Collision Detection System

UCS503 Software Engineering Project Report for Mid-Semester Evaluation

Submitted By:

102303265 Pranav Keswani
102303266 Sujal Mallick
102303270 Vansh Bhasin
102303437 Rohan Malhotra

BE Third Year, CSE
Group- 3C22

Submitted to: Dr. Sandeep

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department
TIET, Patiala

November, 2025

TABLE OF CONTENTS

Phase-1: Project Selection

1. Software Bid

Group Number: 3

Name	Roll No.	Project Experience	Programming Language used	Signature
Sujal Mallick	102303266	Finance with Gen AI	Python, HTML/CSS/JS	Sujal Mallick
Rohan Malhotra	102303437	Credit Card Fraud Detection System	Python, C++,	Rohan Malhotra
Pranav Keswani	102303265	Finance with Gen AI	Python, HTML/CSS/JS	Pranav Keswani
Vansh Bhasin	102303270	Credit Card Fraud Detection System	Python, C++	Vansh Bhasin

Programming Language / Environment Experience

1. Python + Google Colab -> for deep learning model training
2. Github
3. C++ -> for implementation of YOLO

Choices of Projects:

	Project Name	Unique Selling Point
First Choice	Online Placement Portal	A one-stop, automated placement management platform that ensures accurate, validated data exchange between students, TPOs, and companies—streamlining recruitment while eliminating errors and reducing manual effort.
Second Choice	ViolenceWatch – Student Conflict Detection System	An AI system that detects physical aggression or fights in real-time from CCTV footage using deep learning. The system notifies campus security with location and video clip evidence, enabling faster intervention.
Third Choice	FindIt – Intelligent Lost and Found Portal	A smart campus-wide portal that matches reported lost items with found items using image recognition and natural language processing (NLP). It reduces manual overhead and accelerates the process of retrieving lost belongings.
Fourth Choice	ThaparNav – Smart Campus Navigation App	An indoor-outdoor navigation app tailored specifically for Thapar University. It provides route guidance to lecture halls, labs, cafeterias, and admin offices, including wheelchair-accessible paths and eventbased routing.

2. Project Overview

Overview:
The *Collision Detection System (CDS)* is an AI-powered surveillance and accident detection platform designed to identify vehicle collisions in real time, generate automated alerts to emergency responders, and assist traffic authorities in making informed decisions. By integrating computer vision, object tracking, automated alert workflows, and a structured incident database, CDS reduces emergency response delays, improves situational awareness, and enhances road safety monitoring efficiency.

Key Features

1. Real-Time Collision Detection

- Continuous monitoring of live CCTV feeds.
- Collision detection using YOLO-based object detection models with bounding box intersection logic.
- Reactive highlighting of impacted regions to support quick human verification.

2. Automated Emergency Alert System

- Instant notification to emergency responders with:
  - GPS coordinates
  - Timestamp
  - Collision severity score
  - Pre- and post-incident video clip
- Multi-channel alert delivery (Dashboard, Email, SMS, API).

3. Traffic Authority Monitoring Dashboard

- Web-based interface displaying:
  - Live video streams
  - Bounding boxes on detected vehicles
  - Collision warnings with confidence levels
- Prioritizes clarity, low-latency display, and rapid decision-making.

4. Incident Logging and Analytics

- All incidents are stored with:
  - Time and location metadata
  - Severity classification
  - Linked evidence video
- Enables trend analysis and identification of frequent high-risk zones.

5. Model Management and Threshold Configuration

- Adjustable detection confidence threshold to balance sensitivity and false positives.
- Ability to upload new datasets and retrain models.
- Performance metrics (Precision, Recall, F1-Score) available for evaluation.

System Roles and Capabilities

Emergency Responders

- Receive alert notifications immediately.
- Access evidence video clips for situational assessment.
- Coordinate and dispatch medical or rescue response.

Traffic Authorities

- Monitor multiple live feeds simultaneously.
- Verify detected incidents visually.
- Initiate traffic rerouting or ground assistance.

System Administrators

- Configure detection thresholds and sensitivity values.
- Upload new training data and trigger model retraining.
- Monitor system performance and maintain operational reliability.

Traffic Safety Analysts

- Analyze recorded incident data.
- Identify high-risk geographical zones.
- Support development of road safety improvement strategies.

Technology Stack

Component	Technology
Object Detection Model	YOLOv4 / DenseNet-201
Video Processing	OpenCV
Backend Runtime	Python
Backend Framework	Flask
Frontend Interface	HTML, CSS, JavaScript, Bootstrap

Real-World Applications

- Smart City Traffic Surveillance Centers
- Highway and Expressway Patrol Monitoring
- Emergency Medical Response Systems
- Law Enforcement and Accident Analysis Reporting
- Transportation Safety Policy and Road Planning Departments
- Insurance and Claims Verification Support

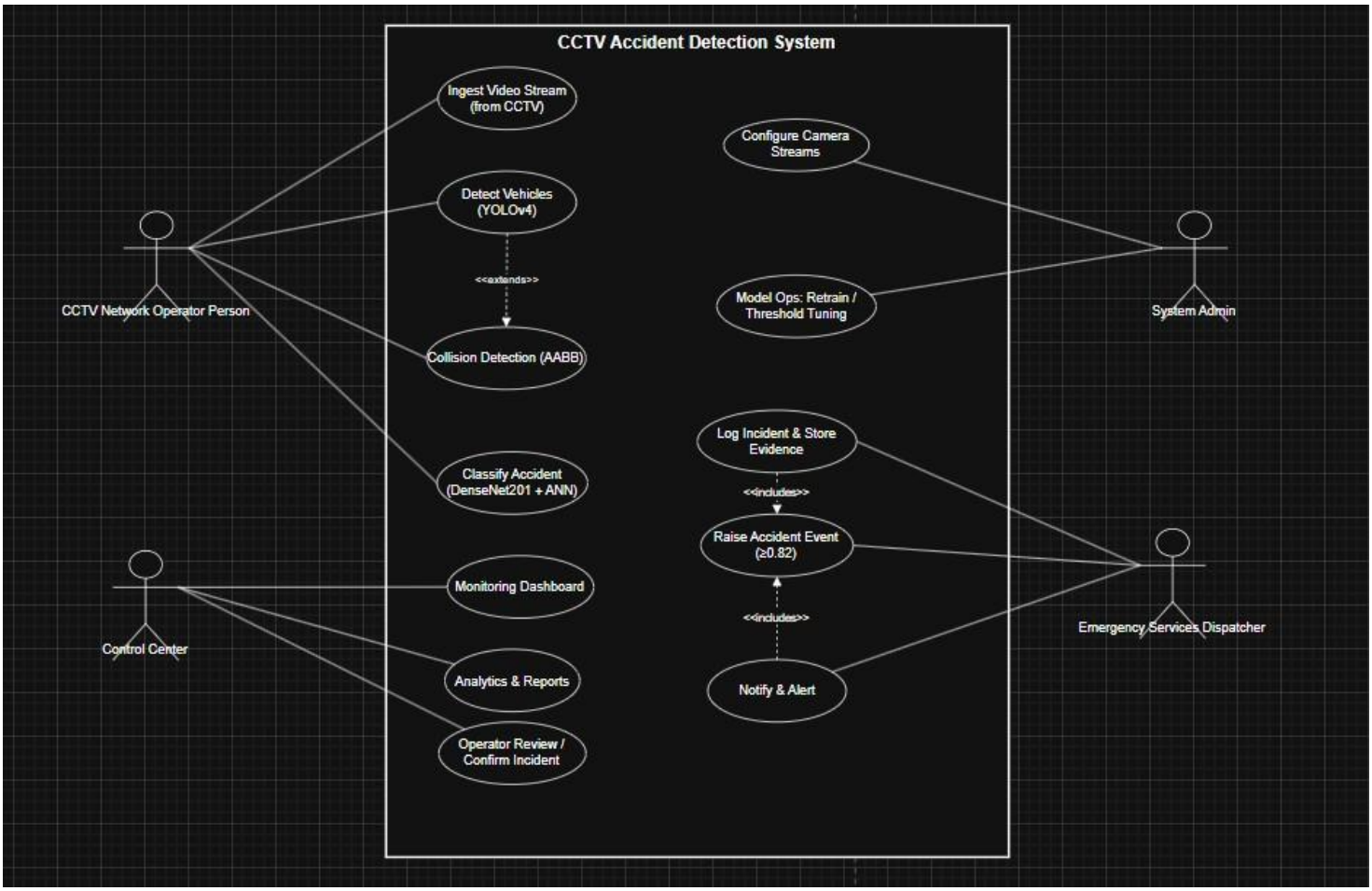
Future Scope

1. **Severity Classification Automation:** Enhance models to classify collision severity levels.
2. **Predictive Accident Prevention:** Analyze risky driving patterns for proactive alerts.
3. **Integrated Emergency Dispatch System:** Direct ambulance routing and ETA synchronization.
4. **IoT Sensor + Vehicle Telemetry Integration:** Combine video with sensor-based crash detection.
5. **Mobile Application for Responders:** Portable access to alerts and navigation routes.
6. **Edge-Based Processing:** Reduce latency by running detection models on local hardware nodes.

