TRIP INSPECTION SYSTEM

Dawson Road Maintenance

Brian Eshpeter, Daniel Fernandez

Mohammad Abbas Yunus, Pui Hei Wong, Syed Mohammed Sadiq Rizvi, Faiyaz Sattar

12/06/2024

Table of Contents

Analysis

Ensuring compliance with vehicle inspection regulations is a critical responsibility for Dawson Road Maintenance (DRM). However, missed pre-trip inspections for heavy trucks can lead to regulatory violations and operational risks, highlighting the need for an efficient notification system. To address this challenge, we have developed an intelligent, automated solution leveraging DRM's existing SkyHawk Vehicle Telematics system. This system uses geofences around DRM yards to track truck movements and integrates with the trip inspection system via APIs. It verifies the completion of mandatory inspections and sends real-time email notifications to supervisors if inspections are missed.

To enhance transparency and usability, we developed a user-friendly frontend interface that displays all missed inspections and truck movements in real-time. The system's dashboard allows supervisors to view key information such as inspection status, truck details, and geofence violations in an intuitive and organized format. With this web-based interface, managers can access critical insights at a glance, enabling immediate action to ensure compliance. By streamlining this process and providing actionable visibility, our solution not only upholds regulatory standards but also fosters a safer and more accountable fleet management system.

Problem Description

Efficient and compliant vehicle operations are crucial for ensuring road safety and adhering to legal standards. In this context, pre-trip inspections of heavy trucks are not only a legal requirement but also a critical step in maintaining operational reliability. However, instances of missed trip inspections can lead to safety risks and legal repercussions, particularly during random checks by Commercial Vehicle Safety Enforcement (CVSE) officers. To address this issue, the Dawson Road Maintenance (DRM) division of The Dawson Group has identified the need for an automated notification system.

This project, "Trip Inspection System," leverages SkyHawk vehicle telematics to detect when a heavy truck exits the yard and cross-references this event with the company's trip inspection records through an API. If a valid and current inspection is not found, an immediate notification is sent to the respective supervisor, ensuring timely corrective actions. By implementing this system, DRM aims to enhance compliance, reduce risks, and streamline operations, contributing to safer roadways and improved efficiency.

Requirements

The Trip Inspection System is engineered to enhance compliance with pre-trip inspection regulations through the seamless integration of advanced geofence monitoring, inspection validation, and notification mechanisms. These requirements serve as the foundation for the system's development, ensuring that it meets the operational needs of Dawson Road Maintenance while maintaining scalability, reliability, and efficiency. By addressing both functional and non-functional aspects, the outlined requirements provide a comprehensive framework to guide the design and implementation of a robust, user-friendly, and performance-oriented solution.

System Functional Requirements

The functional requirements outline the core capabilities the system must provide to achieve its objectives:

1. **Geofence Monitoring and Detection**:
    - Establish geofence detection around Dawson yards using SkyHawk telematics.
    - Detect vehicle departures and categorize exits (temporary or extended).
2. **Pre-Trip Inspection Validation**:
    - Validate truck inspections by querying the DRM Trip Inspection API.
    - Confirm that the inspection was performed within an 8-hour (default) timeframe when a geofence alert is received.
3. **Automated Supervisor Notifications**:
    - Trigger automated email alerts using SendGrid when a truck exits the yard without a valid inspection.
    - Include detailed information such as truck ID, yard location, supervisor name & contact information, and departure timestamp in the alerts.

4. **Real-Time Data Processing**:
    - Process geofence alerts and inspection validations in real-time to ensure timely notifications.
    - Minimize delays by optimizing API responses and system computations.
5. **Dynamic GPS Data Validation**:
    - Validate the accuracy of GPS data, ensuring it is recent and falls within valid coordinate ranges.
    - Disregard stale or static GPS data to improve the reliability of alerts.
6. **Speed and Distance Verification**:
    - Implement speed checks and calculate distances between the truck and the geofence to confirm actual departures.
    - Use Haversine formula-based calculations for precision.
7. **Inspection Database Queries**:
    - Query the DRM Trip Inspection Database to retrieve inspection status and validate data integrity.
    - Handle incomplete or missing inspection data gracefully by retrying or notifying the appropriate entity.
8. **Frontend Monitoring Dashboard**:
    - Provide a dashboard displaying truck statuses, inspection details, yard locations, and supervisor contacts.
    - Allow supervisors to monitor compliance metrics visually.
9. **Error Logging and Troubleshooting**:
    - Record system errors, such as API failures and geofence misconfigurations, in detailed logs.

10. **Prototype Testing and Deployment**:
    - Conduct extensive testing with dummy APIs and real data to ensure seamless functionality.
    - Deploy the system on a scalable and accessible platform like Heroku for production readiness.

System Non-Functional Requirements

The non-functional requirements define the operational standards the system must meet to function effectively and reliably:

1. System Expansion Capability

- The system architecture must support the monitoring of multiple yards and vehicles simultaneously, accommodating the growth of Dawson Road Maintenance's fleet and operational areas.
- The design should allow seamless integration of additional APIs or monitoring modules without disrupting existing functionality.

2. Real-Time Validation and Processing

- GPS and inspection data must be validated in real-time to ensure immediate and accurate decision-making.
- The geofence monitoring system must distinguish between temporary exits and actual departures, reducing unnecessary notifications.

3. Data Accuracy and Consistency

- Inspection data retrieved from the DRM Trip Inspection API and GPS records from the SkyHawk system must be cross-verified for consistency before triggering notifications.
- The PostgreSQL database schema must ensure structured and reliable storage of inspection and geofence data, avoiding duplication or inconsistencies.

4. Secure Configuration Management

- All sensitive credentials, including API keys for SendGrid, SkyHawk, and PostgreSQL, must be securely managed using Heroku's environment variable feature.
- Access to configuration settings must be restricted to authorized personnel to prevent unauthorized modifications.

5. Supervisor-Focused Usability

- The system must provide concise and actionable email notifications, including critical details like truck ID, inspection status, and yard location, ensuring supervisors can respond promptly.
- The frontend dashboard must display inspection records, geofence events, and truck statuses in an intuitive and user-friendly format.

6. Robust Error Handling and Resilience

- Logs must capture and categorize errors such as API timeouts, invalid GPS data, or email delivery failures, ensuring quick debugging and resolution.
- The system must handle temporary API unavailability gracefully, retrying failed requests without disrupting overall functionality.

7. Modular and Flexible Design

- The codebase must be modular, enabling independent development and maintenance of components like geofence monitoring, inspection validation, and notification systems.
- The system design should support future enhancements, such as integrating advanced analytics tools or additional data sources.

