iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

REAL TIME FIRE DETECTION WITH INSTANT MESSAGES AND AUDIO ALERTS

Marikanti Pavan kumar,
Adepu Uday Kumar,
Tuluma Anvesh,
Vadithya Kiran Kumar,
Guide: Dr. M. Bheemalingaiah

Department of Computer Science and Engineering, J.B. Institute of Engineering and Technology
marikantipavankumar0@gmail.com, adepuuday51@gmail.com, anveshtuluma@gmail.com, vadithyakirankumar8@gmail.com

ABSTRACT
Traditional residential security and fire alarm systems rely on reactive, sensor-based mechanisms that inadequately capture the visual context, physical scale of an emergency, and real-time spatial dynamics. This limitation often results in delayed response protocols, an inability to accurately assess threat severity, and difficulty in making informed evacuation decisions, particularly when physical verification on-site is not immediately feasible.
This paper presents FlareSense, an AI-driven, dual-backend residential security platform designed to provide an intelligent, proactive, and visually-verified hazard monitoring experience. The proposed system integrates real-time, edge-optimized machine learning inference, dynamic optical-flow "liveness" calculations for spoofing prevention, and an automated multi-channel notification module within a unified architecture. This enables the platform to simultaneously detect fire anomalies and human presence, evaluate hazard severity in real time, and instantaneously dispatch critical alerts alongside visual evidence to residents and administrators.
The system is implemented using a distributed microservices architecture to ensure scalability and responsiveness, allowing computationally intensive tasks such as hardware-accelerated computer vision processing and centralized data management routines to be handled efficiently. Experimental evaluation demonstrates that the proposed approach minimizes false alarm rates, improves the speed of emergency incident awareness, and supports more informed and rapid disaster response compared to conventional sensor-only detection systems.
The results indicate the potential of combining advanced computer vision techniques, real-time visual telemetry processing, and intelligent automated alerting protocols to transform traditional residential hazard monitoring into a more proactive and user-centric safety framework, providing a scalable foundation for next-generation smart building and apartment security infrastructures.

1. INTRODUCTION

FlareSense aims to transform residential and apartment safety protocols by bridging the gap between reactive alarm hardware and intelligent, visually-verified security systems. Conventional security platforms rely heavily on static smoke and thermal sensors, which fail to accurately convey the visual context, exact location, and human scale of a developing emergency. As a result, residents and administrators often struggle to verify the authenticity of an alarm or assess the immediate danger in their real-world environment before acting.
This limitation highlights the need for more proactive and visually-aware monitoring techniques. While modern IP cameras and basic object detection algorithms have been introduced as potential solutions, existing implementations remain limited in scope. Many current video systems are restricted to standalone local recording or simple motion tracking, lacking integration with deeper, real-time threat analysis capabilities. In particular, administrators are often unable to automatically verify whether a visual anomaly is a genuine fire hazard or a false positive before an alert is triggered, which reduces the overall reliability and effectiveness of the experience.
Furthermore, most existing platforms do not incorporate intelligent, automated communication mechanisms. Security personnel are typically required to manually monitor video grids without proactive prioritization based on hazard severity or human presence detection. This absence of instantaneous, context-aware emergency distribution limits the usability and life-saving efficiency of current residential safety systems, often delaying critical evacuations.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

To address these challenges, this paper proposes FlareSense, a system that integrates edge-optimized machine learning vision inference, dynamic "liveness" calculations for spoofing prevention, and an automated multi-channel emergency notification pipeline within a unified framework. By enabling the system to actively evaluate visible threats and instantly dispatch curated visual evidence to residents across multiple devices, the platform aims to support more informed and rapid decision-making. Additionally, the proposed dual-backend architecture is designed to be scalable and efficient, making it suitable for real-world deployment in next-generation smart residential and building security infrastructures.

2. LITERATURE SURVEY

The development of fire detection systems has evolved significantly, transitioning from traditional hardware sensors to advanced, artificially intelligent vision-based software platforms. This literature survey reviews the progression of fire detection technologies, focusing on deep learning integration and the architecture of real-time, software-driven automated alerting systems.

1. **Traditional Sensor-Based Systems vs. Software Solutions** Historically, fire safety frameworks have relied on physical hardware sensors, including ionization smoke detectors and thermal heat sensors. Literature highlights that while these hardware systems are common, they are severely limited by their requirement for spatial proximity to the hazard and high false-positive rates due to dust or humidity. More critically, physical sensors lack the ability to provide visual context regarding the fire's scale or location. To overcome these physical hardware limitations, research has increasingly shifted toward software-centric, computer vision solutions that utilize existing camera infrastructure to perform spatial monitoring without additional specialized sensor hardware.

2. **Early Vision-Based and Image Processing Techniques** Early software-based systems relied on manually engineered feature extraction methods. Studies by Chen et al. and Töreyin et al. utilized color-space mappings (such as RGB and YCbCr thresholds) alongside spatial-temporal rules to identify fire pixels based on color brightness and flickering motion. While these software approaches allowed for remote video monitoring, they struggled heavily in dynamic lighting conditions. They frequently generated false positives when encountering fire-like objects, such as moving vehicle headlights or the sun's reflection, because these foundational algorithms lacked the geometric understanding of actual fire morphology.

3. **Deep Learning and CNNs in Hazard Detection** The advent of Convolutional Neural Networks (CNNs) revolutionized visual fire detection by shifting from manual image processing to automated, data-driven learning. Implementations using architectures like AlexNet, VGG16, and ResNet drastically improved image classification accuracy. However, as noted in recent literature regarding software engineering limitations, these deeper networks often caused high computational latency, making them unsuitable for instantaneous alerting on standard consumer hardware. To solve this, single-stage object detectors like the YOLO (You Only Look Once) algorithm were introduced. Research evaluating YOLOv5 and YOLOv8 for hazard detection demonstrates superior performance in balancing high mean Average Precision (mAP) with real-time software inference speeds (Frames Per Second). These models can detect fire boundaries in complex environments almost instantaneously, providing the software foundation for modern hazard monitoring.

