



Name	Student Number
Nicholas Dobson	u23671964
Shaylin Govender	u20498952
Lonwabo	u22516949
Mpho	u22668323
Aryan Mohanlall	u23565536

# Traffic Guardian AWS SRS\_V4

## Team: Quantum Quenchers

quantumquenchers@gmail.com

COS301

Capstone Project

University of Pretoria

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1 Business Context and Justification.....	3
1.2 Purpose.....	3
1.3 Document Scope.....	4
<b>2. User Stories / User Characteristics.....</b>	<b>5</b>
2.1 Primary Users.....	5
2.1.1 Traffic Management Centre Operators.....	5
2.1.2 Emergency Response Coordinators.....	6
2.1.3 Traffic Management Supervisors.....	7
2.2 Secondary Users.....	8
2.2.1 System Administrators.....	8
2.2.2 Data Analysts.....	9
2.3 Tertiary Users.....	10
2.3.1 Maintenance Personnel.....	10
2.3.2 Training Coordinators.....	10
2.4 User Interaction Patterns.....	11
2.4.1 High-Frequency Interactions (Multiple times per shift).....	11
2.4.2 Medium-Frequency Interactions (Daily/Weekly).....	11
2.5 User Success Criteria.....	12
<b>3. Updated Use Case Diagrams.....</b>	<b>13</b>
Use Case 1: Login and Registration.....	13
Use Case 2: Incident Reporting.....	14
Use Case 3: Incident Status Update.....	14
Use Case 4: Media Ingestion.....	15
Use Case 5: Real Time Incident Alerts.....	15
Use Case 6: AI Incident Detection.....	16
<b>4. Updated Domain Model.....</b>	<b>18</b>
<b>5. Service Contracts.....</b>	<b>19</b>
5.1 Contract Architecture Overview.....	19
5.2 Authentication and Authorization Contract.....	19
5.3 Incident Management Service Contract.....	20
5.4 Alert and Notification Service Contract.....	20
5.5 Camera Feed and Media Processing Contract.....	21
5.6 Error Handling and Reliability Contracts.....	21
5.7 Data Format and Protocol Standards.....	22
5.8 Testing and Validation Framework.....	22
<b>7. Quality Requirements.....</b>	<b>28</b>
Q1 Usability.....	28
Q2 Extensibility.....	29
Q3 Interoperability.....	30
Q4 Availability.....	31
Q5 Performance.....	32
Q6 Scalability.....	33
<b>8. Technologies Requirements.....</b>	<b>34</b>
8.1 Technologies Overview.....	35
8.1.1 Cloud Platform – AWS (Free Tier).....	35
8.1.2 Video Ingestion – OpenCV.....	35
8.1.3 Computer Vision – YOLO, OpenCV.....	35
8.1.4 Backend API – Node.js.....	35
8.1.5 Frontend Dashboard – React.js.....	36
8.1.6 Database & Storage – PostgreSQL.....	36
8.1.7 Notification System – AWS SNS, WebSockets.....	36
8.1.8 Security – HTTPS, AWS IAM, AES Encryption.....	36
8.1.10 CI/CD Pipeline – GitHub Actions.....	37
<b>9. Agile Methodology and Proposed Roadmap.....</b>	<b>38</b>
9.1 Preparation Phase.....	38
9.2 Basic Video Processing and Incident Detection.....	38
9.3 Enhanced Detection and Real-time Notification.....	39
9.4 Geolocation Integration and Advanced Analytics.....	39
9.5 Security Hardening and Production Readiness.....	40

# 1. INTRODUCTION

## 1.1 Business Context and Justification

Gauteng province manages South Africa's busiest road network, where traffic incidents can escalate into major congestion events affecting thousands of commuters and causing significant economic impact. Current incident detection relies on manual camera monitoring, citizen reports, and patrol observations, introducing critical 5-15 minute delays between incident occurrence and emergency response.


Traffic Guardian is envisioned to address this challenge by providing:

- **Automated Detection:** Incident identification within seconds using computer vision
- **Intelligent Classification:** AI-powered severity assessment and incident categorisation
- **Immediate Response:** Real-time alerts to traffic management operators
- **Operational Efficiency:** Reduced human monitoring load and faster emergency dispatch

## 1.2 Purpose

**Traffic Guardian** is envisioned as a real-time traffic incident detection and reporting system that enhances road safety and operational efficiency across Gauteng's high-volume highways. Using live video streams from existing traffic cameras, the system will automatically detect and classify incidents such as accidents, congestion, stalled vehicles, and road hazards.

As the implementing team, our approach is structured around building a robust and scalable prototype within the constraints of the **AWS Free Tier**. We will use **computer vision** and **deep learning models** to analyse live camera feeds, identify anomalies, and trigger real-time alerts with geolocation data. This information will be displayed via a **web-based dashboard** designed for use by traffic control centre operators.



To ensure a modular and maintainable system, our development process will be guided by software engineering best practices, including:

- **Agile methodology** to adapt to changes and iterate quickly.
- **Domain-driven design** to ensure all components align with real-world traffic operations.
- **Security-by-design principles** to meet POPI compliance and enforce data protection.

Our solution will emphasise automation, accuracy, and ease of use, transforming passive camera networks into active monitoring tools while giving traffic personnel the insights needed to respond swiftly and effectively.

### 1.3 Document Scope

This SRS encompasses all functional and non-functional requirements for delivering a production-ready prototype, a foreseeable demo with comprehensive incident detection capabilities. The scope includes:

- **Core Functionality:** Real-time video processing, AI-powered incident detection and classification, automated alerting system
- **User Interface:** Web-based dashboard for traffic management centre operators with live feed monitoring and incident management
- **Data Management:** Incident archiving with searchable metadata, analytics, and POPI Act-compliant data handling
- **Cloud Integration:** Complete AWS service integration operating within Free Tier limitations
- **Security Requirements:** End-to-end encryption, role-based access control, and regulatory compliance

The system excludes mobile applications, weather correlation features, and license plate recognition for the initial demo, but these may be implemented in future.



## 2. User Stories / User Characteristics

### 2.1 Primary Users

#### 2.1.1 Traffic Management Centre Operators

**Profile:** Experienced traffic control personnel responsible for real-time highway monitoring and emergency response coordination. Work in high-stress environments requiring rapid decision-making and multi-tasking across multiple camera feeds and communication channels.

**Technical Proficiency:** Moderate - familiar with traffic management software and dashboard interfaces

**Primary Goals:** Minimise incident response time, maintain traffic flow, coordinate emergency services

#### User Stories:

- **US01:** As a traffic operator, I want to receive immediate visual and audio alerts when incidents are detected so that I can respond within 30 seconds of occurrence
- **US02:** As a traffic operator, I want to view live camera feeds with AI-highlighted incident areas so that I can quickly verify and assess the situation severity
- **US03:** As a traffic operator, I want to acknowledge alerts with one-click actions so that I can prevent duplicate emergency dispatches while coordinating response
- **US04:** As a traffic operator, I want to see the incident location on an interactive map so that I can provide precise coordinates to emergency responders
- **US05:** As a traffic operator, I want to add manual notes to incidents so that I can document additional context for response teams and shift handovers

## 2.1.2 Emergency Response Coordinators

**Profile:** Senior personnel responsible for dispatching ambulance, fire, police, and tow truck services. Require detailed incident information to allocate appropriate resources and brief response teams effectively.