8. Deployment and Hosting Adaptability

- Heroku must provide a stable hosting environment with streamlined deployment processes and the ability to scale resources as demand grows.

- The deployment pipeline should support automated testing and configuration for smoother updates and rollouts.

9. Alert Accuracy Optimization

- False positives must be minimized through GPS validation mechanisms, speed checks, and distance calculations.
- Buffer durations must be implemented to differentiate between short-term yard exits and actual departures.

10. Insightful Data Visualization

- PostgreSQL must support seamless integration with Power BI to generate detailed dashboards, providing supervisors and stakeholders with actionable insights into inspection compliance and truck movements.
- Visualizations should focus on presenting real-time and historical data for decision-making and performance evaluations.

Analysis

The Missed Trip Inspection Notification System addresses the critical need to automate compliance monitoring for pre-trip vehicle inspections, reducing risks associated with manual oversight. This section outlines the in-depth analysis conducted, leveraging extensive research, technical exploration, and knowledge gathering. By understanding the problem domain and defining the key system components, the team established a solid foundation for a scalable and reliable solution.

1. Problem Understanding and Definition

The analysis began with a clear understanding of the operational challenges faced by Dawson Road Maintenance:

- The legal requirement for pre-trip vehicle inspections under CVSE regulations.
- The inefficiency and risks posed by manual inspection tracking, leading to missed inspections and potential penalties.
- The need for real-time monitoring and notifications to improve compliance and safety.

This understanding underscored the importance of a system that seamlessly integrates geofence monitoring, inspection validation, and automated notifications.

2. Research and Knowledge Gathering

The team extensively researched tools, platforms, and technologies to address project requirements:

1. **Heroku**:
    - Studied Heroku's capabilities for deploying scalable web applications and hosting the system's backend.
    - Learned to configure Heroku pipelines, manage environment variables, and optimize deployment processes.

2. **Postman**:
    - Used Postman to test and debug API integrations, including SkyHawk telematics, DRM Trip Inspection, and Yard APIs.
    - Gained expertise in validating request and response data flows for seamless integration.
3. **Trello**:
    - Implemented Trello as a project management tool, creating a Kanban board to track tasks, assign responsibilities, and monitor progress.
    - Used Trello for sprint planning and improving team collaboration.
4. **Programming Languages and Frameworks**:
    - Initially explored Node.js for backend scripting, focusing on its asynchronous capabilities.
    - Transitioned to Python for its extensive library support and ease of integration with APIs and databases.
    - Researched and implemented Python libraries such as Flask, Requests, pytz, and geopy to address specific technical needs.
5. **Databases and Visualization**:
    - Researched PostgreSQL for storing geofence alerts, inspection records, and truck movement data.
    - Learned to connect PostgreSQL to Power BI for creating interactive dashboards that provide actionable insights.

3. Key System Components

Through analysis and knowledge gathering, the following components were identified as critical to the system:

1. **Geofence Monitoring**:
    - SkyHawk telematics and geofence data were analyzed for real-time truck movement tracking.
    - Developed mechanisms to identify valid departures while filtering out temporary exits and calibration errors.
2. **Inspection Validation**:
    - Established validation logic to ensure inspections were performed within the required 8-hour window.
    - Used the DRM Trip Inspection API to fetch and verify inspection records in real time.
3. **Notification System**:
    - Integrated SendGrid for automated email notifications.
    - Ensured notifications included detailed truck information, yard location, and timestamps for immediate supervisor action.
4. **Database and Data Handling**:
    - PostgreSQL was selected for its reliability and scalability in managing inspection and geofence data.
    - Designed efficient query mechanisms for retrieving and processing inspection data.
5. **Visualization and Reporting**:
    - Created Power BI dashboards to visualize compliance metrics, truck movements, and inspection statuses.

4. Challenges and Solutions

The analysis identified several challenges, which were addressed through iterative improvements:

- **Real-Time Processing**:
    - Implemented speed and distance checks to minimize false positives caused by temporary yard exits.
- **API Integration**:
    - Resolved communication issues between SkyHawk and DRM APIs using Postman for debugging and validation.
- **Scalability**:
    - Designed a modular architecture to handle multiple trucks and geofences without performance degradation.
- **Data Visualization**:
    - Researched methods to connect PostgreSQL to Power BI, enabling effective visualization of compliance data. By applying newer AWS SSL certificates, we resolved the connectivity issue.
- **Timezone Conversion:**
    - The APIs returned timestamps in UTC, causing issues with time-sensitive operations. To combat this, we integrated **Pytz** to convert timestamps to the local timezone (PST), ensuring consistent handling of time across all processes.

5. Outcomes of Analysis

The analysis phase resulted in the following outcomes:

- A well-defined problem scope and solution framework.

- A scalable, modular architecture integrating geofence monitoring, inspection validation, and notifications.
- Expertise in deploying and testing the system on Heroku, with optimized API and database interactions.
- A robust data visualization platform using Power BI for actionable insights.

This thorough analysis, grounded in research and technical exploration, ensured that the Missed Trip Inspection Notification System met the operational requirements of Dawson Road Maintenance. By addressing challenges and optimizing the integration of various components, the project achieved a scalable and reliable solution tailored to real-world needs.

Design

The design of the **Trip Inspection System** integrates real-time geofence monitoring, inspection validation, and automated notifications. This section presents the system's key components and workflows, supported by detailed architectures that define the logic and data flow. The design is modular, scalable, and tailored to Dawson Road Maintenance's operational requirements.

1. System Overview

The system is designed to automate pre-trip inspection compliance monitoring by leveraging:

1. **Geofence Alerts**: Detecting truck movements using SkyHawk telematics.
2. **Inspection Validation**: Querying the DRM Inspection API to verify inspection compliance.
3. **Notification Workflow**: Sending actionable alerts to supervisors or fallback recipients.
4. **Database Integration**: Tracking alerts, inspections, and notification status.
5. **Visualization**: Presenting compliance metrics and truck movements through Power BI dashboards.

