



Kingdom of Saudi Arabia
Ministry of Higher Education
Imam Abdulrahman Bin Faisal University
College of Computer Sciences & Information Technology

Final Proposal Report

Title: Harnessing Deep Learning to Optimize Emergency Response

*A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Artificial Intelligence*

By:

#	Name	ID	Role
1	Mohammed Kamal Hadi	2220004360	Leader
2	Abdulrhman Mohammed Alshehri	2220006250	Member
3	Saud Ali Rawdhan	2220002478	Member
4	Ahmad Abdulqader Alakhdar	2220002324	Member
5	Ali Ibrahim Asiri	2220000306	Member

Supervised by:

Dr. Mohammad Aftab Alam Khan (Supervisor) and Dr. Atta-ur-Rahman (Co-supervisor).

Committee Member Names:

Dr. Ito Wasio and Dr. Rabab Alkhalifa

Date:

17 December, 2025

ACKNOWLEDGEMENT

We want to sincerely thank our supervisor, Dr. Mohammed Aftab Alam Khan, for all the guidance and support he's given us throughout this project proposal. His valuable insights and expert advice on the technical side really helped shape our research. We also deeply appreciate our co-supervisor, Dr. Atta-ur-Rahman, for his constant encouragement and patience along the way. Their feedback and suggestions have been incredibly helpful and will make a big difference in improving our work.

ABSTRACT

*Car collisions are among the most severe and urgent problems for public health where every minute lost in emergency response can swing the scale from life to death for the injured. Presently, the systems are more dependent on the calls made by humans, which commonly cause delays because of the lack of information or the ignorance of the situation, thus aggravating the traffic and causing other accidents. The scheme called, "**Harnessing Deep Learning to Optimize Emergency Response**" puts forth an artificial intelligence-powered resolution that automatically detects accidents in real-time from the feeds of live traffic and security cameras, and notifies the responders at once with their exact location, extent, and visual proof to reduce response times dramatically.*

The methodology utilizes the Waterfall development model, which provides a structured way of moving from one stage to another in the project; this model was chosen because it is compatible with the open-source tools that are going to be used, namely, Python and YOLOv8 for object detection, CNNs for pattern recognition, OpenCV for video processing, and Flask for the alert dashboard. The system is trained on a variety of public datasets that include, among others, the 5,700-video traffic accident collection, and it aims at beyond 2-seconds detection latency and 85%+ precision/recall/F1-score, thus, filling the gaps left by previous studies like lack of real-time CCTV validation and bad weather handling issues.

The system's immediate life-saving potential is not its only benefit, as it also lessens the urban traffic mess by making quick rerouting possible and integrates with Saudi Arabia's Vision 2030 for smarter, safer cities through scalable infrastructure support. The system of the project thus shifts the burden of human error in reporting from its shoulders and benefits the users with instant intelligence, collaborates with the reliable coordination of emergencies, and advocates for polices with support of data, moreover, the country's high-risk areas are laid open for such systems to cover them.

Table of Contents

Chapter 1: Introduction.....	11
1.1 Introduction	11
1.2 Problem Statement.....	11
1.3 Motivation	12
1.4 Justification.....	12
1.5 Aims & Objectives	12
1.6 Scope / Limitation of the Study	12
1.7 Social, Professional, Legal, and Ethical Implications	13
1.8 Project Organization	13
Chapter 2: Background and Literature Review	15
2.1 Introduction	15
2.2 Literature Review	15
Chapter 3: Software Project Management Plans (SPMP)	28
3.1 Project Overview	28
3.1.1 Purpose, Scope, and Objectives.....	28
3.1.2 Assumptions, Constraints and Risks.....	29
3.1.3 Project Deliverables.....	30
3.1.4 Schedule and Budget Summary.....	31
3.1.5 Evolution of the Plan	32
3.1.6 References.....	34
3.1.7 Definitions and Acronyms	34
3.1.8 Document Structure	35
3.2 Project Organization	36
3.2.1 External Interfaces	36
3.2.2 Internal Structure	36
3.2.3 Roles and Responsibilities	37
3.3 Managerial Process Plans	38
3.3.1 Startup Plan.....	38
3.3.2 Work Plan	39
3.3.3 Project Tracking Plan	43
3.3.4 Risk Management Plan	45
3.3.5 Project Closeout Plan.....	46
3.4 Technical Process Plans.....	46
3.4.1 Process Model.....	46
3.4.2 Methods, Tools, and Techniques	48
3.4.3 Infrastructure.....	49

3.4.4 Product Acceptance	49
3.5 Supporting Process Plans.....	49
3.6 Additional Plans	50
3.6.1 Data Privacy and Security	50
3.6.2 Documentation and User Operation	50
3.6.3 Maintenance and Updates.....	50
3.6.4 Deployment Considerations.....	50
3.6.5 Ethical and Legal Considerations	50
Chapter 4: Software Requirements Specification (SRS).....	51
4.1 Introduction	51
4.1.1 Purpose	51
4.1.2 Scope.....	51
4.1.3 Definitions, Acronyms, and Abbreviations	51
4.1.4 References.....	52
4.2 Overall description	53
4.2.1 Product perspective.....	53
4.2.2 Product functions	53
4.3 Specific requirements	57
4.3.1 External interface requirements	57
Chapter 5: Software Design Specification (SDS).....	66
5.1 Introduction	66
5.1.1 Purpose	66
5.1.2 Scope.....	66
5.1.3 Definitions, Acronyms, and Abbreviations	66
5.1.4 References.....	67
5.2 System overview.....	69
5.2.1 System Functionality	69
5.2.2 User Functionality	69
5.3 Design Considerations.....	71
5.3.1 Assumptions and Dependencies	71
5.3.2 General Constraints	73
5.3.3 Security and Threat Considerations	74
5.4 User Interface Design	75
5.4.1 Overview of User Interface	75
5.4.2 Interface Design Rules.....	75

5.4.3 Screen Images	76
5.4.4 Screen Objects and Actions	81
5.4.5 Other Interfaces	83
5.5 System Architecture	85
5.5.1 Architectural Design Approach	85
5.5.2 Architectural Design.....	86
5.5.3 Subsystem Architecture	88
5.6 Data Design	92
5.6.1 Data Description	92
5.6.2 Data Dictionary.....	92
5.6.3 Database Description.....	96
5.7 Component Design	98
5.7.1 Detection Engine Component.....	98
5.7.2 Event Orchestrator Component	98
5.7.3 Assignment Engine Component	98
5.7.4 Notification Component	99
5.7.5 Repository Layer (Data Access Component)	99
5.8 Detailed System Design.....	100
5.8.1 Classification, Definition and Responsibilities.....	100
5.8.2 Composition.....	101
5.8.3 Uses/Interactions.....	101
5.8.4 Resources.....	102
5.8.5 Processing	103
5.8.6 Interface/Exports.....	106
5.8.7 Detailed Subsystem Design	107
5.9 Other Design Features	109
5.10 Requirements Traceability Matrix.....	110
Chapter 6: Conclusion	114
6.1 Introduction	114
6.2 Findings And Contributions	114
6.2.1 Future Findings	114
6.2.2 Contributions	115
6.3 Limitations.....	116
6.4 Lesson & Skills Learned.....	117
6.5 Recommendations for future works	117
References.....	118

Table of Tables

Table 1: List of Abbreviations.....	10
Table 2: Literature Review Summary.....	27
Table 3: Key projects risks	30
Table 4: Project Deliverables.....	31
Table 5: Schedule summary.....	32
Table 6: Acronyms	35
Table 7: Roles and Responsibilities.....	37
Table 8: Human Resources and Phase Duration.....	39
Table 9: Required Skills for Project Roles	39
Table 10: Work Breakdown Structure.....	41
Table 11: Schedule Allocation (First Semester).....	43
Table 12: Technical and Informational Resources	43
Table 13: Key Project Metrics	45
Table 14: Comprehensive Risk Management Plan.....	45
Table 15:The Waterfall Phases	48
Table 16: Used Tools.....	48
Table 17: The Essential Requirements	49
Table 18: Supporting Process Plans	50
Table 19: Glossary of Acronyms and Abbreviations	52
Table 20: Login Interface	57
Table 21: Admin Dashboard Interface	58
Table 22: Camera Management Interface.....	59
Table 23: User Management Interface	59
Table 24: Responder Dashboard Interface	61
Table 25: Accident Alert Interface	62
Table 26: Accident Alert Interface	63
Table 27: Glossary of Terms and Acronyms.....	67
Table 28: STRIDE Threat Model	74
Table 29: User Interface Controls and Interactions.....	83
Table 30: System Data Dictionary.....	96
Table 31: Component Definitions and Responsibilities.	101
Table 32: Component Constraints and Conditions.....	101
Table 33: Resource Allocation and Concurrency Management.....	103
Table 34: Process Specification of Video Acquisition & Detection.	103
Table 35: Process Specification of Emergency Event Management.	104
Table 36: Process Specification of Emergency Units Management.....	105
Table 37: Process Specification of Notification & Alerting.....	105

Table 38: Process Specification of UI Dashboard.....	106
Table 39: Process Specification of Data Management & Storage.....	106
Table 40: System Interface Specifications.	107
Table 41: Requirements Traceability Matrix Linking SRS Functional Requirements to SDS System Components	113

Table of Figures

Figure 1: Internal structure	37
Figure 2: Gantt Chart: Project Schedule.....	44
Figure 3: The Waterfall Model.....	46
Figure 4: Admin Use Case.....	56
Figure 5: First Responders Use Case.....	56
Figure 6: Admin workflow system.....	60
Figure 7: The Responders Login Flow	61
Figure 8: AI Alert System Flow	64
Figure 9: Secure emergency response login screen with system branding and role-based sign-in.	76
Figure 10: Reset Password Interface.	77
Figure 11: System Administrator Operational Dashboard.	77
Figure 12: Camera Management Interface.	78
Figure 13: User Management Interface.	78
Figure 14: System Audit Log Interface.	79
Figure 15: First Responder Active Incidents Dashboard.....	79
Figure 16: First Responder Incident Reports Interface.....	80
Figure 17: Accident Details Interface.....	80
Figure 18: Physical Entity Relationship Diagram.	96
Figure 19: ERD Mapping Relational Schema.	97
Figure 20: Sequence of Interactions Across Subsystems.	102

List of abbreviations

AI	Artificial Intelligence
YOLO	You Only Look Once (Object Detection Model)
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
UAV	Unmanned Aerial Vehicle
LSTM	Long Short-Term Memory (Neural Network type)
GPS	Global Positioning System
STP	Software Testing Plan
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
SDS	Software Design Specification
MTT	Motion Temporal Templates (used in motion tracking)
CNN Encoder	Convolutional Neural Network Encoder
ViT	Vision Transformer
BPM	Business Process Management (related to drone incident response)
LLM	Large Language Model
TAD	Traffic Accidents Dataset
TAA	Traffic Accident Anticipation
DeepSORT	Deep Learning-based Sorting Algorithm for tracking multiple objects

Table 1: List of Abbreviations.

Chapter 1: Introduction

This chapter presents an overview of the project, highlighting the key details and examining the scope and limitations of the addressed problem. It also outlines the main causes and reasons behind the issue, along with the goals this project aims to achieve. The chapter wraps up by explaining how the project is structured to guide the reader through what's coming next.

1.1 Introduction

Our project, Optimizing Emergency Response, is all about helping emergency teams react to car accidents faster and more effectively. The project uses advanced computer vision and AI to monitor live video feeds from traffic and security cameras. Whenever our system detects that an accident has happened, it instantly sends an alert to emergency services, no waiting for someone to call it in. These alerts include key details like the location and severity of the accident, giving police, paramedics, and firefighters the head start they need.

At the core of our solution are AI models trained to spot crash patterns and unusual road behavior. Using technologies such as YOLO and convolutional neural networks, our system analyzes footage in real time, recognizing collisions within seconds. Once detected, surrounding authorities are notified immediately so they can act quickly and prepare the right resources.

The benefits go beyond saving lives. Rapid detection helps traffic authorities reroute vehicles to ease congestion around accident scenes, which reduces the risk of secondary crashes and keeps the roads moving. It's a step toward building safer, smarter cities where technology actively works to protect people.

By removing human errors from accident reporting and delivering precise, timely information, the system empowers first responders to make better decisions under pressure. The result is faster rescues, less chaos, and a more coordinated emergency response.

Optimizing Emergency Response isn't just a technological upgrade, it's a vision for how AI can strengthen public safety and make our cities more resilient. We're showing what's possible when innovation directly serves the well-being of communities.

1.2 Problem Statement

Every second matters when it comes to responding to car accidents, but emergency teams often lose time because reports come in too slowly or with incomplete details. In many cases, crashes go unreported for several minutes, time that can make a life-or-death difference. Even once help is on the way, traffic congestion and unsafe driving around emergency vehicles can slow responders down further, putting both victims and rescuers at greater risk.

Right now, emergency responses still depend heavily on human calls and eyewitness accounts. That system leaves too much room for delays and errors. Our project tackles this challenge by using AI and computer vision to detect accidents the moment they happen. By analyzing live video feeds in real time, the system can instantly alert emergency teams with accurate information on location and severity, helping them mobilize immediately.

1.3 Motivation

- Help emergency teams spot accidents faster so they can save lives.
- Stop delays by automatically detecting accidents instead of waiting for calls.
- Give clearer, more accurate info to first responders.
- Reduce traffic jams caused by accidents for smoother roads.
- Support Saudi Arabia's Vision 2030 goal of safer roads and smarter cities.
- Use modern AI technology to improve public safety.
- Make emergency response quicker, smarter, and more reliable.

1.4 Justification

This project matters because fast and accurate accident detection can make the difference between life and death. Today, emergency teams usually rely on people calling in accidents, which often leads to delays or missing details. By detecting accidents automatically and sending out alerts right away, responders can act faster and more effectively. The initiative also aligns with Saudi Arabia's Vision 2030, supporting the country's goal of building safer, smarter cities through advanced technology. It closes a critical gap in how emergencies are reported and managed, offering a practical solution that benefits both first responders and the public they serve.

1.5 Aims & Objectives

The goal of this project is to create a smart system that can spot car accidents as they happen by using advanced computer vision technology. By detecting accidents in real time, the system will help emergency services respond faster, save lives, and improve overall road safety. It will also deliver precise details about the accident's location, giving first responders the information they need to prepare effectively before reaching the scene.

To make this possible, the project will build a system that continuously watches live videos from surveillance cameras, analyzes them using powerful deep learning models, and identifies accidents within seconds. Once an accident is detected, alerts will be sent immediately to emergency crews along with key information, while traffic authorities can reroute vehicles to prevent congestion. In doing so, the project aims to boost the efficiency of emergency response and contribute to Saudi Arabia's Vision 2030 by promoting safer roads and smarter cities.

1.6 Scope / Limitation of the Study

This project aims to develop a smart system that can detect car accidents in real time using video feeds from surveillance cameras. Its focus includes identifying accidents as they happen, sending instant alerts to emergency services, and helping manage traffic flow around the affected area.

There are, however, a few limitations to consider. The system's performance depends heavily on the availability and clarity of camera footage, which can be influenced by factors like weather, lighting conditions, and camera angles. Privacy and data security regulations may also restrict access to certain

video sources. In addition, the detection models might occasionally generate false alarms or miss complex accident situations, and hardware issues could impact reliability.

Lastly, this project is designed mainly for urban traffic settings, so results may differ in rural or less monitored regions.

1.7 Social, Professional, Legal, and Ethical Implications

This project aims to save lives by speeding up emergency response and improving road safety. It quickly detects accidents, protecting drivers, passengers, and pedestrians while giving emergency teams accurate, real-time information to act faster. It also supports smarter city planning through valuable traffic data.

Legally and ethically, the project must protect privacy and secure all video data. It should follow regulations, use technology responsibly, minimize errors, and ensure transparency and accountability to maintain public trust.

1.8 Project Organization

The organization of this project is structured into eight main chapters, each focusing on important aspects of the development and implementation process:

❖ Chapter 1: Introduction

This chapter provides a clear overview of the project, outlining the problem, its scope, limitations, motivations, and objectives. It also explains why this project is important and how it fits within broader goals like Saudi Arabia's Vision 2030.

❖ Chapter 2: Background and Review of Literature

Here, the project examines existing research and technologies related to accident detection, AI, and computer vision, giving context to our approach and highlighting advancements and gaps.

❖ Chapter 3: Software Project Management Plans (SPMP)

This section details the project's scope, timeline, deliverables, risks, assumptions, and resource planning to ensure smooth execution.

❖ Chapter 4: Methodology/Software Requirement Specification (SRS)

This chapter explains the technical approach, including AI models, data requirements, and system functionalities that are necessary to achieve the project's goals.

❖ Chapter 5: Software Design Specification (SDS)

Here, the architecture of the system is described in detail, outlining how components like video processing, accident detection algorithms, alert mechanisms, and user interfaces interact.

❖ Chapter 6: Implementation

This chapter covers the practical development of the system, including training AI models, integrating with hardware, testing for accuracy and reliability, and refining based on results.

❖ **Chapter 7: Software Testing Plan (STP)**

This chapter outlines how the system will be tested to ensure it works correctly and reliably. It covers the testing methods for all key components, verifying accuracy, performance, and user experience.

❖ **Chapter 8: Conclusion**

The final chapter summarizes the project's outcomes, key achievements, limitations, and lessons learned. It also suggests areas for future improvements and research.

Chapter 2: Background and Literature Review

This chapter gives an overview of past research on using artificial intelligence and computer vision for real-time accident detection and emergency response. It summarizes the methods used, the datasets and video sources tested, and how each system's performance was evaluated. The review also points out the main results and limitations found in previous studies. By doing so, it helps identify what has already been achieved, where the gaps remain, and how these insights shape the design of our own accident detection system.

2.1 Introduction

This literature review examines key studies on using artificial intelligence for real-time traffic accident detection and emergency response. The research review explores different techniques, including deep learning, computer vision, and multimodal methods, applied to surveillance footage, sensor data, and emergency management platforms.

These studies also highlight common challenges, such as limited datasets, the difficulty of predicting accidents, maintaining model accuracy, and integrating solutions into smart city systems. By identifying current trends, proven methods, and existing gaps, this review provides valuable guidance for designing an effective accident detection system that supports the goals of Saudi Arabia's Vision 2030.

2.2 Literature Review

Adewopo et al. tackled one of the biggest challenges in AI-based accident detection, the lack of large, diverse datasets. [16] They compiled around 5,700 accident samples from multiple sources, combining aerial TrafficCam footage with ground-level DashCam videos to capture different accident scenarios. Their dataset enabled advanced models like I3D-CONVLSTM2D to achieve higher accuracy in detecting accidents as they occur. [16]

Bakheet and Al-Hamadi developed a fast, efficient detection system that uses motion temporal templates (MTTs) and fuzzy time-slicing to record and interpret motion changes. [17] By turning motion tracking into a classification task, their adaptive deep neural network reached a 98.5% accuracy rate in real-time conditions, demonstrating strong potential for practical deployment in vehicles. [17]

Liao et al. introduced CRASH, a framework designed for autonomous vehicles that recognizes and anticipates collisions. [18] It uses spatial-temporal attention modules to focus on high-risk objects and a frequency-domain context-aware module to detect subtle visual cues like lane markings. Tested on real-world data, CRASH performed well even with missing information, showing its promise for proactive accident prevention. [18]

Behboudi et al. reviewed 191 studies on accident risk prediction, severity assessment, and duration modeling. [19] Their findings highlight deep neural networks as the most effective for managing complex data sources, including weather conditions and social media inputs. They emphasize the importance of integrating predictive systems into traffic management while noting ongoing challenges like unbalanced datasets and the need for more adaptive models. [19]

Zhang et al. developed SeeUnsafe, a next-generation framework that uses multimodal large language models to create an interactive accident analysis system. [20] It organizes and prioritizes video events by severity, uses visual prompts for precise localization, and allows engineers to interact through conversational queries. With a success rate above 51%, SeeUnsafe represents a major step toward explaining and managing accidents through natural language-based systems. [20]

Karim et al. created a fast traffic incident detection system by combining YOLOv8 with Deep-SORT for multi-object tracking. [1] Tested on urban video datasets, the system achieved strong precision and recall, proving effective in controlled environments. However, the study was limited by a small sample size, no cross-dataset validation, and insufficient testing of how emergency services could use it in real time. [1]

Chen et al. addressed the shortage of labeled accident data by developing a generative adversarial network (GAN) that produces realistic synthetic traffic features such as speed and acceleration patterns. [2] This helps train classifiers and improves accident prediction accuracy, but the focus on trajectory data instead of raw video limits scene reconstruction and generalization. [2]

Arefin et al. built a quick accident detection pipeline using updated YOLO models applied to a large, annotated image dataset. [3] Their system offers fast, consistent detection for vision-based monitoring, though it relies on still images instead of continuous video, has limited testing in harsh conditions, and lacks validation on lightweight hardware. [3]

Ahmed et al. proposed a hybrid real-time vision system that combines YOLO detection with DeepSORT tracking and adds modules for classifying accident severity and detecting fires after collisions. [4] It performed well on CCTV datasets in varied scenarios but has not been tested on uninterrupted video streams or benchmarked for resource efficiency on lightweight devices. [4]

Ayesha et al. introduced CIRS, a multi-agent machine learning framework using YOLOv11 for frame-level accident detection, VideoLLaMA3 for semantic captioning, and an agent-based layer to coordinate alerts. [5] The system worked well on CCTV datasets but faced constraints such as low training sampling rates, domain bias from mixed data sources, and no resilience testing for continuous video or deployment on edge devices. [5]

Fang et al. differentiate between Traffic Accident Detection (TAD), which focuses on identifying accidents as they occur, and Traffic Accident Anticipation (TAA), which aims to predict them before they happen. [21] They highlight how rare and complex traffic accidents are, stressing challenges like timing, uncertainty, and fair evaluation. By reviewing 31 public datasets, they propose a structured framework that helps researchers compare models more consistently, encouraging better reporting on accuracy, detection timing, and responsiveness.

Akter, Shihab, and Sharma explore how foundational models such as large language models and vision-language models can be applied to crash analysis. Their review examines techniques that combine visual and textual information for tasks like captioning, question answering, and causal reasoning. [21]

Zhou et al. introduce an enhanced YOLOv9 model integrated with a Generalized Efficient Layer Aggregation Network (GELAN) to detect traffic accidents in real time. [10] Trained on a varied dataset of accident images, the model reached 94.2% precision after 100 epochs, outperforming previous YOLO versions in both recall and average precision. This improvement shows strong potential for faster and more accurate emergency response systems. [10]

Omari Alaoui et al. present a wide-ranging review of vision systems used in emergency vehicles, charting the shift from handcrafted features and Support Vector Machines (SVMs) to deep learning approaches like Convolutional Neural Networks (CNNs), Vision Transformers (ViT), Siamese networks, and GAN-based data augmentation. [24] They highlight how combining robust detectors with context encoders improves recognition and prioritization in dense urban areas. The study also identifies ongoing challenges related to system scalability, latency, and the need for standardized multimodal testing frameworks. [24]

Nishanth et al. design a simple, low-cost accident detection solution using classical computer vision tools available in OpenCV. [22] Centered on motion cues and object tracking, their system runs efficiently in real time and is ideal for small-scale or resource-limited environments. While it struggles with occlusion, lighting changes, and limited temporal understanding, it provides a solid baseline for comparing the progress and advantages of newer deep learning or multimodal methods. [22]

Elhag et al. suggest a new approach for handling accident reports by bringing drones into Saudi Arabia's emergency response system. [6] Their tests show that drones can speed up incident response by about 40%, especially during busy travel times, by quickly capturing and sharing accident information. This demonstrates how adding drones can directly improve data collection and response times, supporting the country's smart city goals and Vision 2030 plans for better emergency management. [6]

Xi et al. created a powerful accident detection system by combining advanced AI tools, Generative Adversarial Networks, Convolutional Neural Networks, and Vision Transformers. [7] They use synthetic accident data and smart classification to achieve a 95% accuracy rate in identifying crashes from video. This work shows the potential in blending computer-generated data with deep learning for fast, dependable accident detection in modern smart cities. [7]

Ragab et al. focus on hospital emergency rooms, using machine learning with Graph Convolutional Networks to help manage patient flow. [8] Their system classifies cases by severity and predicts how long patients will stay, reaching around 92% accuracy with real hospital data. This kind of technology is especially useful during busy periods, helping hospitals allocate resources better and make quicker decisions when it matters most. [8]

Ghahremannezhad et al. developed a system using YOLOv4 for real-time accident detection that tracks road users, maps their movements, and analyzes those patterns to spot possible accidents. [9] The system achieves over 93% detection accuracy, showing strong reliability for traffic monitoring. By combining object tracking with motion analysis, it detects and flags accidents efficiently. [9]

Wu et al. (2024) provides a massive benchmark dataset for AI-powered accident detection tailored to the demands of smart cities. [25] By testing advanced deep learning models like YOLO on this data,

they achieve improved detection accuracy and stronger, more dependable traffic monitoring systems. This work paves the way for robust accident detection tools in large-scale urban environments. [25]

Mane et al. (2025) developed a real-time system for detecting accidents using YOLOv8 for object detection alongside OpenCV for analyzing traffic videos. [11] Their model, trained on a large Crash Car Detection Dataset, stands out with impressive performance, 93.8% precision, 98% recall, and 96.1% mean average precision, outperforming previous YOLO and CNN models. Rigorous testing across different road environments shows that this system adapts well and is a solid option for advanced traffic safety monitoring. [11]

Zhang and Sung designed a hybrid detection approach that combines YOLOv5 with background subtraction, a CNN encoder for spatial details, and a transformer decoder to track motion over time. [12] By filtering out static elements from scenes, the system can focus on analyzing moving objects, and the transformer provides sophisticated tracking across video frames. Trained on data from vehicle crashes, it achieves 96% accuracy for spotting severe accidents, showing how noise reduction and combining spatial and temporal features lead to effective, real-time detection, even in complicated traffic conditions. [12]

Florence and Kirubasri created a deep learning solution that uses YOLOv5 on CCTV footage to spot accidents in real time. [13] They tested it on both the GIS-CADP dataset and a variety of YouTube videos, ensuring it could handle different vehicles and weather or lighting situations. Their platform aims to reduce mistakes and speed up emergency responses. They also propose adding 3D-CNNs for grading accident severity and a web alert system to notify hospitals, demonstrating that YOLOv5 delivers the best balance between speed and accuracy for fully automated traffic surveillance. [13]

Pawar and Attar reframe accident detection as spotting anomalies, something out of the ordinary, by using a mix of LSTM autoencoders. [14] These models learn what normal traffic flow looks like, then flag any sudden changes, like abrupt stops or crashes, as potential accidents. This method doesn't need tons of labeled data and adapts well to different traffic conditions. It performs strongly in real-world video tests with few false alarms, though it can struggle in poor visibility or with minor incidents. [14]

Xu et al. created the Traffic Accidents Dataset (TAD), a comprehensive resource with over 800 accidents and 24,000 labeled video frames covering a range of roads, lighting, and weather. [15] TAD gives researchers a robust way to test how well different models work in tough, real-life scenarios. While YOLOv5, ResNet, and 3D-CNNs do well in clear-cut cases, they still have trouble with rare, rapid collisions or when the view is blocked. The study suggests that bigger datasets and using semi-supervised learning could further strengthen accident detection in the real world. [15]

No.	Author/s	Year	Title	Dataset Sources	Technique/s	Results	Limitations	Gap Analysis
1	Karim et al.	2024	YOLOv8 + DeepSORT CCTV-based detection.	Accident frame annotations Image set (roboflow) (~1,000 / 500) of annotated accident frames. Dataset Link	YOLOv8 + DeepSORT	High detection/tracking metrics in controlled environments; lacks dataset size, cross-validation, latency data.	Image level training (not long CCTV streams); reasonably small / web-sourced set; no extensive adverse-condition testing; no end to end latency to camera to alert.	This is required to perform Needs CCTV stream evaluation, temporal/track aggregation, cross-dataset validation and edge/device latency tests.
2	Z. Chen et al.	2021	Traffic Accident Data Generation Based on Improved GANs.	Simulated trajectory /feature vectors (speed, accel, etc.) of highway / vehicle data; GAN-generated samples.	GAN with DenseNet	Synthetic data improves classifier training. limited scope and no cross-dataset validation.	Produces feature vectors (non-raw video); only vehicle/highway cases; does not have any temporal/sequence validation and cross-dataset tests.	Solid feature-level augmentation technique - however, it is not a replacement of the training data in forms of videos, there is still a need to bridge to visual/temporal data.
3	M. S. Arefin et al.	2025	Real-time rapid accident detection for Bangladesh	Huge annotated image corpus (approximately 9,000 of them reported: train/val/test split) generated through Roboflow; training by image.	YOLO variants on large image dataset	Stable and fast detection; limited live video and edge device testing.	Focus on still-images (not long CCTV streams); potential web / viewpoint bias; minimal adverse condition, end-to-end deployment testing.	Test on continuous CCTV streams, temporal aggregation/tracking, bad conditions lighting/occlusion and real end-to-end latency.

