SAMFMS

Software Requirement Specification

Team Firewall Five:

Team Member	Student number
Herman Engelbrecht	**u22512374**
Morné van Heerden	**u21482153**
Laird Glanville	**u22541332**
Stefan Jansen van Rensburg	**u22550055**
Nicolaas Johan Jansen van Rensburg	**u22592732**

Contents

1. Introduction

1.1 Purpose

SAMFMS is a modular fleet management platform designed for small to medium-sized businesses. It allows organizations to track, monitor, and manage fleets through a customizable, extensible system using plugin-style modules called **SBlocks**.

Problem Statement:

Small to medium-sized businesses (SMBs) struggle to adopt **Fleet Management Systems (FMS)** due to:

1. **High Costs**: Traditional FMS solutions are expensive and often bundle unnecessary features.
2. **Lack of Flexibility**: Most systems are monolithic, forcing businesses to pay for unused functionalities.
3. **Complexity**: Overwhelming interfaces and steep learning curves deter SMBs with limited technical resources.
4. **Scalability Issues**: Growing businesses need modular systems that adapt to their evolving needs without costly migrations.

Without an affordable, modular, and user-friendly FMS, SMBs face inefficiencies in fleet tracking, maintenance, and route optimization, leading to higher operational costs and reduced competitiveness.

How will we solve this?

SAMFMS addresses these challenges by offering:

1. **Modular Core (MCORE)**: A lightweight base system with essential FMS features.
2. **Plug-and-Play SBlocks**: Businesses pay only for needed features (e.g., GPS tracking, maintenance scheduling).
3. **Customizable Dashboard**: Tailored views for different user roles (e.g., fleet managers, drivers).
4. **Scalability**: Seamless addition of SBlocks as business needs grow.
5. **Cost-Effectiveness**: Affordable entry-level pricing with optional premium modules.

2. User Stories & User Characteristics

2.1. User Characteristics:

User Type	Role	Needs/Pain Points
Fleet Manager	**Oversees fleet operations.**	**Needs real-time tracking, maintenance alerts, and driver analytics.**
Driver	**Operates vehicles.**	**Requires clear route instructions, trip logs, and emergency alerts.**
Admin/IT Staff	**Manages system configuration.**	**Prioritizes easy SBlock integration and security.**

User Story 1: Administrator

As an administrator:

1. I want to add and remove vehicles.
2. I want to add and remove drivers.
3. I want to add and remove functionality.
4. I want to be able to do what a fleet manager does.

User Story 2: Fleet Manager

As a fleet manager:

1. As a fleet manager I want to schedule regular maintenance that could be based on mileage or time as a preventative measure, so that vehicle breakdowns are minimized.
2. I want to assign drivers to available vehicles, so that tasks are carried out efficiently and on schedule and avoid traffic issues.
3. I want to see which vehicles are currently available or in use, so that I can make informed scheduling decisions.
4. I want to monitor the fuel consumption of each vehicle, so that I can identify inefficiencies and reduce fuel costs.
5. I want to receive alerts when a vehicle is due for service, so that I can take timely action and avoid unscheduled downtime.
6. I want to generate weekly and monthly reports on vehicle usage, so that I can analyze performance and optimize fleet operations.
7. I want to monitor how many hours each driver has worked, so that I can comply with labor regulations and prevent overworking.
8. I want to track real-time GPS locations of all vehicles on a map, so that I can monitor positions and respond quickly to delays or emergencies.
9. I want to be notified of any delivery delays or route issues, so that I can take corrective action and inform stakeholders.
10. I want to track the certifications and licenses of each driver, so that I ensure only qualified drivers operate specific vehicles.
11. I want to record any accidents or breakdowns, so that I can investigate causes and improve preventative procedures.

12. I want to view the complete history of each vehicle (maintenance, usage, incidents), so that I can make informed decisions about repairs or replacements.
13. I want to filter vehicles by status (moving, idle, offline), so that I can quickly identify issues.
14. I want to view historical trip data (routes, stops, distance, fuel consumption), so that I can analyze past performance.
15. I want to set up geofence alerts, so that I receive notifications when vehicles enter or leave designated zones.
16. I want to see speed violations highlighted, so that I can address reckless driving.
17. I want to export trip reports (PDF/CSV), so that I can share data with stakeholders.
18. I want to click on a vehicle to see driver details (name, contact info), so that I can communicate quickly.
19. I want to view live traffic data, so that I can reroute vehicles to avoid delays.
20. I want to set custom alerts (e.g., for unauthorized stops), so that I can prevent misuse.
21. I want to see vehicle health status (engine faults, maintenance due), so that I can prevent breakdowns.
22. I want to receive emergency alerts (e.g., accidents), so that I can respond immediately.

