Software Requirements Specification

for

Wildlife Monitoring and Poaching Prevention

Prepared by

Vedant Baldwa	**23UCC611**	**23ucc611@lnmiit.ac.in**
Vidhi Jain	**23UCC613**	**23ucc613@lnmiit.ac.in**
Vasu Sharma	**23UCC610**	**23ucc610@lnmiit.ac.in**
Prateek Bajpai	**23UCC585**	**23ucc585@lnmiit.ac.in**

Instructor: Dr. Ashish Kumar Dwivedi

Course: Software Development Lab

Date: - 06/10/2025

Table Of Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0.0	Vedant, Vidhi, Vasu, Prateek	Completed the first draft of the SRS document along with all the diagrams for specifying the requirements.	XX/XX/XXXX

1. Introduction

1.1. Document Purpose

This SRS describes the functional and non-functional requirements for the **Wildlife Protection and Poaching Detection** software system. The primary goal is to provide clear, testable requirements that will guide design, implementation, verification, and validation for a system that uses computer vision and supporting software to monitor wildlife, detect poaching events, and generate actionable alerts.

1.2. Product Scope

The product is a software system that ingests data from cameras (images and video streams), runs object detection and poacher/activity detection models, visualizes detections (bounding boxes, labels), logs events and issues alerts (email/SMS/push). Product focuses on wildlife species detection (35 classes) and video/image-level detection and visualization also include poaching detection models.

Need of the Product:

- Reduce response time to poaching incidents by automating detection and alerting.
- Scale monitoring across many camera feeds and locations.
- Provide accurate species-level detection to support population monitoring and research.
- Offer an auditable event log and dashboard for rangers and conservation managers.

1.3. Intended Audience and Document Overview

This SRS is intended for the project team (developers, ML engineers), project managers, test engineers, and stakeholder representatives who will review and approve the requirements. The key readers include:

- Professor: To review the requirements, ensure they align with project goals, and evaluate the proposed solution.
- Development Team: Software developers, AI specialists, and testers who will design, implement, and maintain the system based on these requirements.

- **Primary Stakeholders:** Park rangers, anti-poaching units, conservation NGOs, wildlife researchers.
- **Secondary Stakeholders:** System administrators, data scientists/model developers, local communities, law enforcement partners.

Document Overview:

This document specifies the functional and non-functional requirements of the software, providing a comprehensive description of its intended capabilities. It includes the system's purpose, scope, objectives, detailed functional requirements, external interfaces, and design constraints. The SRS serves as a reference for developers, testers, and stakeholders to ensure a shared understanding of the system's requirements and expectations.

1.4. Definitions, Acronyms & Abbreviations

Acronym	Full-Form/Definition
YOLO	You Only Look Once (real time object detector)
CV	Computer Vision
ML	Machine Learning
POI	Point of Interest
API	Application Programming Interface
FPS	Frames per Second
ER	Entity – Relationship
SRS	Software Requirements Specification
UML	Unified Modelling Language

1.5. Document Conventions

- Main heading Titles:
  - Font: Arial
  - Face: Bold White
  - Size: 22
- Subheading Titles:
  - Font: Arial
  - Face: Bold Black
  - Size: 14
- Other text explanations:
  - Font: Arial
  - Face: Black
  - Size: 11

1.6. References and Acknowledgement

- Python Documentation: Available at :https://docs.python.org/3/
- African Wildlife Dataset (Kaggle) – Training Data.
- Animals Detection Images Dataset (Kaggle) – Training Data.
- https://www.visual-paradigm.com/guide/
- UML Diagrams Guide
- FastAPI Documentation: Available at: https://fastapi.tiangolo.com/
- Lecture Slides.

2. Overall Description

2.1. Product Overview

This system is a standalone monitoring and alerting system that can operate on the edge (on-device inference on an embedded GPU/CPU) or centrally (server-side inference). The ML model (YOLO) is a key sub-system. The software integrates with camera inputs, local storage, a model inference runtime, a dashboard web application, and alerting channels.

System Boundaries:

- **In Scope:** Image/video ingestion, model inference (current wildlife detection), visualization (bounding boxes), event logging and basic alerts.
- **Out of Scope:** Automated hardware control (camera PTZ), advanced multi-camera tracking and re-identification, direct integration with third-party dispatch systems (can be supported later via APIs).