4	M. M. Ahmed et al.	2023	Deep learning traffic accident detection and alerting	The inclusion of custom CCTV + couple web/semi-synthetic frames; modules: YOLOv5+DeepSORT (detection/tracking), severity classifier, ResNet152 (fire). Dataset Link	YOLO, DeepSORT, classification modules	Robust detection/classification; partial synthetic data. limited long-sequence benchmarking	Frame level analysis (not a running stream); partial use of non pure CCTV data; limited edge / operational deployment tests.	Test on continuous CCTV streams, test on edge hardware, test in the real-world and test robustness.
5	S. Ayesha et al.	2025	CIRS multi-agent machine learning for real-time detection	Self-recorded video/CCTV data (paper reports of 10k frames: 5,200 accident, 4,800 non-accident; proprietary CCTV videos).	YOLOv11, VideoLLaMA3, multi-agent coordination	Sensitive detection and captioning; low sampling rate; domain bias; frame based eval	Mixed training data (public/proprietary), low training time, frame- of- training measures and no continuous stream metrics, Vlm risk of hallucination.	Valuable multi-agent, semantic augmentation approach, gaps are temporal robustness, domain generalization on cities, field level dispatch / latency test.
6	S. M. Elhag et al.	2023	Accident response enhancement using drones	No public dataset was used; data came from real-world cases, Najm's records, and	Drone integration with BPM	40% reduced emergency turnaround; supports Vision 2030 goals	The system is untested in real-world conditions, with uncertain	The system needs real-world validation and full AI automation to

				drone simulation reports.			travel time, battery, and weather performance, and manual reporting causing delays and inaccuracies.	improve accuracy and response efficiency.
7	Z. Xi et al.	2025	GAN + CNN + ViT for enhanced accident detection	Dataset built from YouTube video frames, split into accident and non-accident sets, with GAN-based synthetic data used for augmentation. Accident Detection from CCTV Footage on Kaggle	GAN, CNN, Vision Transformer	95% accuracy on video streams; robust synthetic and classification integration	Synthetic data may not capture real-world complexity, models need high-quality labels, and testing has been limited to pre-collected datasets, with potential difficulties in complex urban scenarios.	Key challenges include closing the gap between synthetic and real data performance and real-time validation across diverse urban conditions.
8	M. Ragab et al.	2023	GCN for emergency department patient flow management	Uses the Cleveland and Statlog healthcare datasets for emergency department monitoring and classification. Cleveland Heart Disease Dataset Statlog Heart Disease Dataset	Graph Convolutional Networks	92% accuracy; improved resource allocation and crowd control	Focused on healthcare data classification, relying on feature selection and tuning, but lacks validation on large-scale, real-time datasets and doesn't address real-time incident detection	Applying these techniques to real-time accident monitoring and multi-source data integration would enhance practical emergency response.
9	H. Ghahremannezhad et al.	2022	Real-time accident detection in	Collected 26 YouTube videos of day and night intersection accidents, manually annotated, and used MS COCO	YOLOv4, trajectory and conflict analysis	93.1% detection rate; effective in real traffic surveillance	Dependent on YOLOv4 and Kalman tracking, which struggle in crowds; heuristic	Stronger tracking, multi-modal analysis, and larger datasets are needed

			traffic surveillance	for YOLOv4 pretraining. GitHub dataset link with accident videos			analysis may miss subtle accidents; dataset is small.	to improve real-world reliability.
10	Y. Zhang and Y. Sung	2023	Accident detection using trajectory tracking and influence maps	Uses the Car Accident Detection and Prediction (CADP) dataset with 1,416 annotated CCTV video segments of traffic accidents, covering various times of day, weather conditions, and traffic densities. CADP dataset link	YOLOv5, CNN, trajectory analysis	95% accuracy in spatial-temporal accident detection	Deep SORT may mistrack under occlusion, the influence map needs refinement, and the CNN detects only binary accidents.	Future work should enhance tracking robustness, add severity prediction, and use transfer learning for better generalization.
11	D. T. Mane et al.	2023	Real-time vehicle accident recognition using YOLOv8 and OpenCV	Used Crash Vehicle Detection Dataset which includes 2,525 annotated images from surveillance and social media sources showing crashes, congestion, and normal traffic. Crash Car Dataset	YOLOv8, OpenCV	96.1% mAP; 93.8% precision; adaptable to various environments	Struggles in low light, false positives	Incorporate temporal or motion-based analysis to enhance detection reliability and reduce false positives.
12	Y. Zhang and Y. Sung	2023	YOLOv5 + background subtraction + transformer decoding	Uses Car Crash Dataset, a dashcam dataset with 1,500 accident and 3,000 normal videos, annotated for weather,	YOLOv5, CNN encoder, transformer decoder	96% accuracy; enhanced temporal modeling	Sensitive to lighting and background	Develop lightweight, robust models adaptable to real-world surveillance.

				time of day, ego-vehicle involvement, and bounding boxes for participants. Car Crash Dataset				
13	J. A. Ruby Florence and G. Kirubasri	2022	Accident detection system using deep learning	A custom dataset of 100 CCTV videos collected from YouTube and CADP, showing different vehicles and weather conditions.	YOLOv5	Real-time vehicle detection across conditions	Very small dataset, limited coverage	Expand dataset and add severity classification.
14	K. Pawar and V. Attar	2022	Deep learning-based accident detection as anomaly recognition	The study used IITH (127,138 normal frames, 863 accidents), Iowa DOT (100 train, 150 test freeway videos), and DoTA (4,677 dashcam accident videos with annotations) datasets. IITH Dataset Iowa DOT Dataset DoTA Dataset	Spatiotemporal autoencoder + LSTM	Can detect anomalies without labeled accident data; shows potential for adaptability	Trained only on normal data, lacks active learning	Needs training on diverse anomaly types and adaptive learning for new patterns.
15	Y. Xu et al.	2024	Traffic Accidents Dataset (TAD) benchmark	TAD dataset includes 344 CCTV videos from highways, city, and village roads with four accident types and detailed annotations.	YOLOv5, ResNet, 3D CNN	Large-scale dataset; challenges with rare and occluded accidents	Benchmark only, no new detection method	Use TAD for advanced multi-class, real-time detection models.

				<u>TAD Dataset</u>				
16	V. Adewopo et al.	2024	Big data and deep learning for AI-driven traffic accident detection	5.7K annotated, multi-perspective accident data (TrafficCam/DashCam) gathered from open-source repositories to cover 8 accident types. Dataset Link	Multi-source datasets (TrafficCam, DashCam)	5,700 samples; enhances detection with RGB + optical flow	Imbalanced class distribution, US geographic bias	Needs more coverage of rare accident types, rural incidents, and global contexts
17	S. Bakheet and A. Al-Hamadi	2022	Motion temporal templates and fuzzy time-slicing	Evaluation on motion temporal templates derived from public traffic video datasets, using sequences that capture various accident and normal traffic scenarios. DataSet Link	Motion history images, fuzzy-slicing, deep net	98.5% hit rate at 28 FPS; efficient motion feature extraction	Dark/cluttered scenes degrade performance	Expand to nighttime/low-light detection, improve occlusion handling
18	H. Liao et al.	2024	CRASH: Crash recognition and anticipation system	Tested on several real-world accident video datasets including challenging scenarios (occlusion, multi-agent); utilizes annotated video frames and context labels. Dataset Link 1 Dataset Link 2	Object Focus Attention, Context-Aware FFT module	Effective prediction and handling of accident scenarios	Sensor-dependent, needs further edge optimization	Real-time operation on edge, generalization across vehicle types
19	N. Behboudi et al.	2024	Review of advances in traffic accident	Meta-review of over 191 studies using data from government road accident records, on-	ML methods, DNNs	Highlights multi-source data use; addresses imbalance	The field is constrained by heterogeneous datasets, feature	Standardize open, multi-source data; create frameworks to fix data imbalance;

			analysis and prediction	board vehicle sensors, GPS, weather data, and social media feeds. Dataset Link		and integration challenges	differences, persistent data imbalance and under-reporting, and a lack of standardized accident severity labels	and implement real-time data in traffic systems.
20	R. Zhang et al.	2025	Multimodal LLMs for video-based traffic accident analysis	Performance was validated on extensive traffic video datasets (clips, prompts, and labels), employing segmentation masks for precise event localization. Dataset Link	Multimodal large language models	Enhances accident understanding using language and vision integration	Lacks high-frequency accident scenes; moderate accuracy	Contextual reasoning for rare events, scale to urban surveillance needed
21	J. Fang et al.	2023	Vision-based traffic accident detection and anticipation	Surveys 31+ datasets with links to major public accident sets like DoTA, DAD, CADP, DADA-2000, VIENA2, and others. DoTA CCD DADA-2000 SUTD-TrafficQA GTACrash	Survey of detection and anticipation methods	Comprehensive review of TAD and TAA techniques	Traffic accident detection is hindered by data imbalance, rare events, tough conditions like low light and occlusion, and limited datasets. Models often mistake anomalies for accidents and depend on unreliable detection and tracking,	The field lacks large, balanced datasets and robust models capable of handling occlusion, ambiguity, and data imbalance in real-world accident detection.

							reducing real-world accuracy.	
22	N. M. L. et al.	2025	Automated system for accident detection using OpenCV	No public datasets used; authors collected proprietary data from TrafCam and dashcams without releasing benchmarks.	OpenCV classical vision techniques	Practical implementation for real-time detection	Handcrafted methods are rigid, prone to false positives, and lack data diversity and reasoning for varied accidents.	Classical vision methods rely on heuristics and small proprietary data, highlighting the need for deep learning on large realistic datasets for better adaptability and accuracy.
23	S. Akter et al.	2025	Large language models for crash detection: methods and challenges	Reviews benchmark datasets like DAD, CADP, BDD100k, UCF-Crime, SHRP 2 NDS, and synthetic CrashEvent. DAD CADP BDD100k UCF-Crime SHRP 2 NDS	Survey of LLMs in crash detection	Analysis of datasets, fusion techniques, and current challenges	The field faces limited and inconsistent datasets, difficulty with detailed annotations, and scalability issues from long videos. Models also struggle with occlusion, domain shifts, and high complexity, while current metrics fail to capture crash semantics well.	Despite LLM advances, the lack of diverse annotated datasets and standard evaluation frameworks limits model generalization and adoption in crash detection.
24	A. Omari Alaoui et al.	2025	Emergency vehicle systems with deep learning	No accident-specific dataset used; the study cites general vision datasets like ImageNet, KITTI, Cityscapes, Pascal VOC, and	CNNs, ViT, GAN augmentation	Advances in vehicle recognition and routing; needs integration of multimodal reasoning	Models struggle in bad weather and occlusion, lack diverse annotated data, require heavy computation, and	Despite deep learning advances, large annotated emergency vehicle datasets are scarce, and multimodal

				COCO while focusing on technique reviews.			rarely use multimodal sensors, reducing reliability in crowded urban settings.	integration and real-time optimization remain limited in complex urban settings.
25	X. Wu et al.	2024	Comprehensive dataset for AI traffic accident detection	A public dataset of 5,700+ labeled traffic and accident videos from multiple sources, annotated with Labelbox for AI-based detection research. Dataset Link	Large-scale urban dataset, YOLO models	Improved detection accuracy and system robustness for smart cities	The dataset faces class imbalance, annotation inconsistency, and quality variation, requiring high computation and limiting real-world generalization.	Despite its size and variety, the dataset faces class imbalance, inconsistent annotations, and few rare accidents, highlighting the need for better curation and augmentation to boost robustness and fairness.

Table 2: Literature Review Summary.

Chapter 3: Software Project Management Plans (SPMP)

3.1 Project Overview

The aim of the project, the scope and the objectives of which are defined in this Software Project Management Plan, is to support the development of the Harnessing Deep Learning to Optimize Emergency Response project. Furthermore, this section contains an explanation of the project assumptions and constraints, the enumeration of project deliverables, as well as a brief description of project schedule and budget.

3.1.1 Purpose, Scope, and Objectives

- **Purpose**

To put it concisely, the project aims at the creation of an emergency response optimization system driven by AI that will make use of deep learning and computer vision for the purpose of real-time detection of car accidents through the analysis of the footage from surveillance cameras. The system is designed to cut down the time taken for emergency response by auto notifying the relevant authorities (police, ambulance, fire services) right away after accident detection thus making the roads safer and even saving lives in some cases.

- **Scope**

This project includes the preparation, creation, testing, and launching of a real time accident detection system based on YOLO and Convolutional Neural Network (CNN) models. The solution will be able to handle the feeds from the traffic cameras in real time, detect accidents, and send alerts to first responders automatically. Furthermore, the project will aim for the 24/7 live deployment of the system and cover the entire process from detection to alert generation.

- **Objective**

The primary goals of this project include:

- Achieving at least 85% accuracy (precision, recall, and F1-score) of the model on the test dataset of accident detection.
- Aiming for less than 2 seconds detection time, video streams will be processed in real-time with negligible latency.
- An automated alert generation functional prototype system will be made that is capable of detecting accidents and thus is able to generate alerts.
- Creating complete documentation that consists of SRS, SDS, STP, and User Manual according to IEEE standards.
- Aligning the project phases in the timeline of the 2025 2026 academic year.
Showing the system efficiency by going through testing and validation stages.

- **Project Relationship**

This project is to be done along with the course requirements at Imam Abdulrahman Bin Faisal University, College of Computer Science & Information Technology, during Term 1 2025/2026. It is the next step of research already carried out around computer vision and emergency response systems by applying top-class deep learning methods to the problems and needs of emergency management.

- **Requirements Reference**

The project requirements are officially documented in the Software Requirements Specification (SRS) document, which will be made according to the IEEE Standard 830-1998 and submitted on October 30, 2025.

3.1.2 Assumptions, Constraints and Risks

- **Assumptions**

The project is based on the following assumptions:

- Adequate public datasets containing traffic accident videos are available for training and validating the model
- Team members are skilled or can become skilled in Python, TensorFlow/PyTorch, and computer vision
- University computing resources and labs are accessible throughout the whole project period
- If the local hardware is not adequate, then GPU resources from the cloud (Google Colab Pro, AWS) will be available for use
- The project supervisor will give prompt critique and direction when the project reaches important points
- Internet access and collaborative tools (GitHub, Microsoft Teams) will be available all the time

- **Constraints**

The project is being carried out with the strength of the following imposed constraints:

- **Schedule Constraints:**
 - § The project has a time limit that corresponds to the academic year 2025/2026.
 - § The deadlines for submission of the reports are bi-weekly and mid-semester reports by October 16, 2025, and the final report by December 11, 2025, which are unchangeable.
 - § The final presentation is going to take place on December 17-18, 2025.
- **Budget Constraints:**
 - § Budget allocation is minimal or not at all (mainly open-source tools and existing resources will be used).
 - § A limited budget may be possible for cloud GPU services if required for intensive model training.
 - § No funding is allocated for commercial software licenses or proprietary datasets.
- **Resource Constraints:**
 - § The composition of the team is limited to five undergraduate students.
 - § Development will rely on personal computers and university lab facilities.
 - § Public datasets are the only source of data to be used (no access to proprietary traffic camera feeds).
 - § Access to high-performance computing resources is very limited.

Risks

The key project risks and the corresponding strategies for their mitigation:

Risk ID	Risk Description	Probability	Impact	Mitigation Strategy
R1	Insufficient number of accidents or their poor video quality for training	Moderate	Quality, Schedule	Combine data from several public sources; apply data augmentation methods
R2	Low model performance not reaching the target accuracy	Moderate	Quality	Testing with different architectures; deep hyperparameter tuning
R3	Not enough computing power to train the model efficiently	High	Budget, Schedule	Cloud GPU access setup with primary/backup solution
R4	Manual annotation takes longer than expected and affects the schedule	Moderate	Schedule	Implementing the use of efficient tools (CVAT); splitting the workload among team members

Table 3: Key projects risks

3.1.3 Project Deliverables

To fulfill the goals set for the ARTI-511 class and the project charter, the items listed below will be delivered:

Deliverable	Description	Delivery Date	Format
Bi-weekly Report #1	Initial planning and role assignment	September 18, 2025	MS Word document
Bi-weekly Report #2	Literature review findings	October 2, 2025	MS Word document
Mid-Semester Report	Project progress, SPMP development	October 16, 2025	MS Word document
Software Project Management Plan (SPMP)	Complete project management documentation	October 16, 2025	MS Word document (IEEE 1058-1998)
Bi-weekly Report #3	Software Requirements Specification	October 30, 2025	MS Word document

Software Requirements Specification (SRS)	Detailed functional and non-functional requirements	October 30, 2025	MS Word document (IEEE 830-1998)
Bi-weekly Report #4	Software Design Specification	November 13, 2025	MS Word document
Software Design Specification (SDS)	System architecture and design details	November 13, 2025	MS Word document (IEEE 1016-1998)
Bi-weekly Report #5	Design finalization progress	November 27, 2025	MS Word document
First Semester Final Report	Comprehensive project documentation	December 11, 2025	MS Word document
Final Presentation	Oral presentation of project outcomes	December 17-18, 2025	PowerPoint presentation
Trained AI Models	YOLOv8 and CNN model weights	Phase completion	Python/PyTorch format
Source Code Repository	Complete system implementation	Project completion	GitHub repository
User Manual	System operation and maintenance guide	Project completion	MS Word document (IEEE format)

Table 4: Project Deliverables

Delivery Location: The university's learning management system will be used to submit all documents and/or Dr. Mohammad Aftab Alam Khan, the project supervisor, will receive them directly. On GitHub, source code and models will be recorded/archived.

Packaging and Handling: Digital submission will be carried out as per the university's submission guidelines. Source code will be managed via version control using Git, accompanied with complete documentation and README files.

3.1.4 Schedule and Budget Summary

- **Schedule summary**

The project has a timeline from August 31, 2025, to February 20, 2026, which is divided into seven major phases:

Phase	Duration	Timeline	Accountable Actions

1. Project Initiation & Proposal	2.5 weeks	Aug 31 - Sep 18, 2025	Proposal approval, team planning, and role assignment
2. Literature Review	2 weeks	Sep 21 - Oct 2, 2025	Thorough examination of past research and present technologies
3. Project Plan & Mid-Semester Report	2 weeks	Oct 5 - Oct 16, 2025	SPMP creation, work plan development, mid-semester report
4. System Requirements (SRS)	2 weeks	Oct 19 - Oct 30, 2025	Gathering and analyzing requirements
5. System Design (SDS)	2 weeks	Nov 2 - Nov 13, 2025	Architecture designing, data flow specification
6. Final Design & Report Writing	4 weeks	Nov 16 - Dec 11, 2025	Design finalizing and writing of the first semester final report
7. Final Presentation	1 week	Dec 14 - Dec 18, 2025	Presentation preparation and delivery

Table 5: Schedule summary

The timeline of the first semester covers planning, requirements, and design, while the second semester (January–February 2026) is allocated for implementation, testing, and deployment.

- **Budget summary**

The financial aspect of the project is quite meager since it mostly supports itself with institutional resources and open-source software:

- o Primary Resources (No Cost):
 - ♣ Open-source development tools: Python, TensorFlow, PyTorch, OpenCV
 - ♣ University computer labs and personal PCs equipped with NVIDIA GPUs
 - ♣ Free collaboration tools: GitHub, Microsoft Teams
 - ♣ Publicly available datasets for training and testing
- o Possible Costs:
 - ♣ Approximate \$50-150 for cloud GPU services (Google Colab Pro, AWS EC2) when their local capacity is exceeded due to intensive training
 - ♣ The total budget is estimated at \$0-150 for the entire project duration

Note: The actual budget for computational resources will be based on actual needs during the implementation phase.

3.1.5 Evolution of the Plan

- **Compliance to Standards**

The layout of the Software Project Management Plan adheres to the IEEE standard 1058-1998 (IEEE Standard for Software Project Management Plans). In addition, it is intended that all the related documents such as SRS, SDS, STP, and User Manual will also adopt the relevant IEEE standards to ensure consistency and high-quality professional work.

- **Plan Updates**

- Scheduled Updates:
 - ♣ At the end of each major project phase, the SPMP will be subjected to a review and update procedure.
 - ♣ The mid-semester review (October 2025) and the start of the second semester (January 2026) will mark the times for announcing formal updates.
 - ♣ Each change will be tracked by version control along with a transparent revision history.
- Unscheduled Updates:
 - ♣ In case of significant changes in the scope of the project, probably lack of resources or any risk event that has a huge impact on the project plan, emergency updates will be developed.
 - ♣ Any unscheduled updates will have to be approved by the project supervisor and will need to be accompanied by a justification.
- Dissemination of Updates
 - ♣ Microsoft Teams will be the channel through which all plan updates will be sent to team members, and they will also be stored in the shared Google Drive.
 - ♣ Updates will be formally communicated to Dr. Mohammad Aftab Alam Khan, project supervisor.
 - ♣ The newest versions along with the respective version tags will be uploaded to the GitHub repository.
- Configuration Management
 - ♣ Until the team leads by the supervisor, the first edition of this SPMP, will be subjected to configuration management right after being approved by the supervisor.
 - ♣ Change management for the SPMP will be conducted through a formal change control process which will require the consensus of the team and the supervisor's approval, and this process will be applied to all changes thereafter.

- **Change Control**

After the initial issue of this plan, all changes will be controlled through the following process:

- Change request presented by team member along with the reason
- Impact evaluation performed by project leader
- Team consideration and debate in the weekly meeting
- Manager's approval for major alterations that affect the project's scope, schedule, or resources
- Incorporation of sanctioned changes into the revision history
- Circulation of the new version to all stakeholders.

3.1.6 References

The present Software Project Management Plan includes the following documents and sources of information:

- **IEEE Std 1058-1998**, IEEE Standard for Software Project Management Plans, Institute of Electrical and Electronics Engineers, 1998
- **IEEE Std 830-1998**, IEEE Recommended Practice for Software Requirements Specifications, Institute of Electrical and Electronics Engineers, 1998
- **IEEE Std 1016-1998**, IEEE Recommended Practice for Software Design Descriptions, Institute of Electrical and Electronics Engineers, 1998
- **ARTI 511 Course Syllabus**, Term 1 (2024/2025), Imam Abdulrahman Bin Faisal University, College of Computer Science & Information Technology
- **Ultralytics YOLOv8 Documentation**, Available at: <https://docs.ultralytics.com/>, Accessed: October 2025
- **TensorFlow Documentation**, TensorFlow Core v2.x, Available at: <https://www.tensorflow.org/>
- **PyTorch Documentation**, PyTorch v2.x, Available at: <https://pytorch.org/docs/>
- **OpenCV Documentation**, OpenCV 4.x, Available at: <https://docs.opencv.org/>

Note: Additional references for specific technical implementations, research papers, and methodologies will be documented in the Literature Review section and will also be included in subsequent deliverables.

3.1.7 Definitions and Acronyms

- **Definitions**
 - **Accident Detection**: via live monitoring video clips, the whole process of detections of accidents using machine vision and deep learning methods is done.
 - **Alert Generation**: the notifications are created and sent to the emergency response services instantly right after the accident is detected.
 - **Deep Learning**: the aspect of machine learning that utilizes sophisticated multi-layered neural networks to extract intricate patterns from very large datasets.
 - **Real-time Processing**: the ability to process video streams and detect accidents within a time frame of 2 seconds (the target is to achieve a latency of under 2 seconds).
 - **Model Training**: the training procedure of a neural network for identification of accident detection patterns by a dataset containing labeled training examples.
- **Acronyms**

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
ARTI	Artificial Intelligence (course code)
AWS	Amazon Web Services
CNN	Convolutional Neural Network
CCTV	Closed-Circuit Television
CVAT	Computer Vision Annotation Tool

GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
mAP	Mean Average Precision
OpenCV	Open Source Computer Vision Library
QA	Quality Assurance
SDS	Software Design Specification
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
STP	Software Test Plan
STS	Software Test Specification
UI/UX	User Interface/User Experience
YOLO	You Only Look Once (object detection architecture)

Table 6: Acronyms

3.1.8 Document Structure

The Software Project Management Plan that follows the recommendations of IEEE Std 1058-1998 is divided into six main sections:

- **Section 1: Project Overview** - This section gives an extensive description of the project attributes such as purpose, scope, objectives, assumptions, constraints, risks, deliverables, schedule and budget summary, plan evolution, references, and definitions.
- **Section 2: Project Organization** - The section introduces the organizational context, the interface with external stakeholders, the internal team structure and the detailed roles and responsibilities of the individual team members.
- **Section 3: Managerial Process Plans** - The plan discusses the startup plan (estimates, staffing, training), work plan (WBS, schedule, resources, budget), project tracking plan (requirements management, schedule control, quality control, reporting, metrics), risk management plan, and project closeout plan in detail.
- **Section 4: Technical Process Plans** - The section delineates the software development process model (Waterfall), methods, tools, and techniques to be used, infrastructure requirements, and product acceptance criteria.
- **Section 5: Supporting Process Plans** - The section describes documenting standards, formats, preparation responsibilities, and reviewing procedures for all project deliverables.
- **Section 6: Additional Plans** - The section goes over issues like data privacy and security, documentation and user operation, maintenance and updates, and deployment ethical and legal compliance; all which are included in supplementary considerations.

Every section consists of well-defined subsections which impart extensive assistance for all project management facets, thus allowing systematic progress and quality control to flow through the project lifecycle.