User Story 3: Driver

As a driver:

1. I want to view the truck and route assigned to me for the day, so that I know what vehicle to drive and which deliveries to make.
2. I want to check the status and condition of my assigned vehicle, so that I can report any issues before starting the trip.
3. I want to check in when I start and end a trip, so that the fleet manager knows my driving schedule and hours.
4. I want to log any vehicle problems during or after a trip, so that maintenance can be scheduled promptly.
5. I want to receive real-time updates or changes to my assigned route, so that I can avoid delays or reroute as needed.
6. I want to log fuel refills, mileage, and location, so that the company can track expenses and schedule maintenance accurately.
7. I want to submit feedback after completing my trip, so that I can report delays, road conditions, or concerns.

8. I want to upload proof of delivery (e.g., signature or photo), so that there is evidence that the delivery was completed successfully.
9. I want to see my past trips, hours driven, and performance metrics, so that I can monitor and improve my own driving.
10. I want to receive automatic maintenance reminders, so that I know when my vehicle needs service.
11. I want to acknowledge alerts, so that my fleet manager knows I acknowledge them.
12. I want to upload photos of vehicle issues, so that mechanics can diagnose problems faster.
13. I want to take my vehicle to the service center when scheduled or instructed, so that the vehicle remains in safe and optimal condition.
14. I want to see urgent vs. routine alerts, so that I can prioritize repairs.
15. I want to receive notifications for recalls, so that I can take action if needed.
16. I want to contact support directly from the alert, so that I can get help fast.

---

3. Functional Requirements

3.1 Subsystems

1. MCore (Modular Core)
2. User Interface
3. GPS SBlock
4. Trip Planning SBlock
5. Vehicle Maintenance SBlock
6. Potential Future SBlocks

3.2 Requirements

SAMFMS will:

1. Provide role based log in [system: UI, MCore]
2. Provide a way to add or remove functionality [UI, MCORE] (admin)
3. Provide a way to add or remove vehicles in the fleet [UI, MCORE] (admin)
4. Provide a way to add or remove drivers [UI, MCORE] (admin)
5. Provide a way to plan trips and routes for vehicles in the fleet [Trip Planning SBlock] (fleet manager)

6. Provide a way to schedule maintenance [Vehicle Maintenance SBlock] (fleet manager)
7. Provide a way to track the vehicles in the fleet with GPS [GPS SBlock] (fleet manager)
8. Show information about the trip [Future SBlock] (driver)

3.3 Functional Requirements

1 MCORE (Modular Core)

1. **User Authentication & Authorization**
   - 1.1 The system shall support role-based access (Fleet Manager, Driver, Admin).
   - 1.2 Users shall log in via email/password or SSO (Google/Microsoft).
2. **SBlock Management**
   - 2.1 Admins shall install/uninstall SBlocks via a plugin marketplace.
   - 2.2 The system shall validate SBlock compatibility before installation.
3. **Data Interoperability**
   - 3.1 MCORE shall provide a standardized API for SBlocks to share data (e.g., vehicle IDs, driver details).
   - 3.2 All SBlocks shall store data in a central database with unified schemas.
4. **Dashboard Framework**
   - 4.1 Users shall customize dashboard widgets (drag-and-drop).
   - 4.2 The dashboard shall aggregate data from all active SBlocks in real time.
5. **Notification System**
   - 5.1 MCORE shall route alerts from SBlocks to users via email/in-app notifications.
   - 5.2 Users shall configure alert thresholds (e.g., "Notify if speed > 120km/h").

---

2. User Interface

6. **Responsive Design**
   - 6.1 The UI shall adapt to desktop, tablet, and mobile screens (Bootstrap/React).
   - 6.2 Dark/light mode shall be user-selectable.
7. **Account Management**
   - 7.1 Users should be able to update their details whenever needed.
   - 7.2 Admins should be able to grant Admin permissions to other users.

