School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

Human-Animal Conflict

Prepared by

Name	**SAP ID**	**Specialization**	**Batch**
Om Vats	500105231	DevOps	B1
Rakshan Sharma	500105046	DevOps	B5
Panchika Gupta	500101647	DevOps	B3
Kunal Sinha	500105636	DevOps	B5

Table of Contents

1. Introduction

1.1. Purpose of the Project

The purpose of this project is to develop an AI-powered Human-Animal Conflict (HAC) Prediction and Mitigation System using DevOps practices, cloud computing, and algorithmic modeling. The system will collect real-time data from IoT sensors, camera traps, and GPS collars to predict potential conflict zones and suggest mitigation strategies. It aims to bridge the gap between human settlements and wildlife conservation efforts by utilizing AI-driven analytics to prevent conflicts.

1.2. Target Beneficiary

Forest Departments & Wildlife Conservation Agencies: To prevent conflicts and protect endangered species.

Farmers & Local Communities: To reduce economic losses caused by animal intrusions.

Environmental Researchers & NGOs: To analyze patterns of wildlife movement and enhance conservation efforts.

Integrate stakeholder feedback to refine and improve predictive accuracy.

Offer preventive measures based on AI predictions to minimize human-wildlife conflicts.

1.3. Project Scope

Predict human-animal conflict zones using AI models.

Implement a real-time alert system for affected areas.

Use cloud-based storage and processing for scalability.

Provide a user-friendly dashboard for monitoring and analytics.

Support mobile and web-based access for stakeholders.

1.4. References

[1] Joshi, A., & Singh, P. (2018). GIS-based predictive modeling for human-animal conflict zones. *Wildlife Conservation Journal, 12*(3), 45-59.

[2] Kumar, R., Patel, S., & Sharma, D. (2021). Integrating satellite imagery and tracking data for conflict prediction. *Environmental Monitoring and Assessment, 189*(4), 102-118

[3] Sharma, V., & Rao, M. (2022). Deep learning applications in wildlife movement and human encroachment analysis. *International Journal of AI in Conservation, 9*(1), 23-38.

[4] Patel, N., Verma, S., & Rao, K. (2019). Cloud computing for real-time wildlife monitoring: A case study. *Cloud and IoT Journal, 7*(2), 54-69.

[5]Gupta, P., Mehta, R., & Roy, L. (2020). Machine learning approaches for predicting wildlife movement and conflict risk. *AI in Ecology Research, 6*(4), 78-94.

[6] Desai, H., Banerjee, A., & Choudhury, T. (2023). CNN-based wildlife species identification for conflict mitigation. *Journal of Applied Machine Learning, 15*(3), 112-130.

[7] Nair, K., & Sharma, V. (2019). DevOps for efficient AI model deployment in conservation efforts. *Computational Sustainability Review, 5*(1), 29-45.

[8] Banerjee, S., & Choudhury, P. (2021). Scalable wildlife data processing using containerization and cloud platforms. *Journal of Cloud Computing in Ecology, 8*(2), 37-52.

[9] State of Tigers, Co-predators & Prey in India (2022)

Wildlife Conservation Reports

IoT-based monitoring systems

Machine Learning models for prediction

DevOps methodologies for continuous development

Remote sensing and GIS-based wildlife monitoring techniques

Government policies on wildlife conservation and human-animal conflict mitigation

Research papers on AI-driven geospatial analysis for wildlife management

2. Product Description

2.1 Reference Algorithm

Random Forest Classifier for prediction based on past incidents.

LSTM (Long Short-Term Memory) for time-series analysis of movement patterns.

Geospatial Clustering (DBSCAN/K-Means) for hotspot detection.

Decision Trees for real-time alert generation based on risk factors.

2.2 Data/Data Structure

Input Data: GPS coordinates, weather data, past conflict records, sensor data, local reports.

Data Sources: IoT sensors, satellite imagery, government records, citizen reports.

Storage: Cloud-based databases (AWS RDS, Firebase, MongoDB, PostgreSQL).

Processing: Big data pipelines for handling large datasets in real-time.

2.3 SWOT Analysis

Strengths:

- Real-time analysis, cloud scalability, automated alerts, robust prediction models
- Scalable cloud architecture.
- AI-powered, real-time alerting system

Weaknesses:

- Requires high-quality sensor data, potential false alarms, high computational demand
- High computational power required

- Dependence on continuous sensor data

Opportunities:

- Integration with government systems, expansion to other conservation projects, global wildlife monitoring
- Collaboration with government agencies
- Expansion to new wildlife conservation projects

Threats:

- Sensor failures, connectivity issues in remote areas, resistance to technology adoption.
- Connectivity issues in remote areas
- Sensor failures affecting accuracy

2.4 Project Features

- Real-time data collection: from IoT devices and GPS trackers.Real-time Processing:
- AI based prediction : of high risk zone
- Automated alerts: via SMS, email, and mobile app notifications.
- Cloud based dashboard: with geospatial data visualization.
- Role-based access control: for different stake holders
- CI/CD pipeline: for continuous updates and improvements.
- Data logging: for trained analysis and future training of AI models.

2.5 User Classes and Characteristics

- Admin (Forest Officers): Full access to system controls, alert management, and data insights.
- Researchers: Access to analytical data, AI model results, and reports.
- Local Authorities & Communities: Receive alerts, submit conflict reports, access mitigation guidelines.
- Interaction with the System: Receives high-priority notifications on potential conflicts.

