Title Page

Project On

"SRS On Vehicle Tracking System"

Submitted by

Deepak Kumar Rajbhar

Roll No.25, Reg.No.(12307910)

Sonu Kumar

Roll No.26, Reg.No.(12303430)

Vikas Gangwar

Roll No.27, Reg.No (12302265)

Nikhilesh Mitra

Roll No.28, Reg.No. (12300559)

Submitted to

Dr. Amanpreet Singh

In partial fulfilment for the requirements of the award of the degree of

"Bachelor of Computer Applications"

"School of Computer Application"

<u>Lovely Professional University Phagwara, Punjab.</u>

INDEX

1. Introduction

1.1 Purpose: In the modern era of transportation and logistics, tracking and managing vehicles efficiently has become a critical need for both public and private sectors. The **Vehicle Tracking System (VTS)** is a comprehensive solution designed to monitor the movement of vehicles in real time using GPS (Global Positioning System) technology. This system serves a variety of purposes including ensuring vehicle safety, optimizing routes, improving fuel efficiency, and enabling better fleet management. The system leverages GPS devices installed in vehicles to collect data such as location, speed, and route history, which is then transmitted to a centralized server and displayed on a user-friendly dashboard.

The VTS software is intended for use by fleet managers, business owners, government agencies, and transport companies who need accurate, real-time data on their vehicles' locations and activities. It not only provides live tracking capabilities but also offers features such as geofencing, alert generation, route history analysis, and detailed reporting. The integration of mobile alerts through SMS and email ensures instant notifications for any unauthorized activity or route deviation.

This Software Requirements Specification (SRS) document outlines all aspects of the Vehicle Tracking System, including its purpose, features, interfaces, performance requirements, and design constraints. The aim is to provide a clear and comprehensive reference for stakeholders including developers, testers, clients, and end users. By defining the system's functionality in detail, this SRS ensures smooth and aligned development, resulting in a robust and scalable vehicle tracking solution.

This document helps:

- Developers design and implement the system.
- Testers create test plans.
- Project managers estimate resources.
- Users understand system features

1.2 Scope: The **Vehicle Tracking System (VTS)** is a web and mobile-based solution designed to provide real-time tracking, monitoring, and reporting for individual and fleet vehicles. The system uses GPS-enabled devices installed in vehicles to send live location data to a cloud server. This data is then processed and displayed on an interactive dashboard accessible by administrators, fleet managers, and other authorized users.

The system's primary function is to offer accurate and continuous location tracking of vehicles along with essential metrics such as speed, direction, distance traveled, and idle time. In addition to real-time updates, the system supports features like route history playback, geofencing (virtual boundaries), and automated alerts for violations such as route deviations, speeding, or unauthorized vehicle movement. These features ensure better decision-making and enhance the safety and efficiency of fleet operations.

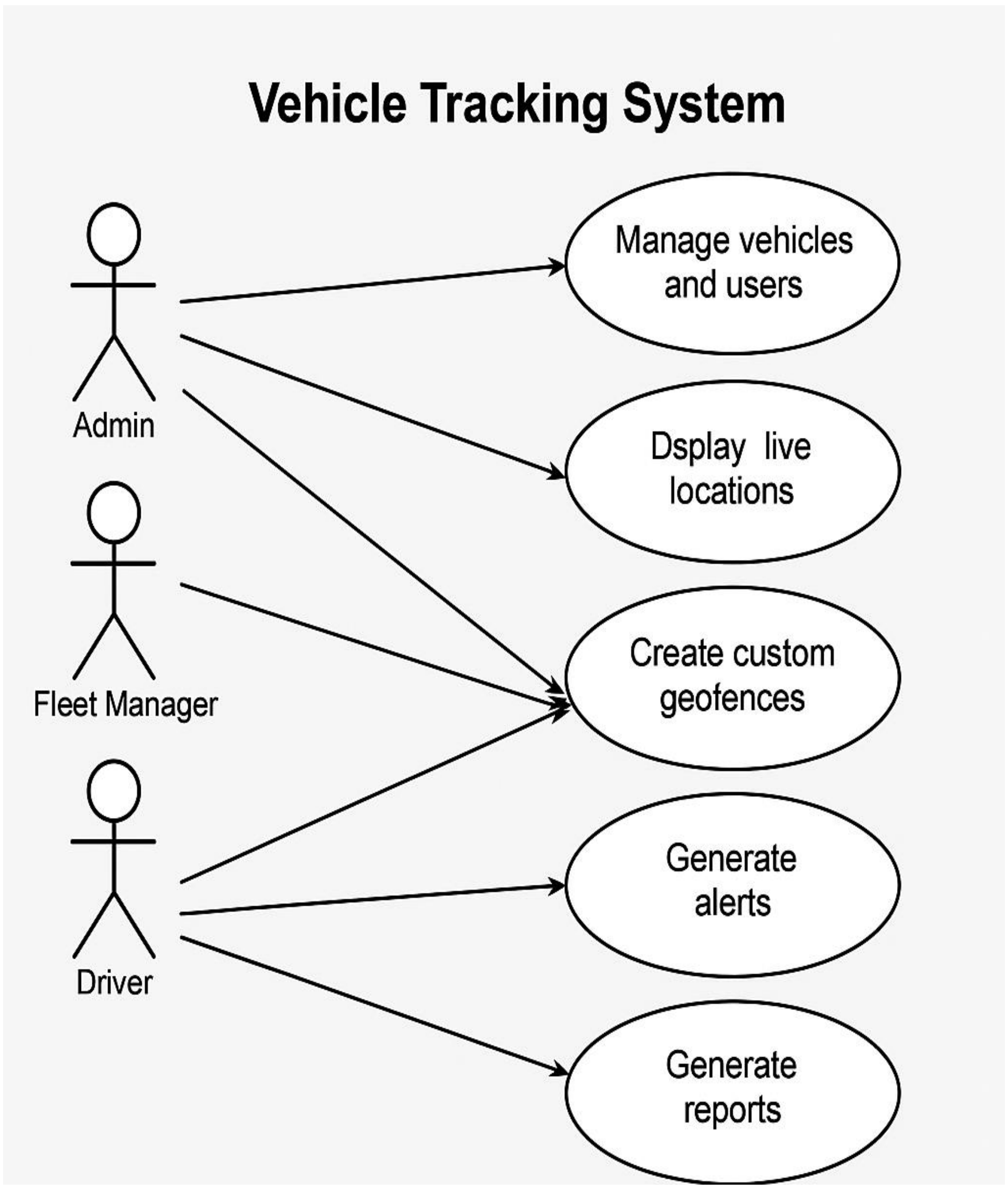
The scope of this project includes developing a secure login-based user interface, role-based access controls, and a responsive dashboard. Backend development will include GPS data processing, report generation, and API integrations for maps and external communication services (SMS, Email). The system will also be scalable and deployable on cloud infrastructure such as AWS or Microsoft Azure.

Outside the scope are the physical installation of GPS hardware and third-party service subscriptions (e.g., SMS gateways). However, the software will be designed to support integration with standard GPS tracking hardware and third-party APIs.