8. **Driver Portal**
    - 8.1 Drivers shall view assigned trips/maintenance tasks in a simplified UI.
    - 8.2 Drivers shall submit emergency alerts with location/GPS coordinates.
9. **Reporting**
    - 9.1 Users shall generate prebuilt reports (PDF/Excel) for fuel usage, idle time, etc.
    - 9.2 Custom report templates shall be savable for reuse.

---

3 GPS SBlock

10. **Real-Time Tracking**
    - 10.1 The system shall display vehicle locations on a map
    - 10.2 Historical routes shall be replayable with speed/stop annotations.
11. **Geofencing**
    - 11.1 Users shall draw geofences on the map and set entry/exit alerts.
    - 11.2 Geofences shall be assignable to specific vehicles/drivers.
12. **Telemetry Data**
    - 12.1 The system shall capture and store speed
    - 12.2 Data shall be accessible via API for third-party integrations.

---

4 Trip Planning SBlock

13. **Route Optimization**
    - 13.1 The system shall calculate fuel/time-optimal routes using Google Maps/OSRM.
    - 13.2 The system shall allow the user to add constraints to a trip
      13.2.1 Trip shall avoid tolls
14. **Multi-Stop Planning**
    - 14.1 Fleet Managers shall drag-and-drop stops to reorder routes.
    - 14.2 The system shall auto-group nearby stops for efficiency.
15. **Driver Instructions**
    - 15.1 Turn-by-turn navigation shall be pushable to drivers' mobile apps.
    - 15.2 Routes shall include POIs (fuel stations, rest areas).