2. Core Design Architectures

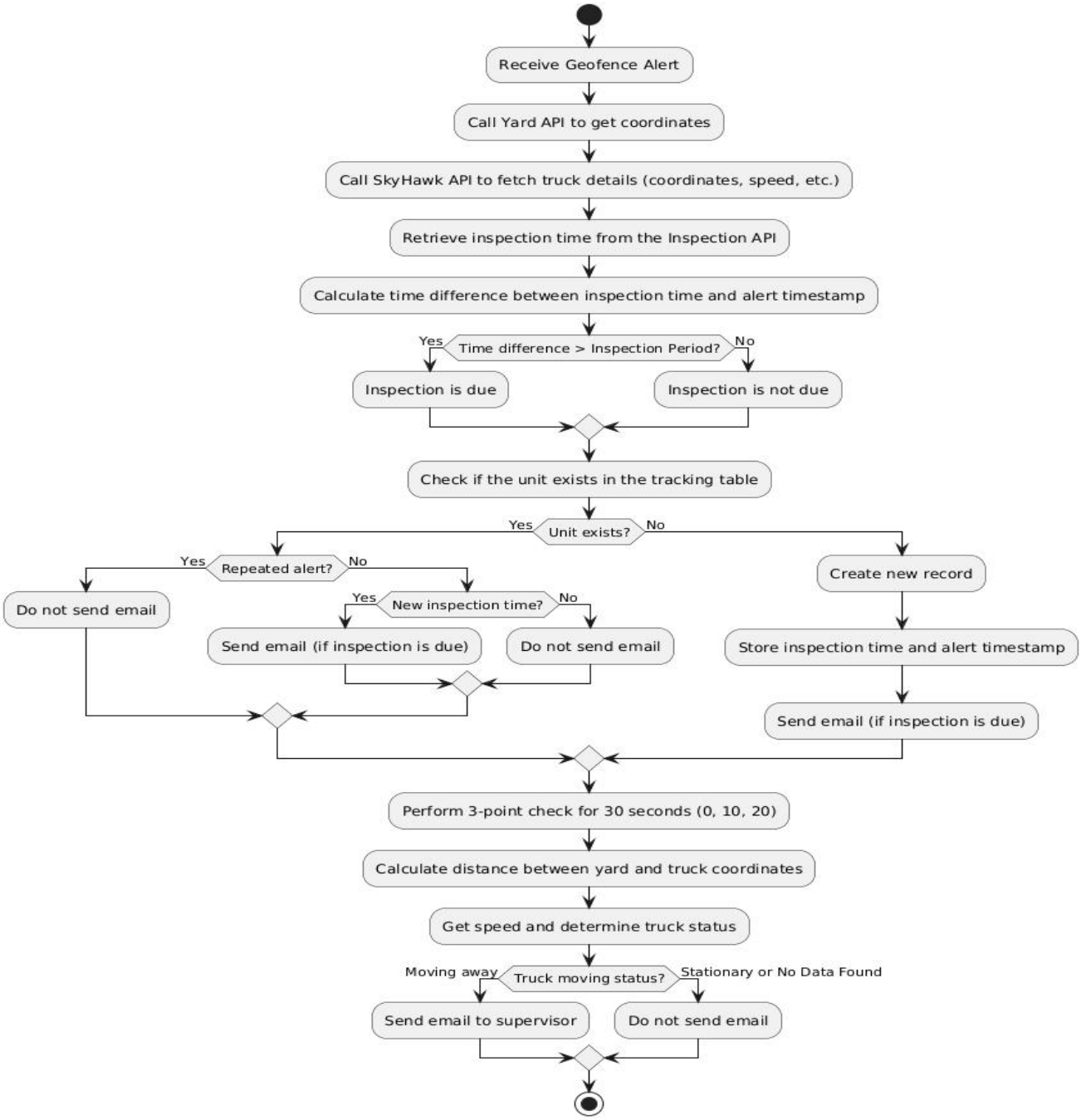
2.1. Full Project Architecture

The project architecture ensures the end-to-end functionality of the system, starting from geofence alert reception to supervisor notification.
Key Steps:

1. **Receive Geofence Alerts**: SkyHawk API triggers an alert when a truck crosses a geofence.

2. **Fetch Data**:
    - **Yard API** retrieves geofence and yard-specific details.
    - **SkyHawk API** provides truck coordinates, speed, and movement data.
    - **Inspection API** returns the last inspection timestamp.
3. **Inspection Validation**:
    - Validates if the last inspection timestamp is within the required inspection period.
4. **Movement Status Check**:
    - Uses a three-point check (over 30 seconds) to confirm if the truck is stationary or moving.
5. **Notification Workflow**:
    - Determines whether an email should be sent based on inspection and movement results.
6. **Database Updates**:
    - Logs alerts, inspections, and email statuses for future reference.

"Full Project Architecture" diagram for a visual representation:



