MINISTRY OF EDUCATION
IMAM ABDULRAHMAN BIN
FAISAL UNIVERSITY
COLLEGE OF COMPUTER SCIENCE
& INFORMATION TECHNOLOGY

وزارة التعليم
جامعة الإمام
عبدالرحمن بن فيصل
كلية علوم الحاسب
وتقنيةالمعلومات

جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

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 Radwan	2220002478	Member
4	Ahmad Abdulqader Alakhdar	2220002324	Member
5	Ali Ibrahim Asiri	2220000306	Member

Supervised by:

Dr. Mohammad Aftab Alam Khan

Committee Member Names:

……

Date:

……..

ACKNOWLEDGEMENT

We want to sincerely thank our supervisor, Dr. Mohammed Aftab, 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

The project "Harnessing Deep Learning to Optimize Emergency Response" is designed to help emergency teams react more quickly and effectively to car accidents. Using advanced AI and computer vision, the system watches live videos from traffic and security cameras to spot accidents as they happen. It then sends detailed alerts to emergency responders, including the exact location of the crash.

By using powerful AI models like YOLO and convolutional neural networks, the system identifies collisions in real time, cutting down response times and improving outcomes. Beyond saving lives, it also helps ease traffic jams around accident sites, working toward the goal of safer, smarter cities. By automating accident detection and reducing the chance of human error, this project aims to make public safety and emergency coordination more reliable and efficient.

Table of Contents

Table of Tables

Table of Figures

LIST OF ABBREVIATIONS

Table 1 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

Chapter 1: Introduction

This chapter kicks off with a brief introduction to the project, highlighting key details and exploring the scope and limitations of the problem at hand. 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 use 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 it 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 examine 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]

Table 2 Literature Review Summary

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.

				TAD Dataset				
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.

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 covering 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 detection of 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 during 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	Detailed functional and non-functional requirements	October 30, 2025	MS Word document (IEEE 830-1998)

Specification (SRS)			
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:

  - 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
  - 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.

o 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.

o 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.

o 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:

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

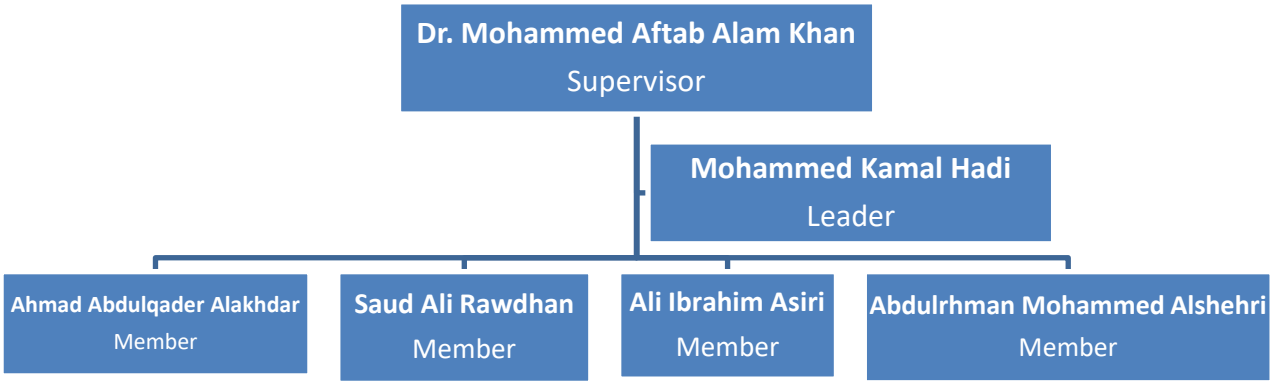

Figure 1: Internal structure

3.2.3 Roles and Responsibilities

The project manager shares responsibilities fairly among all the team members and he controls the performance of their duties assigned. 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 Alakhdar — 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 Human Resources	Estimated Duration
Phase 1: Planning & Requirement Analysis	All Team Members	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	• Leadership, • Project Scheduling, • Risk Management, • Team Coordination, • Communication
AI/ML Developer	• Python, • TensorFlow/PyTorch, • OpenCV, • Computer Vision Models (YOLO, CNNs), • Data Annotation, • Model Training & Evaluation
Backend Developer	• API Development (Flask/Django), • System Integration, • Real-time Data Handling
Data Specialist	• Data Sourcing & Cleaning, • Video Preprocessing, • Data Annotation Tools (CVAT), • Data Augmentation
QA & Testing Engineer	• 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	*Submit Bi-weekly Report #1*	-	-	*Sep 18, 2025*
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	*Submit Bi-weekly Report #2*	-	-	*Oct 2, 2025*