2.2. Product Functionality

- Ingest Images and video streams from cameras or uploading media.
- Run object detection per frame/video, produce bounding boxes and labels with confidence scores.
- Display detections on images/videos using OpenCV overlays.
- Store Detections and metadata into a database with timestamps, GPS (when available), and cameraID.
- Provide search/filtering by species, confidence, camera, time-range.
- Generate alerts for suspicious events, human presence, or poachers detected.
- Export event reports for stakeholders.

2.3. User Classes and Characteristics

- **Rangers/Field Users:** Need an urgent alerts vis SMS/push. Low tolerance for false negative, moderate tolerance for false positives.
- **Admins:** Configure cameras, manage system settings, review logs.

- **Data Scientists:** Access raw images, inference logs, and model performance for retraining.
- **Researches:** Query aggregated detection statistics for ecological studies.

2.4. Operating Environment

- **Server(central) deployment:** Linux (Ubuntu 20.04+), Python 3.8+, GPU (optimal, CUDA 11.X) for inference/training.
- **Edge deployment:** Jetson/TX2/RTX Embedded, Raspberry Pi 4 (CPU-only) with ONNX runtime for optimized model.

2.5. Design and Implementation Constraints

- **Model size and runtime:** YOLO model must meet resource constraints on chosen edge devices
- **Camera connectivity:** often intermittent, system must tolerate offline operation and batch upload.
- **Privacy/regulatory:** human face data must be handled per local privacy laws; system should allow redaction.

2.6. Assumptions and Dependencies

- Cameras provide timestamps and optionally GPS metadata.
- Initial ML training datasets (from Kaggle) are representative but further labelled data will be required for poaching detection.
- Users have basic smartphone or ranger radios for receiving alerts.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

The system does not provide a traditional end-user interface such as a dashboard or web application. Instead, the primary "interface" to the user is the visual output of the detection pipeline in the form of annotated images and video frames.

- **Image Predictions:**

- **Video Prediction:**
  - Video Prediction -01
  - Video Prediction -02

3.1.2. APIs

- REST API for querying events, uploading media, and retrieving detection metadata.
- Authentication via JWT/OAuth2 for API endpoints.

3.1.3. Hardware Interfaces

- Support RTSP/HTTP camera streams and scheduled photo uploads.
- Local disk or cloud storage for media.

3.1.4. Third-party Services

- Optional SMS gateway (Twilio or local equivalent), Email SMTP, Push notifications service.

3.2. System Requirements (Functional Requirements)

3.2.1. FR-001: Ingest Image and Video Sources: System shall accept input from RTSP camera streams, uploaded image files, and uploaded video files.

Inputs: RTSP URL, image (.jpg/.png), video (.mp4/.avi).

Outputs: Stored media entry and raw frames enqueued for inference.

3.2.2. FR-002: Run Object Detection Inference: For each input frame, run the YOLO model and produce a list of detections (class, confidence, bounding box in pixel coordinates).

Inputs: Frame image.

Outputs: JSON detection record with class_id, class_name, confidence, bbox= [x, y, width, height], timestamp, camera_id.

3.2.3. FR-003: Visualize Detections on Media: System shall render bounding boxes and labels onto frames for display in the dashboard and generate annotated media for exports.

Inputs: Raw frame and detections.

Outputs: Annotated image or video stream.

3.2.4. FR-004: Store and Index Detection Events: Persist detection metadata (camera_id, timestamp, species, bbox, confidence, media reference) into a searchable database.

Inputs: Detection JSON.

Outputs: DB records and indexes for quick querying.

Non-functional constraints: DB must handle X inserts/sec (see Performance Requirements).

3.2.5. FR-005: Event Filtering and Search: Dashboard shall support querying events by species, time-range, camera, confidence threshold, and geolocation.

3.2.6. FR-006: Alerts for Suspicious Events: System shall generate alerts when specified conditions are met (e.g., human detected in protected area, human with weapon detected, repeated human presence at night) and send them via configured channels (SMS/Email/Push).