Phase - 2
Requirements Analysis & Specification

2.1 Use-Cases

2.1.1 Use Case Diagram



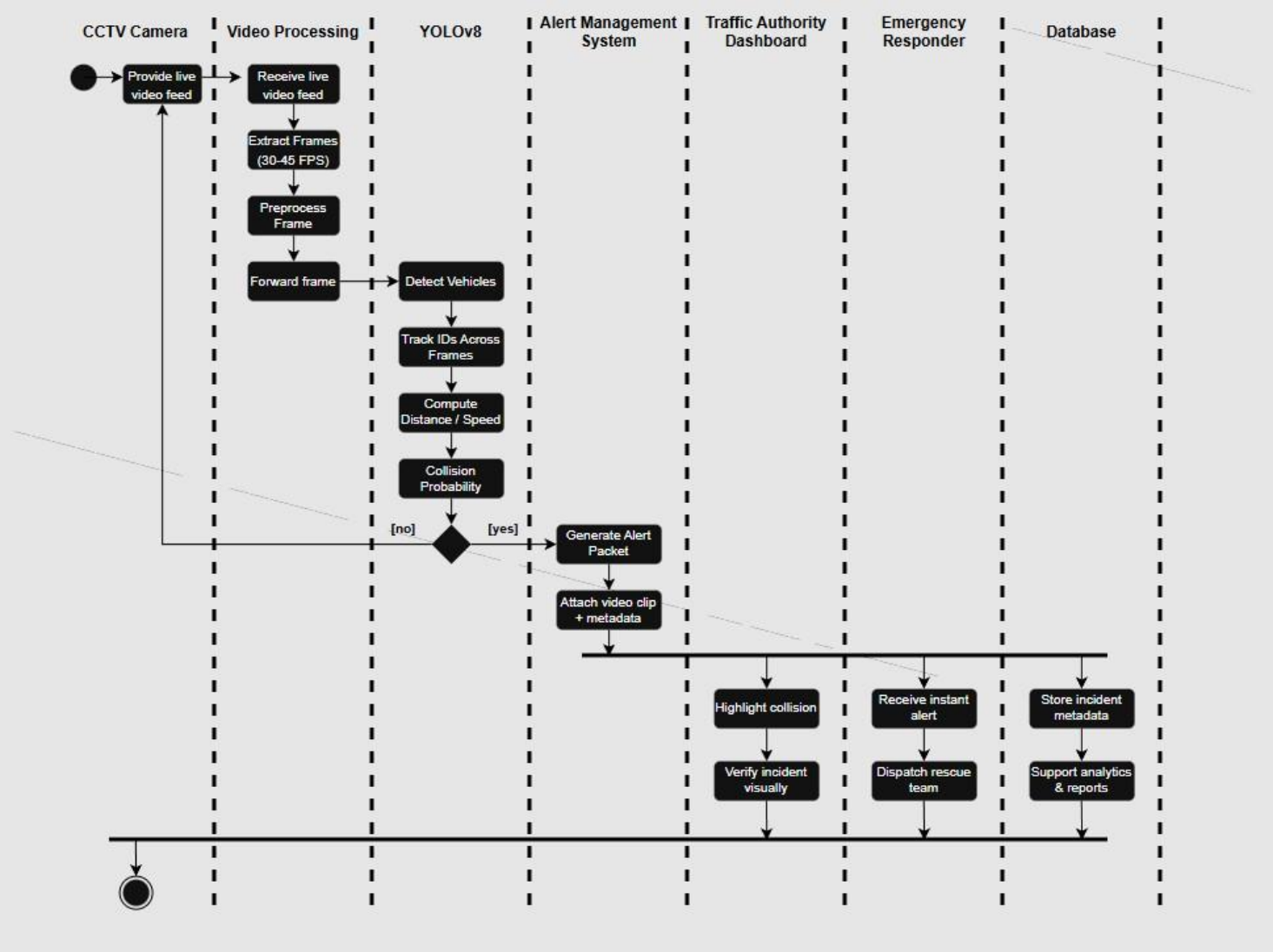
2.1.1 Use Case Templates

Use Case ID	CDS-UC-01
Use Case Title	Collision Detection and Accident Classification
Actors	**CCTV Network Operator Person** – Monitors camera feeds and manages incident confirmations. **Control Center** – Oversees system performance, analytics, and reporting. **System Admin** – Configures camera streams, retrains models, and tunes detection thresholds. **Emergency Services Dispatcher** – Receives automated alerts of confirmed accidents.
Description	This use case describes how the CCTV Accident Detection System identifies vehicle collisions using real-time video streams. It employs a **YOLOv4**-based object detection model to track vehicles and an **AABB (Axis-Aligned Bounding Box)** method to detect collisions. Once a potential collision is detected, a **DenseNet201 + ANN** classifier determines whether it is a genuine accident. Confirmed accidents are logged, evidence is stored, and notifications are sent to the control center and emergency dispatch.
Pre-Conditions	CCTV cameras are active and configured properly. System is connected to the video feed network. Detection models (YOLOv4, DenseNet201 + CNN) are trained and deployed. Network connectivity to control center and dispatch systems is available.
Post-Conditions	Accident event (if confirmed) is logged and stored with timestamp, location, and video evidence. Alerts are dispatched to emergency services and control center. Operator receives and reviews the alert in the monitoring dashboard.
Main Success Scenario (Normal Flow)	**Ingest Video Stream** – System continuously captures CCTV video input. **Detect Vehicles (YOLOv4)** – Vehicles are detected and tracked frame-by-frame. **Collision Detection (AABB)** – The system monitors bounding box intersections to identify collisions. **Classify Accident (DenseNet201 + ANN)** – Determines if the detected collision is a true accident. **Raise Accident Event (≥0.82 Confidence)** – If the classifier's confidence exceeds the threshold, an accident event is generated. **Log Incident & Store Evidence** – The system stores relevant video footage and metadata. **Notify & Alert** – Alerts are sent to the control center dashboard and emergency dispatch. **Operator Review / Confirm Incident** – Operator validates or rejects the detected accident in the monitoring dashboard.

Alternate Flows	**A1. Low Confidence Detection** • If classifier confidence is below 0.82, the event is flagged as *uncertain*. • The system sends it to manual review without alerting emergency dispatch. **A2. Camera Stream Disconnection** • System detects loss of video feed and marks camera status as "offline." • Operator is notified to troubleshoot or switch to backup feed. **A3. Model Threshold Adjustment** • Admin adjusts confidence or detection parameters (via Model Ops) to improve accuracy.
Exception Flows	**E1. Sensor or Feed Failure** • If a camera feed or processing node fails, the system logs the failure and switches to fallback nodes. **E2. Storage Full or Logging Failure** • If evidence cannot be stored, the system issues a "Storage Alert" to the control center. **E3. False Positive Alert** • Operator can manually mark an event as false; system retrains with feedback data during next model ops cycle.