**Technical Proficiency:** Basic to Moderate - primarily use communication systems and dispatch software

**Primary Goals:** Optimise resource allocation, ensure appropriate response level, minimise response time

### User Stories:

- **US06:** As an emergency coordinator, I want to receive prioritised incident alerts with severity levels so that I can dispatch appropriate resources (ambulance vs tow truck)
- **US07:** As an emergency coordinator, I want to access incident snapshots and video clips so that I can brief response teams on scene conditions
- **US08:** As an emergency coordinator, I want to track incident status updates so that I can monitor response progress and determine when additional resources are needed
- **US09:** As an emergency coordinator, I want to see the estimated incident duration based on type and severity so that I can plan resource deployment schedules
- **US10:** As an emergency coordinator, I want to receive escalation alerts for unacknowledged critical incidents so that no emergency response is delayed

### **2.1.3 Traffic Management Supervisors**

**Profile:** Senior operations staff overseeing traffic control activities, analysing system performance, and making strategic operational decisions. Responsible for staff scheduling, system optimisation, and reporting to management.

**Technical Proficiency:** High - extensive experience with traffic management systems and data analysis

**Work Environment:** Operations oversight with access to comprehensive system data and reports

**Primary Goals:** Optimise system performance, analyse operational patterns, ensure service quality

#### **User Stories:**

- **US11:** As a traffic supervisor, I want to access incident analytics dashboards so that I can identify traffic pattern trends and high-incident locations
- **US12:** As a traffic supervisor, I want to monitor system performance metrics (detection accuracy, response times) so that I can identify areas for improvement
- **US13:** As a traffic supervisor, I want to configure alert thresholds and detection sensitivity so that I can balance false positives with detection coverage
- **US14:** As a traffic supervisor, I want to generate comprehensive incident reports so that I can provide performance data to management and stakeholders
- **US15:** As a traffic supervisor, I want to review operator response times and actions so that I can identify training needs and best practices

## 2.2 Secondary Users

### 2.2.1 System Administrators

**Profile:** IT professionals responsible for system maintenance, security, and technical operations. Ensure system availability, manage user access, and maintain technical infrastructure.

**Technical Proficiency:** Expert - comprehensive knowledge of cloud systems, databases, and security protocols

**Work Environment:** IT operations centre with full system access and monitoring tools

**Primary Goals:** Maintain system uptime, ensure security compliance, optimise performance

#### User Stories:

- **US16:** As a system administrator, I want to monitor system health metrics so that I can proactively address performance issues before they impact operations
- **US17:** As a system administrator, I want to manage user accounts and permissions so that I can ensure appropriate access control and security compliance
- **US18:** As a system administrator, I want to configure camera feeds and detection parameters so that I can optimise system performance for different locations
- **US19:** As a system administrator, I want to access detailed system logs so that I can troubleshoot issues and maintain audit trails for compliance
- **US20:** As a system administrator, I want to perform system backups and recovery procedures so that I can ensure data protection and business continuity



### **2.2.2 Data Analysts**

**Profile:** Specialists responsible for analysing traffic patterns, incident trends, and system effectiveness. Generate insights for operational improvements and strategic planning.

**Technical Proficiency:** High - advanced data analysis skills with experience in visualisation tools

**Work Environment:** Analyse workstations with access to historical data and reporting tools

**Primary Goals:** Extract actionable insights, identify improvement opportunities, measure system effectiveness

#### **User Stories:**

- **US21:** As a data analyst, I want to export incident data in multiple formats (CSV, JSON) so that I can perform detailed analysis using specialised tools
- **US22:** As a data analyst, I want to query historical incidents by multiple criteria so that I can identify patterns and correlations
- **US23:** As a data analyst, I want to visualise incident frequency by time, location, and type so that I can create comprehensive trend reports
- **US24:** As a data analyst, I want to measure detection accuracy against validated incidents so that I can assess and improve AI model performance
- **US25:** As a data analyst, I want to correlate incident data with external factors so that I can provide recommendations for preventive measures

## **2.3 Tertiary Users**

### **2.3.1 Maintenance Personnel**

Profile: Technical staff responsible for camera equipment maintenance and positioning. Require system feedback to optimise camera placement and ensure optimal detection coverage.

#### **User Stories:**

- **US26:** As maintenance personnel, I want to receive camera status alerts so that I can prioritise equipment repairs and maintenance schedules
- **US27:** As maintenance personnel, I want to access camera performance metrics so that I can identify cameras requiring adjustment or replacement

### **3.3.2 Training Coordinators**

Profile: Personnel responsible for operator training and system onboarding. Need a comprehensive understanding of system capabilities and user workflows.

#### **User Stories:**

- **US28:** As a training coordinator, I want to access system documentation and user guides so that I can develop effective training programs
- **US29:** As a training coordinator, I want to use historical incident scenarios so that I can create realistic training exercises for operators



## **2.4 User Interaction Patterns**


### **2.4.1 High-Frequency Interactions (Multiple times per shift)**

- Monitoring live camera feeds and system alerts
- Acknowledging and managing incident notifications
- Updating incident status and adding operational notes
- Coordinating with emergency response teams

### **2.4.2 Medium-Frequency Interactions (Daily/Weekly)**

- Generating incident summary reports
- Reviewing system performance metrics
- Configuring alert parameters and thresholds
- Analysing incident patterns and trends

### **2.4.3 Low-Frequency Interactions (Monthly/As needed)**

- System administration and user management
  - Comprehensive data analysis and strategic reporting
  - System configuration updates and optimisation
  - Training and documentation updates
- 

## **2.5 User Success Criteria**

**Traffic Operators:** Reduce incident notification time from 5-15 minutes to under 2 minutes, increase incident detection rate by 40%, improve operator confidence in system reliability

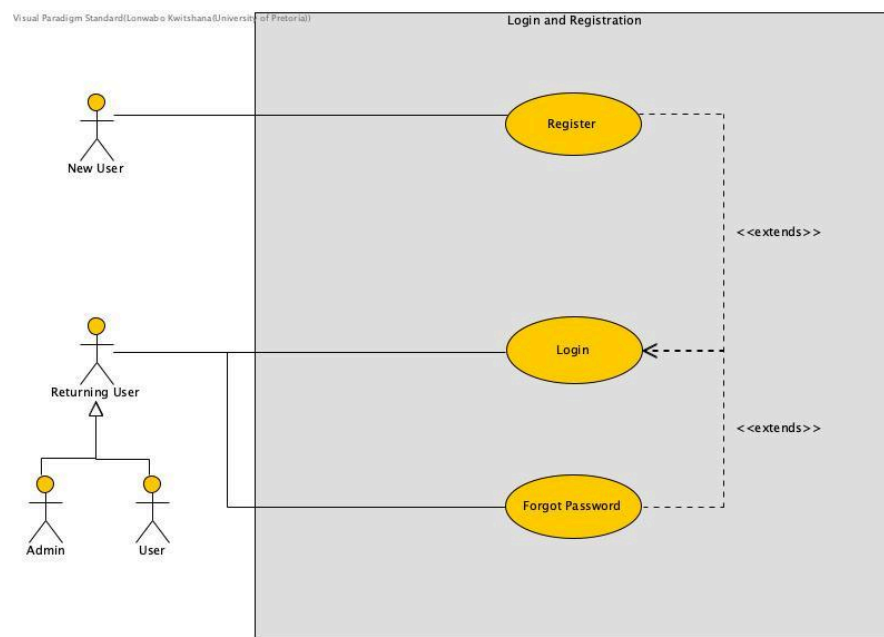
**Emergency Coordinators:** Achieve 95% appropriate resource allocation based on AI severity classification, reduce false dispatches by 60%, improve response team preparation through enhanced incident information

**Supervisors:** Gain comprehensive operational visibility, reduce manual reporting time by 70%, and identify actionable improvement opportunities through data-driven insights

**System Administrators:** Maintain 99.5% system uptime, achieve full POPI Act compliance, ensure zero security incidents related to data access or handling