3.2 Project Organization

This section provides an overview of the organizational context of the project, its internal structure, and the key roles and responsibilities that are needed to deliver the Optimizing Emergency Response system. It outlines the interaction of the project with external parties, the structure of the team, and the kind of activity various positions are supposed to undertake.

3.2.1 External Interfaces

The project is carried out under the university's supervision and will be reviewed by the faculty evaluation committee. Key external interfaces include:

- Parent organization / sponsor: Imam Abdulrahman Bin Faisal University, in particular the College of Computer Science and Information Technology, offers academic supervision, institutional resources and formal assessment in the form of the project supervisor and the assigned committee.
- Primary users / customers: emergency-response agencies (police, ambulance, fire and rescue) and traffic/transport authorities who will be notified and provided with outputs of the operations.
- Data owners: CCTV network operators and traffic control centers of cities that can provide the video streams or the sample footage to develop and test it.

3.2.2 Internal Structure

The project follows a structured approach to roles and functional units to ensure consistent coordination and efficient workflow. The basic units are:

- **Project management:** planning, monitoring of progress, reporting and liaising with the supervisor and committee.
- **Development:** implementation of the video ingest pipeline, object detection model, tracking, and alert logic.
- **Design & user interface:** operator displays, dashboard design and usability checks.
- **Quality assurance & documentation:** checking, verification and making of reports and submission materials.

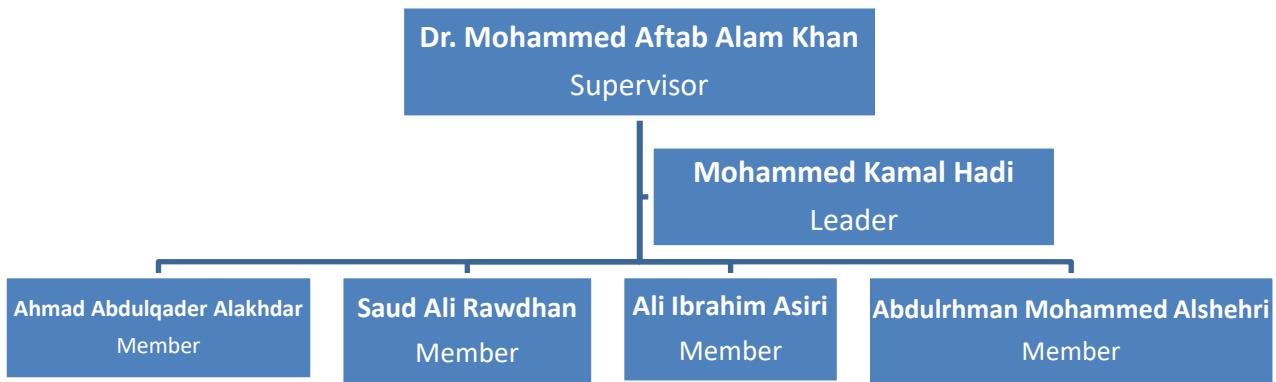


Figure 1: Internal structure

3.2.3 Roles and Responsibilities

The project manager shares responsibilities fairly among all the team members and he controls the performance of their assigned duties. In this model, every team member has a role to play in the entire life of development. Table 5 provides details on the assigned duties and responsibilities of each individual in the team.

Role	Responsibilities
Project Leader- Mohammed Hadi	General planning and coordination; schedule and milestone control; major liaison with supervisor; makes deliverables of the first quality.
Abdulrhman Mohammed Alshehri, Saud Ali Rawdhan, Developers	Build and integrate the detection and tracking modules, implement the streaming pipeline, and support testing and optimization.
Ahmad Abdulqader Alakhda , Designer / Tester	UI/UX design of hub displays; prepares test plans, executes test plans, documentation of test results and bug reports.
Ali Ibrahim Asiri, Documentation & Quality Lead	Develop and implement test plans (unit/frame level and sample stream tests), defects logging and fixes verification.
Supervisor: Dr. Mohammad Aftab Alam Khan	Technical oversight, decision approval on important design issues, milestone evaluations.

Table 7: Roles and Responsibilities

3.3 Managerial Process Plans

The section outlines the entire management plan of our project "Harnessing Deep Learning to Optimize Emergency Response". It elaborates on the intended procedures of project launching, implementation, performance tracking, risk avoidance and the formal closure.

3.3.1 Startup Plan

The startup plan will explain the components that will be the basis in the successful start up of the project. It involves preliminary estimates, project staffing, and the necessary competencies to run the project.

3.3.1.1 Estimates

The cost, resource and time requirements of the project have been carefully planned to make the plan realistic and achievable.

- **Cost Estimate:** The project is designed to be very cost-effective since it uses mostly open-source systems (Python, OpenCV, TensorFlow, and PyTorch) and already owned university hardware. The only expected expense is the possible use of cloud-based services of GPGUs (e.g., Google Colab Pro, AWS) to train models that are computationally intensive. Therefore, temporary budget will be allocated to these services in the event it is required.
- **Resource Estimate:** Resources of the project are further broken down into:
 - Human Resources: A team of five students.
 - Technical Resources: The high-performance personal computers with GPUs, the access to the university labs, and a huge collection of traffic-accident videos to train the model.
- **Estimated Time:** The project will be estimated to be completed during the 2025-2026 academic year. The Work Plan section has a detailed Gantt chart indicating a definite plan on each phase, thus, ensuring that all the deadlines are adhered to.

3.3.1.2 Staffing

The project team will include five motivated students of the College of Computer Science and Information Technology. The sharing of roles and responsibilities is based on personal expertise in order to streamline the working process, and provide high-quality results.

Project Phase	Allocated Resources	Human	Estimated Duration
Phase 1: Planning & Requirement Analysis	All Team Members	Human	4 Weeks
Phase 2: Data Collection & Preprocessing			4 Weeks
Phase 3: Model Design, Training & Validation			10 Weeks
Phase 4: System Implementation & Integration			8 Weeks
Phase 5: System Testing & Quality Assurance			3 Weeks

Phase 6: Final Documentation & Project Closeout		4 Weeks
--	--	---------

Table 8: Human Resources and Phase Duration

3.3.1.3 Project Staff Training

In order to meet the technical needs of the project, the team will implement and develop specific competencies associated with artificial intelligence and computer vision.

Role	Key Skills & Competencies
Project Leader	<ul style="list-style-type: none"> • Leadership, • Project Scheduling, • Risk Management, • Team Coordination, • Communication
AI/ML Developer	<ul style="list-style-type: none"> • Python, • TensorFlow/PyTorch, • OpenCV, • Computer Vision Models (YOLO, CNNs), • Data Annotation, • Model Training & Evaluation
Backend Developer	<ul style="list-style-type: none"> • API Development (Flask/Django), • System Integration, • Real-time Data Handling
Data Specialist	<ul style="list-style-type: none"> • Data Sourcing & Cleaning, • Video Preprocessing, • Data Annotation Tools (CVAT), • Data Augmentation
QA & Testing Engineer	<ul style="list-style-type: none"> • Test Plan Development & Execution, • Performance Analysis, • Bug Tracking, • System Validation

Table 9: Required Skills for Project Roles

3.3.2 Work Plan

The work plan provides a systematic way of implementing the project, listing the task breakdown, a graphical schedule and allocation of the necessary resources.

3.3.2.1 Work Breakdown Structure

The project is outlined into a distinct order of activities and sub-activities. The table below further breaks down the activities of the work, the resources that will be needed, timelines and the key deliverables of the work involved in each component of the project.

Task ID	Activity	Resources	Timeframe	Deliverable
1.0	Project Initiation & Planning	MS Office Suite	4 Weeks	SPMP & SRS Documents
1.1	Define Project Scope & Objectives	MS Word	1 Week	Approved Scope
1.2	Conduct In-depth Literature Review	Academic Databases	2 Weeks	Tech Stack Decision
1.3	Develop SPMP & SRS Documents	MS Word	1 Week	SPMP & SRS
2.0	System Design	Diagramming Tools, MS Word	2 Weeks	SDS Document
2.1	Design System Architecture	Draw.io	1 Week	Architecture Diagram
2.2	Design Alert Mechanism & Data Flow	MS Word	1 Week	Design Specifications
3.0	Data Management	Python, CVAT/LabelImg	4 Weeks	Prepared Dataset
3.1	Acquire Video Datasets	Web Browser	2 Weeks	Raw Video Data
3.2	Annotate & Preprocess Data	Python, CVAT	2 Weeks	Annotated Labels
4.0	Implementation Development	Python, AI Libraries	18 Weeks	Trained Model & System
4.1	Develop & Train Detection Model	Google Colab, Jupyter	10 Weeks	Trained Model Weights
4.2	Implement Video Processing Module	Python, OpenCV	4 Weeks	Video Capture Script

4.3	Develop Backend Alert System	Flask/Django	4 Weeks	Alerting API
5.0	Testing & Validation	Python, Test Plans	3 Weeks	STP/STS & Test Report
5.1	Evaluate Model Performance Metrics	Jupyter	1 Week	Performance Metrics Report
5.2	Conduct Integration & System Testing	Local Server	2 Weeks	Integration Test Log
6.0	Final Documentation & Delivery	MS Office, GitHub	4 Weeks	Final Report & Code
6.1	Write Final Project Report	MS Word	3 Weeks	Final Report
6.2	Prepare Final Presentation & Archive	PowerPoint, GitHub	1 Week	Presentation & Archive

Table 10: Work Breakdown Structure

3.3.2.2 Schedule Allocation

The table below outlines the project schedule, and every task is scheduled in accordance with the official due dates that are outlined in the course outline of first semester.

No.	Work Activity	Start Date	Duration	End Date
1	Project Initiation & Proposal	Aug 31, 2025	2.5 Weeks	Sep 18, 2025
1.1	Proposal Discussion & Approval	Aug 31, 2025	5 days	Sep 4, 2025
1.2	Initial Team Planning & Role Assignment	Sep 7, 2025	11 days	Sep 18, 2025
1.3	<i>Submit Bi-weekly Report #1</i>	-	-	<i>Sep 18, 2025</i>
2	Literature Review	Sep 21, 2025	2 Weeks	Oct 2, 2025
2.1	Conduct In-depth Literature Study	Sep 21, 2025	11 days	Oct 2, 2025
2.2	<i>Submit Bi-weekly Report #2</i>	-	-	<i>Oct 2, 2025</i>

3	Project Plan (SPMP) & Mid-Semester Report	Oct 5, 2025	2 Weeks	Oct 16, 2025
3.1	Develop Work Plan, Risks, & Resources	Oct 5, 2025	11 days	Oct 16, 2025
3.2	Consolidate Mid-Semester Report	Oct 5, 2025	11 days	Oct 16, 2025
3.3	<i>Submit Midterm Report</i>	-	-	<i>Oct 16, 2025</i>
4	System Requirements (SRS)	Oct 19, 2025	2 Weeks	Oct 30, 2025
4.1	Gather & Analyze Requirements	Oct 19, 2025	11 days	Oct 30, 2025
4.2	<i>Submit Bi-weekly Report #3 (SRS)</i>	-	-	<i>Oct 30, 2025</i>
5	System Design (SDS)	Nov 2, 2025	2 Weeks	Nov 13, 2025
5.1	Design System Architecture & Data Flow	Nov 2, 2025	11 days	Nov 13, 2025
5.2	<i>Submit Bi-weekly Report #4 (SDS)</i>	-	-	<i>Nov 13, 2025</i>
6	Final Design & Report Writing	Nov 16, 2025	4 Weeks	Dec 11, 2025
6.1	Finalize Project Design	Nov 16, 2025	11 days	Nov 27, 2025
6.2	<i>Submit Bi-weekly Report #5</i>	-	-	<i>Nov 27, 2025</i>
6.3	Write First Semester Final Report	Nov 16, 2025	25 days	Dec 11, 2025
6.4	<i>Submit Final Report</i>	-	-	<i>Dec 11, 2025</i>
7	Final Presentation	Dec 14, 2025	1 Week	Dec 18, 2025
7.1	Prepare Oral Presentation	Dec 11, 2025	5 days	Dec 16, 2025
7.2	<i>Deliver Final Presentation</i>	-	-	<i>Dec 17-18, 2025</i>

Table 11: Schedule Allocation (First Semester)

3.3.2.3 Resource Allocation

The project will also take advantage of a collection of modern hardware, software, and informational sources to enable the growth.

Category	Allocated Resources
Hardware	<ul style="list-style-type: none"> • High-performance PCs with NVIDIA GPUs, • University Computer Labs, • Cloud-based GPU Instances (as needed).
Software	<ul style="list-style-type: none"> • Development: Python, VS Code, Jupyter Notebook • AI/ML Libraries: TensorFlow, PyTorch, OpenCV, Scikit-learn • Collaboration: Git, GitHub, Microsoft Teams • Documentation: Microsoft Office Suite
Informational	<ul style="list-style-type: none"> • Research Papers (IEEE Xplore, arXiv), • Official Software Documentation, • Online AI/ML Community Forums.

Table 12: Technical and Informational Resources

3.3.2.4 Budget Allocation

Budgetary allocations are not yet decided but the project is expected to have low expenditure on the project but no expenses to be made on software licensing. One of the possible costs can be the cloud computing services in case an intensive model training is necessary.

3.3.3 Project Tracking Plan

This plan outlines steps to follow to monitor progress, changes, and also to make sure that the project deliverables are of the highest quality.

3.3.3.1 Requirements Management

In order to ensure that the project is a success, it is important that the project requirements are given priority and clearly defined. This kind of clarity makes it easy to monitor the status of the project. The requirements will be addressed as required to make any amendments or updates. In cases where change is justified, one needs to consider the impact that the project is likely to be affected by change. In case it is necessary and crucial, the revision of requirements will go through only with the unanimous agreement of the whole team and consultants, and the process of revision will require significant effort. On the other hand, when the change is not very critical and it does not affect the project flow, implementation will depend on the timing.

3.3.3.2 Schedule Control

The monitoring of the project advancement will be done proactively and in relation to the detailed schedule. The Gantt chart as the main visual tracking tool will be used to show task dependencies, durations and milestones. This allows the delays to be noticed at once and corrective measures to be planned.

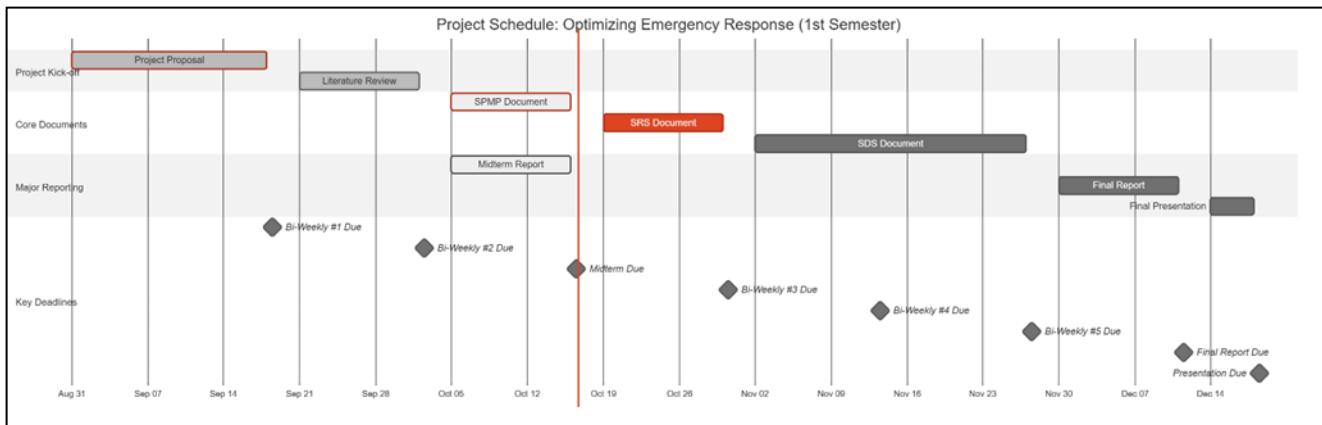


Figure 2: Gantt Chart: Project Schedule

3.3.3.3 Quality Control

The quality assurance will be ensured by the strict testing and following the standards defined. The main quality requirements are:

- **Model Performance:** The AI model should obtain a high accuracy, precision and recall on a special test set.
- **System Performance:** The video streams should be processed by the system with low latency to achieve real-time processing.
- **Code Quality:** The code will be version-controlled, documented, and peer-reviewed.

3.3.3.4 Reporting

The team and the supervisor will maintain constant communication and transparency on the status of the projects through a well-organized reporting system.

- **Team Meetings:** Meeting will take place on a weekly basis with internal members to report progress and address any arising issues and to discuss future functions. There will be recording of minutes, which will be distributed among the members.
- **Bi-Weekly Reports:** As required by the course syllabus, formal progress reports will be presented to the supervisor after every two weeks where information on the work done, status and further plans of the next cycle will be given.
- **Milestone Reports:** This will include major deliverables that will be prepared and presented formally to be reviewed by the supervisor and the evaluation committee, including the Mid-Semester and Final reports. Such reports will give a detailed review of the project progress and success at the critical points.

3.3.3.5 Project Metrics

Quantifiable metrics will be used to track the performance of the project.

Metric	Description	Frequency
Schedule Variance	Measures the deviation from the planned schedule to identify delays early.	Weekly

Model Accuracy	Quantifies the model's detection performance (mAP, F1-Score).	Per Training Cycle
System Stability	Tracks the number of critical bugs identified and resolved during testing.	During Testing Phases

Table 13: Key Project Metrics

3.3.4 Risk Management Plan

There are possible risks in the project that are identified in this plan and proactive measures to capture the effect of these risks.

Risk ID	Category	Description	Probability	Impact	Mitigation Strategy	Monitoring
R1	Technical	Insufficient accident video quality for training	Moderate	Quality	Data augmentation, multi-source datasets	Weekly model accuracy tests
R2	Technical	Model accuracy <85%	High	Reliability	Ensemble YOLOv8+CNN, hyperparameter tuning	Bi-weekly validation runs
R3	Infrastructure	GPU shortages for training	High	Schedule	Cloud fallback (Google Colab/AWS)	Resource usage dashboard
R5	Operational	False positives/negatives in detection	High	Safety	Confidence thresholds (>0.8), human override	False positive rate tracking
R6	Operational	CCTV stream failures (>2s latency)	High	Response	Auto-reconnect, redundant feeds	Stream uptime monitoring
R7	Security	Video data privacy breaches	Medium	Legal	AES-256 encryption, 24h retention	Security audit logs
R8	Scalability	100+ camera overload	Medium	Performance	Kubernetes scaling, GPU load balancing	System load metrics

Table 14: Comprehensive Risk Management Plan

3.3.5 Project Closeout Plan

The project closeout phase will be used to ensure that the project is professional and in an orderly manner ended.

- **Final Verification:** Making sure that all systems function and requirements have been covered.
- **Project Archiving:** The project materials such as source code, trained models and documentation will be arranged and archived on GitHub.
- **Final Reporting:** It will produce a final report and user manual that has undergone a review and is submitted.
- **Knowledge Transfer:** The group will provide a final presentation and hold a session of lessons learned to document lessons learned that will be used in future projects.

3.4 Technical Process Plans

This section includes process models, methods, tools, and techniques. Moreover, it includes infrastructure and product acceptance.

3.4.1 Process Model

The Waterfall model was selected as the primary software development approach for this project, Optimizing Emergency Response, because of its structured, sequential, and documentation-oriented nature. Each phase produces a set of deliverables that serve as verified inputs to the next phase. Given that this project follows academic supervision, institutional milestones, and fixed submission deadlines, the Waterfall model ensures systematic progress and traceable quality control across all development stages.

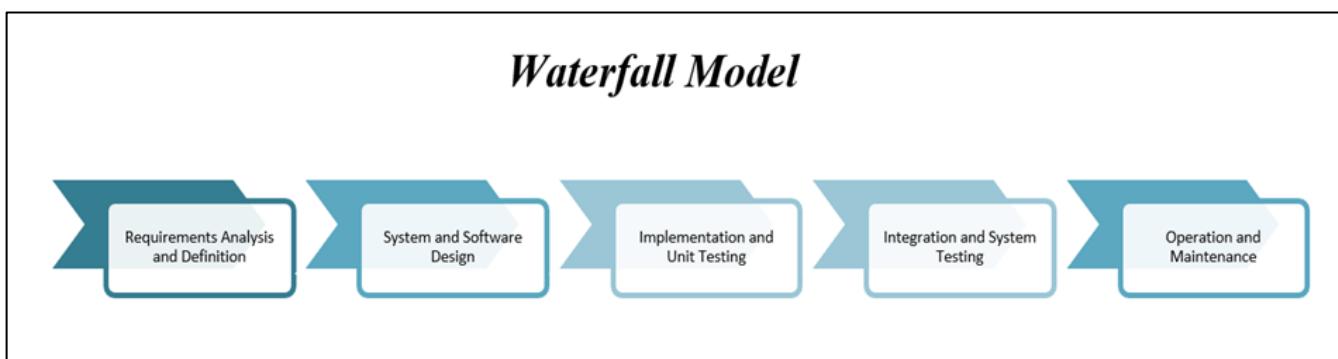


Figure 3: The Waterfall Model

Table 13 shows the waterfall phases, along with their Phase, Duration, Description, Main Activities, Deliverables and Outcomes.

Phase	Duration	Description	Main Activities	Deliverables / Outcomes
	n			

1. Requirements Analysis	Nov 1 – Nov 20 2025	Identify, analyze, and document all functional and non-functional requirements for the system.	<ul style="list-style-type: none"> Gather requirements from literature and emergency response protocols. Define system functions (accident detection, alert generation, and reporting). Specify hardware & software constraints, datasets, and ethical considerations. 	<ul style="list-style-type: none"> Software Requirements Specification (SRS). Use-case and context diagrams. Dataset description document.
2. System and Software Design	Nov 21 – Dec 10 2025	Translate the requirements into detailed architecture and technical design.	<ul style="list-style-type: none"> Design overall system architecture (camera feed → YOLOv8 model → alert module). Define data flow and database structure. Select frameworks and AI tools (TensorFlow 2.x, PyTorch, OpenCV). 	<ul style="list-style-type: none"> Software Design Specification (SDS). Architecture and dataflow diagrams. Database schema and interface design.
3. Implementation (Coding)	Dec 11 – Jan 10 2026	Convert the design into executable modules and integrate the core AI model.	<ul style="list-style-type: none"> Develop and train YOLOv8 and CNN models on labeled datasets. Implement real-time accident detection and alert system in Python. Develop API and dashboard interface for emergency notifications. Use GitHub for version control and continuous testing. 	<ul style="list-style-type: none"> Trained AI models. Source code repository. Integrated prototype with detection and alert modules.
4. Testing and Verification	Jan 11 – Jan 25 2026	Validate that the system meets specified requirements and performance targets.	<ul style="list-style-type: none"> Conduct unit testing for each module. Perform integration and system testing. Evaluate accuracy (Precision, Recall, F1-Score). Simulate accident scenarios for response time testing. 	<ul style="list-style-type: none"> Software Testing Plan (STP). Test reports and performance metrics. Validated system ready for deployment.
5. Deployment and Integration	Jan 26 – Feb 5 2026	Deploy the validated system for demonstration and evaluation.	<ul style="list-style-type: none"> Deploy system on cloud platform (Google Colab or AWS EC2). Integrate real-time video feed for testing. Prepare user manual and presentation materials. 	<ul style="list-style-type: none"> Operational prototype demonstration. Deployment report. User manual and demo video.
6. Maintenance	Feb 6 – Feb 20 2026	Conduct system review, bug fixes,	<ul style="list-style-type: none"> Resolve issues from testing and presentation reviews. Optimize model parameters and update dataset. Document 	<ul style="list-style-type: none"> Updated system version. Maintenance

and Enhancement		and enhancements based on feedback.	lessons learned and future improvements.	log. • Final project closure report.
------------------------	--	-------------------------------------	--	--------------------------------------

Table 15: The Waterfall Phases

3.4.2 Methods, Tools, and Techniques

This section describes the software development methods, tools, and techniques utilized throughout the project Optimizing Emergency Response. The project integrates Artificial Intelligence, computer vision, and web technologies to detect car accidents in real time using live surveillance footage. The adopted methods focus on ensuring high detection accuracy, efficient model performance, and seamless communication with emergency services. The tools and frameworks were selected based on their reliability, open-source support, and suitability for AI-based computer vision applications.

Category	Tool / Framework	Purpose / Description
Development Environment	Python 3.10	Primary programming language for system development, AI modeling, and automation.
AI & Deep Learning Frameworks	YOLOv8 (Ultralytics)	Used for real-time object and accident detection in traffic footage. Offers state-of-the-art accuracy and fast inference speed.
	TensorFlow 2.x / PyTorch	For training and fine-tuning Convolutional Neural Network (CNN) models and integrating YOLOv8 models.
Computer Vision Library	OpenCV	Used for video stream handling, frame extraction, and pre-processing of camera footage.
Data Labeling and Preparation	LabelImg / Roboflow	Tools for annotating accident datasets and managing dataset versions for model training.
Database Management	MySQL / SQLite	Used to store incident records, alert logs, and user information.
Backend Framework	Flask (Python)	Lightweight web framework used to handle the system's API for alert generation and response communication.
Frontend Development	HTML, CSS, JavaScript (React optional)	Used to design the monitoring dashboard for displaying live detection and alerts.
Version Control	Git & GitHub	For collaborative source code management, version control, and tracking development progress.
Testing Tools	PyTest / Unittest	Used for unit and integration testing of modules to ensure system stability and correctness.
Deployment Platform	Google Colab / AWS EC2	Used for model training, testing, and hosting the prototype in a cloud environment.
Project Management & Documentation	Microsoft Word / Excel / Trello	Used for documentation, scheduling tasks, and progress tracking throughout the development cycle.

Table 16: Used Tools

3.4.3 Infrastructure

This section outlines the hardware and software infrastructure used in each development phase of the Optimizing Emergency Response project. The infrastructure was selected to support deep learning model training, real-time video processing, and deployment of the system prototype. Each phase of the Waterfall model relies on specific tools and environments to ensure efficiency, accuracy, and scalability.

Project Phase	Tools / Infrastructure Used
Requirements Analysis	Microsoft Word, Excel, Trello, Google Drive
System and Software Design	Draw.io, Lucidchart, Visual Paradigm, MySQL Workbench
Implementation (Coding)	Python 3.10, TensorFlow 2.x, PyTorch, YOLOv8, OpenCV, Flask
Testing and Verification	PyTest, Unittest, Jupyter Notebook, Confusion Matrix Metrics
Deployment and Integration	Google Colab, AWS EC2, GitHub, Flask Server
Maintenance and Enhancement	GitHub Issues Tracker, Google Colab, MySQL, Documentation Tools

Table 17: The Essential Requirements

3.4.4 Product Acceptance

This section defines the criteria and procedures that will be followed to ensure the *Optimizing Emergency Response* system meets its intended objectives and quality standards before final submission and deployment. Product acceptance is based on verifying that all functional and non-functional requirements are satisfied, that the system performs reliably in real-time environments, and that it operates according to its design specifications.