This document captures the system's expected behavior, performance requirements, limitations, and user expectations to guide development and testing activities.

:

- Track real-time vehicle location.
- Create and monitor geofences.
- Review route history.
- Access analytics reports.
- Manage access based on roles.


Vehicle Tracking System
Admin
Fleet Manager
Driver
Manage vehicles and users
Dsplay live locations
Create custom geofences
Generate alerts
Generate reports

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
GPS	Global Positioning System
VTS	Vehicle Tracking System
API	Application Programming Interface
UI	User Interface
Admin	System manager with full access
Geofence	Virtual boundary for tracking alerts
REST API	Communication protocol for systems

1.4 References

- IEEE Std 830-1998
- OWASP Security Guidelines
- ISO/IEC 27001
- Government vehicle tracking policies
- Google Maps API Documentation

1.5 Overview

This document includes:

- System overview
- Functional & non-functional requirements
- System interfaces
- Feature breakdown

- Compliance and maintenance details

- This Software Requirements Specification (SRS) document outlines all necessary information for the successful development and implementation of the Vehicle Tracking System (VTS). It is organized into well-structured sections that provide a comprehensive view of the system’s requirements, design goals, and compliance needs. Below is a brief explanation of each point included in this overview:
-
- • **System Overview:**
- The system overview introduces the Vehicle Tracking System and presents a high-level description of its purpose, functionality, and key objectives. It highlights how the VTS integrates GPS technology, cloud infrastructure, and interactive dashboards to track vehicle activity in real-time. This section provides the foundational context for understanding the remainder of the document by describing what the system is designed to do and who its intended users are, including administrators, fleet managers, and drivers.
-
- • **Functional & Non-Functional Requirements:**
- This part of the document breaks down the core functions the system must perform (functional requirements), such as tracking vehicle location, sending alerts, and generating reports. It also includes non-functional aspects like performance, reliability, security, and usability. These requirements ensure that the system is not only operationally effective but also secure, scalable, and user-friendly.
-

System Interfaces:

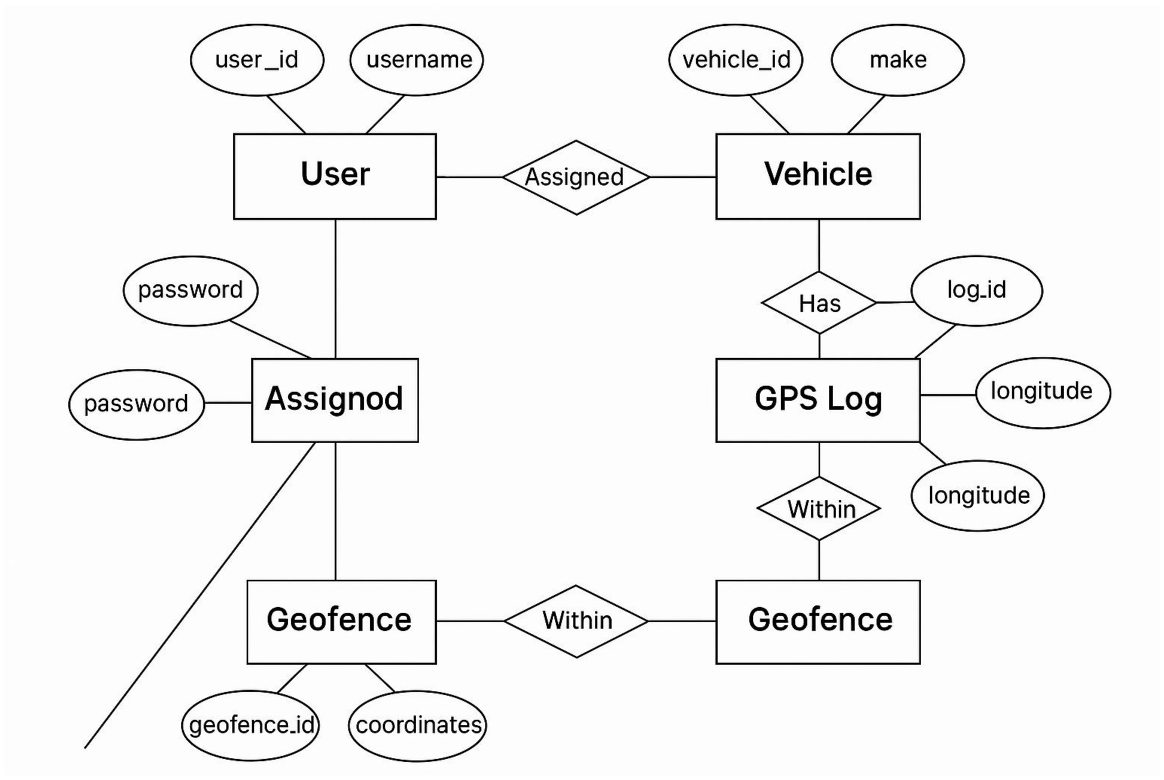
The system interfaces section explains how the VTS interacts with other systems and hardware. This includes communication with GPS devices, APIs (like Google Maps), and external services such as SMS or email for alerts. It ensures that developers understand all integration points needed for seamless operation.

Feature Breakdown

This section lists and explains all major features of the VTS in detail, such as real-time tracking, route playback, geofencing, report generation, and user role management. It outlines the capabilities users can expect and how those features are implemented from a user and system perspective.

2. Overall Description

2.1 Product Perspective



The **Vehicle Tracking System (VTS)** is a comprehensive, scalable, and standalone software application designed to monitor the real-time location and movement of vehicles using GPS technology. It integrates with GPS tracking devices installed in vehicles, which transmit location and status data to a centralized backend system. This data is then processed and displayed to users through an intuitive and interactive interface on both web and mobile platforms.

From a **system perspective**, the VTS serves as a middleware that bridges hardware (GPS trackers, GSM modules) and software (cloud backend, user interface). It does not function in isolation but interacts seamlessly with various subsystems, such as:

- GPS hardware embedded in vehicles
- Communication networks (e.g., mobile internet, GSM, or satellite)
- Mapping services (like Google Maps API)
- Backend servers for data storage and processing
- Frontend user interfaces for real-time monitoring and control

The product is **modular in nature**, allowing customization and expansion based on organizational needs. For instance, additional features like fuel monitoring, temperature tracking (for cold-chain logistics), or driver behavior analytics can be integrated without changing the core functionality.

It supports **multi-user roles** and is designed to handle a high volume of concurrent users and vehicles, making it suitable for various sectors including logistics, public transportation, cab aggregators, and emergency services. Furthermore, the product ensures cross-platform compatibility (Android, iOS, Web) and supports both vertical (more features) and horizontal (more users/vehicles) scalability.

In short, the Vehicle Tracking System provides a flexible foundation for businesses and institutions looking to improve vehicle oversight, reduce costs, and enhance operational efficiency through smart mobility solutions.

