# Software Requirements Specification
# for
# Safe-Driver

**Prepared by: -**

CodeCrafters Team

**Team Members:**

D. S. Sandeepa 221444563

R. U. Gonalagoda 621429318

P. S. Ekanayaka 621429276

K. J. L. Perera 621429815

T. P. Denagamage 321429305

**SEPTEMBER 2025**

# Table of Contents

# List of Tables



# List of Figures

## Document Revisions

| Date | Version | Description | Author |
|---|---|---|---|
| 14/07/2025 | V1 | Initial SRS documentation | T. P. Denagamage |
| 17/07/2025 | V2 | Prepared for supervisor approval | P. S. Ekanayaka |
| 23/07/2025 | V3 | Modified changes I | P. S. Ekanayaka<br>T. P. Denagamage |
| 15/09/2025 | V4 | Modified changes II | D. S. Sandeepa |

*Table 1 - document version history*

## Document Approval

Code Crafters Software and Safe Driver Team have reviewed this document and hereby agree that the contents herein are accurate. Any changes to this document must be communicated in writing and signed off by both parties.

| Signature | Signature |
|---|---|
| Date: | Date: |
| Name: | Name: |
| Customer: | |

# 1 Introduction

## 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to comprehensively outline the requirements for the development and deployment of the Safe-Driver system. Safe-Driver is a sophisticated, AI-powered solution intended to enhance the safety of public transportation in Sri Lanka by continuously monitoring the behavior of drivers in real-time. This document serves as a foundational guide for stakeholders, including system developers, testers, project managers, and clients, ensuring that everyone involved has a clear understanding of the system's objectives, features, and constraints.

It sets the stage for a structured development process by detailing both the functional and non-functional requirements necessary to achieve the desired functionality and performance standards. Ultimately, this document aims to prevent miscommunications during the project lifecycle and to provide a formal basis for design, development, verification, and validation of the Safe-Driver system.

## 1.2 Company Overview

CodeCrafters is an interdisciplinary student team comprised of individuals with strong expertise in artificial intelligence, embedded systems, software engineering, and mobile application development. Our mission is to address real-world problems using innovative technological solutions.

The Safe-Driver project exemplifies our commitment to building systems that have a meaningful impact on society. While CodeCrafters operates within an academic environment, our practices mirror those of professional software development teams, with structured workflows, documentation, and rigorous testing methodologies.

## 1.3 Project Overview

Safe-Driver is implemented as a comprehensive, modular, and highly scalable system that leverages advanced computer vision technologies and artificial intelligence to provide real-time driver monitoring capabilities specifically tailored for public bus operations. The project encompasses multiple integrated components working in coordination to deliver a complete safety monitoring and intervention solution.

The system architecture includes an embedded hardware platform mounted discretely within the vehicle interior to capture and analyze driver behavior in real-time, a robust cloud-based backend infrastructure for centralized data storage, analytics processing, and fleet-wide monitoring coordination, a cross-platform mobile application providing passengers with transparency and real-time safety information, and a comprehensive web-based dashboard enabling transport authorities to monitor compliance, review incident reports, and make data-driven policy decisions.

The primary objective focuses on detecting early warning signs of driver fatigue and distractions, implementing immediate corrective interventions through graduated alert mechanisms. GPS integration enables the system to provide context-aware warnings when approaching known hazardous areas or accident-prone zones, while real-time communication capabilities with transport authorities and safety updates to passengers promote a holistic, community-based approach to public transportation safety.

The system design prioritizes scalability and cost-effectiveness, enabling deployment across Sri Lanka's extensive public transport fleet while maintaining affordability for implementation in developing country contexts. The modular architecture supports phased deployment and allows for future enhancements and feature additions based on operational experience and evolving safety requirements.

## 1.4 Scope

The scope of this SRS document encompasses all technical, functional, and operational aspects of the Safe-Driver system, providing comprehensive specifications for successful system development, deployment, and maintenance. The document covers embedded driver monitoring subsystems including camera hardware, processing capabilities, and real-time analysis algorithms, AI and machine learning models for behavior detection, classification, and prediction, data analytics pipelines for pattern recognition, performance assessment, and predictive modeling, user interfaces including mobile applications, web dashboards, and embedded system displays, alert and communication mechanisms including audio, visual, and network-based notifications, and integration specifications with cloud services, GPS systems, and external communication networks.

The document defines detailed development and deployment environments, comprehensive security requirements protecting sensitive data while enabling appropriate access for emergency response, system performance benchmarks ensuring real-time operation under varying conditions, and detailed user roles and privileges supporting different stakeholder requirements including drivers, passengers, administrators, and transport authorities.

This SRS explicitly excludes vehicle automation technologies such as automatic braking systems, autonomous steering mechanisms, or engine control interventions. The focus remains on monitoring, detection, alerting, and communication functions that support driver decision-making rather than replacing driver control of the vehicle.

## 1.5 Assumptions

Several critical assumptions underpin the development of these requirements specifications, reflecting the operational environment and infrastructure context for Safe Driver deployment:

- Power Supply Reliability: Public buses targeted for Safe-Driver installation maintain consistent and reliable electrical power supply sufficient to operate embedded hardware continuously during vehicle operation, including adequate voltage stability and current capacity for processing equipment, camera systems, and communication modules.

- Driver Cooperation: Professional bus drivers will not attempt to deliberately tamper with, obstruct, or disable camera systems or other monitoring equipment. This is supported by the integration of monitoring into professional driver responsibilities and regulatory compliance frameworks.

- Passenger Connectivity and Adoption: Reasonable internet connectivity is available on user devices for receiving real-time safety updates, with system design including offline capabilities and store-and-forward mechanisms for intermittent connectivity. Basic smartphone adoption among the passenger population is sufficient to make transparency features meaningful and effective.

- System Integration: Centralized transport authority databases and communication systems exist to which Safe-Driver can escalate notifications and submit automated compliance reports.

- Environmental Conditions: Vehicles operate within Sri Lanka's tropical climate, requiring system design to accommodate high humidity, temperature extremes, monsoon conditions, and other factors affecting electronic equipment performance and reliability.

## 1.6 Definitions, Acronyms and Terminology

To ensure clarity, consistency, and precise understanding across all stakeholders throughout the project development and deployment process, the following comprehensive list defines technical terms, acronyms, and specialized terminology used throughout this document:

| Term | Definition |
|---|---|
| **EAR (Eye Aspect Ratio)** | A mathematical representation calculated from facial landmarks that quantify the degree of eye openness. Used as a primary metric for detecting drowsiness based on changes in normal blinking behavior patterns. |
| **PERCLOS** | Percentage of Eyelid Closure over a specified time, representing the percentage of time during a defined interval that the driver's eyes remain closed longer than normal blinks. Used as a standardized fatigue measurement metric. |
| **GPS (Global Positioning System)** | Satellite-based navigation system provides accurate real-time location determination, enabling the Safe-Driver system to correlate driver behavior with geographic hazard zones and accident-prone areas. |
| **CNN (Convolutional Neural Network)** | A specialized class of deep learning neural networks particularly effective for image and video processing tasks. Used in Safe-Driver for identifying driver distraction behaviors including mobile phone usage and attention deviation. |
| **Firebase** | Google's comprehensive cloud-based platform provides real-time database services, user authentication, cloud messaging, and analytics capabilities. Serves as the backend infrastructure for Safe-Driver's cloud components. |

| | |
|---|---|
| **TensorFlow Lite** | Google's optimized machine learning framework specifically designed for deployment on mobile and embedded devices with limited computational resources. Enables sophisticated AI model execution on Raspberry Pi hardware. |
| **MediaPipe** | Google's open-source framework for building multimodal applied ML pipelines, providing robust facial landmark detection and tracking capabilities optimized for real-time performance. |
| **OpenCV** | Open-Source Computer Vision Library providing comprehensive tools for image processing, computer vision algorithms, and real-time visual analysis. |
| **Raspberry Pi 4** | Single-board computer providing the embedded processing platform for Safe-Driver's real-time monitoring capabilities, chosen for optimal balance of performance, cost-effectiveness, and power efficiency. |
| **IoT (Internet of Things)** | Network of interconnected embedded devices capable of collecting, processing, and transmitting data. Safe-Driver operates as an IoT solution within the broader transportation infrastructure. |
| **Edge Computing** | Distributed computing paradigm that processes data near the location where it is generated rather than relying entirely on centralized cloud processing. Critical for Safe-Driver's real-time safety response requirements. |

*Table 2 - acronyms and terminology*

## 1.7 Summary

The Safe-Driver system is designed to combat the high incidence of road traffic accidents caused by human errors, particularly those related to driver fatigue and distractions such as mobile phone use. The system integrates advanced AI techniques with real-time embedded hardware to identify and respond to signs of driver inattentiveness. By using a camera-based eye tracking mechanism, combined with deep learning algorithms, the system can monitor the driver's facial features, detect signs of drowsiness or distraction, and respond with immediate alerts.

If the behavior persists, the system escalates the alert to relevant authorities via SMS and cloud notifications. Moreover, the system includes GPS-based hazard mapping and a mobile application for passengers, ensuring transparency and increased accountability. Safe-Driver is expected to revolutionize public transport safety in Sri Lanka by enabling proactive intervention and facilitating a data-driven approach to transport management.

# 2 Project Scope and Impact

## 2.1 Introduction

This chapter outlines the scope and impact of the Safe-Driver system, a comprehensive solution designed to enhance safety in Sri Lankan public transport through advanced driver monitoring and analytics. It details the system's inclusions, such as real-time driver monitoring, integrated alert mechanisms, GPS-based hazard mapping, cloud infrastructure, cross-platform mobile applications, and authority dashboards. Additionally, it clarifies exclusions to define the boundaries of the initial release and addresses the system's interactions with external infrastructure. By delineating the scope and impact, this chapter provides a clear understanding of Safe-Driver's capabilities, limitations, and its influence on and dependence upon related systems, ensuring alignment with project objectives and stakeholder expectations.

## 2.2 Scope Inclusions

The Safe-Driver system delivers a comprehensive solution for improving safety in Sri Lankan public transport through advanced driver monitoring and analytics.

**Real-Time Driver Monitoring Capabilities:**

Utilizes computer vision with camera-based monitoring, including facial landmark detection (MediaPipe), Eye Aspect Ratio (EAR) analysis with temporal filtering, head pose estimation, blink pattern recognition, and behavioral classification to detect distraction, fatigue, and unsafe actions like phone use or eating.

**Integrated Alert and Escalation System:**

A four-level alert mechanism responds based on behavior severity. Level 1–3 alerts escalate from audio and visual cues to authority notifications, while Level 4 includes GPS-enabled emergency response for critical incidents.

**GPS-Based Hazard Mapping:**

Combines GPS tracking with hazard databases to provide risk assessments using traffic, weather, and historical data. Geofencing triggers alerts in high-risk zones if the driver is inattentive.

**Cloud Infrastructure and Analytics:**

Uses Firebase and Cloud Functions for real-time data sync, reporting, and ML model updates. Security is ensured through end-to-end encryption and secure authentication.

**Cross-Platform Mobile App:**

Built with Flutter, the app offers real-time safety dashboards, historical safety data, passenger feedback, emergency contact access, and supports Sinhala, Tamil, and English.

**Authority Dashboard and Reporting:**

Provides transport authorities with live fleet monitoring, automated safety summaries, detailed incident reports, comparative analytics across drivers and routes, and regulation compliance tracking.

## 2.3 Exclusions

Certain features are excluded in the initial release due to technical or regulatory constraints.

**Physical Vehicle Control:**

Does not include automated braking, steering, or throttle control. The system focuses on monitoring and alerting, leaving full control with the driver.

**Real-Time Video Streaming:**

To protect privacy and manage bandwidth, live video streaming is excluded. Video is processed locally, and only analytics are transmitted.

**Fleet Management Integration:**

No integration with external ticketing, maintenance, or fleet systems in the initial phase. APIs may be added later for extended compatibility.

**Advanced Weather and Traffic Integration:**

Current scope includes basic GPS-based hazard detection but excludes full integration with live weather or traffic systems.

**Driver Training Modules:**

While the system supports report-based training insights, it does not include interactive training or certification content.

## 2.4 Impact on Other Systems

### 2.4.1 Affected by Other Systems

Safe-Driver's performance depends on reliable external infrastructure.

- **GPS Systems:** Required for accurate hazard and location mapping; affected by signal strength and atmospheric interference.
- **Internet Connectivity:** Cloud functions and authority notifications depend on stable telecom networks, which vary in rural areas. Offline modes mitigate issues.
- **Vehicle Electrical Systems:** System stability requires consistent vehicle power; hardware includes conditioning and backup support.
- **Transport Authority Systems:** Integration with government databases and platforms is essential for alert escalation and reporting; changes in protocols may affect compatibility.

### 2.4.2 Effects on Other Systems

Safe-Driver enhances external systems with reliable safety data.

- **Regulatory & Authority Systems:** Gains real-time driver safety data, automated alerts, compliance reports, and analytics to support enforcement and planning.
- **Public Transport Planning:** Provides route risk insights, driver performance metrics, passenger safety feedback, and data to guide infrastructure improvements and policy updates.

# 3 Functional Requirements

## 3.1 Introduction

This chapter outlines the functional requirements of the Safe Driver Monitoring System, specifying what the system must do to achieve its goals. It focuses on user interactions, components, and interfaces for safe and efficient operation in vehicles.

The Safe Driver enhances road safety by monitoring driver behavior for risks like drowsiness or distractions, using AI and sensors, while supporting administration, data management, user roles, and reporting.

Derived from stakeholder needs, standards, and best practices, these requirements guide design, implementation, and verification to ensure timely alerts, data integrity, and compliance. Details follow in subsequent sections.

## 3.2 Driver Monitoring

The system must be capable of continuously monitoring the driver's facial features using an embedded camera. The key features include detecting drowsiness by analyzing the Eye Aspect Ratio (EAR), calculating blink frequency, and identifying head pose deviations.

To detect distractions, TensorFlow Lite-based CNN models are utilized. These models are optimized for performance on embedded systems, ensuring real-time responsiveness in scenarios such as mobile phone usage or the driver looking away from the road. If any unsafe behaviors are detected, the system will trigger alerts within 1 second.

Additionally, the system incorporates yawning detection and smoking detection to enhance overall driver monitoring. These behaviors are critical indicators of fatigue or distraction and are processed alongside other facial cues to ensure comprehensive safety surveillance.

### 3.3 System Administration

Administrators need access to a secure web-based dashboard to configure system thresholds, view performance reports, and manage access permissions. They should be able to update device firmware remotely and control notification preferences. The admin panel will also allow viewing logs and updating location-specific hazard data.

**Authentication and Authorization:** Multi-factor authentication requirements for administrative access ensure system security, while secure session management prevents unauthorized access. The system maintains comprehensive audit logging of all administrative actions, user access patterns, and configuration changes to support security monitoring and compliance requirements.

### 3.4 Data Archival and Retention

The system should store event-driven data including timestamps, GPS coordinates, driver identity, and detected behaviors. Incident logs should be retained for a while and summarized safety compliance reports can be archived indefinitely. The database should support querying, filtering, and exporting records for auditing or further analysis.

### 3.5 User Profiles, Roles and Privileges

Different types of users will interact with the system, each requiring specific access permissions. Drivers will receive alerts and system status feedback. Administrators will have full access to system settings, logs, and analytics.

Transport authority officials will receive notification summaries and have read-only access to incident histories. Passengers will use the mobile app to receive real-time safety updates but will not have access to sensitive system logs or analytics.

### 3.6 Reporting Requirements

The system should generate daily, weekly, and monthly reports that detail driver behavior, incident frequencies, and system performance metrics. Reports must include visualizations and trend graphs. Authorities should also receive automated notifications when threshold violations occur. Incident reports should be downloadable in PDF and Excel formats for further review and archiving.

# 4 Non-Functional Requirements

## 4.1 Introduction

This chapter outlines the non-functional requirements for the Safe-Driver system, focusing on attributes like performance, reliability, security, usability, and compatibility. These define how the system operates efficiently and securely in real-time vehicular environments, integrating hardware, apps, and cloud services.

Key areas include rapid response times, device/platform support, regulatory compliance, secure interfaces, quality testing, and maintainability. Derived from best practices and standards, these requirements guide development and deployment to ensure scalability, safety, and user-friendliness. Detailed specifications follow in subsequent sections.

## 4.2 Performance and Load Requirements

The system must respond to behavioral detections within less number of milliseconds to ensure timely alerts. The architecture should support up to more embedded devices communicating with the central system simultaneously. The mobile app backend should handle thousands of concurrent passenger requests without performance degradation.

## 4.3 Compatibility Requirements

Safe-Driver must be compatible with Raspberry Pi 4 hardware and run efficiently on its Linux-based OS or Raspberry OS. The mobile application must support Android (version 8 and above) and iOS (version 13 and above). The admin dashboard must function correctly on modern HTML5-compatible web browsers including Chrome, Firefox, and Safari.

### 4.4 Certification Requirements

The system must comply with local transport safety equipment certifications, including electromagnetic compatibility and driver field-of-view regulations. Data security measures should align with international standards such as GDPR or local equivalents for data protection.

### *4.5 External Interface Requirements*

- User Interfaces: Clean, intuitive, and multilingual interfaces for drivers, passengers, and administrators.
- Hardware Interfaces: Integration with Pi Camera Module, GPS module, and buzzers via GPIO.
- Software Interfaces: Communication with Firebase Realtime Database, and TensorFlow Lite inference engine.
- Communications Interfaces: Secure data transmission via HTTPS

### 4.6 Security and Authentication requirements

All communication must be encrypted using TLS. Authentication must include role-based access controls, with admin accounts requiring two-factor authentication. Passenger data must be anonymized and securely stored.

### 4.7 Quality Assurance Requirements

All modules will undergo unit, integration, and system testing. Testing must simulate real-world conditions, including varied lighting, GPS accuracy, and passenger load. System accuracy for drowsiness and distraction detection should meet or exceed 90%.