3.5 Supporting Process Plans

This section outlines the documentation standards, responsibilities, and review procedures followed throughout the *Optimizing Emergency Response* project. All project documents are prepared in accordance with the relevant IEEE standards and university-provided templates to ensure consistency, accuracy, and traceability across all development phases. Each document is collaboratively prepared by the project team and reviewed under the supervision of **Dr. Aftab**, ensuring that all deliverables meet academic and professional quality standards before submission.

Document Type	Format Standard	Prepare Document	Review Document
Introduction and Literature Review	Provided by the supervisor	All team members	Project Supervisor: Dr. Aftab
Mid-Semester Report	Provided by the supervisor	All team members	Project Supervisor: Dr. Aftab
Software Project Management Plan (SPMP)	IEEE Standard 1058-1998	All team members	Project Supervisor: Dr. Aftab
Software Requirements Specification (SRS)	IEEE Standard 1058-1998	All team members	Project Supervisor: Dr. Aftab
Methodology	Provided by the supervisor	All team members	Project Supervisor: Dr. Aftab

Software Design Specification (SDS)	IEEE Standard 1058-1998	All team members	Project Supervisor: Dr. Aftab
First Semester Final Report	Provided by the supervisor	All team members	Project Supervisor: Dr. Aftab
Software Test Plan	IEEE Standard 1058-1998	All team members	Project Supervisor: Dr. Aftab
User Manual	IEEE Standard 1058-1998	All team members	Project Supervisor: Dr. Aftab
Senior Project Source Code and Report	Provided by the supervisor	All team members	Project Supervisor: Dr. Aftab

Table 18: Supporting Process Plans

3.6 Additional Plans

This section describes the supporting considerations for data handling, documentation, system operation, and compliance with ethical and legal standards. These plans help ensure the project is properly organized, secure, and adaptable to different use contexts.

3.6.1 Data Privacy and Security

All datasets used in this project are publicly available and pre-annotated. No personal information is collected or stored. Data processing is performed locally to maintain security and control. Appropriate care will be taken to ensure data is used responsibly and handled according to standard academic and institutional guidelines.

3.6.2 Documentation and User Operation

A structured set of instructions will accompany the system, detailing the environment setup, model execution steps, and how to view and interpret results. This documentation ensures that the system can be operated consistently and understood clearly by authorized users.

3.6.3 Maintenance and Updates

The project resources, including the source code and model files, are organized to support proper tracking and structured management. This organization facilitates adjustments or technical refinements when necessary, without requiring major system changes.

3.6.4 Deployment Considerations

The system is developed to operate on standard computing environments. Its structure allows it to be integrated with external components or operational setups as needed. The design emphasizes modularity, making it easier to adapt the system to different configurations or extend its functionality when required.

3.6.5 Ethical and Legal Considerations

The system complies with ethical standards by avoiding the use of any personal data and ensuring responsible handling of input sources. In all cases, data usage and system operation are expected to align with relevant legal and regulatory frameworks concerning surveillance and information security.

Chapter 4: Software Requirements Specification (SRS)

4.1 Introduction

This Software Requirements Specification (SRS) document describes the requirements for the project Harnessing Deep Learning to Optimize Emergency Response. It defines the overall purpose, features, and behavior of the system, along with its expected performance and interaction with users. The document serves as a guide for the project team, supervisor, and evaluators, ensuring that all members share a clear understanding of what the system will do and how it will operate.

4.1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to clearly define and document the functional and non-functional requirements of the proposed system. It provides a foundation for design, development, and testing, ensuring that the system aligns with its intended goals of accuracy, reliability, and usability. This document also facilitates communication by establishing a shared understanding of the system's objectives, scope, and constraints.

4.1.2 Scope

This project has as its goal the design, testing, and implementation of a real-time emergency response optimization system which utilizes deep learning for the detection of accidents. It takes in real-time video streams from traffic cameras and uses YOLO and CNN algorithms to recognize car crashes. Then, it initiates the alert procedure for the emergency responders (police, ambulance, fire services) automatically. By covering the whole process from detection to alert creation, the system supports live deployment all day, every day aiming at cutting down emergency response time and enhancing road safety.

4.1.3 Definitions, Acronyms, and Abbreviations

- **Definitions**

- **Accident Detection:** the entire procedure of letting machines see and learn to detect accidents through the application of state-of-the-art technology in vision processing and deep learning is carried out using video clips in real time monitoring.
- **Alert Generation:** immediately after the accident is detected, the alerts are generated and sent to the emergency response services.
- **Deep Learning:** the area of machine learning that relies on extremely complex and layered neural networks to identify very subtle patterns among the colossal amount of data.
- **Real-time Processing:** it is a video processing performance that recognizes accidents at a maximum duration of 2 seconds (the target is to make the delay shorter than 2 seconds).
- **Model Training:** the process of teaching a neural network to recognize accident detection patterns through a dataset that has labeled training examples.

- **Acronyms**

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
ARTI	Artificial Intelligence (course code)
AWS	Amazon Web Services
CNN	Convolutional Neural Network
CCTV	Closed-Circuit Television
CVAT	Computer Vision Annotation Tool
GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
mAP	Mean Average Precision
OpenCV	Open Source Computer Vision Library
QA	Quality Assurance
SDS	Software Design Specification
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
STP	Software Test Plan
STS	Software Test Specification
UI/UX	User Interface/User Experience
YOLO	You Only Look Once (object detection architecture)

Table 19: Glossary of Acronyms and Abbreviations

4.1.4 References

- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans, Institute of Electrical and Electronics Engineers, 1998
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, Institute of Electrical and Electronics Engineers, 1998

- IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions, Institute of Electrical and Electronics Engineers, 1998
- ARTI 511 Course Syllabus, Term 1 2025-2026, Imam Abdulrahman Bin Faisal University, College of Computer Science & IT
- Ultralytics YOLOv8 Documentation, Available at <https://docs.ultralytics.com>, accessed October 2025
- TensorFlow Documentation, TensorFlow Core v2.20, Available at <https://www.tensorflow.org>
- PyTorch Documentation, PyTorch v2.90, Available at <https://pytorch.org/docs>
- OpenCV Documentation, OpenCV 4.12.0, Available at <https://docs.opencv.org>

4.2 Overall description

The product is a smart software system, which uses artificial intelligence and computer vision to track live video feeds and identify road incidents automatically. It runs live camera feeds through trained models to detect and categorize accidents, retains temporary footage to verify them, and directly transmits the relevant emergency response agencies including police, ambulance, and fire departments. It is an autonomous system as well as it does not require any direct interaction with the user and is instead used as a background monitoring system to assist emergency teams by providing proper, reliable and timely alerts. It may be used as a standalone solution, or work with the current city or agency infrastructure, guaranteeing that it will operate continuously under the different environmental and network parameters. The main aspects will be privacy, data processing, and reliability of the work, ensuring that the entire visual information is processed safely and the system does not drop even when the quality of input or connection is impaired.

4.2.1 Product perspective

The software is mainly a stand-alone program which may also have an integration as a part of a larger emergency management or smart city program. It independently observes live video streams, identifies an incident on the road, and is automatically able to dispatch structured messages to the emergency authorities, including police, ambulance, and fire services. The software can be integrated with the available dispatch or analytics platforms to share incident data and metadata of the camera to facilitate coordinated and prompt reactions. When used independently, it does the same monitoring and alerting functions on its own and has its own local database and communication interfaces.

Major Inputs and Outputs:

Inputs: Continuous live video streams from surveillance or traffic cameras, along with optional metadata such as camera identifiers and geographic coordinates.

Outputs: Automatically generated messages with event information, time and place of events, severity measurements, and graphical evidence, which are safely sent to assigned emergency response units or systems.

4.2.2 Product functions

The proposed system, Harnessing Deep Learning to Optimize Emergency Response, is designed to automatically detect and report car accidents using real-time video analysis powered by deep learning

and computer vision. Its main functions work together to minimize human error and reduce emergency response time.

1. Real-Time Video Monitoring

- The system continuously receives live video streams from surveillance and traffic cameras placed across urban roads and intersections.
- Video feeds are processed in real time to ensure that the system can immediately analyze ongoing traffic events.
- A video buffering mechanism maintains short-term footage storage for immediate review or evidence collection.

2. Accident Detection Using Deep Learning

- The system uses AI models such as YOLO and Convolutional Neural Networks (CNNs) to detect accidents within video frames.
- Upon detection, the system validates the event to minimize false alarms, ensuring high accuracy and reliability.

3. Accident Classification and Severity Assessment

- After identifying an incident, the system classifies it by severity level (e.g., minor, moderate, or major crash) based on visual cues such as collision intensity, number of vehicles, and fire or debris presence.
- Severity classification helps emergency control centers allocate the right resources (ambulance, police, or fire services).

4. Automated Alert and Notification System

Once an accident is confirmed, the system automatically generates an alert that includes:

- Accident location (via GPS or mapped camera coordinates)
- Time and estimated severity
- Reference camera ID and a visual of the incident

The alert is then sent to relevant emergency units through mobile notification, or integration with emergency service platforms.

5. System Management and Monitoring Dashboard

Authorized users, such as emergency control staff, can access a centralized Emergency Response Dashboard.

The dashboard enables users to:

- View live camera feeds and add or remove connected Surveillance Nodes.
- Monitor real-time accident alerts and update linked Emergency Units (hospitals, fire departments, police stations).
- Review past incidents and generate analytics reports for performance and response tracking.

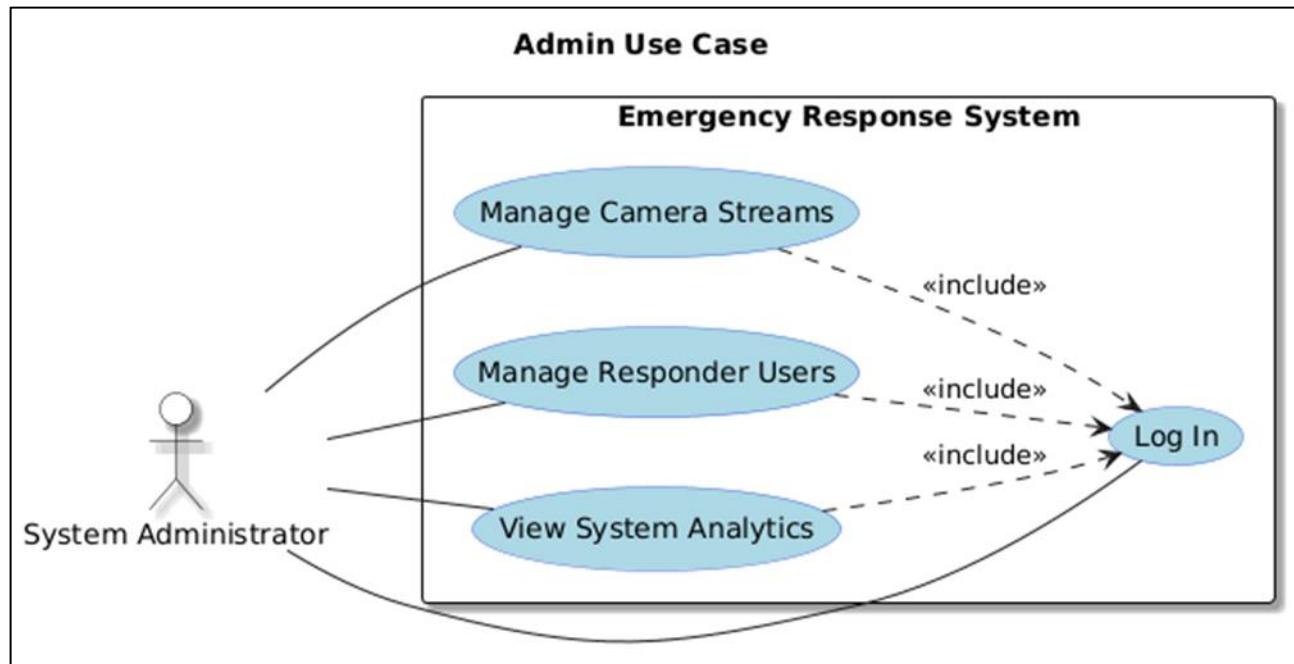


Figure 4: Admin Use Case

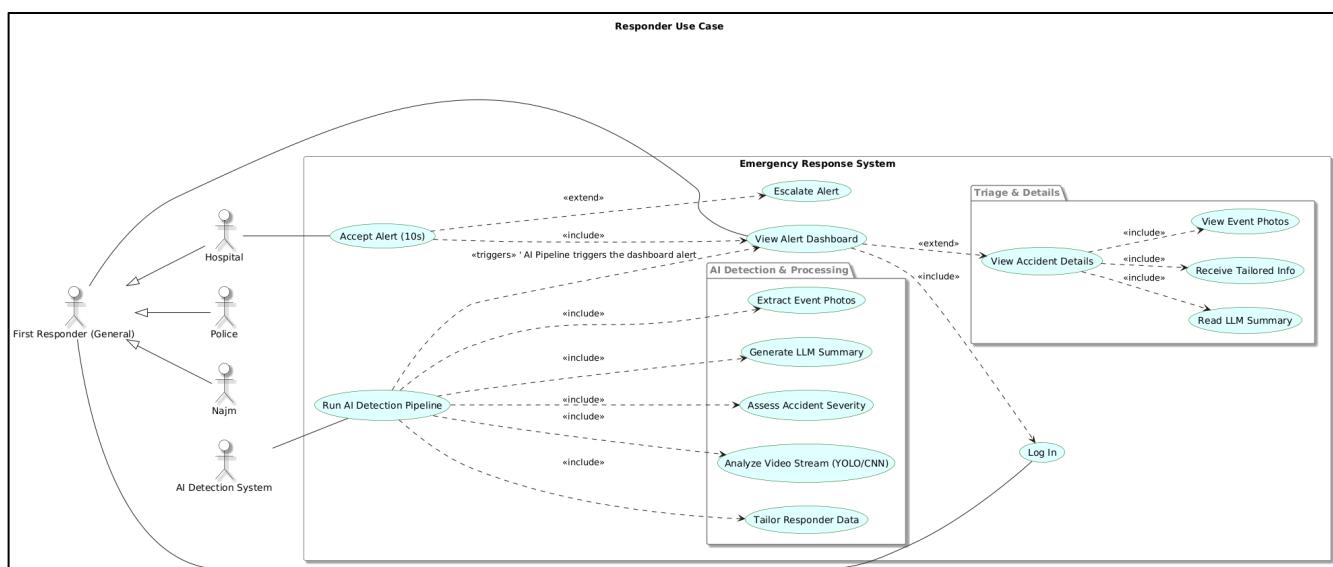


Figure 5: First Responders Use Case

4.3 Specific requirements

This section outlines the detailed functional and non-functional requirements of the Emergency Response Optimization System. It defines how the system will operate, including data input from surveillance cameras, real-time accident detection using deep learning models, alert generation, and communication with emergency units. The requirements specify user interactions, system performance, reliability, and security standards to ensure accurate detection, fast response, and seamless integration with existing smart city infrastructure.

4.3.1 External interface requirements

This section details the interfaces connecting the system to its users and other external components.

4.3.1.1 User interfaces

The system will provide a web-based graphical user interface (GUI) built using the React framework to ensure a responsive, modern, and high-performance user experience. The interface will be simple and intuitive, tailored to two primary user roles: System Administrator and First Responder User.

4.3.1.1.1 Common Interfaces

4.3.1.1.1.1 Login Interface

This is the secure entry point for all users.

Field	Format	Level	Input/Out put	Comment
Username / Email	Text (Email format)	Required	Input	User's registered email address.
Password	Text (Masked)	Required	Input	User's secure password.
Login Button	Button	Required	Input	Submits credentials for authentication.
Forgot Password	Link	Optional	Input	Initiates password reset flow.
Error Message	Text (Red)	N/A	Output	Displays if authentication fails (e.g., "Invalid username or password").

Table 20: Login Interface

4.3.1.1.2 System Administrator Interfaces

Accessible after a user with "Admin" role logs in.

4.3.1.1.2.1 Admin Dashboard (Homepage)

Main navigation hub for the administrator.

Field	Format	Level	Input/Output	Comment
Header	Text	Required	Output	Displays "Administrator Dashboard" and logged-in user's name.
Navigation Menu	Links	Required	Input	Links to "Camera Management" and "User Management".
System Status	Text / Graphics	Required	Output	A summary view showing system health (e.g., "All systems operational").
Active Streams	Number	Required	Output	A count of currently active and processing video streams.
Logout Button	Button	Required	Input	Logs the administrator out of the system.

Table 21: Admin Dashboard Interface

4.3.1.1.2.2 Camera Management Interface

Allows the admin to add, view, and manage the city's camera streams ("Surveillance Nodes").

Field	Format	Level	Input/Output	Comment
Camera List	Table	Required	Output	Displays all registered cameras with their ID, Name, Stream URL, and Status (e.g., "Active", "Offline").
Add Camera Button	Button	Required	Input	Opens the "Add Camera" modal/form.
View Stream	Button	Optional	Input	Allows admin to view the live feed from the camera.
Edit/Delete	Buttons	Optional	Input	Allows editing or removing an existing camera from the list.
Camera ID (Form)	Text	Required	Input	A unique identifier for the camera (e.g., "CAM-001-KSA").
Camera Name (Form)	Text	Required	Input	A human-readable name (e.g., "King Fahd Rd @ Olaya St").

Stream URL (Form)	Text (URL)	Required	Input	The RTSP or HLS link for the live video feed.
Save Camera Button	Button	Required	Input	Saves the new camera to the system.

Table 22: Camera Management Interface

4.3.1.1.2.3 User Management Interface

Allows the admin to create, manage, and update "Emergency Units" (responder accounts).

Field	Format	Level	Input/Output	Comment
Agency List	Table	Required	Output	Displays all registered agencies (Hospital, Police, Najm) with their name, user email, and type.
Add Agency Button	Button	Required	Input	Opens the "Add Agency User" modal/form.
Agency Name (Form)	Text	Required	Input	e.g., "King Faisal Hospital ER" or "Najm - Riyadh".
User Email (Form)	Text (Email)	Required	Input	The login email for the new user account.
Agency Type (Form)	Dropdown	Required	Input	Options: "Hospital", "Police", "Civil Defense", "Najm".
Contact Number (Form)	Text (Phone)	Required	Input	Emergency contact number for alerts.
Save User Button	Button	Required	Input	Creates the new responder user account.
Edit/Delete	Buttons	Optional	Input	Allows editing or removing an existing agency account.

Table 23: User Management Interface

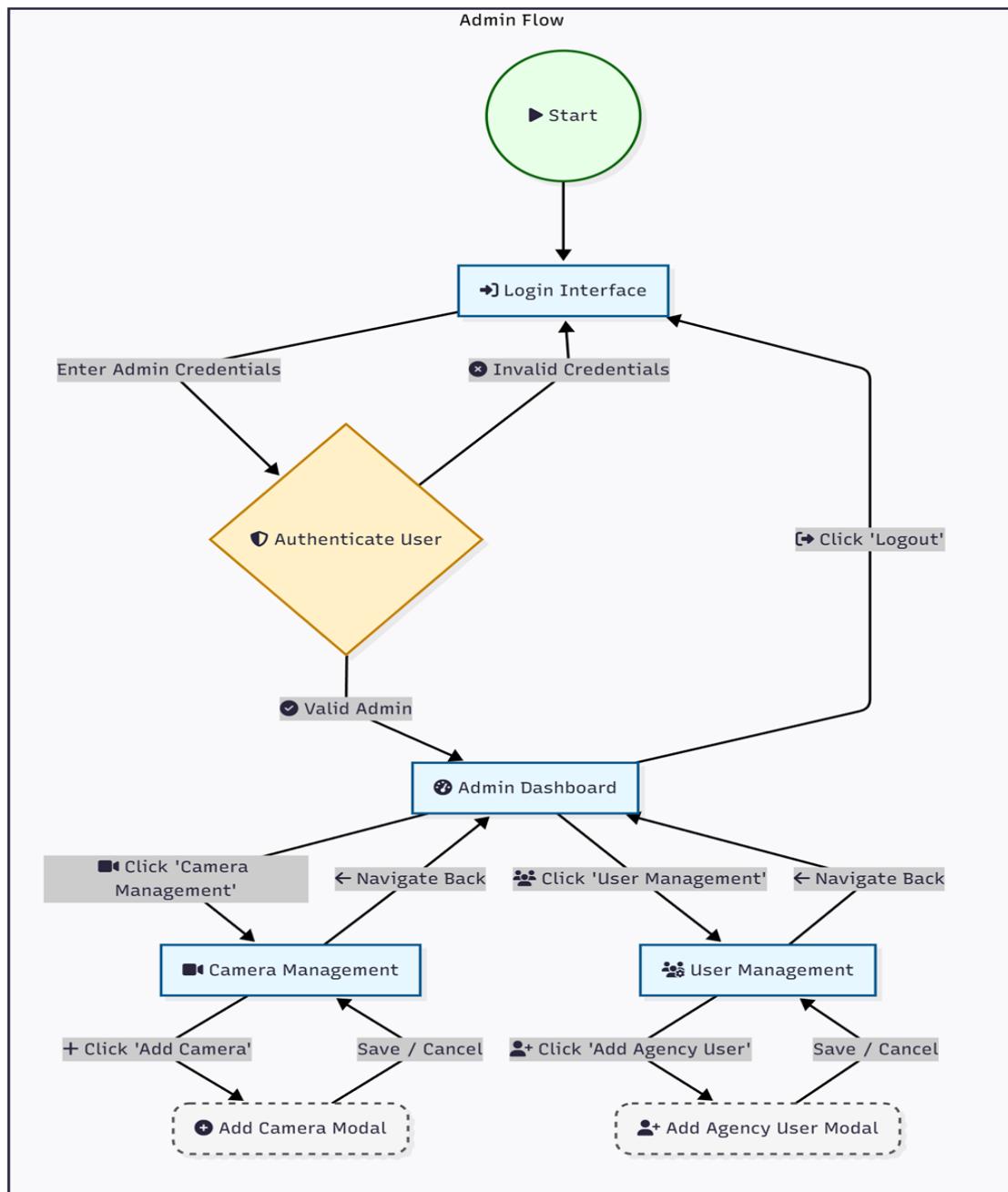


Figure 6: Admin workflow system

4.3.1.1.3 First Responder User Interfaces

Accessible after a user with a "Responder" role (e.g., Hospital, Police, Najm) logs in.

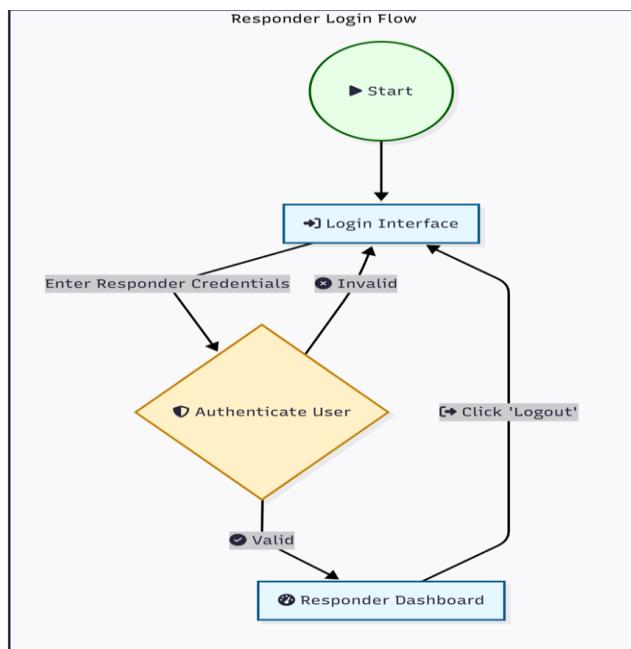


Figure 7: The Responders Login Flow

4.3.1.1.3.1 Responder Dashboard (Homepage)

Main dashboard for viewing and managing active accident alerts.

Field	Format	Level	Input/Output	Comment
Header	Text	Required	Output	Displays agency name (e.g., "King Faisal Hospital ER") and logged-in user.
Alerts List	List / Table	Required	Output	A list of active alerts, sorted by time (newest first). Each item shows location, time, and status (e.g., "New", "Accepted").
Alerts Map	Interactive Map	Required	Output	A map (e.g., OpenStreetMap) showing pins for all active accident locations.
Logout Button	Button	Required	Input	Logs the responder out.

Table 24: Responder Dashboard Interface

4.3.1.1.3.2 Accident Alert Interface (Modal)

This is the immediate notification that appears on the Responder Dashboard.

Field	Format	Level	Input/Output	Comment

Alert Title	Text	Required	Output	"!! NEW ACCIDENT DETECTED !!"
Location	Text	Required	Output	Address or GPS coordinates of the accident.
Time	Timestamp	Required	Output	Time of detection.
Accept Button	Button	Required (for Hospitals)	Input	Responder clicks to accept the case. after 10 seconds if not clicked/responded, it will send to the second nearest hospital.
View Details	Button	Required	Input	Navigates to the full Accident Details Interface.
Timer	Countdown	Required (for Hospitals)	Output	A visual 10-second countdown timer.

Table 25: Accident Alert Interface

4.3.1.1.3.3 Accident Details Interface

This page provides the full, LLM-generated summary of the accident event.

Field	Format	Level	Input/Output	Comment
Case ID	Text	Required	Output	Unique identifier for the accident.
Location	Text & Map	Required	Output	Static map and text address.
Time of Incident	Timestamp	Required	Output	The time the accident was detected.
Accident Photos	Image Gallery	Required	Output	Key-frame snapshots captured by the AI.
Event Summary (LLM)	Text Block	Required	Output	A natural language summary of what happened (e.g., "A blue sedan collided with a white truck...").
Agency-Specific Info (LLM)	Text Block	Required	Output	<p>This section is tailored.</p> <ul style="list-style-type: none"> Hospital: "Estimated Injuries: 3"

				<ul style="list-style-type: none"> Najm: "Fault Assessment: 70% Blue Sedan, 30% White Truck." Police: "Vehicles: 2. Traffic Impact: High."
Action Log	Text	Required	Output	A log of actions (e.g., "Alert sent to KFH 14:32:01", "Alert accepted by KFH 14:32:08").