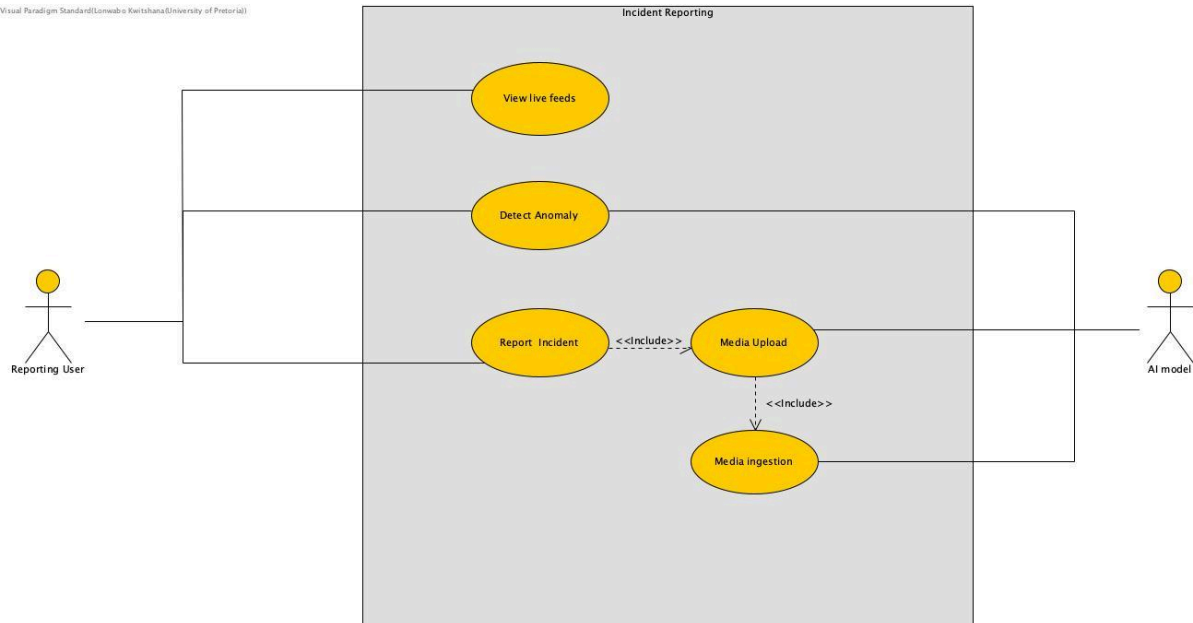
### 3. Updated Use Case Diagrams

#### Use Case 1: Login and Registration



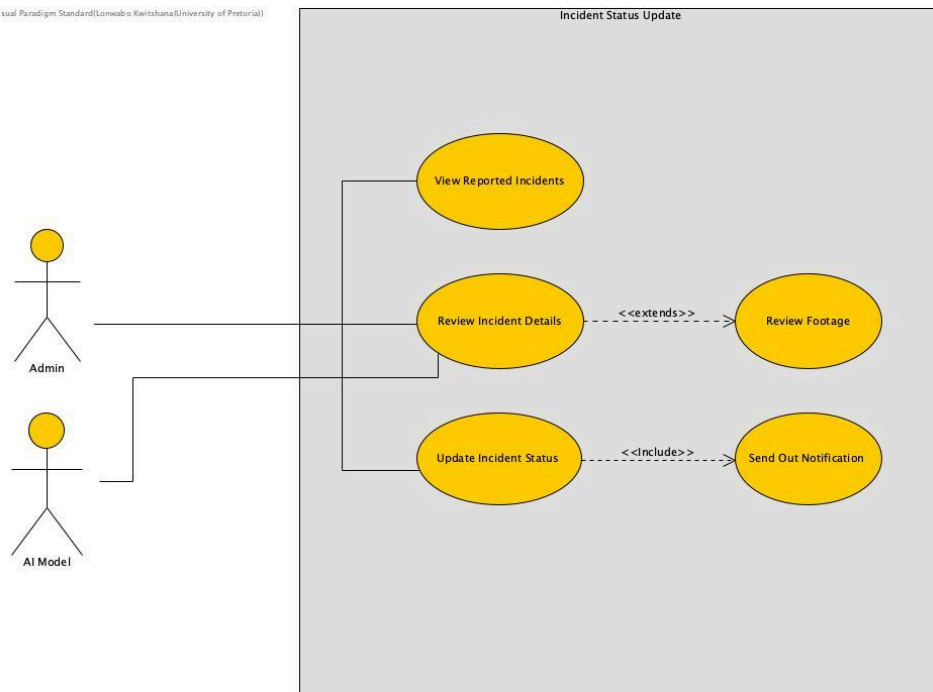
## Use Case 2: Incident Reporting

Visual Paradigm Standard(Lonwabo Kwitshana/University of Pretoria)



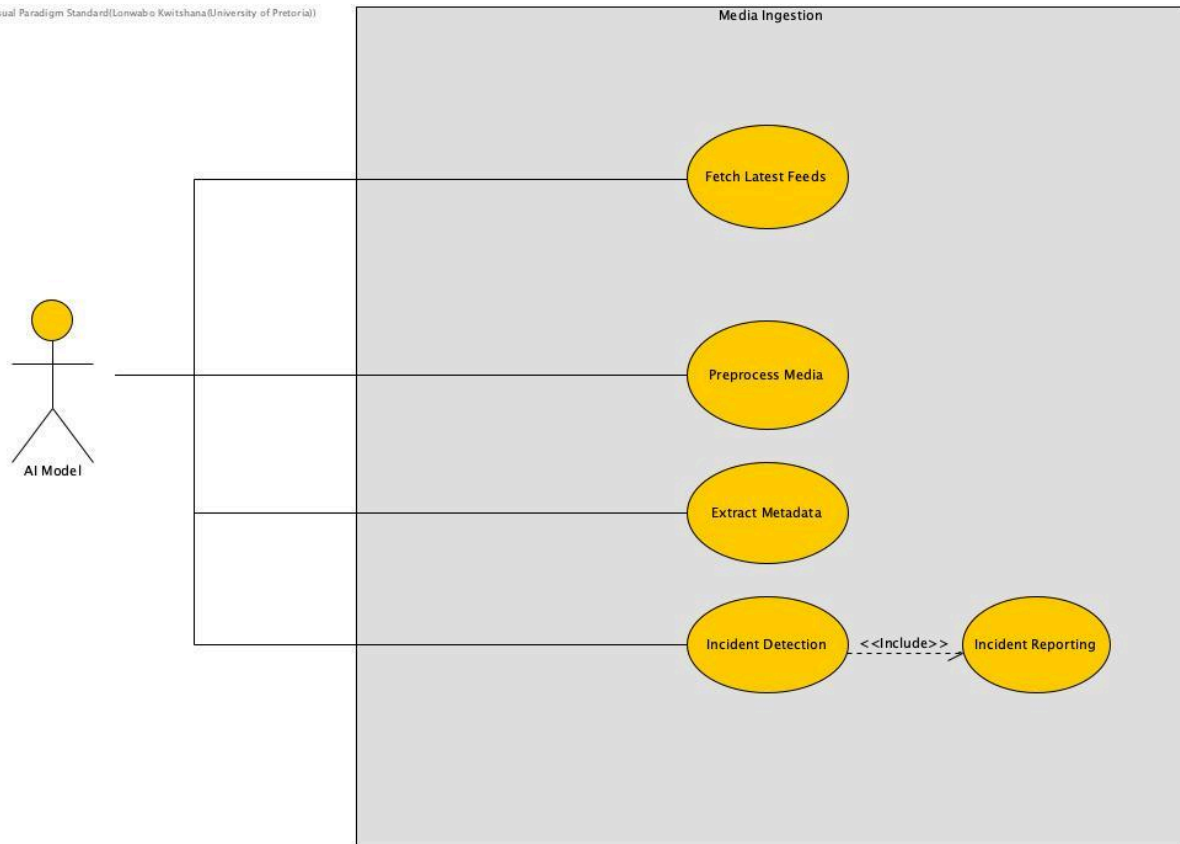
## Use Case 3: Incident Status Update

Visual Paradigm Standard(Lonwabo Kwitshana/University of Pretoria)