2.2 Product Functions

- Live location updates
- Route playback
- Geofence alerts
- Report generation
- Role-based login

  The **Vehicle Tracking System (VTS)** offers a comprehensive suite of functionalities aimed at providing efficient vehicle monitoring and control. These core features work in unison to ensure users have real-time data, historical insights, and control over their fleets. The key product functions include:

- **Live Location Updates**
- The primary function of VTS is to provide **real-time tracking** of vehicles using GPS data. As vehicles move, their updated coordinates are sent to the central server and displayed on a live map interface. This feature helps fleet managers instantly determine the location, direction, and speed of each vehicle. It enhances decision-making by allowing timely rerouting and handling of emergencies.

- **Route Playback**
- Route playback enables users to **revisit and review the travel history** of vehicles over a selected time period. This feature shows the exact path taken, stop durations, speed variations, and timing at different points. It is especially helpful in analyzing driver behavior, validating delivery times, and investigating complaints or route deviations.

- **Geofence Alerts**
- Geofencing allows administrators to **set virtual boundaries** around specific geographical areas. When a vehicle enters or exits these boundaries, the system automatically generates alerts (via email, SMS, or app notifications). This function is crucial for ensuring vehicles stay within assigned zones, preventing misuse, and improving security.

Report Generation

- VTS generates **customizable reports** based on various parameters such as travel distance, idle time, fuel usage (if integrated), and geofence violations. These reports can be exported in formats like PDF or Excel, providing valuable insights for operational optimization and performance evaluation.
- **Role-Based Login**
- To ensure secure access and workflow management, VTS includes **role-based login mechanisms**. Different users—admins, fleet managers, or support staff—have tailored access rights. For instance, an admin can manage users and vehicles, while a fleet manager may only have viewing and reporting privileges.

2.3 User Classes and Characteristics

- **Admin:** Full control, user/vehicle management
- **Fleet Manager:** Monitoring, reporting
- **Driver:** Passive participant

The **Vehicle Tracking System (VTS)** is designed to cater to different user roles, each with its specific responsibilities and access privileges. This role-based approach ensures data security, task efficiency, and an organized workflow. The primary user classes in the system are **Admin**, **Fleet Manager**, and **Driver**, each having distinct characteristics:

Admin

The Admin holds **full system control and privileges**. This role is typically assumed by system owners, IT heads, or senior management in the organization. The Admin is responsible for:

- Adding or removing users (managers, staff)
- Registering and configuring vehicles
- Setting up geofences and system preferences
- Managing system settings such as alert types, data refresh intervals, or user permissions
  They have access to all reports, dashboards, and logs. The Admin's decisions shape how the system functions for all other users.

Fleet Manager

Fleet Managers are responsible for the **day-to-day tracking, monitoring, and performance reporting** of vehicles. They use the system to:

- Monitor live vehicle locations
- Analyze route history
- View fuel usage and vehicle behavior

- Generate and interpret operational reports
  Fleet Managers play a crucial role in ensuring vehicle productivity, timely delivery, and overall route efficiency. They have limited administrative privileges but full access to analytical tools.

Driver

Drivers are considered **passive participants** in the system. Their role does not involve active system interaction, but their vehicles are the source of tracking data. The system may be configured to provide drivers with:

- Location sharing toggles
- Notifications (for overspeeding or route deviation)
- App-based support or SOS buttons
  Drivers help ensure that data sent to the system is valid through responsible vehicle use

2.4 Operating Environment

- Platforms: Android, iOS, Web
- Browsers: Chrome, Firefox
- Server: Cloud (AWS/Azure)
- DB: MySQL/PostgreSQL

The **operating environment** defines the platforms, tools, and technologies where the **Vehicle Tracking System (VTS)** will function efficiently. To ensure seamless tracking, data processing, and user interaction, the system is designed to be compatible across a wide range of modern environments.

Platforms: Android, iOS, Web

The VTS is a cross-platform solution, accessible on **mobile devices (Android and iOS)** and **desktop browsers via a web interface**.

- **Android/iOS apps** offer a mobile-friendly experience for fleet managers and drivers on the move. Users can receive real-time notifications, monitor vehicle status, and respond to alerts instantly.
- The **web version** provides a full-feature dashboard suitable for administrative tasks like analytics, configuration, and report generation.

Browsers: Chrome, Firefox

To ensure maximum accessibility, the system is fully compatible with all major modern browsers, particularly **Google Chrome** and **Mozilla Firefox**. These browsers are chosen due to their performance, security features, and widespread usage. The system will also provide fallback support for other modern browsers to maintain broad compatibility.

Server: Cloud (AWS/Azure)

The backend server infrastructure is hosted on reliable cloud service providers such as **Amazon Web Services (AWS)** or **Microsoft Azure**. Cloud hosting ensures:

- High availability and uptime
- Scalability with increasing vehicle load
- Secure data storage and quick disaster recovery options

Database: MySQL/PostgreSQL

All vehicle, user, and tracking data are stored in a **relational database system** such as **MySQL** or **PostgreSQL**. These databases are known for their robustness, security features, and ability to handle complex queries efficiently.

2.5 Constraints

- GPS updates must occur every 5 seconds
- HTTPS security mandatory

Constraints define the limitations and essential conditions under which the **Vehicle Tracking System (VTS)** must operate. These are critical to ensuring the reliability, performance, and security of the system. The two major constraints for VTS are:

• GPS Updates Must Occur Every 5 Seconds

One of the key performance constraints of the system is the **frequency of GPS updates**, which is mandated to occur every **5 seconds**. This constraint ensures that the system maintains near real-time location accuracy for each vehicle.

- **Rationale:** For fleet management, logistics, and emergency services, even a few seconds of delay in location tracking can lead to operational inefficiencies or missed response times.
- **Implication:** The system architecture must be optimized for rapid data acquisition from GPS modules, quick data processing, and immediate visualization on the user interface.
- **Challenge:** Network instability or signal loss in certain geographic regions may interfere with this frequency. Therefore, fallback mechanisms like buffering GPS data temporarily and syncing it when the connection is restored should be implemented.

• HTTPS Security Mandatory

All communication between the user’s device and the system server must be secured using **HTTPS (HyperText Transfer Protocol Secure)**.

- **Rationale:** Given the sensitivity of vehicle data (location, movement history, user credentials), using HTTPS ensures that data is encrypted and protected from tampering or interception during transmission.
- **Implication:** All API endpoints, login interfaces, and dashboards must support secure SSL/TLS protocols.
- **Compliance:** This constraint helps the system conform to **data privacy regulations** such as **GDPR**, **ISO/IEC 27001**, and local cyber laws.

2.6 Documentation

- User Manual
- Setup Guide
- API Reference

Proper documentation is a critical component of any software project, as it serves as a reference point for users, developers, administrators, and stakeholders. The **Vehicle Tracking System (VTS)** will be accompanied by comprehensive documentation to ensure ease of understanding, installation, usage, and integration. The three major documentation components for this system include:

• User Manual

The **User Manual** is designed to guide end-users—such as Admins, Fleet Managers, and other operational staff—on how to interact with the system effectively.

- It includes step-by-step instructions for tasks like logging in, tracking vehicles, setting up geofences, and generating reports.
- Screenshots, tooltips, and user-friendly language will be used to enhance clarity.
- Troubleshooting tips and FAQs are also part of this manual, aimed at helping users resolve common issues independently.