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	*Submit Midterm Report*	-	-	*Oct 16, 2025*
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	*Submit Bi-weekly Report #3 (SRS)*	-	-	*Oct 30, 2025*
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	*Submit Bi-weekly Report #4 (SDS)*	-	-	*Nov 13, 2025*
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	*Submit Bi-weekly Report #5*	-	-	*Nov 27, 2025*
6.3	Write First Semester Final Report	Nov 16, 2025	25 days	Dec 11, 2025
6.4	*Submit Final Report*	-	-	*Dec 11, 2025*
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	*Deliver Final Presentation*	-	-	*Dec 17-18, 2025*

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	• High-performance PCs with NVIDIA GPUs, • University Computer Labs, • Cloud-based GPU Instances (as needed).
Software	• Development: Python, VS Code, Jupyter Notebook • AI/ML Libraries: TensorFlow, PyTorch, OpenCV, Scikit-learn • Collaboration: Git, GitHub, Microsoft Teams • Documentation: Microsoft Office Suite
Informational	• 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 cost 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.

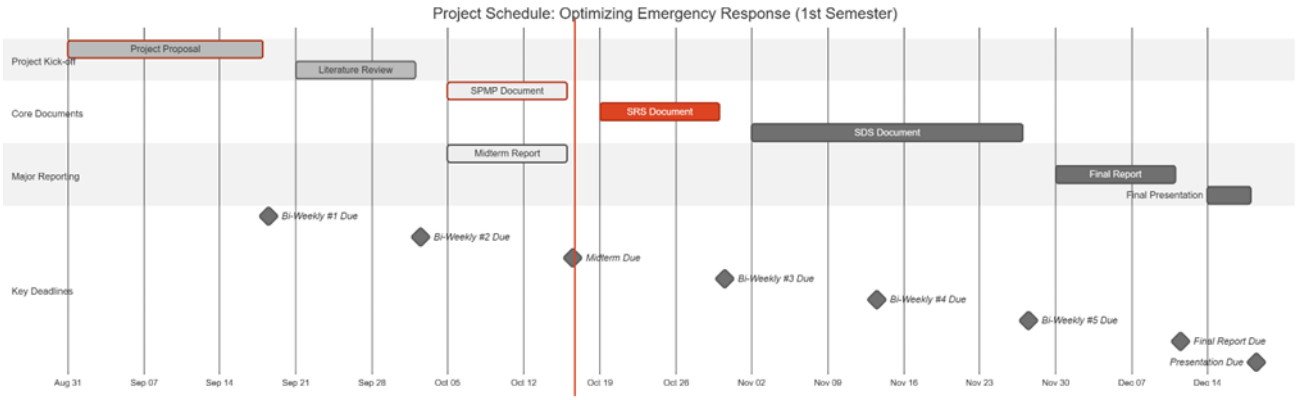

Figure 2: Gantt Chart: Project Schedule

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	Risk Description	Probability	Mitigation Strategy	Impact
R1	**Data Scarcity/Poor Quality:** Lack of sufficient or diverse accident footage for robust model training.	Moderate	Proactively aggregate data from multiple public sources; employ data augmentation techniques.	Quality, Schedule
R2	**Low Model Performance:** The trained model fails to achieve the desired accuracy for reliable, real-world detection.	Moderate	Experiment with various state-of-the-art model architectures and conduct extensive hyperparameter tuning.	Quality
R3	**Insufficient Computing Power:** Local hardware is inadequate for training the deep learning model efficiently.	High	Establish and test access to cloud GPU platforms (e.g., Google Colab Pro) as a primary or backup solution.	Budget, Schedule
R4	**Time-Consuming Annotation:** Manual labeling of video frames takes longer than allocated in the schedule.	Moderate	Utilize efficient annotation tools (e.g., CVAT) and distribute the workload evenly among team members.	Schedule

Table 14: 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 functions 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.