## Use Case 4: Media Ingestion

Visual Paradigm Standard(Lonwabo Kwitshana(University of Pretoria))



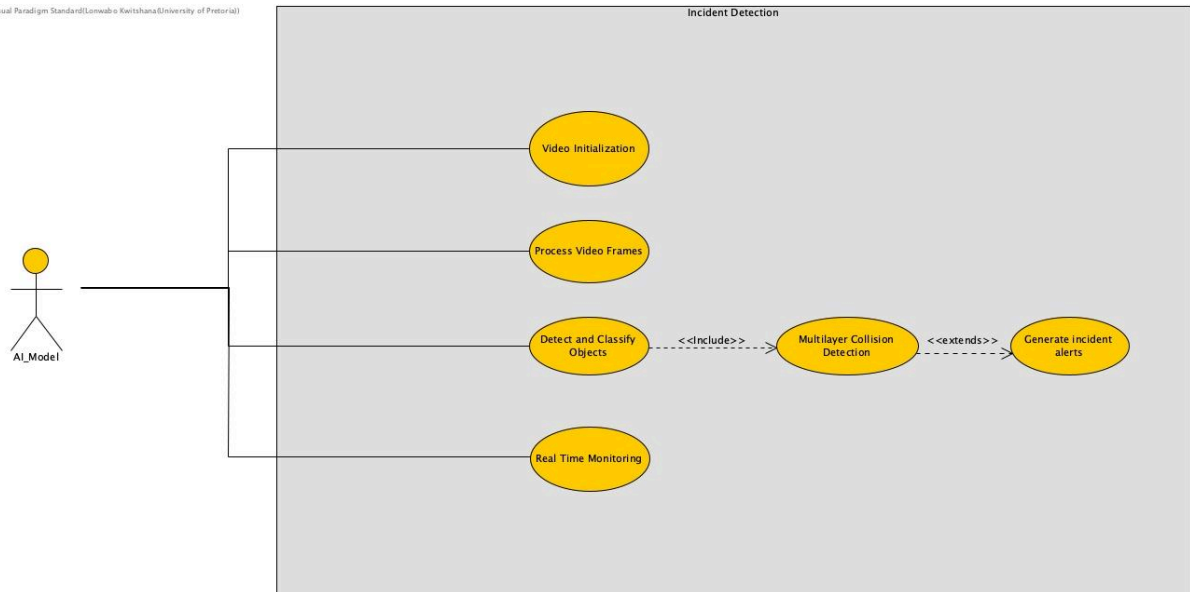
## Use Case 5: Real Time Incident Alerts

Visual Paradigm Standard(Lonwabo Kwitshana(University of Pretoria))



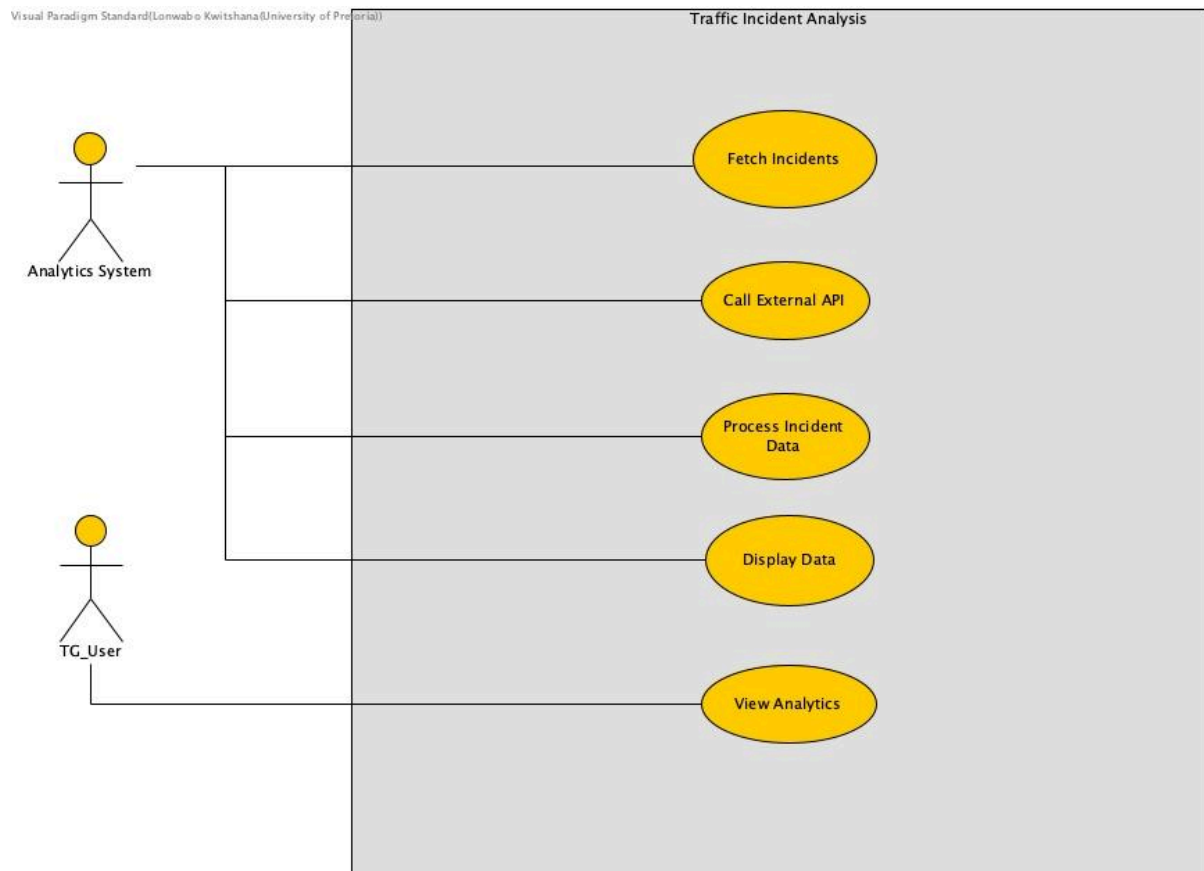
## Use Case 6: AI Incident Detection

Visual Paradigm Standard (Lomwabo Kwitshana/University of Pretoria)