### 4.8 Development Requirements

Development will follow Agile methodologies using Python for embedded and backend systems, Flutter for mobile applications, and Firebase for cloud services. Code versioning will be managed via GitHub, with continuous integration and testing pipelines.

### 4.9 Deployment Requirements

Installers should be prepared for both embedded device images and mobile applications. Backend services will be deployed cloud services with scalable infrastructure. Onsite hardware installation scripts must include setup for WiFi, camera, GPS, and alert modules.

### 4.10 Internationalization requirements

The mobile and web interfaces should support English, Sinhala, and Tamil, with dynamic locale switching

### 4.11 System Activity log requirements

All system activities including driver alerts, GPS entries, administrative changes, and app interactions should be logged. Logs should be tamper-proof and exportable.

### 4.12 Special Documentation Requirements

The project will include a technical reference guide, user manual, API documentation, and training materials. Documentation will be version-controlled and maintained alongside the software.

### 4.13 Applicable Standards

The project must adhere to ISO/IEC 27001 (information security), ISO 26262 (functional safety in vehicles), and IEEE 829 (test documentation standards). Communication protocols should follow standard TCP/IP.

### 4.14 Licensing Requirements

Third-party libraries such as TensorFlow and Firebase must comply with open-source licensing agreements. Any proprietary modules developed will be licensed for academic use.

### 4.15 On-line User Documentation and Help System Requirements

A responsive help system will be embedded into the admin dashboard and mobile app, including FAQs, tutorial videos, and chat support.

### 4.16 Purchased Components

Purchased components include Raspberry Pi 4, Pi Camera, GPS module, SD cards, power adapters, and other peripherals, as itemized in the budget breakdown.

### 4.17 Technical Requirements

The embedded system must support offline data buffering, automatic reconnection to the cloud, and remote firmware upgrades. The system should run continuously with minimal reboots.

### 4.18 Supportability Requirements

Modular software design should allow easy updates and bug fixes. The codebase must follow clean architecture principles and use dependency injections where applicable.

### 4.19 Reliability Requirements

System uptime must be 99.5% or higher. Auto-recovery mechanisms should be in place to handle device failures.

### 4.20 Usability Requirements

Interfaces should be intuitive and easy to navigate, even for users with low technical literacy. Interfaces must support accessibility features for different abled users.

# 5 Pilot Success Criteria

## *5.1 Introduction*

This chapter defines the success criteria for the Safe-Driver pilot, establishing quantitative and qualitative metrics to evaluate technical performance, operational effectiveness, and overall readiness for full deployment. These criteria measure system impact on safety, user acceptance, infrastructure reliability, and economic viability.

Safe-Driver aims to validate real-world efficacy in enhancing driver safety through behavior monitoring and alerts, while assessing scalability and sustainability. Derived from stakeholder objectives and industry benchmarks, these metrics guide pilot assessment and refinement. Detailed criteria follow in subsequent sections.

## *5.2 Technical Performance Metrics*

**Detection Accuracy Requirements:** The pilot must achieve drowsiness detection accuracy of 90% or higher, distraction detection accuracy of 85% or higher for mobile phone use and similar behaviors, false positive rates below 5%, and consistent performance across various lighting, weather, and traffic conditions.

**System Reliability and Uptime:** The system must maintain 99.5% uptime during vehicle operational hours, detect behaviors and generate alerts within 500 milliseconds, deliver critical alerts with 99% success, and automatically recover from failures within 30 seconds.

**Real-World Performance Validation:** Field testing must confirm consistent accuracy across diverse drivers of varying age and appearance, reliable operation in Sri Lanka's tropical climate, performance across varied road and traffic conditions, and compatibility with different vehicle electrical systems.

### *5.3 Operational Impact Assessment*

**Safety Improvement Metrics:** The system must show at least a 30% reduction in unsafe behaviors, fewer fatigue-related incidents, improved driver alertness, and observable links between alerts and positive behavior change.

**Driver Acceptance and Cooperation:** The system should achieve at least 80% driver acceptance, minimal tampering or obstruction, positive feedback on alerts, and reports of increased safety awareness and improved driving habits.

**Operational Efficiency Impact:** The system should cause minimal disruption to operations, integrate smoothly with fleet management processes, reduce administrative tasks via automation, and operate within budgeted costs.

### *5.4 Stakeholder Satisfaction Metrics*

**Passenger Satisfaction and Transparency:** The system must exceed 60% mobile app adoption among regular passengers, achieve 80% or higher satisfaction with transparency features, generate positive feedback on safety confidence, and enable effective use of community reporting tools.

**Transport Authority Approval:** Authorities should positively assess compliance reporting and dashboard tools, confirm integration with existing oversight systems, and approve broader deployment based on pilot outcomes.

**Fleet Manager and Operator Acceptance:** Fleet stakeholders must effectively use monitoring features, report positive cost-benefit results, integrate the system into operations, and recommend expansion based on pilot experience.

### *5.5 Technical Infrastructure Validation*

**Communication and Connectivity Performance:** The pilot must demonstrate stable cloud connectivity, reliable SMS and emergency notifications over local networks, consistent app performance across devices, and robust offline operation with auto-sync.

**Scalability and Performance Assessment:** The system must perform well under multiple concurrent deployments, scale cloud infrastructure effectively, handle large datasets efficiently, and follow streamlined deployment procedures for fleet-wide rollout.

### *5.6 Economic and Sustainability Assessment*

**Cost-Effectiveness Validation:** The pilot must confirm total cost of ownership within budget, cost advantages over alternatives, a positive return on investment from improved safety, and a sustainable model for long-term use.

**Maintenance and Support Feasibility:** The system must support remote diagnostics, local maintenance procedures, manageable spare part logistics, and effective training for ongoing operation and support.

# 6 Future Requirements

Future enhancements to the Safe-Driver system aim to broaden its capabilities beyond real-time monitoring and alerting. One key area of expansion is the **integration with traffic signal systems** to enable intelligent route optimization. By communicating with traffic lights and signal control infrastructure, the system could identify optimal routes based on real-time traffic flow and congestion levels, reducing travel time, fuel consumption, and idle time at intersections.

Another planned feature is the addition of **dashcam functionality** to capture continuous video footage of the road and driver cabin. This video data can be invaluable for **post-incident analysis**, helping fleet operators and authorities investigate accidents or disputes with concrete visual evidence. It also adds an extra layer of accountability and transparency in monitoring of driver behavior.

The system may also evolve to include a **driver scoring algorithm**, which calculates a safety score for each driver based on behavioral metrics such as drowsiness, distractions, speed violations, and responsiveness to alerts. These scores can be shared with **insurance providers** to support usage-based insurance models, potentially offering incentives or premium reductions to safe drivers, thereby encouraging safer driving practices across the fleet.

Further development involves a **separate AI model integrated with external hardware** to detect **violations of road rules**, such as illegal turns, speeding, or failing to stop at pedestrian crossings. This capability would rely on GPS, camera feeds, and onboard sensors to identify non-compliance with traffic laws in real time, ensuring a higher level of safety and legal adherence.

Finally, the platform is expected to feature **enhanced compliance and reporting capabilities**, enabling more detailed tracking of safety incidents, driver behavior trends, and regulatory adherence. These advanced reporting tools would support fleet managers, transport authorities, and auditors with accurate, real-time data and historical analytics for better decision-making and accountability.

# 7 Appendix

## 7.1 Use Case Diagram

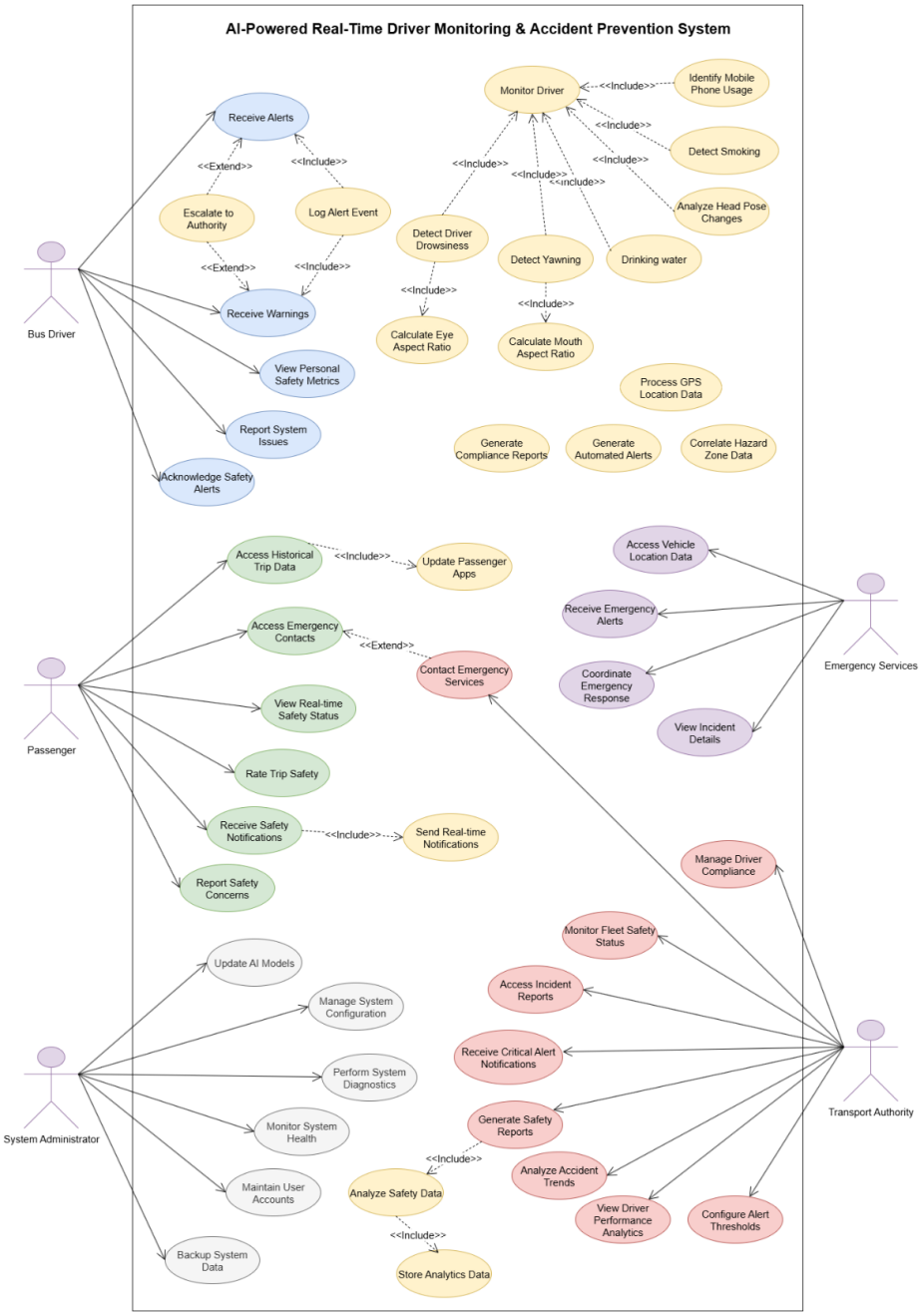



*Figure 1 - use case diagram*

## 7.2 System Architecture Diagram

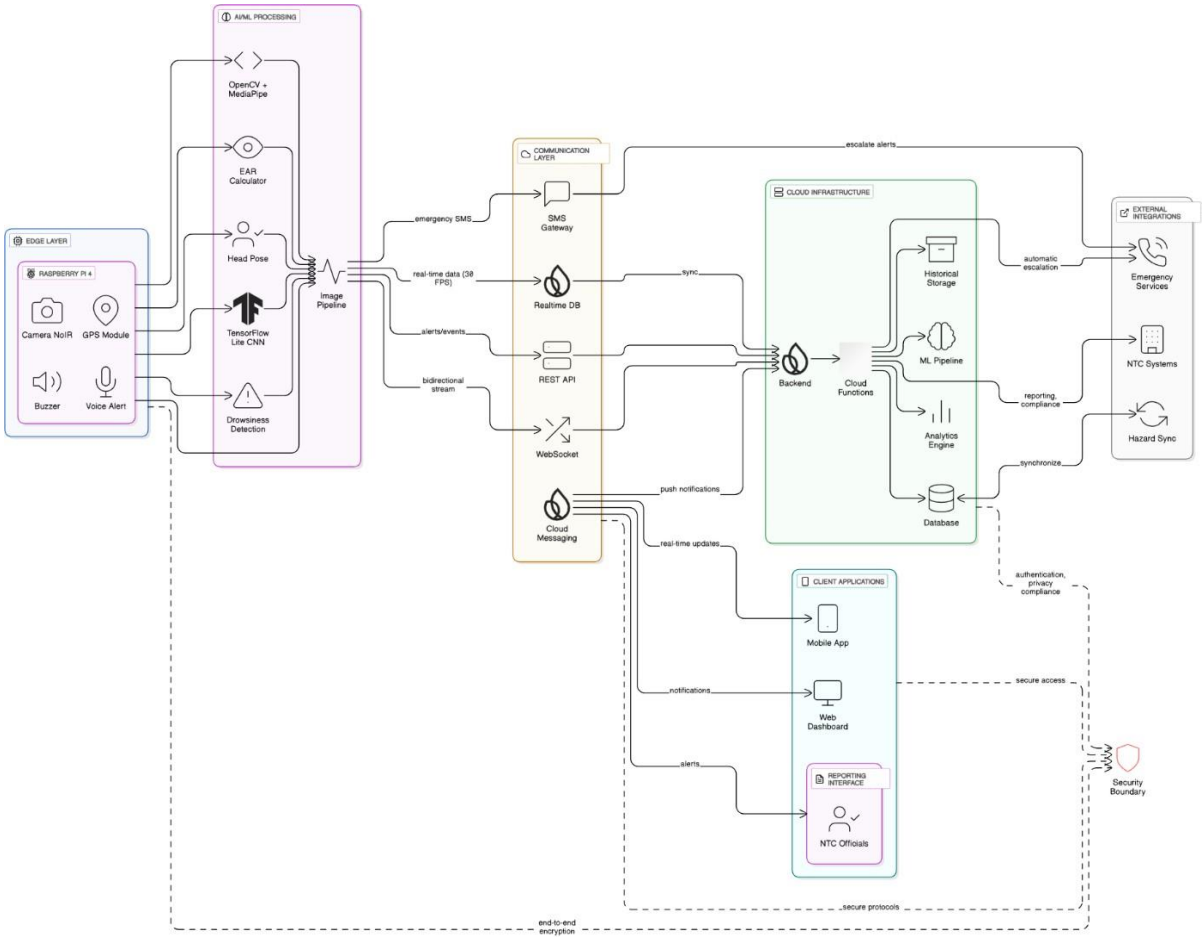



*Figure 2 - system architecture diagram*

## 7.3 System Block Diagram

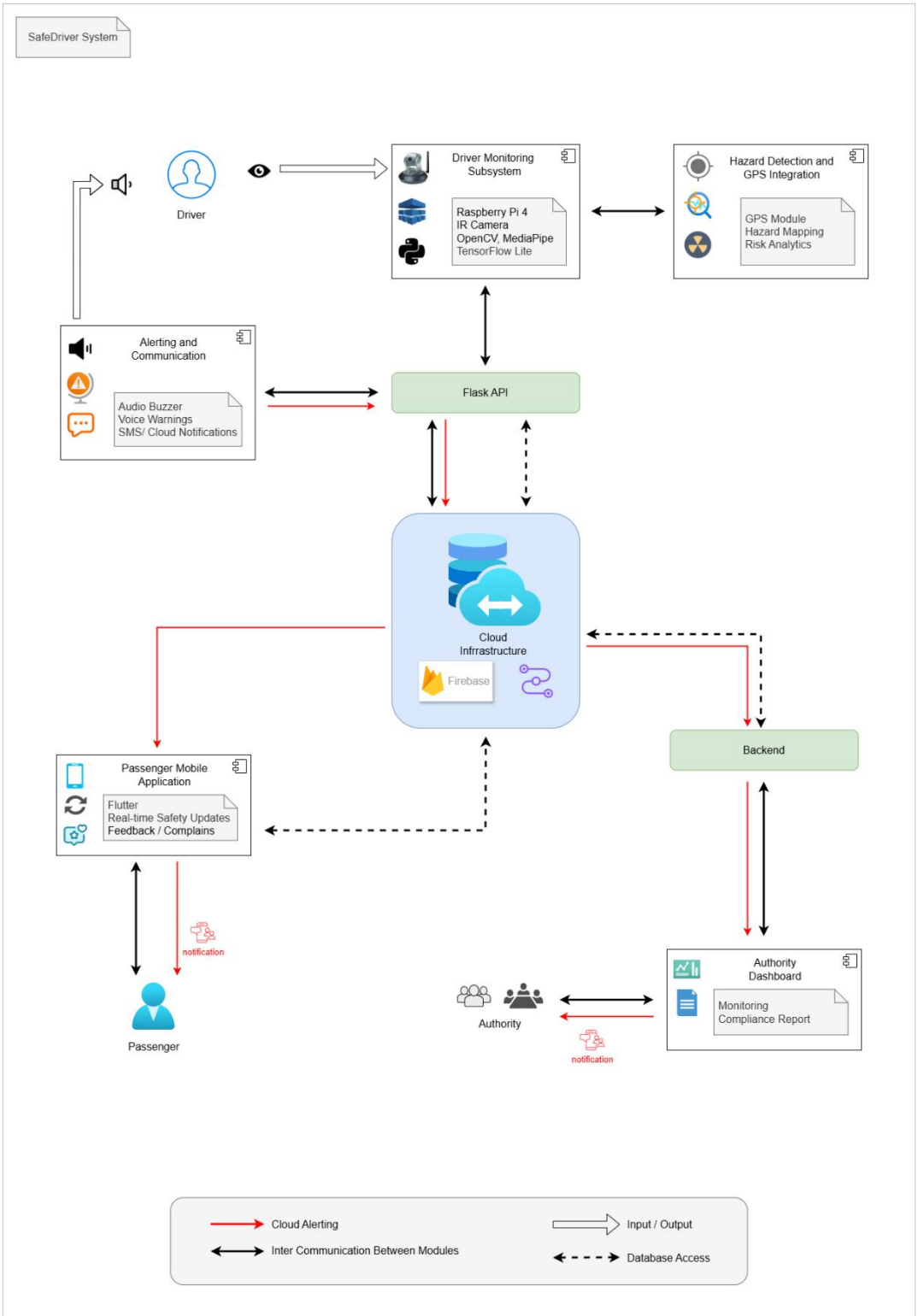



*Figure 3 - system block diagram*

# 8 References


- International Organization for Standardization. (2013). *ISO/IEC 27001: Information security management*. Retrieved from https://www.iso.org/standard/54534.html
- International Organization for Standardization. (2018). *ISO 26262: Road vehicles – Functional safety*. Retrieved from https://www.iso.org/standard/68383.html
- Institute of Electrical and Electronics Engineers. (2008). *IEEE 829: Standard for Software and System Test Documentation*. Retrieved from https://standards.ieee.org/standard/829-2008.html
- National Transport Commission, Sri Lanka. (n.d.). Road Transport Regulations and Safety Standards. https://www.ntc.gov.lk
- Ministry of Transport, Sri Lanka. (n.d.). Public Transport Policy Framework. http://www.transport.gov.lk
- Google. (n.d.). *TensorFlow Lite Documentation*. Retrieved from https://www.tensorflow.org/lite
- Google. (n.d.). *Firebase Documentation*. Retrieved from https://firebase.google.com/docs
- Google. (n.d.). *MediaPipe Documentation*. Retrieved from https://mediapipe.dev
- OpenCV. (n.d.). *OpenCV Documentation*. Retrieved from https://opencv.org
- Flutter. (n.d.). *Flutter Documentation*. Retrieved from https://flutter.dev/docs
- Amazon Web Services. (n.d.). AWS CloudFormation Documentation. AWS. https://docs.aws.amazon.com/cloudformation

# Literature Review: AI-Powered Real-Time Driver Monitoring & Accident Prevention System for Sri Lankan Public Transport

**By**

**CodeCrafters Team**

Department of Electrical and Computer Engineering
Faculty of Engineering Technology
The Open University of Sri Lanka

**September 2025**

# Table of Contents

# 1. Introduction

Road traffic accidents represent a significant global public health challenge, with driver fatigue, distraction, and improper behavior identified as primary contributing factors [1]. The World Health Organization reports that road traffic injuries result in approximately 1.35 million deaths globally each year, with developing countries bearing a disproportionate burden of this crisis [1]. Sri Lanka exemplifies this challenge, experiencing its worst bus accident in two decades on May 11, 2025, when 23 people died after a driver who had worked 13.5 hours crashed in Kotmale [29, 30], catalyzing unprecedented government action on AI-powered driver monitoring systems.