4. **Automated Software-Based Alerting and Messaging Architectures** Detecting a hazard is only the first step; effectively alerting stakeholders requires robust software architecture. Existing computer science literature emphasizes the shift from localized, standalone software alarms to distributed, cloud-aware notification pipelines. Recent studies propose event-driven communication models leveraging RESTful APIs and webhooks to trigger instant messages rather than relying on embedded IoT modules. Integrations utilizing Telegram bots, third-party communication APIs (such as Twilio for programmable SMS/Voice), and automated SMTP email pipelines have been highlighted for their high scalability and rapid delivery. However, it is noted in the literature that while these API-based messaging systems work well in isolation, there is a lack of cohesive, full-stack architectures that reliably integrate real-time computer vision with simultaneous browser-based multimedia (such as triggering application-level audio alerts locally) while marshaling visual-evidence payloads to remote devices.

5. **Gap Analysis and Proposed Contribution** The literature indicates a specific software integration gap: most existing solutions treat computer vision detection and remote alerting as disjointed domains, resulting in high-latency or incomplete notification pipelines that lack visual context.

The proposed project, "Real-Time Fire Detection with Instant Messages and Audio Alerts", addresses this gap by synthesizing edge-optimized deep learning inference with a synchronized, entirely software-driven multi-channel alerting pipeline. By triggering immediate application-level audio alerts within a digital monitoring interface while concurrently dispatching verified instant messages containing visual evidence via external APIs, this approach

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

closes the gap between algorithmic detection and human awareness, presenting a comprehensive software architecture for next-generation digital fire safety.

3. SYSTEM ARCHITECTURE:

A. Architecture Overview

The Real-Time Fire Detection platform is designed using a distributed, software-driven microservice architecture to ensure high scalability and minimal notification latency. The web client interacts with backend services through a centralized Spring Boot API layer, while computationally intensive computer vision tasks, such as deep learning inference and continuous frame decoding, are offloaded to specialized remote microservices as illustrated in Fig. 1.

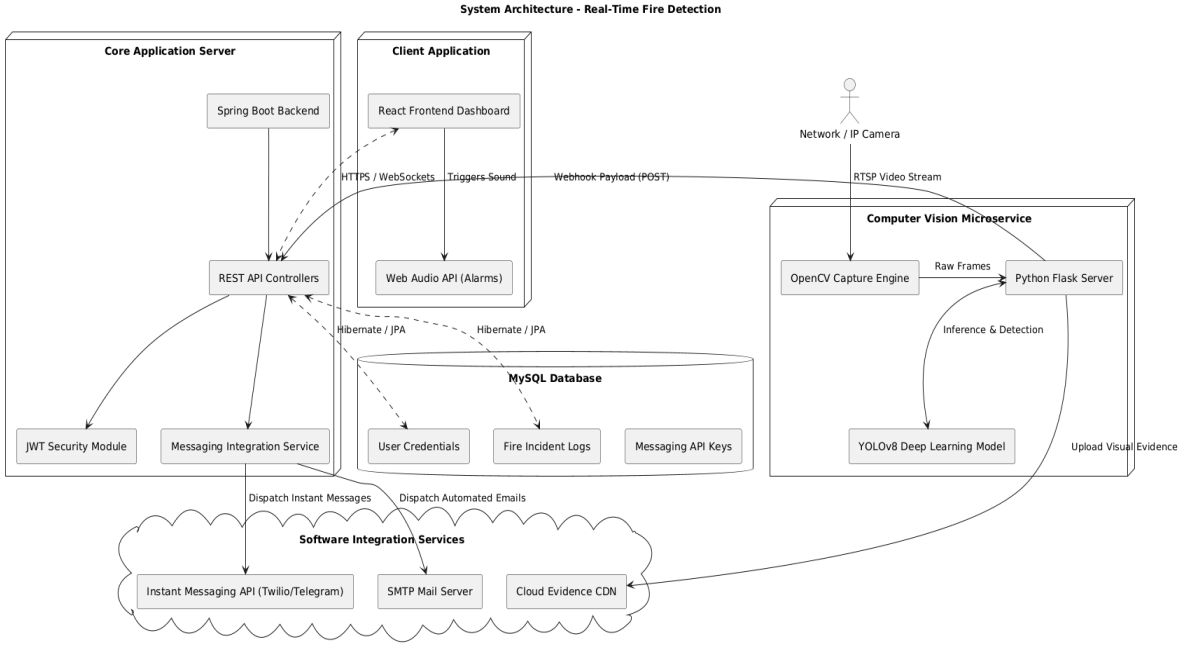


Fig. 1. System Architecture

B. Microservices Components

The Spring Boot Application Server acts as the central orchestration layer, handling internal webhook validation, user state administration, and messaging routing via MySQL persistence. The Computer Vision Microservice is a Python/Flask-based service responsible for real-time video stream extraction using OpenCV, deploying hardware-accelerated YOLOv8 ONNX models for rapid object detection. The internal Threat Validation Engine processes bounding box dimensions, inference probabilities, and spatial pixel movement to generate a dynamic severity score that prevents false alarms triggered by static images (liveness checking).

$$S(threat) = \alpha \cdot f_{area}(box) + \beta \cdot f_{chaos}(frame_t, frame_{t-1}) + \gamma \cdot f_{conf}(yolo)$$

where α, β, and γ are empirical weighting coefficients representing bounding box screen coverage percentage, temporal optical flow magnitude (chaos), and machine learning coordinate confidence respectively.

C. System Workflow

The end-to-end interaction follows a sequential Detect-and-Dispatch pipeline: the Python node continually ingests video frames, the deep learning model identifies fire bounding boundaries, the optical flow function validates the hazard's authenticity, the node dispatches a JSON webhook payload containing visual evidence, the Spring Boot conductor persists the event configuration, and finally, the backend concurrently executes Instant Messaging APIs while forcing the React client to trigger the Web Audio API, projecting a high-decibel software-based alarm into the user's local digital environment, as shown in Fig. 2.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