Inputs: Detection events and configured alert rules.

Outputs: Alert messages and log entries.

3.2.7. FR-007: Model Management and Retraining Pipeline: Provide interfaces for uploading new annotated datasets, triggering retraining of models (or exporting training-ready datasets and metadata), and versioning models

Outputs: Trained model artifacts with version metadata.

3.2.8. FR-008: Bounding Box Export and Reports: Export detection events and annotated snapshots as reports (CSV, JSON, PDF) for stakeholders.

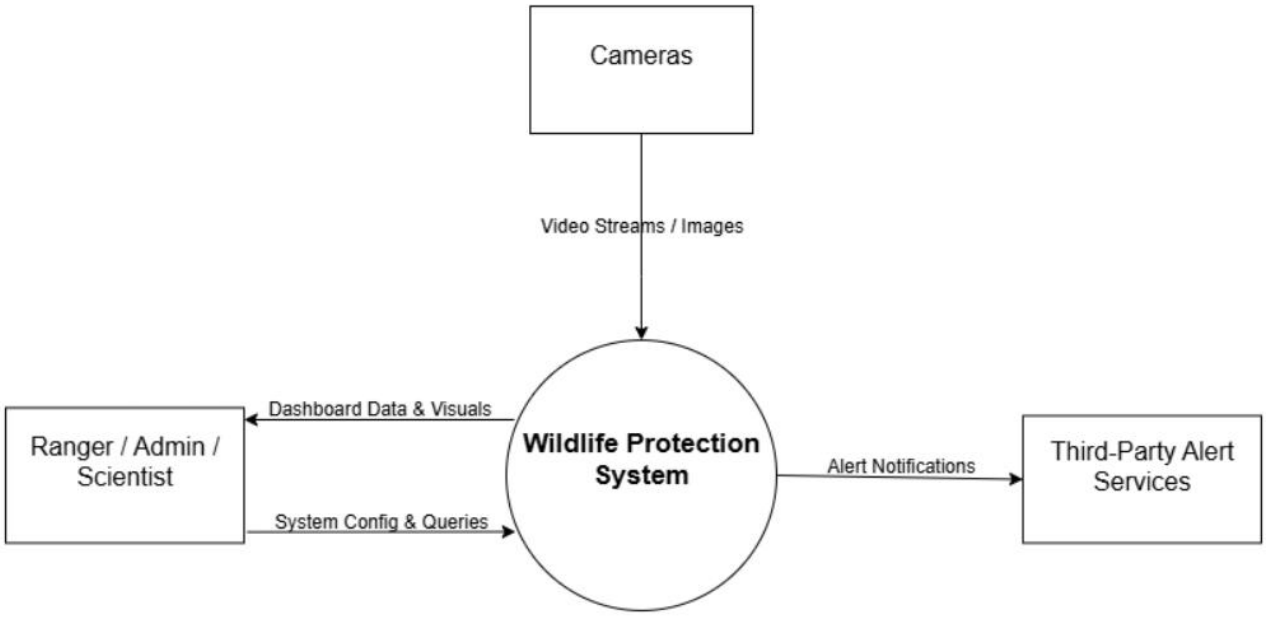
3.2.9. FR-010: Audit Log and User Management: Maintain an audit trail of user actions and system events. Provide role-based access control (RBAC) for system features.

(Additional FRs may be added for poaching-detection-specific models, multi-camera tracking, and UAV integration.)