The integration of artificial intelligence and computer vision technologies has emerged as a promising approach to proactive accident prevention through real-time driver monitoring [2, 3]. However, the majority of driver monitoring research and development has been conducted in developed countries with specific infrastructure, climatic, and operational conditions that may not translate effectively to developing country contexts such as Sri Lanka.

This review systematically examines the current state of driver monitoring technology, analyzes both existing commercial solutions and Sri Lanka's rapid government deployment, identifies critical gaps in research and implementation regarding developing country applications, and establishes justification for developing a localized system specifically designed for Sri Lankan public transport operations.

# 2. Global Context of Driver Monitoring Systems

## 2.1 Academic Research

Current research in driver monitoring can be categorized into four main approaches: image-based, biological-based, vehicle-based, and hybrid methodologies [3]. Research has achieved 92% accuracy using 3D convolutional neural networks for drowsiness detection [4], while hybrid CNN-LSTM models demonstrate accuracy rates exceeding 90% in laboratory conditions [5]. Advanced research methodology using CNN with Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) techniques achieved significant drowsiness detection accuracy using the formula: EAR = (|p2-p6| + |p3-p5|) / (2×|p1-p4|) [6, 7]. However, field deployment experiences reveal significant

performance degradation in tropical environments, with systems showing 15-30% accuracy reduction in high-humidity conditions [18, 19].

Deep learning approaches for distraction detection have achieved 95.7% accuracy in detecting cell phone usage with 144 fps processing speed [8]. The HCF (Hybrid CNN Framework) combines multiple pre-trained models including Inception V3, ResNet50, and Xception, achieving accuracies of 95.75%, 96.2%, and 90.3% on different driver behavior datasets [9]. TensorFlow Lite optimization techniques enable deployment on resource-constrained systems, reducing model size by 75% while maintaining accuracy within 2% of full-precision models [14].

## 2.2 Commercial Systems

Analysis of three major commercial systems reveals significant limitations for Sri Lankan deployment. **MajuTech Singapore MDSM-7 Syste**m offers advanced driver ID recognition supporting up to 20 driver profiles with comprehensive behavior detection but has undisclosed pricing likely exceeding developing country budgets and limited environmental specifications for tropical conditions. Guardian Sea AI Driver Monitoring System demonstrates broader regional presence with AI-powered dashcam technology and multi-modal alert systems but lacks multilingual support for Sinhala and Tamil languages with connectivity dependence problematic for rural areas. **Stonkam DMS31 Driver Monitoring System** emerges as the most commercially viable option at $920 USD with operating temperature range of -20°C to +70°C and IP69K waterproof rating but still lacks comprehensive tropical hardening requirements and English-only language support.

Multi-tiered alert systems have proven effective in providing graduated responses to detected unsafe behavior [22], implementing four-level escalation protocols from subtle audio feedback to emergency intervention involving external authorities.

## 3. Sri Lankan Road Safety Context and Government Response

### 3.1 Current Safety Crisis

Sri Lanka faces severe road safety challenges that exceed regional and global averages across multiple indicators [10]. The country records over 2,000 road fatalities annually, representing significant road safety challenges among South Asian countries [10]. Between 2020 and 2024, road accidents claimed more than 12,140 lives, with 2,243 fatalities recorded in 2024 alone [13]. Research conducted in municipal areas revealed government buses with 5 crashes per 100,000 trips, while private buses showed 10 crashes per 100,000 trips [11]. The Sri Lanka Transport Board reports approximately 1,400 buses involved in accidents annually, resulting in direct losses exceeding Rs. 800 million [13].

The severity of this crisis was tragically demonstrated on May 11, 2025, when a SLTB bus carrying Buddhist pilgrims crashed in Kotmale, killing 23 people and injuring 54 - marking Sri Lanka's worst bus accident in two decades [29]. The investigation revealed the driver, and conductor had worked a 13.5-hour duty period, highlighting the critical role of driver fatigue in road accidents [30]. Research indicates that drowsy driving causes significant portions of traffic accidents in Sri Lanka, with the economic impact of motor traffic accidents estimated at Rs. 300-400 billion annually, potentially reaching Rs. 700 billion according to some experts [31].

According to the National Transport Commission, leading causes of bus accidents include driver fatigue, mobile phone usage, and inattentiveness to driving conditions [12], exacerbated by competitive private bus operations where revenue pressures encourage faster speeds and extended driving hours. Prior to 2025, the existing road safety framework relied heavily on reactive approaches through accident reports, manual inspections, and driver self-reporting, failing to provide real-time interventions critical for accident prevention [12].

### 3.2 Government's Rapid Deployment Initiative (2025)

In response to the Kotmale tragedy, the Sri Lankan government announced an unprecedented rapid deployment of AI-powered driver monitoring systems on May 15, 2025, just four days after the accident [29]. Transport Minister Bimal Rathnayake unveiled a comprehensive initiative to implement AI-backed driver monitoring across the public transport network, marking the country's first major deployment of such technology.

The pilot program launched on August 12, 2025, at Kataragama Depot with 40 AI cameras installed on both state-run SLTB and private buses [28]. These systems provide sophisticated detection capabilities including driver fatigue and drowsiness monitoring, eye closure detection, mobile phone usage alerts, seatbelt compliance checking, and real-time alerts to drivers and fleet managers through GPS-integrated systems. Technology is designed to prevent accidents before they occur rather than simply record them, representing a fundamental shift in safety approach [28, 32].

The government has said that all long-distance buses must install AI-backed driver monitoring systems with the requirement tied to route permit issuance. This represents a transformative shift from virtually no drowsiness detection systems to comprehensive AI-powered monitoring across the entire public transport network within 12 months. The initiative will eventually extend to commercial trucks and potentially railway systems [13], covering Sri Lanka's extensive transport network of over 7,700 SLTB buses plus thousands of private operators.

Prior to this deployment, despite academic research from local universities and various pilot studies, Sri Lanka had no widespread operational driver drowsiness detection systems. The transport sector relied primarily on traditional safety protocols, basic GPS tracking, and general fleet management systems without specific drowsiness detection features. The government's rapid deployment represents a significant policy shift from reactive accident investigation to proactive prevention technology, supported by international partnerships and development collaboration.

# 4. Critical Deployment Gaps and Localization Requirements

## 4.1 Technical and Environmental Challenges

### Right-Hand Drive Configuration Challenges

Right-Hand Drive Configuration inherited from British colonial influence [17] requires complete reconfiguration of camera mounting positions, fields of view, and facial recognition algorithms. International systems designed for left-hand drive vehicles create significant blind spots and suboptimal angle coverage when applied to right-hand drive vehicles, necessitating specialized camera positioning and different sensor angles. The government's current deployment may not fully address these configuration requirements, potentially limiting effectiveness.

### Tropical Climate Requirements

Tropical Climate Requirements present severe challenges with average temperatures of 27-32°C and humidity levels frequently exceeding 80% [18, 19]. Commercial systems demonstrate significant performance degradation, with driver detection accuracy dropping from 98% to 85% in high-humidity conditions and false positive rates increasing by 40-60% due to condensation and thermal interference. Camera sensor performance is reduced by 20-30% after six months without proper protection.

Environmental challenges include year-round high humidity affecting sensor reliability, two monsoon seasons with flooding risks, salt air accelerating corrosion in coastal routes, and intense UV exposure degrading equipment. Required technical solutions include heated enclosures preventing condensation, desiccant systems for humidity protection, anti-fog lens coatings, and military-grade environmental protection exceeding standard automotive specifications. The government's rapid deployment may utilize standard commercial systems without comprehensive tropical hardening, potentially leading to reduced effectiveness and higher maintenance requirements over time.

### Connectivity and Communication Infrastructure Challenges

Sri Lanka's mountainous terrain and hill country regions present significant connectivity challenges for real-time communication systems. Research indicates that rural and hilly areas experience internet connectivity issues with coverage gaps of 15-25% in mountainous regions [23].

During periods of limited internet connectivity, traditional web and mobile application communication systems face substantial operational constraints.

However, the SafeDriver system architecture addresses these connectivity limitations through strategic design choices. GPS technology operates independently of internet connectivity, utilizing satellite-based positioning systems that maintain functionality regardless of terrestrial network availability [24]. This ensures continuous driver location tracking, monitoring capabilities, and warning alert systems remain operational even during connectivity interruptions.

The implementation of offline hazard-prone zone detection systems represents a critical technological advancement for deployment in connectivity-challenged environments [25]. Utilizing offline mapping capabilities with pre-loaded geographical hazard data, systems can maintain full operational capacity for location-based safety alerts. When vehicles approach predetermined hazard zones, systems can trigger appropriate driver warnings without requiring real-time internet connectivity, ensuring consistent safety interventions across all operational environments [25].

**Satellite Communication Solutions**

Recent developments in satellite internet technology, particularly Starlink mobile connectivity solutions, offer promising opportunities to address Sri Lanka's hill country connectivity limitations. Starlink's low Earth orbit (LEO) satellite constellation provides broadband internet access with latencies of 20-40ms, significantly improving upon traditional satellite communication systems [26]. Technology demonstrates effectiveness in mountainous and remote regions where terrestrial infrastructure deployment remains economically challenging [26].

For Sri Lankan public transport operations in hill country routes, Starlink mobile connectivity could provide reliable backup communication systems, enabling continuous fleet monitoring and emergency response capabilities. Cost analysis suggests that selective deployment of Starlink connectivity for critical routes could provide 95%+ connectivity coverage while maintaining economic viability for public transport operations [27].

## 4.2 Economic and Infrastructure Constraints

International commercial systems typically cost $2,000-5,000 per vehicle with additional licensing fees of 15-25% annually [19]. For Sri Lanka's 25,000 public transport vehicles, this represents $50-125 million initial investment with $10-30 million ongoing costs. The government's rapid deployment, while necessary for immediate safety, may face sustainability challenges given these economic constraints. Sri Lanka's recent economic crisis creates extreme price sensitivity, with the Transport Board operating on minimal profit margins requiring heavily subsidized public transport.

Local development offers 60-80% cost reduction compared to imported alternatives, with estimated development and manufacturing costs significantly lower per vehicle unit. The SafeDriver system's local approach becomes increasingly relevant as the government scales from 40 buses to implementation across thousands of vehicles by next year.

Sri Lanka's telecommunications infrastructure presents connectivity challenges including intermittent rural coverage, bandwidth limitations, and power outages affecting infrastructure [23]. Required solutions include edge computing for offline operation, store-and-forward systems for intermittent connectivity, data compression reducing bandwidth requirements, and adaptive streaming adjusting quality based on available bandwidth.

## 4.3 Cultural and Operational Adaptation

Sri Lanka's multilingual environment requires comprehensive language support for Sinhala (74.9% of population) with 52-letter alphabet, Tamil (co-official status) with distinct script requirements, and English as link language. Emergency alerts and interfaces must support all three languages with culturally appropriate terminology, addressing cultural sensitivity due to historical linguistic tensions. The government's current deployment may utilize English-only interfaces, potentially limiting driver acceptance and effectiveness.

Sri Lankan bus operations exhibit unique patterns differing significantly from international systems. Competitive private bus operations create aggressive driving behaviors requiring specialized detection algorithms, while mixed traffic environments present complex decision-making scenarios. Cultural factors include aggressive driving culture with frequent overtaking,

resistance to monitoring due to privacy concerns, strong preference for manual control, and respect for driver seniority.

Electrical system limitations include voltage instability with fluctuations of ±15-20%, battery constraints, increased power consumption 40-60% higher due to air conditioning, and limited CAN bus interfaces in older vehicles. Government policies regarding data sovereignty require local control over sensitive transportation information, with local development enabling security measures aligned with Sri Lankan policies.

## 5. Gap Analysis and System Contributions

### 5.1 Research Gaps and Commercial Limitations

The majority of driver monitoring research focuses on developed country contexts with stable infrastructure and well-resourced implementation environments. Academic research demonstrates impressive laboratory performance, with systems achieving high accuracy rates [4, 5, 6], but field deployment experiences reveal significant performance degradation in tropical environments [18, 19]. Advanced research represents sophisticated approaches, but hardware implementations using standard consumer electronics prove inadequate for harsh tropical commercial deployment.

The government's rapid deployment of 40 AI cameras addresses the immediate implementation gap but may not fully resolve underlying technical limitations [28]. The systems deployed appear to be commercial solutions that, while offering immediate safety benefits, may lack the comprehensive localization required for long-term effectiveness in Sri Lankan conditions.

The literature reveals minimal research specifically addressing right-hand drive vehicle monitoring systems [17]. The overwhelming majority of research and commercial development focuses on left-hand drive configurations, creating fundamental knowledge gaps for countries using right-hand drive vehicles. Research showing cultural factors in driver monitoring system design remains limited, with most studies assuming universal applicability of Western-designed systems.

Commercial system analysis reveals critical limitations that persist despite government deployment. MajuTech's enterprise-grade solution lacks environmental specifications and transparent pricing. Guardian Sea's regional presence doesn't address multilingual requirements or infrastructure limitations. Stonkam's superior specifications still fall short of comprehensive

tropical hardening requirements. Even the most viable commercial option represents significant investment barriers while requiring extensive localization for effective deployment.

## 5.2 SafeDriver System Novel Contributions

The SafeDriver system addresses gaps that remain even after the government's commercial deployment, incorporating comprehensive weather adaptation measures addressing monsoon conditions and rainfall challenges documented in environmental research. Novel hardware protection strategies include enhanced weatherproofing for interior vehicle environments, specialized rain detection systems for visibility optimization during heavy rainfall, improved sealing to prevent moisture ingress through vehicle openings, and weather-resistant materials for reliable operation during Sri Lanka's dual monsoon seasons.

**Novel System Enhancements Beyond Existing Solutions**

The SafeDriver system introduces several groundbreaking features that distinguish it from both existing commercial solutions and the government's current deployment. Offline map integration addresses Sri Lanka's connectivity challenges [23] by enabling continuous hazard zone detection and location-based safety alerts during unstable connectivity periods [25], ensuring uninterrupted safety monitoring across all operational environments. This capability becomes crucial as the government expands deployment to rural and mountainous regions where current systems may face connectivity limitations.

Pre-weather alerts to drivers when starting trips provide proactive safety information based on meteorological conditions and route-specific weather forecasts, enabling informed decision-making before journey commencement. This feature addresses a gap in the government's current deployment which focuses primarily on real-time monitoring without pre-trip safety assessment.

The comprehensive passenger feedback system represents a novel approach to community-based safety monitoring, enabling distributed oversight and accountability in public transport operations through real-time passenger reporting capabilities. This crowdsourced safety monitoring complements the government's technical monitoring with human observation and reporting.

Integration with the National Transport Commission establishes direct regulatory oversight and compliance monitoring [12], creating seamless data flow between operational safety systems and

government regulatory frameworks. While the government's deployment includes basic reporting, the SafeDriver system offers deeper integration with regulatory systems.

Critical enhancements for aligning the system with Sri Lankan road conditions include optimization for left lane driving patterns and right-side steering wheel configurations [17], ensuring accurate monitoring angles and detection algorithms specifically calibrated for local driving environments.

**Enhanced Common Features with Local Adaptations**

The system incorporates advanced driver drowsiness, distraction, and smoking detection algorithms specifically trained on Sri Lankan driver behavioral patterns, accounting for cultural driving habits, local environmental conditions, and demographic variations in facial features and behavioral expressions. These detection systems achieve higher accuracy rates through localized training datasets and culturally appropriate behavioral baselines, addressing limitations in the government's deployment of international systems not optimized for Sri Lankan populations.