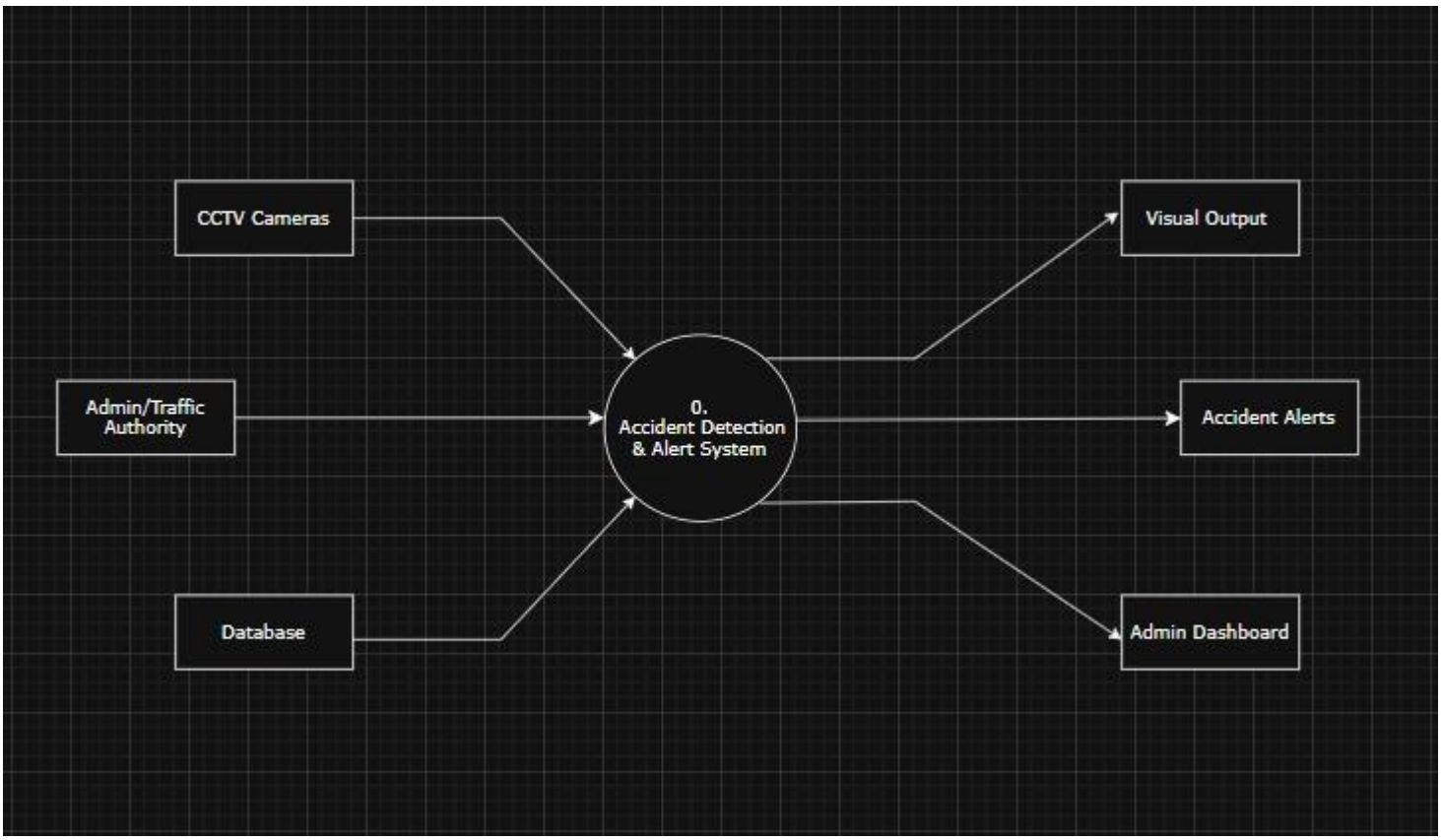
2.2 Activity and Swimlane Diagrams

Activity Diagram

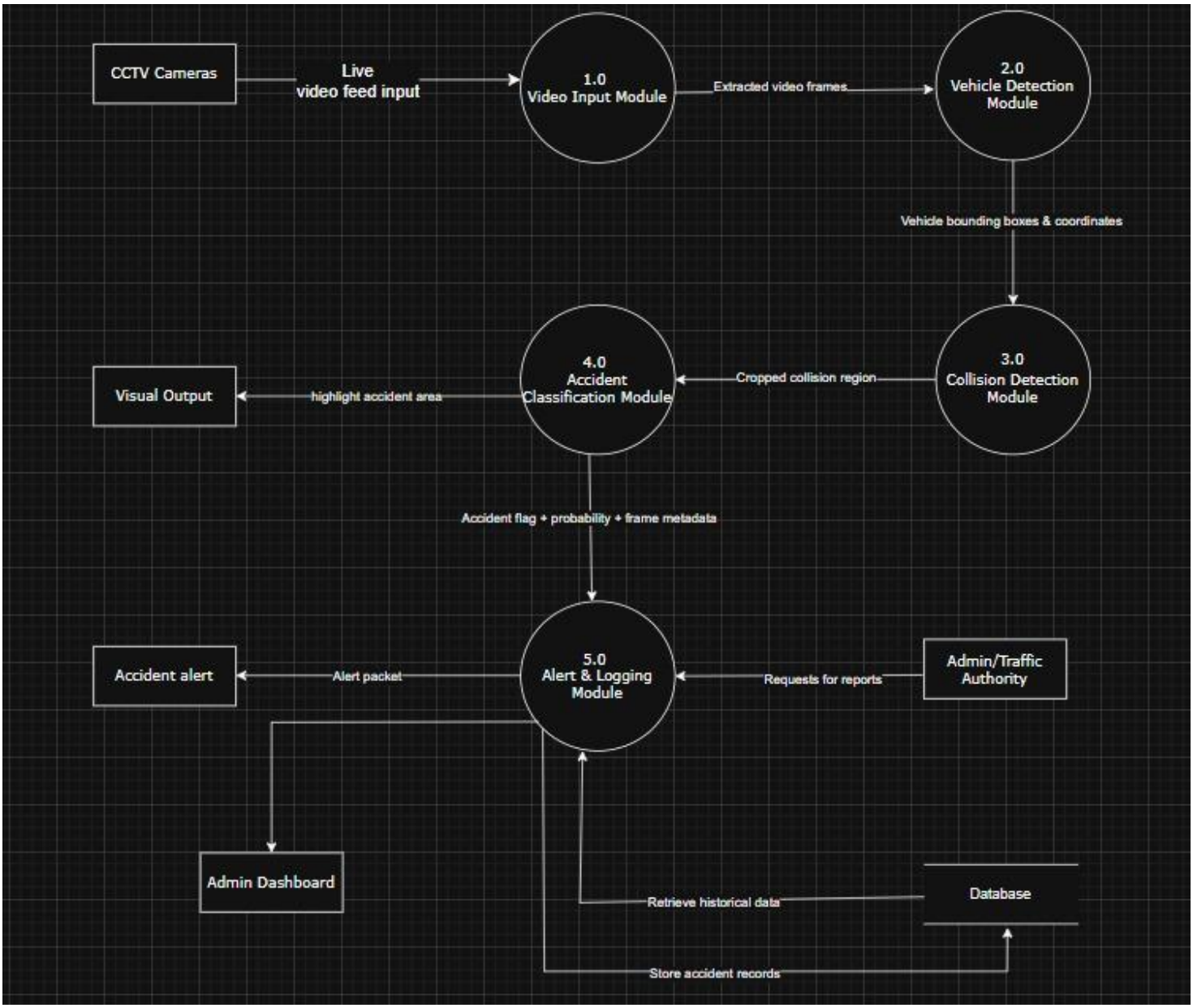


2.3 Data Flow Diagrams(DFD)

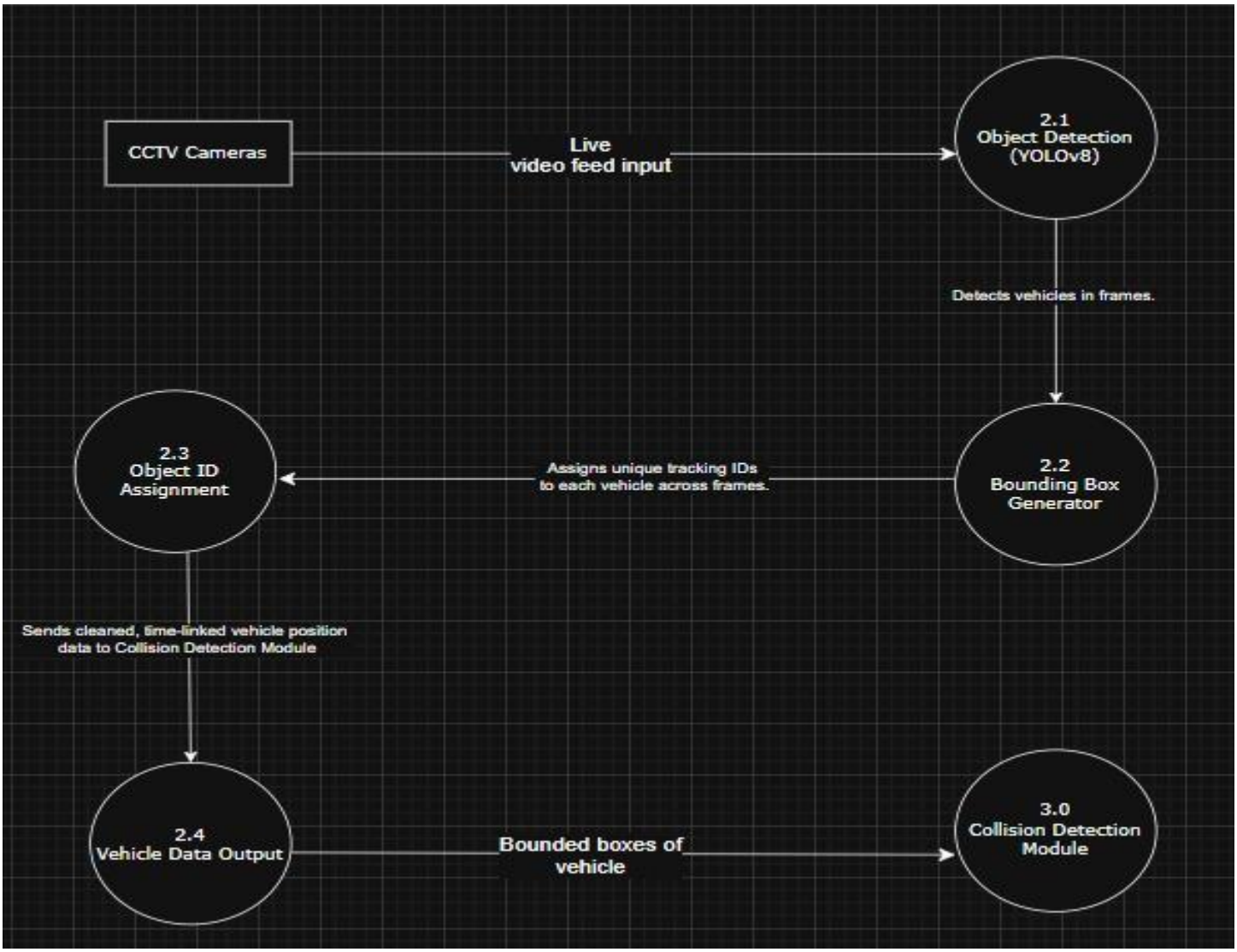
2.3.1 DFD Level-0



2.3.2 DFD Level-1



2.3.3 DFD Level-2



The **Level-2 DFD for Process 2.0 – Collision Detection & Alert Generation** consolidates multiple video input sources (e.g., highway CCTV, traffic junction cameras, and city surveillance streams) into a single abstracted data entity labeled **"Live Video Feed Source."** Although the system may ingest feeds from various physical camera networks, the processing workflow for frame extraction, object detection, collision checking, and alert triggering remains **uniform across all feeds**.

By representing all input cameras under one unified entity, the Level-2 diagram avoids repetitive sub-process blocks, maintains clarity, and preserves full logical alignment with the Level-1 DFD. This ensures the model remains **balanced, non-redundant, and easy to interpret**, while still fully expressing the system's continuous, multi-stream real-time monitoring behavior.

2.4 Software Requirement Specification

Please refer to the following page for the commencement of the Software Requirement Specification (SRS).

Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) for the **Collision Detection and Alert System** is to formally outline the functional and non-functional requirements necessary for the design, development, and validation of an automated traffic hazard monitoring and collision prevention solution. The system is intended to continuously observe vehicle movement, detect possible collision conditions in real time, and trigger timely alerts to reduce accidents.

This SRS serves as the single authoritative reference for developers, testers, and stakeholders. It ensures that the implementation remains technically consistent, verifiable, and aligned with safety objectives, particularly the prevention of road mishaps through early threat recognition and automated alerting mechanisms.

1.2 Product Scope

The Collision Detection and Alert System is designed to monitor real-time traffic surroundings using camera/video feed and smart detection algorithms. The system aims to:

- **Increase Road Safety:** Detect and warn about potential collisions before they occur.
- **Real-time Monitoring:** Continuously analyze vehicle movement and spacing.
- **Automated Alerts:** Trigger visual/voice/buzzer alerts to prevent impact.
- **Scalability:** Support integration with multiple camera inputs or scaling to larger road/intersection systems.
- **Efficiency:** Minimize processing delays using optimized processing workflows.

This system can be deployed on intelligent transportation systems, smart CCTV infrastructures, and vehicle safety modules.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Collision Detection System	Proposed system to detect potential vehicle collisions.
CCTV / Camera Feed	Video input used for detecting vehicle objects and motion.
Detection Module	Sub-system responsible for identifying vehicles and movement patterns.
YOLO / CNN Model	Machine learning models used for object detection .
Alert System	Notification system used to warn of collision risk.

1.4 References

- https://www.geeksforgeeks.org/opencv-and-computer-vision

- https://pjreddie.com/darknet/yolo/
- https://ieeexplore.ieee.org/document/vehicle-safety
- https://www.kaggle.com/datasets/vehicle-detection-datasets

2. Overall Description

2.1 Product Perspective

The Collision Detection and Alert System functions as an independent monitoring module integrated with video surveillance setups. The system processes live or recorded video feeds to detect vehicles, identify unsafe proximity, and evaluate speed/distance thresholds to predict collision scenarios.

It can operate as:
• A standalone system for real-time monitoring
• A backend module supporting traffic management dashboards

2.2 Product Functions

The main functional components include:
• **Video Input Handling:** Capture and preprocess video frames.
• **Vehicle Detection:** Identify and classify vehicles in real-time.
• **Distance/Movement Analysis:** Track object motion, spacing, and speed.
• **Collision Prediction: Apply** logic rules or ML-based inference to detect risk.
• **Alert Generation:** Warning alerts when required.

2.3 User Classes and Characteristics

User Class	Description	Key Characteristics
Traffic Control Operator	Person monitoring traffic conditions from a control room.	Basic computer knowledge, decision-making ability.
Driver (if in-vehicle unit)	Vehicle operator receiving in-car alerts.	Requires quick reaction ability.
System Administrator	Maintains system hardware/software.	Technical knowledge in configuration & monitoring.

2.4 Design and Implementation Constraints

i. The system must operate in real-time, minimizing latency.
ii. Requires stable video feed input from cameras mounted at appropriate angles.
iii. Processing hardware must support OpenCV / ML model execution (GPU recommended for YOLO).
iv. Environmental conditions (rain, fog, night) may affect detection accuracy.
v. Alert mechanisms must be clear, fast, and fail-safe.

2.5 Assumptions and Dependencies

i. Cameras are assumed to be properly calibrated and operational.
ii. Adequate lighting conditions are assumed for optimal detection.
iii. System depends on stable power and processing hardware availability.
iv. The user acknowledges alerts and responds responsibly

3. External Interface Requirements

3.1 User Interfaces (UI)

The system shall provide a clear and intuitive visual interface to monitor traffic conditions and collision alerts. The UI will consist of:

Live Video Display Screen: Shows real-time camera feed with detected vehicles highlighted using bounding boxes.
Alert Indicator Panel: Displays collision warnings, proximity alerts, and vehicle risk levels (Low/Medium/High).
Event Log View: Lists timestamps of detected collision events for review and reporting.
Settings Panel: Allows administrators to configure detection sensitivity, threshold distance, alert volume, and camera source.

The UI will prioritize clarity, readability, and minimal distraction, ensuring that alerts are instantly noticeable.

3.2 Software Interfaces

Camera Input System: The system shall interface with CCTV/vehicle-mounted cameras to obtain continuous video feed. (USB/IP/RTSP supported).
Detection Algorithm: The detection module shall process video frames to identify vehicles and distance relationships.
Alert System Interface: The system shall integrate with audible alarms, LED indicators, or dashboard UI notifications to deliver warnings in real-time.

4. Specific Requirements (Functional)

4.1 UC01: Initialize System

F.1.1: The system shall load camera feed(s) and detection modules at startup.
F.1.2: The system shall perform hardware and source connectivity checks.
F.1.3: The system shall display system status (Active, Standby, Error) on the UI.

4.2 UC02: Vehicle Detection

F.2.1: The system shall detect and mark vehicles in the camera feed in real-time.
F.2.2: The system shall track the movement of detected vehicles across frames.
F.2.3: The system shall assign unique IDs to vehicles to maintain identification consistency.

4.3 UC03: Distance and Collision Evaluation

F.3.1: The system shall calculate the distance between vehicles continuously.
F.3.2: The system shall compare measured distance with pre-defined threat thresholds.
F.3.3: The system shall identify high-risk collision scenarios based on distance and speed trends.

4.4 UC04: Alert Triggering

F.4.1: The system shall issue alerts when collision risk exceeds the safe threshold.
F.4.2: Alerts may be visual (UI highlight), auditory (buzzer/alarm), or digital (dashboard notification).
F.4.3: The system shall log each alert event with timestamp and risk severity.

4.5 UC06: Data Logging and Reporting

F.5.1: The system shall store history of detected collision alerts.
F.5.2: The system shall allow administrators to review previous logs.

5. Other Non-functional Requirements

5.1 Performance Requirements

- The system shall process video feed and detect vehicles at **a minimum of 20 frames per second**.Scalability
- Collision alert response time must be **less than 1 second** from event detection.