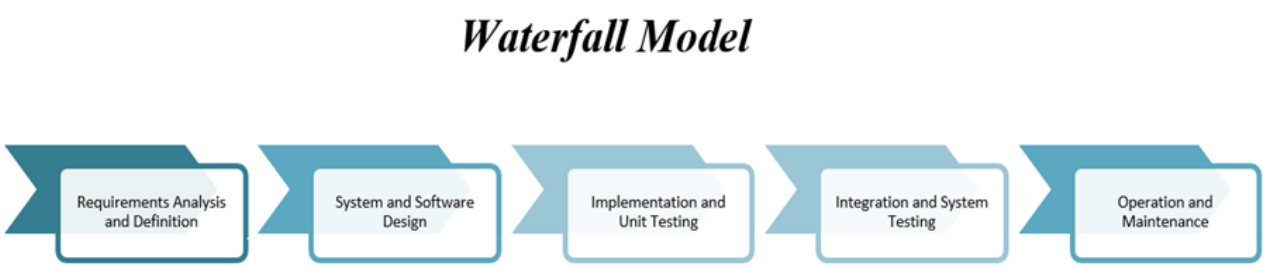

Figure 3: The Waterfall Model

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

Phase	Duration	Description	Main Activities	Deliverables / Outcomes
1. Requirements Analysis	**Nov 1 – Nov 20 2025**	Identify, analyze, and document all functional and non-functional requirements for the system.	• Gather requirements from literature and emergency response protocols.• Define system functions (accident detection, alert generation, and reporting).• Specify hardware &	• Software Requirements Specification (SRS).• Use-case and context diagrams.• Dataset description document.

			software constraints, datasets, and ethical considerations.	
2. System and Software Design	**Nov 21 – Dec 10 2025**	Translate the requirements into detailed architecture and technical design.	• 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).	• 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.	• 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.	• 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.	• Conduct unit testing for each module.• Perform integration and system testing.• Evaluate accuracy (Precision, Recall, F1-Score).• Simulate accident scenarios for response time testing.	• 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.	• Deploy system on cloud platform (Google Colab or AWS EC2).• Integrate real-time video feed for testing.• Prepare user manual and presentation materials.	• Operational prototype demonstration.• Deployment report.• User manual and demo video.
6. Maintenance and Enhancement	**Feb 6 – Feb 20 2026**	Conduct system review, bug fixes, and enhancements based on feedback.	• Resolve issues from testing and presentation reviews.• Optimize model parameters and update dataset.• Document lessons learned and future improvements.	• Updated system version.• Maintenance 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 in its goal the designer, tester, 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.