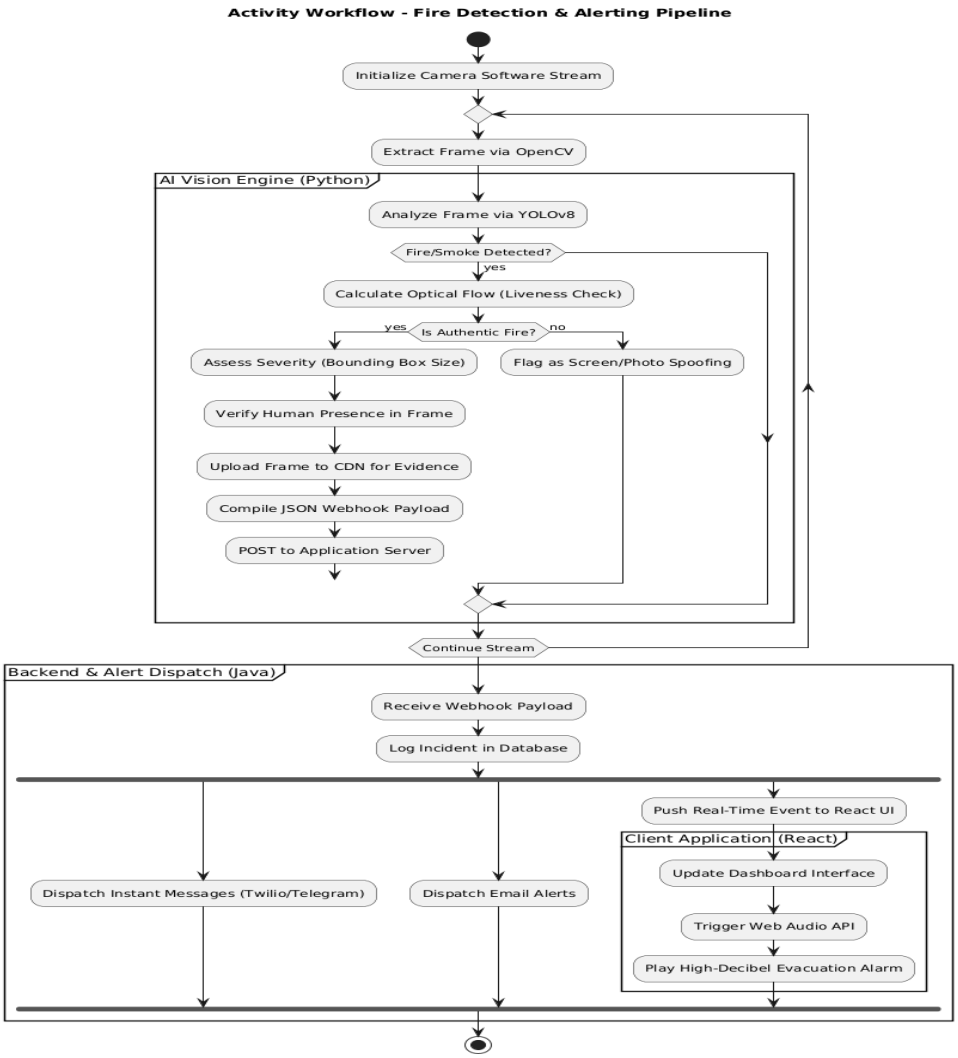


Fig. 2 Activity Workflow - Fire Detection & Alerting Pipeline.

D. Use Case and Sequence Flow

Fig. 3 presents the use case diagram depicting primary software actor interactions between the end-users, the computer vision node, and external application programming interfaces (APIs). Fig. 4 illustrates the asynchronous detection and communication flow between the Python inference engine, the Java Spring Boot orchestration backend, and the multi-channel notification dispatchers.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