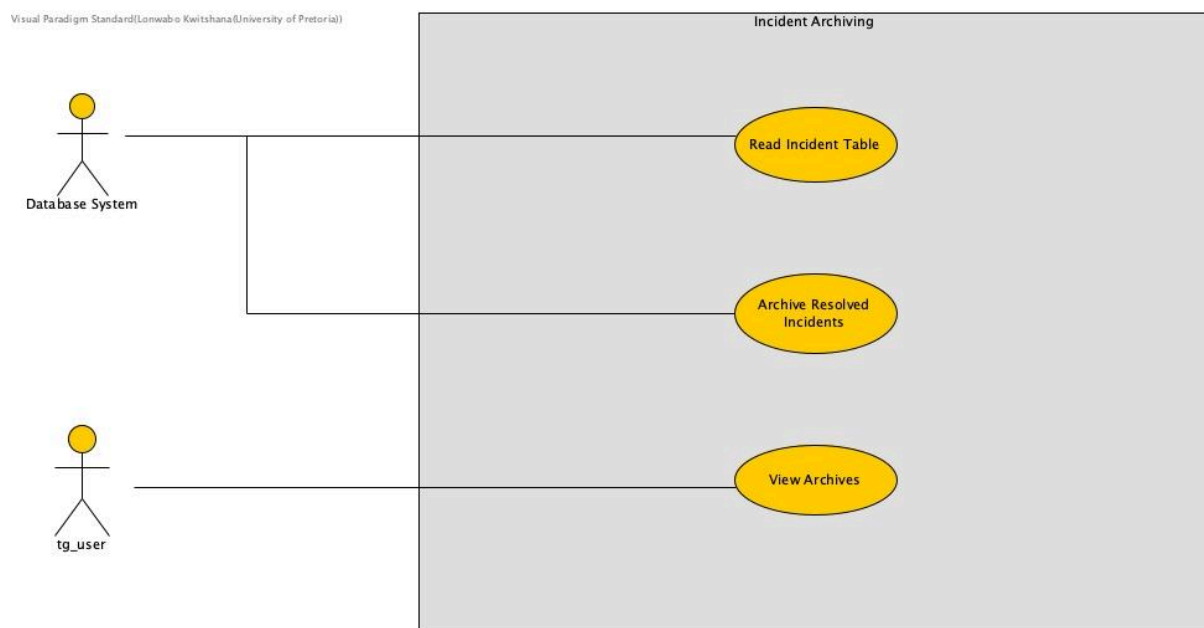




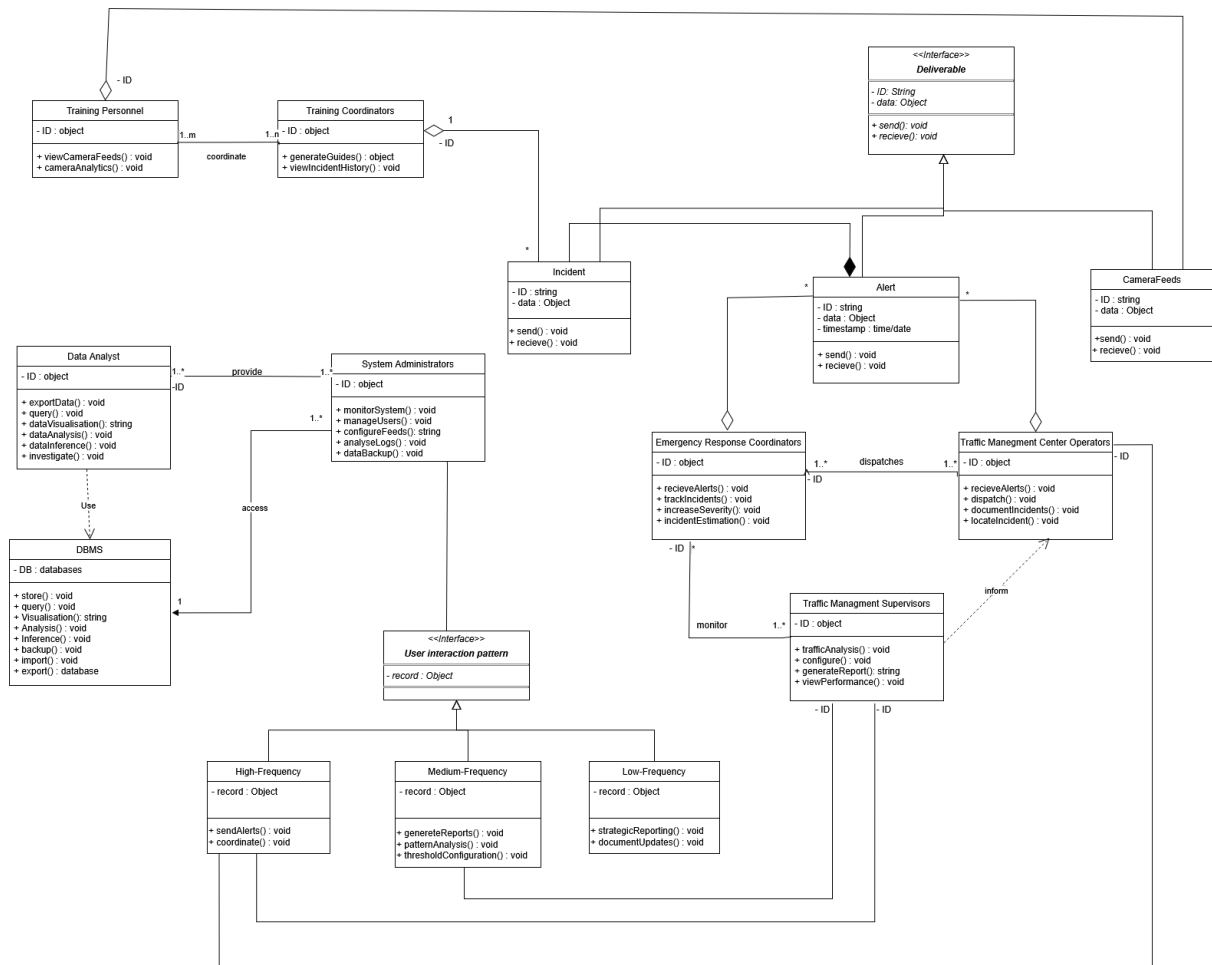
## Use Case 7: Traffic Analysis



## Use Case 8: Incident Archiving



## 4. Updated Domain Model



## 5. Service Contracts

This section defines the service contracts that govern communication between major components in the Traffic Guardian system. These contracts specify API interfaces, data exchange formats, communication protocols, and reliability mechanisms to ensure robust integration between microservices while maintaining loose coupling and enabling independent development.

### 5.1 Contract Architecture Overview

The Traffic Guardian system employs a contract-first approach with standardized REST APIs managed through an API Gateway pattern. All services communicate using JSON data format over HTTPS, with JWT-based authentication and comprehensive error handling. The base API endpoint <https://api.trafficguardian.aws/v1> serves as the single entry point, implementing rate limiting, request routing, and protocol mediation to achieve the required  $\leq 200\text{ms}$  API latency and 99% integration success rate.

### 5.2 Authentication and Authorization Contract

The Authentication Service establishes security contracts for system access through a centralized JWT token management system. User credentials are validated against secure endpoints, returning time-limited tokens that authorize subsequent API calls. The contract specifies a 5-second timeout for authentication requests with no automatic retry policy to prevent brute-force attacks. Token expiration is set to 24 hours, requiring periodic renewal to maintain session security.

#### **Key Contract Elements:**

- **Protocol:** HTTPS REST with JWT Bearer authentication
- **Data Format:** JSON request/response structures
- **Error Handling:** 401 Unauthorized for invalid credentials, 429 for rate limiting
- **Security:** AES-256 encryption for token generation and validation

### **5.3 Incident Management Service Contract**

The Incident Management Service represents the core business logic contract, handling traffic incident lifecycle from detection through resolution. This contract defines how incident data flows between the AI detection system, operator dashboard, and external emergency services.

**Primary Operations Contract:** The service exposes RESTful endpoints for incident creation, retrieval, status updates, and querying. Incident objects contain standardized fields including type classification (accident, congestion, hazard, emergency), severity levels (low, medium, high, critical), geolocation data, confidence scores, and temporal metadata. The contract guarantees 3-second response times for write operations and 2-second response times for read operations.

**Real-time Event Contract:** WebSocket connections enable live incident updates to connected dashboard clients. The contract specifies event-driven notifications for incident state changes, ensuring operators receive immediate alerts without polling. Connection management includes 30-second heartbeat intervals and automatic reconnection with exponential backoff to maintain reliability.

**Data Consistency Contract:** All incident operations maintain ACID compliance through PostgreSQL transactions, ensuring data integrity across concurrent access scenarios. The contract defines optimistic locking mechanisms to handle simultaneous updates from multiple operators.

### **5.4 Alert and Notification Service Contract**

The Alert Service contract governs multi-channel notification delivery through REST APIs, AWS SNS integration, and WebSocket broadcasts. This contract ensures critical incident information reaches appropriate personnel through their preferred communication channels.