• Setup Guide

The **Setup Guide** is targeted toward system administrators or IT teams responsible for deploying the system.

- It includes prerequisites such as hardware requirements, supported platforms, and software dependencies.
- Instructions for server setup (e.g., on AWS or Azure), database configuration (e.g., MySQL or PostgreSQL), and frontend/backend deployment are clearly outlined.
- It ensures the system is installed correctly, securely, and optimally configured for performance and scalability.

• API Reference

The **API Reference** is an essential document for developers who want to integrate third-party applications or extend the system’s functionality.

- It contains detailed information on each RESTful API endpoint, including request/response formats, parameters, authentication methods, and error codes.
- Code examples and usage scenarios will help developers interact with the VTS backend programmatically with confidence and clarity.

2.7 Assumptions and Dependencies

- GPS-enabled devices available
- Internet connection required

The successful development, deployment, and functioning of the **Vehicle Tracking System (VTS)** depend on several assumptions and external factors. These assumptions and dependencies outline the conditions and components that must be met or made available for the system to operate efficiently and as intended.

• GPS-Enabled Devices Available

One of the primary assumptions is that all vehicles intended to be tracked are equipped with **GPS-enabled tracking devices**. These devices are responsible for continuously gathering geolocation data and transmitting it to the backend system through a secure channel.

- Without GPS hardware, the core functionality of live location tracking, route playback, and geofence monitoring becomes impossible.
- It is also assumed that the GPS devices are properly installed, functional, and capable of sending updates every five seconds (as mentioned in constraints).
- Regular maintenance and calibration of GPS modules are also considered part of this assumption to ensure accuracy.

• Internet Connection Required

A **stable and reliable internet connection** is essential for real-time communication between the GPS device, the central server, and the client application (web or mobile).

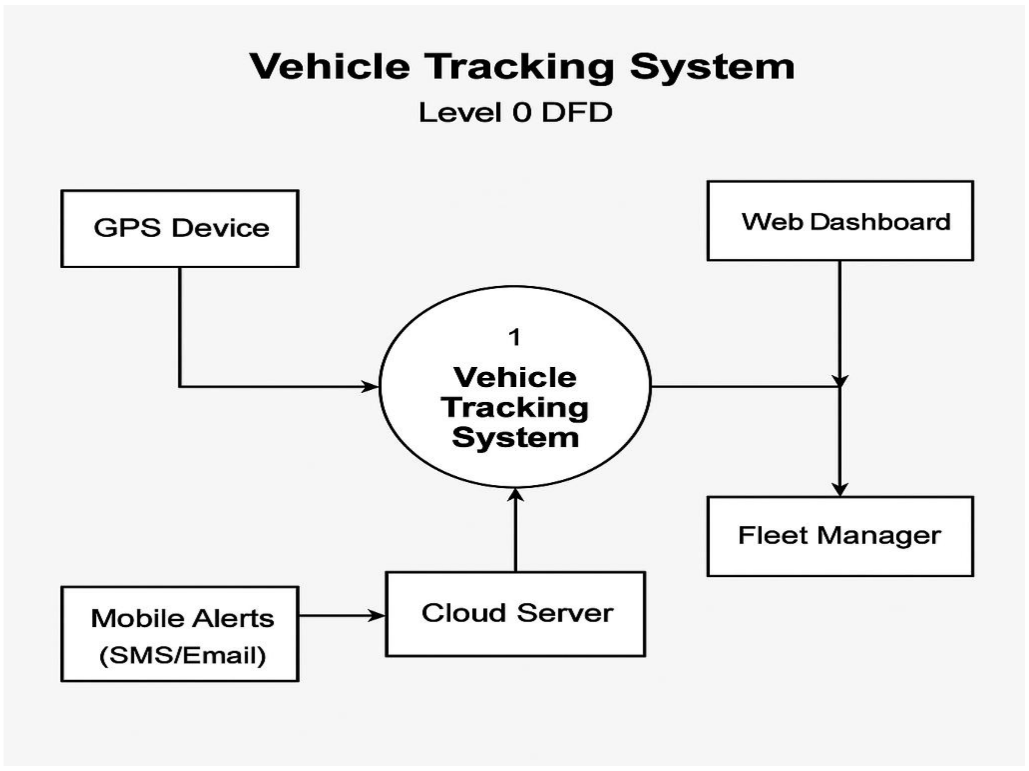
- The system depends on internet connectivity to transmit data packets such as vehicle location, alerts, notifications, and reports.
- All users—whether Admin, Fleet Manager, or Driver—require internet access to interact with the system in real-time.
- The servers hosting the application and databases must also maintain consistent connectivity to the cloud (e.g., AWS or Azure) to ensure uptime and availability.

---

3. Specific Requirements

3.1 Functional Requirements

1. Admin can manage vehicles and users.
2. System displays live locations.
3. Custom geofences can be created.
4. Alerts generated on geofence breach.
5. Multiple report formats supported.
6. Secure login based on user roles.
7. Activity logs maintained.



Functional requirements define what the system **should do** — the core services, tasks, and operations it must perform. Below are the detailed explanations of each listed requirement for the Vehicle Tracking System (VTS):

---

1. Admin can manage vehicles and users

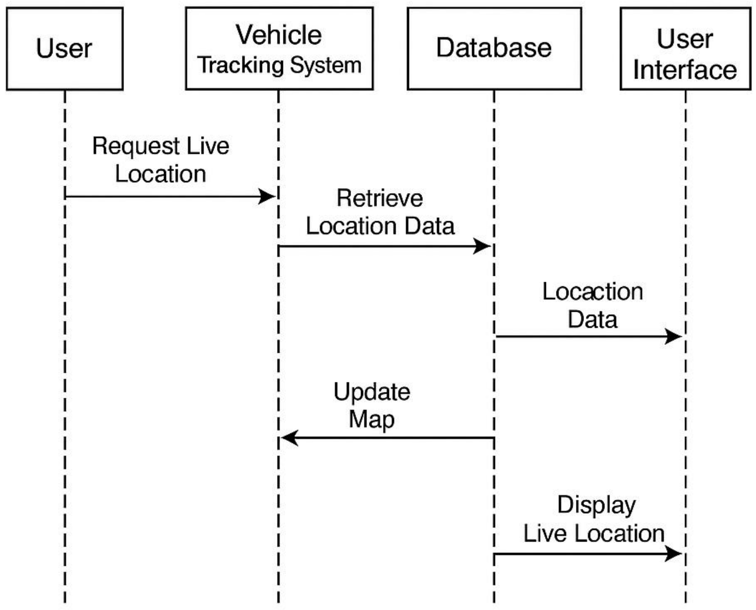
The system must provide an intuitive interface to **administrators**, allowing them to manage users (Fleet Managers, Drivers) and vehicles in the system.

- Admins should be able to add new vehicles, assign them to drivers, edit vehicle details (registration number, model, tracker ID), and remove them if necessary.