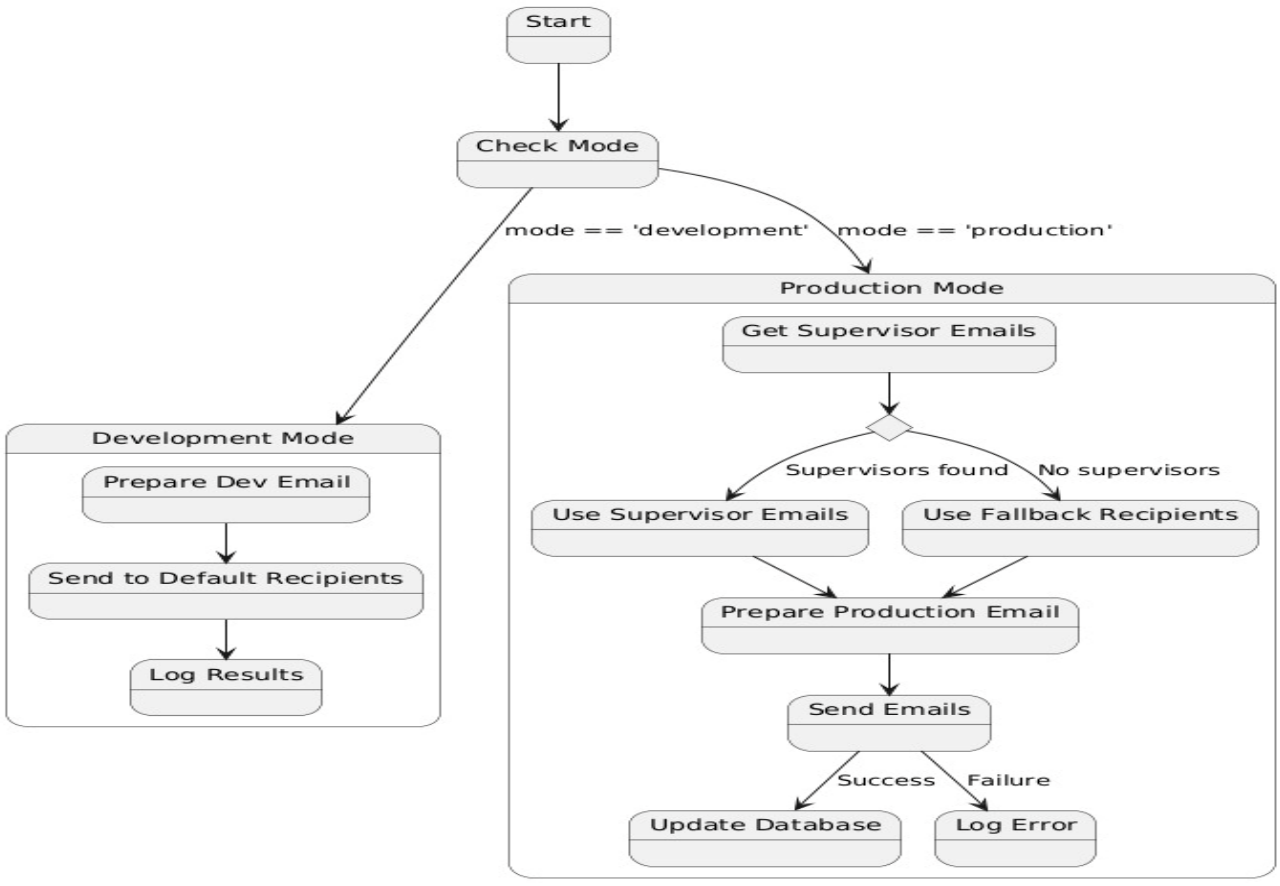
2.2. Email Sending Logic

The email sending workflow ensures clear communication while differentiating between development and production environments.

Key Features:

1. **Environment Modes**:
    - **Development Mode**: Sends test emails to default recipients for validation.
    - **Production Mode**: Sends real emails to supervisors or fallback recipients.
2. **Email Preparation**:
    - Gathers inspection, geofence, and movement details to craft actionable emails.
3. **Error Handling**:
    - Logs failures in email sending and retries when possible.
4. **Database Integration**:
    - Tracks email status to avoid redundant notifications.

"Email Sending Logic" diagram for a detailed workflow:



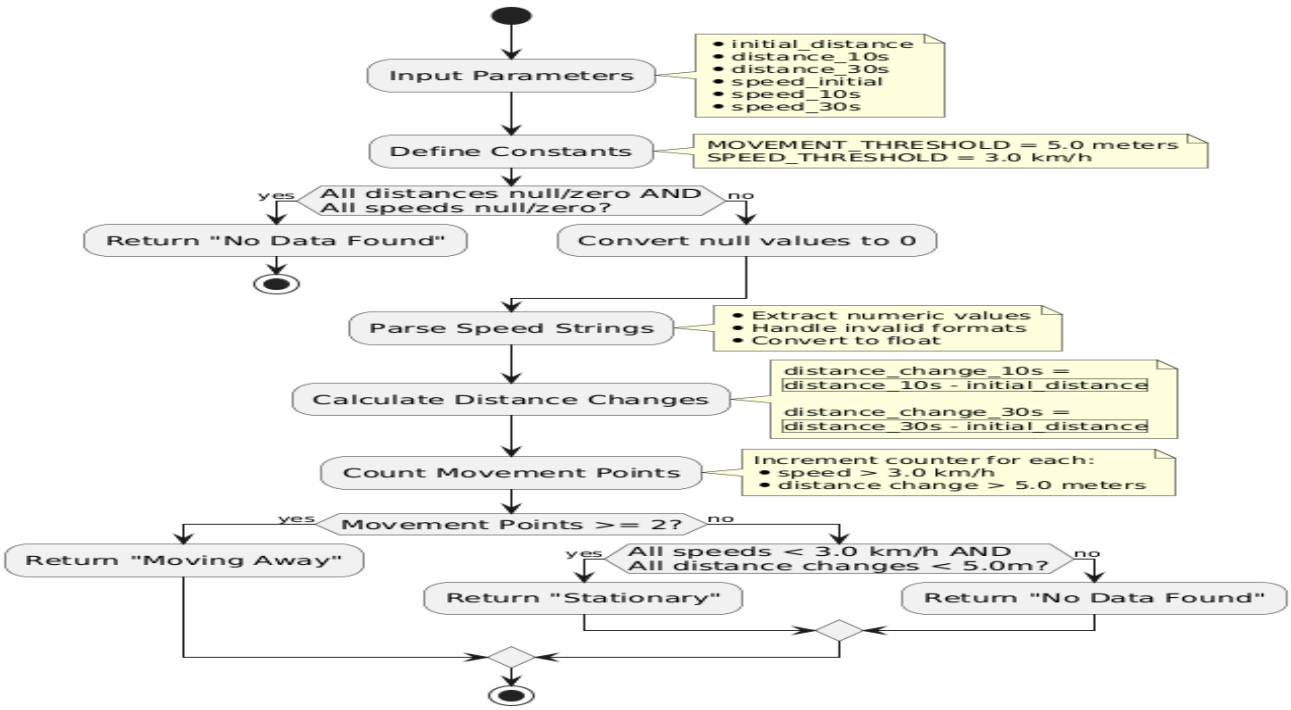
2.3. Moving Status Logic

The moving status logic ensures accurate detection of a truck's movement by analyzing GPS and speed data.

Logic Details:

1. **Three-Point Check**:
    - Collects data at intervals of 0, 10, and 20 seconds to assess movement.
2. **Movement Points**:
    - Assigns points based on speed and distance thresholds.
3. **Status Determination**:
    - **Moving Away**: At least 2 movement points.
    - **Stationary**: All values below thresholds.
    - **No Data Found**: Insufficient or missing inputs.

"Moving Status Logic" diagram for specifics:



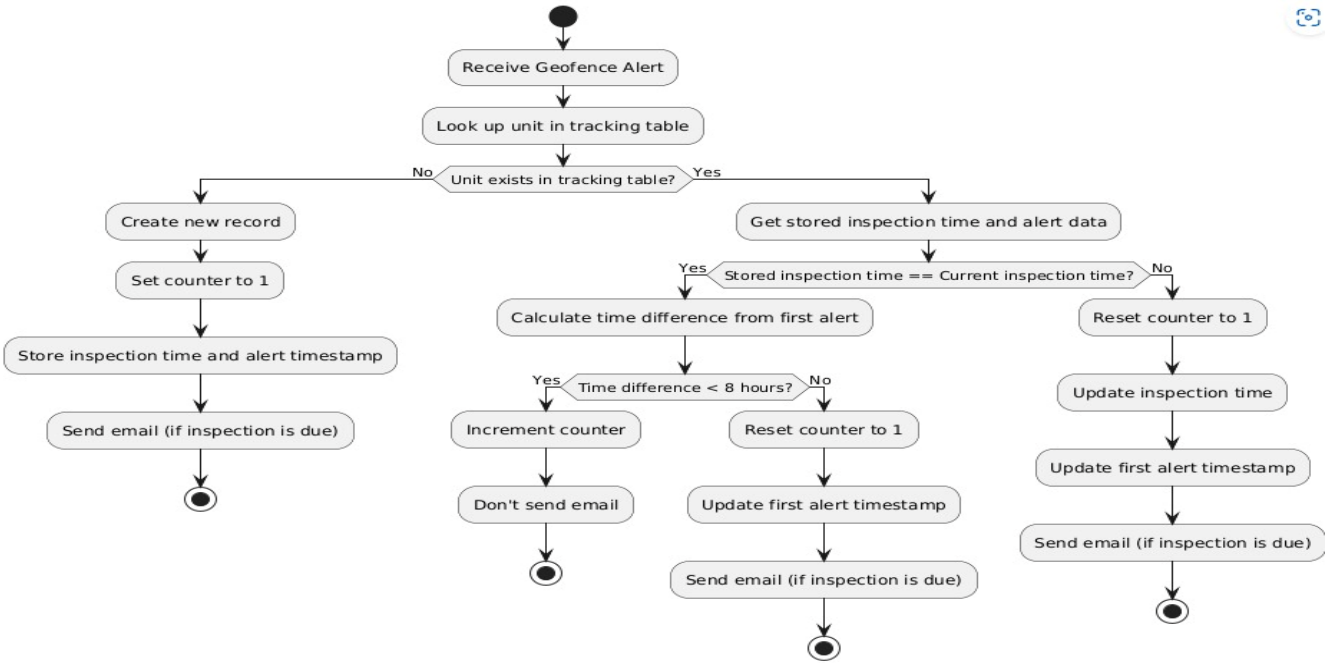
2.4. Multi-Alert Logic

This logic prevents redundant notifications by implementing alert tracking and cooldown periods.

Workflow:

1. **Tracking Alerts**:
    - Maintains inspection timestamps, first alert time, and alert counters in the database.
2. **Cooldown Period**:
    - Ensures no duplicate emails are sent within an 8-hour window for the same inspection.
3. **Reset Conditions**:
    - Resets counters for new inspections or alerts outside the cooldown period.
4. **Fallback Notifications**:
    - Ensures supervisors receive emails even if there is no inspection record available.

"Multi-Alert Logic" diagram for more details:



3. Data Flow and Interaction

The system integrates multiple components through a seamless flow of data:

1. **Input**:
    - Geofence alerts, GPS data, and inspection records.
2. **Processing**:
    - Validates data, determines truck status, and decides notification necessity.
3. **Output**:
    - Sends email alerts and updates the database with status and timestamps.

4. Security and Resilience

- **Environment Variables**:
    - API keys and credentials are securely stored to prevent unauthorized access.
- **Error Handling**:
    - Logs all failures for debugging while ensuring the system continues to operate for unaffected components.

5. Modular Design

The system is structured into distinct modules:

1. **Alert Handling**:
    - Processes geofence alerts and fetches required data.
2. **Validation**:
    - Validates inspection compliance and truck movement status.

3. **Notifications**:
    - Prepares and sends emails based on validation results.
4. **Database Integration**:
    - Tracks alerts, inspection times, and email statuses for future reference.

The design ensures that the system is capable of handling real-time geofence alerts, inspection validations, and automated notifications. With modular components and detailed logic, the system meets Dawson Road Maintenance's requirements while maintaining scalability and reliability.

Implementation

The **Trip Inspection System** was implemented using a modular and scalable approach to ensure real-time monitoring, reliable notifications, and ease of deployment. Below is an overview of the implementation process and components:

1. System Architecture

The system integrates multiple APIs, a database, and a web interface to monitor truck movements and inspection compliance:

- **Input:** Geofence alerts from the SkyHawk API and inspection data from the DRM API.
- **Processing:** The system validates inspection compliance and truck status using geofence data and movement analysis logic.
- **Output:** Notifications sent to supervisors and dashboards displaying compliance metrics.

2. Backend Development

- Developed using **Flask**, which provides a lightweight framework for building RESTful APIs and rendering web pages.
- Key APIs:
    - **SkyHawk API:** Retrieves truck telemetry data, including GPS coordinates and speed.
    - **DRM API:** Provides inspection records and supervisor contact details.
    - **Custom Endpoints:**
        - /geofence-alert: Handles incoming geofence alerts.
        - /all-geofence-alerts: Displays a history of geofence alerts.

3. Database Integration

- **PostgreSQL** was used for robust and scalable data storage.
- Schema Design:
    - geofence_alerts: Stores records of geofence breaches.
    - unit_alert_tracking: Tracks alerts for multi-alert management.
    - yard_supervisors: Stores supervisor details for each yard.
- **Psycopg2** was used for database communication.

4. Movement Detection Logic

- Implemented a three-point movement detection algorithm:
    - Monitored truck speed and distance over three intervals (initial, 10s, 30s).
    - Classified movements into "Stationary," "Moving Away," or "No Data Found" based on thresholds:
        - Speed > 15 km/h or distance > 10 meters → "Moving Away."
        - Speed < 5 km/h and stable distance → "Stationary."

5. Automated Notifications

- **SendGrid** was integrated for email alerts.
    - Alerts include truck ID, geofence location, inspection status, and timestamps.
    - Multi-alert management prevents redundant notifications by implementing an 8-hour cooldown period.

6. Web Dashboard

- Built using **Jinja2 templates** for dynamic HTML rendering.
- Key Features:
    - Displayed geofence alerts and truck statuses.
    - Configurable settings for alert periods and thresholds.
    - Accessible at /all-geofence-alerts and /settings endpoints.

7. Deployment

- Hosted on **Heroku** for scalability and ease of access.
    - Environment variables (e.g., API keys, database credentials) were securely configured using .env files.
    - **Gunicorn** served the Flask application in production.

8. Testing and Debugging

- Comprehensive testing was conducted using **Postman** and curl for API endpoints.
- Logs and database queries were used to debug and validate system behavior:
    - Verified geofence alert processing and movement detection.
    - Ensured email alerts were delivered accurately.

9. Power BI Integration

- Connected **PostgreSQL** to **Power BI** for real-time visualization of inspection compliance and truck movements.

- Dashboards provided actionable insights with drill-down capabilities for specific yards and trucks.

This implementation ensures a robust, real-time system that meets Dawson Road Maintenance's operational needs while remaining scalable and reliable for future enhancements.

Testing

The **Trip Inspection System** was tested rigorously across multiple phases to ensure functionality, reliability, and robustness. The testing included geofence alerts, inspection validations, movement status determination, email notifications, error handling, and deployment processes. This section highlights both successful and failed scenarios, providing a detailed overview of the iterative improvements made during testing.