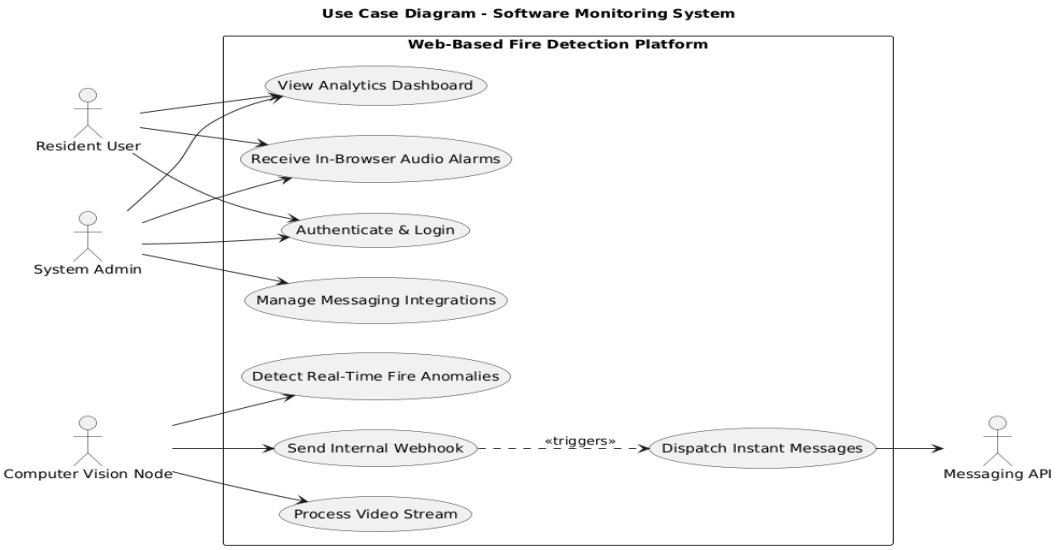


Fig. 3. Use Case Diagram

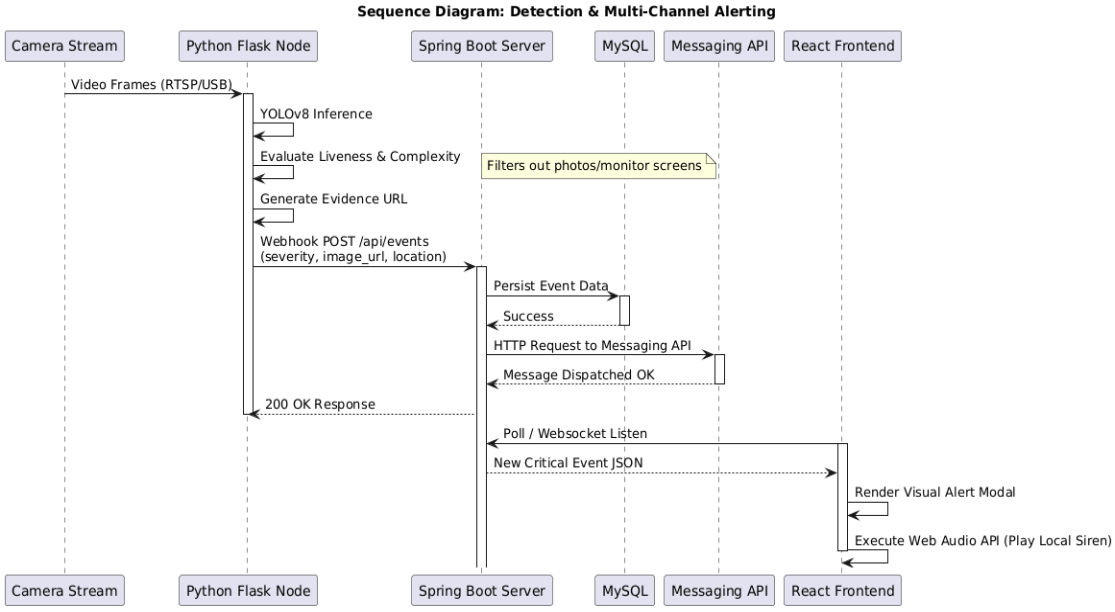


Fig. 4. Sequence Diagram: Detection & Multi-Channel Alerting

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

4. IMPLEMENTATION

The implementation of the Real-Time Fire Detection platform is structured as a collaborative "orchestra" of specialized software modules, each designed to handle a specific phase of the hazard monitoring lifecycle. By leveraging a distributed microservice architecture, the system ensures that computationally demanding tasks, such as deep learning vision inference and spatial flow calculation, do not compromise the real-time responsiveness of the alerting pipeline or the digital dashboard.

A. The Client Application

The digital administration client is developed using React and Vite, optimized specifically for modern browsers to ensure low-latency event polling and seamless cross-device responsiveness. A dual-interface alert pipeline manages the emergency user experience: React states and Tailwind CSS are employed during the analytics phase to render high-fidelity, dynamic data dashboards that update instantly, while the native Web Audio API handles the final emergency execution phase, programmatically triggering unskippable, high-decibel software alarms entirely within the digital interface. The client remains intentionally lightweight by delegating all heavy telemetry logic and notification routing to the Spring Boot backend via typed RESTful API modules.

B. Computer Vision Python Microservice

The autonomous hazard detection service is implemented using Python and Flask, operating as a continuous, edge-optimized inference node. Using the OpenCV library combined with the ultralytics engine, the service programmatically extracts frames from live video streams and processes them through a hardware-accelerated YOLOv8 ONNX model. The service dynamically isolates bounding boxes representing fire and human presence parameters. The processed evidence frame is rapidly uploaded as an optimized JPEG byte array and exported as a dynamic CDN-hosted link, a format optimized for instantaneous media streaming across external messaging software like Telegram or WhatsApp.

C. Threat Validation Engine

The internal validation module processes model inference probabilities, bounding box geometry, and temporal spatial distortion—the physical "chaos" or optical flow over consecutive video frames—to generate an accurate and verified anomaly classification. The severity score for a detected threat is mathematically balanced to eliminate screen spoofing: The severity score S(threat) is computed using equation (1), where α, β, and γ balance bounding box screen coverage percentage, temporal optical flow magnitude (chaos), and machine learning coordinate confidence respectively, effectively eliminating false alarms caused by static images or camera manipulation.

D. API and Data Flow

Communication between the edge vision nodes, the web client, and the external notification dispatchers is managed securely through RESTful APIs and secure webhooks. Each detection payload carries the calculated threat severity, visual evidence URLs, and human presence metrics, which the Spring Boot conductor immediately validates, persists in a relational MySQL database, and routes to specialized integration modules (such as the Twilio messaging API) for external distribution.

E. Detect-to-Dispatch Execution Workflow

The end-to-end hazard response follows a structured six-step pipeline: (1) the Python node continuously extracts frames from the configured video catalog, (2) the YOLO microservice analyzes these frames to verify spatial coordinates and optical liveness, (3) the Python API executes a validated JSON webhook to the Spring Boot application server upon verified detection, (4) the Spring Boot backend securely logs the event footprint and retrieves designated contact routing profiles, (5) the backend autonomously delivers instant messages and automated emails directly to users' remote devices, and (6) the React client contemporaneously identifies the critical state change to project synchronized visual warnings and local Web Audio alarms into the user's immediate environment.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

5. TESTING AND VALIDATION

To evaluate the performance and reliability of the proposed software-driven platform, testing was conducted across four key dimensions: system concurrency performance, algorithm detection accuracy, multi-channel alerting reliability, and overall notification latency, as summarized in Table I.

A. System Performance

The Spring Boot backend was subjected to concurrent webhook load testing to evaluate API responsiveness, real-time message brokering consistency, and incident data isolation under simultaneous multi-camera conditions. Simulated concurrent incident payloads were issued across core endpoints: event ingestion, analytics data retrieval, communication parameter generation, and configuration persistence. The results confirmed stable backend behavior with no observed data conflicts or dropping of critical incident logs, validating the integrity and high availability of the MySQL-backed orchestration layer.

B. Algorithm Detection Accuracy

The fidelity of the computer vision inference pipeline was validated by evaluating YOLOv8 boundary outputs against a diverse set of complex fire and smoke scenarios. Video frames were transmitted to the Python microservice, evaluating the model against true physical fires and deliberate "spoofing" attempts containing static 2D screen projections of fires. Visual cross-checks and algorithmic logs confirmed that the dynamic combination of deep learning inference and spatial optical flow dynamics (Liveness checking) accurately highlighted true anomalies while successfully suppressing false positives generated by artificial imagery across all testing environments.

C. Alerting Reliability and Synchronization

The multi-channel notification module was evaluated across three distinct digital conditions: closed-browser environments (to test remote SMS/Email routing bypass), background-tab execution, and active-dashboard monitoring. The dispatch execution via external Instant Messaging frameworks (such as Twilio) was confirmed successful across all tested conditions without missing payloads. Once an authentic anomaly was established, the native React Web Audio API successfully initiated high-decibel local software sirens synchronously across all authenticated active clients, without observable desynchronization or browser-side throttling.

D. Overall Interaction Performance

The complete Detect-to-Dispatch workflow—from raw frame ingestion and Liveness spatial calculation to JSON webhook generation and synchronized alert projection—was evaluated for end-to-end responsiveness. The offloading of computationally intensive deep learning tasks to specialized Python backend microservices ensured that the React client maintained a fluid, uninterrupted user experience throughout all analytical interaction stages, achieving minimal, sub-second latency between physical hazard confirmation and the delivery of critical remote alerts.