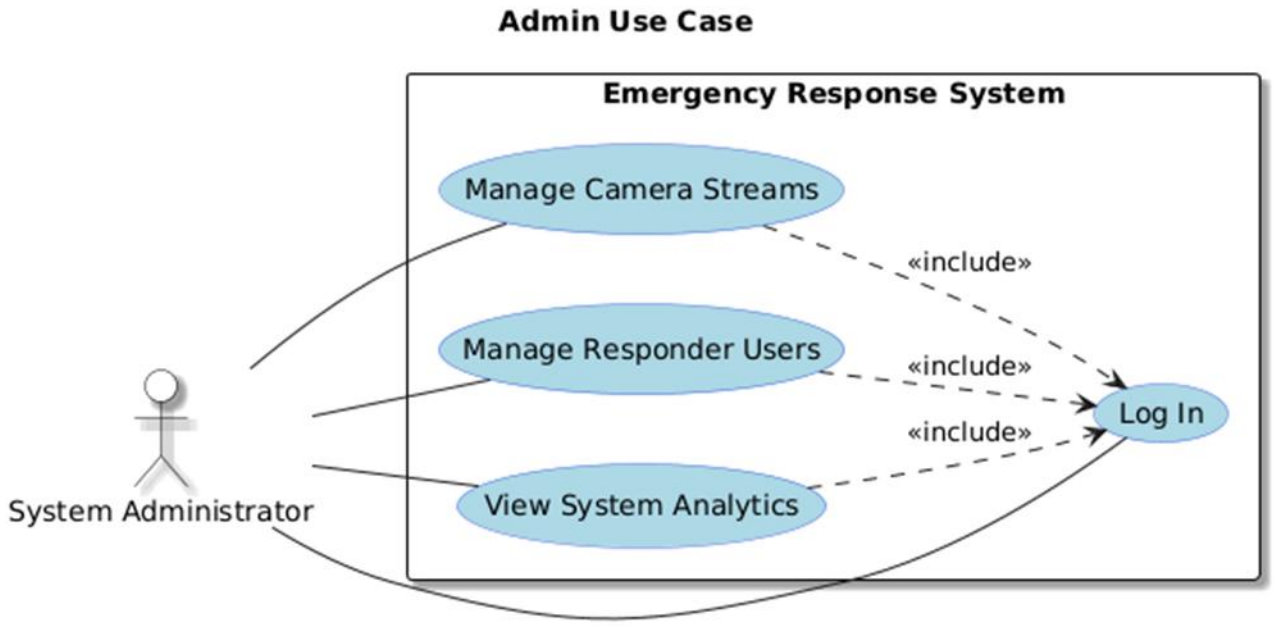

Figure 4: Admin Use Case

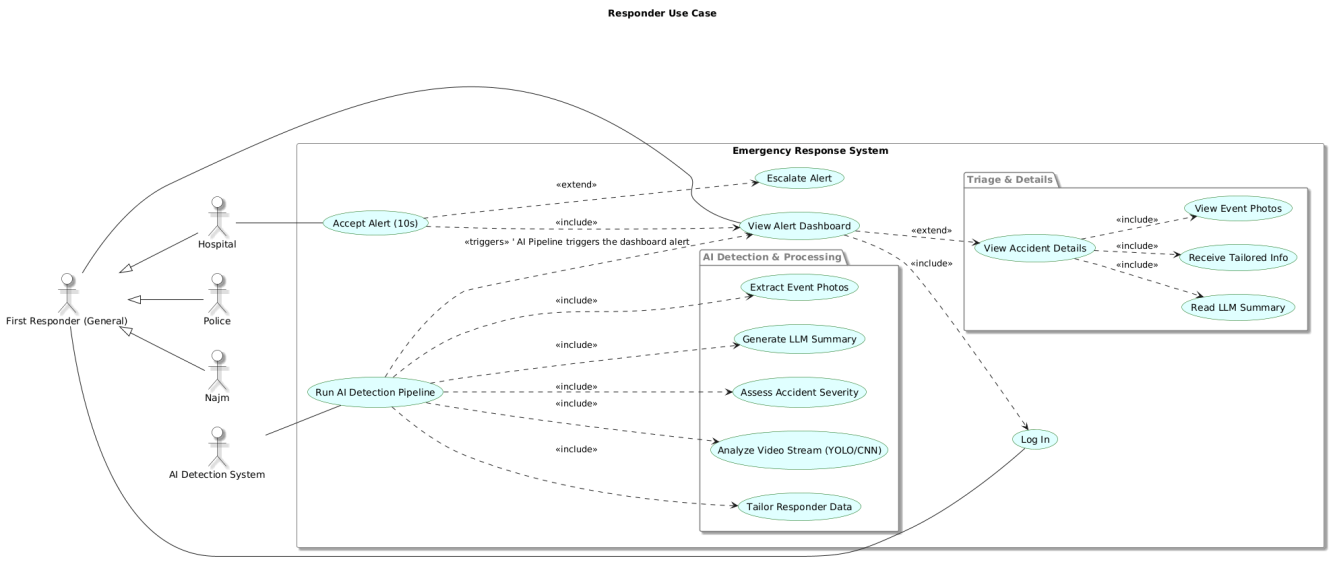

Figure 5: First Responders Use Case

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/Output	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