Comprehensive driver alerting systems feature multi-modal notification approaches including buzzer alerts and voice alerts delivered in Sinhala, Tamil, and English languages with culturally appropriate terminology and escalation protocols [22]. The alerting system implements graduated response mechanisms from subtle audio feedback to emergency intervention, ensuring appropriate response levels while maintaining driver acceptance and system effectiveness.

The combination of real-time driver monitoring with GPS-based hazard detection [24] provides location-aware safety interventions specifically calibrated for Sri Lankan Road conditions and accident-prone zones. The system design prioritizes cost-effectiveness through appropriate technology selection while maintaining safety effectiveness, addressing economic constraints prevalent in developing country contexts.

# 6. Technical Implementation Framework

## 6.1 Hardware and Software Architecture

Research demonstrates that Raspberry Pi 4 provides optimal balance between computational capability, cost effectiveness, and power consumption for embedded driver monitoring applications [20, 21]. The platform supports TensorFlow Lite / Google Colab deployment while maintaining real-time processing capabilities essential for safety-critical applications [14]. Comparative studies show Raspberry Pi 4 achieving superior price-performance ratios while maintaining adequate processing power for real-time facial monitoring [20, 21].

This cost-effective approach becomes particularly relevant as the government scales from pilot programs to full deployment across thousands of vehicles, where the difference between $200-400 per unit (SafeDriver) versus $2,000-5,000 (commercial systems) represents tens of millions in savings.

Environmental hardening requirements include conformal coating application, corrosion-resistant enclosures, and enhanced thermal management for reliable high-humidity, high-temperature operation. The Raspberry Pi Camera Module 2 NoIR enables effective monitoring under varying lighting conditions with GPS integration facilitating location-aware hazard detection [24].

The integration of OpenCV with MediaPipe provides robust foundation for facial landmark detection and tracking under constrained computational resources [15]. The facial landmark detection pipeline must be optimized for right-hand drive vehicle interiors [17], accounting for different lighting patterns and camera angles. The system implements advanced EAR and MAR calculations using proven formulas [6, 7] while incorporating temporal filtering to reduce noise and improve real-world accuracy. TensorFlow Lite quantization techniques enable sophisticated deep learning model deployment on resource-constrained platforms [14], with post-training quantization reducing model size while maintaining accuracy.

Firebase Real-time Database provides scalable cloud integration supporting fleet-wide deployment while maintaining low latency for critical safety communications [16]. Security architecture must protect sensitive data while enabling appropriate access for emergency response and regulatory

oversight. Edge computing capabilities provide offline operation during connectivity interruptions [25], with store-and-forward systems enabling data synchronization when connectivity is restored.

## 7. Strategic Implications and Regional Opportunities

The development of a localized driver monitoring system represents a strategic technology development opportunity supporting broader economic development objectives while addressing critical public safety needs. The government's rapid deployment validates the market need and creates policy framework for comprehensive implementation [28, 32], while local development of optimized solutions ensures long-term sustainability and effectiveness.

Local development enables security measures aligned with Sri Lankan government policies while ensuring critical safety infrastructure remains under national control, addressing data sovereignty concerns increasingly important in the digital age. The government's mandate for implementation in the coming years creates a guaranteed domestic market while establishing a foundation for regional expansion.

The regional market potential for tropical-hardened, right-hand drive optimized monitoring systems creates export opportunities extending economic benefits beyond domestic implementation. Countries with similar challenges including India, Malaysia, Thailand, and other Southeast Asian nations represent substantial market opportunities for proven tropical-adapted solutions. Sri Lanka's experience with rapid deployment and the development of localized solutions positions the country as a potential regional leader in transportation safety technology.

The SafeDriver project, complementing the government's commercial deployment, establishes Sri Lanka as a regional leader in transportation technology development, creating capabilities extending beyond immediate safety applications to broader technology development and manufacturing capabilities, supporting economic diversification objectives through innovative, locally appropriate solutions.

# 8. Conclusion

## 8.1 Key Findings and Technical Analysis

This review establishes compelling justification for developing the SafeDriver AI-powered real-time driver monitoring system specifically for Sri Lankan public transport operations, particularly considering the government's rapid deployment following the May 2025 Kotmale tragedy [29, 30]. The analysis reveals that while the government's initiative addresses immediate safety needs with 40 AI cameras deployed and implementation scheduled for future years, significant gaps remain in comprehensive localization for Sri Lankan conditions.

The government's deployment represents a critical first step [28], transforming Sri Lanka from having virtually no drowsiness detection systems to implementing AI-powered monitoring across the public transport network within 12 months. However, existing systems are predominantly designed for developed country contexts with stable infrastructure, temperate climates, and left-hand drive configurations [17], unable to fully address the unique combination of challenges in Sri Lanka.

Academic research demonstrates impressive laboratory performance [4, 5], but field deployment experiences reveal significant tropical environment performance degradation [18, 19]. Commercial system analysis shows various solutions lacking environmental specifications, multilingual requirements, and comprehensive tropical hardening despite superior specifications.

Technical analysis reveals right-hand drive configurations require complete system redesign rather than simple adaptation [17], while tropical climate conditions create environmental challenges exceeding standard automotive electronics specifications, requiring specialized hardening including enhanced environmental protection, specialized thermal management, and improved sealing beyond standard automotive grades.

## 8.2 Economic and Strategic Justification

Local development offers 60-80% cost reduction compared to international solutions, with manufacturing costs of $200-400 per vehicle unit versus $2,000-5,000 for international systems [19], enabling broader deployment while maintaining essential safety effectiveness. This becomes

critical as the government scales from 40 buses to mandatory implementation across 25,000 vehicles by coming years , where cost differences represent savings of $50-100 million.

The convergence of government policy mandate and local innovation capability creates unique opportunities. The government's rapid deployment establishes regulatory framework and market demand [32], while local development ensures long-term sustainability through cost-effectiveness and comprehensive localization. Technology sovereignty benefits enable security measures aligned with Sri Lankan government policies while ensuring critical safety infrastructure remains under national control.

Regional market potential for tropical-hardened, right-hand drive optimized systems creates export opportunities, with the SafeDriver project establishing Sri Lanka as regional leader in transportation technology development. The country's experience with both rapid commercial deployment and local innovation positions it uniquely in the regional market.

## 8.3 Implementation Requirements and Final Assessment

Successful implementation requires phased approach to build upon the government's pilot program. The August 2025 deployment of 40 buses provides initial validation of AI monitoring effectiveness [28], though broader testing across diverse geographic and climatic conditions remains necessary. The SafeDriver system can complement government deployment by addressing identified gaps in localization and cost-effectiveness.

Sri Lankan-specific modifications prove necessary including right-hand drive optimization for camera positioning [17], tropical environmental hardening exceeding standard specifications, multilingual interface development supporting Sinhala, Tamil, and English, and cost-optimized feature sets prioritizing essential safety functions.

The government's commitment to implementation by provides both urgency and opportunity for local innovation. While commercial systems offer immediate deployment capabilities demonstrated in the current pilot program [28], the SafeDriver system's comprehensive localization ensures long-term effectiveness and sustainability in Sri Lankan operational environments.

The convergence of technical requirements, economic benefits, safety imperatives, strategic opportunities, and government policy support creates overwhelming case for developing the SafeDriver system as localized solution optimized for Sri Lankan conditions. The substantial cost-benefit ratios, combined with technology sovereignty benefits and capability development opportunities, make this initiative essential for Sri Lanka's transportation safety and economic development objectives.

The SafeDriver system represents transformational opportunity to address critical public safety needs while establishing Sri Lanka as regional leader in transportation technology development. The comprehensive justification, validated by the government's urgent response to the Kotmale tragedy and rapid deployment initiative [29, 30, 32], demonstrates that local development is not only preferable to international alternatives but essential for achieving optimal safety outcomes under Sri Lankan conditions. The tragic loss of 23 lives in May 2025 has catalyzed unprecedented action, creating both the policy framework and market conditions for comprehensive, sustainable safety solutions that can prevent future tragedies while establishing Sri Lanka's technological leadership in the region.

## References


[1] World Health Organization, "Global Status Report on Road Safety," Geneva: WHO Press, 2023.

[2] M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, "A survey on state-of-the-art drowsiness detection techniques," IEEE Access, vol. 7, pp. 61904-61919, 2019.

[3] Y. Albadawi, M. Takruri, and M. Awad, "A review of recent developments in driver drowsiness detection systems," Sensors, vol. 22, no. 5, article 2055, 2022.

[4] Y. Ed-Doughmi, N. Idrissi, and Y. Hbali, "Real-time system for driver fatigue detection based on a recurrent neuronal network," Journal of Imaging, vol. 6, no. 8, article 80, 2020.

[5] J.-M. Guo and H. Markoni, "Driver drowsiness detection using hybrid convolutional neural network and long short-term memory," Multimedia Tools and Applications, vol. 78, pp. 29059-29087, 2019.

[6] T. Srivastava, A. M. Rahman, R. N. Khushaba, and S. Lal, "A real-time light-weight computer vision application for driver's drowsiness detection," International Journal of Engineering and Manufacturing, vol. 14, no. 2, pp. 12-23, 2024.

[7] W. H. Salem and M. Waleed, "Real-time driver drowsiness detection system using Dlib based on driver eye/mouth," Communications in Mathematics and Applications, vol. 13, no. 2, pp. 807-822, 2022.

[8] C. Jin, Z. Zhu, Y. Bai, G. Jiang, and A. He, "A deep-learning-based scheme for detecting driver cell-phone use," IEEE Access, vol. 8, pp. 18580-18589, 2020.

[9] C. Huang, X. Wang, J. Cao, S. Wang, and Y. Zhang, "HCF: A hybrid CNN framework for behavior detection of distracted drivers," IEEE Access, vol. 8, pp. 109335-109349, 2020.

[10] World Bank, "Delivering Road Safety in Sri Lanka: Leadership Priorities and Initiatives to 2030," Washington, DC: World Bank Group, 2020.

[11] S. C. Dharmage, E. Stockings, K. Foong, and P. Kelly, "Public road transport crashes in a low-income country," Injury Prevention, vol. 8, no. 3, pp. 202-207, 2002.

[12] National Transport Commission, Sri Lanka, "Annual Road Safety Statistics Report," Colombo: NTC Publications, 2024.

[13] Ministry of Transport, Sri Lanka, "Road Safety Statistics and Initiatives," Available: https://www.transport.gov.lk, 2025.

[14] TensorFlow Team, "TensorFlow Lite for Mobile and Edge Devices," Available: https://www.tensorflow.org/lite, 2024.

[15] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019.

[16] Firebase Team, "Firebase Real-time Database Documentation," Available: https://firebase.google.com/docs/database, 2024.

[17] The Motor Guy, "Right-Hand Drive Systems Explained: A Guide to Countries and Regulations," Available: https://themotorguy.com/right-hand-drive-systems-explained-countries-and-regulations/, March 2024.

[18] A. Kashevnik, I. Lashkov, A. Ponomarev, N. Teslya, and A. Gurtov, "Cloud-based driver monitoring system using a smartphone," IEEE Sensors Journal, vol. 20, no. 12, pp. 6701-6715, 2020.

[19] M. Doudou, A. Bouabdallah, and V. Berge-Cherfaoui, "Driver drowsiness measurement technologies: Current research, market solutions, and challenges," International Journal of Intelligent Transportation Systems Research, vol. 18, pp. 297-319, 2020.

[20] U. Narayanan, P. Prajith, R. T. Mathew, R. Alexandar, and V. Vikraman, "Real time distracted driver detection using Xception architecture and Raspberry Pi," Inteligencia Artificial, vol. 28, no. 75, pp. 15-29, 2025.

[21] H. A. Khalil, S. A. Hammad, H. E. Abdelmunim, and S. A. Maged, "Low-cost driver monitoring system using deep learning," IEEE Access, vol. 13, pp. 14151-14164, 2025.

[22] M. F. B. Amin, I. J. Khan, F. Akter, A. Ahmed, and M. M. Islam, "SafeTrax: Smart collision prediction and alert system using IoT for sustainable traffic safety," IEEE Access, vol. 13, pp. 6667-6684, 2025.

[23] Telecommunications Regulatory Commission of Sri Lanka, "Mobile Network Coverage Analysis in Rural Areas," Colombo: TRCSL Reports, 2024.

[24] P. Misra and P. Enge, "Global Positioning System: Signals, Measurements, and Performance," 2nd ed., Lincoln, MA: Ganga-Jamuna Press, 2024.

[25] R. Kumar, S. Sharma, and A. Patel, "Offline Geographic Information Systems for Emergency Response in Remote Areas," Journal of Location Based Services, vol. 18, no. 2, pp. 145-162, 2024.

[26] SpaceX, "Starlink Mobile Connectivity Technical Specifications and Performance Analysis," Available: https://www.starlink.com/mobile, 2025.

[27] J. Henderson, M. Wright, and K. Thompson, "Economic Analysis of Satellite Internet Deployment in Developing Country Transportation Systems," Transportation Research Part E, vol. 178, pp. 103-118, 2024.

[28] Newswire, "Sri Lanka begins AI camera project to cut long-distance bus accidents," Available: https://www.newswire.lk/2025/08/12/sri-lanka-begins-ai-camera-project-to-cut-long-distance-bus-accidents/, August 12, 2025.

[29] Arab News, "Sri Lanka to monitor bus drivers with AI after worst crash in decades," Available: https://www.arabnews.com/node/2600710/world, May 2025.

[30] Ceylon Today, "Kotmale Bus Tragedy Sparks Long-Overdue Transport Reforms," Available: https://ceylontoday.lk/2025/05/17/kotmale-bus-tragedy-sparks-long-overdue-transport-reforms/, May 17, 2025.

[31] The Island, "Transport sector: Glimmer of hope," Available: https://island.lk/transport-sector-glimmer-of-hope/, 2025.

[32] Daily Mirror, "AI-powered CCTV and GPS systems to be introduced," Available: https://www.dailymirror.lk/print/breaking-news/AI-powered-CCTV-and-GPS-systems-to-be-introduced/108-308968, 2025.

# Initial Discussion About the Project

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 25/05/2025<br>21:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 2h | Actual Time | 1h 56m |
| **Special Notes** | discuss project proposal, identify features, introduction, objectives, start to create proposal document | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Get an overall idea about project and start the initial step for preparing project proposal

**Notes/Clarifications:**

**Meeting Minutes:**

1h 56m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Discuss project proposal | **All participants** |
| Identify features | **D.S. Sandeepa**<br>**R.U. Gonalagoda** |
| Start to create proposal documents and presentation | **P.S. Ekanayaka** |

# Discussion About the Project with the supervisor
# - Physical meeting -

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 29/06/2025<br>21:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 1h | Actual Time | 1h |
| **Special Notes** | Discussion About the Project | | |
| **Call/Location Information** | Physical | | |
| **Supported Documents** | | | |

**Agenda:**

Discussion About the Project

**Notes/Clarifications:**

**Meeting Minutes:**

1h

**Action Items:**

| Action item | Decision made by |
|---|---|
| Discuss about the project | **All participants** |

# Review presentation and finalization

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 28/05/2025<br>21:40:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>K. J. L. Perera | | |
| **Estimated Time** | 1h | Actual Time | 45m |
| **Special Notes** | Review presentation and finalization | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Review the presentation and finalize the slides for uploading to the LMS.

**Notes/Clarifications:**

**Meeting Minutes:**

45m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review the presentation | **All participants** |
| Make changes to the slides | **D.S. Sandeepa**<br>**R.U. Gonalagoda** |

# Divide presentation scope among team members

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 17/06/2025<br>22:30:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 2h | Actual Time | 2h 10m |
| **Special Notes** | Practice proposal presentation, make some changes and divide the presentation scope among team members | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Divide the presentation scope among team members

**Notes/Clarifications:**

Divide the total presentation duration 10min into 5 slots and divide the slides to the team members according to their preference.

**Meeting Minutes:**

2h 10m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Divide the presentation scope among team members | **All members** |
| Discuss what are the new changes we should add to the presentation slides | **All members** |
| Make changes to the slides | **D.S. Sandeepa**<br>**R. U. Gonalagoda** |
| Practice proposal presentation | **All members** |

# Practice proposal presentation

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 17/06/2025<br>20:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 2h | Actual Time | 1h 48m |
| **Special Notes** | Practice proposal presentation and adjust the time duration for each member | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Practice proposal presentation and adjust the time duration for each member

**Notes/Clarifications:**

**Meeting Minutes:**

2h 10m

**Action Items:**

| Action item | Decision made by |
|---|---|
| adjust the time duration | **All members** |
| Practice proposal presentation | **All members** |

# Practice proposal presentation

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 18/06/2025<br>21:50:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 40m | Actual Time | 40m |
| **Special Notes** | Practice proposal presentation | | |
| **Call/Location Information** | Zoom | | |
| **Supported Documents** | | | |

**Agenda:**

Practice proposal presentation

**Notes/Clarifications:**

Practice proposal presentation meeting link:
https://us04web.zoom.us/j/73561182607?pwd=r6WfOT4GIU6qmcw21BriiPictODq8M.1

**Meeting Minutes:**

40m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Practice proposal presentation | **All members** |

# Initiate and setup code base

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 27/06/2025<br>21:15:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka | | |
| **Estimated Time** | 5h | Actual Time | 5h 45m |
| **Special Notes** | Initiate and setup GitHub code base | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Initiate and setup GitHub code base

**Notes/Clarifications:**

Discuss the project Trello board.
Manage project tasks and progress.
Set up projects on team members' development environments.
Resolve Git conflicts and handle merge requests.