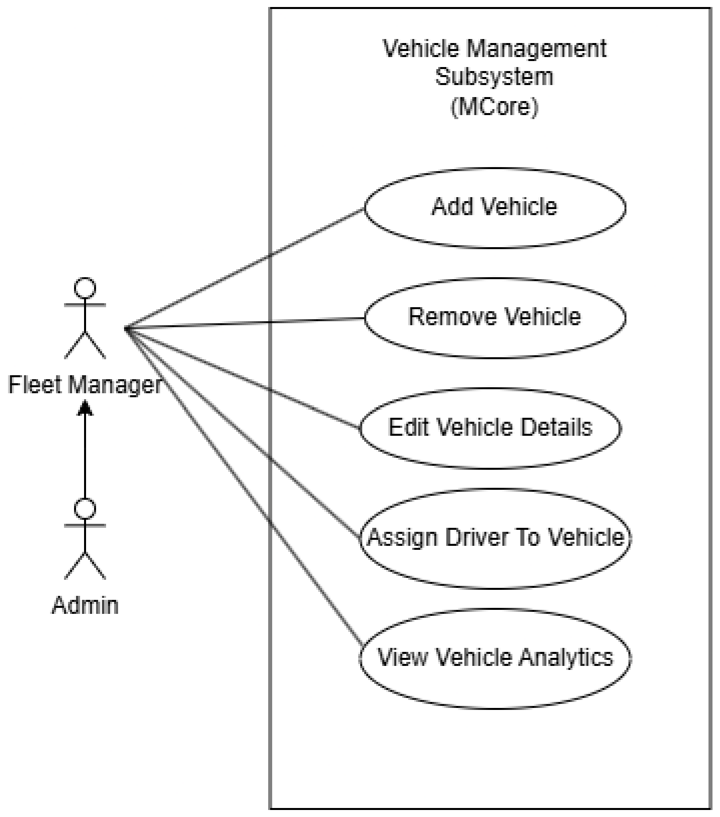
---

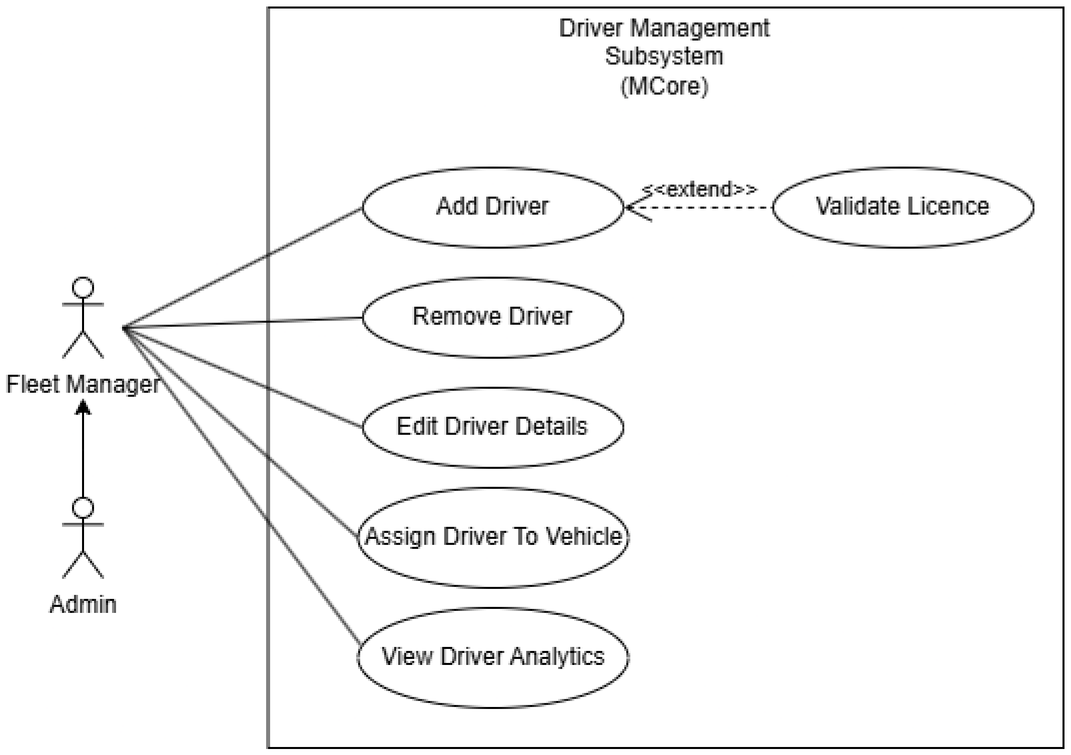
5 Vehicle Maintenance SBlock

16. **Maintenance Scheduling**
    - 16.1 The system shall trigger alerts based on mileage/time/OEM guidelines.
    - 16.2 Fleet managers shall assign service tasks to specific garages/mechanics.
    - 16.3 Drivers will be able to log a service when it is complete.
    - 16.4 Fleet managers will log all vehicle services.
    - 16.5 The system will predict when services for vehicles should occur.
17. **Diagnostics Integration**
    - 17.1 The system shall ingest OBD-II data to predict engine issues (e.g., "Check engine light").
    - 17.2 Mechanics shall annotate service records with photos/notes.
18. **Parts Inventory**
    - 18.1 Users shall track spare parts (tires, filters) with low-stock alerts.
    - 18.2 Barcode scanning shall log part installations.