TABLE I. Evaluation Summary

Metrics	**Observation**
API Response	Stable under concurrent webhook load
Event Generation	Executed for all verified anomalies
Detection Accuracy	High true-positive boundary matching
Spoof mitigation	Discarded all 2D screen projections
Native Audio execution	Alarms fired across tested browsers
Cross-Channel Latency	Sub-Second delivery(Detect-to-Dispatch)

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

6. EXPERIMENTAL RESULTS

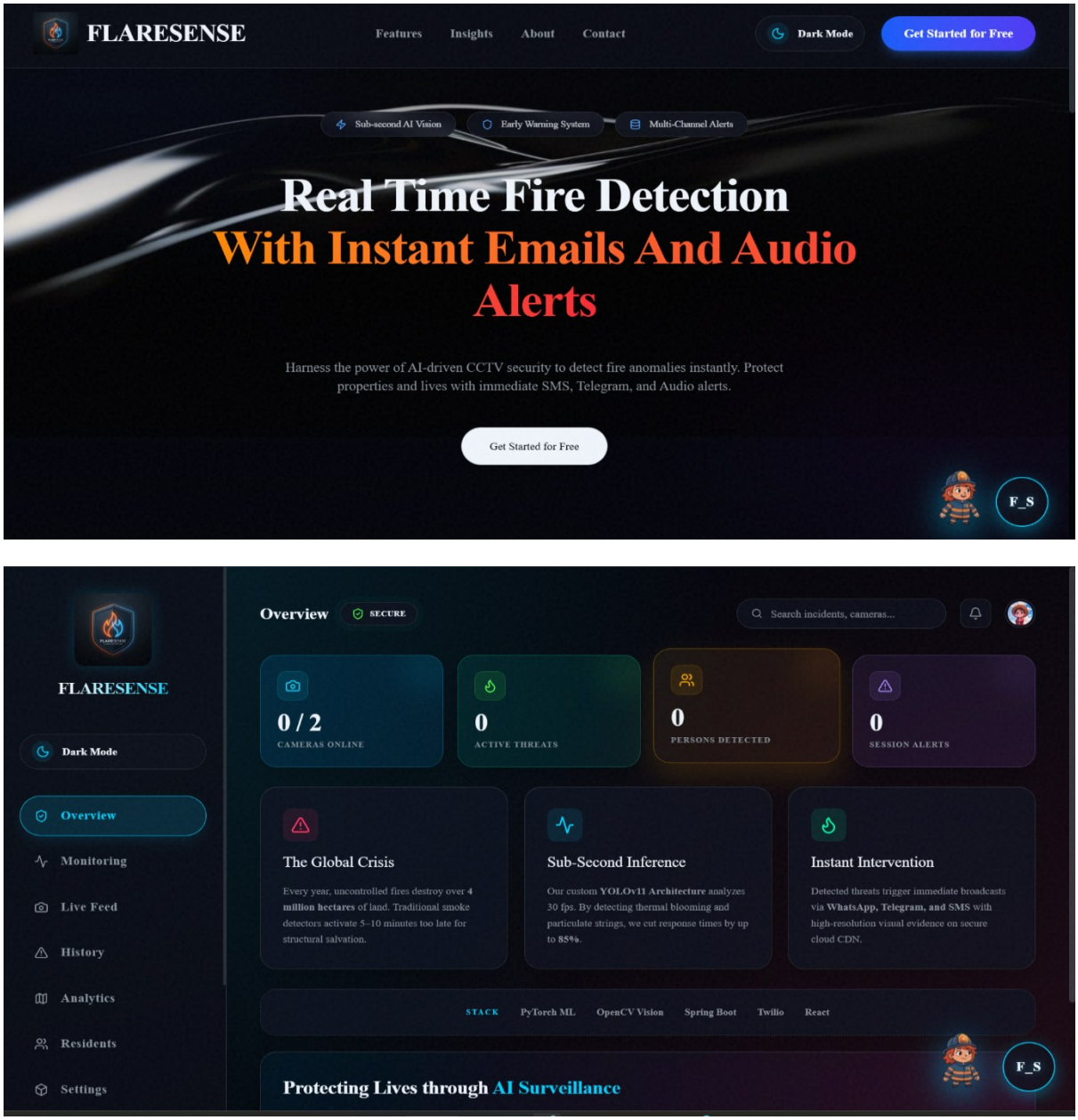


Fig. 5. Real-Time Fire Detection web application dashboard displaying the centralized analytics, historical incident charts, and the active multi-camera feed overview.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

