Technical report
- Car Fleet Management System -

Isache Bogdan, Doltu Eliza, Popa Andrei Sabin, Stegaru Razvan

May 13, 2025

1 Setting the limitations of the system

1.1 Purpose of the system

The purpose of the system is to manage the car fleet of a company. The system will allow the user to add, remove, and update information about cars, as well as track their maintenance and repair history. Through a GPS system, each car will be monitored and various information about it will be extracted. The system will generate reports on the car fleet, such as total mileage, maintenance costs, hours in traffic and repair history. On this basis, the program will suggest improvements to the fleet, such as replacing certain cars, changing their maintenance schedule, optimizing each car's route. The accent will be on making the fleet system as **efficient and eco-friendly as possible, reducing substantial the Co2 emissions and the fuel consumption.** This is the main idea that will differentiate this application from the rest:

- Fleet Complete
- Samsara
- Verizon Connect

1.2 Limitations of the system

The system will not be responsible for the physical maintenance of the cars, such as oil changes, tire rotations, or other repairs. It will only track the maintenance history and provide reminders and AI recommendations for scheduled maintenance. The system will not be responsible for the physical GPS installation in the cars, but it will work with existing GPS systems. The system will not be responsible for the physical security of the cars, such as theft prevention or tracking stolen vehicles. It will only provide information about the car's location and status.

2 Target audience

The target audience for the car fleet management system includes fleet managers, logistics coordinators, and company executives who are responsible for managing a fleet of vehicles. The system is designed to be user-friendly and accessible to individuals with varying levels of technical expertise. We aim to provide a solution not only to small, startups companies, but also to large ones, with a fleet of hundreds of cars. The system will be designed to be scalable and flexible, allowing it to adapt to the needs of different organizations. The main advantage of the system will be the ability to make the company much more eco-friendly and helping it in achieving a green certificate.

3 Technical details

In this section, we will provide more information about an existing solution closed to the one proposed by us - Fleet Complete.

3.1 Architecture

1. **Cloud Infrastructure (AWS-Powered)**
   Fleet Complete leverages Amazon Web Services (AWS) to ensure global scalability and reliability. This infrastructure supports over 50 million daily requests and processes more than 1 billion IoT data points, including vehicle speed, location, and diagnostics.

2. **FC Hub: Centralized Fleet Management Interface**
   Primary interface for fleet managers, offering:
   - Live Maps & Dashboards: Real-time tracking of vehicles and assets.
   - Automated Reporting: Customizable reports and event notifications.
   - Mobile & Web Access: Accessible via both web browsers and mobile devices.
   - Third-Party Integrations: Seamless integration with other FC products and external applications.
3. **CONNVEX: Big Data & Analytics Engine**
   Fleet Complete's cloud-based IoT platform that:
   - Aggregates Data: Collects and processes vast amounts of anonymized vehicle and asset data.
   - Predictive Analytics: Provides insights into vehicle health, driver behavior, and maintenance needs.
   - Ecosystem Collaboration: Facilitates data sharing among OEMs, suppliers, and fleet operators.
4. **API & Developer Tools**
   Fleet Complete provides a comprehensive API that allows for:
   - Data Access: Retrieval of vehicle diagnostics, location data, and driver behavior metrics.
   - Custom Integrations: Integration with third-party applications and services.

3.2 Technologies

- AI & Machine Learning: Utilized in FC Vision for detecting unsafe driving behaviors and providing real-time feedback.
- Predictive Maintenance: Analyzes vehicle data to predict and prevent potential failures, reducing downtime.

- Compliance Tools: Includes electronic logging devices (ELDs) and inspection tools to ensure regulatory compliance.
- Scalable Architecture: Built on AWS to accommodate growing data volumes and user bases.

3.3 Marketing approaches

1. **Strategic Partnerships**
   Fleet Complete has established partnerships with major telecom providers and vehicle manufacturers to expand its market reach and offer integrated solutions.
2. **Customer-Centric Solutions**
   The company emphasizes customizable solutions tailored to various industries, ensuring that businesses of all sizes can optimize their fleet operations effectively.
3. **Global Presence**
   With operations in North America, Europe, Mexico, and Australia, Fleet Complete leverages its international footprint to cater to a diverse clientele and adapt to regional market needs.