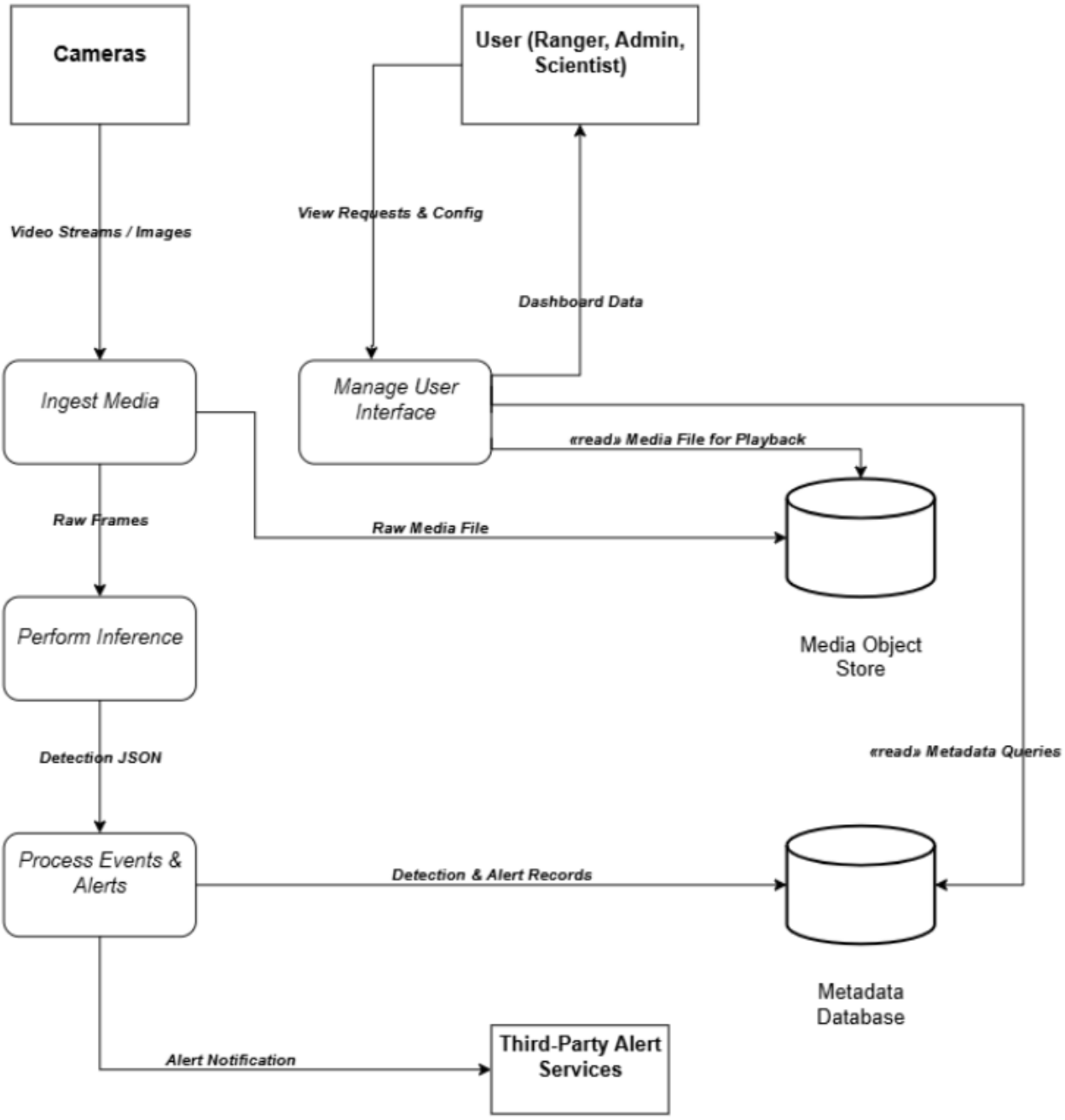
3.2.10. FR-009: Offline Buffering and Batch Update: Edge nodes must buffer detection metadata and media while offline and sync to the server when connectivity is restored.