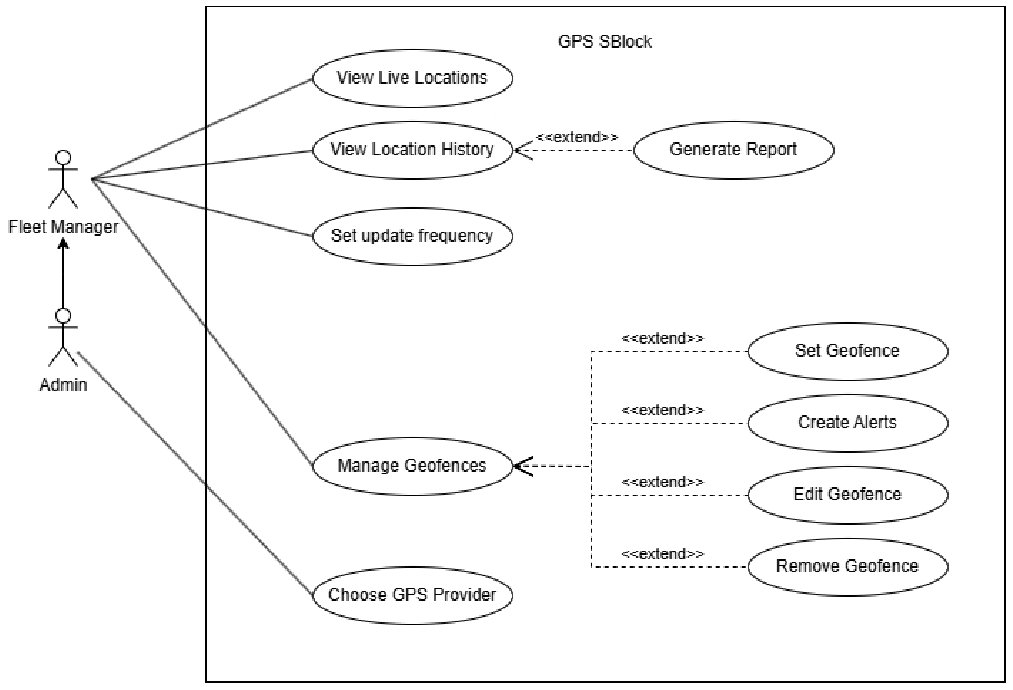
3.4 Use Cases

1. Use Case: Manage Vehicles

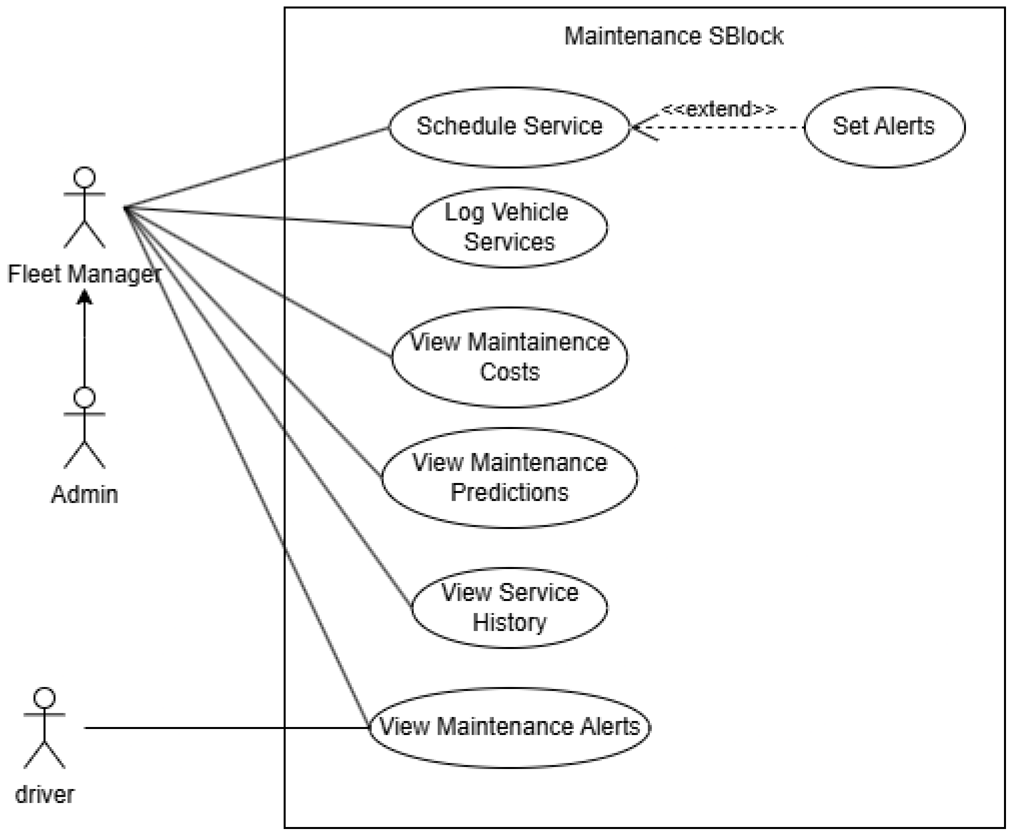


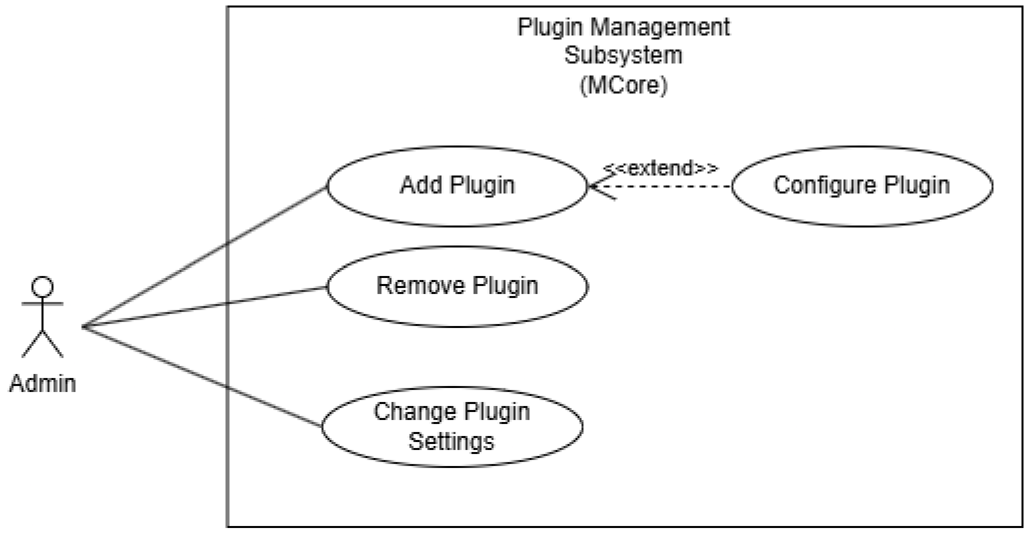


2. Use Case: Track Fleet (GPS SBlock)



3. Use Case: Schedule Maintenance (Maintenance SBlock)





4. Non Functional Requirements

1. Performance

- **1.1 Latency:**
  - GPS tracking data must be displayed with ≤10-second latency.
  - Dashboard widgets shall load within **4 seconds** under normal load.
- **1.2 Scalability:**
  - The system must support **up to 1,000 concurrent users** and **5,000 vehicles** without degradation in performance.
- **1.3 Data Processing:**
  - Route optimization calculations (Trip Planning SBlock) must complete within **5 seconds** for 50+ stops.

2. Reliability & Availability

- **2.1 Fault Tolerance:**
  - If an SBlock fails, MCORE shall continue operating core functionalities (e.g., authentication, basic vehicle tracking).
  - Telemetry data loss during outages must not exceed **1 minute** of data.
- **2.2 Backup & Recovery:**
  - All critical data (vehicle logs, maintenance records) shall be backed up **daily** with a recovery time objective (RTO) of **≤1 hour**.

3. Security

- **3.1 Authentication:**
  - All user sessions shall expire after **30 minutes of inactivity**.
  - Brute-force attacks shall be mitigated via **account lockout** after 5 failed attempts.
- **3.2 Data Protection:**
  - All sensitive data (driver details, GPS history) must be encrypted **at rest (AES-256)** and **in transit (TLS 1.3+)**.
  - Compliance with **GDPR/POPIA** for data privacy and user consent.
- **3.3 API Security:**
  - SBlock APIs shall enforce **OAuth 2.0** for third-party integrations.

4. Usability

- **4.1 Learnability:**
    - A new user shall be able to perform basic tasks (e.g., track a vehicle) within **10 minutes** without training.
- **4.2 Mobile Responsiveness:**
    - Dashboard shall adapt to **screen sizes ≥320px wide** (mobile phones).