4 Requirements identification

4.1 Functional requirements

1. The system must allow the user to add, remove, and update information about cars in the fleet.
2. The system must track the maintenance and repair history of each car.
3. The system must provide reminders for scheduled maintenance.
4. The system must generate reports on the car fleet, including total mileage, maintenance costs, hours in traffic, and repair history.
5. The system must provide suggestions for optimizing the fleet, such as replacing certain cars, changing their maintenance schedule, optimizing each car's route, utilizing various ML algorithms. This will be the main

focus of the project, as we want to make the fleet system as efficient and eco-friendly as possible, reducing substantial the Co2 emissions and the fuel consumption.

6. The system must allow the user to view the location and status of each car in real-time.

7. The system must work with existing GPS systems to track the location and status of each car.

8. The system must be scalable and flexible, allowing it to adapt to the needs of different organizations.

9. The system must be secure, protecting sensitive information about the cars and their maintenance history.

10. The system must be compatible with existing GPS systems and other software used by the organization.

11. The system must be able to handle a large volume of data, including information about hundreds of cars and their maintenance history.

4.2 Non-functional requirements

1. The system must be user-friendly and accessible to individuals with varying levels of technical expertise.

2. The system must be reliable, providing accurate and up-to-date information about the car fleet.

3. The system must be able to generate reports quickly and efficiently, allowing the user to access the information they need in a timely manner.

4.3 Overview of technologies used

The system will use a blend of cloud services from both Google and Microsoft on backend. We will mostly use Google Cloud services and API's, but also incorporate some API's from Microsoft Azure. We settled with these technologies mostly because we are familiar with them and have previously

worked at projects utilizing them, so learning these technologies won't slow the development process. Another big advantage for us is the price of this technologies. Both offer free trials and credits for students, so, when talking about the economical part this is the best choice. Also, both Google Cloud and Azure ensure global scalability and the infrastructure to support over 50 million daily requests and processes more than 700 million IoT data points, including vehicle speed, location, and diagnostics.

4.3.1 Google Services

In this section we will present some of the Google Cloud services that we will use in our project:

1. AppEngine
    - Use: Host our web dashboard (e.g., admin panel, trip history, fleet status).
    - Why: It's a fully managed platform, so we don't have to worry about servers — just deploy your code and it scales automatically.
2. Cloud Firestore
    - Use: Store and sync real-time vehicle data like GPS location, engine status, and driver logs.
    - Why: Real-time updates and offline support make it great for tracking.
3. Cloud Functions
    - Use: Respond to events like geofence triggers, scheduled maintenance alerts, or driver check-ins.
    - Why: It's lightweight, pay-as-you-go, and perfect for background logic.
4. BigQuery
    - Use: Perform deep analytics on vehicle usage, route efficiency, fuel consumption, and driver behavior.
    - Why: Ideal for generating insights and reports from large datasets.

4.3.2 Google API's

The following 3 API's will be used:

1. Firebase Cloud Messaging (FCM) API
    - Use: Send push notifications to drivers or managers — like maintenance alerts, trip summaries, or safety warnings.
    - Example: "Vehicle 27 exceeded speed limit" → push notification to fleet supervisor.
    - Integration: Trigger from Cloud Functions via FCM API.
2. Vertex AI Prediction API
    - Use: Run predictions using trained machine learning models.
    - Examples:
        - Predict when a vehicle will need maintenance based on mileage and usage patterns
        - Score driver behavior for safety monitoring
        - Estimate fuel consumption for a route
    - How it works: You train a model (using BigQuery + Vertex AI), deploy it, then send data via the Prediction API and get smart insights back.

4.3.3 Microsoft Azure API's

The following 3 API's will be used:

1. Azure Maps
    - Use: For geolocation, routing, and traffic data.
    - Example: Real-time traffic updates for route optimization.
    - Integration: Use REST API to fetch map data and integrate with our dashboard.

2. Azure Cognitive Services

    - Use: For image recognition (e.g., license plate recognition) and natural language processing (e.g., driver feedback analysis).
    - Example: Analyze driver feedback from text messages or emails.
    - Integration: Call the API from our backend to process images or text.

3. Azure Functions

    - Use: For serverless computing to handle events like new vehicle data or maintenance alerts.
    - Example: Trigger a function when a vehicle enters a geofenced area.
    - Integration: Set up triggers from Azure Event Grid or HTTP requests.

4.4 Front-end technologies

On front-end we used React framework for building the user interface. React is a JavaScript library for building user interfaces, and it allows us to create reusable UI components. We chose React because it is widely used, has a large community, and provides excellent performance for building dynamic web applications. Beside that we utilize Vite, a modern frontend build tool that provides a faster and leaner development experience for web projects.