- Similarly, user management includes adding new user accounts, setting roles (Admin, Fleet Manager, Driver), updating credentials, resetting passwords, and removing users from the system.
- Admin activities must be logged and auditable for accountability and system monitoring.

---

2. System displays live locations



Sequence Diagram for Live Location Updates

The system should continuously fetch and display the **real-time GPS location** of each vehicle on a digital map interface (such as Google Maps or Mapbox).

- Each vehicle's position should update at a minimum interval of every 5 seconds.
- The map interface should show vehicle identifiers, direction, speed, and last updated timestamp.
- Users must be able to select individual vehicles or view all on a dashboard.
- Color-coded or icon-based visualizations can enhance clarity, such as idle, moving, or stopped status.

---

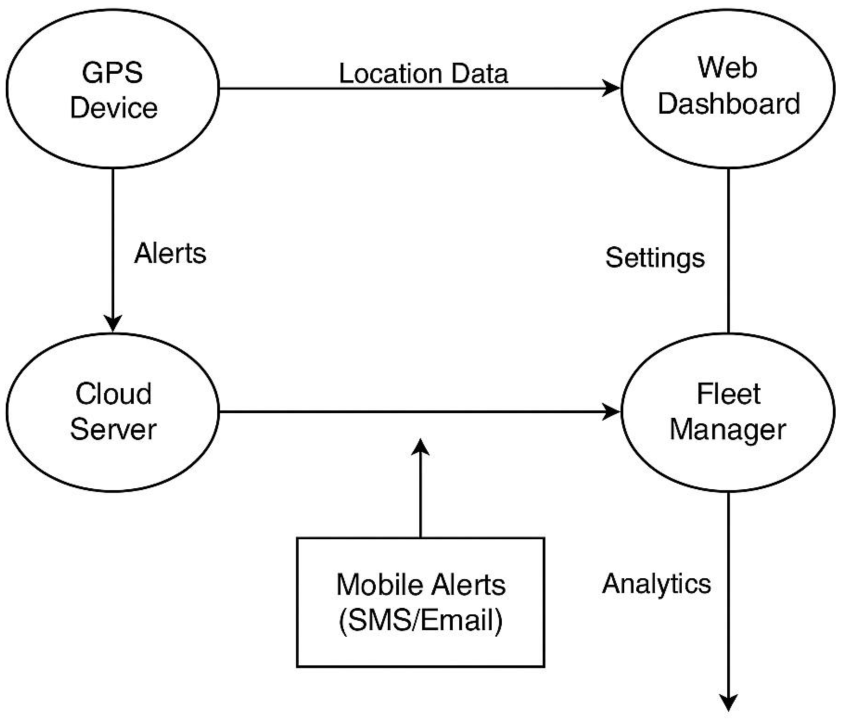
3. Custom geofences can be created

A **geofence** is a virtual boundary around a specific geographic area. The system must allow authorized users (Admin or Fleet Manager) to create, modify, or delete custom geofences.

- Geofences may be drawn on the map in polygon, circle, or rectangular shapes.
- Users should assign geofences to one or more vehicles.
- The system should store geofence metadata: name, location coordinates, assigned vehicles, and active/inactive status.
- Geofences should be editable with immediate effect on monitoring.

---

4. Alerts generated on geofence breach



Data Flow Diagram of Vehicle Tracking System

When a vehicle enters or exits a geofenced area, the system must generate **real-time alerts**.

- Alerts can be sent through multiple channels like in-app notifications, emails, or SMS, based on user settings.
- Alerts should include vehicle ID, time, location, and the nature of the event (entry or exit).

- Each alert must be logged for later analysis and reporting.
- Alert severity can be categorized (e.g., Info, Warning, Critical) depending on geofence purpose.

---

5. Multiple report formats supported

The system should generate **comprehensive reports** on vehicle usage, route history, stop durations, speed violations, and geofence breaches.

- Users can select report type, date range, vehicle(s), and format.
- Supported formats may include PDF, Excel (XLSX), and CSV for easy sharing and analysis.
- Reports should be exportable and downloadable directly from the dashboard.
- Advanced filters, such as driver-based, location-based, or time-based, enhance report flexibility.
- Automated daily/weekly reports sent via email is a desired enhancement.

---

6. Secure login based on user roles

User authentication must be enforced through a **secure login system** using HTTPS, encrypted credentials, and role-based access control.

- Upon login, the system must verify the role (Admin, Fleet Manager, Driver) and redirect to the appropriate dashboard.
- Admins should have full control, Fleet Managers limited to assigned fleets, and Drivers may have view-only access.
- Two-factor authentication (2FA) can be included for enhanced security.
- Role privileges must be enforced in backend logic to prevent unauthorized access, even through API calls.

---

7. Activity logs maintained

For security and accountability, the system must record **activity logs** for every user action.

- Logs must include timestamp, user ID, activity performed (e.g., login, report generated, geofence edited), IP address, and result (success/failure).
- The admin should be able to view and filter logs in the admin panel.
- Logs must be stored securely, with backup policies in place.
- Retention duration should be configurable, with default set to 90 days or more.
- This feature supports internal auditing, incident investigation, and compliance with data protection regulations.

These functional requirements are the foundation for designing and developing the Vehicle Tracking System. They ensure that the system not only fulfills its purpose but also remains **secure, scalable, and user-friendly**.

3.2 Non-Functional Requirements

- **Performance:** 1000+ concurrent users
- **Usability:** Easy-to-use interface
- **Security:** End-to-end encryption
- **Availability:** 99.9% uptime
- **Scalability:** Vertical and horizontal scaling supported

Non-functional requirements (NFRs) describe the **quality attributes** of the system. While functional requirements specify **what** the system does, non-functional requirements focus on **how well** the system performs under various conditions. They are essential to ensure the system is **efficient, secure, maintainable, and user-friendly**.

---

1. Performance: Support for 1000+ Concurrent Users

The system must be capable of handling a minimum of **1,000 simultaneous active users** without compromising on responsiveness or system stability.
This includes users accessing the web dashboard, mobile applications (Android/iOS), and APIs for real-time updates.

- **Response Time**: The system should respond to requests (e.g., fetching live location, generating reports) within **2 seconds** under normal load.
- **Throughput**: It should be able to process **hundreds of GPS location updates per second** from tracked vehicles.
- **Stress Testing**: During peak loads, the system must maintain core functionalities with no critical service outages.
- **Load Balancing**: Use of a load balancer is recommended to distribute traffic evenly across servers, preventing bottlenecks.
- **Caching**: Use of intelligent caching mechanisms (e.g., Redis, Memcached) for frequently accessed data like map tiles or user preferences.

---

2. Usability: Easy-to-Use Interface

The Vehicle Tracking System should prioritize a **user-centered design** to ensure ease of use across all user classes — Admins, Fleet Managers, and Drivers.

- **Intuitive Navigation**: Clear menus, logical workflows, and minimal learning curve for new users.
- **Responsive Design**: The UI should be optimized for various screen sizes and devices, including smartphones, tablets, and desktops.
- **Accessibility**: Interfaces should comply with WCAG (Web Content Accessibility Guidelines) to support users with disabilities.
- **Help & Support**: On-screen tooltips, help sections, and guided tutorials should be included.
- **Feedback Mechanism**: The interface should provide users with clear success/failure messages, loading indicators, and real-time validation for form inputs.