5.2 Security Requirements

- Access to system settings shall be **restricted to authorized users** only.
- Video data and logs shall be protected from unauthorized modification.

2.5 User Stories and Story Cards

1. User Stories:

Role	Goal (User Story)	Reason
Emergency Responder	As an Emergency Responder, I want to receive an instant alert with the accident's GPS location, timestamp, and severity, so that I can dispatch aid immediately and reduce medical intervention time.	Rapid response saves lives.
Traffic Authority	As a Traffic Authority, I want to view live camera feeds and visual alerts on a dashboard when an accident occurs, so that I can verify the incident and decide on traffic rerouting.	Efficient traffic management and accident verification.
System Administrator	As a System Administrator, I want to configure detection thresholds and tolerances, so that I can minimize false positives and ensure only meaningful alerts are triggered.	Keeps alerts reliable and reduces noise.
Traffic Safety Analyst	As a Traffic Safety Analyst, I want to access the database of recorded incidents (time, location, vehicle count), so that I can identify high-risk zones and suggest safety policies.	Data-driven traffic safety improvements.
System Administrator	As a System Administrator, I want to upload new video footage and labeled datasets, so that I can retrain AI models (YOLOv8/DenseNet-201) to improve detection accuracy.	Continuous AI model improvement.
Emergency Responder	As an Emergency Responder, I want access to short pre- and post-event video clips, so that I can gain immediate situational awareness and plan rescue efficiently.	Faster understanding of accident context.
System Operator	As a System Operator, I want the system to process multiple live CCTV feeds at high frame rate (~45 FPS), so that I can monitor multiple locations without lag.	Smooth, real-time monitoring across locations.

2. Story Cards:

2.1 <u>Alert Generation for Emergency Responders</u>

Field	Detail
Story ID	CDS-ER-001
Role	Emergency Responder
Goal	Receive instant alert with GPS, timestamp, and severity of accidents
Benefit	Dispatch aid immediately and reduce intervention time
Priority	High
Estimation	3 days
Component	Alert Management Module, Backend, Web Application
Dependencies	Collision Detection Module
Acceptance Criteria	1. Alert triggers within 5 seconds if collision probability ≥ 0.82.2. Alert includes timestamp, GPS, severity/probability, and link to live/recorded feed.3. Alert delivered via interface, email, SMS, or API.4. Short pre- and post-event video clip packaged with alert.

2.2 Real-Time Visual Verification for Traffic Authorities

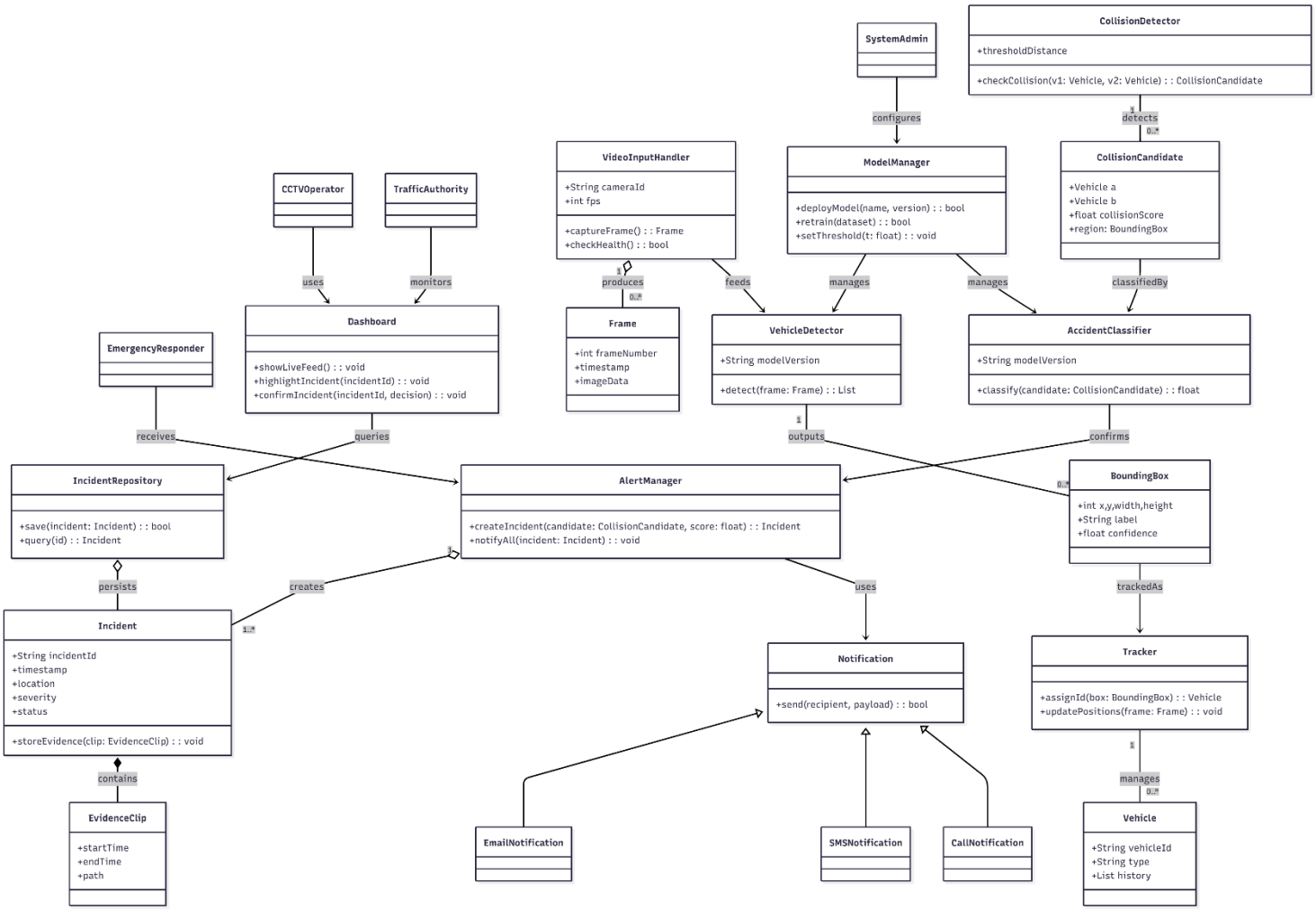
Field	Detail
Story ID	CDS-TA-002
Role	Traffic Authority
Goal	View live camera feed and visual alert (bounding boxes) on a dashboard
Benefit	Verify incident and decide on traffic rerouting
Priority	High
Estimation	4 days
Component	Web Application (Frontend), YOLOv8, Backend Server
Dependencies	Object Detection, Collision Detection
Acceptance Criteria	1. Live feed displayed at ≥30 FPS.2. Bounding boxes overlay detected vehicles.3. Collision region highlighted in red.4. Real-time status panel with confidence score.5. Dashboard is clean, intuitive, and fast.

2.3 Model Configuration and Tuning

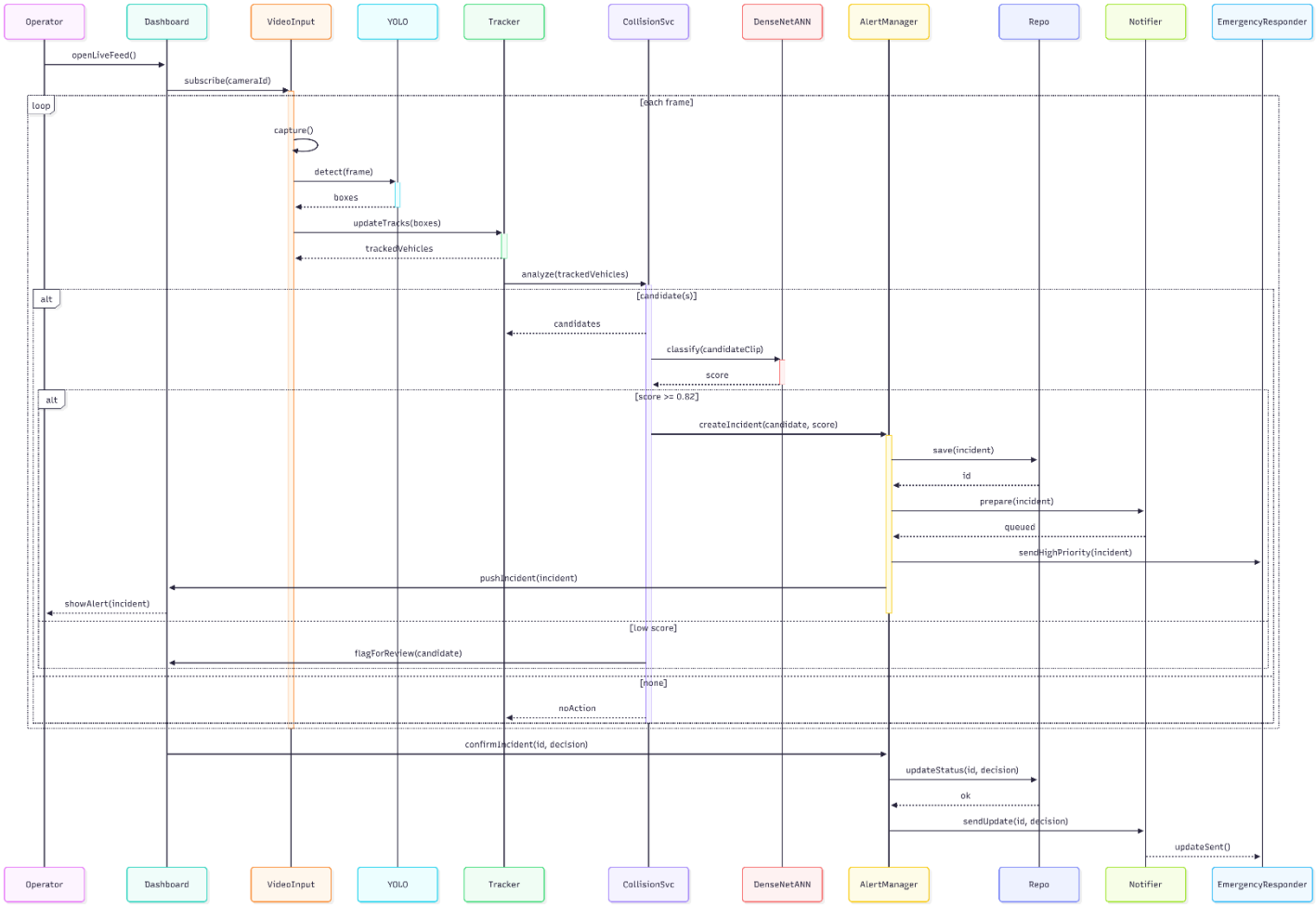
Field	Detail
Story ID	CDS-ADM-003
Role	System Administrator
Goal	Configure detection threshold and upload new training datasets
Benefit	Reduce false positives and improve detection accuracy
Priority	Medium
Estimation	5 days
Component	Admin Interface, Model Training Pipeline, Database
Dependencies	Dataset Management, Deployment Framework (Docker)
Acceptance Criteria	1. Admin interface allows setting classification threshold (default ≥ 0.82).2. Admin can retrain YOLOv8/DenseNet-201 with new dataset (5,000+ images).3. Training/inference Dockerized for portability.4. Admin can monitor Precision, Recall, F1 after retraining.