**Delivery Contract Specifications:** Alert requests specify recipient lists, delivery channels (push notifications, dashboard), and priority levels that determine routing urgency. The contract allows 10-second timeout windows to accommodate external service dependencies while maintaining delivery confirmation tracking. AWS SNS integration provides reliable message queuing with automatic retry mechanisms for failed deliveries.

**Channel-Specific Contracts:** Each notification channel implements specific formatting contracts - email alerts include structured subject lines and HTML content, SMS notifications use condensed text format with incident reference numbers, and dashboard notifications provide rich media including location maps and camera feeds.

## 5.5 Camera Feed and Media Processing Contract

The Camera Feed Service contract manages video stream ingestion, frame analysis, and real-time processing coordination with the AI detection system. This contract balances performance requirements with resource constraints within AWS Free Tier limitations.

**Stream Management Contract:** Camera registration requires unique identifiers, location metadata, stream URLs, and technical specifications (resolution, frame rate). The contract supports various input formats (RTSP, HTTP streams) while standardizing internal processing pipelines. Health monitoring tracks camera uptime and frame delivery metrics to ensure reliable incident detection coverage.

**Frame Processing Contract:** Individual frame analysis accepts multipart form data containing binary image data (JPEG/PNG format, maximum 5MB) with associated timestamp and metadata. Processing responses include detected object classifications, confidence scores, bounding box coordinates, and incident flags. The contract specifies 15-second maximum processing time to balance accuracy with real-time requirements.

**AI Integration Contract:** The service coordinates with YOLO-based detection models through standardized inference contracts. Frame preprocessing, model execution, and post-processing steps follow defined interfaces that enable algorithm updates without service disruption.

## 5.6 Error Handling and Reliability Contracts

All service contracts implement standardized error handling mechanisms to ensure consistent behavior across system failures and degraded performance scenarios.

**Error Response Standards:** All services return structured error responses containing error codes, human-readable messages, timestamps, request identifiers for tracking, and retry guidance. HTTP status codes follow REST conventions with specific mappings for different error categories - 400 for client errors, 401 for authentication failures, 404 for missing resources, 422 for validation errors, 429 for rate limiting, and 500/503 for server issues.

**Timeout and Retry Policies:** Service-specific timeout values balance responsiveness with reliability - authentication services use 5-second timeouts with no retry, incident management allows 2-3 seconds with exponential backoff retry (3 attempts), alert delivery permits 10 seconds with retry mechanisms, and camera processing extends to 15 seconds given computational requirements.

**Circuit Breaker Implementation:** All external service calls implement circuit breaker patterns with 5-failure thresholds, 30-second timeout periods, and 10-second recovery intervals. This prevents cascade failures and maintains system stability during external service outages.

## **5.7 Data Format and Protocol Standards**

The system standardizes on JSON data interchange format for all API communications, ensuring consistent parsing and validation across services. Temporal data follows ISO 8601 formatting, geolocation uses decimal degree notation, and binary data employs Base64 encoding when transmitted through JSON APIs.

**Protocol Specifications:** REST APIs handle synchronous request-response patterns while WebSocket connections manage real-time event streams. All communications use TLS 1.3

## **5.8 Testing and Validation Framework**

Each service contract includes comprehensive testing specifications to ensure reliable integration between components and enable independent development workflows.

**Performance Validation:** Load testing scenarios verify contract adherence under the specified performance requirements: 50 concurrent users, 100 incident detections per minute,  $\leq 200\text{ms}$  API response times, and 2,592 requests per second throughput capacity.

## **6.Updated Functional Requirements**

### **R1: User Registration and Login**

#### **R1.1: Sign Up**

- The system shall allow new users to register with a username, email address and password.
- The system shall validate that the email is correctly formatted and ensure passwords are a minimum of 6 characters in length.
- The system shall display a “Sign up” link on the login page.

#### **R1.2: Log In**

- The system shall allow registered users to log in using their email and password.
- The system shall display a “Forgot?” link to reset a user’s password using a time-limited email token.

#### **R1.3: Log Out**

- The system shall allow users to log out and immediately invalidate their authentication token.

### **R2: Dashboard Overview and Metrics**

#### **R2.1: Key Metrics Display**

- The system shall display the number of active incidents requiring immediate attention.
- The system shall display the number of cameras that are online and the operational percentage.
- The system shall display the average response time for incidents in the last 24 hours.
- The system shall display today’s incident count and a comparison to yesterday’s total.

#### **R2.2: System Status & Time**

- The system shall display an “All Systems Operational” status when no critical errors exist.
- The system shall display the current local time in the dashboard header.

### **R2.3: Auto-Refresh**

- The system shall automatically refresh all dashboard metrics and incident information at regular intervals (e.g. every 60 seconds).

### **R2.4: Action Cards**

- The system shall display clickable “Quick Action” cards on the dashboard for:
  - Live Feed
  - Report Incident
  - Analytics
  - Archive
- Clicking a card shall navigate the user to the corresponding page.

## **R3: Live Video Feed Ingestion & Display**

### **R3.1: Feed Ingestion and Display**

- The system shall ingest live video streams from a configurable set of cameras (minimum of 6 feeds) and display them simultaneously.

## **R4: Automated Incident Detection, Classification and Alerting**

### **R4.1: Real-Time Analysis**

- The system shall continuously analyse incoming video feeds using deep learning models.

### **R4.2: Classification**

- The system shall classify detected incidents into predefined categories (Vehicle Accident, Vehicle Breakdown, Traffic Congestion, Road Debris, Weather Hazard, Construction Zone, Emergency Vehicle and Other).

### **R4.3: Alert Generation**

- The system shall generate a real-time alert card on the dashboard whenever a new incident is detected.

### **R4.4: Geolocation Mapping**

- The system shall map each detected incident to a precise GPS coordinate and store that metadata.



## **R5: Active Incidents Management**

### **R5.1: Incident Listing**

- The system shall list all active incidents and show the incident ID, date, type, location, camera, severity, status, and actions.

### **R5.2: Details View**

- The system shall allow users to click the view icon (eye) on any incident card to see full metadata (status, reported by, assigned to).

### **R5.3: Edit**

- The system shall allow users to click on the edit icon to make changes to an incident.

### **R5.4: Resolution**

- The system shall allow users to click on the dropdown to change an incident's status to "Resolved" and remove it from the active list.

## **R6: Incident Search & Filtering**

### **R6.1: Search**

- The system shall provide a search box to find incidents by ID, location text, or type.

### **R6.2: Filters**

- The system shall allow filtering the incidents list by status, severity, type, and date range (date from / date to).

### **R6.3: Error Handling**

- If the API request fails, the system shall display a clear error card ("Failed to load incidents. Please check your connection.").

## **R7: Manual Incident Reporting**

### **R7.1: Reporting Form**

- The system shall allow users to manually report a new incident by specifying the required fields: Incident Date, Camera ID, Location, Incident Type, Severity, Description and Reporter Name.
- The system shall allow users to input optional fields: Location Details (Latitude and Longitude), Weather Conditions, Traffic Impact, Injuries Reported and Reporter Contact Information.
- The system shall allow users to optionally input photos via drag and drop or through “browse files”.

### **R7.2: Confirmation**

- The system shall display a success notification and add the new incident to the active list in real time upon submission.

## **R8: Camera Status Monitoring**

### **R8.1: Camera List**

- The system shall list all configured cameras with ID, name, current status (Active/Offline), incident count and last-seen timestamp.

### **R8.2: Offline Alerts**

- The system shall detect when a camera goes offline and highlight it with a red indicator and “Offline” label.

## **R9: Incident Archive View**

### **R9.1: Archive Listing**

- The system shall provide an Archive page listing historical incidents in a table with the same columns as the active list.
- The system shall support the same search, filters, and pagination for archived incidents.