Table 26: Accident Alert Interface

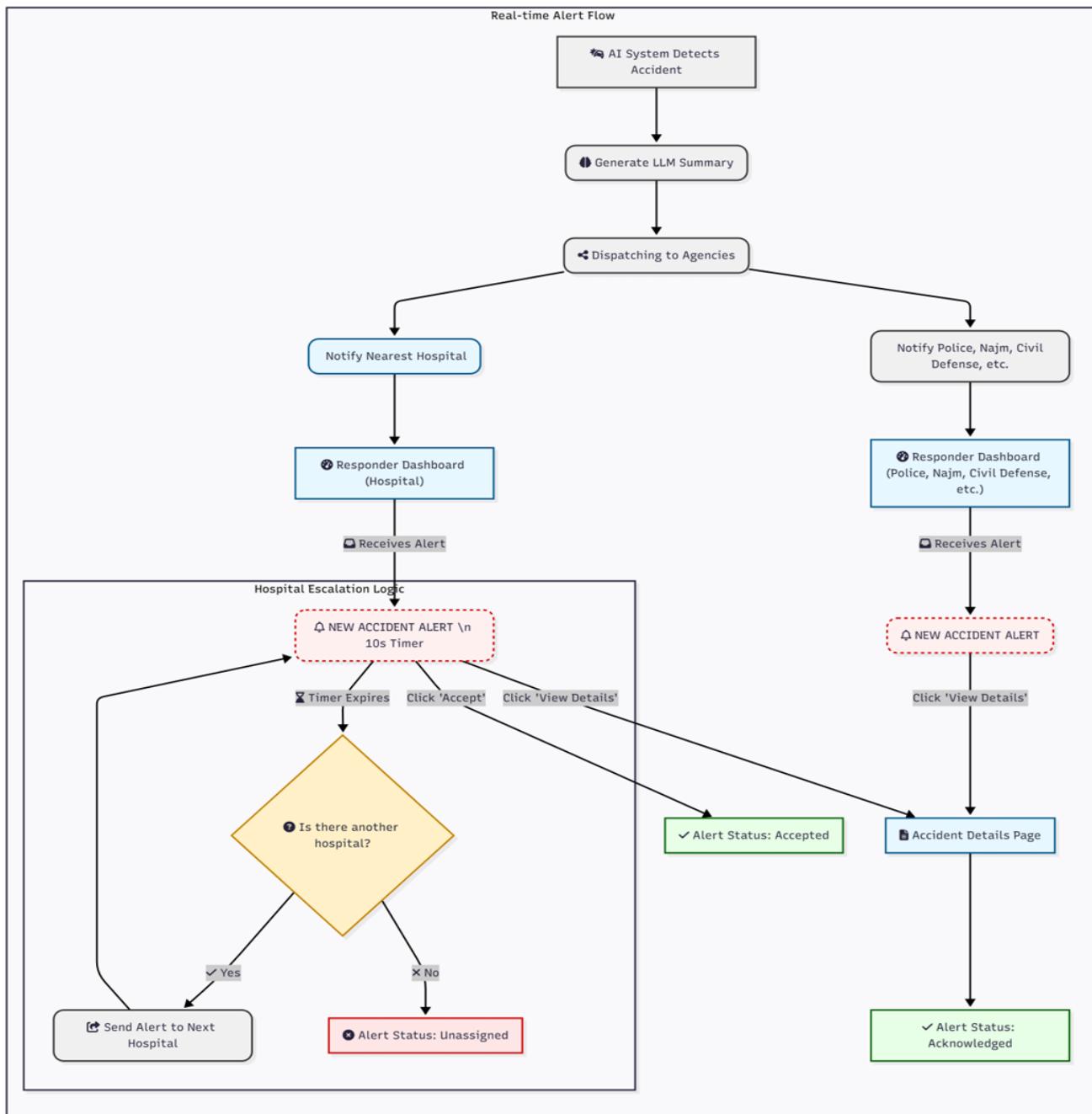


Figure 8: AI Alert System Flow

4.3.1.2 Hardware interfaces

- CCTV Cameras:** The system will connect to IP-based surveillance cameras, to which it will purchase discoverable video streams (e.g. by RTSP or HLS).
- Application Server:** The back end of the system is the system of AIs and a web server that will run on a high-performance server, which usually has between one and more GPUs to able deep learning inference.

- **User Devices:** The web-based user interface is going to be available on the normal desktop computers (e.g. dispatch centers) and also on the handheld devices (i.e. field responders) using modern web browsers.

4.3.1.3 Software interfaces

- **AI Backend:** The AI models (TensorFlow or PyTorch) will be serving out of a Python-based backend, based on frameworks like Flask or FastAPI, to the business logic of the system.
- **Web Frontend:** The client-side part will be a one-page application programmed in JavaScript, namely, in the React framework.
- **Database:** Data on users, camera metadata and logs of incident events will be stored in a relational database (or PostgreSQL) or a non-relational database (or mongoDB).
- **LLM API:** The system will be calling upon external application programming interfaces to a big language model, like the Gemini API of Google, to produce brief summaries of accidents.
- **Mapping API:** The system will connect to a cartographic web service (e.g., OpenStreetMap or the Google Maps API) to map spatially incident locations.
- **Notification API:** The system will be connected to a third-party telephony or short messaging service, like Twilio, in order to send emergency call alerts to responding agencies.

4.3.1.4 Communication interfaces

- **HTTPS:** The entire traffic between the users and the webserver will be encrypted using the HTTPS protocol.
- **RTSP/HLS:** The system will use the Real Time Streaming Protocol (RTSP) or the HTTP Live Streaming (HLS) to feed on the video feeds of cameras.
- **WebSockets:** To ensure the delivery of real-time alerts to the Responder Dashboard, the server will use a persistent WebSocket connection to dispatch the information.
- **REST API:** The Python backend will be connected to the React-based frontend through a secure RESTful API to exchange all the ancillary data transactions (e.g., authentication, the retrieval of inventories of the cameras).

Chapter 5: Software Design Specification (SDS)

5.1 Introduction

This Software Design Specification (SDS) details the architecture and organization of the "Harnessing Deep Learning to Optimize Emergency Response" system. It defines the architectural decisions, subsystems, interfaces, and data structures necessary to implement real-time video processing, deep learning accident detection, and automated emergency alert generation, serving as the essential blueprint for developers.

5.1.1 Purpose

The purpose of this SDS is to describe the complete design blueprint of the Real-Time Accident Detection and Emergency Alert System, outlining the system architecture, major components, data design, interfaces, and operational workflows. This document translates the functional and non-functional requirements defined in the SRS into detailed design elements that guide development, integration, and testing. It provides a shared technical reference for developers, testers and supervisors to ensure consistent understanding of how the system processes CCTV video streams, detects accidents using deep learning models, and dispatches alerts with minimal latency. This SDS forms the foundation for implementation and future system verification activities.

5.1.2 Scope

The scope of this SDS covers the full design of the accident detection and emergency alerting system, including the video processing pipeline, deep learning inference modules, alert dispatch mechanisms, data structures, and interface definitions. It describes the system's architecture, constraints, operational behavior, and component responsibilities required to support continuous real-time monitoring. While the SDS focuses solely on the design of the software, related details such as testing procedures, deployment plans, and management activities are addressed in separate project documents.

5.1.3 Definitions, Acronyms, and Abbreviations

- **Definitions**

- **Accident Detection:** the entire procedure of letting machines see and learn to detect accidents through the application of state-of-the-art technology in vision processing and deep learning is carried out using video clips in real time monitoring.
- **Alert Generation:** immediately after the accident is detected, the alerts are generated and sent to the emergency response services.
- **Real-time Processing:** it is a video processing performance that recognizes accidents at a maximum duration of 2 seconds (the target is to make the delay shorter than 2 seconds).
- **Incident Log:** A persistent, timestamped record of every detected and confirmed accident, stored in the database for historical review and system auditing.
- **User Authentication:** The process of verifying a user's identity (Admin or Responder) before granting access to the system's dashboards and data.

- o **Latency:** The time delay measured from when an event occurs in the video stream to when the system reports the final alert.

- **Acronyms**

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
ARTI	Artificial Intelligence (course code)
AWS	Amazon Web Services
CNN	Convolutional Neural Network
CCTV	Closed-Circuit Television
CVAT	Computer Vision Annotation Tool
GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
mAP	Mean Average Precision
OpenCV	Open Source Computer Vision Library
QA	Quality Assurance
SDS	Software Design Specification
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
STP	Software Test Plan
STS	Software Test Specification
UI/UX	User Interface/User Experience
YOLO	You Only Look Once (object detection architecture)

Table 27: Glossary of Terms and Acronyms.

5.1.4 References

- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans, Institute of Electrical and Electronics Engineers, 1998

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, Institute of Electrical and Electronics Engineers, 1998
- IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions, Institute of Electrical and Electronics Engineers, 1998
- ARTI 511 Course Syllabus, Term 1 2025-2026, Imam Abdulrahman Bin Faisal University, College of Computer Science & IT
- Ultralytics YOLOv8 Documentation, Available at <https://docs.ultralytics.com>, accessed October 2025
- TensorFlow Documentation, TensorFlow Core v2.20, Available at <https://www.tensorflow.org>
- PyTorch Documentation, PyTorch v2.9.0, Available at <https://pytorch.org/docs>
- OpenCV Documentation, OpenCV 4.12.0, Available at <https://docs.opencv.org>

5.2 System overview

The Harnessing Deep Learning to Optimize Emergency Response system is an autonomous, real-time traffic monitoring solution designed to reduce accident response times in urban areas. The system functions as a continuous intelligence pipeline that monitors live video feeds from cameras, detects car crashes using deep learning models, assesses incident severity, and automatically generates tailored alerts for first responders. It operates primarily as a background service with a dedicated web interface for authorized personnel (Police, Hospital, Fire Services) to view and manage alerts in real time. The core design is built for high availability and low latency, aiming to confirm and dispatch an alert in the least possible time after an accident occurs on the video stream.

5.2.1 System Functionality

The system is designed to fulfill three primary roles:

1. **AI Detection System:** The core intelligence, responsible for video ingestion, frame-by-frame analysis using the AI models (YOLOv8/CNN) and generating a structured incident report upon confirmation.
2. **Alerting and Communication Engine:** The module responsible for taking the structured incident report, generating a natural language summary using a Large Language Model (LLM), determining the nearest relevant responder, and initiating the alert escalation.
3. **Presentation and Management Layer:** serves as the user-facing component of the system, implemented as a responsive web application. It is the centralized hub for system configuration and real-time incident response, ensuring that all user interactions are secure and role specific.

5.2.2 User Functionality

The system operates with two primary user roles: Admin and Responder. The following section outlines the specific functionalities granted to each.

5.2.2.1 Common Functionalities

Both Admin and Responder users can log into the system to access their functionalities.

- Enable Secure Authentication (Sign In/Sign Out).
- Allow Password Reset/Retrieval.
- Allow users to Update Profile (change contact info, password).
- Receive system Notifications and alerts.

5.2.2.2 Admin Functionalities

The Admin has full control over system configuration, user access, and resource management.

- Manage user accounts (Add, edit, and delete users).
- Manage agency permissions and roles for external units (e.g., Police, Civil Defense, and Hospitals).
- Add and remove hospitals and other response agencies from the system contact list.
- Manage surveillance infrastructure (Add, remove, and monitor cameras).
- Review and Export the System Audit Log.

5.2.2.3 First Responders Functionalities

The Responder manages real-time emergency incidents and coordinates response units.

- View Real-Time Alerts and Incidents (list and map view).
- Accept incoming alerts and initiate the response process.
- View detailed Accident Details (location, time, severity).
- Log all actions taken during incident response.
- View and Create New Incident Reports.
- Generate PDF reports for individual cases

5.3 Design Considerations

This section outlines the key issues, constraints, assumptions, and external factors that influence the system architecture and design decisions. Understanding these considerations ensures that the proposed design is realistic, maintainable, and compatible with the operational environment.

5.3.1 Assumptions and Dependencies

The design of the system is based on several assumptions and external dependencies that may affect its functionality or performance. These must be acknowledged to ensure consistency in implementation and deployment.

5.3.1.1 Related Software or Hardware

The system depends on the presence and performance of the following hardware and software components:

- The system requires working traffic or surveillance cameras providing stable, continuous video streams with adequate resolution and coverage for accurate accident detection.
- Processing hardware, either on a server or local machine, equipped with GPUs for deep learning tasks. Cloud-based GPU platforms can be used for training and inference if local resources are inadequate.
- **Software Dependencies:**
 - Python
 - Deep learning frameworks (TensorFlow / PyTorch)
 - OpenCV for video handling
 - Flask / Django for backend APIs
 - YOLOv8 model and necessary libraries
 - CVAT/LabelImg for dataset annotation

5.3.1.2 Network, Connectivity, and Streaming

- The system assumes stable network connectivity to continuously receive CCTV video streams without interruptions.
- Adequate bandwidth is required to handle real-time feeds from all connected cameras.
- Video streams should maintain a consistent frame rate (15–30 FPS) and resolution (720p or higher) for accurate detection.
- Low network latency is required to ensure real-time accident detection and timely alert generation.

5.3.1.3 Dataset Availability and Quality

- The system assumes access to publicly available accident and non-accident video datasets suitable for training and evaluating the deep learning models.
- It depends on accurate and consistent manual annotations (bounding boxes, accident labels) to ensure proper model learning.
- Training data is assumed to include sufficient variability in lighting, weather, camera angles, and traffic conditions to generalize well to real CCTV environments.
- The quality and resolution of dataset videos are assumed to be adequate for object detection models such as YOLOv8 (e.g., 720p or higher).

- Model performance is limited by the diversity and realism of the datasets, rare accident types may be underrepresented.

5.3.1.4 End-User Characteristics

The system will serve two main user groups assuming that they are familiar with basic computer systems and emergency response workflows:

1. Emergency Responders

- Require real-time, accurate alerts with minimal interface complexity.
- Typically non-technical users who rely on clear visuals and actionable information.
- Expect alerts containing time, location, severity, and detection confidence.

2. System Administrators

- Responsible for managing camera inputs, system updates, logs, and performance monitoring.
- More technically proficient (IT staff or trained operators).
- Require dashboard access to models, logs, camera settings, and alert statistics.

5.3.1.5 Possible And/Or Probable Changes in Functionalities

- **Integration with Real CCTV Networks:**

Currently, training uses public datasets. In the future, integration with live municipal camera networks may require changes to ingestion modules.

- **Scalability Requirements:**

The design may need to accommodate:

- More cameras
- Higher traffic density
- Multiple processing nodes

- **Deployment Changes:**

Early versions may run locally, final versions may require:

- Edge devices for real-time inference
- Cloud-based distributed processing

- **Model Updating:**

Improvements in AI models may trigger migration to more efficient architectures.

5.3.2 General Constraints

The following inherent limitations have a great influence on software design and the consequent impact on the architectural and implementation options.

5.3.2.1 Hardware / Environment

- **Constraint:** Deployment involves the use of common computing nodes that have real-time video processing (edge devices or small servers).
- **Impact:** Model and pipeline optimization (frame sampling, model compression/quantization) in such a way that the system can meet low inference latency and fixed throughput on a limited hardware.

5.3.2.2 Software Stack & Libraries

- **Constraint:** System depends on Python 3.x and key libraries (OpenCV, PyTorch or TensorFlow for training, TensorRT for edge inference).
- **Impact:** Fix compatible library versions and provide reproducible environments to prevent conflicts during development and deployment.

5.3.2.3 Network & Camera Protocols

- **Constraint:** Live video is fed by CCTV/traffic video cameras using RTSP, nets may be limited by bandwidth or behind a NAT.
- **Impact:** Implement adaptive streaming, local buffering, and connection-resilient logic to maintain service despite variable network conditions.

5.3.2.4 Video Sources & Robustness

- **Constraint:** Video sources are heterogeneous (different codes, resolutions, frame rates) and connections may be intermittent.
- **Impact:** Use adaptive frame sampling, resolution fallback, and reconnect strategies so detection quality degrades gracefully rather than failing outright.

5.3.2.5 Performance & Latency

- **Constraint / Design goal:** The system should be capable of real-time or near real-time detection of accidents with low end to end delay.
- **Impact:** This involves the employment of efficient model architectures, regulation of the per-camera processing load, the employment of asynchronous pipelines, and minimization of the needless UI updates to make the system responsive to different workloads.

5.3.2.6 Storage & Capacity

- **Constraint:** Limited local storage; video and logs grow rapidly.
- **Impact:** Enforce rolling logs, TTL-based object retention, and configurable upload/archival to central storage.

5.3.2.7 Interoperability & Interfaces

- **Constraint:** It needs to be an always-on integrated system with external systems (including traffic control APIs and SMS gateways, webhooks).
- **Impact:** specification of stable JSON schemas, authentication (api keys/ HMAC) and failing on delays / backoff behavior when making external calls.

5.3.2.8 Security & Privacy

- **Constraint:** Protected data and privacy regulations apply; system must secure PII and video material.
- **Impact:** Encrypt data at rest and in transit, implement RBAC, audit logging, and optional face-masking/anonymization features.

5.3.2.9 Standards & Documentation

- **Constraint:** Deliverables should also be based on institutional and IEEE-style of documentation and traceability practices.
- **Impact:** Have versioned documents, requirements traceability matrix and SDD/TEST change control.

5.3.2.10 Verification & Validation

- **Constraint:** Acceptance must be proven to display accuracy, latency and combination tests with representative streams.
- **Impact:** Test evidence and metrics Staged testing (unit) to simulated streams to live pilot.

5.3.3 Security and Threat Considerations

The following part introduces a STRIDE-inspired threat model that highlights the primary security risks facing the emergency response system. STRIDE inspects Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege on video inputs, AI verification, notifications, and operator controls. All threats have corresponding countermeasures indicated to the respective SDS components thus ensuring an all-round security design.

Threat Category	Potential Threats	Impact	Mitigation (SDS Reference)
Spoofing	Unauthorized access to responder dashboard	High (false alerts)	JWT auth, role-based access (5.4 User Interface Design)
Tampering	Altered video feeds or alert data	Critical (wrong response)	HMAC signatures, input validation (5.8.4 Resources)
Repudiation	Unlogged alert acceptance	Medium (accountability)	Audit logs in all actions (5.6.3 Database Description)
Information Disclosure	Exposure of accident videos/GPS	High (privacy)	TLS 1.3 everywhere, field-level encryption (5.3.2 Constraints)
Denial of Service	Flooded streams crashing detection	High (outages)	Rate limiting, auto-scaling (5.8.5 Processing)
Elevation of Privilege	Admin escalation via UI flaws	High	Least privilege, input sanitization (5.7 Components)

Table 28: STRIDE Threat Model

5.4 User Interface Design

This section outlines the user interfaces and their design principles, including example screens and a description of the main screen elements and their actions.

5.4.1 Overview of User Interface

The user interface of the system is a secure role-based web platform and is equipped to work in real time. Depending on the role assigned to the user, the user is allowed access to certain functionality which is verified at the login point. There are two primary perspectives:

1. **System Administrator:** The perspective of the administrator is centered on the health and set up of the system. Their primary dashboard will display the high-level view of the key performance indicators (KPI) such as the state of the system, active cameras, and the quantity of active incidents. At this main hub, they will be able to go out to modules and make use of the core parts of the system. This contains a Camera Management area to view, add or delete surveillance feeds, and User Management area where accounts to various responder agencies can be controlled. A detailed Audit Log also allows the administrators to overview all the major activities performed in the system in terms of its security and responsibility.
2. **First Responder:** The view of the responder is functional and incident oriented. They are designed based on their experience. The main dashboard is an Active Incidents map which shows the actual geographical placement of all the emergencies in real-time and supplemented by a list of the most recent ones. This perspective is updated by default when the AI identifies new events. On the Incident Report page, rescuers are able to view a history of all incidents, which can be filtered and viewed chronologically. The selection of a particular incident will lead to an Accident Details screen, where all important information, such as a map and computer-generated summaries, will track the response between the detection until the resolution.

5.4.2 Interface Design Rules

The interface design is driven by the fact that it is necessary to be clear and fast in high pressure conditions.

- **Clarity and Readability:** The interface is developed on dark-mode theme with high contrast to reduce eye strain, which is normal in 24/7 command center layout. The display of the information is done in a clear typeface and understandable icons.
- **Role-Based Simplicity:** Role-based access is strictly adhered to in the system. There is only a navigation sidebar that contains links that are specific to the role of the logged-in user. This eliminates distractions of administrative environments to responders and secures sensitive settings to the operating personnel.
- **Visual Status Feedback:** The system provides a visual representation of the status of some of the critical components. Cameras are also specifically marked as Online or Offline, and the incidents marked with their status (e.g. New, Acknowledged, Scene Cleared) so that one can get an immediate overview.
- **Layout Consistency:** Each of the screens has a similar layout: on the left there is the main navigation sidebar, on the header is the title of the page, and there is a main content area. This

uniform design enables the user to learn their way around the various modules easily and in a predictable fashion.

5.4.3 Screen Images

This section provides graphical representations of the most important user interfaces (UI) of the system, its structure, components and behavior. Each visual is described with a short explanation of what the interface is and how the user interacts with the interface.

5.4.3.1 Login Interface

This is the secure entry point for all users. The interface is divided into two parts where the system branding and other important functionalities (Real-Time Detection, Multi-Agency Coordination) appears on the left, and the login form is situated on the right. To access it, users fill their email and password. Password retrieval and fast access links are provided depending on the user roles.

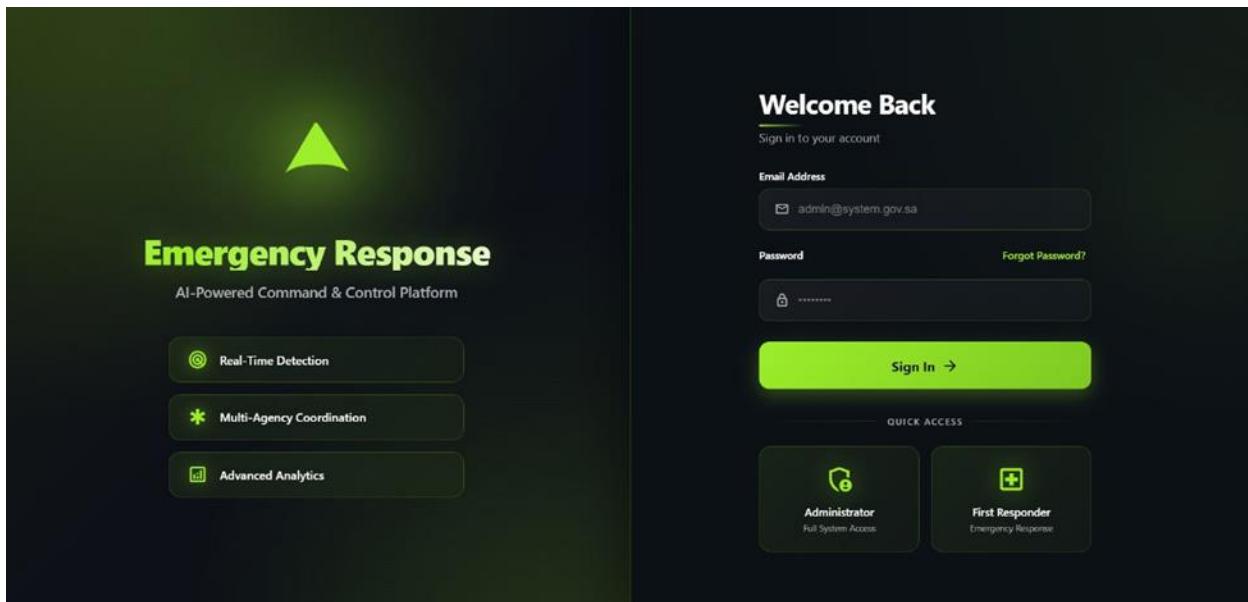


Figure 9: Secure emergency response login screen with system branding and role-based sign-in.

5.4.3.2 Reset Password Interface

It is a screen on which one can access a forgotten password. The user is requested to enter her registered email. It will then send by email, a reset link or verification code, to their email, to enable them to securely establish a new password.

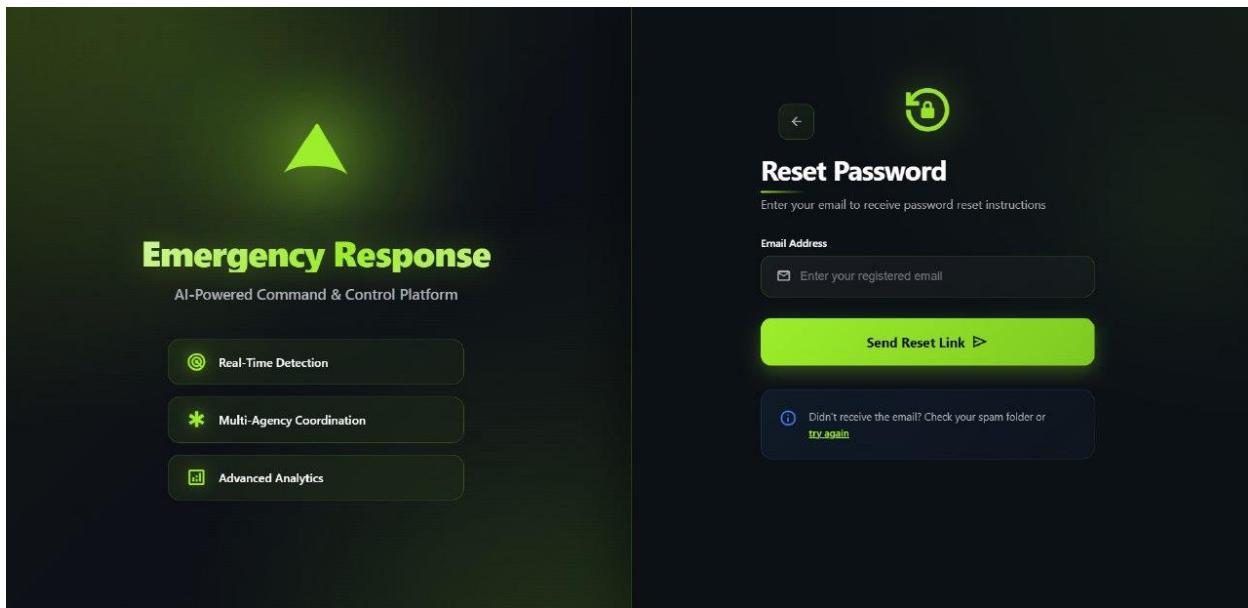


Figure 10: Reset Password Interface.

5.4.3.3 Administrator - Operational Dashboard

This is a landing page intended to be used by System Administrators and gives them a general report of the overall system health. It contains the main metric cards, which are the number of ongoing incidents, the number of active cameras, and the average response time.

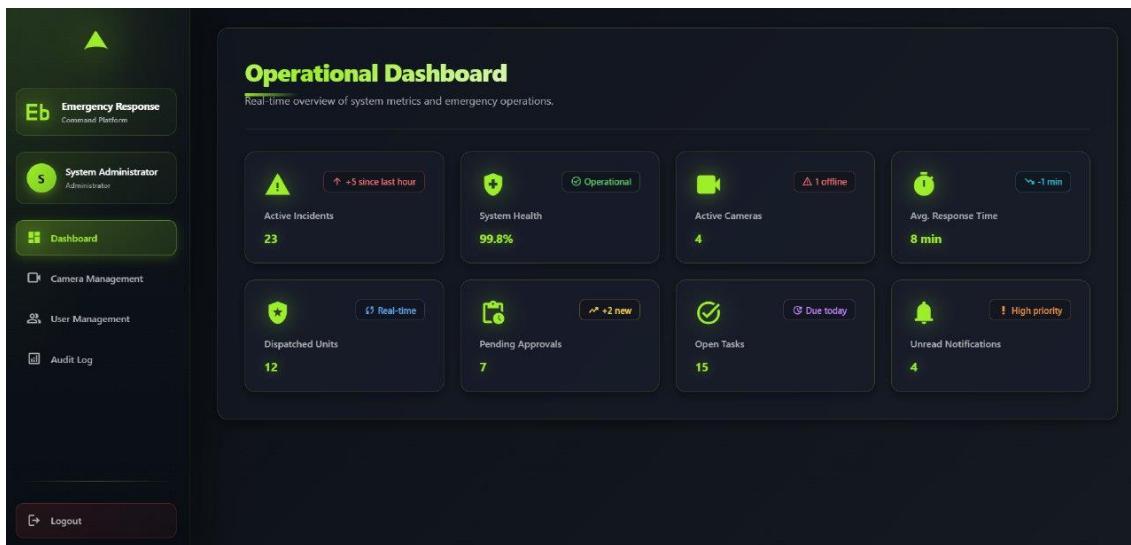


Figure 11: System Administrator Operational Dashboard.