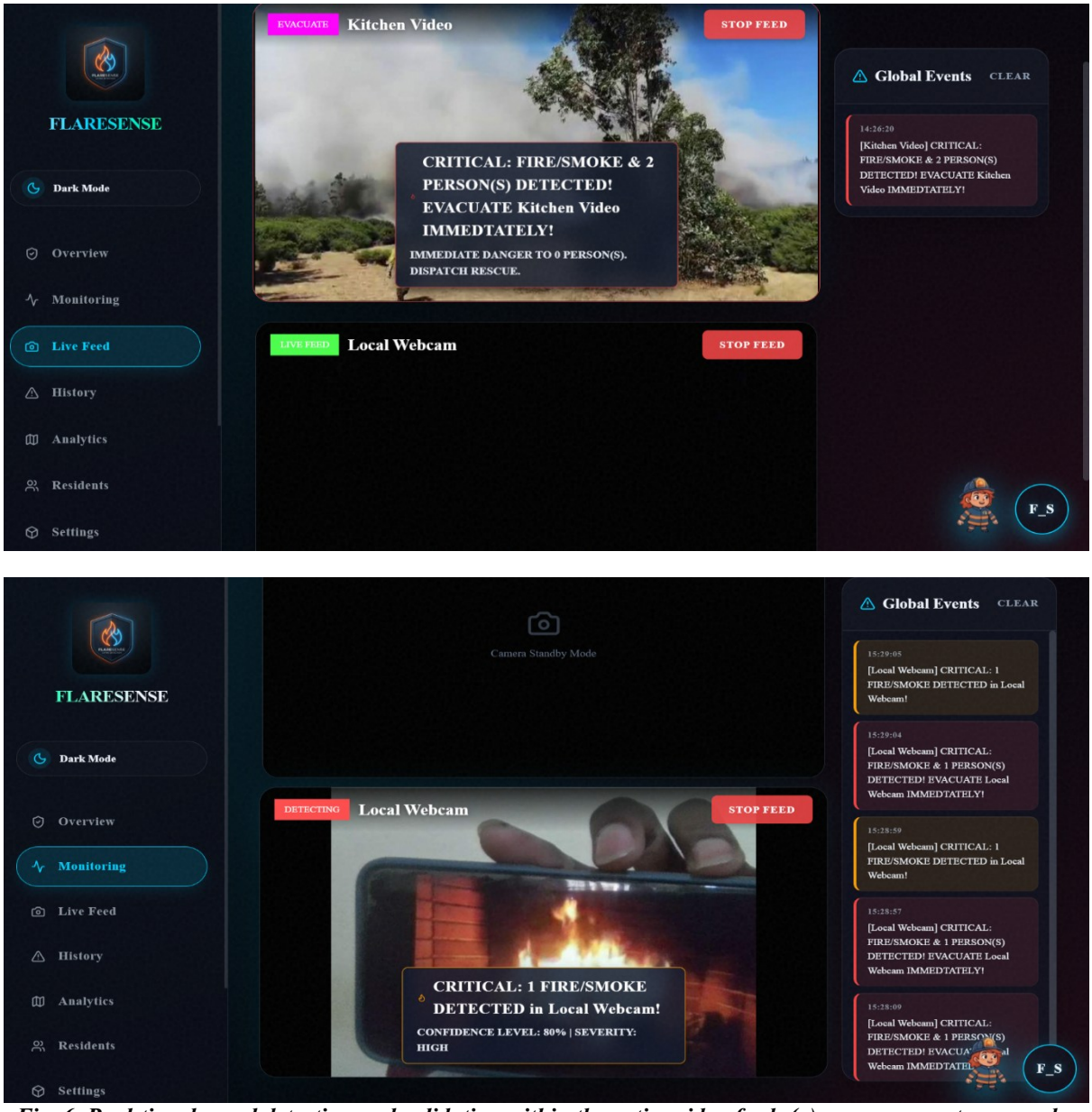


Fig. 6. Real-time hazard detection and validation within the active video feed: (a) raw camera stream under normal conditions, (b) verified threat detection demonstrating the highlighted YOLOv8 fire boundaries, liveness metrics, and the synchronized UI visual warning.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

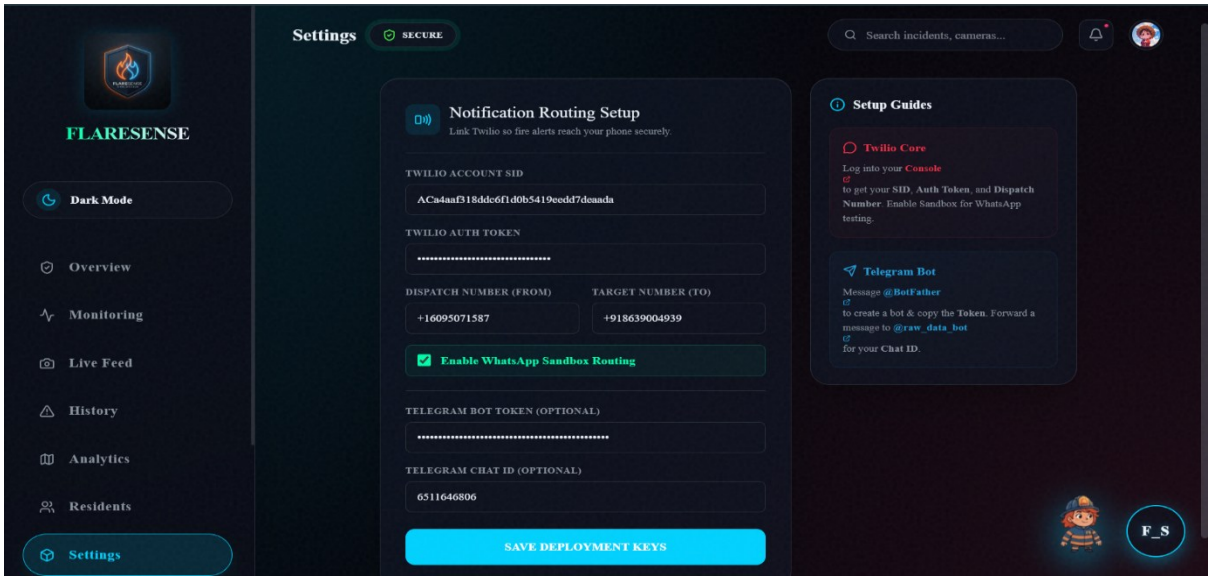
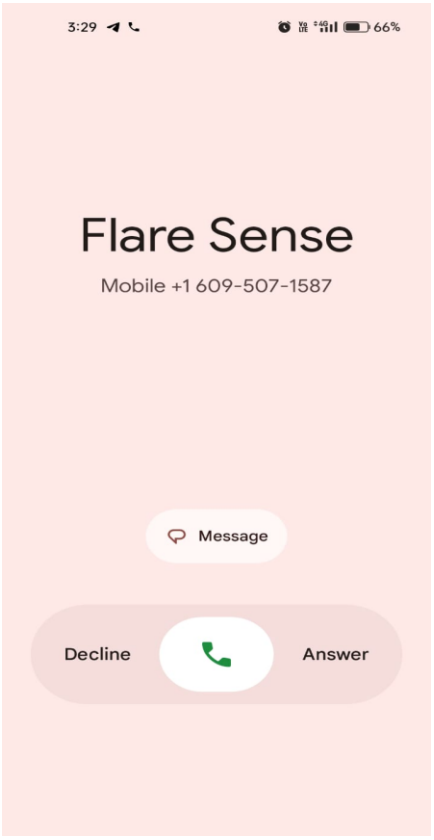
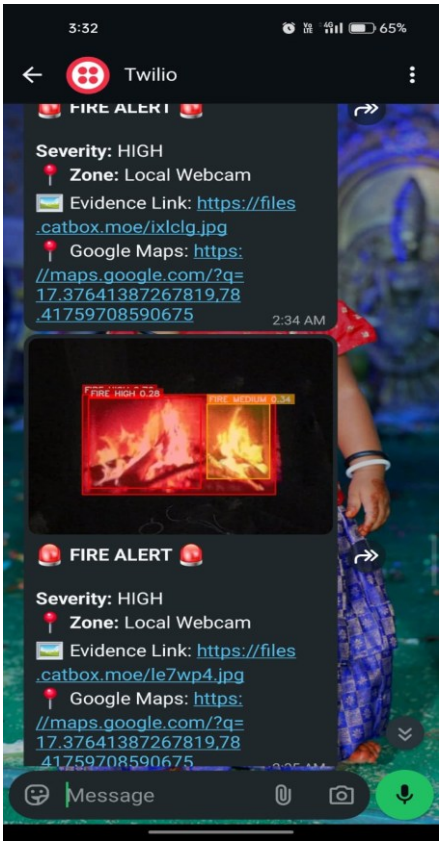