3.3. Use Case Model

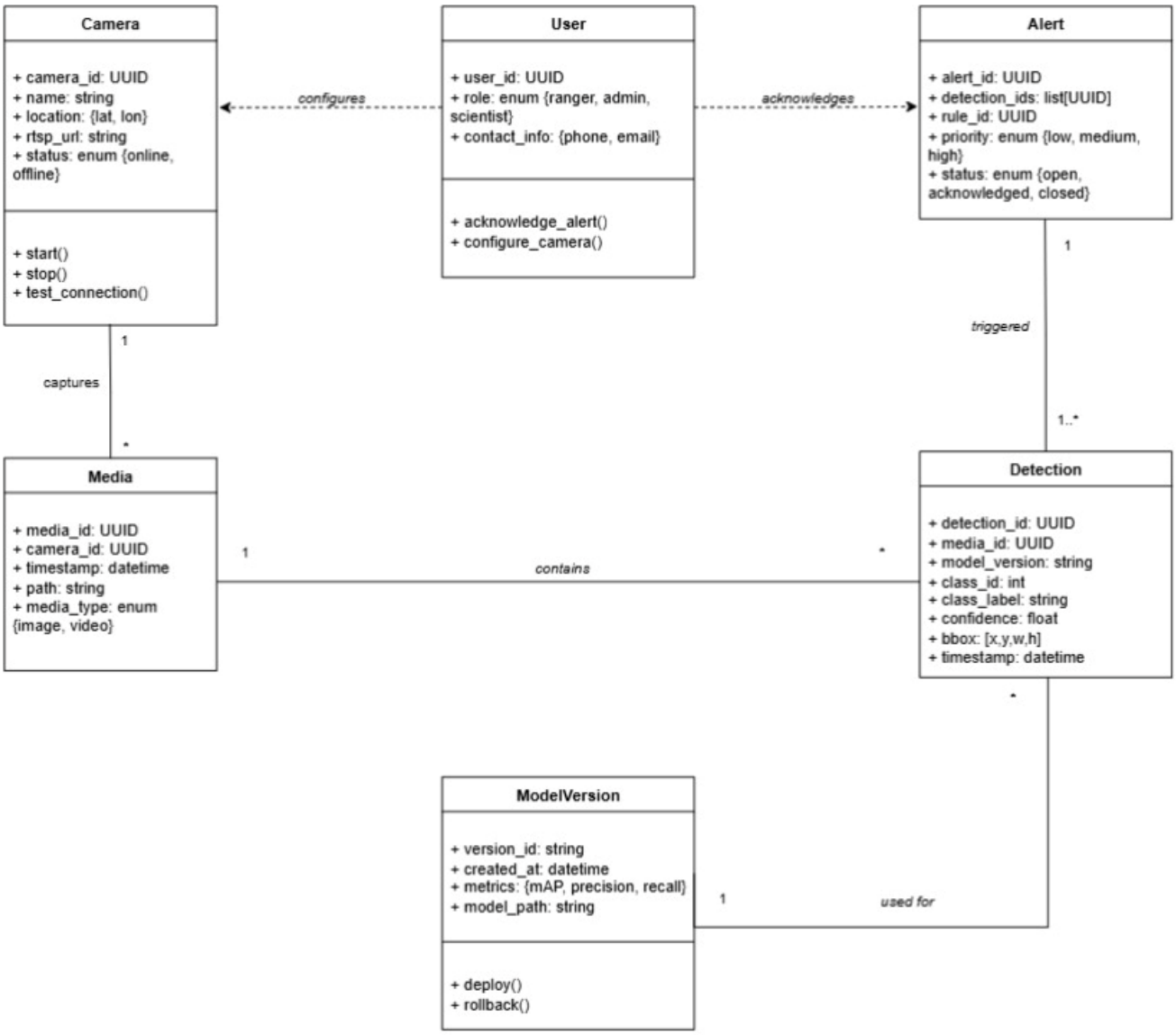
3.3.1. Data Flow Diagram (Level 0): (Full Diagram Link)