A well-designed UI reduces errors, increases productivity, and improves overall satisfaction.

---

3. Security: End-to-End Encryption

Security is paramount for a system that tracks vehicle locations and manages sensitive data (user details, driving routes, login credentials). The system must follow industry-standard security practices:

- **Data Transmission**: All data (especially GPS updates and login credentials) must be encrypted during transmission using **TLS/SSL** (HTTPS protocol).
- **Data Storage**: Sensitive information such as passwords must be **hashed and salted** before being stored in the database.
- **Authentication**: Implement **role-based access control (RBAC)** and **multi-factor authentication (MFA)** for critical operations.
- **Session Management**: Sessions must have timeouts and token expiration to prevent misuse.
- **Vulnerability Protection**: The system should be tested against common threats like **SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF)**.
- **Audit Trails**: Every login, logout, and significant action should be logged and auditable for internal investigations or legal compliance.

Regular **security audits** and **penetration testing** must be conducted to proactively identify and fix vulnerabilities.

---

4. Availability: 99.9% Uptime

High availability is crucial for a real-time vehicle tracking system. The system must be operational **99.9% of the time**, which equates to a maximum of 8.76 hours of downtime per year.

- **Cloud Hosting**: Deployed on reliable cloud infrastructure providers like AWS or Azure with multi-zone redundancy.
- **Failover Mechanisms**: In case of server failure, traffic should be redirected automatically to backup servers.
- **Health Monitoring**: System health must be continuously monitored using tools like **Prometheus**, **Grafana**, or **New Relic** to detect issues early.
- **Database Replication**: Databases should support replication and backups to ensure recovery from data corruption or outages.
- **Maintenance Windows**: Scheduled downtimes should be clearly communicated in advance to users and should occur during low-traffic hours.

The goal is to make the system **as available and reliable as possible** for continuous tracking and reporting.

---

5. Scalability: Vertical and Horizontal Scaling Supported

The Vehicle Tracking System must be designed with scalability in mind to handle future growth in terms of users, vehicles, and data.

- **Vertical Scaling**: The architecture should allow upgrading individual servers with more CPU, RAM, or storage to meet higher demand.
- **Horizontal Scaling**: Support for adding more servers to distribute load and achieve high availability. Microservices architecture or container-based deployment (e.g., Docker + Kubernetes) can help achieve this.
- **Database Scaling**: Use of **read-replica databases** for read-heavy operations like displaying reports and vehicle locations.
- **Modular Architecture**: Each module (tracking, reporting, alerting, user management) should be independently scalable.
- **Auto-scaling**: Integration with cloud provider services to automatically scale resources up or down based on load (e.g., AWS Auto Scaling Groups).

This ensures the system remains performant as the user base or number of tracked vehicles increases significantly over time.

By meeting these non-functional requirements, the Vehicle Tracking System will not only perform its intended operations efficiently but also maintain a high level of user trust, reliability, and adaptability.

4. External Interface Requirements

4.1 User Interfaces

- Login
- Real-time dashboard
- Report section
- Alerts & notifications

In the **External Interface Requirements** section of the Software Requirements Specification (SRS) for a vehicle tracking system, the **User Interfaces** outline the different screens or sections the user will interact with. Here's a description for each component:

1. Login

- **Purpose**: The login interface provides a secure entry point for authorized users to access the vehicle tracking system.
- **Functionality**:

The user will be required to enter a **username** and **password**.

There may be options for **password recovery** or **two-factor authentication** for enhanced security.

After successful authentication, the user will be directed to the main dashboard.

2. Real-time Dashboard

- **Purpose**: This interface provides the user with a live overview of the system's status, including the location of vehicles, vehicle health, speed, and other important metrics.
- **Functionality**:

Display a **map** with real-time tracking of the vehicles, showing their **current locations** and **routes**.

Present key information like **speed**, **fuel levels**, **battery health**, and **engine status**.

Provide quick access to specific vehicle details by clicking on them.

Interactive elements like zooming in/out of the map or selecting a specific vehicle to view detailed information.

3. Report Section

- **Purpose**: This section allows users to generate and view detailed reports about the vehicles' performance, history, or other relevant data.
- **Functionality**:

Users can select from predefined report types (e.g., **travel history**, **maintenance schedules**, **fuel consumption**).

The system should allow users to filter reports by **date range**, **vehicle type**, or specific **events**.

Reports can be exported in various formats, such as **PDF**, **Excel**, or **CSV**.

A summary of important metrics or insights should be presented along with the detailed data.

4. Alerts & Notifications

- **Purpose**: To notify users about critical system events or vehicle status changes (e.g., maintenance alerts, speed violations, or vehicle breakdowns).
- **Functionality**:

Alerts will appear on the **dashboard** and can be categorized (e.g., **maintenance alerts**, **geo-fencing violations**, **speeding alerts**).

Users should be able to configure which alerts they receive and how (e.g., via **email**, **SMS**, or **push notifications**).

Alerts should include details such as **time**, **location**, **vehicle ID**, and a brief description of the event.

A history of past alerts should be accessible for review and tracking purposes.

Each of these interfaces should be user-friendly, with intuitive navigation and clear visual cues to enhance the user experience.

4.2 Hardware Interfaces

- GPS trackers
- GSM module-enabled devices

In this section, the hardware interfaces describe how the vehicle tracking system interacts with the physical hardware components that provide data or functionality. Here's an explanation of each component:

1. GPS Trackers

- **Purpose**: The GPS trackers are used to determine and transmit the real-time location of the vehicles in the system.
- **Functionality**:

GPS trackers should be installed in each vehicle to track its **current geographic location** using **satellite signals**.

These trackers transmit location data (latitude, longitude, speed, and time) to the central server via the network.

The data should be sent periodically or based on predefined intervals (e.g., every minute or every time the vehicle changes location by a significant distance).

GPS trackers should also report other parameters such as **vehicle speed**, **heading**, and **altitude** to give a full view of vehicle movement.

The system must handle potential errors in GPS data (e.g., signal loss or interference) and should be capable of **recovering** or **retrying** data transmission.

GPS trackers must be compatible with the system's communication protocols (such as **serial communication** or **TCP/IP**).

2. GSM Module-enabled Devices

- **Purpose**: GSM (Global System for Mobile Communications) modules enable the vehicle tracking system to send and receive data through **cellular networks**, especially in remote areas where internet connectivity may be unreliable.
- **Functionality**:GSM modules in the system will be used for **communication** between the vehicle trackers and the central server, especially for sending alerts, location updates, and emergency messages.

The GSM module enables the system to send **SMS notifications** to the user, such as alerts for **speed violations**, **geofence breaches**, or **maintenance reminders**.

It also supports receiving **commands** remotely (e.g., enabling or disabling the vehicle's ignition, or locking/unlocking the vehicle in case of theft).