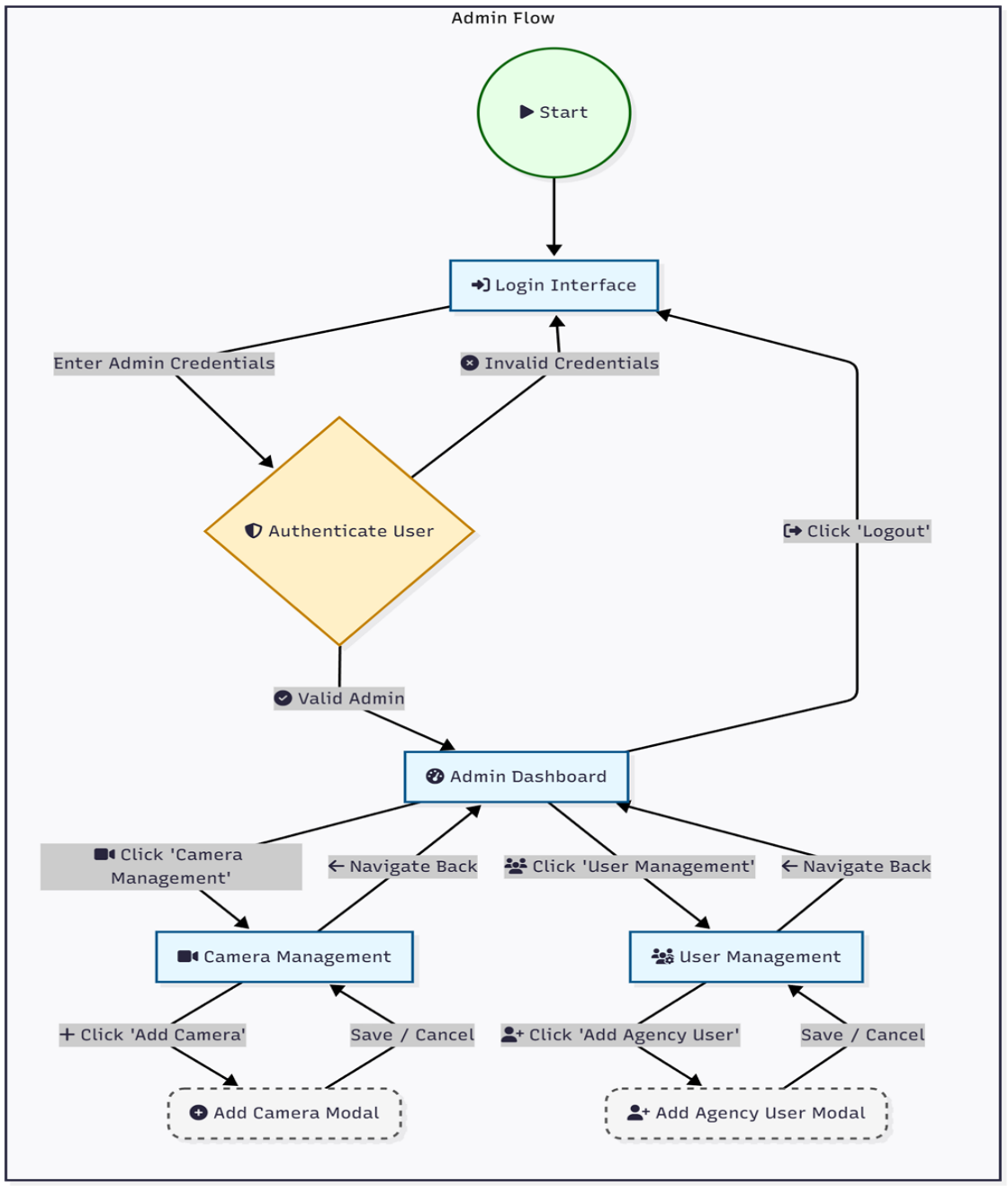

Figure 6: Admin workflow system

4.3.1.1.3 First Responder User Interfaces

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

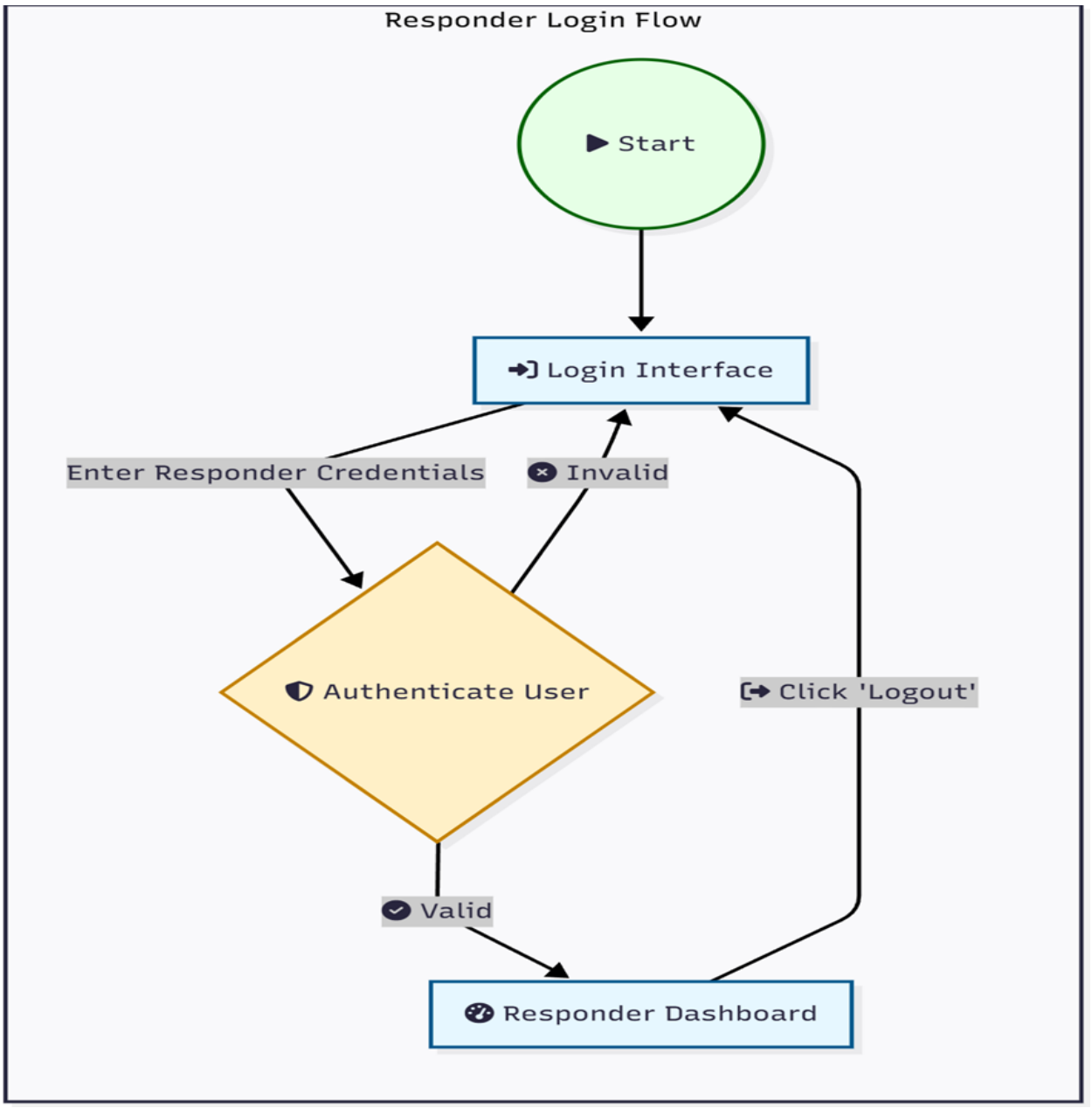

Figure 7: The Responders Login Flow

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/responeded, 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	**This section is tailored.** • **Hospital:** "Estimated Injuries: 3" • **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