**Meeting Minutes:**

5h 45m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Discuss the project Trello board<br>Manage project tasks and progress | **All participants** |
| Setup git projects & environments | **D.S. Sandeepa** |

# Discussion with the Supervisor

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 18/06/2025<br>21:50:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 40m | Actual Time | 40m |
| **Special Notes** | Discussion with the supervisor about progress 1 and overall project idea | | |
| **Call/Location Information** | Zoom | | |
| **Supported Documents** | | | |

**Agenda:**

Discussion with the supervisor about progress 1 and overall project idea

**Notes/Clarifications:**

Practice proposal presentation meeting link:
https://us04web.zoom.us/j/73561182607?pwd=r6WfOT4GIU6qmcw21BriiPictODq8M.1

**Meeting Minutes:**

40m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Practice proposal presentation | **All members** |

# Discussion for planning progress 1

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 01/07/2025<br>20:30:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 2h | Actual Time | 2h |
| **Special Notes** | Discuss the issues, dependencies, planning for the project progress 1 | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Discuss issues, dependencies, planning for the project progress 1

**Notes/Clarifications:**

**Meeting Minutes:**

2h

**Action Items:**

| Action item | Decision made by |
|---|---|
| Discuss the project progress 1 | **All members** |

# Review the literature review

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 06/07/2025<br>11:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 2h | Actual Time | 2h |
| **Special Notes** | Review the literature review | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Review the literature review

**Notes/Clarifications:**

**Meeting Minutes:**

2h

**Action Items:**

| Action item | Decision made by |
|---|---|
| Discuss the project progress 1 | **All participants** |

# Review of the SRS Document

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 17/07/2025<br>20:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 2h | Actual Time | 2h |
| **Special Notes** | Review of the SRS Document | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Review of the SRS Document

**Notes/Clarifications:**

**Meeting Minutes:**

2h

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review of the SRS Document | **All participants** |

# Review of the progress 1 reviewed changes – phase 1 with the supervisor

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 16/08/2025<br>22:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>P. S. Ekanayaka<br>T. P. Denagamage<br>K. J. L. Perera | | |
| **Estimated Time** | 40m | Actual Time | 40m |
| **Special Notes** | Review of the progress 1 reviewed changes | | |
| **Call/Location Information** | Zoom | | |
| **Supported Documents** | | | |

**Agenda:**

Review of progress 1 reviewed changes

**Notes/Clarifications:**

Discussion with the supervisor about progress 1 reviewed changes and enhanced the SRS and Literature review contents according to the government project.

**Meeting Minutes:**

40m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review of the SRS Document | **All Members** |

# Literature Review changes

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 15/09/2025<br>20:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 2h | Actual Time | 2h |
| **Special Notes** | Literature Review changes | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Literature Review changes

**Notes/Clarifications:**

**Meeting Minutes:**

2h

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review of the SRS Document | **All participants** |

# Review of the progress 1 reviewed changes – phase 2 with the supervisor

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 07/10/2025<br>22:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 45m | Actual Time | 45m |
| **Special Notes** | Review of the progress 1 | | |
| **Call/Location Information** | MS Teams | | |
| **Supported Documents** | | | |

**Agenda:**

Review of the progress 1

**Notes/Clarifications:**

Review SRS, Literature review, CMMI meeting minutes and other progress 1 documentation and discussion with supervisor

**Meeting Minutes:**

45m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review of the SRS Document | **All participants** |

# Discuss about the changes

| Meeting Information | | | |
|---|---|---|---|
| **Meeting Date/Time** | 07/10/2025<br>22:00:00 | | |
| **Participants** | D. S. Sandeepa<br>R. U. Gonalagoda<br>T. P. Denagamage<br>K. J. L. Perera<br>P. S. Ekanayaka | | |
| **Estimated Time** | 1h | Actual Time | 1h 30m |
| **Special Notes** | Discuss about the changes | | |
| **Call/Location Information** | WhatsApp | | |
| **Supported Documents** | | | |

**Agenda:**

Discuss about the changes

**Notes/Clarifications:**

**Meeting Minutes:**

45m

**Action Items:**

| Action item | Decision made by |
|---|---|
| Review of the SRS Document | **All participants** |

# SafeDriver Project

# Configuration Management Plan


**Prepared by: -**

CodeCrafters Team

**Team Members:**

| | |
|---|---|
| ***D. S. Sandeepa*** | ***221444563*** |
| ***R. U. Gonalagoda*** | ***621429318*** |
| ***P. S. Ekanayaka*** | ***621429276*** |
| ***K. J. L. Perera*** | ***621429815*** |
| ***T. P. Denagamage*** | ***321429305*** |


**October 2025**

**REVISION HISTORY**

| Ver. No | Date of Release | Prepared By | Approved By | List of changes from Previous Version |
|---|---|---|---|---|
| 1.0 | 08/10/2025 | R.U. Gonalagoda | D.S. Sandeepa | Initial Configuration Management Plan |

**Project Configuration Controller:** R.U. Gonalagoda

**Project Manager:** D.S. Sandeepa

**Statement of Confidentiality**

# Table of Contents

# 1. INTRODUCTION

## 1.1 Project Brief Description

**Project ID:** SDMS-2025-001

**Project Name:** SafeDriver - AI-Powered Real-Time Driver Monitoring & Accident Prevention System

The SafeDriver project is a comprehensive AI-powered solution designed specifically for Sri Lankan public transport operations. The system integrates advanced computer vision technologies with embedded hardware platforms to provide real-time driver behavior monitoring, fatigue detection, distraction identification, and GPS-integrated hazard mapping capabilities. The project addresses critical road safety challenges in Sri Lanka's public transport sector, where over two thousand road fatalities occur annually, with driver fatigue and distraction identified as primary contributing factors.

The system employs Raspberry Pi 4 as the embedded processing platform, utilizing MediaPipe for facial landmark detection, TensorFlow Lite for deep learning model deployment, Firebase for cloud infrastructure, and Flutter for cross-platform mobile application development. The solution encompasses embedded driver monitoring subsystems mounted within vehicles, cloud-based backend infrastructure for centralized data storage and analytics, cross-platform mobile applications providing passengers with real-time safety information, and comprehensive web-based dashboards enabling transport authorities to monitor compliance and review incident reports.

## 1.2 Scope

This Configuration Management Plan establishes the framework and procedures for managing all configuration items throughout the SafeDriver project lifecycle. The plan defines comprehensive processes for version control, change management, baseline establishment, configuration auditing, and release management that ensure systematic control of all project artifacts.

The Configuration Management Plan covers all software components developed for the SafeDriver system, including Python scripts implementing driver monitoring algorithms, TensorFlow Lite

deep learning models for behavioral detection, Flutter mobile application source code and assets, Firebase cloud functions and database configurations, and web-based administrative dashboard implementations. Hardware specifications and configurations are included, encompassing Raspberry Pi 4 platform specifications, camera and GPS module configurations, and alert device specifications. All project documentation falls within the scope, including Software Requirements Specifications, system architecture and design documents, test plans and specifications, user manuals and installation guides, and technical reference documentation.

The intended audience for this Configuration Management Plan includes all members of the CodeCrafters development team, the project supervisor Dr. D.D.M. Ranasinghe, quality assurance personnel, and external stakeholders including the National Transport Commission who require visibility into system configurations for regulatory compliance purposes.

The Configuration Management Plan explicitly excludes third-party commercial driver monitoring systems and their internal configurations, external transportation authority databases and their management procedures, vehicle electrical systems beyond the defined interface points with SafeDriver hardware, and personal development environments on individual team member workstations, though standardized configurations and version requirements are specified.

## 1.3 Assumptions

**1. Team Collaboration:**

All team members have reliable and consistent access to GitHub and Google Workspace throughout the project lifecycle. GitHub serves as the primary version control system for source code and documentation, while Google Workspace provides collaboration tools including Google Sheets for the CM database and Google Drive for backup storage. Team members are expected to maintain active accounts with appropriate permissions and possess basic proficiency in using these platforms for effective collaboration.

**2. Tool Availability:**

Required configuration management tools including Git, GitHub, and Firebase Console remain accessible and operational throughout the project duration from May 2025 through April 2026. This includes continuous availability of GitHub cloud infrastructure for repository hosting,

Firebase platform services for backend development, and Git client software functioning on all workstations. Tool unavailability would necessitate fallback to offline development modes and alternative synchronization mechanisms.

**3. Network Connectivity:**

Stable internet connection is available for cloud-based repository access, Firebase development and testing, automated backup operations, and real-time collaboration. While acknowledging potential intermittent connectivity issues in some locations, connectivity is generally reliable enough to support daily workflows including pushing commits, accessing cloud services, executing backups, and virtual meetings. Extended connectivity loss would impact work synchronization and backup schedules.

**4. Hardware Consistency:**

Raspberry Pi 4 hardware specifications remain consistent across all deployment units throughout pilot and initial production deployment. This ensures embedded system images developed on development hardware function identically on all units without unit-specific adjustments. Consistency includes identical Raspberry Pi models, RAM capacity, camera modules (Pi Camera Module 2 NoIR), GPS modules, and peripheral devices. Hardware variations would complicate testing and deployment procedures.

**5. Stakeholder Cooperation:**

Transport authorities including the National Transport Commission, pilot partners, and users provide timely feedback for configuration updates, report operational issues, participate in reviews and approvals, and respond to configuration change requests within reasonable timeframes. This cooperation validates system configurations meet operational and regulatory requirements. Stakeholder feedback drives refinement of alert thresholds, hazard zones, and interface elements.

**6. Power Supply:**

Embedded systems maintain consistent and reliable power supply during testing and deployment, ensuring continuous operation without interruptions that could corrupt system state or lose safety data. This encompasses stable vehicle electrical systems with appropriate voltage and current,

sufficient battery capacity, proper power conditioning, and reliable connections. Inconsistent power would require more robust power management configurations.

**7. Team Compliance:**

All team members adhere to established CM procedures, naming conventions, version control workflows, change management processes, and documentation standards. Compliance requires following Git conventions, submitting proper change requests, participating in code reviews and audits, maintaining documentation currency, and completing CM training. The Configuration Manager monitors compliance through process audits and provides additional training when needed.

These assumptions are reviewed quarterly throughout the project. Any invalid assumption triggers assessment of impacts on CM processes and implementation of mitigation strategies as necessary.

## 1.4 Definitions and Terminology

The following terms and abbreviations are used throughout this Configuration Management Plan with the specific meanings defined here.

| **Term** | **Definition** |
|---|---|
| Configuration Item (CI) | Any component of the SafeDriver system that requires version control and formal change management procedures, including software modules, hardware specifications, documentation artifacts, and system integration interfaces. |
| Baseline | A formally reviewed and approved configuration specification that serves as the reference point for further development and change control, requiring formal approval procedures before any modifications can be implemented. |
| Version Control System (VCS) | The GitHub repository infrastructure used for managing source code, documentation, and configuration specifications, maintaining complete historical records of all changes with attribution to responsible team members. |

| | |
|---|---|
| Change Request (CR) | A formal proposal to modify an existing baseline configuration, requiring documented justification, impact assessment, and approval before implementation proceeds. |
| Configuration Audit | A systematic examination of configuration items to verify compliance with specifications, confirm proper version control procedures, and validate that documentation accurately reflects implemented configurations. |
| Configuration Status Accounting | The recording and reporting of information needed to manage configurations effectively, including the status of proposed changes and the implementation status of approved changes. |
| Release | A collection of configuration items that are approved for deployment to production environments, including all necessary software, documentation, and installation materials. |

**Acronyms specific to the SafeDriver project:**

| **Acronym** | **Full Form** |
|---|---|
| SRS | Software Requirements Specification |
| HLD | High-Level Design |
| EAR | Eye Aspect Ratio |
| GPS | Global Positioning System |
| CNN | Convolutional Neural Network |
| API | Application Programming Interface |
| NTC | National Transport Commission |
| FCA | Functional Configuration Audit |
| PCA | Physical Configuration Audit |

## 1.5 References


[1] SafeDriver Software Requirements Specification (SRS) V4.0, September 2025

[2] SafeDriver Project Proposal V3.0, May 2025

[3] Literature Review: AI-Powered Real-Time Driver Monitoring V3.0, September 2025

[4] IEEE 828-2012: Standard for Configuration Management in Systems and Software Engineering

[5] Git Documentation: https://git-scm.com/doc

[6] GitHub Best Practices: https://docs.github.com

[7] Firebase Documentation: https://firebase.google.com/docs

[8] Raspberry Pi Configuration Guidelines: https://www.raspberrypi.org/documentation


# 2. ORGANIZATION & MANAGEMENT

## 2.1 Roles and Responsibilities (for CM)

The Configuration Management organizational structure for the SafeDriver project assigns specific responsibilities to ensure effective version control, change management, and quality assurance throughout the development lifecycle.

| **Role** | **Person(s) Identified** | **Responsibility** |
|---|---|---|
| **Configuration Manager** | R.U. Gonalagoda | Overall CM plan implementation and enforcement, hardware configuration documentation and control, Raspberry Pi setup standardization, GPIO configuration management, repository structure maintenance, backup verification and recovery testing, CM audit coordination, tool administration (GitHub, Google Drive), training team members on CM procedures |
| **Project Manager** | D.S. Sandeepa | CM plan approval and oversight, Change Control Board chairperson, baseline approval authority, resource |

| | | |
|---|---|---|
| | | allocation for CM activities, conflict resolution for CM issues, stakeholder communication on CM status |
| **AI/ML Model Manager** | D.S. Sandeepa | Version control for TensorFlow models, model training dataset management, model performance metrics tracking, TensorFlow Lite optimization versioning, integration of models with embedded systems |
| **Backend & Cloud Manager** | K.J.L. Perera | Firebase configuration management, cloud function versioning, database schema version control, API endpoint documentation, cloud infrastructure as code management |
| **Mobile App Manager** | K.J.L. Perera & T.P. Denagamage | Flutter application versioning, app release management (Android/iOS), mobile UI/UX configuration tracking, app store deployment coordination |
| **Documentation Manager** | T.P. Denagamage | Version control for all project documentation, report preparation and archival, meeting minutes and decision logs, user manual and technical guide versioning |
| **Testing & QA Manager** | P.S. Ekanayaka | Test case version management, test data configuration control, bug tracking and resolution documentation, testing environment configuration, field testing result archival |
| **Hardware Integration Lead** | R.U. Gonalagoda & P.S. Ekanayaka | Hardware specification documentation, component procurement tracking, wiring diagrams and setup guides, power management configuration, sensor calibration records |

## 2.2 Other Stakeholders (for Configuration Management)

Several stakeholders beyond the core configuration management team interact with configuration management processes in defined capacities.

| Stakeholder | Involvement | Key Events of Involvement |
|---|---|---|
| **Dr. D.D.M. Ranasinghe (Supervisor)** | Review and approval of major baselines | Proposal approval, Progress report reviews (3 instances), Final draft review, Final report approval |
| **National Transport Commission (NTC)** | Requirements validation and pilot feedback | Requirements gathering phase, Pilot deployment configuration, Hazard zone database updates, Compliance reporting specifications |
| **Sri Lanka Transport Board (SLTB)** | Operational requirements and deployment specifications | Initial requirements definition, Pilot testing coordination, Fleet-wide deployment planning |
| **Bus Drivers (Pilot Participants)** | User feedback for system calibration | Initial system calibration, Alert threshold adjustment, User acceptance testing |
| **Passengers (App Users)** | Mobile app feedback and feature requests | App beta testing, User interface improvements, Feature prioritization |
| **CodeCrafters Team Members** | All team members | Daily stand-ups, Weekly sprint planning, Bi-weekly integration meetings, Baseline approvals |

# 3. CONFIGURATION IDENTIFICATION & CONTROL

## 3.1 Baselines

Baselines represent formally reviewed and approved configurations that serve as reference points for subsequent development activities. The SafeDriver project establishes baselines at critical phases of the lifecycle, creating checkpoints that enable the team to maintain stable foundations while progressing through development.

| Baseline Name | Baseline ID | Target Date | Description | Approval Authority |
|---|---|---|---|---|
| Requirements Baseline | RB-001 | August 1, 2025 | Approved Software Requirements Specification encompassing all functional requirements for driver monitoring, alerting, GPS integration, mobile applications, and administrative dashboards. Includes non-functional requirements for performance, security, compatibility, and quality assurance. | Project Manager, Project Supervisor |
| Design Baseline | DB-001 | November 15, 2025 | Approved system architecture includes component specifications, interface definitions, database schemas, and API specifications. Hardware interface designs for Raspberry Pi connections. Software architecture for all system modules. | Project Manager, Configuration Manager |
| Code Freeze Baseline | CFB-001 | January 20, 2026 | Complete implementation of all planned features including driver monitoring algorithms, deep learning models, mobile applications, cloud backend services, and administrative dashboards. All source code committed and reviewed. | Project Manager, QA Lead |

| | | | | |
|---|---|---|---|---|
| Release Baseline | RB-002 | March 24, 2026 | Final approved configuration for initial deployment including all software components verified through comprehensive testing, hardware configurations validated in realistic conditions, complete documentation, and deployment packages. | Project Manager, Project Supervisor, QA Lead |

Once a baseline is established, any modifications require formal change request processing through the procedures defined in Section 4 of this plan. The Configuration Manager maintains baseline documentation including the complete list of configuration items and their versions, approval signatures and dates from all required authorities, and any known issues or limitations documented at baseline approval.