Phase - 3
Design Phase

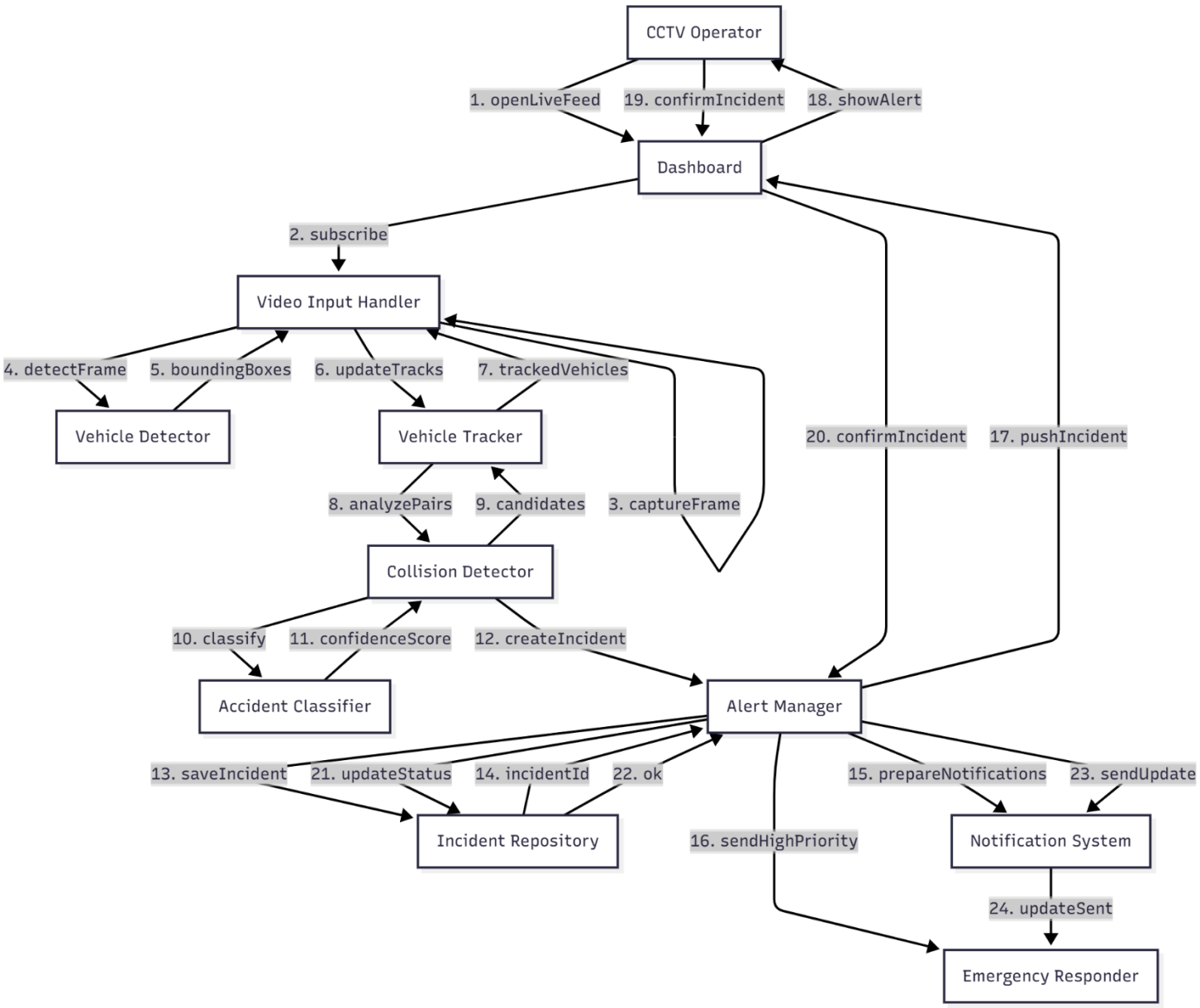
3.1 Class Diagram



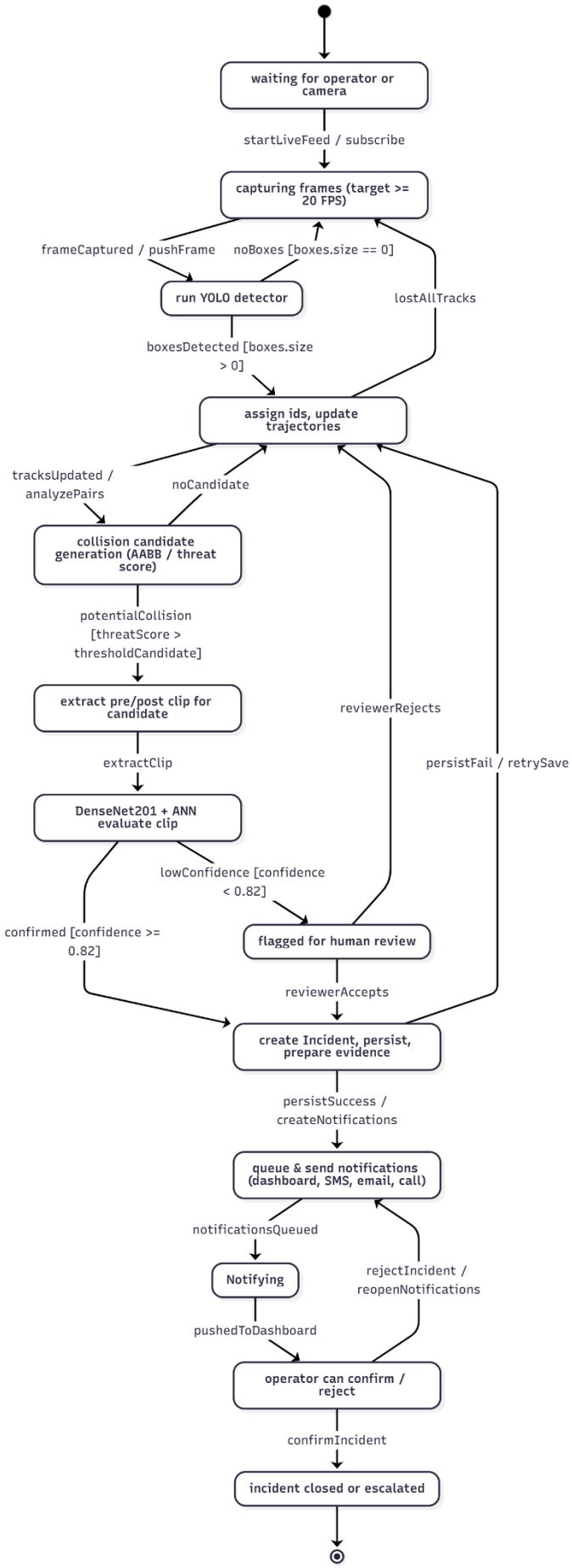
3.2 Sequence Diagram



3.3 Collaboration Diagram

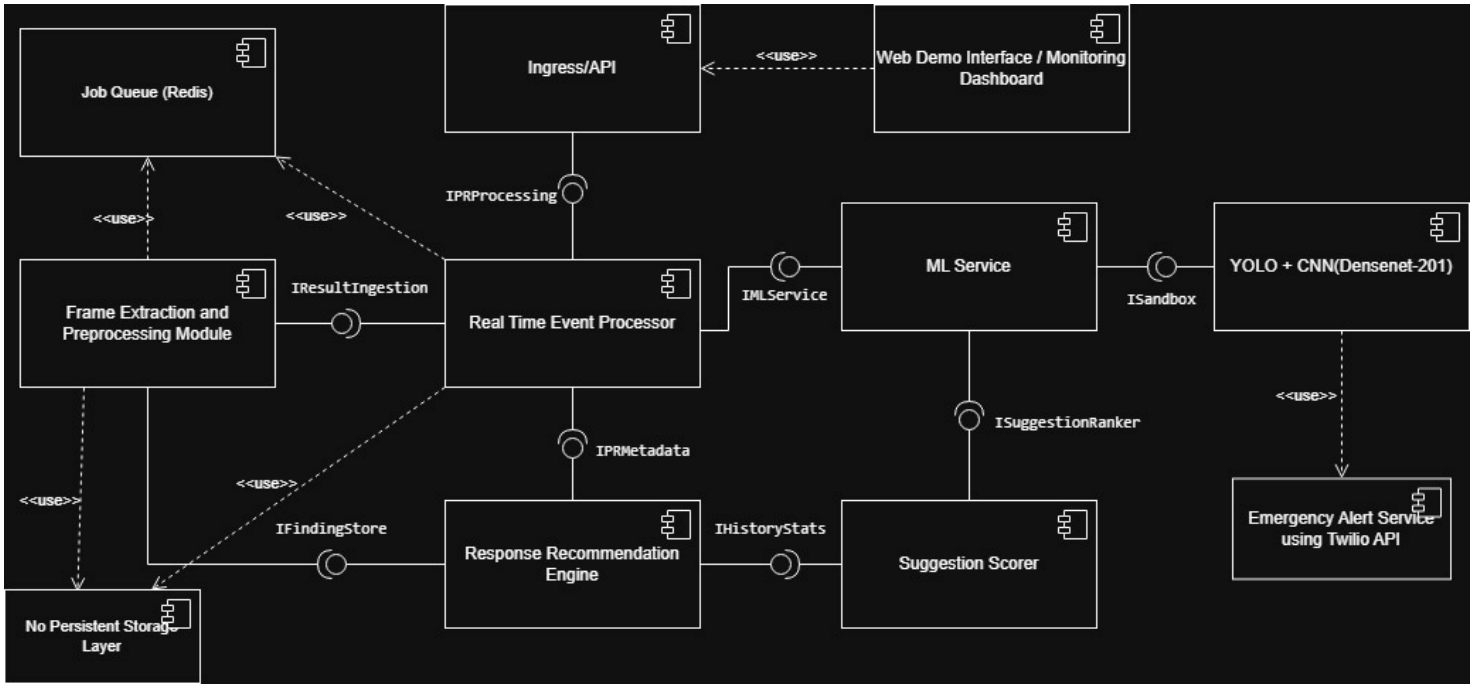


3.3 State Chart Diagram

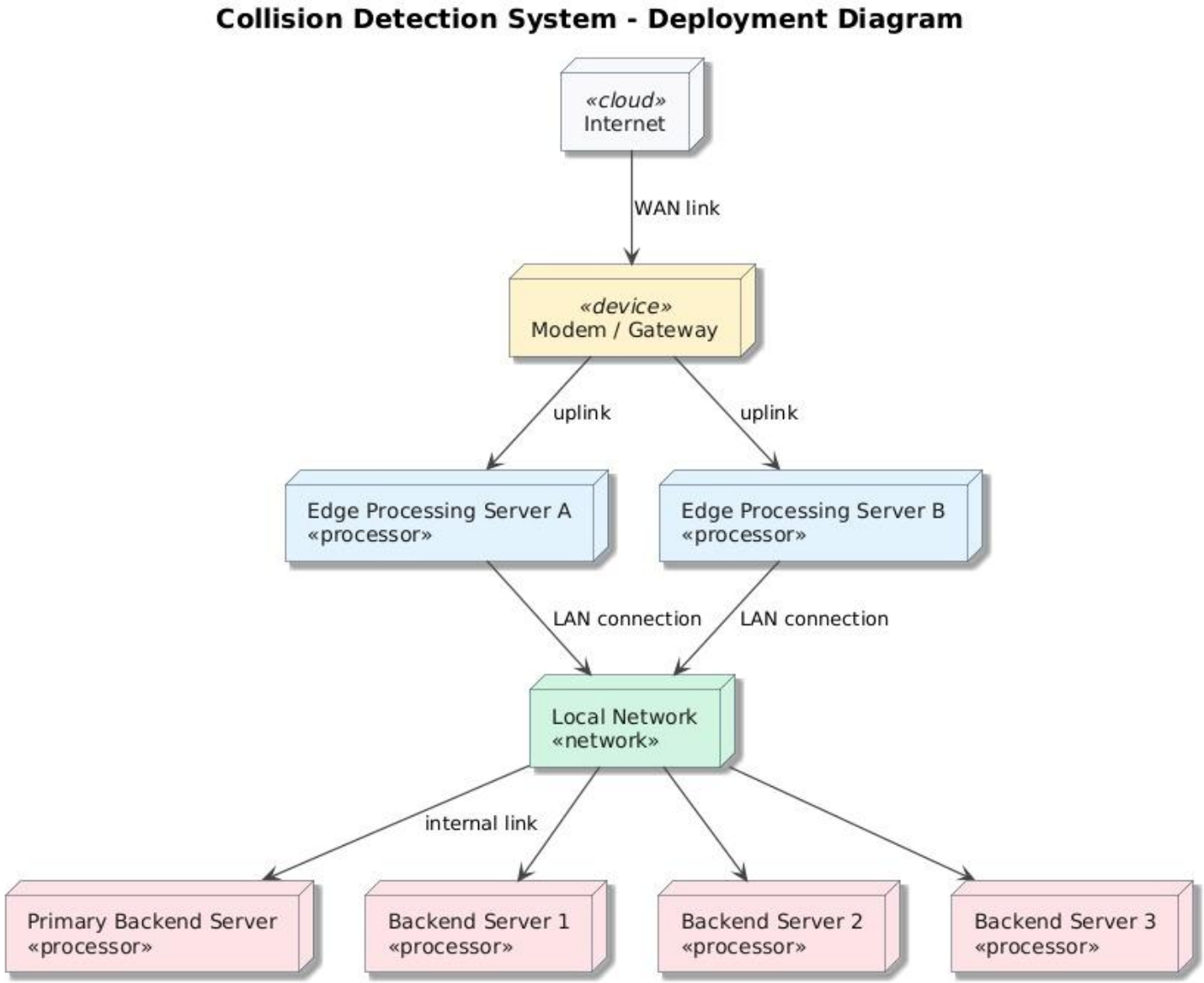


Phase - 4
Implementation

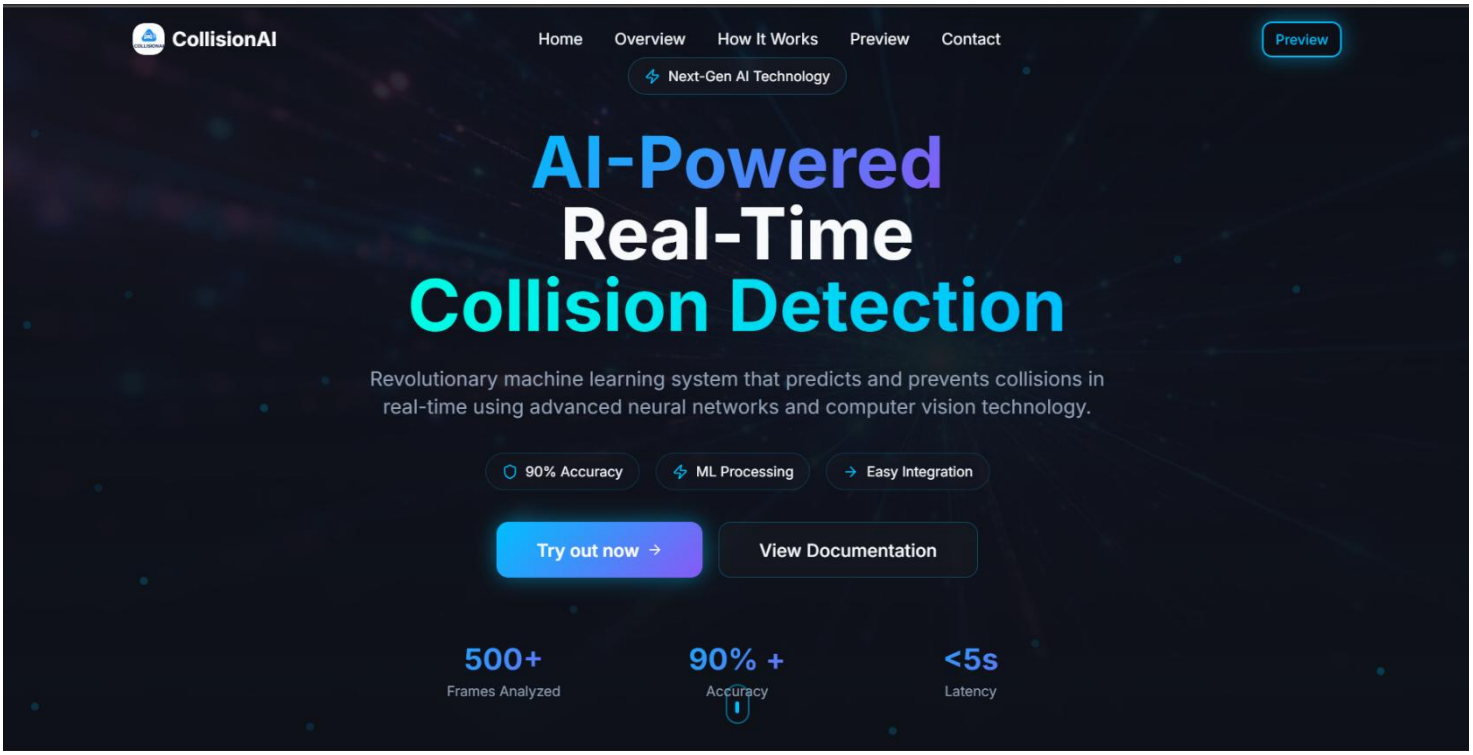
4.1 Component Diagrams

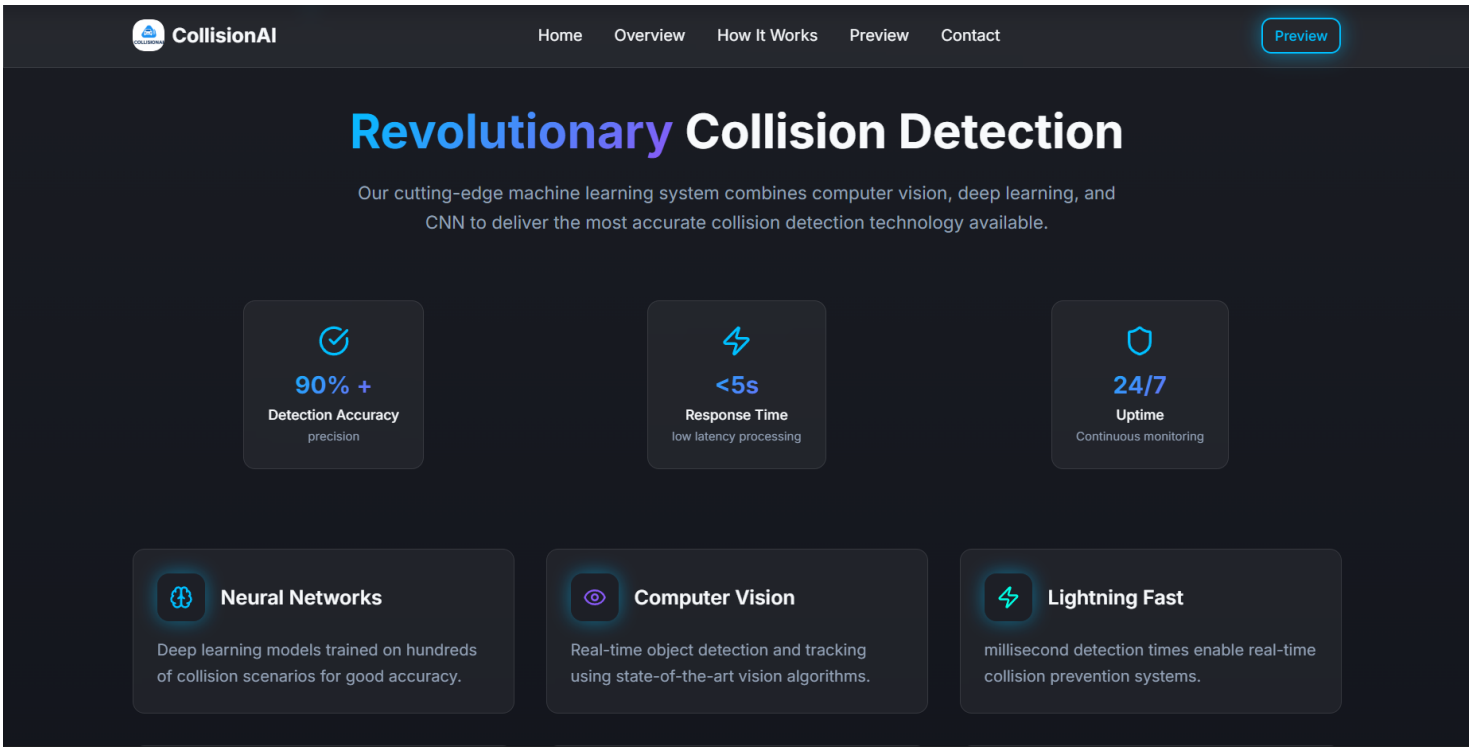