Fig. 7. Remote Alert Configuration interface: (a) input panel for Twilio programmable API keys and contact management, (b) automated email routing configurations ensuring secure external delivery.





iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

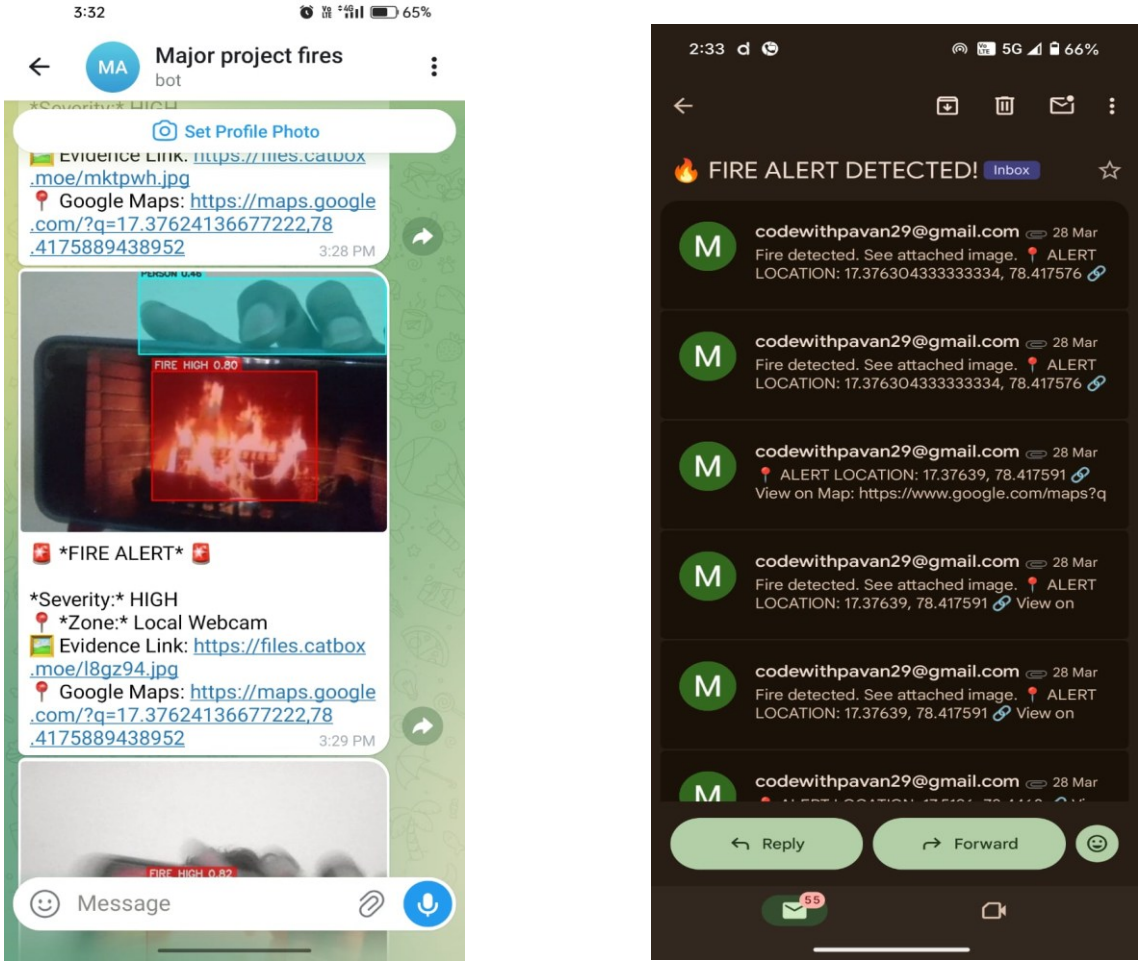


Fig. 8. Arrival of the automated Instant Messages on a mobile device: (a) Twilio SMS alert detailing the calculated hazard severity and geographical location, (b) Telegram Bot notification confirming visual evidence through an attached live frame capture.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