2.6 Design and Implementation Constraints

1. Hardware Requirements: IoT sensors, GPS trackers, cloud infrastructure, edge computing devices.

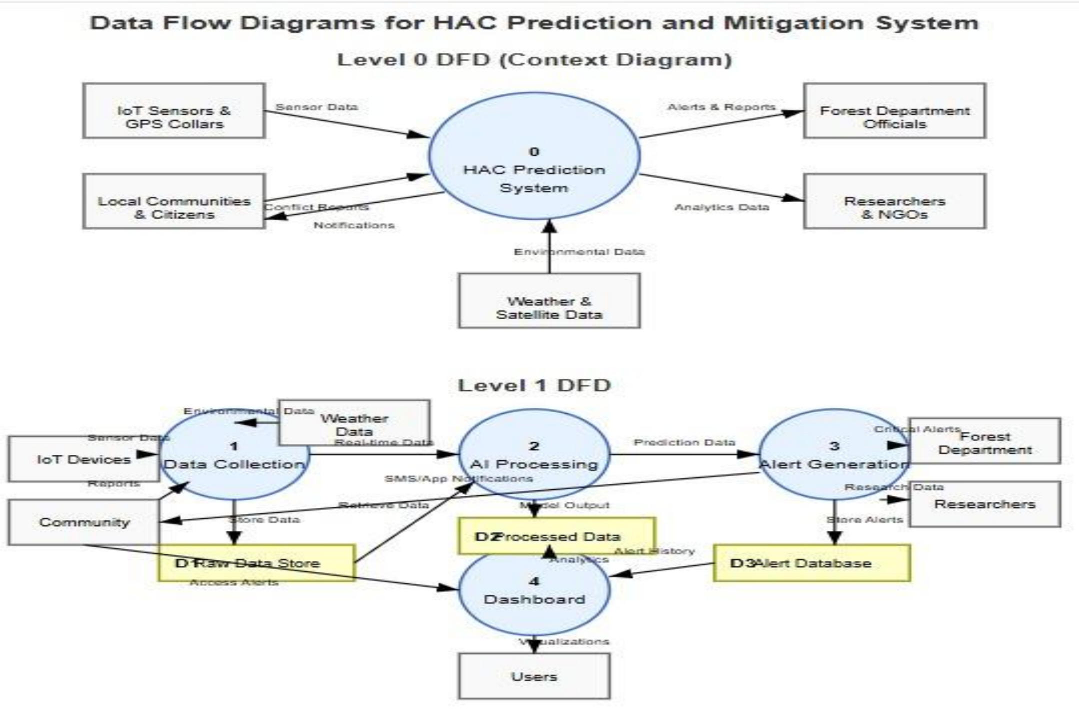
2. Software Requirements: Python, TensorFlow, Flask/Node.js for backend, React.js for frontend.

3. Regulatory Constraints: Compliance with wildlife conservation policies, data privacy regulations

4. Connectivity Constraints: Low-latency communication for real-time data processing.

2.7 Design Diagrams

Data Flow Diagram (DFD):



2.8 Assumptions and Dependencies

- Availability of reliable sensor data and proper calibration.
- Stable internet connectivity for cloud integration.
- Cooperation from local authorities and stakeholders for data collection.
- Cloud service provider for hosting backend infrastructure.

3. System Requirements

3.1 User Interface

- Web dashboard: Built with React.js, includes analytics, geospatial maps, and real-time alerts
- Mobile application: Provides notifications, simplified monitoring, and reporting features.

3.2 Software Interface

- Backend: Flask (Python) or Express.js (Node.js) for API development.
- Cloud storage: AWS S3, Firebase, MongoDB Atlas for data management.
- Machine learning frameworks: TensorFlow, PyTorch, Scikit-learn for AI model training and inference.
- Data processing: Apache Kafka or AWS Lambda for streaming and event-driven architecture.

3.3 Database Interface

- Relational database: PostgreSQL/MySQL for structured data and transactional logging.
- NOSQL database: MongoDB/Firebase for handling unstructured data like event logs, alerts.
- Geospatial database: PostgreSQL with PostGIS for analyzing location-based trends.

3.4 Protocols

1.Data Transmission: MQTT for IoT devices, RESTful APIs for communication between backend and frontend.

2.Security: OAuth2 for user authentication, AES encryption for sensitive data, HTTPS for secure data exchange.

3.Cloud Communication: AWS IoT Core or Google Cloud Pub/Sub for real-time event streaming.

4. Database Communication Protocols: These protocols manage secure and efficient interactions with databases

5. Remote Access and Device Communication Protocols: Used for managing remote devices and ensuring reliable connectivity

6. Alert and Notification Protocols: These protocols handle real-time user notifications

3.5 Security Considerations

1. User Authentication & Authorization: Multi-factor authentication (MFA) for user access control

2. Regular Security Audits: Periodic security assessments to detect vulnerabilities

3. Backup & Disaster Recovery: Automated cloud backups and disaster recovery mechanisms

3.6 Performance Requirements

1. Scalability: It should support an increasing number of users and sensor devices without performance degradation

2. Uptime: 99.9% availability with failover mechanisms

3. Data Throughput: Must handle at least 10,000 data points per second without delays

4. Latency: Ensure alerts are generated within 2 seconds of detecting an anomaly

3.7 Future Enhancements

1. Voice-based Alerts: AI-powered voice assistants to notify local authorities about detected risks

2. Self-Learning AI Models: Implement reinforcement learning to improve prediction accuracy over time

Mentor’s Sign