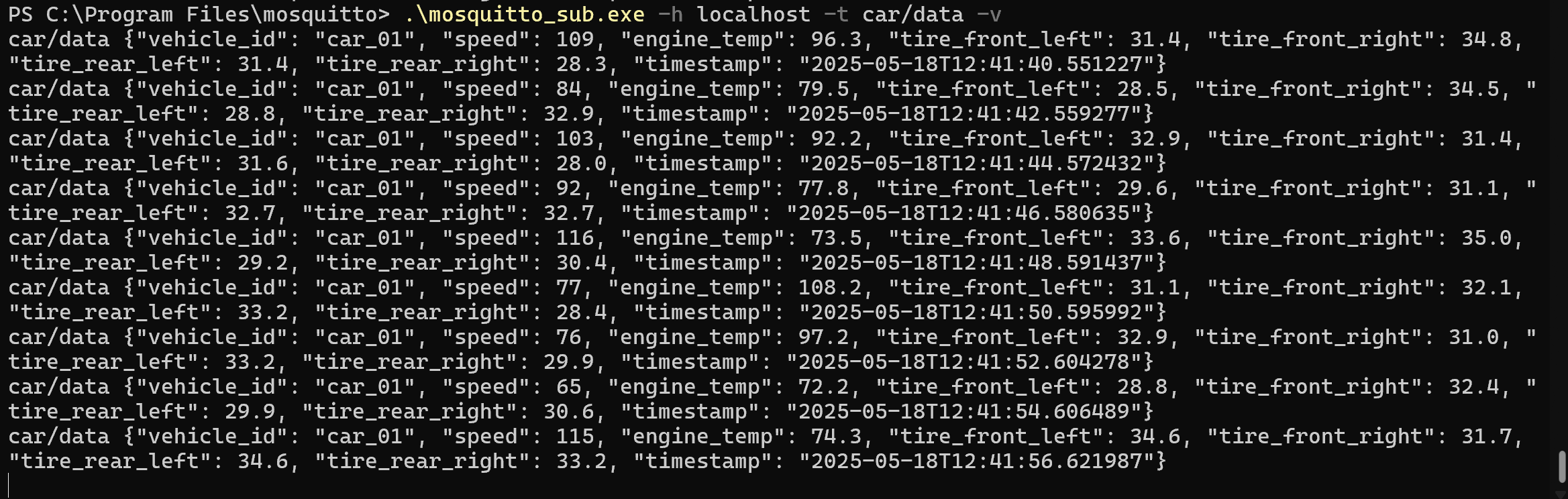
⬇️

🔗 AWS SNS / Webhook (notifications)