5.4.3.4 Administrator - Camera Management

Through this screen, administrators can keep track of all the surveillance cameras attached and make all the necessary arrangements. It presents an outline of online, offline, and the total cameras. The main screen displays a grid of all registered cameras, each with their name, location and an obvious Online or Offline status badge. Via this page, administrators can add new cameras or update/delete those that are already in existence.



Figure 12: Camera Management Interface.

5.4.3.5 Administrator - User Management

The administrators use this interface to maintain the user accounts of all the external agencies. It gives a list of registered agencies (e.g. hospitals, police departments, civil defense) in a card format. The administrators have the capability to add new users to the agency or update and remove accounts to manage the system access.

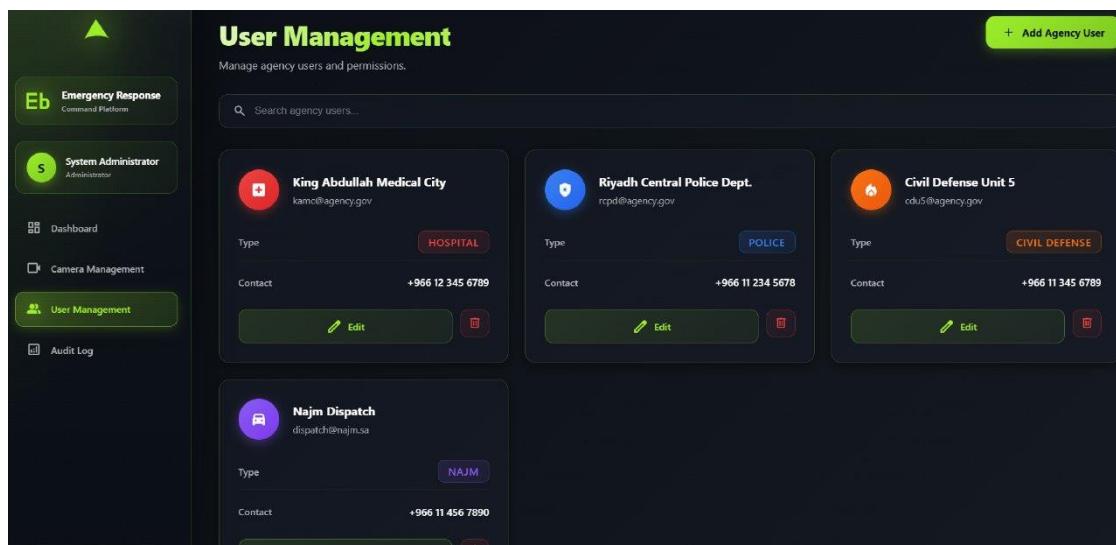


Figure 13: User Management Interface.

5.4.3.6 Administrator - System Audit Log

This screen provides a comprehensive, chronological record of system activities to ensure accountability and facilitate troubleshooting. The log captures all critical events, including user logins, camera additions / deletions or incident notifications, the user who performed the action, time and IP address. The log may be sorted by date, user or type of action.

The screenshot shows the 'System Audit Log' page. On the left is a sidebar with icons for Emergency Response (Eb), System Administrator (S), Dashboard, Camera Management, User Management, and Audit Log (highlighted in green). The main area has a title 'System Audit Log' and a subtitle 'Complete chronological record of system activities'. It includes filters for 'Date Range', 'By User', and 'Action Type'. Below is a table of audit logs:

Date	User	Action	IP Address
11/13/2025, 10:26:45 PM	admin@system.gov	Created new camera: CAM-042	192.168.1.100
11/13/2025, 10:16:45 PM	admin@system.gov	Updated user permissions for responder@hospital.gov	192.168.1.100
11/13/2025, 10:01:45 PM	system	System backup completed successfully	127.0.0.1
11/13/2025, 9:46:45 PM	admin@system.gov	Deleted camera: CAM-021	192.168.1.100
11/13/2025, 9:31:45 PM	responder@hospital.gov	Acknowledged incident ER-20240521-0012	192.168.1.100

At the bottom right is a 'Logout' button.

Figure 14: System Audit Log Interface.

5.4.3.7 First Responder - Active Incidents Dashboard

This is the main operational dashboard of First Responders. The interface is centered around an interactive map displaying the real-time locations of all active incidents, utilizing color-coded pins to indicate severity levels. On the left, there is a list of Recent Incidents that gives an overview of the new events when AI detects them.

The screenshot shows the 'Active Incidents' dashboard. On the left is a sidebar with icons for Emergency Response (Eb), Dr. Sarah Ahmed (First Responder), Dashboard (highlighted in green), and Incidents. The main area has a title 'Active Incidents' and a subtitle 'Real-time monitoring and response coordination'. It shows a map with several colored pins (red, yellow, green) indicating incident locations. At the top right are counts for 'New: 1', 'Acknowledged: 1', and 'Total: 2'. Below the map is a section for 'Recent Incidents' with two entries:

- ER-20240521-0012** (High severity, red pin): King Fahd Road @ Olaya Junction. Multi-vehicle collision involving a commercial truck and two passenger cars. Reported 5m ago. [View Details](#)
- ER-20240521-0011** (Moderate severity, orange pin): Northern Ring Road @ Exit 7. Single vehicle collision with highway barrier. White sedan appears to have lost ... Reported 18m ago. [View Details](#)

At the bottom right is a 'Logout' button.

Figure 15: First Responder Active Incidents Dashboard.

5.4.3.8 First Responder - Incident Reports

This screen gives a historical view of all the past and ongoing incidents filterable. The responders are able to search according to the report ID or keyword and to filter according to the status (e.g., All, New, Acknowledged) or dates. The incidents are shown as cards that contain a summary, date and time with an option to open the full details.

Figure 16: First Responder Incident Reports Interface.

5.4.3.9 First Responder - Accident Details

This is a screen that contains all the information on one incident that has been selected. It also includes a huge map of the specific location, the essentials of the incident (time, location text), and an eminent Action Log. It is a critical feedback mechanism; this log displays responders a live-updating timeline of everything that is happening.

Figure 17: Accident Details Interface.

5.4.4 Screen Objects and Actions

In this section, the description of the common interface elements used in the system will be highly detailed, and the actions related to these elements will be described. In every screen, there are different interactive features, including buttons, links, icons and input fields, which allow users to navigate and interact with the system to its full capacity. The actions, type, and functionality of each element are described here to improve the interaction of the system as well as the usability.

Screen	Object Name	Object Type	Action / Description
Login	Email Address	Text Field	Allows the user to enter their registered email address.
	Password	Text Field	Allows the user to enter their secure password (input is masked).
	Sign In	Button	Submits the entered credentials and attempts to log the user into the system.
	Forgot Password?	Link	Navigates the user to the password recovery screen.
Reset Password	Back Arrow	Icon Button	Navigates the user back to the Login screen.
	Email Address	Text Field	Allows the user to enter their email to receive a password reset link.
	Send Reset Link	Button	Validates the email and starts the password recovery process.
Admin Dashboard	Navigation Sidebar	Link List	Main menu for admins to navigate between dashboard, camera, user, and log pages.
	Metric Cards	Display	Read-only cards showing key system statistics such as active incidents and health.
Camera Mgt.	Add Camera	Button	Opens a form for the administrator to register a new camera feed.
	Search cameras...	Search Field	Filters the camera list by name or ID as the admin types.
	View (Eye Icon)	Icon Button	Opens a live video preview for the selected camera.

	Edit (Pencil Icon)	Icon Button	Opens a form to modify the selected camera's settings.
	Delete (Trash Icon)	Icon Button	Removes the selected camera from the system after confirmation.
User Mgt.	Add Agency User	Button	Opens a form for the admin to create a new responder user account.
	Search agency...	Search Field	Filters the list of agency users by name as the admin types.
	Edit (on card)	Button	Opens a form to update the selected agency user's account details.
	Delete (Trash Icon)	Icon Button	Removes the selected agency user account after confirmation.
Audit Log	Filter Dropdowns	Dropdown	Filters log entries by criteria such as date, user, or action type.
	Export Log	Button	Downloads the filtered log as a file (for example, PDF or CSV).
Responder	Navigation Sidebar	Link List	Main menu for responders to switch between Dashboard (map) and Incidents views.
Dashboard	Recent Incidents	List	Live-updating list of new alerts; selecting an item opens its details.
	Incidents Map	Map	Displays pins for the locations of all active emergencies.
Incident Reports	Search by Report ID...	Search Field	Helps the responder find a specific incident by its report identifier.
	Status / Date Range	Dropdown	Filters incidents by status and/or the date range in which they occurred.
	View Details	Link	Navigates to the full details page for the selected incident.
Accident Details	Generate PDF	Button	Creates a downloadable PDF summarizing all information for the current incident.

	Action Log	List	Read-only timeline of all recorded response steps (for example, alert sent/accepted).
--	------------	------	---

Table 29: User Interface Controls and Interactions.

5.4.5 Other Interfaces

5.4.5.1 Video Stream Ingestion Interface

- **Description:** This interface takes the live video feeds of the IP-based surveillance cameras which form the main source of data in the system.
- **Technology/Protocol:** The system is intended to consume feeds provided by cameras with the help of standard streaming protocols like RTSP (Real-Time Streaming Protocol) or HLS (HTTP Live Streaming).
- **Initiation/Closure:** The server makes and provides a permanent connection to the stream URL of the camera. This connection will be closed in the event that the camera is pulled out of the system, or the service is discontinued.
- **Error Conditions:** Failing and interrupting of a stream connection will be recorded and a visual indicator reflecting the error will be displayed in the Camera Management dashboard showing the camera as being offline.

5.4.5.2 Client-Server API (REST)

- **Description:** This interface covers the main request-response of the user web-browser (frontend) to the backend server. It is utilized in the majority of actions that are initiated by a user, e.g., the logging in, camera list retrieval, or historical incident reports.
- **Technology/Protocol:** A secure **RESTful API** operating over **HTTPS**.
- **Message Format:** The exchange of all data is done through the use of JSON. An HTTP request is sent by the client (browser), and a server replies to it with a response in the format of JSON.
- **Error Conditions:** In case the request is rejected because of an authentication mistake (e.g. an expired session), the user is redirected to the login screen. In case the server has an internal error, a message of failed to load data will appear on the UI.

5.4.5.3 Real-time Alert Interface (WebSocket)

- **Description:** This interface offers two-way, real-time and constant communication between the server and the client. It is also critical in pushing live information to the customer, including new accidents and live updates to the Action Log, without the user having to refresh the page.
- **Technology/Protocol: WebSockets (WSS).**
- **Initiation/Closure:** A WebSocket connection is initiated by the browser of the client, once the client has successfully logged in. This relationship is open until the user logs off or closes the browser tab.

5.4.5.4 External Third-Party Service Interfaces

- **Description:** This stipulates how the system is related to the different external third-party services that offer expert, non-core services.
- **Technology/Protocol:** All external services are accessed via secure **HTTPS REST APIs**.
- **Services Used:**
 - **AI/LLM Service:** Used to send incident data and receive the AI-generated natural-language summaries for accident reports.
 - **Mapping Service:** Used to fetch the map tiles and geocoding data required to display the maps on the dashboards.
 - **Notification Service:** Meant to send alerts (e.g. SMS) to agency contact numbers in case of failure.

5.5 System Architecture

The system architecture provides a high-level view of how the Smart Emergency Response and Monitoring System is organized and how its main functions are structured and connected. This section describes the major subsystems, their responsibilities, and the way they interact to deliver real-time accident detection, event management, and emergency response coordination. The goal of this architecture is to ensure a modular, scalable, and efficient design where each component performs a specific role while working together seamlessly to support the overall system functionality.

5.5.1 Architectural Design Approach

The architectural design of the System follows a modular, layered, and service-oriented approach to ensure scalability, maintainability, and reliable performance during emergency situations. The system integrates multiple data sources such as live camera feeds, emergency unit locations (hospitals, fire departments, police), and administrative interactions. therefore requiring a clear and logical partitioning of responsibilities across subsystems.

High-Level Design Philosophy:

The system architecture was structured using **three** guiding principles:

1. **Separation of Concerns:** Each subsystem handles a distinct functional responsibility (e.g., video processing, emergency unit management, user dashboard). This prevents overlap and reduces system complexity.
2. **Distributed Responsibility:** To support real-time performance, processing-intensive tasks (e.g., accident detection using AI models) are isolated in specialized modules that operate independently but communicate via defined interfaces.
3. **Scalability and Extensibility:** The design supports future additions such as new camera locations, integrating new emergency departments, or improving AI models without affecting the entire system.

The system was decomposed into **subsystems** based on the **nature** of the tasks and data flow:

1. **Video/AI processing** tasks are computationally heavy and best isolated to avoid affecting user dashboard performance.
2. **Emergency resource management** (hospitals, fire departments, police units) requires a structured backend subsystem for CRUD operations and live status management.
3. **Dashboard visualization** is user-facing and must remain responsive; therefore, UI components are separated from backend logic.
4. **Database operations** are centralized to ensure data integrity and avoid duplication.
5. **Notification and alerting** mechanisms require a dedicated module to manage timing, communication, and routing of alerts.

Alternative architectures such as monolithic structure were rejected because they would make the system harder to scale, harder to maintain, and prone to performance bottlenecks especially around real-time video analysis.

At the highest level, the system must support:

1. Real-time video collection and processing

2. Automatic accident detection using AI/ML techniques
3. Storing and retrieving system data (cameras, emergencies, units)
4. Managing emergency response units (hospitals, firefighters, police)
5. Displaying a centralized monitoring dashboard
6. Generating alerts and event logs

Collaboration of Higher-Level Components

The collaboration between these subsystems follows a pipeline-like workflow:

1. **Camera feeds**: processed by the Video Processing Subsystem
2. **Detected accident**: forwarded to Event Management Subsystem
3. **Event details**: stored in Data Management Subsystem
4. **Affected units**: retrieved via Emergency Units Subsystem
5. **Alert + details**: pushed to Notification Subsystem
6. **Updated information**: displayed on Dashboard Subsystem

This layered interaction ensures that no subsystem is overloaded and that responsibilities are clearly distributed.

5.5.2 Architectural Design

The system is divided into six main modules:

1. **User Interface & Dashboard Module**
2. **Video Acquisition & Accident Detection Module**
3. **Emergency Event Management Module**
4. **Emergency Units Management Module**
5. **Notification & Alerting Module**
6. **Data Management & Storage Module**

Each module is independent but interconnected to support real-time detection, reporting, and management of emergency events.

1. User Interface & Dashboard Module

Role: Displays live camera feeds, accident alerts, emergency units, analytics, and management controls.

Responsibilities:

- Fetch system data through backend APIs
- Visualize accident events in real-time
- Provide administrators with CRUD interfaces for cameras, hospitals, and departments

Collaboration:

- Retrieves processed results from Event Management
- Displays data stored in the Data Management subsystem

2. Video Acquisition & Accident Detection Module

Role: Processes live camera feeds and detects accidents using computer vision / machine learning.

Responsibilities:

- Connect to camera streams
- Run AI models
- Send detection results to the Emergency Event Management Module

Collaboration:

- As soon as an accident is detected, it notifies the Event Management module
- Logs critical frames into the database

3. Emergency Event Management Module

Role: Central controller of all accident-related events.

Responsibilities:

- Create and manage accident event objects
- Update event status (detected, verified, resolved)
- Coordinate with Notification and Units Management modules

Collaboration:

- Receive detection signals from the Accident Detection module
- Stores event details in the Data Management subsystem
- Send alerts to the Notification Module

4. Emergency Units Management Module

Role: Manages hospitals, fire departments, police units, and their statuses.

Responsibilities:

- Add, remove, or edit emergency unit details
- Maintain live availability status
- Provide nearest or available units to an event

Collaboration:

- Supplies Event Management with recommended response units
- Stores updated unit information in the database

5. Notification & Alerting Module

Role: Sends notifications to dashboard users and logs event updates.

Responsibilities:

- Issue alerts for new accidents
- Update users when event statuses change
- Integrate with future communication channels (SMS, email, etc.)

Collaboration:

- Triggers alerts based on data from Event Management
- Sends UI update signals to the Dashboard module

6. Data Management & Storage Module

Role: Provides centralized storage for all system data.

Responsibilities:

- Maintain camera data
- Store accident events, timestamps, severity
- Store emergency units information
- Provide APIs/repositories for all modules

Collaboration:

- Serves as the shared data layer for all modules
- Ensures data consistency and integrity

5.5.3 Subsystem Architecture

The system was decomposed into the selected subsystems to maximize modularity, reduce coupling, and isolate high-processing tasks such as video analysis into independent modules. Separating event orchestration from unit management ensures clearer responsibility boundaries and makes the system easier to scale and maintain. A dedicated notification subsystem improves reliability and avoids mixing alert logic inside business modules. A centralized data subsystem ensures consistency across all modules. Alternative designs such as a monolithic architecture were rejected due to performance, maintainability, and scalability limitations.

1. UI & Dashboard Subsystem

- **Subcomponents:**
 - UI Client (Browser / SPA) pages and components
 - UI API Gateway Proxy light server to route UI requests / SSE or WebSocket endpoints
 - Dashboard Renderer / Charts Module
 - Admin Management Pages (Cameras, Units)
- **Responsibilities:** render live feeds and event lists, allow CRUD for cameras/units, receive push updates.
- **Exports/Interfaces:** `GET /api/events`, `GET /api/cameras`, `WS /ws/updates`, `POST /api/cameras`, `PUT /api/units/:id`.

Internal Interaction Summary:

The UI Client sends all requests through the API Gateway Proxy, which forwards them to backend services. The Dashboard Renderer subscribes to WebSocket/SSE channels to display real-time accident updates. Admin Pages interact with the backend using CRUD APIs for cameras and units. All UI components rely on the shared data models retrieved from the Data Management subsystem.

2. Video Acquisition & Accident Detection Subsystem

- **Subcomponents:**
 - Stream Connector : connects to RTSP/HTTP camera endpoints
 - Frame Capture & Buffer frame extractor, keyframe storage
 - Preprocessing Pipeline resizing, normalization
 - Detection Engine ML model(s) (object detection, anomaly classifier)

- Local Cache / Short-term Storage stores recent frames and detection metadata
- Detector-to-Event Adapter formats detection into Event object and posts to Event Management
- **Responsibilities:** reliably ingest streams, run detection models in near real-time, forward detections.
- **Exports/Interfaces:** `POST /events/detections` (internal API), message queue topic `detections`.

Internal Interaction Summary:

The Stream Connector ingests live video streams and passes frames to the Frame Capture & Buffer module. Frames are preprocessed and forwarded to the Detection Engine, which runs the AI inference. Detection results are temporarily stored in the Local Cache and then converted to event-ready messages by the Detector-to-Event Adapter before being sent to the Event Management subsystem.

3. Emergency Event Management Subsystem

- **Subcomponents:**
 - Event Orchestrator creates/manages Event lifecycle (detected → verified → assigned → resolved)
 - Event Repository Adapter persists events via Data Management
 - Verification Service optional manual verification workflow (operator confirmation)
 - Assignment Engine selects candidate units (calls Units Management)
 - Event API layer APIs for CRUD and status updates
- **Responsibilities:** authoritative source for all event state, coordination with notification and units.
- **Exports/Interfaces:** `POST /events`, `GET /events/:id`, `PUT /events/:id/status`, publishes to `events` topic.

Internal Interaction Summary:

The Event Orchestrator manages the lifecycle of each event and interacts with the Verification Service when operator approval is required. The Assignment Engine retrieves available units from the Units Management subsystem and updates the event record via the Event Repository Adapter. The Event API exposes endpoints to the UI and publishes updates to the Notification subsystem.

4. Emergency Units Management Subsystem

- **Subcomponents:**
 - Units Repository CRUD for hospitals/fire stations/police units
 - Geolocation & Proximity Service compute nearest units to an event
 - Availability Tracker tracks unit status (available/assigned/offline)
 - Units API layer APIs for unit data and live status updates
- **Responsibilities:** maintain unit catalog and availability, respond to assignment queries.
- **Exports/Interfaces:** `GET /units/nearby?lat=&lon=&radius=`, `PUT /units/:id/status`.

Internal Interaction Summary:

The Units Repository stores all unit entities, while the Availability Tracker keeps status information updated. The Geolocation & Proximity Service computes the nearest units for an event. The Units API layer exposes this data to the Event Management subsystem to facilitate automatic or manual assignment.

5. Notification & Alerting Subsystem

- **Subcomponents:**
 - Alert Router decides recipients and channels
 - Push Notifier WebSocket/SSE to UI clients
 - External Notifier Adapter connectors for SMS/Email (future)
 - Notification Audit Log store sent-notifications
- **Responsibilities:** reliably deliver alerts and updates; retry/fallback logic.
- **Exports/Interfaces:** POST /notify/event/:id, WebSocket channel /ws/updates.

Internal Interaction Summary:

The Alert Router receives event updates and determines routing logic. The Push Notifier delivers these alerts to dashboard users via WebSocket/SSE channels. The External Notifier Adapter prepares optional SMS/Email notifications. All outgoing alerts are logged by the Notification Audit Log.

6. Data Management & Storage Subsystem

- **Subcomponents:**
 - Relational DB (Postgres) canonical event, unit, user tables
 - Object Store (S3) video clips, keyframes, logs
 - Repository Layer / DAOs typed access to storage
 - Analytics Cache / OLAP store (optional) aggregated metrics for dashboard
- **Responsibilities:** durable persistence, transactional integrity, efficient queries for UI and services.
- **Exports/Interfaces:** DB schema, repository APIs, migration scripts.

Internal Interaction Summary:

The Repository Layer provides structured access to the relational database. The Object Store manages large media (e.g., detected accident frames). Analytics Cache stores aggregated metrics for fast dashboard queries. All subsystems interact with this module through repository classes or direct queries exposed by the API layer.

Functional high-level Data Flow Diagram (DFD)

[Camera Streams] → (Video Acquisition & Detection)
(Video Acquisition & Detection) → detection messages → (Event Management)
(Event Management) ↔ (Units Management)
(Event Management) → (Notification & Alerting) → (UI & Users)
(UI & Dashboard) ↔ (Data Management & Storage)
(All subsystems read/write) → (Data Management & Storage)

Key data items:

- Camera metadata (id, location, RTSP URL)
- Detection payload (timestamp, camera_id, confidence, bbox, clip ref)
- Event record (id, severity, status, assigned_units, timestamps)
- Unit record (id, type, location, availability, contact)

- Notification record (recipient, channel, status)

This DFD highlights a pipeline: *Ingest* → *Detect* → *Event* → *Assign/Notify* → *Display* → *Persist*.

Object-Oriented view (subsystem model & key classes)

core classes and relationships:

Class: Camera

- id, location, rtsp_url, status

Class: Detection

- id, camera_id, timestamp, confidence, bbox, clip_uri

Class: Event

- id, detections[], severity, status, assignedUnits[], created_at, updated_at

Class: Unit

- id, type, location, availability, contact

Class: Notification

- id, event_id, recipient, channel, sent_at, status

Sequence Diagram Descriptions

A. Accident Detection → Event Creation Sequence

1. Camera stream arrives to Stream Connector.
2. Frame is extracted by Frame Buffer.
3. Detection Engine analyzes frame and identifies accident.
4. Detector-to-Event Adapter sends a detection payload.
5. Event Management creates a new Event.
6. Event Repository saves event in database.
7. Notification Module sends “New Event” alert to UI.
8. UI Dashboard updates in real-time.

B. Event Verification and Unit Assignment Sequence

1. Operator views event on UI dashboard.
2. Operator clicks "Verify Event".
3. UI calls PUT /events/:id/status.
4. Event Orchestrator updates status to "verified".
5. Assignment Engine requests nearby units from Units Management.
6. Units Management returns list of available units.
7. Event Management assigns recommended units.
8. Notification Module sends assignment notifications.
9. Dashboard refreshes with assigned units.

5.6 Data Design

This section will represent an in-depth analysis of the data architecture of the system, which includes the description of how the information domain presented in the SRS, which consists of the real life objects, including users, surveillance cameras, and emergency incidents, is converted into structured and persistent data structures. The design can meet the important requirement of real-time responsiveness, high reliability and data integrity demanded to support emergency operations.

5.6.1 Data Description

The information domain of the system is pegged on four main concepts, including: Users (including administrators and responders), Cameras (which are the surveillance nodes), Alerts (which are the accident events), and the Audit Log (a detailed record of all the actions). This domain is then modeled and realized in form of normalized and relational database tables.

The key data entries are held and handled in the following fashion:

1. **User Data (Users):** It is a repository of all data necessary to carry out authentication (login credentials) and authorization (role). This data is greatly processed by the Web Server (API) to handle the user sessions and control access to system functions a process that is confirmed by the User Management and the login screen.
2. **Camera Data (Cameras):** This object maintains the parameters of configuration of each of the video streams being followed by the system. The Admin communicates with this organization through the Web API, and then the data is processed by the AI Processing Service. The AI service reads this table to determine the RTSP streams to read, which is confirmed by the “Camera Management” screen.
3. **Alert Data (Alerts):** This is the most vital and dynamic data represented by this entity. When an accident is detected, it is instantiated by the AI Processing Service. The object has AI-generated data (severity, summaries) and event data (time, location, human-readable ID). Then the Real-time Service and the Web Server (API) process it to record responder actions, and this is supported by active Incidents screen and the Accident Details screen.
4. **Log Data (AuditLog):** This is a system-wide log of all important actions. All other services update and create it. An example is Web Server logs which capture login and camera creation entries and AI Service logs capture alert detection, and Responder Portal logs capture alert acknowledgments. The given design is directly aligned with the screenshots of the System Audit Log and Action Log.

A PostgreSQL relational database will be used as the main data storage medium. This technology has been chosen due to its high reliability, transactional integrity (ACID compliance), and high ability to support standard relational data (users, cameras) as well as the semi-structured data (via JSONB or TEXT type when summarizing LLM results).

5.6.2 Data Dictionary

This table provides a comprehensive list and description of all data elements used in the system.

Entity	Attribute	Type	Constraints	Description
Alerts	alert_id	UUID	Primary Key	The internal unique identifier for this specific accident event.
	case_id	VARCHAR(50)	Not Null, Unique	The human-readable ID for display, (e.g., ER-20240521-0011).
	camera_id	UUID	Foreign Key (Cameras)	The camera that detected this accident.
	timestamp	TIMESTAMP	Not Null	The exact date and time the accident was detected by the AI.
	status	VARCHAR(50)	Not Null	The current state of the alert (e.g., "New", "Acknowledged", "Scene Cleared").
	severity	VARCHAR(50)	Nullable	AI-assessed severity (e.g., "Minor", "Moderate", "High", "Critical").
	llm_summary	TEXT	Nullable	Full, general-purpose LLM-generated summary of the event.
	llm_hospital	TEXT	Nullable	Tailored info for hospitals (e.g., "Estimated injuries: 3").
	llm_najm	TEXT	Nullable	Tailored info for Najm (e.g., "Fault assessment...").
	llm_police	TEXT	Nullable	Tailored info for police (e.g., "Traffic impact: High").
	assigned_to_user_id	UUID	Foreign Key (Users), Nullable	The user_id of the agency that officially accepted the alert.