The system should ensure that the GSM module is capable of operating within the required **network frequencies** and **signal strength** to ensure uninterrupted communication.

The GSM module must be compatible with **SIM cards** for network access, and it should support **SMS**, **voice communication**, and **data transmission**.

Both of these hardware components must be robust, reliable, and designed to function seamlessly with the software interface to provide real-time tracking, alerts, and vehicle management capabilities. The interaction between the GPS trackers, GSM modules, and the server is critical to the overall performance and effectiveness of the vehicle tracking system.

4.3 Software Interfaces

- Google Maps API
- GPS tracking APIs
- RESTful Web Services

In the **Software Interfaces** section of the Software Requirements Specification (SRS), we describe how the vehicle tracking system will interact with external software components or services. Here's a description of each software interface listed:

1. Google Maps API

- **Purpose**: The Google Maps API will be used to integrate maps and geolocation services into the vehicle tracking system. It will display real-time locations, routes, and geographic information on the user interface.
- **Functionality**:

The system will use Google Maps to **render interactive maps** where vehicle locations, routes, and other relevant data (e.g., traffic conditions, landmarks) will be visualized.

The Google Maps API will help plot **real-time vehicle movements** on the map based on the location data received from the GPS trackers.

It will provide **geocoding** services to convert addresses into latitude and longitude coordinates and vice versa.

The API will support features like **street view**, **route calculation**, **distance and time estimation**, and **geofencing** to help users define virtual boundaries for vehicles and trigger alerts when vehicles enter or exit these areas.

Map markers will be used to indicate the current position of each vehicle, with the option to click on the marker to view more detailed information about the vehicle.

2. GPS Tracking APIs

- **Purpose**: These APIs will handle the communication between the vehicle tracking system and the GPS devices installed in the vehicles, enabling the transmission of location data and other vehicle parameters.
- **Functionality**:

The GPS Tracking APIs will be responsible for processing and receiving location data (latitude, longitude, speed, time, etc.) sent by the **GPS trackers** installed in vehicles.

It will provide methods to query and track the **real-time position** of each vehicle, ensuring accurate and timely updates.

The APIs will allow for the **integration** of vehicle data such as **engine status**, **fuel levels**, and **route history** into the tracking platform.

GPS Tracking APIs will support **data transmission protocols** (such as HTTP, MQTT, or proprietary protocols), ensuring the smooth transmission of data to and from the server.

It will also handle **error management** for situations where GPS signals are weak or unavailable, possibly retrying the transmission or using the last known position.

3. RESTful Web Services

Purpose: RESTful web services will enable communication between the vehicle tracking system's front-end and back-end components, allowing for efficient and scalable data transfer.

Functionality:

RESTful APIs will expose endpoints to interact with various features of the tracking system, including vehicle data, user information, and alerts.

These APIs will follow the **REST architecture**, ensuring stateless interactions, scalability, and ease of integration.

Examples of endpoints include:

GET /vehicles: To retrieve a list of all vehicles and their status.

POST /alerts: To create or trigger new alerts based on vehicle activity (e.g., speed violations, geo-fencing violations).

GET /vehicle/{id}/location: To retrieve the current location of a specific vehicle.

RESTful APIs will allow the system to integrate with external systems or mobile apps, providing **real-time updates** via **JSON** or **XML** responses.

Security measures such as **authentication** (e.g., OAuth 2.0) and **data validation** will be implemented to ensure that only authorized users and systems can access or manipulate the data.

These software interfaces will play a crucial role in ensuring smooth interaction between different components of the vehicle tracking system, including mapping, location data retrieval, and system integration. By utilizing Google Maps, GPS tracking APIs, and RESTful web services, the system can provide real-time, accurate tracking, reporting, and user interaction.

4.4 Communication Interfaces

- Protocols: HTTPS, MQTT
- Ports: 443

The **Communication Interfaces** section defines the communication methods and protocols used by the vehicle tracking system to ensure data transfer between the system components and external services. Here's an explanation for each item listed:

1. Protocols: HTTPS, MQTT

- **HTTPS (HyperText Transfer Protocol Secure)**

Purpose: HTTPS is used for secure communication over the internet, ensuring the confidentiality and integrity of the data being transmitted between the client (user interface) and the server.

Functionality:

All web-based communication, such as sending requests to the dashboard or retrieving vehicle location data, will be done over HTTPS.

HTTPS ensures that **data encryption** is in place, making it difficult for unauthorized parties to intercept or tamper with sensitive data (e.g., login credentials, vehicle data).

It uses **SSL/TLS encryption** to create a secure channel between the user interface (e.g., web browser, mobile app) and the backend system, protecting data from man-in-the-middle attacks.

Port 443 is typically used for HTTPS communication, ensuring secure data transfer on the web.

MQTT (Message Queuing Telemetry Transport)

Purpose: MQTT is a lightweight, publish/subscribe messaging protocol designed for efficient communication in environments with limited bandwidth or high-latency networks, making it ideal for IoT (Internet of Things) devices like GPS trackers in vehicles.

Functionality:

The system will use MQTT for real-time data transmission between the GPS trackers in vehicles and the central server.

The MQTT protocol will handle the **publish** and **subscribe** model where the vehicle's GPS tracker **publishes** its location data and other relevant parameters (e.g., speed, fuel level) to a topic on the broker.

The server or mobile apps can **subscribe** to the relevant topics to receive real-time updates.

MQTT is optimized for low bandwidth, low power consumption, and high-frequency messaging, making it ideal for transmitting data from GPS trackers even in remote areas with limited connectivity.

2. Ports: 443

Purpose: Port 443 is the standard port used for secure HTTPS communication over the internet.

Functionality:

All secure web traffic, including API requests, user login, and real-time data updates, will be routed through **Port 443** to ensure encrypted communication.

Port 443 is widely recognized as the default for HTTPS traffic, ensuring compatibility with firewalls and network security systems that allow secure connections.

This port will be used to handle both the communication with the **user interface** (for accessing dashboards, reports, alerts, etc.) and the **back-end services** (for exchanging vehicle location and status updates).

Together, the combination of **HTTPS** for secure web traffic and **MQTT** for lightweight, real-time messaging will enable efficient and secure communication between all system components, from the GPS trackers in vehicles to the server and user interfaces. The use of **Port 443** ensures that these communications are secured and follow standard protocols for web traffic.

5. System Features

5.1 Real-Time Tracking

- Inputs: GPS data
- Outputs: Live map with markers

The **Real-Time Tracking** feature is a core functionality of the vehicle tracking system, allowing users to monitor the live location and movement of vehicles on a map.

- **Inputs: GPS Data**

The primary input for real-time tracking comes from the **GPS trackers** installed in each vehicle. These trackers provide the following data:

Latitude and **Longitude**: Geographical coordinates representing the vehicle's current position.

Speed: The current speed of the vehicle.

Heading: The direction the vehicle is moving in.

Timestamp: The exact time the location data was recorded.

Altitude (if applicable): The height above sea level of the vehicle.

Vehicle ID: The unique identifier for each vehicle to differentiate them in the system.