# Component Breakdown:

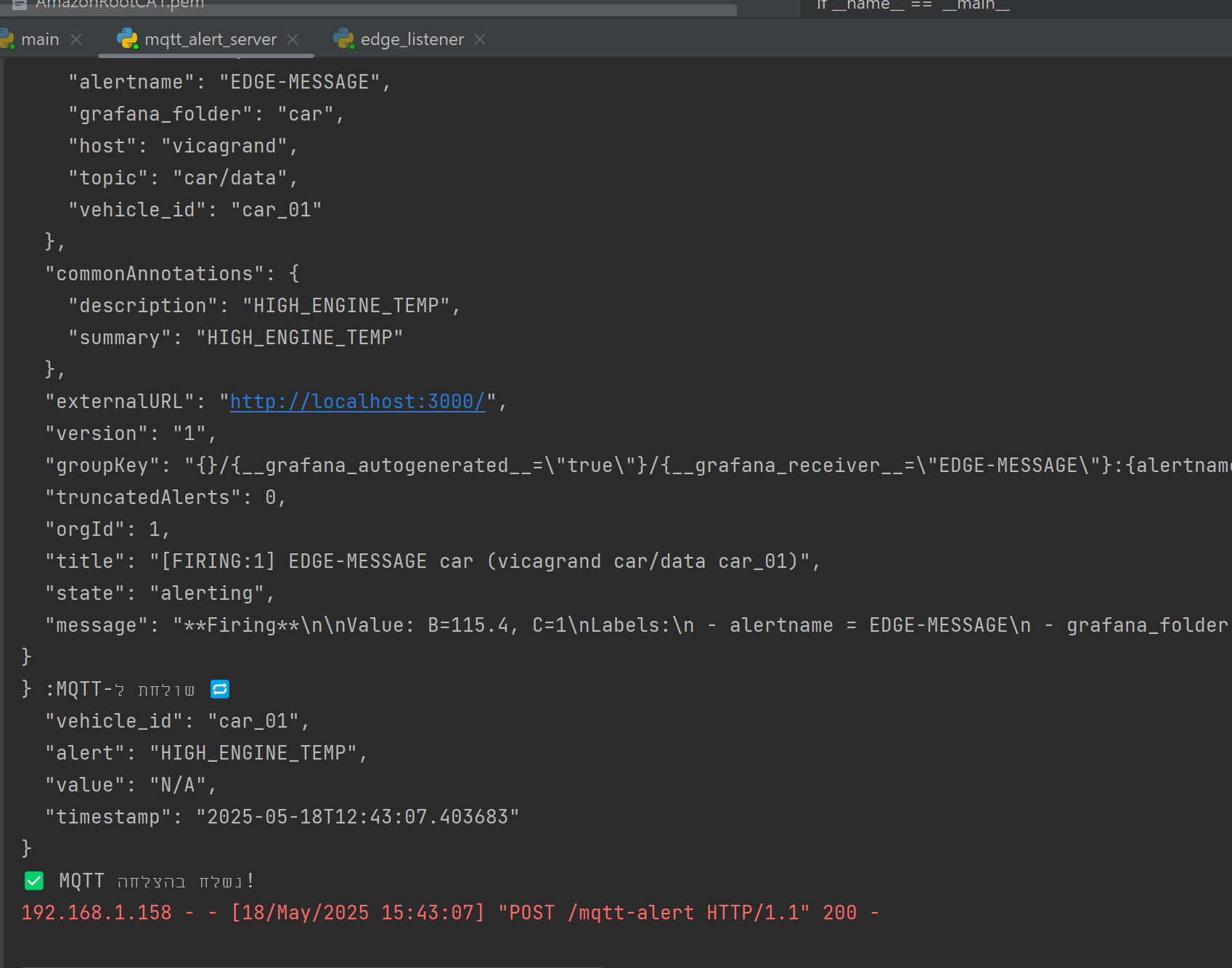
1. Edge Simulation:  
 - The Python script `main.py` simulates sensor data every 2 seconds.  
 - Data includes speed, engine temperature, and four tire pressures.  
 - Data is published to a public MQTT broker (`broker.hivemq.com`) and Mosquitto instance.  
A screen shot of a computer screen

AI-generated content may be incorrect.