1. Geofence Alert Handling

Test Case ID	**Test Description**	**Input**	**Expected Outcome**	**Result**
T1	Single geofence alert	Simulated truck geofence alert	Yard API and SkyHawk API are called successfully; correct alert is logged.	**Pass**
T2	Multiple geofence alerts	Simulated alerts for 2 trucks	Both alerts are processed and tracked correctly.	**Pass**
T3	Invalid geofence data	Malformed geofence alert data	Error is logged; no further processing occurs.	**Pass**
T4	SkyHawk API failure	Valid geofence alert	API failed to return truck details; error logged.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**

2. Inspection Validation

Test Case ID	Test Description	Input	Expected Outcome	Result
T5	Inspection overdue	Inspection timestamp older than 8 hours	Inspection flagged as overdue.	**Pass**
T6	Valid inspection	Inspection timestamp within 8 hours	No email is sent; status logged correctly.	**Pass**
T7	Missing inspection data	No inspection timestamp retrieved	Alert triggered for overdue inspection.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**

3. Movement Status Determination

Test Case ID	Test Description	Input	Expected Outcome	Result
T8	Moving away	Speed > 3 km/h and distance > 5 meters	Status set to "Moving Away."	**Pass**
T9	Stationary	Speed < 3 km/h and distance < 5 meters	Status set to "Stationary."	**Pass**
T10	No data available	Missing GPS or speed data	Status set to "No Data Found"; no email sent.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**
T11	Brief yard re-entry	Speed and distance change intermittently	No email is sent; status remains "Stationary."	**Pass**

4. Multi-Alert Logic

Test Case ID	Test Description	Input	Expected Outcome	Result
T12	Repeated alert for same truck	Alerts within 8-hour window	No duplicate email is sent.	**Pass**
T13	Multiple trucks	Simulated alerts for multiple trucks	Separate emails sent for overdue inspections.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**
T14	New inspection time	New inspection timestamp detected	Tracking resets; email sent only if overdue.	**Pass**

5. Notification System

Test Case ID	Test Description	Input	Expected Outcome	Result
T15	Valid supervisor emails	Supervisor email exists	Email is sent successfully; database updated.	**Pass**
T16	Missing supervisor emails	No supervisor email found	Email sent to fallback recipients.	**Pass**
T17	Email server failure	Simulated SendGrid outage	Error logged; no system crash occurs.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**

6. Error Handling

Test Case ID	Test Description	Input	Expected Outcome	Result
T18	API timeout	Simulated timeout for SkyHawk API	Error logged; retry logic executed successfully.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**

Test Case ID	Test Description	Input	Expected Outcome	Result
T19	Malformed API response	Invalid data from Inspection API	Error logged; invalid data skipped.	**Pass**
T20	Missing geofence alert data	Alert without truck ID or coordinates	Error logged; no processing occurs.	**Pass**

7. Deployment and Performance

Test Case ID	Test Description	Input	Expected Outcome	Result
T21	Heroku deployment	Full system deployed	System deployed without errors.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**
T22	High alert volume	Simulated alerts for 10+ trucks	System processes alerts without performance degradation.	**Pass**
T23	SkyHawk API failure	SkyHawk API returns no data	Error logged; other components unaffected.	**Fail (1st Phase)**; Fixed in 2nd Phase **Pass**

8. Results and Observations

- **1st Phase**:
    - Multiple failures identified in multi-alert logic, API integrations, and email notifications.
    - SkyHawk API timeout handling was missing, causing data gaps.
    - Email dispatch failed occasionally due to configuration issues with SendGrid.
    - Heroku deployment encountered environment variable misconfigurations.

- **2nd Phase**:
    - All identified issues were addressed:
        - Robust error handling for API failures and malformed responses.
        - Improved multi-alert logic to handle simultaneous alerts efficiently.
        - Fixed email dispatch issues and added fallback mechanisms for missing data.
        - Deployment pipeline on Heroku stabilized with proper configurations.

The testing process revealed initial gaps that were systematically resolved during iterative testing phases. By simulating real-world scenarios, such as multiple trucks and API failures, the system demonstrated robust functionality, reliability, and resilience in its final form.

Conclusion

Dawson Road Maintenance has taken a giant leap forward with the Trip Inspection and Geofence Alert System in its commitment to improving compliance and operational efficiency. The system integrates advanced geofencing, inspection validation and automated notifications to guarantee that pre-trip inspection regulations are met with precision. The fleet managers' challenges are well understood, and its user-friendly interface, robust backend architecture, and seamless API integrations demonstrate this.

Real time data validation, secure notifications and dynamic dashboards were integrated into a rigorous development process, from initial concept to final deployment, to provide supervisors actionable insights at a glance. Additionally, the system's ability to handle challenges such as time zone consistency, API communication errors and redundant notifications shows the system's resilience and scalability.

This project is a result of our team's efforts working with the most up to date technologies such as PostgreSQL, Flask, and Power BI, and delivering a solution that is perfectly aligned with Dawson's operational needs. Future scalability has been factored into the design of the system so that it will remain relevant as fleet operations increase.

References:

Python Libraries:

1. Flask - https://flask.palletsprojects.com
2. Psycopg - https://www.psycopg.org/doc
3. Pytz - https://pypi.org/project/pytz
4. Geopy - https://geopy.readthedocs.io
5. SendGrid API Client - https://sendgrid.com/docs/API_Reference
6. Jinja2 - https://jinja.palletsprojects.com

Database and Visualization:

1. PostgreSQL - https://www.postgresql.org/docs
2. PGAdmin - https://www.pgadmin.org/
3. Power BI - https://learn.microsoft.com/en-us/power-bi
4. AWS RDS SSL Certificates - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html

Deployment and Hosting:

1. Heroku - https://devcenter.heroku.com/](https://devcenter.heroku.com

APIs Used:

1. SkyHawk API - Provided by Dawson Maintenance for truck telemetry (speed, location, etc.).
2. Geofence Alert API - API to log alerts when trucks exit geofenced areas.
3. DRM Trip Inspection API - Provides data for inspections, yard details, and supervisor contact information.

Tools Used:

1. Postman - https://learning.postman.com
2. GitHub - https://docs.github.com

Python Runtime:

1. Python - https://docs.python.org/3/

Other Technologies:

1. HTML & CSS(Jinja2 Templates) - https://jinja.palletsprojects.com/en/stable/

Appendixes

1. Weekly Reports

The following section provides a detailed summary of weekly progress and team discussions throughout the development of the " Trip Inspection System."