This GPS data is periodically transmitted from the vehicle’s GPS tracker to the server, typically via **MQTT** or **HTTPS**, depending on the protocol used.

Outputs: Live Map with Markers

The system will display the received GPS data on a **live map**, which could be powered by **Google Maps API** or another mapping service.

Live map: The map will display the real-time geographical positions of all vehicles in the system, along with the relevant details such as the route and location updates.

Markers: The vehicles will be represented as **markers** on the map. Each marker will show the vehicle’s current position, and users can click on these markers to view more details such as:

Vehicle ID or name

Speed

Heading or direction

Timestamp of the last update

Any other relevant vehicle data (fuel level, health status, etc.)

The map will also allow users to zoom in or out to see the vehicles in more detail or in broader geographic areas. Users will be able to **track multiple vehicles** simultaneously on the map, with **different colored markers** or **icons** for visual distinction.

In case of **location updates**, the markers on the map will move accordingly, updating their position to reflect the vehicle's current status. The data will be updated in real-time (or based on predefined intervals) for continuous tracking.

This feature enables users to monitor vehicle movements as they happen, providing critical information for decision-making in logistics, fleet management, or personal vehicle monitoring.

5.2 Route History

- Inputs: Date, vehicle ID
- Outputs: Route displayed on map

The **Route History** feature allows users to view the past travel routes of a specific vehicle over a chosen period.

Inputs:

Date: The user selects a date or date range to specify when the route data should be retrieved.

Vehicle ID: The user provides the unique ID of the vehicle they wish to track.

Outputs:

A **route** is displayed on the map, showing the path the vehicle traveled during the specified time frame.

The route will be represented as a **line** or series of markers, allowing users to visually trace the vehicle’s movement.

5.3 Alerts and Notifications

- Conditions: Speed, geofence
- Outputs: Email/SMS/app alert

The **Alerts and Notifications** feature notifies users when specific conditions are met, providing real-time updates about vehicle behavior.

Conditions:

Speed: Alerts are triggered if a vehicle exceeds a predefined speed limit.

Geofence: Alerts are triggered if a vehicle enters or exits a designated geofenced area (virtual boundary around a location).

Outputs:**Email/SMS**: Users receive an email or SMS notification informing them about the speed violation or geofence breach.

App Alert: Users will also receive instant push notifications via the app for immediate attention.This feature ensures users are promptly informed of any unusual or predefined vehicle behaviors.

5.4 Report Generation

- Types: Usage, idle time, distance
- Formats: PDF, Excel

The **Report Generation** feature allows users to generate various types of reports related to vehicle usage and performance.

Usage: Reports detailing the total operational time, routes traveled, and the vehicle's active periods.

Idle Time: Reports showing the periods when the vehicle was not in motion, helping to monitor inefficiencies.

Distance: Reports tracking the total distance traveled by the vehicle over a specified period.

Reports can be generated in **PDF** format for easy sharing and printing.

Reports can also be exported to **Excel** format for further analysis or record-keeping.

This feature aids in analyzing vehicle performance, improving fleet management, and identifying areas for optimization.

5.5 Role Management

- Add/edit/delete users and assign roles

The **Role Management** feature enables administrators to control user access and permissions within the system.

Add/Edit/Delete Users: Administrators can manage user accounts by adding new users, editing existing user details, or deleting inactive accounts.

Assign Roles: Users can be assigned specific roles, such as **Admin**, **Manager**, or **Viewer**, each with varying levels of access to the system's features and data. For example:

Admin: Full access to all system features, including user management and configuration.

Manager: Limited access to specific vehicle tracking and reporting features.

Viewer: Access only to view data without modification rights.

5.5 Role Management

- Add/edit/delete users and assign roles

The Role Management feature allows administrators to add, edit, or delete user accounts. Administrators can also assign roles (e.g., Admin, Manager, Viewer) to control access levels and permissions. This ensures that users only have access to the features and data relevant to their responsibilities.

5.6 Vehicle Dashboard

- Metrics: Speed, fuel level, location

The **Vehicle Dashboard** provides real-time data on key vehicle metrics for efficient monitoring.

Metrics:

Speed: Displays the current speed of the vehicle.

Fuel Level: Shows the current fuel level to monitor vehicle consumption.

Location: Displays the live location of the vehicle on a map.

This dashboard gives users a quick overview of vehicle status and performance, aiding in fleet management and decision-making.

5.7 Settings and Preferences

Theme selection, notification toggle

The **Settings and Preferences** feature allows users to customize their experience within the system.

Theme Selection: Users can choose between different visual themes (e.g., light or dark mode) for a personalized interface appearance.

Notification Toggle: Users can enable or disable notifications, including alerts for speed, geofence breaches, or vehicle status updates.

This feature enhances user comfort and ensures the system aligns with individual preferences.

6. Other Requirements

6.1 Backup and Recovery

- Daily backups
- Monthly recovery drills

- **Daily Backups**: The system will perform daily backups to ensure data integrity and availability in case of unexpected failures.

- **Monthly Recovery Drills**: Monthly drills will be conducted to test the effectiveness of recovery processes and ensure data can be restored quickly in case of a disaster.

6.2 Maintenance

- Quarterly updates
- Live performance dashboards

- **Quarterly Updates**: The system will receive quarterly updates to improve functionality, security, and performance, ensuring it remains up-to-date with evolving technologies.

- **Live Performance Dashboards**: Maintenance will include monitoring system performance through live dashboards, ensuring continuous system health and optimal user experience.

6.3 Legal Compliance

- GDPR and local IT laws
- Encryption for personal data

- **GDPR and Local IT Laws**: The system will comply with **GDPR** and relevant local IT laws to ensure data privacy and protection.

- **Encryption for Personal Data**: All personal data will be encrypted during storage and transmission, ensuring that sensitive information remains secure and protected from unauthorized access.

6.4 Risk Management

Risk	Mitigation
Server failure	Use of backups and cloud redundancy
GPS loss	Notify users; log incident
Security breach	Regular penetration testing

7. Appendices

A. Glossary

- **Geofence**: Digital boundary
- **Idle Time**: Inactivity of vehicle

Geofence: A **virtual or digital boundary** set around a real-world geographic area. When a vehicle enters or exits this area, the system triggers alerts or actions.

Idle Time: The **duration when a vehicle is stationary** with the engine running or not in use, indicating non-productive time that can affect fuel efficiency and overall performance.

B. Tools and Technologies

- Languages: Python, JavaScript
- Frameworks: React, Node.js
- Database: MySQL
- Cloud: AWS

B. Tools and Technologies

- **Languages**:
  - **Python** – Used for backend processing, data handling, and integration with APIs.
  - **JavaScript** – Powers interactive front-end functionalities and enhances user experience.
- **Frameworks**:
  - **React** – A front-end JavaScript library for building dynamic user interfaces.
  - **Node.js** – A backend runtime environment used for handling server-side logic and real-time data.
- **Database**:
  - **MySQL** – A relational database management system used to store and manage structured vehicle tracking data.
- **Cloud**:
  - **AWS (Amazon Web Services)** – Used for hosting, storage, backups, and scalable deployment of the application.

.