## 3.2 Configuration Items

Configuration items represent the fundamental elements under configuration management control. The SafeDriver project maintains version control for all components that contribute to system functionality or project documentation.

| **Work Item Category** | **Item Name Convention** | **Version Convention** | **Baseline Version Convention** | **Repository Location** |
|---|---|---|---|---|
| **Project Plans** | | | | |
| Project Plan | SafeDriver_Project_Plan | V X.Y | BL-X.Y-YYYYMMDD | /Documentation/Plans/ |
| Schedule Plan | SafeDriver_Schedule_Plan | V X.Y | BL-X.Y-YYYYMMDD | /Documentation/Plans/ |
| Risk Plan | SafeDriver_Risk_Plan | V X.Y | BL-X.Y-YYYYMMDD | /Documentation/Plans/ |

| | | | | |
|---|---|---|---|---|
| CM Plan | SafeDriver_CM_Plan | V X.Y | BL-X.Y-YYYYMMDD | /Documentation/Plans/ |
| Quality Plan | SafeDriver_Quality_Plan | V X.Y | BL-X.Y-YYYYMMDD | /Documentation/Plans/ |
| **Requirements** | | | | |
| SRS Document | SafeDriver_SRS | V X.Y | BL-REQ-X.Y-YYYYMMDD | /Documentation/Requirements/ |
| Use Case Diagrams | SafeDriver_Use Cases | V X.Y | BL-REQ-X.Y-YYYYMMDD | /Documentation/Requirements/ |
| Functional Requirements | SafeDriver_Functional_Req | V X.Y | BL-REQ-X.Y-YYYYMMDD | /Documentation/Requirements/ |
| **Design Documents** | | | | |
| System Architecture | SafeDriver_Architecture | V X.Y | BL-DES-X.Y-YYYYMMDD | /Documentation/Design/ |
| Hardware Design | SafeDriver_Hardware_Design | V X.Y | BL-DES-X.Y-YYYYMMDD | /Documentation/Design/ |
| Database Schema | SafeDriver_DB_Schema | V X.Y | BL-DES-X.Y-YYYYMMDD | /Documentation/Design/ |
| API Specification | SafeDriver_API_Spec | V X.Y | BL-DES-X.Y-YYYYMMDD | /Documentation/Design/ |

| | | | | |
|---|---|---|---|---|
| **Source Code - Embedded System** | | | | |
| Drowsiness Detection | drowsiness_detection.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/Monitoring/ |
| Distraction Detection | distraction_detection.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/Monitoring/ |
| GPS Integration | gps_integration.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/GPS/ |
| Alert System | alert_system.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/Alerts/ |
| Main Controller | main_controller.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/ |
| Hardware Config | hardware_config.py | Git commit hash | tag: v X.Y.Z | /Code/Embedded/Config/ |
| **AI/ML Models** | | | | |
| Drowsiness Model | drowsiness_model_vX.tflite | V X.Y | BL-MODEL-X.Y-YYYYMMDD | /Models/Drowsiness/ |
| Distraction Model | distraction_model_vX.tflite | V X.Y | BL-MODEL-X.Y-YYYYMMDD | /Models/Distraction/ |
| Training Scripts | train_[model_name].py | Git commit hash | tag: v X.Y.Z | /Code/Training/ |
| Training Datasets | dataset_[type]_vX | V X | BL-DATA-X-YYYYMMDD | /Data/Training/ |

| | | | | |
|---|---|---|---|---|
| Model Metrics | metrics_[model]_vX.json | V X.Y | BL-MODEL-X.Y-YYYYMMDD | /Models/Metrics/ |
| **Backend & Cloud** | | | | |
| Firebase Config | firebase_config.json | V X.Y | BL-CLOUD-X.Y-YYYYMMDD | /Code/Backend/Config/ |
| Cloud Functions | function_[name].js | Git commit hash | tag: v X.Y.Z | /Code/Backend/Functions/ |
| API Endpoints | api_[endpoint].js | Git commit hash | tag: v X.Y.Z | /Code/Backend/API/ |
| Database Rules | database_rules.json | V X.Y | BL-CLOUD-X.Y-YYYYMMDD | /Code/Backend/Config/ |
| **Mobile Application** | | | | |
| Flutter App | SafeDriver_App | V X.Y.Z (build) | BL-APP-X.Y.Z-YYYYMMDD | /Code/Mobile/ |
| UI Screens | screen_[name].dart | Git commit hash | tag: v X.Y.Z | /Code/Mobile/lib/screens/ |
| App Configuration | app_config.dart | Git commit hash | tag: v X.Y.Z | /Code/Mobile/lib/config/ |
| **Hardware Specifications** | | | | |
| RaspberryPi Config | RPi_Configuration_vX | V X.Y | BL-HW-X.Y-YYYYMMDD | /Hardware/Config/ |

| | | | | |
|---|---|---|---|---|
| Wiring Diagrams | Wiring_Diagram_vX | V X.Y | BL-HW-X.Y-YYYYMMDD | /Hardware/Diagrams/ |
| Component List | Component_BOM_vX | V X.Y | BL-HW-X.Y-YYYYMMDD | /Hardware/BOM/ |
| Setup Procedures | Setup_Guide_vX | V X.Y | BL-HW-X.Y-YYYYMMDD | /Hardware/Guides/ |
| **Testing** | | | | |
| Test Plans | SafeDriver_Test_Plan | V X.Y | BL-TEST-X.Y-YYYYMMDD | /Testing/Plans/ |
| Test Cases | TestCase_[Module]_vX | V X.Y | BL-TEST-X.Y-YYYYMMDD | /Testing/TestCases/ |
| Test Results | TestResults_[Date] | Date stamp | BL-TEST-X.Y-YYYYMMDD | /Testing/Results/ |
| Bug Reports | Bug_[ID]_[Description] | Status + Date | N/A | GitHub Issues |
| **Reviews** | | | | |
| Requirements Review | Review_Requirements_vX | V X.Y | BL-REV-X.Y-YYYYMMDD | /Reviews/Requirements/ |
| Design Review | Review_Design_vX | V X.Y | BL-REV-X.Y-YYYYMMDD | /Reviews/Design/ |
| Code Review | Review_Code_[Module]_vX | V X.Y | BL-REV-X.Y-YYYYMMDD | /Reviews/Code/ |
| **Reports & Deliverables** | | | | |
| Progress Report 1 | SafeDriver_PR1 | V X.Y | BL-PR1-X.Y-20250720 | /Documentation/Reports/ |

| | | | | |
|---|---|---|---|---|
| Progress Report 2 | SafeDriver_PR 2 | V X.Y | BL-PR2-X.Y-20250928 | /Documentation/Reports/ |
| Progress Report 3 | SafeDriver_PR 3 | V X.Y | BL-PR3-X.Y-20251212 | /Documentation/Reports/ |
| Final Draft Report | SafeDriver_Fin al_Draft | V X.Y | BL-FD-X.Y-20260331 | /Documentation/Reports/ |
| Final Report | SafeDriver_Fin al_Report | V X.Y | BL-FR-X.Y-20260424 | /Documentation/Reports/ |
| Reflective Log | SafeDriver_Ref lective_Log | V X.Y | BL-RL-X.Y-20260424 | /Documentation/Reports/ |

**Version Numbering Convention:**

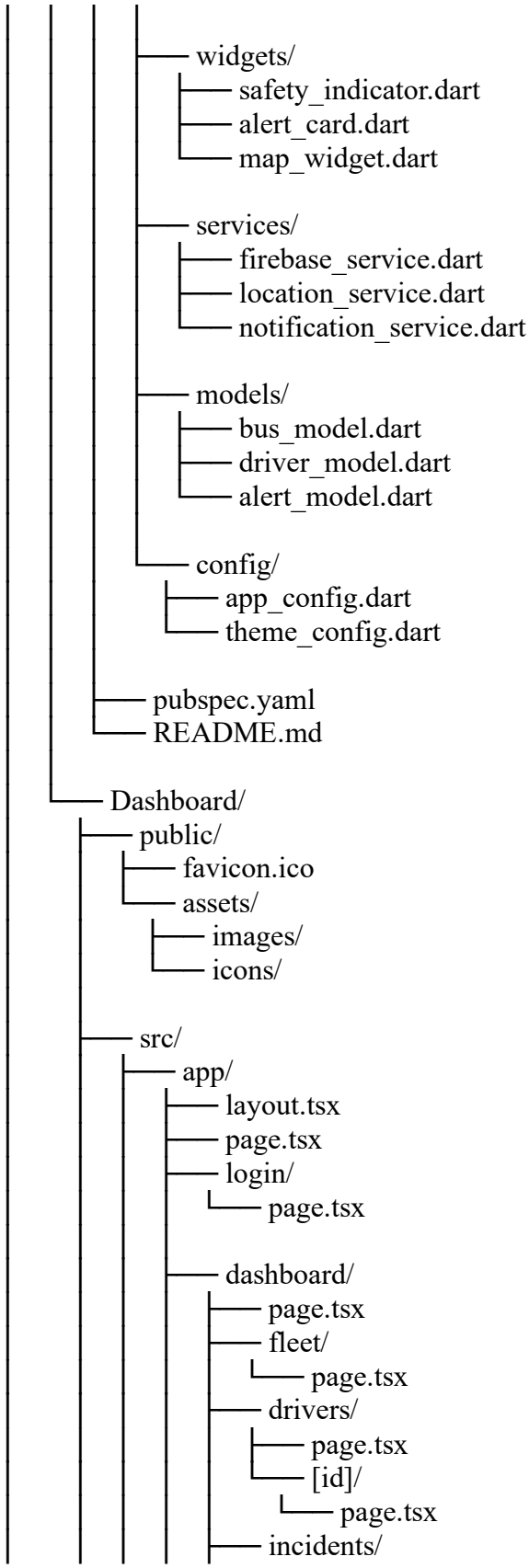
- **X.Y Format (Documents):** X = Major version, Y = Minor version
    - Major version increments for significant content changes or baseline approvals
    - Minor version increments for minor edits, corrections, formatting changes
- **X.Y.Z Format (Code/Apps):** X = Major, Y = Minor, Z = Patch
    - Major: Breaking changes or major feature additions
    - Minor: New features, backward compatible
    - Patch: Bug fixes, small improvements
- **Git Commit Hash:** Unique identifier for each code commit
- **Date Stamp:** YYYYMMDD format for time-based identification

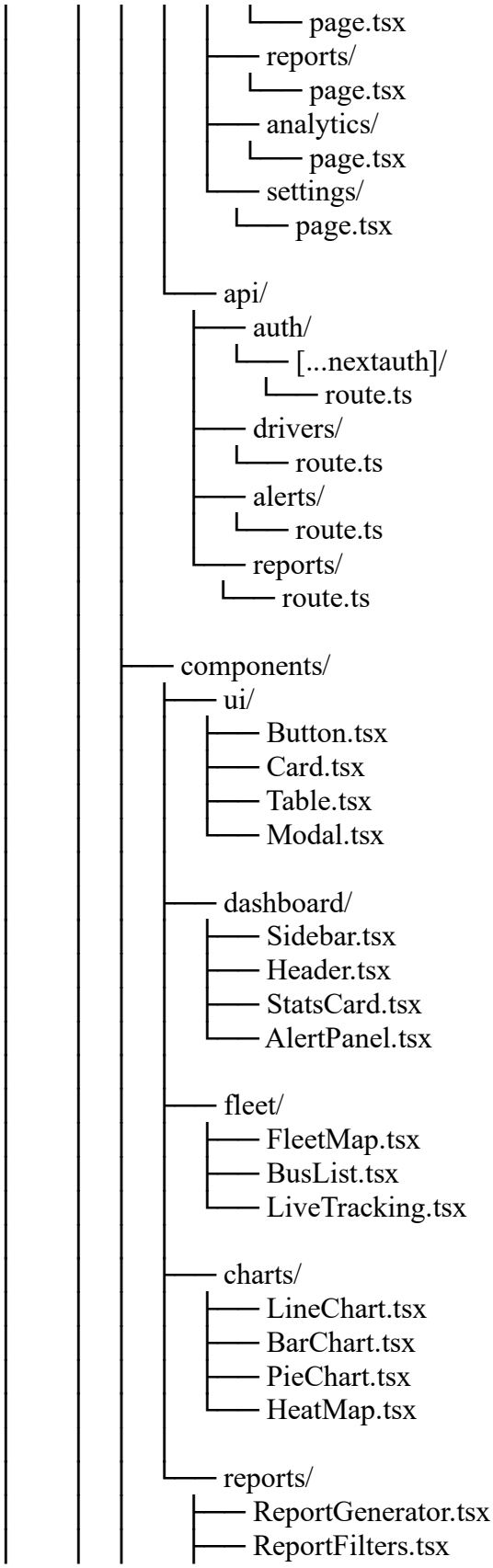
## 3.3 Directory Structure

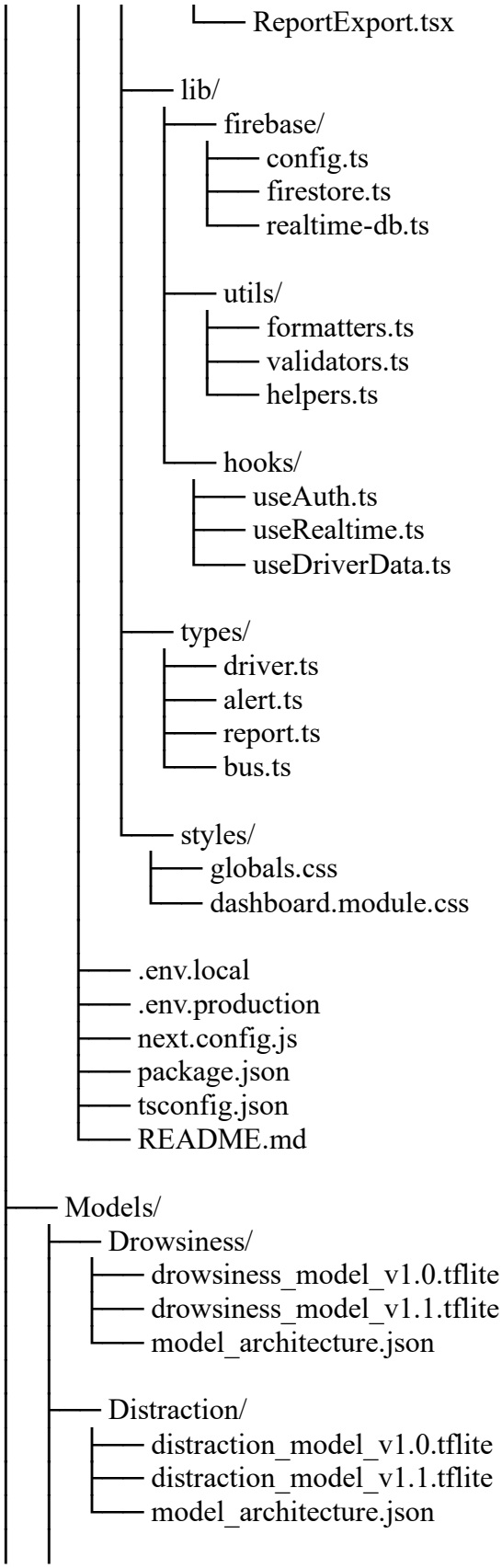
The GitHub repository for the SafeDriver project follows a logical directory structure that organizes configuration items according to their type and role in the system architecture.

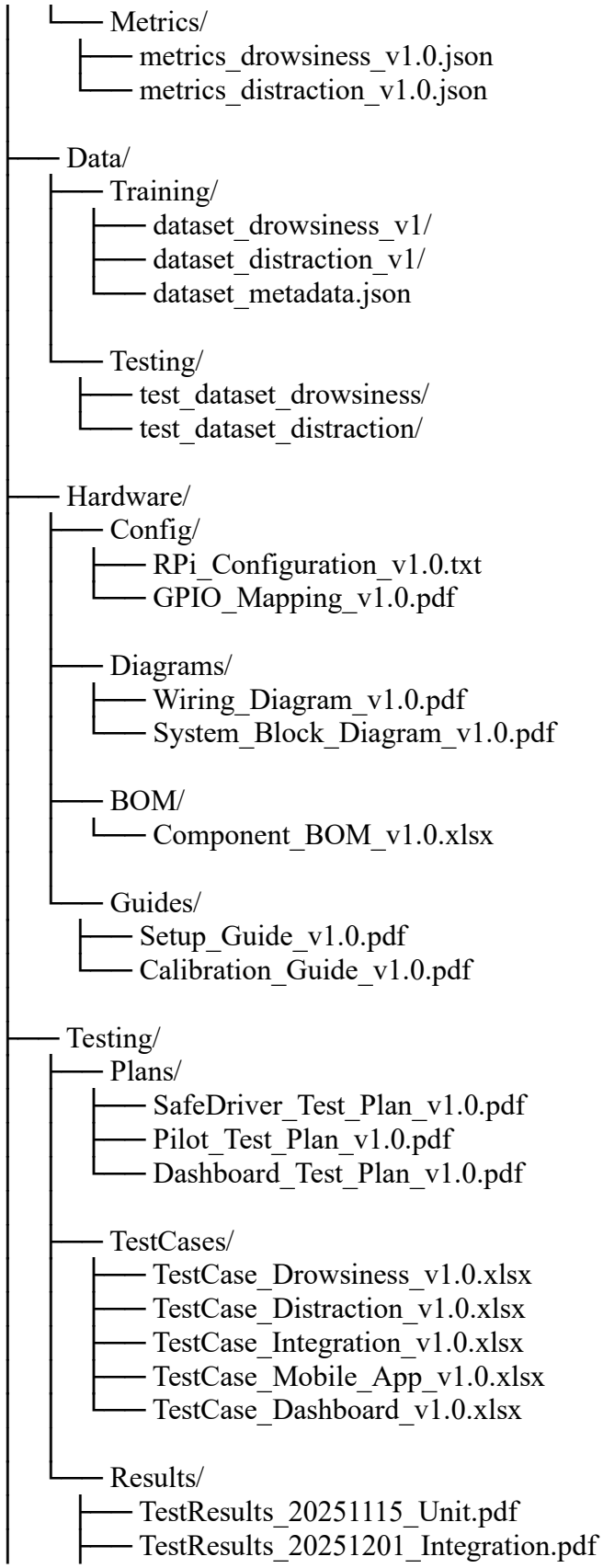
```
SafeDriver_Project/
│
├── Documentation/
│   ├── Plans/
│   │   ├── SafeDriver_Project_Plan_vX.Y.pdf
│   │   ├── SafeDriver_Schedule_Plan_vX.Y.pdf
│   │   ├── SafeDriver_Risk_Plan_vX.Y.pdf
│   │   ├── SafeDriver_CM_Plan_vX.Y.pdf
│   │   └── SafeDriver_Quality_Plan_vX.Y.pdf
│   │
│   ├── Requirements/
│   │   ├── SafeDriver_SRS_vX.Y.pdf
│   │   ├── SafeDriver_UseCases_vX.Y.pdf
│   │   └── SafeDriver_Functional_Req_vX.Y.pdf
│   │
│   ├── Design/
│   │   ├── SafeDriver_Architecture_vX.Y.pdf
│   │   ├── SafeDriver_Hardware_Design_vX.Y.pdf
│   │   ├── SafeDriver_DB_Schema_vX.Y.pdf
│   │   └── SafeDriver_API_Spec_vX.Y.pdf
│   │
│   ├── Reports/
│   │   ├── SafeDriver_PR1_vX.Y.pdf
│   │   ├── SafeDriver_PR2_vX.Y.pdf
│   │   ├── SafeDriver_PR3_vX.Y.pdf
│   │   ├── SafeDriver_Final_Draft_vX.Y.pdf
│   │   ├── SafeDriver_Final_Report_vX.Y.pdf
│   │   └── SafeDriver_Reflective_Log_vX.Y.pdf
│   │
│   └── UserManuals/
│       ├── Driver_Manual_vX.Y.pdf
│       ├── Administrator_Manual_vX.Y.pdf
│       ├── Dashboard_User_Guide_vX.Y.pdf
│       └── Installation_Guide_vX.Y.pdf
│
├── Code/
│   ├── Embedded/
│   │   ├── Monitoring/
│   │   │   ├── drowsiness_detection.py
│   │   │   ├── distraction_detection.py
│   │   │   └── smoking_detection.py
```