## **R10: Analytics Dashboard**

### **R10.1: Trend Charts**

- The system shall display interactive charts of incident counts over time, response-time distributions and category breakdowns
- The system shall allow users to select a custom date range to drive all analytics queries.

## **R11: Optional Features**

### **R11.1: Weather Correlation (Optional)**

- The system may ingest weather-station data and correlate incidents to current conditions.

### **R11.2: Field-Responder Mobile App (Optional)**

- The system may expose incident details via a lightweight mobile interface for on-site teams.

### **R11.3: Public Reporting Mobile App (Optional)**

- The system may provide a public mobile app for citizens to report new incidents.

### **R11.4: ANPR Integration (Optional)**

- The system may perform Automatic Number-Plate Recognition on video feeds and store plate data for vehicles of interest.



## 7. Quality Requirements

### Q1 Usability:

Aspect	Specification
Stimulus Source	Traffic management centre operators, emergency responders, and system administrators
Stimulus	The user needs to quickly identify, understand, and respond to traffic incidents through the system interface
Response	- Provide an intuitive dashboard with clear incident visualisation - Display actionable information with minimal cognitive load - Enable rapid incident acknowledgement and response coordination - Offer contextual help and guidance for complex operations - Support multiple user roles with appropriate interface customisation
Response Measure	<ul style="list-style-type: none"><li>• Usability Testing- Achieve a satisfaction score of at least 80% (4 out of 5) on usability surveys and questionnaires conducted.</li><li>• Users can recover from errors (e.g., incorrect input, navigation mistakes) within 10 seconds for 95% of cases, with contextual help available.</li><li>• Accessibility Compliance: WCAG 2.1 AA standard compliance</li></ul>
Environment	High-stress traffic management centre with multiple concurrent incidents, numerous alerts and AI detection which may misfire.
Artifact	Web-based dashboard, mobile-responsive interface, alert notification system, incident detail views, and system configuration panels

## Q2 Extensibility:

Aspect	Specification
Stimulus Source	Client / Developer / Product Owner
Stimulus	Request to add a new feature (e.g. ANPR integration, mobile reporting interface or advanced analytics module) to the deployed system
Response	- Register the new microservice with the API Gateway - Update service discovery and routing configuration - Add corresponding UI components or connectors - Execute automated unit and integration tests
Response Measure	- No existing services require code changes, aside from Gateway registration - Deployment incurs $\leq 5$ minutes of downtime (if any) - Regression test suite passes at 100% - Time and effort fit within agreed sprint scope
Environment	Already deployed Traffic Guardian environment (staging or production)
Artifact	- Source-code repository (service and UI modules) - CI/CD pipeline definitions - API Gateway configuration files

## Q3 Interoperability

Aspect	Specification
<b>Stimulus Source</b>	External Systems (e.g. Traffic Management Centres, Weather API providers, Third-party analytics tools)
<b>Stimulus</b>	Need to ingest real-time camera feeds and weather data, or push incident data into municipal dashboards or data lakes
<b>Response</b>	- Expose RESTful JSON APIs conforming to OpenAPI 3.0 - Support GeoJSON, CSV export, MQTT for TTL messaging - Provide OAuth 2.0 / JWT connectors - Document endpoints with Swagger UI (/docs)
<b>Response Measure</b>	- End-to-end integration tests pass against mock services - 0 OpenAPI linter errors - External API call latency $\leq 200$ ms (95th percentile) - $\geq 99\%$ integration success rate over 1,000 calls
<b>Environment</b>	Integration or staging environment with representative external endpoints
<b>Artifact</b>	- OpenAPI contract file (.yaml/.json) - Connector modules in the service layer - API Gateway routing and security policies

## Q4 Availability

Aspect	Specification
Stimulus Source	System failures, network outages, AWS service disruptions, hardware faults, and high traffic loads
Stimulus	Critical infrastructure component fails during active traffic monitoring or emergency incident response
Response	<ul style="list-style-type: none"><li>- Automatically detect service failures and initiate recovery procedures</li><li>- Maintain core incident detection capabilities during partial system outages</li><li>- Provide graceful degradation when some camera feeds are unavailable</li><li>- Ensure alert delivery through redundant channels</li><li>- Restore full service within acceptable time limits</li></ul>
Response Measure	<ul style="list-style-type: none"><li>- System uptime <math>\geq 99.5\%</math> (maximum 3.6 hours downtime per month)</li><li>- Mean Time To Failure (MTTF) <math>\geq 720</math> hours (30 days)</li><li>- Mean Time To Recovery (MTTR) <math>\leq 15</math> minutes for automatic recovery</li><li>- Alert delivery success rate <math>\geq 99.9\%</math> during normal and degraded operations</li><li>- Core video processing resumes within 30 seconds of service restart</li></ul>
Environment	24/7 production traffic monitoring environment with critical emergency response dependencies
Artifact	Microservices architecture, database clusters, load balancers, health monitoring systems, and alert delivery mechanisms

## Q5 Performance

Aspect	Specification
Stimulus Source	Front-end component and AI incident detection model
Stimulus	Continuous input and updates of real-time traffic and incident data from both the front-end and AI components.
Response	Efficient delivery of relevant data to UI components and timely generation of updated analytics for traffic and incident monitoring.
Response Measure	<ul style="list-style-type: none"><li>• Average of 500-550 ms for data responses</li><li>• 2592 req/seconds for data sorting and processing</li></ul>
Environment	Production deployment environment with standard user traffic and live data feeds from sensors and AI detection modules.
Artifact	Back-end data processing service, front-end UI modules, and AI incident detection pipeline.



## Q6 Scalability

Aspect	Specification
Stimulus Source	Client applications and an AI incident detection model
Stimulus	The number of connections of users through the client application and the different detections identified by the AI model
Response	The system dynamically adjusts resources to maintain consistent performance and availability and to prioritise high alert level incidents.
Response measure	<ul style="list-style-type: none"><li>• Supports up to 50 concurrent users</li><li>• Handles about 100 new incident detections</li></ul>
Environment	Will be cloud-based with containerised microservices
Artifact	Backend services, data processing components, AI pipeline, and orchestration layer

## 8. Technologies Requirements

Component	Proposed Tool/Service	Justification
Cloud Platform	AWS (Free Tier)	Scalable and secure cloud infrastructure that aligns with budgetary constraints.
Video Ingestion	OpenCV	OpenCV for seamless video stream ingestion and local processing.
Computer Vision	YOLO, OpenCV	YOLO for deep learning-based incident classification and OpenCV for frame processing.
Backend API	Node.js	Lightweight RESTful API for system communication and integration with the frontend.
Frontend Dashboard	React.js	An interactive, real-time web dashboard for traffic monitoring and reporting.
Database & Storage	PostgreSQL	For media storage, metadata and incident record storage.
Notification System	AWS SNS, WebSockets	Real-time notifications and an alert system to notify operators and field personnel.
Security	HTTPS, AWS IAM, AES Encryption	Encryption for data at rest and in transit, IAM for user access control, and AES encryption for sensitive data.
Incident Archiving	AWS S3 / Amazon RDS	S3 for raw data and RDS for structured archival with search capabilities.
CI/CD Pipeline	GitHub Actions	Automate testing, building, and deployment processes.

## 8.1 Technologies Overview

### 8.1.1 Cloud Platform – AWS (Free Tier)

As final-year Computer Science majors, we are deeply familiar with cloud computing paradigms and cloud-native architecture. **AWS Free Tier** provides us with an industry-grade infrastructure that supports scalability, reliability, and high availability, while keeping us within budget. Leveraging AWS enables us to deploy and test services in a production-like environment, gain practical DevOps experience, and ensure that our solution is easily extensible for future growth.