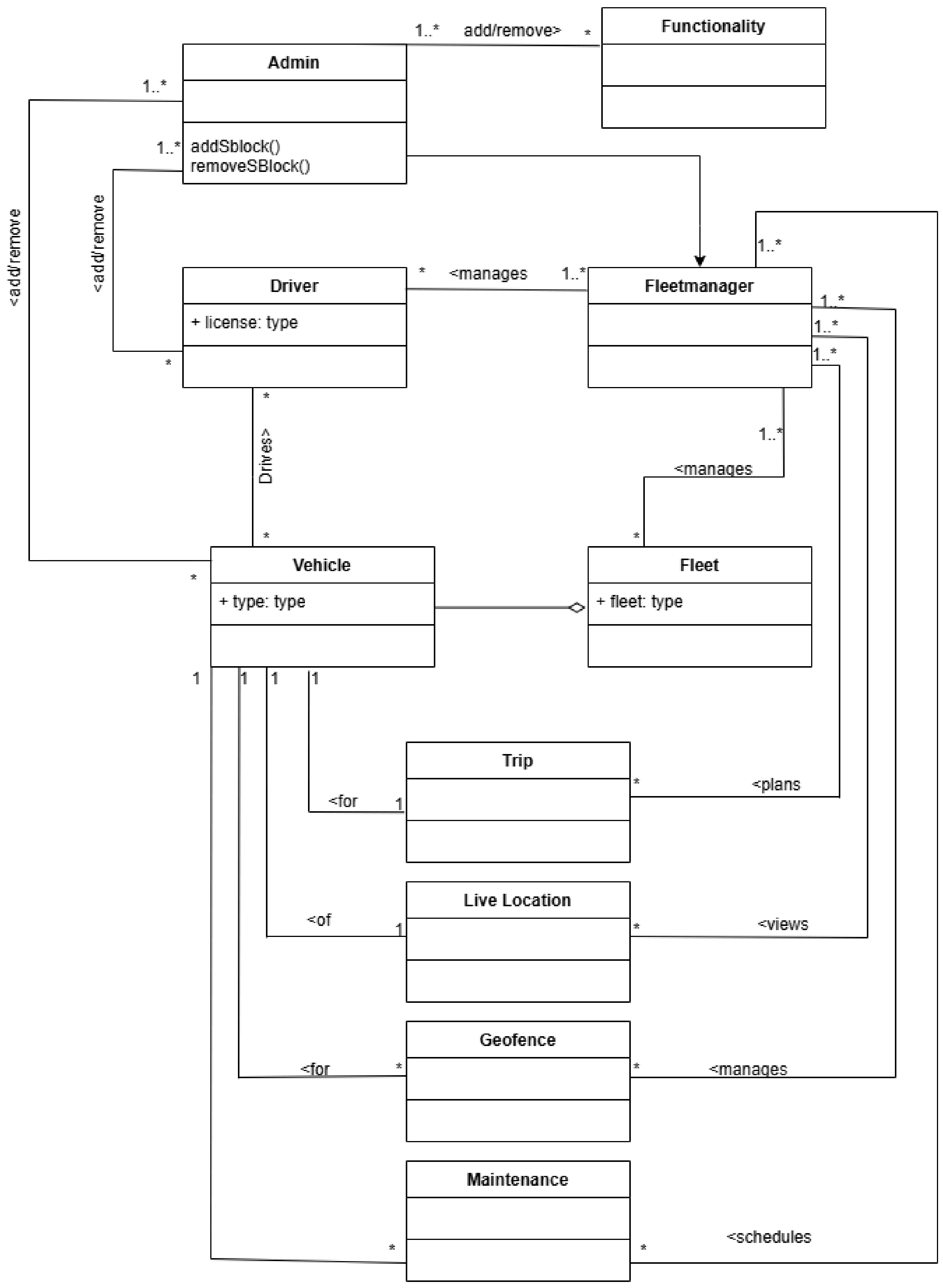
5. Maintainability

- **5.1 Modularity:**
    - SBlocks shall be independently updatable without requiring system downtime.
- **5.2 Logging:**
    - All critical actions (e.g., vehicle assignment, maintenance logs) shall be audited with timestamps and user IDs.

6. Integration

- **6.1 Third-Party APIs:**
    - Map integrations (Google Maps/OSRM) shall handle **≥100 requests/minute** without throttling.
- **6.2 Data Interoperability:**
    - MCORE APIs shall support **JSON/XML** formats for cross-SBlock communication.

Domain Model


Admin
addSblock()
removeSBlock()
Functionality
add/remove>
1..*
*
Driver
+ license: type
Fleetmanager
<manages
Vehicle
+ type: type
Fleet
+ fleet: type
<add/remove
<add/remove
Drives>
<manages
Trip
<for
<plans
Live Location
<of
<views
Geofence
<for
<manages
Maintenance
<schedules
1
*

Service Contracts

Service	What it does	Inputs required	Outputs	How other systems interact
UI	The UI is used by the Admin, Fleet Manager and potentially, drivers	Login details	Fleet statistics	The UI only interacts with the user and the API
MCore	The MCore facilitates communication and authorization between SBlocks, MCore and the UI and also allows the customization of functionality (adding or removing SBlocks)	Current inputs include endpoints for login functionality and a message queue consumer.	Current outputs include endpoints for login functionality, endpoints for storing users and vehicles on its database and a producer for the message queue.	The UI can communicate with the MCore using MCore's API. The SBlocks have to use the message queue to communicate with the MCore
SBlocks (varied)	SBlocks are microservices that provide unique functionality to the system depending on what they are purposed for. The system can have any number of SBlocks installed as plugins.	The input would be messages that it consumes from the message queue as well as the inputs required by whatever service the certain SBlock is providing.	The output consists of only messages produced to the message queue.	The SBlocks would simply subscribe to channels on the message queue that is useful to it. The SBlocks will use the message queue to communicate to each other as well as the MCore. The SBlocks simply produce messages onto the message queue