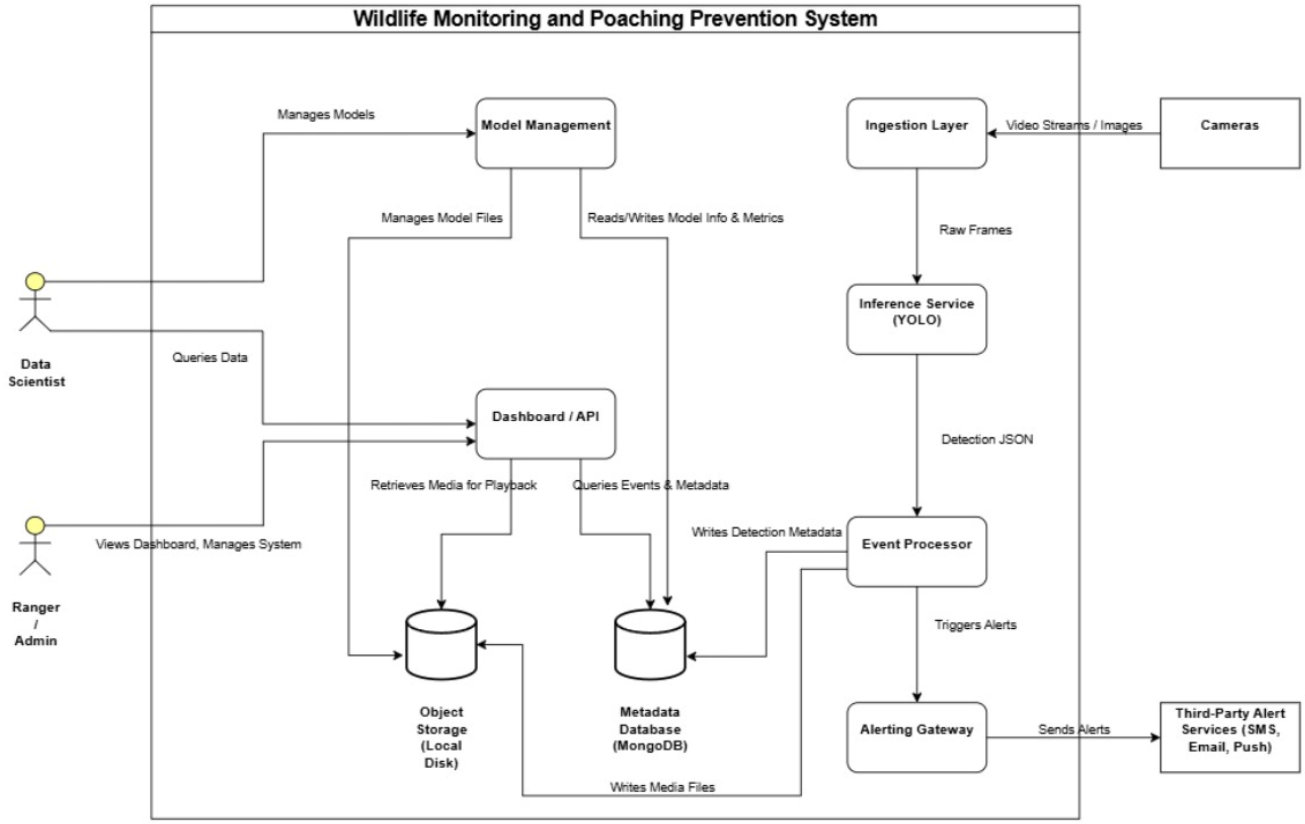
3.3.2. Data Flow Diagram (Level 1): (Full Diagram Link)