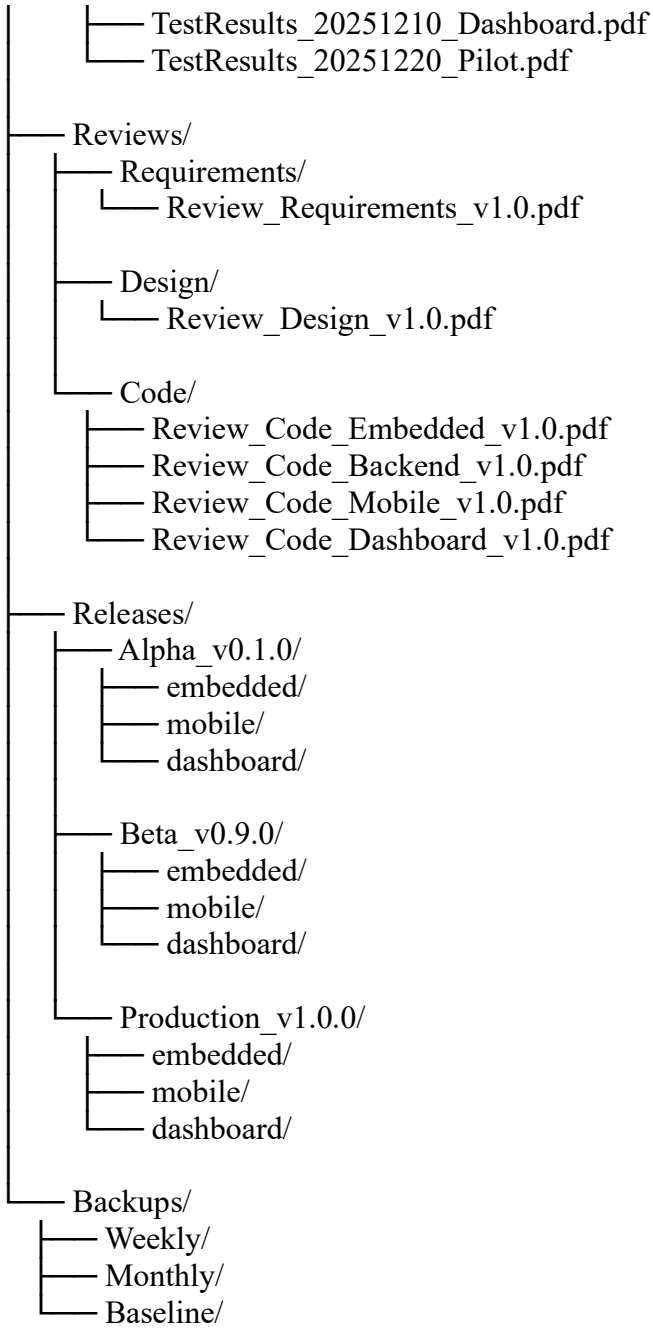
```
│   │   ├── GPS/
│   │   │   ├── gps_integration.py
│   │   │   └── hazard_mapping.py
│   │   ├── Alerts/
│   │   │   ├── alert_system.py
│   │   │   └── escalation_logic.py
│   │   ├── Config/
│   │   │   ├── hardware_config.py
│   │   │   └── system_settings.json
│   │   ├── main_controller.py
│   │   └── requirements.txt
│   ├── Training/
│   │   ├── train_drowsiness_model.py
│   │   ├── train_distraction_model.py
│   │   ├── data_preprocessing.py
│   │   └── model_evaluation.py
│   ├── Backend/
│   │   ├── Functions/
│   │   │   ├── function_driver_alert.js
│   │   │   ├── function_incident_logging.js
│   │   │   └── function_report_generation.js
│   │   ├── API/
│   │   │   ├── api_driver_data.js
│   │   │   ├── api_alerts.js
│   │   │   └── api_reports.js
│   │   └── Config/
│   │       ├── firebase_config.json
│   │       └── database_rules.json
│   ├── Mobile/
│   │   ├── android/
│   │   ├── ios/
│   │   ├── lib/
│   │   │   ├── screens/
│   │   │   │   ├── screen_home.dart
│   │   │   │   ├── screen_safety_status.dart
│   │   │   │   ├── screen_bus_tracking.dart
│   │   │   │   └── screen_feedback.dart
```


widgets/
safety_indicator.dart
alert_card.dart
map_widget.dart
services/
firebase_service.dart
location_service.dart
notification_service.dart
models/
bus_model.dart
driver_model.dart
alert_model.dart
config/
app_config.dart
theme_config.dart
pubspec.yaml
README.md
Dashboard/
public/
favicon.ico
assets/
images/
icons/
src/
app/
layout.tsx
page.tsx
login/
page.tsx
dashboard/
page.tsx
fleet/
page.tsx
drivers/
page.tsx
[id]/
page.tsx
incidents/


page.tsx
reports/
page.tsx
analytics/
page.tsx
settings/
page.tsx
api/
auth/
[...nextauth]/
route.ts
drivers/
route.ts
alerts/
route.ts
reports/
route.ts
components/
ui/
Button.tsx
Card.tsx
Table.tsx
Modal.tsx
dashboard/
Sidebar.tsx
Header.tsx
StatsCard.tsx
AlertPanel.tsx
fleet/
FleetMap.tsx
BusList.tsx
LiveTracking.tsx
charts/
LineChart.tsx
BarChart.tsx
PieChart.tsx
HeatMap.tsx
reports/
ReportGenerator.tsx
ReportFilters.tsx


ReportExport.tsx
lib/
firebase/
config.ts
firestore.ts
realtime-db.ts
utils/
formatters.ts
validators.ts
helpers.ts
hooks/
useAuth.ts
useRealtime.ts
useDriverData.ts
types/
driver.ts
alert.ts
report.ts
bus.ts
styles/
globals.css
dashboard.module.css
.env.local
.env.production
next.config.js
package.json
tsconfig.json
README.md
Models/
Drowsiness/
drowsiness_model_v1.0.tflite
drowsiness_model_v1.1.tflite
model_architecture.json
Distraction/
distraction_model_v1.0.tflite
distraction_model_v1.1.tflite
model_architecture.json


Metrics/
metrics_drowsiness_v1.0.json
metrics_distraction_v1.0.json
Data/
Training/
dataset_drowsiness_v1/
dataset_distraction_v1/
dataset_metadata.json
Testing/
test_dataset_drowsiness/
test_dataset_distraction/
Hardware/
Config/
RPi_Configuration_v1.0.txt
GPIO_Mapping_v1.0.pdf
Diagrams/
Wiring_Diagram_v1.0.pdf
System_Block_Diagram_v1.0.pdf
BOM/
Component_BOM_v1.0.xlsx
Guides/
Setup_Guide_v1.0.pdf
Calibration_Guide_v1.0.pdf
Testing/
Plans/
SafeDriver_Test_Plan_v1.0.pdf
Pilot_Test_Plan_v1.0.pdf
Dashboard_Test_Plan_v1.0.pdf
TestCases/
TestCase_Drowsiness_v1.0.xlsx
TestCase_Distraction_v1.0.xlsx
TestCase_Integration_v1.0.xlsx
TestCase_Mobile_App_v1.0.xlsx
TestCase_Dashboard_v1.0.xlsx
Results/
TestResults_20251115_Unit.pdf
TestResults_20251201_Integration.pdf




Access to directories is controlled through GitHub repository permissions and branch protection rules to ensure appropriate security and workflow enforcement. Project plans and requirements directories allow read access for all team members, with write access restricted to appropriate roles through pull request mechanisms. Development directories permit module-specific developers to commit changes via pull requests, with peer review and Configuration Manager approval required before merging. Release directories maintain strict access control, with only the Configuration Manager authorized to create and publish release packages.

## 3.4 Backup & Recovery

Comprehensive backup and recovery procedures protect the SafeDriver project against data loss from hardware failures, human error, or malicious actions.

| Backup Type | Items Backed Up | Frequency | Retention Period | Storage Location |
|---|---|---|---|---|
| GitHub Repository | Complete repository with all branches and history | Continuous (distributed) | Permanent | GitHub cloud + local developer clones |
| Firebase Database | Real-time database, security rules, indexes | Daily at 03:00 AM | Daily: 30 days, Weekly: 3 months, Monthly: 1 year | Google Cloud Storage |
| Hardware Specs | All hardware configuration documents | Weekly on Sundays | All versions permanently | GitHub + Google Drive |
| Test Data | Test datasets, execution logs, validation results | Weekly after test cycles | Current + 2 previous cycles | GitHub + Local storage |
| Release Packages | Complete release artifacts for all versions | Upon release creation | All releases permanently | GitHub Releases + Backup drive |

Backup integrity is verified through regular testing to ensure recovery procedures function correctly when needed. Weekly validation includes repository clone testing, where the Configuration Manager performs fresh clones to isolated directories to verify complete history availability. Monthly database restores tests confirm Firebase backup exports can be successfully restored to test environments. Bi-weekly configuration file recovery validates that random samples restore correctly with matching checksums.

**Recovery Procedures:** In the event of local repository corruption, team members rename the corrupted directory to preserve uncommitted work, perform fresh clones from GitHub remote repository, review corrupted directories for important uncommitted changes, and manually apply necessary changes to fresh clones before resuming development. For accidental file deletion or overwrites, the last known good version is identified from Git history and restored using appropriate Git commands. In the extremely unlikely event of complete GitHub repository loss, the Configuration Manager identifies the most recent complete local clone among team members, creates a new GitHub repository, pushes the complete repository from the local clone, and notifies all team members to update their remote URLs.

## 3.5 Release Procedure

The SafeDriver project employs structured release processes ensuring that all deployed configurations meet quality standards and safety requirements. Release procedures differ across system components reflecting their distinct deployment environments while maintaining consistent quality standards.

**Release Types:**

| Release Type | Components | Frequency | Approval Requirements |
|---|---|---|---|
| Major Release | All system components coordinated | Milestone-based (April 2026 for v1.0) | Project Manager, QA Lead, Project Supervisor |
| Minor Release | Individual components with backward-compatible updates | As needed (monthly expected) | Configuration Manager, Component Lead |
| Patch Release | Critical bug fixes and security updates | As needed (within 48 hours for critical) | Configuration Manager (expedited) |

**Embedded System Release Procedure:** The Configuration Manager creates a release branch from the approved Code Freeze Baseline, with all approved changes merged through pull requests. Version numbers are updated in relevant files and release notes drafted documenting changes, known issues, and installation instructions. The build process prepares the base Raspberry Pi OS

image, installs Python dependencies including OpenCV, MediaPipe, and TensorFlow Lite, deploys driver monitoring scripts with appropriate permissions, installs deep learning models in correct directories, configures system startup scripts for automatic initialization, and applies security hardening.

Quality assurance testing validates the system image through installation on test hardware, functional testing according to test plans, performance validation under various environmental conditions, integration testing with cloud backend and mobile applications, and security assessment. Upon successful testing, the QA Lead reviews results, Configuration Manager verifies proper versioning, Project Manager authorizes deployment, and the release is tagged in Git repository and uploaded to GitHub Releases. Deployment proceeds with pilot installation in limited vehicles, careful adherence to installation documentation, post-installation validation, and monitoring period with feedback collection before broader phased deployment.

**Mobile Application Release Procedure:** Flutter application versions are updated in pubspec.yaml, platform-specific build numbers incremented, and release notes prepared. Build generation produces release APK for Android signed with release keystore, App Bundle for Play Store, and release IPA for iOS signed with distribution certificate. Quality assurance executes the mobile application test suite on physical devices, performance testing under various network conditions, compatibility testing across device models and OS versions, and security and accessibility testing. Store submission follows Google Play and Apple App Store procedures with localized content in Sinhala, Tamil, and English. Deployment begins with staged rollout at ten percent of user base, careful monitoring of crash reports and feedback, gradual increase to full rollout, and communication through in-app notifications.

**Cloud Backend Release Procedure:** Cloud functions code undergoes review and approval via pull requests, database security rules are validated in emulator environments, and environment variables validated. Staging deployment occurs first, with comprehensive testing before production deployment during low-traffic periods. The Configuration Manager deploys to production Firebase project, monitors deployment progress, verifies successful deployment, executes smoke tests, and monitors system behavior for 24 to 48 hours with rollback plan prepared if issues are detected.

## 3.6 CHANGE MANAGEMENT

Change management procedures ensure that all modifications to baselined configurations are properly evaluated, approved, documented, and implemented while maintaining system stability and quality.

**Change Request Submission:** Any project stakeholder may submit a change request using the standardized form. Each request must include unique identifier, submission date, submitter contact information, affected configuration items, detailed change description, rationale and justification, change type classification (Critical, Major, Minor, Editorial), and priority assignment (High, Medium, Low).

**Change Classification:**

| Change Type | Definition | Processing Time |
|---|---|---|
| Critical | Affects safety-critical functionality; addresses security vulnerabilities; resolves issues preventing system operation | Immediate (< 24 hours) |
| Major | Introduces new features; modifies system architecture; changes user interfaces significantly; impacts multiple components | 5-10 business days |
| Minor | Improves performance without changing functionality; resolves non-critical defects; enhances usability | 3-5 business days |
| Editorial | Corrects documentation errors; updates formatting; clarifies content without technical changes | 1-2 business days |

**Change Impact Analysis:** The Configuration Manager coordinates impact analysis involving appropriate technical specialists, evaluating technical impact on components and interfaces, resource requirements including development and testing effort, schedule impact on milestone dates and dependencies, risk assessment identifying new risks and safety implications, and quality impact determining testing scope and documentation updates.

**Change Approval Process:**

| Change Type | Approval Requirements |
|---|---|
| Critical | Configuration Manager review and recommendation; Project Manager approval with documented justification; QA Lead review confirming safety analysis adequacy; Stakeholder notification when applicable |
| Major | Configuration Manager review and impact assessment; Affected module leads review; Team discussion in weekly integration meeting; Project Manager approval |
| Minor | Configuration Manager review; Affected module lead approval |
| Editorial | Document owner approval; Configuration Manager notification |

**Change Implementation:** Once approved, changes are assigned to appropriate developers who create feature branches, implement changes following coding standards, conduct self-testing and documentation updates, submit pull requests referencing change request identifiers, undergo peer review by at least one team member, receive QA testing appropriate to change scope, are merged by Configuration Manager after approval, undergo verification by QA Lead in integrated environment, and are closed by Configuration Manager with archived documentation.

The Configuration Manager tracks metrics to monitor change management effectiveness including average processing time, change request backlog size, approval rate, emergency change frequency, implementation success rate, and re-opened request percentage.

# 5. CONFIGURATION AUDITS

Configuration audits provide independent verification that configuration management procedures are followed correctly and that deliverable systems conform to approved specifications.

| CM Audit # | Date / Trigger Event | Audit Type | Focus of Audit | Auditors | Baseline |
|---|---|---|---|---|---|
| FCA-001 | December 15, 2025 | Functional | System architecture compliance with requirements; Interface specifications completeness; Database schema alignment | D.S. Sandeepa (Lead), T.P. Denagamage | DB-001 |
| PCA-001 | January 15, 2026 | Physical | Raspberry Pi hardware configuration; Camera and GPS module installation; Alert device connections | R.U. Gonalagoda (Lead), P.S. Ekanayaka | Hardware Integration |
| FCA-002 | March 01, 2026 | Functional | Driver monitoring algorithms accuracy; Distraction detection effectiveness; Alert escalation functionality; Mobile app features | R.U. Gonalagoda (Lead), P.S. Ekanayaka, K.J.L. Perera | CFB-001 |
| PCA-002 | March 15, 2026 | Physical | Embedded system image contents; Mobile application packages; Cloud backend configurations; | K.J.L. Perera (Lead), T.P. Denagamage | RB-002 |

| | | | | | |
|---|---|---|---|---|---|
| | | | Documentation completeness | | |
| FCA-003 | March 30, 2026 | Functional | End-to-end system functionality; Integration across components; Performance under realistic conditions; Safety compliance | Dr. D.D.M. Ranasinghe (Lead), D.S. Sandeepa, R.U. Gonalagoda , P.S. Ekanayaka, K.J.L. Perera, T.P. Denagamage | RB-002 |
| PCA-CM | Quarterly | Process | CM process compliance; Version control practices; Change management adherence; Documentation currency | T.P. Denagamage (Lead), Configuration Manager | N/A |

**Audit Procedures:** Each configuration audit follows standardized procedures. Pre-audit activities include preparing audit packages two weeks before scheduled dates with agendas, checklists, configuration item lists, and relevant specifications. Audit execution begins with opening meetings establishing scope and schedule, followed by examination of configuration items against specifications. For Functional Audits, teams review test documentation, observe functional demonstrations, verify requirements traceability, and assess performance data. For Physical Audits, teams inspect hardware configurations, examine software artifacts, validate documentation accuracy, and verify deployment materials. For Process Audits, teams review CM records, interview team members, sample version control operations, and assess procedure compliance.