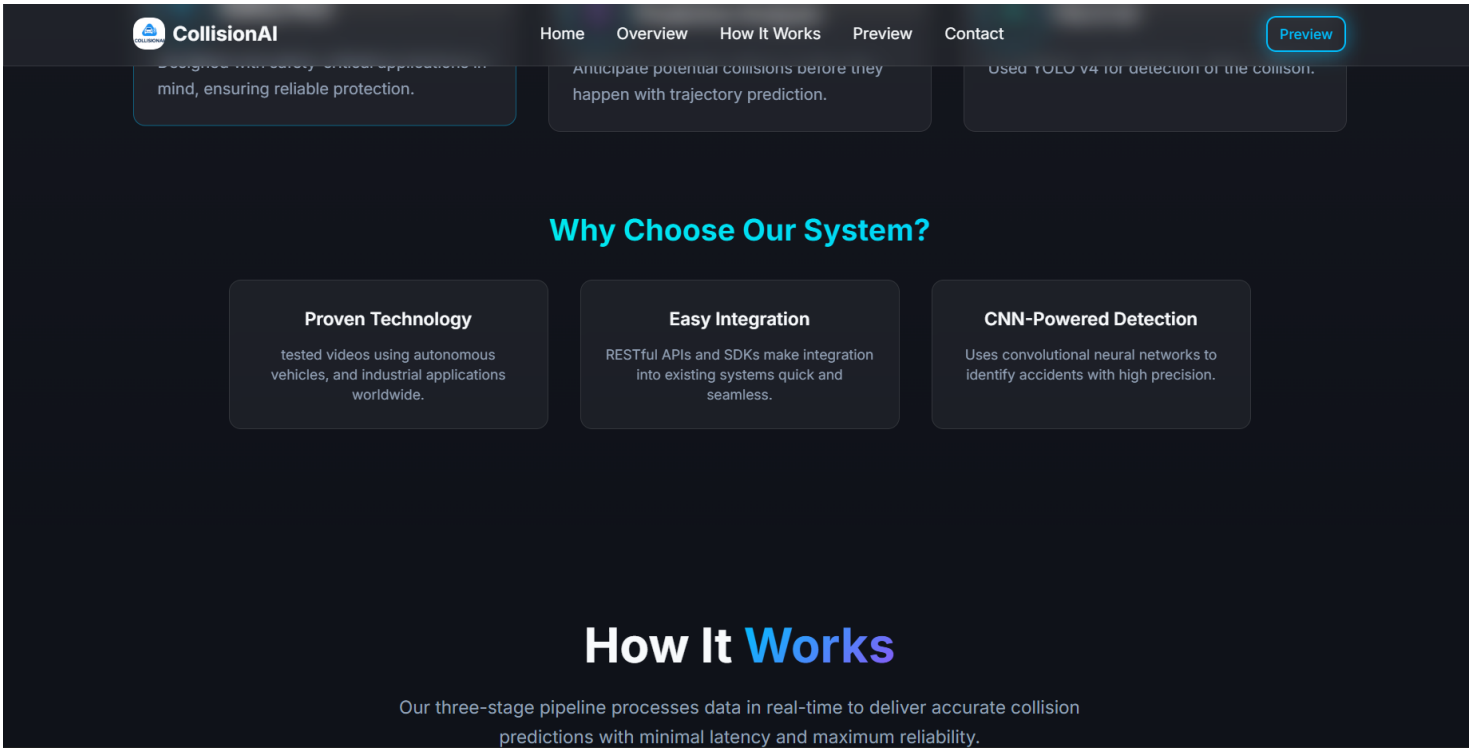
4.2 Deployment Diagrams

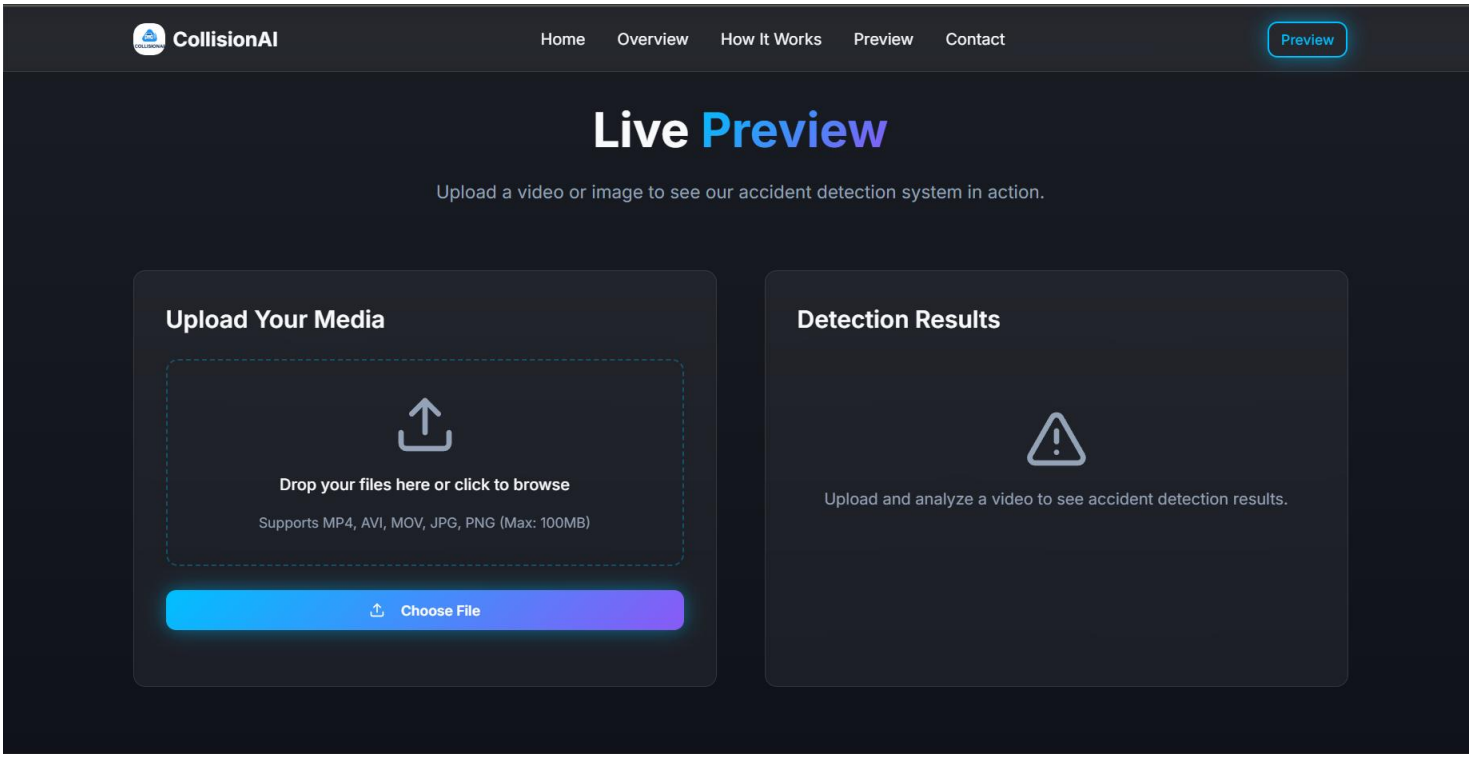


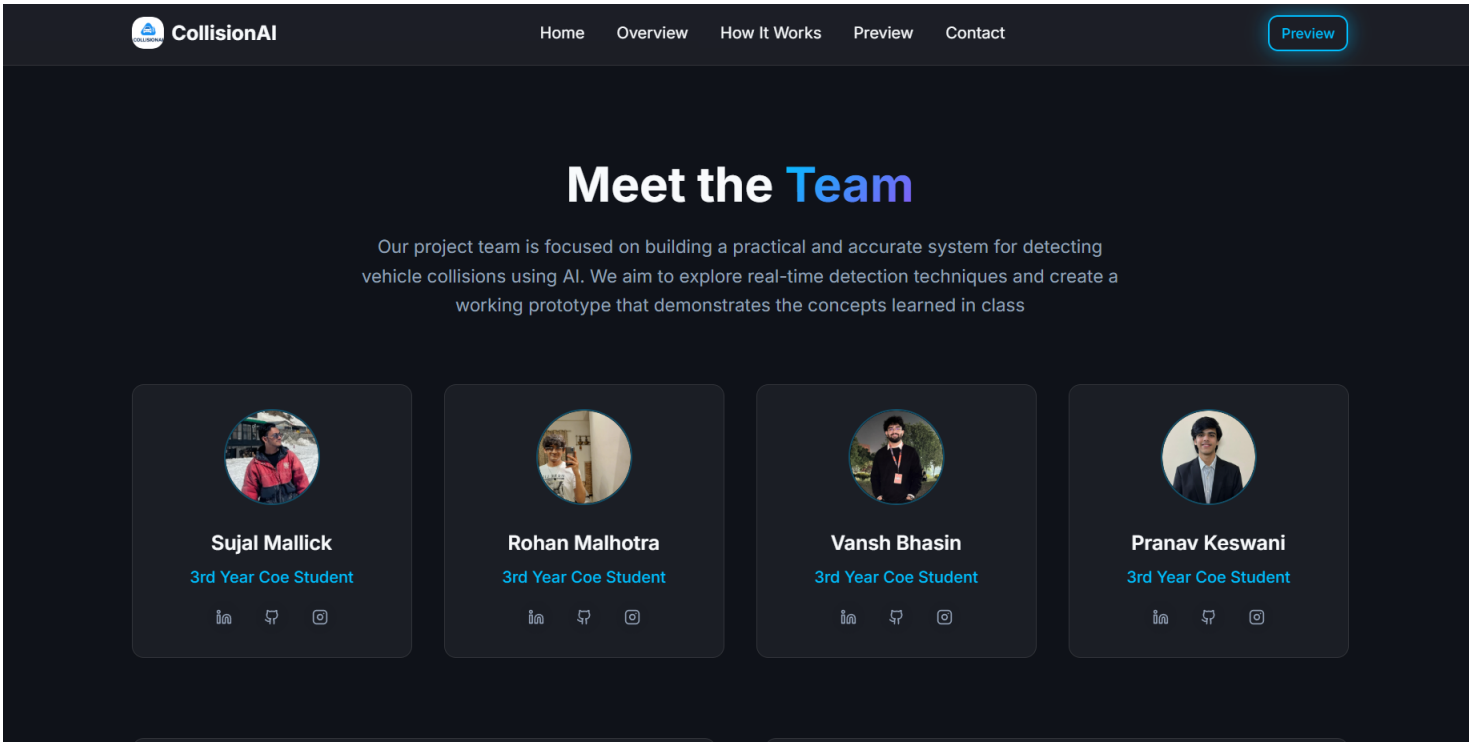
4.2 Screenshots

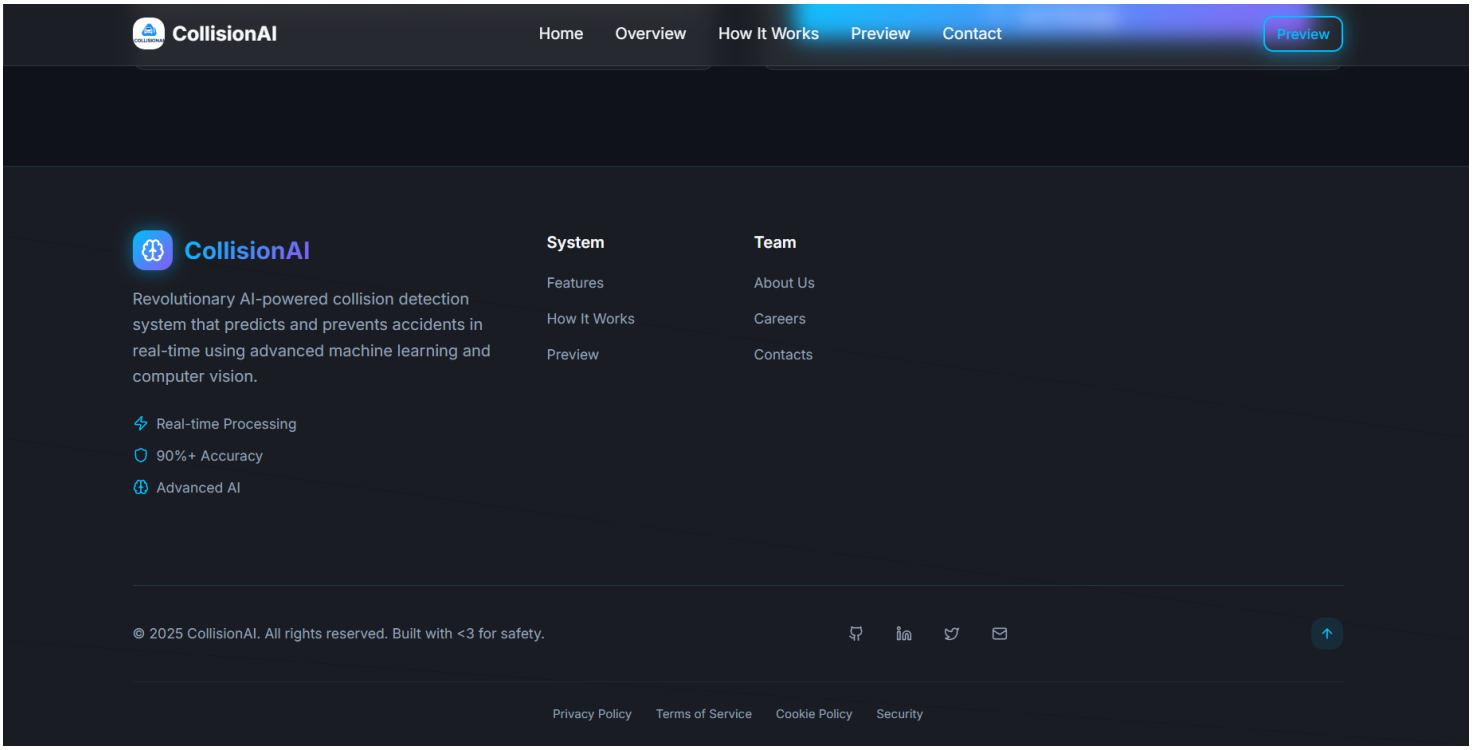





CollisionAI
Home
Overview
How It Works
Preview
Contact
Preview
mind, ensuring reliable protection.
happen with trajectory prediction.
Why Choose Our System?
Proven Technology
tested videos using autonomous vehicles, and industrial applications worldwide.
Easy Integration
RESTful APIs and SDKs make integration into existing systems quick and seamless.
CNN-Powered Detection
Uses convolutional neural networks to identify accidents with high precision.
How It Works
Our three-stage pipeline processes data in real-time to deliver accurate collision predictions with minimal latency and maximum reliability.