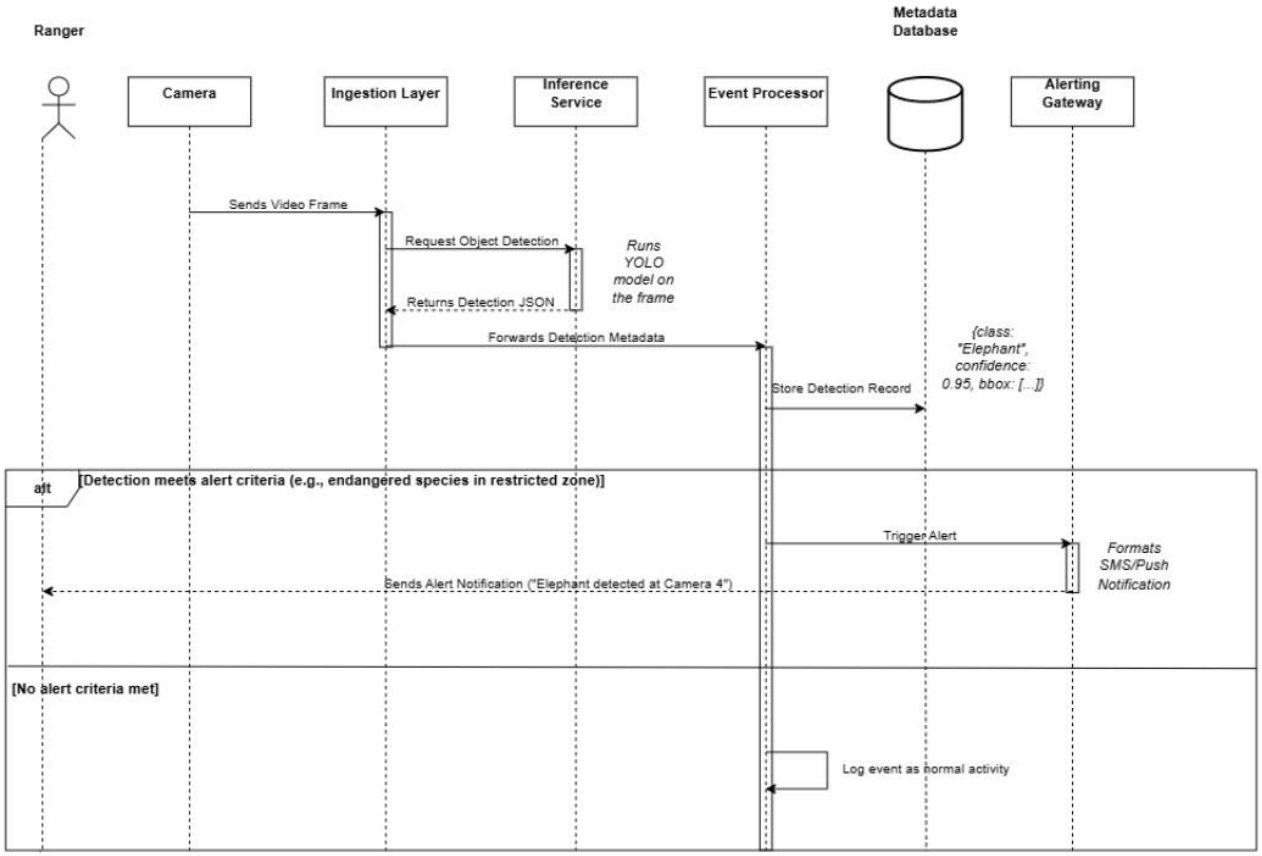
3.3.3. Class Diagram: (Full Diagram Link)



3.3.4. Use Case Diagram: (Full Diagram Link)



3.3.5. Sequence Diagram: (Full Diagram Link)



3.3.6. Decision Table:

	Rule 1: High-Risk Poaching Threat	Rule 2: Suspicious Activity	Rule 3: Endangered Species Sighting	Rule 4: Normal Animal Activity	Rule 5: Low Confidence Event	Rule 6: Default/Else
Conditions						
Detected Class	Human	Human	Endangered Species	Common Species	Any	-
Confidence Score	>=80%	>=80%	>=80%	>=80%	<80%	-
Location / Zone	Restricted	Restricted	Any	Non-Restricted	Any	-
Time of Day	Night	Day	Any	Any	Any	-
Actions						
Log Event	X	X	X	X	X	X
Trigger Alert	X	X	X			
Set Alert Priority	High	Medium	Low			

4. Other Non-functional Requirements

4.1. Performance Requirements

- **PR-001:** The central server shall process up to 10 concurrent RTSP streams at 5 FPS each with detection latency < 500 Ms per frame on a GPU instance
- **PR-002:** Edge deployment shall support at least 1–2 FPS on small embedded devices with an optimized model.
- **PR-003:** Database shall support 100 inserts/sec sustained without data loss (configurable per deployment).
- **PR-004:** The system shall support **multi-tenant deployments**, ensuring data separation between conservation areas (parks).

4.2. Safety Requirements

- **SR-001:** False alerts that could endanger rangers must be minimized, system will include humanin-loop confirmation for high-risk dispatch decisions.

4.3. Security Requirements

Security is critical to protect sensitive wildlife and human data.

- **SEC-001:** All API endpoints must be authenticated and authorized.
- **SEC-002:** Sensitive media and logs must be stored encrypted at rest (AES-256) and transmitted via TLS 1.2+.
- **SEC-003**: Role-based access control shall limit who can view images with humans and who can trigger alerts.