Week 1: Initial Collaboration

- **Tasks Completed**:
    - Developed an internal collaboration plan, arranging team meetings three times a week.
    - Defined individual roles and responsibilities within the team.
- **Key Discussion Points**:
    - Strategies for effective collaboration and communication.
- **Milestone**:
    - Established a collaborative workflow and organized the team structure.

Week 2: Project Planning

- **Tasks Completed**:
    - Contacted Dawson representatives to arrange the first meeting.
    - Conducted background research on Dawson Road Maintenance operations and requirements.
    - Finalized team availability for meetings with Dawson representatives.
- **Key Discussion Points**:
    - Drafted preliminary questions for the first meeting with Dawson.
    - Analyzed potential project challenges based on initial research.

- **Milestone**:
    - Prepared for the first formal meeting with Dawson representatives.

Week 3: Project Kickoff

- **Tasks Completed**:
    - Held the first meeting with Dawson representatives (Brian and Daniel).
    - Reviewed SkyHawk telematics documentation and set up Postman for API testing.
    - Drafted and submitted the first version of the project charter.
- **Key Discussion Points**:
    - Clarified technical aspects of SkyHawk and DRM APIs.
    - Identified initial challenges, including false positives and integration complexities.
- **Milestone**:
    - Confirmed project scope and initiated API testing.

Week 4: Technical Exploration

- **Tasks Completed**:
    - Conducted high-level design of the software architecture.
    - Tested SkyHawk and DRM Trip Inspection APIs using Postman.
    - Configured the Trello Kanban board for task tracking.
- **Key Discussion Points**:
    - Explored methods to validate geofence alerts effectively.

    - Analyzed how API data could be leveraged for inspection validation.
- **Milestone**:
    - Established the architecture and validated core API functionality.

Week 5: Prototype Development

- **Tasks Completed**:
    - Built the initial prototype script for geofence monitoring and inspection validation.
    - Configured the GitHub repository for version control and collaboration.
    - Implemented initial API integrations within the prototype.
- **Key Discussion Points**:
    - Addressed potential data handling issues during API integration.
    - Planned strategies for minimizing false positives in alerts.
- **Milestone**:
    - Delivered a functional prototype demonstrating geofence monitoring.

Week 6: Notification System

- **Tasks Completed**:
    - Configured and tested SendGrid for automated email notifications.
    - Conducted live testing of the notification system using dummy data.
    - Introduced error handling for API failures and invalid data.
- **Key Discussion Points**:
    - Reviewed email formatting for clarity and professionalism.

    - Discussed methods to handle notification delays.

- **Milestone**:
    - Implemented and validated the notification system workflow.

Week 7: Frontend and Database Development

- **Tasks Completed**:
    - Designed a frontend dashboard displaying inspection statuses and truck details.
    - Established a PostgreSQL database for storing geofence alerts and inspection records.
    - Began exploring Power BI integration for compliance visualization.
- **Key Discussion Points**:
    - Discussed frontend requirements for supervisor usability.
    - Identified database schema requirements for efficient data storage and retrieval.
- **Milestone**:
    - Delivered an operational frontend and database.

Week 8: Refinement and Debugging

- **Tasks Completed**:
    - Enhanced GPS validation logic using distance calculations and two-point checks.
    - Introduced logging mechanisms for API failures and invalid data detection.

    - Tested deployment on Heroku for real-time functionality.

- **Key Discussion Points**:
    - Addressed geofence calibration issues and brief exits causing false positives.
    - Discussed ways to improve error resilience.
- **Milestone**:
    - Improved system accuracy and deployed on Heroku for scalability.

Week 9: Testing and GeoFence Alert System Deployment

- **Tasks Completed:**
    - Created a dummy script with test truck data and conducted local server testing.
    - Designed and implemented a database structure to store geofence alert details.
    - Successfully deployed the GeoFence Alert System on Heroku for real-time functionality.
    - Updated the Trello board to reflect ongoing progress and completed tasks.
    - Reviewed and refined strategies for managing false positives, focusing on geofence calibration and detection logic.
- **Key Discussion Points:**
    - Discussed optimization strategies for the database to ensure efficient storage and retrieval of geofence alerts.
    - Identified methods to address potential false positives caused by brief geofence exits or data anomalies.

- **Milestone:**
    - Deployed a scalable GeoFence Alert System on Heroku and finalized database design for improved system efficiency.

Week 10: Modularization and Visualization

- **Tasks Completed**:
    - Modularized the codebase for improved scalability and maintainability.
    - Developed Power BI dashboards for visualizing truck movements and inspection compliance.
    - Conducted integration testing across all system components.
- **Key Discussion Points**:
    - Reviewed Power BI dashboard insights for actionable decision-making.
    - Discussed final refinements to code structure and database queries.
- **Milestone**:
    - Delivered a modular and fully integrated system.

Week 11: System Optimization

- **Tasks Completed**:
    - Optimized API call efficiency to reduce latency.
    - Finalized frontend enhancements for user experience.
    - Conducted rigorous end-to-end testing to ensure system stability.
- **Key Discussion Points**:
    - Addressed final usability improvements for supervisors.

    - Discussed post-deployment monitoring strategies.

- **Milestone**:
    - Delivered a stable and optimized system.

Week 12: Final Preparations

- **Tasks Completed**:
    - Presented Power BI dashboards to Dawson representatives.
    - Addressed feedback related to timestamp errors and notification timings.
    - Prepared comprehensive documentation for system handover.
- **Key Discussion Points**:
    - Finalized user documentation and deployment readiness.
    - Reviewed client feedback for potential post-deployment enhancements.
- **Milestone**:
    - Prepared the system for production deployment and client use.

2. Project Proposal

The Project Proposal served as the foundation for defining the scope and objectives of the " Trip Inspection System." Below is a summary of its key components:

- **Problem Statement**:
    - A pre-trip inspection is mandatory for heavy trucks before leaving the yard. Missed inspections can lead to legal and operational repercussions, particularly during audits by CVSE.
    - There was a need for an automated system to monitor truck departures and validate inspection status to prevent compliance lapses.

- **Proposed Solution**:
    - Leverage the SkyHawk telematics system installed on all heavy trucks, with geofences set up around Dawson yards.
    - Build an automated system to:
        - Monitor truck departures.
        - Validate inspection records through the DRM Trip Inspection API.
        - Trigger email notifications to supervisors for missed or invalid inspections.
- **Objectives**:
    - Ensure consistent compliance with inspection regulations.
    - Minimize manual oversight by automating geofence monitoring and notification workflows.
    - Enhance operational efficiency and supervisor response times.
- **Constraints and Tools**:
    - Integration with SkyHawk and DRM APIs using secure communication protocols.
    - Deployment of the system on Heroku for scalability and accessibility.
    - Use of Postman for API testing and Python for backend development.