<u>Alert Photos</u>	photo_id	UUID	Primary Key	A unique identifier for the snapshot.
	alert_id	UUID	ForeignKey (Alerts)	The parent alert this photo belongs to.
	image_url	VARCHAR(1024)	Not Null	The path (URL or S3 key) to the stored snapshot image.
	timestamp	TIMESTAMP	Not Null	The exact time the photo was captured from the stream.
<u>Audit Log</u>	log_id	UUID	Primary Key	A unique identifier for a single log entry.
	timestamp	TIMESTAMP	Not Null	The time this log event occurred.
	user_id	UUID	Foreign Key (Users), Nullable	The user who performed the action. Null if it's a 'system' action.
	ip_address	VARCHAR(45)	Nullable	The IP address of the user who performed the action.
	action_message	VARCHAR(1024)	Not Null	The log message (e.g., "Created new camera: CAM-042").
	entity_type	VARCHAR(50)	Nullable	The type of entity related to this log (e.g., "Alert", "Camera", "User").
<u>Cameras</u>	entity_id	VARCHAR(255)	Nullable	The ID (UUID or string) of the related entity (e.g., ER-20240521-0011).
	camera_id	UUID	Primary Key	The internal unique identifier for this camera.
	camera_string_id	VARCHAR(50)	Not Null, Unique	The human-readable ID for display (e.g., CAM-001-RUH).

Cameras	name	VARCHAR(255)	Not Null	A human-readable name (e.g., "King Fahd Road @ Olaya Junction").
	location_description	VARCHAR(512)	Nullable	A text address (e.g., "King Fahd Road, Olaya District, Riyadh").
	stream_url	VARCHAR(1024)	Not Null, Unique	The RTSP or HLS link for the video feed.
	location_lat	DECIMAL(9,6)	Nullable	Latitude for mapping the camera.
	location_long	DECIMAL(9,6)	Nullable	Longitude for mapping the camera.
	is_active	BOOLEAN	Not Null, Default: true	A toggle to enable or disable monitoring for this camera.
	added_by_user_id	UUID	Foreign Key (Users)	The user_id of the Admin who added this camera.
Users	user_id	UUID	Primary Key	The unique identifier for this user.
	email	VARCHAR(255)	Not Null, Unique	The user's login email address (e.g., admin@system.gov).
	password_hash	VARCHAR(255)	Not Null	The securely hashed password.
	role	VARCHAR(50)	Not Null	User's role (e.g., "Admin", "Hospital", "Police", "Najm", "Civil Defense").
	agency_name	VARCHAR(100)	Nullable	Human-readable name (e.g., "King Abdullah Medical City").
	contact_phone	VARCHAR(20)		Emergency contact number (e.g., +966 12 345 6789).

	created_at	TIMESTAMP	Not Null	When the user account was created.
--	------------	-----------	----------	------------------------------------

Table 30: System Data Dictionary.

5.6.3 Database Description

The database in the system is developed on the basis of the relational model to ensure the integrity of data, reduce redundancy, and have the single source of truth. The schema is brought to the Third normal form (3NF).

This goal is achieved through isolation of important ideas into separate tables. For example:

- On an Alert, there is no list of photos. Rather, the AlertPhotos table contains entries, which refer to the Alert, which is a typical one-to-many relationship, and thus, allowing an alert to have an unlimited number of photos attached to it.
 - AuditLog table is a central table holding all actions. This is a many-to-one relationship with Users (a user may be able to do many loggable actions). It also provides a polymorphic relationship (through entitytype and entityid) to the rest of the tables, thus allowing to filter according to a particular alert, user or camera. This design is directly involved in supporting the Action Log feature of the page of Accident Details.
 - Everything is connected by non-guessable UUIDs as primary keys to internal references with robust, secure internal references and user-interacting string IDs (caseid, camerastrongid) are used to display and interact with the user.

The diagram below will illustrate the database architecture, entities (Tables) and the relationships between them.

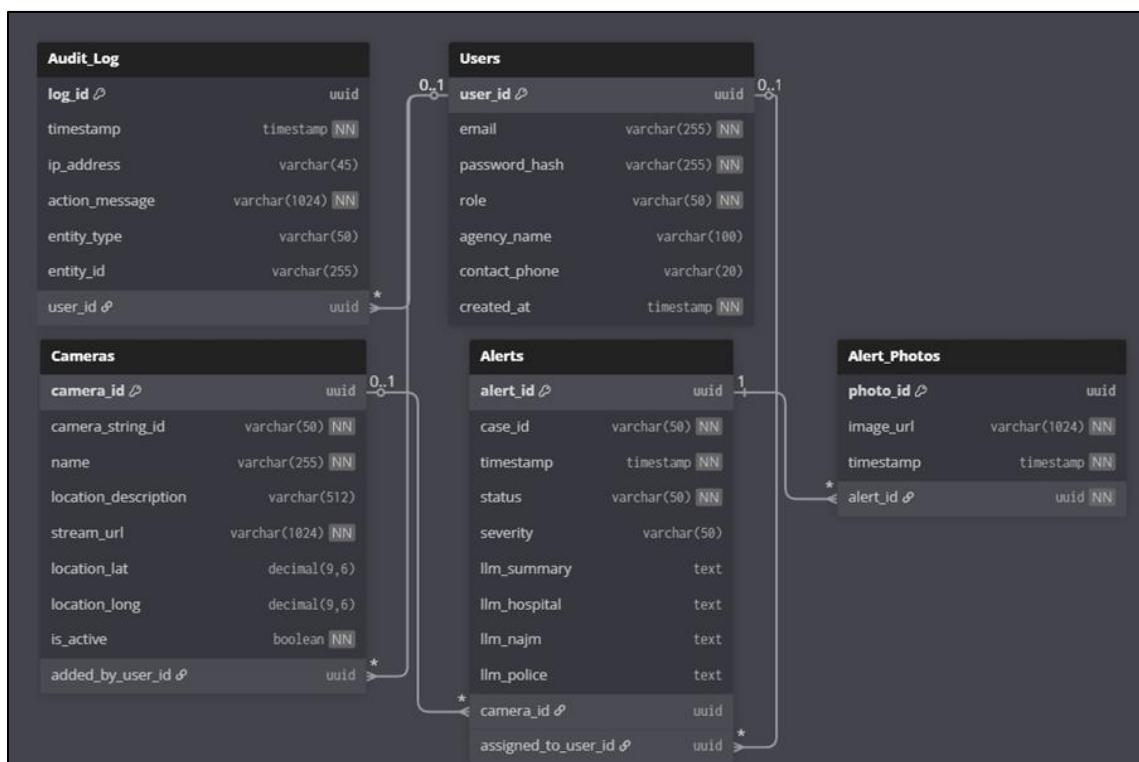


Figure 18: Physical Entity Relationship Diagram.

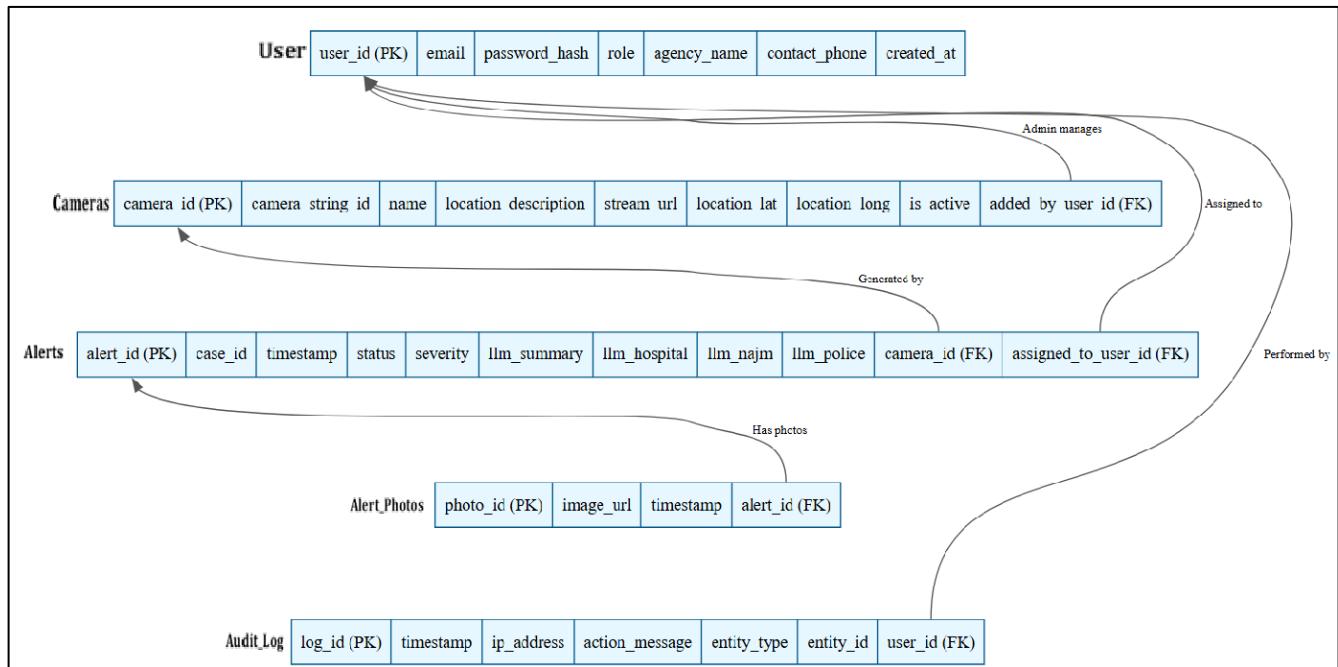


Figure 19: ERD Mapping Relational Schema.

5.7 Component Design

This section summarizes the internal logic of the main system components using simplified procedural pseudocode (PDL). Each component description includes its purpose and core algorithmic behavior. Only essential local data is shown where needed.

5.7.1 Detection Engine Component

Purpose: Processes video frames and detects accidents using the AI model.

Local Data:

THRESHOLD = minimum confidence required.

PDL:

```
function detectAccident(frame):
    processed = preprocess(frame)
    result = model.infer(processed)

    if result.confidence >= THRESHOLD:
        return result
    else:
        return null
```

5.7.2 Event Orchestrator Component

Purpose: Creates and updates accident events.

PDL:

```
function createEvent(detection):
    event = new Event()
    event.status = "detected"
    save(event)
    return event.id

function updateEventStatus(eventId, status):
    event = load(eventId)
    event.status = status
    save(event)
```

5.7.3 Assignment Engine Component

Purpose: Selects the nearest available emergency units.

PDL:

```
function assignUnits(eventLocation):
    units = getAvailableUnits()
    for each unit in units:
        unit.distance = computeDistance(unit.location, eventLocation)
    return sortByDistance(units)
```

5.7.4 Notification Component

Purpose: Sends alerts for new or updated events.

PDL:

```
function sendAlert(eventId):  
    subscribers = getActiveUsers()  
    for each user in subscribers:  
        pushMessage(user, eventId)
```

5.7.5 Repository Layer (Data Access Component)

Purpose: Store and retrieve event data.

PDL:

```
function saveEvent(event):  
    SQL INSERT or UPDATE event
```

```
function getEvent(eventId):  
    return SQL SELECT event WHERE id = eventId
```

5.8 Detailed System Design

The Detailed System Design document provides a comprehensive overview of each component's role, the constraints that govern its operation, and its interaction with other components for the common goal of providing real-time accident detection and response, thereby conforming to the system's requirements and its architecture. Besides, it also lays out the application resources, interactions, and dataflows between services, processing logic at a high-level, and the interface/exports that each subsystem exposes for a maintainable, testable integration throughout the pipeline. Moreover, it contains detailed references for the design of subsystems and employs sequence and UML diagrams to make the distribution of responsibilities, state transitions, and messages across components clear, thus ensuring that developers and reviewers have a common, traceable understanding of behavior and dependencies.

5.8.1 Classification, Definition and Responsibilities

This section summarizes each component's type, purpose, and key duties to ensure clear ownership and traceability from requirements to implementation across the accident detection and response pipeline.

Component	Classification	Definition and Responsibility
Video Acquisition & Detection	Subsystem	It is receiving the live video feeds, preparing the images, running the AI models to identify potential accidents, and subsequently transmitting the detection events in a universally accepted format for further processing.
Emergency Event Management	Subsystem	It is performing the functions of the major accident event lifecycle controller by handling the succession of creating, checking, assigning, updating, and changing the event state.
Emergency Units Management	Subsystem	It continues to manage the lists of responders and their live availability, performs geospatial proximity queries, and helps in assigning the nearest suitable units to incidents.
Notification & Alerting	Subsystem	Through push channels and external connectors, it provides real-time updates to users which implies that alerts and status changes have been delivered in a trustworthy and auditable way.
UI Dashboard	Subsystem	It incorporates a secure web interface for monitoring incidents, displaying them on a map and in a list, and performing CRUD operations for system entities at the administrative level.
Data Management & Storage	Subsystem	It assures adherence to the ACID model for the core records and media artifacts by offering repository interfaces and indexing for fast reading and writing across services.

Cross-Cutting Services	Module set	Besides, it implements authentication, authorization, logging, and performance targets corresponding to the behavior off the subsystem, system-level requirements, and documentation traceability.
-------------------------------	------------	--

Table 31: Component Definitions and Responsibilities.

5.8.2 Composition

This section defines the constraints for each component, including its limitation, pre-conditions, and post-conditions, and summarizes how subcomponents compose the overall behavior.

Component	Limitation	Pre-condition	Post-condition
Video Acquisition & Detection	Requires stable network, 15–30 FPS and 720p+ streams, and sufficient GPU/CPU for inference.	Valid stream URL configured; model loaded; compute resources available.	Standardized detection event produced with timestamp, camera ID, confidence, and media reference.
Emergency Event Management	Must enforce valid lifecycle transitions and idempotent updates; relies on transactional storage.	Detection payload received; database connection available.	Event created/updated atomically; notifications queued; state persisted.
Emergency Units Management	Accurate geolocation and up-to-date availability required; depends on geospatial indexing.	Units registered with coordinates; spatial index initialized.	Ranked nearby available units returned; unit status updated if assigned.
Notification & Alerting	Real-time push required; resilience to channel outages; idempotent deliveries.	Active client session or configured external channel; valid event update.	Alerts delivered to recipients; delivery recorded in audit log.
UI Dashboard	Depends on backend APIs and push channels; require authenticated sessions.	User authenticated and authorized; services reachable.	Live data rendered and actions executed; UI synchronized with backend.
Data Management & Storage	ACID, 3NF schema, indexing, and retention policies must be enforced within capacity limits.	Migrations applied; storage healthy; access credentials valid.	Records persisted consistently; media stored; queries performant on indexes.

Table 32: Component Constraints and Conditions.

5.8.3 Uses/Interactions

Uses/Interactions describe each component's collaborations with other components, specifying which entities use it, what it uses (including side-effects on shared state like databases), and interaction methods such as message queues, REST APIs, or publish-subscribe topics, ensuring loose coupling across the modular pipeline.

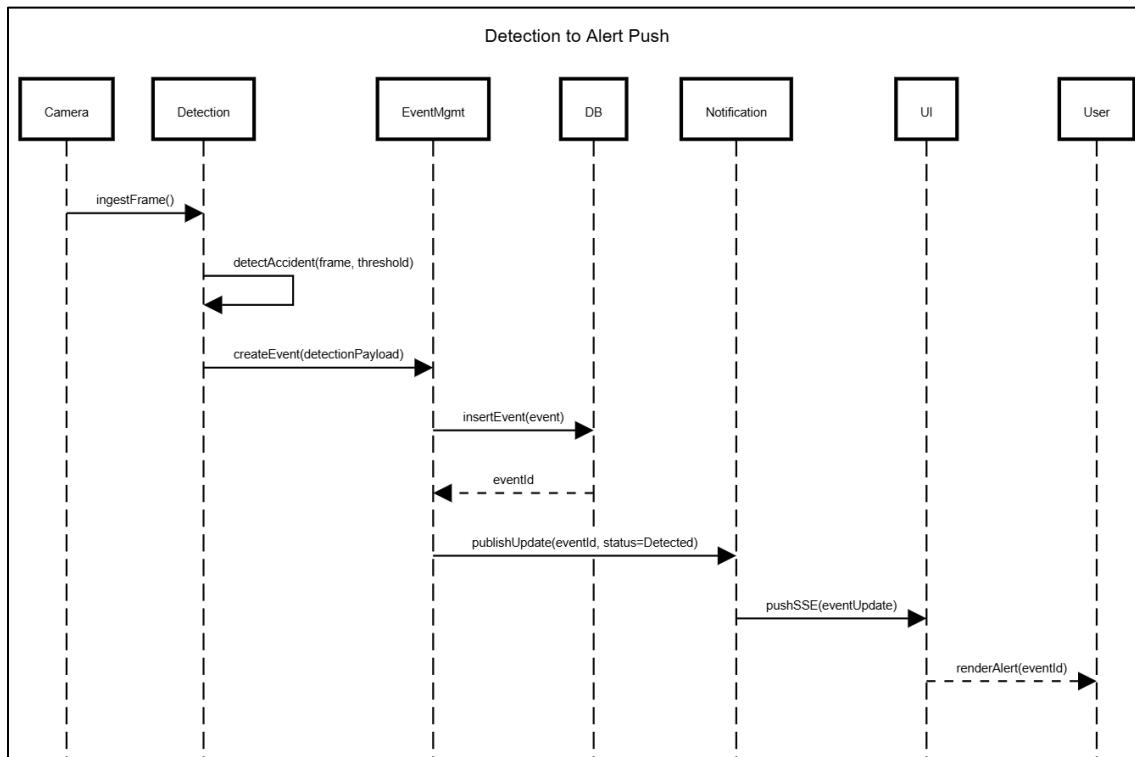


Figure 20: Sequence of Interactions Across Subsystems.

5.8.4 Resources

This section catalogs all external resources (hardware, software, network) required, managed, or impacted by each subsystem, including concurrency risks and mitigation strategies aligned with real-time constraints and shared state access.

Component	Required Resources	Managed Resources	Concurrency Risks	Mitigation
Video Acquisition & Detection	GPU/CPU, RTSP network streams (15-30 FPS), OpenCV/YOLO libraries	Frame buffers, keyframe storage	Buffer overflow on high FPS, multiple stream workers	Bounded queues with backpressure, single-writer per stream, exponential reconnect backoff SDS-Draft.docx
Emergency Event Management	PostgreSQL DB connection pool, message queue	Event records, audit logs	Concurrent event updates, unit assignment races	Row-level locking, optimistic concurrency (versioning), outbox pattern for notifications SDS-Draft.docx
Emergency Units Management	Geospatial indexes (PostGIS), Redis cache (optional)	Unit availability status	Multiple assignment queries updating same unit	Atomic CAS operations, availability leasing with TTL SDS-Draft.docx
Notification & Alerting	WebSocket/SSE connections,	Notification audit records	Thundering herd on mass notifications	Debounced delivery, idempotent message IDs, topic partitioning SDS-Draft.docx

	SMTP/SMS gateways			
UI Dashboard	Browser session storage, HTTPS connections	N/A (stateless client)	Stale UI state during concurrent updates	Server-driven SSE pushes, cache-busting headers SDS-Draft.docx
Data Management & Storage	PostgreSQL server, S3-compatible object storage	All persistent data (ACID)	Deadlock on complex joins, write contention	Consistent lock ordering, short transactions (<100ms), connection pooling SDS-Draft.docx

Table 33: Resource Allocation and Concurrency Management.

5.8.5 Processing

This section describes how each subsystem performs its core responsibilities, specifying inputs, outputs, processing algorithms/steps, and constraints to ensure traceability from SRS requirements to implementation logic for the real-time accident detection pipeline.

5.8.5.1 Video Acquisition & Detection

Table 32 presents the description, inputs, outputs, and constraints of “Video Acquisition & Detection” component.

Attribute	Description
Inputs	RTSP/HLS stream URL, expected frame rate (15-30 FPS)
Outputs	Detection events containing {cameroid, timestamp, confidence, bounding box, keyframe reference}
Purpose	Continuously ingest video streams, preprocess frames, run AI inference to detect accidents in near real-time
Processing Steps	<ol style="list-style-type: none"> 1. Connect to video stream via RTSP connector. 2. Extract frames into bounded buffer. 3. Preprocess frames (e.g., resizing, normalization). 4. Run object detection model (YOLOv8), apply non-maximum suppression. 5. Filter detections by confidence threshold. 6. Capture keyframe when valid detection occurs. 7. Format and send detection payload to Event Management.
Constraints	<ul style="list-style-type: none"> Requires GPU/CPU availability for inference. Must maintain end-to-end latency under 2 seconds. Handle stream interruptions with reconnection and backoff logic.

Table 34: Process Specification of Video Acquisition & Detection.

5.8.5.2 Emergency Event Management

Table 33 presents the description, inputs, outputs, and constraints of “Emergency Event Management” component.

Attribute	Description
Inputs	Detection payloads from Video Acquisition, unit assignment queries
Outputs	Event records including updated event status and assigned units; triggers notifications
Purpose	Manage accident event lifecycle: creation, deduplication, verification, assignment, updates, and persistence
Processing Steps	<ol style="list-style-type: none"> 1. Search existing events overlapping in time and space. 2. Create new or merge detections into existing events. 3. Optionally await manual verification. 4. Query available emergency units near the event. 5. Assign units and update event status. 6. Persist updates atomically. 7. Publish event updates for notification.
Constraints	<p>Must ensure transactional integrity with optimistic locking.</p> <p>Allow only valid state transitions.</p> <p>Support idempotency to avoid duplicates.</p>

Table 35: Process Specification of Emergency Event Management.

5.8.5.3 Emergency Units Management

Table 34 presents the description, inputs, outputs, and constraints of “Emergency Units Management” component.

Attribute	Description
Inputs	Event location (latitude/longitude), unit type filters
Outputs	Ranked list of available emergency units with distances
Purpose	Maintain emergency responder directories, availability status, and provide proximity queries for assignments
Processing Steps	<ol style="list-style-type: none"> 1. Query geospatial index to find units within radius. 2. Filter units by type and availability. 3. Sort by geographic distance. 4. Return top candidates. 5. Update unit availability atomically on assignment.
Constraints	<p>Accurate GPS data required.</p> <p>Geospatial index must be up to date.</p> <p>Use atomic operations to prevent race conditions on availability.</p>

Table 36: Process Specification of Emergency Units Management.

5.8.5.4 Notification & Alerting

Table 35 presents the description, inputs, outputs, and constraints of “Notification & Alerting” component.

Attribute	Description
Inputs	Event update messages {eventId, status, summary}
Outputs	Delivered notifications via WebSocket/SSE and external channels; audit logs
Purpose	Route event updates reliably to subscribed clients and external notification channels
Processing Steps	<ol style="list-style-type: none"> 1. Determine recipients based on roles and locations. 2. Enqueue messages for delivery. 3. Send notifications via WebSocket/SSE. 4. Retry on failure with backoff. 5. Log delivery results for auditing.
Constraints	<ul style="list-style-type: none"> Ensure idempotent delivery for consistency. Handle outages gracefully. Maintain low latency (<1 second).

Table 37: Process Specification of Notification & Alerting.

5.8.5.5 UI Dashboard

Table 36 presents the description, inputs, outputs, and constraints of “UI Dashboard” component.

Attribute	Description
Inputs	API responses for incidents and units; server-sent events (SSE) for real-time updates
Outputs	Rendered UI views including maps, incident lists, and admin management pages
Purpose	Provide an interactive, responsive interface for monitoring incidents and managing system data
Processing Steps	<ol style="list-style-type: none"> 1. Authenticate users and establish SSE connection. 2. Fetch initial state via REST APIs. 3. Render map and lists. 4. Handle user actions with optimistic updates. 5. Sync UI state based on server events.

Constraints	Secure authentication and role-based access. Minimize UI latency and handle conflicting updates smoothly.
--------------------	--

Table 38: Process Specification of UI Dashboard.

5.8.5.6 Data Management & Storage

Table 37 presents the description, inputs, outputs, and constraints of “Data Management & Storage” component.

Attribute	Description
Inputs	CRUD requests from all subsystems; media files (keyframes, clips)
Outputs	Confirmation of persisted data; query results
Purpose	Provide reliable ACID-compliant storage and efficient query capabilities for all system data
Processing Steps	<ol style="list-style-type: none"> 1. Validate data schema and constraints. 2. Perform indexed query or transaction. 3. Upload media to object store. 4. Log all writes in audit logs. 5. Return operation results or error messages.
Constraints	Maintain ACID compliance. Enforce data normalization. Ensure sub-100ms transaction latencies.

Table 39: Process Specification of Data Management & Storage.

5.8.6 Interface/Exports

This section summarizes the main services and interfaces exposed by each subsystem so that developers can understand how to use them and verify that the system functions as expected. It focuses on the purpose and role of each interface rather than low-level implementation details, and it should be read together with the SRS external interface requirements and the detailed subsystem design diagrams.

Subsystem	Interfaces / Services	Purpose
Video Acquisition & Detection	Internal endpoint for detection results (e.g., POST /events/detections), internal message topic for detection events	Provides a standardized accident detection payload (camera identifier, timestamp, confidence, region of interest, and media reference) to the Emergency Event Management subsystem after processing live camera streams.
Emergency Event Management	REST APIs for creating and updating events (e.g., /events, /events/{id}, /events/{id}/status,	Acts as the main access point for creating accident events, updating their lifecycle status, assigning units, and broadcasting event

	/events/{id}/assign), internal event-update topic	changes to other subsystems such as Notification and UI Dashboard.
Emergency Units Management	REST APIs for managing and querying units (e.g., /units, /units/{id}, /units/nearby)	Exposes CRUD operations for hospitals, police, and fire units, and provides a proximity search service used by the Event Management subsystem when selecting suitable responders for an incident.
Notification & Alerting	WebSocket or Server-Sent Events endpoint for real-time updates (e.g., /ws/updates), internal notification trigger endpoint (e.g., POST /notify/{eventId})	Delivers real-time alerts and status changes to connected dashboards and, when configured, to external channels such as SMS or email, while recording notification attempts in the audit log.
UI Dashboard	Web client that consumes backend REST APIs and subscribes to real-time updates	Provides administrator and responder user interfaces for all core functions, including camera management, user management, viewing and verifying incidents, and monitoring assigned units and alerts.
Data Management & Storage	Database schema and repository layer accessed via ORM/DAO classes	Offers persistent storage for users, cameras, alerts, units, notification logs, and audit entries, serving all subsystems through well-defined data access methods rather than direct SQL in business logic.

Table 40: System Interface Specifications.