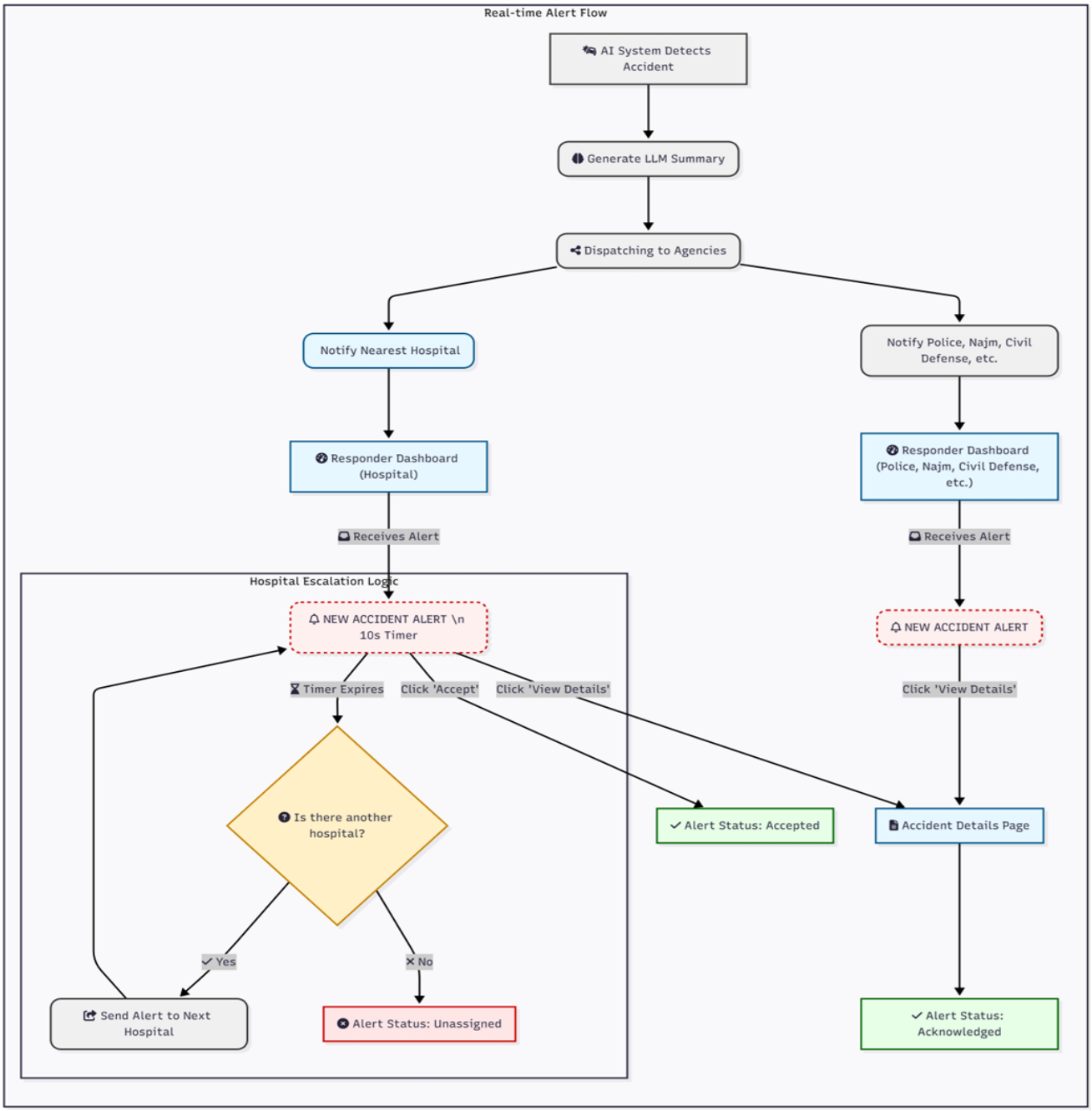

Figure 8: AI Alert System Flow

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 informatio.
- **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).

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.