This proposal laid the groundwork for the project's successful execution, aligning team efforts with Dawson Road Maintenance's operational goals.

3. Midterm Review

The Midterm Review highlighted the team's progress, technical achievements, and addressed challenges encountered during the development of the system. Below is a summary of the key points covered:

- **Progress Overview**:
    - Automated the SkyHawk authentication process and integrated it with the DRM API for accurate data exchange.
    - Developed a functional prototype with a fully implemented notification system using SendGrid for email alerts.
    - Implemented advanced asset tracking by leveraging GPS coordinates, speed, and proximity to DRM yards.
- **Script and System Results**:
    - Alerts included critical details such as asset information, yard name, distance, and supervisor contacts.
    - Notifications were triggered only when inspections were overdue, ensuring precise and actionable alerts.

- **Challenges and Solutions**:
    - **Geofence Alerts**:
        - Challenge: Handling rapid re-entries and false positives.
        - Solution: Implemented geodesic calculations for proximity checks and added a buffer duration for validation.
    - **Scalability**:
        - Challenge: Supporting multiple geofences and trucks simultaneously.

        - Solution: Modularized the design to optimize API calls and reduce performance overhead.
    - **Error Handling**:
        - Challenge: Addressing API failures and notification errors.
        - Solution: Enhanced logging and implemented fallback mechanisms for robust error management.
- **Stretch Goals**:
    - Introduced GPS validation improvements, including range checks and speed validation.
    - Explored Power BI for compliance dashboards and visualizations.
    - Modularized the codebase further for enhanced maintainability and efficiency.

The review concluded with a demonstration of the prototype, showcasing the progress achieved thus far and setting the stage for the system's final development phase.

4. CS Showcase

The CS Showcase presentation provided an opportunity to demonstrate the "Trip Inspection System" to an audience of peers, faculty, and industry professionals. The presentation outlined the project's objectives, technical achievements, and real-world applications. Below is a summary of the key points covered:

Presentation Overview:

1. **Problem Description:**
    - Heavy truck pre-trip inspections are legally required but often missed due to oversight, leading to compliance risks and potential penalties.
    - Supervisors require a robust, automated notification system to address missed inspections effectively.

2. **Proposed Solution:**
    - Automate the tracking of truck movements using SkyHawk telematics and geofences.
    - Verify inspection status through DRM APIs in real-time.
    - Notify supervisors via email when a truck leaves the yard without a valid inspection.
3. **Objectives:**
    - Enhance compliance with inspection regulations.
    - Minimize manual oversight with automation.
    - Improve operational efficiency through real-time monitoring.

Key Features and Demonstration:

The presentation included a live demonstration highlighting the following key features:

- **Real-Time Monitoring:**
    - Truck movements were tracked using geofence alerts from SkyHawk telematics.
    - The system verified inspection records via DRM APIs and identified overdue inspections.
- **Automated Alerts:**
    - Email notifications were sent to supervisors using SendGrid for missed inspections or geofence breaches.
    - Notifications included truck ID, inspection details, yard location, and timestamps for immediate action.

- **User-Friendly Dashboard:**
    - A web-based interface provided real-time insights into truck locations, inspection logs, and compliance metrics.
    - Built using Flask and Bootstrap, the dashboard ensured responsiveness and ease of use.
- **Advanced Movement Detection:**
    - Implemented three-point checks for precision in movement detection.
    - Utilized speed and distance thresholds to differentiate between stationary and moving trucks.
- **Power BI Integration:**
    - Interactive dashboards visualized compliance data, truck movements, and inspection statuses.
    - Enabled fleet managers to analyze trends and identify areas for improvement.

Challenges Addressed:

1. **False Positives:**
    - Brief yard re-entries caused unnecessary alerts.
    - Implemented buffer periods and geodesic calculations to reduce false positives.
2. **Scalability:**
    - System tested to handle multiple trucks and geofence alerts without performance degradation.

3. **API Integration:**
    - Resolved communication and data formatting issues with SkyHawk and DRM APIs using Postman and error logging mechanisms.

5. Final Client Presentation

The final client presentation was an opportunity to showcase the complete "Trip Inspection System" to Dawson Road Maintenance representatives and other stakeholders. It highlighted the problem, solution, technical achievements, and the impact of the project. Below is a summary of the key points covered:

Presentation Overview:

1. **Problem Description:**
    - Heavy truck pre-trip inspections are legally required but are occasionally missed, leading to compliance risks, operational inefficiencies, and potential penalties from Commercial Vehicle Safety Enforcement (CVSE).
    - Supervisors lacked an automated system to monitor, and address missed inspections effectively.
2. **Proposed Solution:**
    - Develop an automated, real-time system to monitor truck movements, validate inspection status, and notify supervisors of missed or overdue inspections.
3. **Objectives:**
    - Ensure compliance with inspection regulations.
    - Automate processes to reduce manual oversight.
    - Provide supervisors with actionable information in real-time.

Key Features Demonstrated:

1. **Real-Time Monitoring:**
    - The system utilized geofencing technology via the SkyHawk API to detect truck movements and validate trip inspections.
    - Integrated with the DRM API to retrieve inspection data for compliance verification.
2. **Automated Notifications:**
    - Notifications were sent to supervisors via SendGrid for missed or overdue inspections.
    - Fallback mechanisms ensured all alerts reached alternative contacts if primary supervisors were unavailable.
3. **User-Friendly Web Dashboard:**
    - A dashboard provided supervisors with real-time visibility into truck locations, inspection logs, and geofence breaches.
    - Built using Flask and Bootstrap, ensuring a responsive and intuitive interface.
4. **Advanced Movement Detection:**
    - The system implemented three-point checks to determine truck status (e.g., moving away or stationary) based on speed and distance measurements.
    - Reduced false positives through precise calculations.
5. **Data Centralization and Power BI Integration:**
    - Centralized data storage in PostgreSQL enabled seamless retrieval and analysis of inspection records and geofence alerts.

- Power BI dashboards provided actionable insights into compliance trends, truck movements, and operational metrics.

Challenges Addressed:

1. **False Positives:**
    - Implemented buffer periods and geodesic calculations to minimize unnecessary alerts caused by brief re-entries into geofenced areas.
2. **Scalability:**
    - Designed the system to handle an increasing number of trucks and geofence alerts without performance degradation.
3. **API Integration:**
    - Resolved data formatting and latency issues with SkyHawk and DRM APIs through debugging tools like Postman.