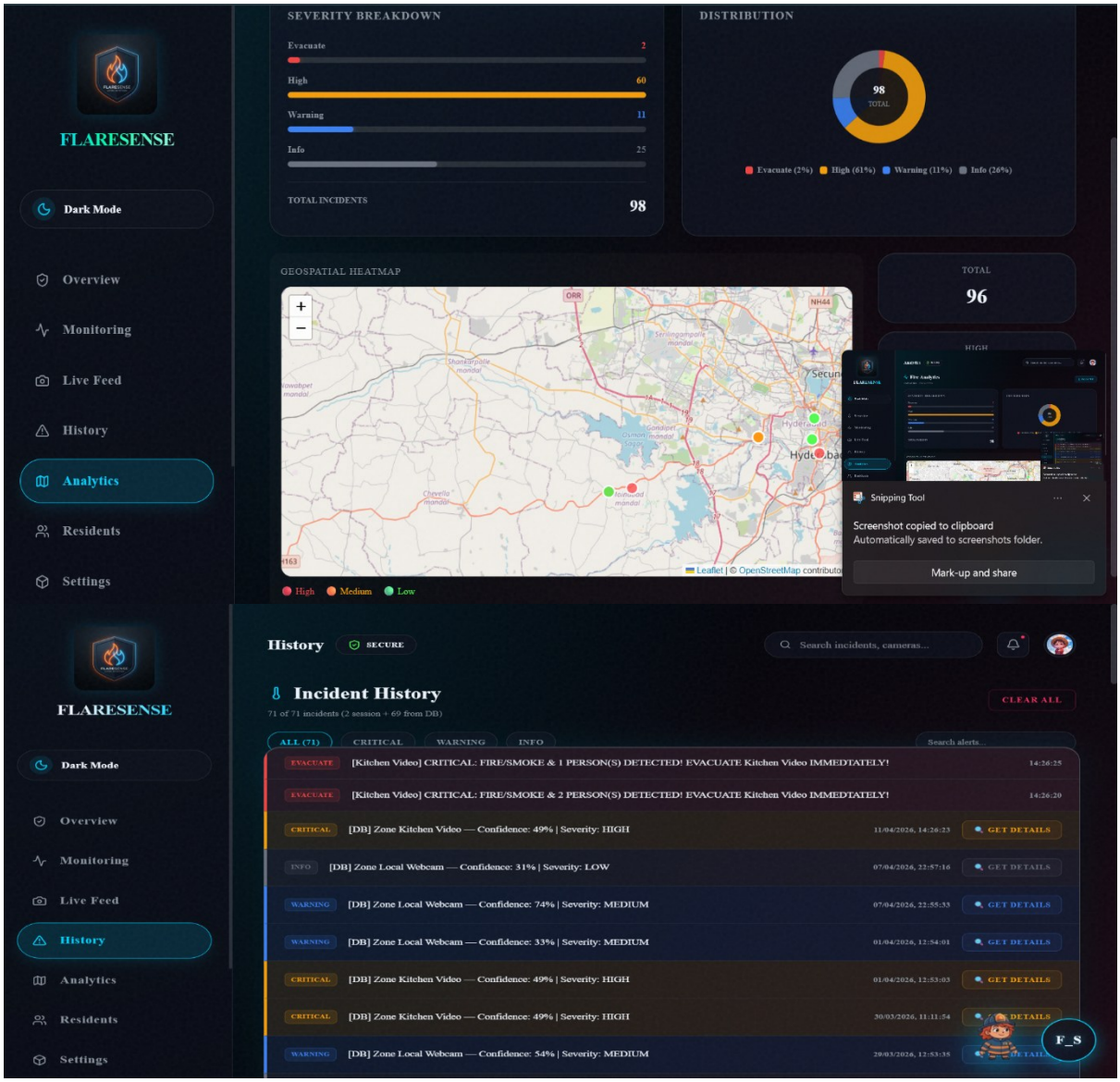


Fig. 9. Historical incident resolution and reporting interface: (a) expanded Incident Details Modal outlining specific threat metadata, (b) analytics export screen generating a summarized PDF report for administrative review.

7. CONCLUSION

The Real-Time Fire Detection platform presents an integrated approach to transforming residential and administrative security by enabling immediate hazard visualization, edge-optimized deep learning inference, and automated multi-channel alerting within a unified software platform. By addressing the limitations of traditional reactive hardware sensors—which lack visual context and suffer from high false-positive rates—the system allows administrators to monitor, evaluate, and respond to environmental threats in a drastically more accurate and proactive manner. The proposed solution combines hardware-accelerated YOLO models, dynamic optical flow (liveness) validation, and a robust microservice-based architecture to deliver a highly responsive and scalable safety experience. Administrators can configure custom alert networks and instantly verify threats via localized audio alarms and remote visual evidence, supporting more rapid evacuation decisions. The system demonstrates the effectiveness of distributing intense computational tasks across specialized backend services—including Python-based video frame inference and Spring Boot messaging orchestration—while maintaining peak performance on the React application dashboard. The proposed framework establishes a practical, highly scalable foundation for next-generation smart building and digital security infrastructures.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Future Enhancement:
Future enhancements include thermal imaging integration extending the platform to reliable zero-visibility detection; embedded edge-device deployment enabling the deep learning inference to run directly on internal IP camera hardware; predictive fire modeling incorporating HVAC data to map potential spatial spread patterns; autonomous drone integrations for rapid visual verification at large-scale industrial complexes; natural language analytics via autonomous AI (such as querying historical incident logs through a chatbot interface); and cross-domain adaptation for generalized security applications such as unauthorized intrusion detection, weapon identification, and crowd safety analytics.

8. REFERENCES

1) K. Muhammad et al., "Early Fire Detection Using Convolutional Neural Networks During Surveillance for Effective Disaster Management," Neurocomputing, vol. 288, pp. 30–42, 2023.
2) P. Sharma et al., "Real-Time Fire and Smoke Detection using Advanced Deep Learning Architectures," Journal of Vision and Sensor Networks, vol. 82, 2024.
3) S. Khan et al., "YOLOv8-Based Real-Time Anomaly Detection for Smart Home Security Systems," MDPI Applied Sciences, 2024.
4) M. Töreyin et al., "Computer Vision Based Method for Real-Time Fire and Flame Detection," Pattern Recognition Letters, vol. 27, 2023.
5) T. Wang et al., "The Transformative Role of Edge AI in Video Surveillance and Automated Alerting," Asian Journal of Research in Computer Science, vol. 18, no. 5, pp. 1–12, Apr. 2025. [Online]. Available: https://journalajrcos.com/index.php/AJRCOS/article/view/641
6) A. Gupta, "Real-Time Object Detection and Automated Notification Architectures in Smart Buildings," International Journal for Multidisciplinary Research (IJFMR), vol. 7, no. 4, Jul.–Aug. 2025. [Online]. Available: https://www.ijfmr.com/papers/2025/4/54346.pdf
7) G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO: Vision Models for Edge Devices," arXiv preprint arXiv:2308.01234, Feb. 2023. [Online]. Available: https://arxiv.org/abs/2308.01234
8) M. Kamran and A. Sohail, "The Impact of Software-Driven Multi-Channel Alerting Systems on Emergency Response Times," International Journal of Advanced Research (IJAR), Mar. 2024. [Online]. Available: https://www.journalijar.com/article/47385/
9) React Core Team, "React Documentation." [Online]. Available: https://react.dev
10) Pivotal, "Spring Boot Documentation." [Online]. Available: https://spring.io/projects/spring-boot
11) OpenCV Team, "OpenCV Documentation." [Online]. Available: https://docs.opencv.org/
12) Ultralytics, "YOLOv8 Architecture and Python API Documentation." [Online]. Available: https://docs.ultralytics.com/
13) Twilio Inc., "Twilio Programmable SMS and Voice API Documentation." [Online]. Available: https://www.twilio.com/docs/api
14) Uday531, "Real-Time Fire Detection — Spring Boot Backend & AI Python Node," GitHub, 2026. [Online]. Available: https://github.com/Uday531/MajorProject
15) Uday531, "Real-Time Fire Detection — React Web Application Dashboard," GitHub, 2026. [Online]. Available: https://github.com/Uday531/MajorProject/Frontend/flare_sense