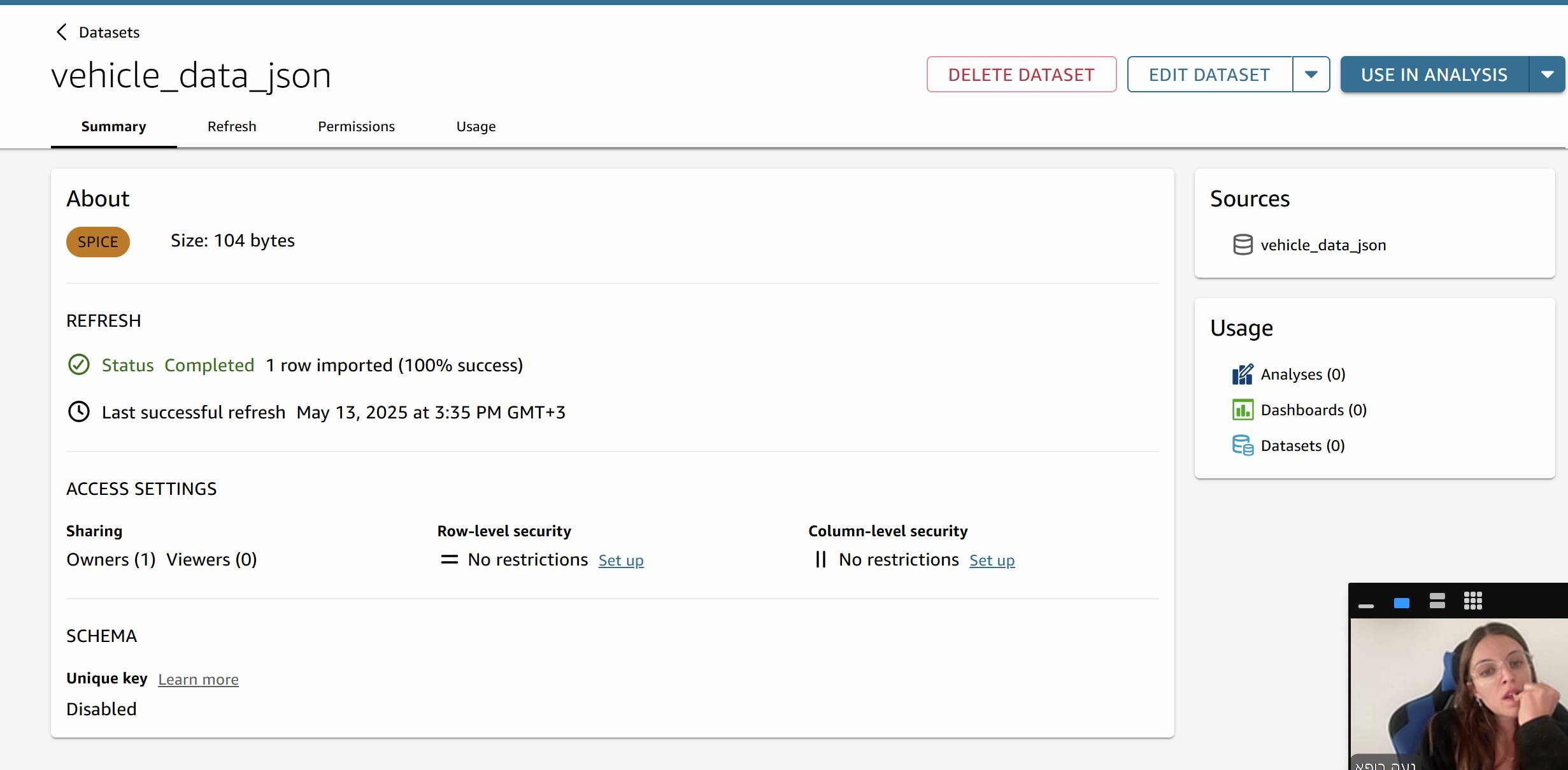
2. Edge Processing & Alerting:  
 - An `edge\_listener.py` script subscribes to the MQTT topic.  
 - It checks for anomalies (e.g., temp > 110°C, tire pressure < 30).  
 - Alerts are printed in the terminal and published back to the MQTT topic `car/alert`.  
A screen shot of a computer

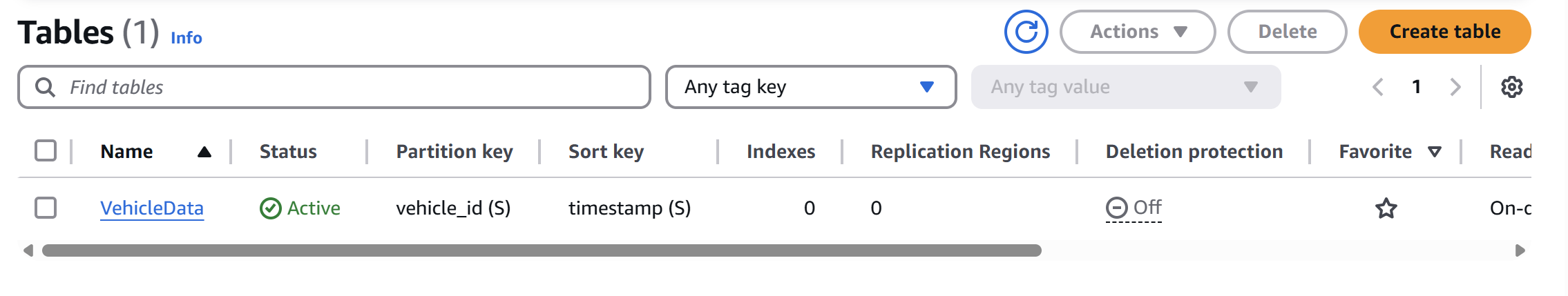
AI-generated content may be incorrect.