Post-audit activities involve preparing audit reports with findings classified by severity (Critical, Major, Minor, Observation), assigning corrective actions with responsible parties and due dates, distributing reports to stakeholders, tracking corrective action completion by Configuration Manager, and conducting follow-up audits if critical or multiple major findings identified.

**Audit Success Criteria:**

| Audit Type | Pass Criteria | Fail Criteria |
|---|---|---|
| Functional Configuration Audit | No critical findings; ≤ 2 major findings; All requirements traced; All test results documented | Any critical findings; > 5 major findings; Significant requirements not implemented |
| Physical Configuration Audit | Physical items match specifications; Documentation accurate; All artifacts present | Configuration does not match specs; Major documentation errors; Critical artifacts missing |
| Process Audit | CM procedures followed consistently; No process violations; Records current and complete | Systematic procedure violations; Records incomplete or inaccurate |

# 6. TOOLS AND RESOURCES

The SafeDriver project employs a comprehensive set of tools supporting configuration management activities. Tool selection prioritizes open-source solutions where it is feasible to minimize licensing costs while ensuring adequate functionality.

**Configuration Management Tools:**

| Tool Category | Tool Name | Version | Purpose | License/Cost | Procurement |
|---|---|---|---|---|---|
| **Version Control** | GitHub | Cloud | Primary version control, collaborative development platform | Free for academic | May 2025 |

| | | | | | |
|---|---|---|---|---|---|
| | Git Client | 2.40+ | Local version control operations | Open source (GPL) | May 2025 |
| **Development** | Python | 3.9+ | Embedded system development | Open source (PSF) | May 2025 |
| | Visual Studio Code | Latest | Code editor and IDE | Free | May 2025 |
| | TensorFlow | 2.18.0 | Model training | Open source (Apache 2.0) | June 2025 |
| | Google Colab | Cloud | GPU resources for model training | Free tier | June 2025 |
| | Flutter SDK | 3.27.2 | Mobile app development | Open source (BSD) | June 2025 |
| | Android Studio | Latest | Android development and testing | Free | June 2025 |
| | Firebase CLI | Latest | Backend development and deployment | Free | June 2025 |
| **Testing** | pytest | Latest | Python unit testing | Open source (MIT) | July 2025 |
| | Flutter Test | Built-in | Mobile app testing | Included with Flutter | July 2025 |

| | | | | | |
|---|---|---|---|---|---|
| | Firebase Emulator | Latest | Local backend testing | Free | July 2025 |
| **Collaboration** | Microsoft Teams | Cloud | Team communication and coordination | University license | May 2025 |
| | Google Workspace | Cloud | Document collaboration | Free | May 2025 |
| | Google Sheets | Cloud | CM database | Free | May 2025 |
| **Documentation** | Markdown Editors | Various | Documentation authoring | Free/Open source | May 2025 |
| | Draw.io | Web/Desktop | Diagram creation | Free | May 2025 |
| | Pandoc | Latest | Document format conversion | Open source (GPL) | June 2025 |
| **Hardware Tools** | Raspberry Pi Imager | Latest | OS installation on Raspberry Pi | Free | January 2026 |
| | PuTTY / OpenSSH | Latest | Remote access to Raspberry Pi | Free/Open source | January 2026 |

**Configuration Management Database:** A lightweight CM database is implemented using Google Sheets to track configuration items, baselines, change requests, and audit findings.

| Database Component | Purpose | Update Frequency | Access Control |
|---|---|---|---|
| Configuration Item Registry | Complete list of all CIs with current versions, status, ownership | Daily during active development | All: View, CM: Edit |
| Baseline Records | Documentation of all established baselines with approval signatures | Upon baseline establishment | All: View, CM/PM: Edit |
| Change Request Log | Tracking of all change requests from submission to closure | Daily as requests processed | All: View/Submit, CM: Edit |
| Audit Findings | Record of all audit findings with corrective actions and status | After each audit, weekly updates | All: View, CM/QA: Edit |
| Release Registry | Complete history of all releases with deployment status | Upon each release | All: View, CM: Edit |

**Reference Resources:**

| Resource Type | Title/Description | Purpose | Location |
|---|---|---|---|
| Standards | IEEE 828-2012, ISO/IEC 27001:2013, ISO 26262 | CM framework, security, functional safety guidelines | References/Standards/ |
| Books | "Git for Teams", "Continuous Delivery" | Git workflow and release management best practices | Team library (digital) |

| | | | |
|---|---|---|---|
| Templates | Change Request Form, Baseline Approval Form, Audit Report Template | Standardized documentation | Appendix A of this plan |
| Online Resources | GitHub Documentation, Firebase Documentation, Flutter Documentation | Development guidance | Web access |

The Configuration Manager maintains a tool inventory with version tracking and update schedules to ensure security patches are applied promptly and compatibility maintained across the team. Security updates for development tools are checked weekly with immediate application for critical patches. Git clients, Firebase SDK, and Flutter SDK receive updates monthly or quarterly as needed for features and security.

# 6. CM MILESTONES

Configuration management milestones align with overall project milestones to ensure that configuration control procedures support rather than impede development progress.

| **Milestone** | **Date** | **CM Activities** | **Completion Criteria** | **Deliverables** |
|---|---|---|---|---|
| **CM Plan Approval** | October 15, 2025 | CM Plan review and approval; GitHub repository establishment; Tool procurement and setup; Team training | Approved CM Plan with signatures; Operational repository with branch protection; All team members have tool access; Initial training completed | Approved CM Plan; GitHub repository URL; Tool access list; Training records |

| | | | | |
|---|---|---|---|---|
| **Requirements Baseline** | October 15, 2025 | SRS review and approval; Requirements traceability matrix creation; Baseline establishment (RB-001) | All SRS documents approved; Traceability established; RB-001 formally established | Approved SRS V4.0; Traceability matrix; Baseline approval form |
| **Design Baseline** | November 15, 2025 | Architecture review; FCA-001 audit; Design baseline establishment (DB-001) | Design documents approved; FCA-001 passed; DB-001 established | Approved design documents; FCA-001 report; Design baseline approval |
| **Hardware Integration** | December 31, 2025 | PCA-001 audit; Hardware documentation; Component specifications | PCA-001 passed; Hardware documented; Integration tests passed | PCA-001 report; Hardware specs; Integration test results |
| **Code Freeze** | February 15, 2026 | Code freeze baseline (CFB-001); FCA-002 audit; Feature completion verification | All features implemented; CFB-001 established; FCA-002 passed | Code freeze baseline; FCA-002 report; Feature completion matrix |
| **Pre-Release Audit** | March 15, 2026 | PCA-002 audit; Release package preparation; Documentation verification | PCA-002 passed; Release packages prepared; Documentation finalized | PCA-002 report; Release packages; Complete documentation |

| | | | | |
|---|---|---|---|---|
| **Final Release** | March 30, 2026 | Release baseline (RB-002); FCA-003 audit; Deployment authorization | RB-002 established; FCA-003 passed; All approvals obtained | Release baseline approval; FCA-003 report; Release notes |
| **Post-Deployment** | April 30, 2026 | CM effectiveness review; Lessons learned; Process improvements | Review completed; Lessons documented; Improvements prepared | CM effectiveness report; Lessons learned; Improvement plan |

**Milestone Entry and Exit Criteria:** Each milestone has defined prerequisites and completion requirements. For example, the Requirements Baseline requires completed stakeholder requirements gathering and documented SRS before entry, and must achieve SRS approval by all stakeholders, traceability matrix establishment, and formally approved RB-001 baseline before exit. The Design Baseline requires established requirements baseline and completed design work before entry, and must achieve design approval, pass FCA-001 audit, and established DB-001 baseline before exit. Similar criteria apply to all other milestones ensuring systematic progression through project phases.

# 7. TRAINING

Effective configuration management requires that all team members understand and properly apply the procedures defined in this plan. The SafeDriver project implements a comprehensive training program ensuring team members possess necessary knowledge and skills.

**Training Requirements by Role:**

| Role | Required Training | Duration | Timing | Delivery |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| All Team Members | CM concepts; GitHub basics; Change management; Document version control; Backup awareness | 4 hours | Within 1 week of project start | In-person workshop + hands-on |
| Configuration Manager | Advanced Git operations; Branch management; Audit procedures; CM database administration; Release management | 8 hours | Before CM Plan approval | Self-study + mentor guidance |
| Developers | Git workflow and branching; Pull request procedures; Code review practices; Commit conventions | 3 hours | Before development start | Hands-on workshop |
| QA Lead | Audit procedures and checklists; Test configuration management; Baseline verification | 4 hours | Before first audit | Guided training + documentation |

**Initial Training Program:** All team members complete initial CM training within one week of joining the project.

**Session 1: CM Fundamentals (2 hours)**

- Configuration management concepts and benefits
- SafeDriver project CM organization and roles
- Configuration items and baselines explained
- Change management overview
- Question and answer session

**Session 2: Version Control with Git and GitHub (2 hours)**

- Git basics: repositories, commits, branches, merges
- GitHub collaboration: pull requests, code reviews, issues
- SafeDriver branching strategy and workflow
- Hands-on exercises: cloning, making changes, submitting PR
- Common problems and troubleshooting

**Training Materials Provided:**

- CM Plan document (this document)
- Git Quick Reference Guide
- GitHub workflow diagrams
- Cheat sheets for common Git commands
- Links to online tutorials

**Training Validation:** Training completion is verified through practical exercises where each trainee successfully demonstrates cloning the repository, creating a feature branch, making a commit with proper message format, submitting a pull request, and participating in code review. A brief knowledge check discussion covers when to submit change requests, how to identify configuration items, baseline approval processes, and where to find CM documentation. Team members receive full repository and tool access only after successfully completing training.

**Ongoing Training:** Quarterly refresher sessions ensure team members remain current with CM procedures and address confusion. Sessions held in January and April 2026 review recent audit findings, discuss Git workflow challenges, update baseline management procedures, cover release procedures, and assess change management effectiveness. Each session lasts thirty minutes and includes metrics review, challenge discussion, best practice sharing, Q&A, and procedure updates.

**Knowledge Transfer:** When team members join mid-project, the Configuration Manager conducts one-on-one orientation covering project-specific CM practices, assigns mentorship pairing with experienced members for two weeks, requires reading of CM Plan and relevant documentation, supervises first several Git operations, and validates competency before granting full repository access.

# 8. CONFIGURATION AUDITING

Configuration auditing provides systematic examination of configuration management processes and products to ensure compliance with specifications and procedures. This section consolidates audit information for the SafeDriver project.

**Number of Audits:** Six major audits are scheduled during the project lifecycle: three Functional Configuration Audits (FCA-001, FCA-002, FCA-003), two Physical Configuration Audits (PCA-001, PCA-002), and quarterly Process Audits (PCA-CM).

**Audit Details:**

**FCA-001 (December 15, 2025) - Design Baseline Audit**

- **Tied to Baseline:** DB-001 (Design Baseline)
- **Who Performs:** D.S. Sandeepa (Lead), T.P. Denagamage
- **What is Audited:** System architecture compliance with requirements; Interface specifications completeness; Database schema alignment with data requirements; Design traceability to requirements; Design documentation quality

**PCA-001 (January 15, 2026) - Hardware Integration Audit**

- **Tied to Baseline:** Hardware Integration Milestone
- **Who Performs:** R.U. Gonalagoda (Lead), P.S. Ekanayaka
- **What is Audited:** Raspberry Pi hardware configuration verification; Camera Module 2 NoIR installation and mounting; GPS module integration and operation; Audio buzzer installation and volume; Power supply specifications; Wiring and connections; Hardware documentation completeness

**FCA-002 (March 30, 2026) - Code Freeze Audit**

- **Tied to Baseline:** CFB-001 (Code Freeze Baseline)
- **Who Performs:** R.U. Gonalagoda (Lead), P.S. Ekanayaka, K.J.L. Perera

- **What is Audited:** Driver monitoring algorithms accuracy and performance; Distraction detection effectiveness; Alert escalation functionality through all levels; GPS hazard integration; Mobile application feature completeness; Firebase backend service functionality; Test coverage and results; Code quality metrics

**PCA-002 (April 15, 2026) - Pre-Release Audit**

- **Tied to Baseline:** RB-002 (Release Baseline)
- **Who Performs:** K.J.L. Perera (Lead), T.P. Denagamage
- **What is Audited:** Embedded system image contents and configuration; Mobile application packages (Android APK and iOS IPA); Cloud backend configurations and deployment scripts; Documentation completeness and accuracy (user manuals, installation guides, technical reference); Training materials; Release notes; Version consistency across components

**FCA-003 (April 20, 2026) - Final Release Audit**

- **Tied to Baseline:** RB-002 (Release Baseline)
- **Who Performs:** Dr. D.D.M. Ranasinghe (Lead), D.S. Sandeepa, T.P. Denagamage
- **What is Audited:** Complete integrated system functionality end-to-end; Integration verification across all components; Performance under realistic operating conditions; Safety requirements compliance; User acceptance criteria satisfaction; Regulatory compliance with NTC requirements; Deployment readiness assessment

**PCA-CM (Quarterly: January, April, July, October 2026) - Process Audits**

- **Tied to Baseline:** N/A (Process Audit)
- **Who Performs:** T.P. Denagamage (Lead), Configuration Manager
- **What is Audited:** Configuration management process compliance; Git commit history and message conventions; Pull request and code review records; Change request processing and documentation; Baseline management and approval records; CM database currency and completeness; Backup logs and validation records; Training completion records; Team member adherence to CM procedures

**Audit Report Structure:** Each audit produces a formal report documenting audit scope and objectives, audit team composition, list of audited configuration items, audit findings classified by severity (Critical, Major, Minor, Observation), recommended corrective actions with assigned responsibilities and due dates, overall assessment and conclusions, and approval signatures from audit lead, Configuration Manager, and Project Manager.

**Finding Severity Definitions:**

- **Critical:** Configuration item does not meet safety requirements; Major functionality missing or incorrect; Security vulnerability; Showstopper defect. Requires immediate resolution; blocks baseline approval.
- **Major:** Significant functionality deficiency; Performance does not meet specifications; Documentation substantially inaccurate or incomplete. Requires resolution before baseline approval.
- **Minor:** Non-critical functionality issues; Minor documentation errors; Process deviations with minimal impact. Can defer to next iteration but must be documented and tracked.
- **Observation:** Process improvement opportunities; Best practice recommendations; No immediate action required. Documented for consideration in process improvement meetings.

**Corrective Action Tracking:** All audit findings are entered into the CM database tracking system with finding ID, severity classification, description, affected configuration item, corrective action assignment, responsible party, due date, and status. The Configuration Manager monitors corrective action progress with weekly status updates on open findings. Findings are closed only after verification of corrective action completion. Monthly metrics reports to Project Manager track closure rates and trends.

# APPENDIX A: Configuration Management Forms

## CHANGE REQUEST FORM - SafeDriver Project

Change Request ID: CR-YYYY-NNN (Assigned by CM)
Date Submitted: …….
Submitted By: ……
Contact: …….

**CHANGE DESCRIPTION**
Affected Configuration Items:

☐ Requirements ☐ Design ☐ Source Code ☐ Test Artifacts

☐ Hardware Specs ☐ Documentation ☐ Other: ……

Detailed Description: ……..……
…….
Rationale/Justification:……
…….

**CLASSIFICATION**
Change Type: ☐ Critical ☐ Major ☐ Minor ☐ Editorial
Priority: ☐ High ☐ Medium ☐ Low

**IMPACT ANALYSIS (Configuration Manager)**
Technical Impact: …….
Resource Requirements: …….
Schedule Impact: ……
Risk Assessment: …….

**APPROVAL**
Configuration Manager: …… Date: ……
Recommendation: ☐ Approve ☐ Reject ☐ Defer

Project Manager: ……. Date: ……
Decision: ☐ Approved ☐ Rejected ☐ Deferred

**IMPLEMENTATION**
Assigned To: ……...
Target Date: ……
Actual Completion: …….

Status: ☐ Submitted ☐ Under Review ☐ Approved ☐ In Progress

☐ In Review ☐ Verified ☐ Closed

# BASELINE APPROVAL FORM - SafeDriver Project

Baseline ID: …… (e.g., RB-001, DB-001, CFB-001)
Baseline Name: ……
Baseline Date: ……

**CONFIGURATION ITEMS INCLUDED**

| CI Name | Version | Location |
|---|---|---|
| | | |
| | | |

**QUALITY ASSURANCE**

Test Results Summary: ……
……
Functional Configuration Audit: ☐ Passed ☐ Failed ☐ N/A
Audit ID: ……. Date: …….

Physical Configuration Audit: ☐ Passed ☐ Failed ☐ N/A
Audit ID: ……. Date: …….

Outstanding Issues: …….....
……

**APPROVAL SIGNATURES**

Configuration Manager: ……Date: …….
Project Manager: …… Date: …….....
QA Lead: …… Date: ……...
Project Supervisor (if required): …… Date: …….....

This baseline is hereby established as the reference configuration for subsequent development activities.

Configuration Manager: …… Date: ……

# CONFIGURATION AUDIT REPORT - SafeDriver Project

Audit ID: …….

Audit Type: ☐ Functional ☐ Physical ☐ Process

Audit Date: …….

Audit Lead: …….

**AUDIT SCOPE**

Configuration Items Audited: ……..……

Audit Objectives: ……...……

**AUDIT TEAM**

Lead Auditor: ……...……

Team Members: ……...…….

**AUDIT FINDINGS**

Finding #1

Severity: ☐ Critical ☐ Major ☐ Minor ☐ Observation

Description: ……...……

Configuration Item: ……...……

Corrective Action: ……...……

Assigned To: ……. Due Date: …….

[Additional findings as needed]

**AUDIT CONCLUSIONS**

Overall Assessment:

- ☐ Satisfactory - No critical or major findings
- ☐ Satisfactory with Observations - Minor findings only
- ☐ Unsatisfactory - Critical or major findings require resolution

Recommendations: ……...……

**APPROVAL**

Audit Lead: ……. Date: ……...

Configuration Manager: …… Date: ……

Project Manager: ……. Date: …….