5.8.7 Detailed Subsystem Design

This section presents the detailed design of each major subsystem in the Real-Time Accident Detection and Emergency Alert System, including their internal structure, control flow, and data usage. It refines the high-level architecture by showing how the subsystems are decomposed into subcomponents and how these subcomponents collaborate to implement the functional requirements defined in the SRS.

For each subsystem (Video Acquisition & Detection, Emergency Event Management, Emergency Units Management, Notification & Alerting, UI Dashboard, and Data Management & Storage), the detailed design is documented using a combination of class/Component diagrams, sequence diagrams, and data models that illustrate:

- Internal subcomponents and their responsibilities
- Main control and data flows inside the subsystem
- Key algorithms or processing steps for core functions (e.g., detection, event lifecycle, unit selection, notification routing)

Where appropriate, these details are maintained directly with the source code (e.g., API definitions, repository classes, and model implementations). The SDS therefore references those artifacts and the

diagrams in the System Architecture and Data Design sections instead of duplicating low-level declarations, ensuring the design remains consistent and maintainable as the implementation evolves.

5.9 Other Design Features

This section captures additional design aspects that are not fully covered in the previous sections but still influence how the system behaves, evolves, or integrates into its environment. In this project, the Real-Time Accident Detection and Emergency Alert System adopts several cross-cutting design choices, including the use of an event-driven, modular architecture, separation of AI inference from user-facing components, and centralized auditing of all critical actions for accountability and troubleshooting.

The design also emphasizes extensibility by allowing future integration of new detection models, external notification channels, and additional emergency agencies without major changes to existing subsystems. Configuration is externalized where possible (e.g., camera endpoints, model thresholds, and notification rules), and non-functional concerns such as logging, authentication, and role-based access control are implemented as shared services to reduce duplication and keep business logic focused on accident detection and response workflows.

5.10 Requirements Traceability Matrix

This section links each major functional requirement defined in the SRS to the subsystems and design elements that implement it in the SDS. The full, detailed matrix is maintained in the “Requirements Traceability Matrix” subsection of the SDS and references the SRS Section 4.3 for requirement definitions and SDS Section 5.8 for detailed component design.

Associated ID in SRS	Technical Assumption	Functional Requirement (summary)	User	Associated ID in SDS	System Component
SRS (4.2.2 Product functions – Real-Time Video Monitoring)	Stable RTSP/HLS camera streams; GPU/CPU able to process 15–30 FPS at 720p+	System shall continuously receive and monitor live video streams from surveillance cameras.	Admin, Responder	SDS (5.5.3 Subsystem Architecture – Video Acquisition & Detection; 5.8.5.1 Process Specification of Video Acquisition Detection)	Video Acquisition & Detection Subsystem.
SRS (4.2.2 – Accident Detection Using Deep Learning)	Trained YOLOv8/CNN models; annotated datasets available.	System shall automatically detect car accidents in the incoming video frames using deep learning.	Admin, Responder	SDS (5.7.1 Detection Engine Component; 5.8.5.1)	Detection Engine Component within Video Acquisition & Detection.
SRS (4.2.2 – Accident Classification and Severity Assessment)	Events stored with time and location; severity rules/ML model defined.	System shall classify each confirmed accident by severity (e.g., minor, moderate, major).	Admin, Responder	SDS (5.7.2 Event Orchestrator; 5.8.5.2 Emergency Event Management; Alerts.severity in 5.6.2 Data Dictionary)	Emergency Event Management / Event Orchestrator.
SRS (4.2.2 – Automated Alert	Network connectivity to dashboard and external channels.	System shall generate and send alerts containing time,	Admin, Responder	SDS (5.7.4 Notification Component; 5.8.5.4; System Interface	Notification & Alerting Subsystem.

and Notification System)		location, severity, and camera reference.		Specifications in 5.4.5 & 5.8.6)	
SRS (4.2.2 – System Management and Monitoring Dashboard; UI Tables 21–23)	Role-based web access via HTTPS.	System shall provide an admin dashboard for managing cameras, users, and viewing system status.	Admin	SDS (5.4.3 Screen Images, Tables 20–23; 5.4.4 Screen Objects and Actions; 5.8.5.5 UI Dashboard)	UI Dashboard – Admin views.
SRS (4.2.2 – Emergency Units Management)	Units registered with coordinates and availability.	System shall manage emergency units (CRUD) and expose them for assignment.	Admin	SDS (5.5.3 Emergency Units Management; 5.7.3 Assignment Engine; 5.8.5.3)	Emergency Units Management Subsystem.
SRS (4.2.2 – Nearest Units Selection)	Geospatial index and event location available.	System shall provide the nearest available units for a given incident location.	Admin, Responder	SDS (5.7.3 Assignment Engine; 5.8.5.3 Process Specification of Emergency Units Management)	Assignment Engine / Units Proximity Service.
SRS (4.3.1.1 – Login Interface, Table 20)	Auth service and user table configured.	System shall authenticate admins and responders using email and password.	Admin, Responder	SDS (5.4.3 Figures 1–2; 5.4.4 Screen Objects and Actions)	Login & Reset Password interfaces backed by Auth API.
SRS (4.3.1.1 – Responder Dashboard Interface, Table 24–26)	Active WebSocket/SSE connection.	System shall display an active incidents map and list for responders, with live updates.	Responder	SDS (5.4.3 Figures 7–9; 5.4.4; 5.8.5.5 UI Dashboard)	UI Dashboard – Responder views with real-time alert feed.
SRS (non-functional – audit / logging in SRS 2.2 & 3)	Central relational DB and audit schema deployed.	System shall log key user and system actions for auditing and troubleshooting.	Admin	SDS (5.6.2 Data Dictionary – AuditLog; 5.6.3 ERD; 5.8.4 Resources)	Data Management & Storage + logging in Event Management and Notification.

SRS (4.3.1.1 – Camera Management Interface, Table 22)	Camera records stored in DB with stream URLs; RTSP/HLS supported.	System shall allow admins to add, edit, delete, and view configured camera streams.	Admin	SDS (5.4.3 Figure 4; 5.4.4 Screen Objects and Actions – Camera Mgt.; 5.5.3 Video Acquisition Module)	Camera Management UI + Video Stream Ingestion interface.
SRS (4.3.1.1 – User Management Interface, Table 23)	User and agency tables exist; RBAC model defined.	System shall allow admins to manage responder agency users (create, update, delete).	Admin	SDS (5.4.3 Figure 5; 5.4.4 – User Mgt.; 5.5.3 UI Dashboard Module)	User Management UI + backend User Management API.
SRS (4.3.1.1 – System Audit Log Interface)	AuditLog table and logging middleware enabled.	System shall provide an audit log screen showing key system activities with filters and export.	Admin	SDS (5.4.3 Figure 6; 5.4.4 – Audit Log; 5.6.2 AuditLog entity)	System Audit Log UI + Data Management & Storage.
SRS (4.3.1.1 – Incident Reports & Details, Tables 24–26)	Incidents persisted with IDs, timestamps, and status.	System shall show responders a list of past and active incidents and detailed accident information.	Responder	SDS (5.4.3 Figures 8–9; 5.4.4 – Incident Reports, Accident Details; 5.8.5.5 UI Dashboard)	Incident Reports & Accident Details views backed by Event Management.
SRS (Overall description / non-functional – Authentication & Roles)	Secure HTTPS deployment, hashed passwords, roles Admin/Responder.	System shall restrict access via authenticated login and enforce role-based permissions.	Admin, Responder	SDS (5.3.2 General Constraints – Security & Privacy; 5.4.1 Overview of UI; 5.8.4 Resources)	Cross-cutting security (Auth, RBAC, Audit) integrated into all subsystems.
SRS (Overall description – Real-time Processing)	Hardware and network sufficient to target < 2 s end-to-end latency.	System shall process video streams and generate alerts in near real time (target < 2 seconds).	System-level	SDS (5.3.2 General Constraints – Performance & Latency; 5.8.5 Process tables 32–37)	Pipeline design across Video Acquisition, Event Mgmt, Notification, UI, Data.
SRS (Overall description – Reliability & Recovery)	Reconnect and retry strategies implemented; message delivery tracking available.	System shall handle stream drops and transient failures via reconnection and retry mechanisms.	System-level	SDS (5.3.2 General Constraints – Network & Storage; 5.8.4 Resources; 5.8.5 Processing for affected subsystems)	Stream Connector, Notification & Alerting, Repository Layer.

SRS (Overall description – Data Persistence)	Relational DB and object storage deployed and reachable.	System shall persist accident events, media, and related entities in a consistent, normalized schema.	System-level	SDS (5.6.1 Data Description; 5.6.2 Data Dictionary; 5.6.3 Database Description)	Data Management & Storage (ERD, tables Alerts, AlertPhotos, Users, Cameras, AuditLog).
---	--	---	--------------	---	--

Table 41: Requirements Traceability Matrix Linking SRS Functional Requirements to SDS System Components

Chapter 6: Conclusion

The last chapter of the document gives it a proper ending that is very clear and short. It goes on to mention the major advantages of project implementation based on the study's analysis to keep the stakeholders informed of its worth and thus, to direct them.

6.1 Introduction

The last chapter sums up the most important discoveries, the suggested solution, and the entire contributions of the emergency response optimization project in a short and easy way. It reiterates the use of deep learning and computer vision by the system for the real-time detection of car accidents that leads to the reduction of response delays and the provision of smarter traffic management for the safety of roads. The chapter not only points out the practical advantages but also the limitations that still exist, thus, it stresses the project's ability to influence public safety and at the same time it shows how the system designed can be the starting point for future improvements and being put into practice in the real world.

6.2 Findings And Contributions

This paper is a proposal at the moment, yet it has been suggested and contributions made already in the light of the first extensive literature review, requirements analysis, and system design that have been done until now. Such projections give evidence to the point that this project would enhance smart transportation and public safety enormously, especially in urban areas of Saudi Arabia with traffic and environmental conditions like those.

6.2.1 Future Findings

- **Detecting Accidents in Real-Time Is Feasible**

The project conveys that the YOLO-based object detection and CNN models can be very well applied to the live CCTV streams and would allow the detection of car accidents within 2 seconds or so. This would greatly diminish the dependency on human reports, which usually come with a delay. Such a situation is likely to lead to a quicker incident awareness and an accelerated activation of emergency workflows.

- **Enhanced Accuracy in Accident Detection**

With up-to-date and extensive accident video datasets during training and utilizing deep learning techniques, the system being proposed will likely attain a minimum of 85% in performance measures of precision, recall, and F1-score when it comes to differentiating accident events from the normal flow of traffic. The certainty of this degree of correctness would allow the emergency operators and traffic control centers to make decisions that are more reliable.

- **Enhanced Situational Awareness for Responders**

The unification of severity facets (such as the count of vehicles involved or the strength of impact) and structured alert messages is expected to equip the responders with more comprehensive, more practical

information even before their physical presence at the incident site. This could ease the selection of suitable resources, like ambulance units or fire services, and even secure the hospitals beforehand.

- **Support for Traffic Flow Management**

Real-time accident detection and localization are foreseen to assist traffic authorities in promptly rerouting cars around the incidents which will eventually result in less secondary collisions and reduced traffic. Gradually, the collected data might also be used for infrastructure planning and high-risk location hotspot analysis through informing.

- **Viability Under Practical Constraints**

The project design, which relies on public datasets, open-source tools, and available university or cloud resources, suggests that such a system can be prototyped and evaluated within realistic academic and budget constraints. This finding lends support to the idea that lively deployments can take place in resource-poor communities to develop emergency response.

6.2.2 Contributions

- **Integrated AI Framework for Emergency Response**

The venture gives a common platform that unifies object detection, event orchestration, notification services, and a web-based dashboard into a single pipeline solely for the enhancement of emergency responses. This comprehensive perspective not only focuses on separate models but also presents a feasible example for the implementation of real-world systems.

- **Formalized, IEEE-Based Documentation Set**

Through the creation of SPMP, SRS, and SDS papers based on IEEE standards, the project provides a reference that can be reused for the management, specification, and design of AI-driven safety systems. The documents will serve as templates for the future team in the context of smart city projects with computer vision and real-time alerting, which can be either fully adopted or slightly modified to fit their specific requirements.

- **Bridging Gaps Identified in Literature**

The literature review brought out the same limitations like lack of real-time CCTV validation, small or biased datasets, and poor processing of difficult conditions as recurring ones. The proposed system aims at closing these gaps at the design stage; One such instance is planning training on large multi-source accident datasets and focusing on latency, robustness, and deployment aspects in its metrics.

- **Integration with Saudi Vision 2030**

The idea behind the project is to contribute to Saudi Vision 2030 in a conceptual manner through the improvement of road safety, data-oriented emergency management, and intelligent urban infrastructure. The main theme of the project is the real-time monitoring and analytics with the future cities and better public safety services being the national aspirations it supports.

- **Foundation for Future Implementation and Research**

The proposal presents a systematic roadmap for future researchers and developers to do dataset preparation, model training, system integration, and testing. This framework can easily be expanded to add different types of data, edge deployment, or predictive accident anticipation, thus, providing several options for further research.

Even at the proposal stage, these expected findings and contributions show strong potential to advance how AI and computer vision are used to detect accidents, accelerate emergency response, and improve overall road safety in modern smart cities.

6.3 Limitations

The suggested system can have an impact both positively and negatively but still has several significant limitations which need to be recognized. These limitations affect models' reliability, deployment's strength, and the extent to which results can be interpreted beyond the academia or controlled environments. The suggested system can have an impact both positively and negatively but still has several significant limitations which need to be recognized. These limitations affect models' reliability, deployment's strength, and the extent to which results can be interpreted beyond the academia or controlled environments.

The main reason for the accident detection performance being dependent on the quality and availability of CCTV footage so much is that these factors have control over the whole process from low resolution, poor lighting, onto camera angles, and more. Besides that, publicly available datasets are the main source for current design and evaluation, which is a disadvantage since the system may not be well acquainted with the real-world traffic infrastructures and infrequent accident types despite them being allowed access to municipal networks.

Secondly, the system is built on the premise of uninterrupted network connectivity, ample bandwidth, and powerful compute resources (GPU servers or edge devices) that allow the real-time processing of 15–30 FPS streams at 720p or higher. This means that in reality, network problems, lack of proper hardware, or too many cameras can cause the system not to be able to meet the target of less than 2 seconds consistently, thus leading to delays, dropped frames, or throttled inference.

On the other hand, the training pipeline relies on precise, constant manual annotations and varied datasets that include different roads, illumination, and traffic patterns. Any bias, imbalance, or noise present in the data like lack of rural areas representation, night-time accidents, or severe weather will limit the model's ability to generalize especially when it's used in new cities or environments.

Finally, this project focuses on urban traffic and on the software design and prototyping phases, without yet addressing full-scale operational deployment, legal agreements for live CCTV access, or integration with existing emergency dispatch systems. As a result, real-world performance, long-term maintenance, cybersecurity concerns, and user adoption by emergency agencies remain open challenges that require future implementation and validation work beyond the scope of this proposal.

6.4 Lesson & Skills Learned

The proposal has been a way of showing that there are three major elements in the success of AI projects with a real-time and safety-critical focus: clear requirements, structured planning, and risk management. The group, in implementing an IEEE-compliant SPMP, SRS, and SDS, and using the Waterfall model to enforce a disciplined phase-by-phase manner of progress, was thus able to learn how to negotiate between high-performance aspirations and the practical boundaries of data, hardware, and time set for the project.

From a tech standpoint, the project increases expertise in literature-driven design, dataset planning, and deep learning tools for computer vision, including but not limited to the selection of YOLO-based architectures, modification of labeling processes, and establishing measures like precision, recall, F1-score, and latency for evaluation. Along with these, the team also became acquainted with the development practices of collaboration, through the utilization of GitHub, cloud GPU platforms, and constant reporting, which are very important for quality and traceability in the development of AI systems with multiple members.

6.5 Recommendations for future works

Future work should prioritize building and testing a full prototype on real CCTV streams with emergency agencies to measure true latency, reliability, and usability in live conditions. The system can then be extended with more advanced or multimodal models (for example, adding traffic sensors or drone feeds) to handle occlusions, rare accident types, and challenging environments.

It is also recommended to develop larger, more diverse datasets covering rural areas, highways, and different cities, and to investigate techniques like semi-supervised or federated learning to improve generalization while respecting privacy. Finally, exploring edge deployment, security hardening, and formal alignment with Saudi Vision 2030 smart city programs will support real-world adoption at scale.

References

- [1] Karim et al., "Visual Detection of Traffic Incidents through Automatic Scene Understanding," 2024.
- [2] Z. Chen et al., "Traffic Accident Data Generation Based on Improved Generative Adversarial Networks," Sensors, 2021.
- [3] M. S. Arefin et al., "Real-time rapid accident detection for optimizing road safety in Bangladesh," Heliyon, 2025.
- [4] M. M. Ahmed, A. Al-Fuqaha, and M. Al-Ajlan, "A real-time deep learning-based framework for traffic accident detection and alerting using CCTV feeds," Big Data and Cognitive Computing, vol. 7, no. 2, p. 22, 2023.
- [5] S. Ayesha, A. Aslam, M. H. Zaheer, and M. B. Khan, "CIRS: A Multi-Agent Machine Learning Framework for Real-Time Accident Detection and Emergency Response," Sensors, 2025.
- [6] S. M. Elhag, G. H. Shaheen, and F. H. Alahmadi, "Accident Response Time Enhancement Using Drones: A Case Study in Najm for Insurance Services," International Journal of Information Technology and Computer Science, vol. 15, no. 6, pp. 1–14, 2023.
- [7] Z. Xi, X. Liu, Y. Liu, Y. Cai, and Y. Zheng, "Integrating Generative Adversarial Networks and Convolutional Neural Networks for Enhanced Traffic Accidents Detection and Analysis," IEEE Transactions on Intelligent Transportation Systems, 2025.
- [8] M. Ragab, F. Kateb, M. W. Al-Rabia et al., "A Machine Learning Approach for Monitoring and Classifying Healthcare Data, A Case of Emergency Department of KSA Hospitals," International Journal of Environmental Research and Public Health, vol. 20, no. 4794, 2023.
- [9] H. Ghahremannezhad, H. Shi, and C. Liu, "Real-Time Accident Detection in Traffic Surveillance Using Deep Learning," Conference Paper, 2022.
- [10] Y. Zhang and Y. Sung, "Traffic Accident Detection Method Using Trajectory Tracking and Influence Maps," Mathematics, vol. 11, no. 7, p. 1743, Apr. 2023.
- [11] D. T. Mane et al., "Real-time vehicle accident recognition from traffic video surveillance using YOLOv8 and OpenCV," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 11, no. 5s, pp. 250–258, 2023.
- [12] Y. Zhang and Y. Sung, "Traffic accident detection using background subtraction and CNN encoder–transformer decoder in video frames," Mathematics, vol. 11, no. 13, p. 2884, 2023.
- [13] J. A. Ruby Florence and G. Kirubasri, "Accident detection system using deep learning," in Proc. Int. Conf. on Computational Intelligence in Data Science, 2022, pp. 301–310.
- [14] K. Pawar and V. Attar, "Deep learning based detection and localization of road accidents from traffic surveillance videos," ICT Express, vol. 8, no. 3, pp. 379–387, 2022.
- [15] Y. Xu et al., "TAD: A large-scale benchmark for traffic accidents detection from video surveillance," IEEE Access, 2024.
- [16] V. Adewopo et al., "Big Data and Deep Learning in Smart Cities: A Comprehensive Dataset for AI-Driven Traffic Accident Detection and Computer Vision Systems," arXiv preprint arXiv:2401.03587, 2024.
- [17] S. Bakheet and A. Al-Hamadi, "A deep neural framework for real-time vehicular accident detection based on motion temporal templates," Heliyon, vol. 8, no. 11, p. e11397, 2022.
- [18] H. Liao et al., "CRASH: Crash Recognition and Anticipation System Harnessing with Context-Aware and Temporal Focus Attentions," in Proc. 32nd ACM Int. Conf. on Multimedia, 2024.
- [19] N. Behboudi, S. Moosavi, and R. Ramnath, "Recent Advances in Traffic Accident Analysis and Prediction: A Comprehensive Review of Machine Learning Techniques," arXiv preprint arXiv:2406.13968, 2024.
- [20] R. Zhang, B. Wang, J. Zhang et al., "When language and vision meet road safety: leveraging multimodal large language models for video-based traffic accident analysis," arXiv preprint arXiv:2501.10604, 2025.
- [21] J. Fang, J. Qiao, J. Xue, and Z. Li, "Vision-Based Traffic Accident Detection and Anticipation: A Survey," arXiv preprint arXiv:2308.15985, 2023.
- [22] N. M. L. et al., "An Automated System for Accident Detection Using OpenCV," International Journal for Research in Applied Science & Engineering Technology, vol. 13, no. 4, 2025.
- [23] S. Akter, I. F. Shihab, and A. Sharma, "Large Language Models for Crash Detection in Video: A Survey of Methods, Datasets, and Challenges," arXiv preprint arXiv:2507.02074, 2025.
- [24] Omari Alaoui et al., "Advancing Emergency Vehicle Systems with Deep Learning: A Comprehensive Review of Computer Vision Techniques," Intelligent Systems with Applications, vol. 28, 200574, 2025.
- [25] X. Wu et al., "Big Data and Deep Learning in Smart Cities: A Comprehensive Dataset for AI-Driven Traffic Accident Detection," 2024.

Appendix

Response to comments

Tables below contain all suggestions and comments from the reviewers, and response to each comment:

1- Comment:	Comment was by
The title page should list both the main supervisor and the co-supervisor, and it must also include the names of the evaluators and the submission date.	Dr. Mohammad Aftab Alam Khan
Response	
The title page was updated to include “Dr. Mohammad Aftab Alam Khan (Supervisor)” and “Dr. Atta-ur-Rahman (Co-supervisor),” add the evaluators’ names “Dr. Ito Wasio” and “Dr. Rabab Alkhalifa,” and show the date “22 December, 2025.”	

2- Comment:	Comment was by
The abstract should be expanded to better explain why faster emergency response is needed, clearly describe the proposed AI-based solution, and briefly outline the methods and tools that will be used.	Dr. Mohammad Aftab Alam Khan
Response	
The abstract was revised to add background on the importance of emergency response, a clearer description of the accident-detection system, and a concise summary of the planned methodology and technologies.	

3- Comment:	Comment was by
The supervisor’s full name should be written consistently in the acknowledgement section.	Dr. Mohammad Aftab Alam Khan
Response	
The acknowledgement was edited so that the supervisor is referred to throughout as “Dr. Mohammad Aftab Alam Khan,” using the full name in every occurrence.	

4- Comment:	Comment was by
Language and wording in Section 1.2 need minor correction (for example, the phrasing around “uses”) to improve clarity and correctness.	Dr. Mohammad Aftab Alam Khan
Response	
Section 1.2 was re-read and rephrased to correct the indicated wording and to make the description of the system’s use clearer and grammatically correct.	

5- Comment:	Comment was by
Make sure the table of contents and lists of tables and figures accurately reflect the final structure, numbering, and page numbers of all chapters, tables, and figures.	Dr. Mohammad Aftab Alam Khan
Response	
After final edits, the table of contents, list of tables, and list of figures were regenerated and checked so that all titles, numbers, and page references match the completed report.	

6- Comment:	Comment was by
The report presents a practical, socially relevant, and technically feasible deep learning system for real-time accident detection using YOLO and CNN models, and has strong potential as a final year project if supported by solid validation and integration.	Dr. Rabab Alkhailfa
Response	
The overall concept and architecture of the proposed system were maintained, and additional details about the validation strategy, evaluation metrics, and integration plan were added in the methodology and design chapters to strengthen the project’s feasibility.	

6- Comment:	Comment was by
The title should be revised to “Harnessing Deep Learning to Optimize Emergency Response” for grammatical accuracy and consistency with the abstract.	Dr. Rabab Alkhailfa
Response	
The report title was updated to “Harnessing Deep Learning to Optimize Emergency Response” on the cover page, in the abstract, and everywhere it appears in the document.	

7- Comment:	Comment was by
Chapter 3 should not be separated from the rest of the report; the document needs to be a single continuous report following the normal chapter sequence for better flow and professionalism.	Dr. Rabab Alkhailfa
Response	
The SPMP material in Chapter 3 was merged into the same document as the other chapters, and the chapter numbering and table of contents were adjusted so the report appears as one continuous, professionally formatted file.	

8- Comment:	Comment was by
First-person expressions such as “we aim to” should be avoided in formal writing and replaced with an objective style like “the project aims to.”	Dr. Rabab Alkhailfa
Response	
All chapters were reviewed, and first-person phrases were systematically replaced with objective constructions (for example, “the study proposes,” “the system is designed to”) to maintain a formal academic tone.	

9- Comment:	Comment was by
All reference links must be active and formatted according to IEEE style.	Dr. Rabab Alkhailfa
Response	
The reference list was checked link by link; broken or incomplete URLs were corrected where possible, and entries were reformatted to follow IEEE citation style consistently.	

10- Comment:	Comment was by
The deliverable list reads like coursework milestones and should distinguish between academic deliverables (reports, presentations, documentation) and technical deliverables (trained models, datasets, prototypes, validation results)	Dr. Rabab Alkhailfa
Response	
The deliverables section was restructured into two groups: academic deliverables (reports, presentations, documentation) and technical deliverables (models, datasets, prototype system, validation outputs), each clearly labelled.	
11- Comment:	Comment was by
There is no clear traceability between objectives and deliverables; each main objective should be linked to an experiment or output that verifies it.	Dr. Rabab Alkhailfa
Response	
A brief objective–experiment–deliverable mapping was added, showing for each key objective which experiments, evaluations, or artefacts confirm it, improving traceability and readability.	
12- Comment:	Comment was by
Figure and table captions should be descriptive enough that readers can understand their meaning without reading the main text; Table 2 (Literature Review Summary) should be kept but given a more informative caption.	Dr. Rabab Alkhailfa
Response	
Captions for all figures and tables, including Table 2, were revised to include clearer descriptions of what is shown and why it is important, so that each item is understandable on its own.	

13- Comment:	Comment was by
Spacing, paragraph alignment, and general formatting need to be uniform across the document.	Dr. Rabab Alkhailifa
Response	
The report template was cleaned and reapplied, ensuring consistent paragraph spacing, justification, margins, and alignment throughout all chapters.	

14- Comment:	Comment was by
The visual layout (font sizes, spacing, numbering style) should be consistent across headings, body text, lists, figures, and tables.	Dr. Rabab Alkhailifa
Response	
A single style set was applied for headings, body text, captions, and lists, and numbering styles were standardized so that the document has a uniform visual presentation.	

15- Comment:	Comment was by
Short connecting sentences are needed between sections to make transitions smoother.	Dr. Rabab Alkhailifa
Response	
Linking sentences were inserted at the end or beginning of key sections to explain how each part leads into the next, improving the logical flow for the reader.	

16- Comment:	Comment was by
A brief summary at the end of Chapter 3 is needed to connect the management plan to the upcoming technical chapters and improve coherence.	Dr. Rabab Alkhailifa
Response	
A concluding subsection was added at the end of Chapter 3 that summarizes the management plan and explains how it supports the requirements, design, and implementation work discussed in later chapters.	