A screenshot of a computer program

AI-generated content may be incorrect.  
3. Cloud Integration:  
 - Data is also sent to AWS IoT Core.  
 - Amazon S3 stores the data periodically in JSON format.  
 - AWS Glue and Athena structure the data and allow SQL-like querying.  
 - Amazon QuickSight visualizes historical data and trends.





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

4. Live Dashboard:  
 - Telegraf reads MQTT topic `car/data` and forwards it to InfluxDB.  
 - Grafana is configured to query InfluxDB and render time-series dashboards.  
 - Dashboards include graphs for speed, temperature, and all tire pressures.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

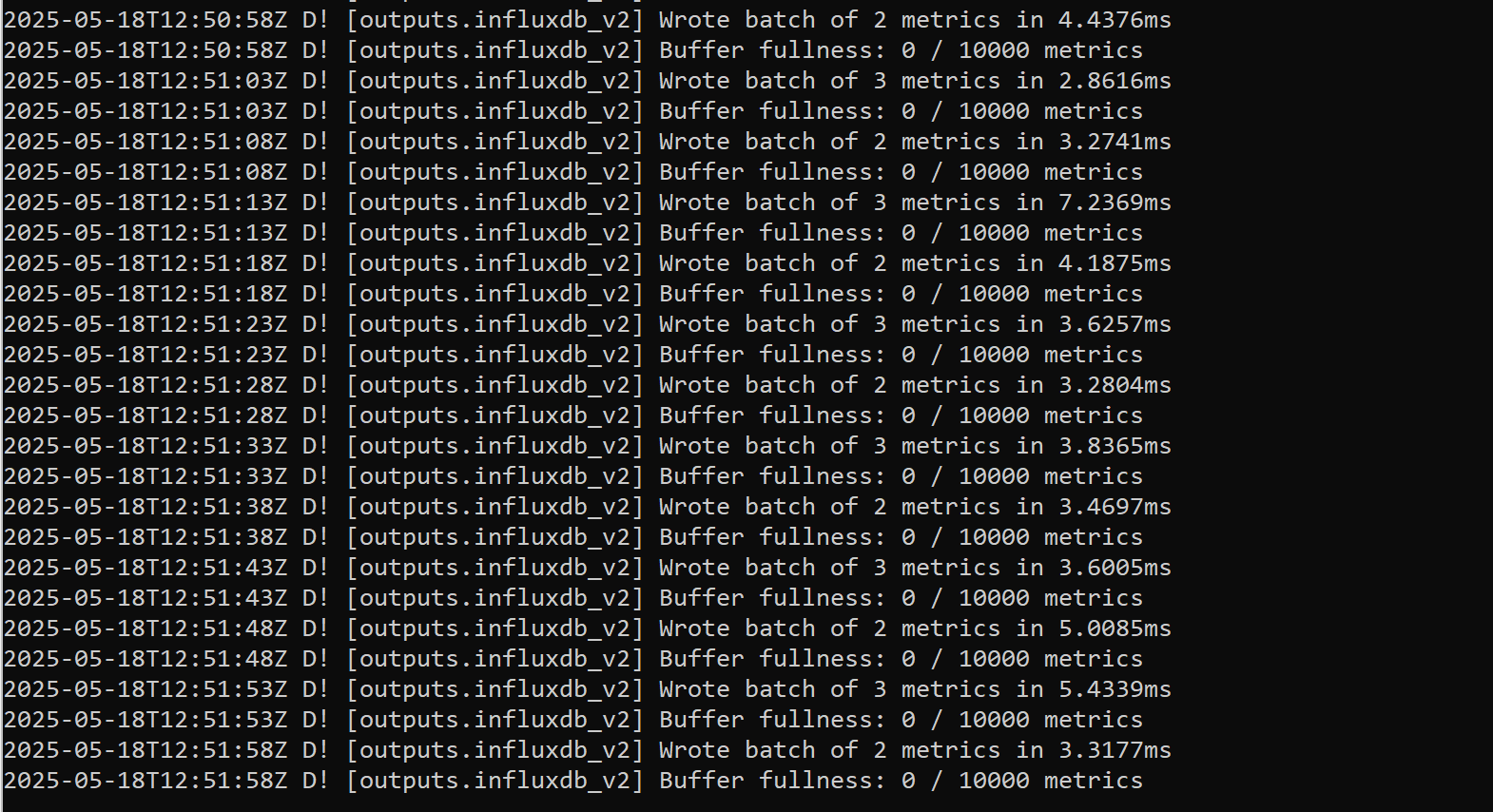
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