CollisionAI
Home
Overview
How It Works
Preview
Contact
Preview
Live Preview
Upload a video or image to see our accident detection system in action.
Upload Your Media
Drop your files here or click to browse
Supports MP4, AVI, MOV, JPG, PNG (Max: 100MB)
Choose File
Detection Results
Upload and analyze a video to see accident detection results.


CollisionAI
Home
Overview
How It Works
Preview
Contact
Preview
Meet the Team
Our project team is focused on building a practical and accurate system for detecting vehicle collisions using AI. We aim to explore real-time detection techniques and create a working prototype that demonstrates the concepts learned in class
Sujal Mallick
3rd Year Coe Student
Rohan Malhotra
3rd Year Coe Student
Vansh Bhasin
3rd Year Coe Student
Pranav Keswani
3rd Year Coe Student


CollisionAI
Home
Overview
How It Works
Preview
Contact
Preview
CollisionAI
Revolutionary AI-powered collision detection system that predicts and prevents accidents in real-time using advanced machine learning and computer vision.
Real-time Processing
90%+ Accuracy
Advanced AI
System
Features
How It Works
Preview
Team
About Us
Careers
Contacts
© 2025 CollisionAI. All rights reserved. Built with <3 for safety.
Privacy Policy
Terms of Service
Cookie Policy
Security

Phase - 5
Testing

5.1 Test Plan

1.1 Introduction

This Test Plan outlines the testing approach, objectives, scope, and methodology for the Collision Detection System, a full-stack cloud-deployed application consisting of:

- A CloudFront-hosted React + TypeScript + Tailwind CSS frontend
- A FastAPI backend hosted on AWS EC2 (Dockerized)
- A CNN model integrated in the backend for collision detection
- API communication via HTTPS endpoints
- The testing ensures that the deployed cloud system works correctly, efficiently, securely, and reliably for end users.

1.2 Objectives

• Validate the complete end-to-end flow: Frontend Upload → CloudFront → EC2 Backend → CNN Inference → Response to UI.

• Ensure CloudFront frontend functions correctly across supported browsers.
• Verify backend API availability and correctness.
• Confirm accurate CNN inference execution.
• Validate proper video/image upload and handling.
• Ensure UI responsiveness and clear user flows.
• Test robustness and error handling.

1.3 Scope

Included:
• Frontend (UI, responsiveness, file upload)
• CloudFront deployment correctness
• Backend API testing
• CNN inference logic
• Integration testing between frontend, backend, and ML pipeline

Excluded:
• Large-scale load testing
• GPU-specific inference
• Offline network simulations
• Mobile-first optimization (unless required)

1.4 Features to be Tested

• Video/Image upload
• Frontend responsiveness
• CloudFront caching behavior
• API endpoint functionality
• CNN inference output validation
• Error and invalid file handling
• End-to-end integration pipeline

1.5 Test Strategy

Functional Testing
• Verify all individual system features work as expected.

Integration Testing
• CloudFront ↔ EC2 API communication
• FastAPI ↔ CNN model inference
• File upload ↔ Processing ↔ Response flow

Performance Testing
• API inference response time < 3–5 sec
• CloudFront page load < 1 sec (cached)
• Backend container stability under repeated requests

Usability Testing
• UI clarity across devices
• Upload workflow intuitiveness
• Proper error messages and states

Basic Security Testing
• HTTPS enforcement
• File type validation
• Reject malformed requests

1.6 Test Environment

Frontend:
• Production URL: https://d10uokp8q1uvzj.cloudfront.net/
• React + TypeScript + Tailwind CSS

Backend:
• AWS EC2 (Amazon Linux)
• Dockerized FastAPI
• TensorFlow-CPU ML inference

1.7 Entry Criteria

• CloudFront deployed successfully
• EC2 backend operational and model loaded
• Docker container builds without error
• Environment variables configured
• All endpoints implemented

1.8 Exit Criteria

• All high-priority test cases pass
• No critical or blocking bugs

5.2 Test Cases

Test Case Template

Test Case #:TC_CDS_01	**Test Case Name:** Collision detection	**Page:** 1
System: Collision Detection and Alert Module	**Subsystem:** Demo/Preview Section	
Designed by: Sujal Mallick	**Design Date:** 19/11/20225	
Executed by: Sujal Mallick	**Execution Date:** 19/11/20225	

Short Description: verifies that the system detects collision event from a live CCTV video feed using yolo-based object detection and bounding-box logic.

Pre-conditions :
YOLO model and collision detection module are loaded in the backend.
The test video file is available and uploaded.
All dependencies are successfully loaded.
Output directory exists locally.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Upload the test video in the dropbox	Worked	pass	
2	Automatically runs the collison detection in the backend	Detection ml model worked	pass	
3	Vehicles appear in the frame.	System draws boundaries and sends image in response	pass	
4	Output returns the image of the detected collison	Shows detection results and confidence.	pass	

Post-conditions :
Collison photos are saved in the folder locally
System awaits till the next video is uploaded.

Test Case Template

Test Case #:TC-CALL-01
System: Collision Detection System (CDS).
Designed by: Vansh Bhasin
Executed by: Vansh Bhasin
Short Description: Testing the emergency calling service

Test Case Name: Automated emergency call when accident is detected
Page: 1 of 1
Subsystem: Alert Management
Design Date:19/11/2025
Execution Date:19/11/2025

Pre-conditions:
CCTV feed is active.
Accident detection model is running.
Emergency phone number is configured (My phone number).
Alert threshold Probability= 0.90.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Start the system and stream accident video	System detects vehicles normally.	Pass	
2	Collision occurs in video.	Collision event generated.	Pass	
3	Model's prediction probability ≥ 0.90 (e.g., 0.94).	System marks Accident Detected.	Pass	
4	System starts alert workflow.	Evidence clip saved + event logged.	Pass	
5	System triggers automatic call to emergency number using Twilio	Tester's phone (my contact here) receives a call.	Pass	
6	Answer the call.	Pre-recorded accident alert message plays.	Pass	

Post-conditions:
1. Emergency phone call is placed immediately after confirmed accident.
2. Evidence (frame) saved.

Test Case Template

Test Case #:TC_CDS_03

Test Case Name: Collision Detection

System: Collision Detection System (CDS).

Subsystem: Detection

Designed by: Rohan Malhotra

Design Date:19/11/2025

Executed by: Rohan Malhotra

Execution Date:19/11/2025

Short Description: Verify that the system detects the accident from the live feed and logs the accident data

Pre-conditions:

YOLO model and collision detection module are loaded in the backend.
The test video file is available and uploaded.
All dependencies are successfully loaded.
Output directory exists locally.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Upload the test video in the dropbox	System detects vehicles normally.	Pass	
2	Collision occurs in video.	Collision event generated.	Pass	
3	Automatically runs the collison detection in the backend	System marks Accident Detected.	Pass	
4	System starts alert workflow.	Evidence clip saved + event logged.	Pass	
5	Output returns the image of the detected collison	Shows detection results and confidence.	Pass	

Post-conditions:

1. Collison photos are saved in the folder locally
2. System awaits till the next video is uploaded.

Test Case Template

Test Case #:TC-CDS-04

Test Case Name: No Emergency Call on Low-Confidence Detection

System: Collision Detection System

Subsystem: Alert Management

Designed by: Pranav Keswani

Design Date:19/11/2025

Executed by: Pranav Keswani

Execution Date:19/11/2025

Short Description: Verify that if classifier confidence < 0.90 the system does not call emergency services but stores evidence and routes event for manual review.

Pre-conditions:

CCTV feed active.

Models deployed and alert threshold = **0.90**.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Start system and stream a video with an ambiguous near-miss/scrape (not clear accident).	Frame ingestion and collision candidate creation.	Pass	
2	Classifier returns confidence < 0.90 (e.g., 0.60)	System marks event No Accident	Pass	
3	System packages short pre/post clip	Evidence saved with frame of highest probability.	Pass	

Post-conditions:

1.No call record exists for this event unless escalated by operator as No Accident is detected by the model.

5.3 Test reports by peer

1.1 Overview

A peer evaluation of the **Collision Detection System (CDS)** was conducted to validate its functional accuracy, emergency alert reliability, real-time video processing stability, and dashboard usability. The primary objective was to ensure that the system performs collision detection correctly, responds rapidly, and maintains stable operation under continuous CCTV feed processing.
The evaluation aimed to assess:

- Collision detection correctness
- Object detection accuracy
- Alert generation reliability
- System performance during real-time video streaming
- Evidence logging and storage stability

Peers were instructed to perform the most essential tasks reflecting the core functionality of CDS. Each peer followed a predefined set of test cases designed to assess both functional and real-time system behavior.

1.2 Peer Evaluation Methodology

The peer testing workflow was structured into four phases:

1. Environment Verification

Testers ensured that:

- CCTV sample video feeds were active and streaming correctly
- Backend (Flask) and object detection models (YOLO + DenseNet) were running
- Database logging service was active
- No dependency or configuration issues existed

2. Free Interaction Stage

Peers interacted freely with the system to examine:

- Video feed smoothness
- Bounding box tracking behavior
- Collision event highlight clarity

3. Execution of Core Test Cases

Testers executed the following essential scenario tests:

- Load video feed
- Detect vehicles and track movement
- Detect collision event
- Automated emergency alert generation
- Incident logging and retrieval

4. Observation Logging

Peers recorded:

- Expected vs. actual detection results
- System latency during detection and alert generation
- UI clarity during emergency events
- Error messages, if any

Test Case ID	Test Case Name	Peer Tester	Result	Comments
TC-CDS-04	No Emergency Call on Low-Confidence	Pranav Keswani	Pass	System correctly avoided calling emergency services for confidence <0.90; evidence saved.
TC_CDS_03	Collision Detection (Live Feed / Video Upload)	Rohan Malhotra	Pass	Vehicle detection, collision event generation, and logging worked as expected.
TC-CALL-01	Automated Emergency Call When Accident Detected	Vansh Bhasin	Pass	System placed emergency call instantly after confidence >0.90; alert message played.
TC_CDS_01	YOLO-Based Collision Detection	Sujal Mallick	Pass	Detection module loaded correctly; bounding-box logic worked; confidence displayed.

5 . Overall Peer Evaluation:

All core features functioned correctly during peer testing.

Emergency call feature worked only on high-confidence cases, as required.

System behaved safely on false positives, with correct manual review flow.

Evidence saving and logging operated normally across all tests.

No crashes or UI failures were reported.