### 8.1.2 Video Ingestion – OpenCV

We will use **OpenCV** for initial frame processing, which is open source and completely free. **Amazon Kinesis Video Streams**, while powerful and natively integrated into AWS, is not included in the AWS Free Tier and may incur charges based on data throughput and retention. Therefore, we will use it **selectively**, primarily during demos or load testing, and otherwise rely on local simulation using OpenCV for development and testing phases.

### 8.1.3 Computer Vision – YOLO, OpenCV

To detect and classify traffic incidents, we will utilise **YOLO** for training and running deep learning models and **OpenCV** for image preprocessing and feature extraction. Our coursework and project experience have given us a solid foundation in AI and machine learning, and this component allows us to apply those skills in a real-world, high-impact context. YOLO's extensive library support and community ecosystem make it ideal for iterating quickly on model performance.

### 8.1.4 Backend API – Node.js

**PostgreSQL** will serve as our primary database for incident metadata. This relational database solution provides ACID compliance, complex querying capabilities, and mature tooling that's well-suited for our structured incident data and reporting requirements. PostgreSQL's JSON support also allows for flexible schema evolution when needed. These choices align with the cloud-native architecture principles we've studied while providing the reliability and query power of a traditional RDBMS.

### **8.1.5 Frontend Dashboard – React.js**

For the dashboard, we will develop a responsive and interactive web interface using **React.js**. This will allow traffic control operators to monitor live feeds, receive alerts, and view analytical reports in an intuitive manner. We've worked extensively with frontend technologies during our academic projects and are confident in our ability to build highly usable and aesthetically pleasing UIs.

### **8.1.6 Database & Storage – PostgreSQL**

**PostgreSQL** will serve as our primary database for incident metadata. This relational database solution provides ACID compliance, complex querying capabilities, and mature tooling that's well-suited for our structured incident data and reporting requirements. PostgreSQL's JSON support also allows for flexible schema evolution when needed. These choices align with the cloud-native architecture principles we've studied while providing the reliability and query power of a traditional RDBMS.

### **8.1.7 Notification System – AWS SNS, WebSockets**

To ensure rapid communication with traffic management teams, we'll integrate **AWS SNS** for alert broadcasting and **WebSockets** for real-time updates in the web dashboard. These tools will help us deliver near-instant feedback when an incident is detected, enhancing situational awareness. Real-time systems are a key part of modern distributed applications, and we're excited to implement these concepts in practice.

### **8.1.8 Security – HTTPS, AWS IAM, AES Encryption**

Security is non-negotiable, especially when dealing with sensitive infrastructure and possibly identifiable vehicle data. We'll enforce **HTTPS** for all communications, use **AES encryption** for stored data, and rely on **AWS IAM** for managing access control across our services. This ensures compliance with the POPI Act and best practices, while giving us a real-world understanding of secure system design.

### **8.1.9 Incident Archiving – AWS S3 / Amazon RDS**

To maintain a searchable, long-term archive of detected incidents, we'll utilise **AWS S3** for storing raw data and **Amazon RDS** for structured querying and analytics. This dual-storage strategy balances performance with cost-efficiency and enables us to deliver meaningful insights through the dashboard and reporting tools. Working with both structured and unstructured data is a core competency we've honed through our coursework.

### **8.1.10 CI/CD Pipeline – GitHub Actions**

We will adopt **GitHub Actions** to automate our testing, build, and deployment workflows, reinforcing a DevOps-oriented development culture. This approach enhances our development efficiency, minimises human error, and ensures that new features or fixes are delivered to production swiftly. As students preparing to enter the industry, this experience is invaluable in applying agile methodologies and continuous delivery strategies

## **9. Agile Methodology and Proposed Roadmap**

We adopted **Agile methodology** with 2-week sprints to ensure rapid iteration and continuous client feedback. With **AWS** as our industry partner, Agile allows us to demonstrate working features early, adapt to changing requirements, and deliver incremental value at each demo. This approach is essential for a complex AI system where requirements evolve as we test and refine our computer vision models.

Our implementation strategy for **Traffic Guardian** follows a phased approach to ensure steady progress, early validation of core concepts, and timely delivery of a production-ready system. Each phase builds upon previous accomplishments while maintaining focus on our core objectives of improving incident detection and response times.

### **9.1 Preparation Phase**

Objective: Initialise project infrastructure and planning.

Deliverables: Repository structure, **AWS environment**, project board, initial wireframes, and dataset preparation plan.

#### **Tasks:**

- Backend: Set up GitHub repository, CI/CD pipeline, AWS environment configuration, and project board setup.
- AI/ML: Research appropriate computer vision models, prepare training datasets.
- Frontend: Set up Figma workspace, draft initial dashboard wireframes, and user flows.

### **9.2 Basic Video Processing and Incident Detection**

Objective: Implement a video ingestion pipeline, basic object detection, and an initial dashboard prototype.

#### **Sprint 1 :**

- Backend: Develop a video ingestion pipeline using **OpenCV**, and create basic API endpoints.
- AI/ML: Implement an initial object detection model for vehicle identification.
- Frontend: Develop dashboard prototype with video display capability.

#### **Sprint 2 :**

- Backend: Set up **postgres** for incident storage, enhance API functionality.
- AI/ML: Expand detection capabilities for accidents and road obstructions.
- Frontend: Add incident display and notification components, conduct first usability test.

### **Demo 1:**

- Demonstrate a basic video processing pipeline and, object detection capabilities.
- Present the dashboard prototype with initial incident visualisation.

## **9.3 Enhanced Detection and Real-time Notification**

Objective: Improve incident classification accuracy, implement real-time alerts, and refine dashboard functionality.

### **Sprint 3:**

- Backend: Implement a notification system using **AWS SNS**, and enhance API scalability.
- AI/ML: Train models for congestion detection, performance optimisation.
- Frontend: Develop real-time alert interface, incident management views.

### **Sprint 4:**

- Backend: Implement incident classification and prioritisation logic.
- AI/ML: Add road hazard detection, model performance metrics.
- Frontend: Build analytics dashboard, integrate feedback from usability tests.

### **Sprint 5:**

- Backend/AI/ML/Frontend: Integration testing, bug fixes.

### **Demo 2:**

- Showcase improved detection accuracy across multiple incident types.
- Demonstrate a real-time notification system and enhanced dashboard functionality.

## **9.4 Geolocation Integration and Advanced Analytics**

Objective: Implement geospatial mapping, historical analysis, and complete system integration.

### **Sprint 6:**

- Backend: Develop geolocation services and a historical data API.
- AI/ML: Implement pattern recognition for recurring incidents.
- Frontend: Add interactive mapping components with incident overlays.

### **Sprint 7:**

- Backend: Build an advanced analytics engine and, reporting API.
- AI/ML: Optimise models for performance and accuracy.
- Frontend: Develop trend analysis views, interactive reporting.

### **Sprint 8:**

- Backend/AI/ML/Frontend: Full system integration, comprehensive testing.

### **Demo 3 :**

- Present geospatial incident mapping with historical data visualisation.
- Demonstrate pattern recognition capabilities and predictive analytics.

## **9.5 Security Hardening and Production Readiness**

Objective: Security implementation, performance optimisation, documentation, and final deployment.

### **Sprint 9:**

- Backend: Security hardening, POPI compliance implementation.
- AI/ML: Final model tuning, deployment optimisation.
- Frontend: Security enhancements, accessibility improvements.

### **Sprint 10:**

- Backend: Performance optimisation, stress testing.
- AI/ML: Model versioning, fallback mechanisms.
- Frontend: Final UI refinements based on usability feedback.

### **Sprint 11:**

- Backend/AI/ML/Frontend: Documentation completion, final testing.
- Team: Prepare user guides, API documentation, and training materials.

### **Demo 4 :**

- Present production-ready system with complete functionality.
- Demonstrate security features, performance metrics, and comprehensive documentation