# Results:

- AWS SNS was used to send email notifications when anomalies were detected.  
- Real-time data is accurately displayed in Grafana.  
- Alerts appear both in edge logs and MQTT topic subscribers.  
- Athena successfully queried structured sensor data from S3.  
- QuickSight enabled exploration of aggregated historical values.

A screenshot of a car engine

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

# Conclusions:

The Smart Real-Time Vehicle Monitoring System successfully showcases the integration of edge computing, cloud services, and real-time data analysis. From data simulation to dashboard visualization, the system demonstrates how IoT-based architectures can enhance awareness, safety, and decision-making for vehicle monitoring. Leveraging a rich set of AWS services alongside open-source tools like InfluxDB and Grafana, the project reflects a scalable and secure solution that can evolve with future needs such as predictive maintenance and fleet-wide deployment.

* The system achieved full end-to-end flow: simulation, edge processing, cloud logging, and visualization.
* The use of multiple cloud services demonstrates extensibility and scalability.
* Real-time capabilities allow immediate insights and alerts.

## Security,Scalability and Future Enhancement

**- Security:**  
 - IAM roles were used in AWS to protect S3 and QuickSight access.  
 - Topic-based separation ensures organized message flows.

- AWS Lambda was integrated with API Gateway using IAM-based execution roles, and was granted scoped sns:Publish permissions to ensure secure and minimal access to SNS topics.

- Incoming alerts from Grafana were routed through HTTPS via Amazon API Gateway, ensuring that only authenticated HTTPS POST requests could invoke the alert handling logic without exposing direct access to internal AWS resources.  
  
- **Scalability**:

* Using **Amazon S3** for cloud storage allows the solution to handle large volumes of structured and unstructured data with virtually unlimited capacity.
* **InfluxDB**, a time-series database, efficiently stores and indexes high-frequency IoT data, making it ideal for real-time telemetry and sensor-based analytics.
* The use of **Telegraf** ensures data is ingested with minimal latency and high throughput.
* **AWS SNS** provides a reliable and scalable notification mechanism to deliver alerts instantly via email, SMS, or other endpoints.
* As the number of vehicles or monitored parameters increases, new simulators or sensors can be added without rearchitecting the system.
* The architecture supports multi-vehicle deployment with minimal changes.

- **Future Enhancements**:

* Add GPS/location data to enrich context.
* Integrate predictive maintenance with machine learning.

**Cloud Services Used:**  
1. AWS IoT Core – MQTT broker integration  
2. Amazon S3 – Object storage for raw data  
3. AWS Glue + Athena – Data cataloging and SQL queries  
4. Amazon QuickSight – Business intelligence visualization  
5. InfluxDB + Grafana – Real-time dashboard and analytics

6. Amazon SNS – Notification service for sending alerts to email/SMS

# References & Sources:

- AWS Documentation: https://docs.aws.amazon.com/  
- Grafana Labs: https://grafana.com  
- HiveMQ MQTT Docs: https://www.hivemq.com/mqtt/  
- Telegraf Docs: https://docs.influxdata.com/telegraf/

## Appendices

- Python scripts: `main.py`, `mqtt\_to\_s3.py`, `mqtt\_to\_grafana.py`, `edge\_listener.py` and else…  
- Configuration files: Telegraf config, Grafana dashboard JSON  
- Screenshots: dashboards, AWS console, MQTT logs  
  
Note: Code can also be found in the GitHub repository : <https://github.com/vicagrand/Smart-Real-Time-Vehicle-Monitoring-System.git>