4.4. Logical Database Requirements

- Entities: Cameras, Media, Detections, Alerts, Users, Models, TrainingJobs.
- **Each Detection record shall include:**
  - detection_id
  - media_id
  - camera_id
  - timestamp
  - class_id
  - class_label
  - confidence
  - bounding box (bbox)
  - model_version
  - extra_metadata

.

4.5. Software Quality Attributes

4.5.1. Usability: The system must be intuitive for field rangers and other end users. Key actions like accessing live camera views or reviewing alerts should require minimal steps.

- **UR-001:** Field rangers shall be able to access live camera views and the latest alerts within **three clicks or less**.
- **UR-002:** The system shall provide an onboarding help page and minimal training material to support quick adoption.

4.5.2. Reliability: The system must maintain while gracefully handling component failures. Edge nodes must work offline and synchronize later.

- **RR-001:** The central server shall have a target availability of **99.5% uptime**.
- **RR-002:** The system shall support **graceful degradation**, i.e., if the model service fails, the system must still record and store raw media without data loss.

- **RR-003:** Edge nodes shall operate **autonomously offline**, collecting and storing data locally, and sync with the central server when connectivity is restored.

4.5.3. Performance

- **PR-001:** System should scale horizontally: additional processing workers can be added to increase throughput.
- **PR-002:** Support multi-tenant deployments (multiple parks) with logical separation

4.5.4. Maintainability and Supportability Requirements:

- **MS-001:** Modular codebase: separate inference, ingestion, database, and dashboard modules.
- **MS-002:** CI/CD pipelines for automated testing and deployment.
- **MS-003:** Model versioning and rollback capability.

4.5.5. Portability

- Use containerization for server components, provide scripts for edge deployment

4.5.6. Legal, Regulatory and Ethical Requirements

- **LE-001:** Comply with local data protection laws for capturing humans on camera.
- **LE-002:** Provide functionality to blur/redact humans in exports on demand.

Appendix A – Data Dictionary

Term / Data Item	Definition	Example Value(s)
Frame	A single image captured by the camera/edge device, usually in JPEG/PNG format, with timestamp metadata.	{image_bytes, ts=2025-10-05 14:23:00, camera_id=05}
Bounding Box (BBox)	A rectangular region around a detected object (animal/human/weapon). Stored as pixel coordinates.	{x_min=120, y_min=80, x_max=260, y_max=200}
Class Label	The category of the detected object as predicted by the YOLO model.	"Elephant", "Lion", "Human"
Confidence Score	The probability assigned by the YOLO model that the detection is correct (range 0–1 or percentage).	0.92 or 92%
Detection Event	A structured record generated after inference, containing object details, confidence, timestamp, and media reference.	{camera_id=05, ts, [detections], media_ref}
Alert Payload	A structured message generated when an alerting rule is triggered, sent to the alerting gateway.	{event_id, camera_id, alert_type="Poaching", ts, media_ref}
Model Version	Identifier for the YOLO model version used during inference (for traceability).	"YOLOv8_wildlife_v3.2"
Metadata Database	Database storing structured detection and alert records, including timestamps, camera IDs, and detection attributes.	PostgreSQL table row
Rule Engine	The logic in the Event Processor that decides whether an alert is raised (e.g., human detected with confidence >=80% in restricted area).	[if class=Human AND area='Restricted'] -> raise alert

Appendix B – Group Log

Date	Task Performed	Remarks/Outcomes
20/08/2025	Conducted Literature survey on wildlife monitoring and poaching detection.	Identified relevant research papers and already existing software.
24/08/2025	Started writing in the SRS Document according to the guidelines given in the document and the instructions given by the professor.	Completed Section 1 of the SRS Document.
31/08/2025	Discussed the functionality, characteristics and Constraints of the project.	Completed Section 2 of the SRS Document.
07/09/2025	Build the use case models, DFD and class diagrams of the project, discussed the requirements of the project.	Completed Section 3 and 4 of the SRS Document.
14/09/2025	Finalized the datasets	Downloaded and prepared datasets for training.
28/09/2025	Pre-processed datasets and trained YOLO model	Achieved ~85% mAP accuracy
05/10/2025	Integrated model with OpenCV for bounding box visualization in images/videos	Successfully detected 35 classes