VISION-BASED FIRE DETECTION AND PREVENTION SYSTEM FOR WAREHOUSE SAFETY USING REAL-TIME CAMERA SURVEILLANCE

P.A.S.Tharana

(IT21822094)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

VISION-BASED FIRE DETECTION AND PREVENTION SYSTEM FOR WAREHOUSE SAFETY USING REAL-TIME CAMERA SURVEILLANCE

P.A.S.Tharana

(IT21822094)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

DECLARATION

I declare that this is my work. This proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning. To the best of my knowledge and belief, it does not contain any previously published material written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
P.A.S.Tharana	IT21822094	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: [illegible] Date 29/8/25

Signature of the co-supervisor: [illegible] Date 29/08/25

ABSTRACT

This project introduces a vision- and IoT-based fire monitoring system that combines computer vision technology with intelligent sensors and alert mechanisms to enhance warehouse safety. By utilizing the capabilities of surveillance cameras to identify fire in real time, along with a network of MQ-7 smoke sensors, LEDs, and buzzers for localized detection and response, the system provides both visual and audible alerts to ensure rapid awareness. The monitoring framework comprises four primary functions: identifying the presence of fire, determining nearby shelves within the fire's proximity, classifying the fire size (small, medium, or large), and predicting the probable direction of fire spread. In addition, the IoT subsystem activates eight LEDs and four buzzers corresponding to directional cues (up, down, left, and right), which light up and sound alarms when fire or smoke is detected providing clear situational awareness even under low-visibility conditions.

The system is developed in Python using OpenCV and TensorFlow for the vision component, while the IoT module communicates with Arduino-controlled smoke sensors and alert units. Fire detection operates through contour identification and deep learning-based image classification, shelf detection through distance estimation and object recognition, and fire spread prediction through sequential frame analysis of environmental changes.

By integrating both vision-based and sensor-based mechanisms, the system ensures higher reliability, early detection, and multi-modal alerting. This hybrid architecture offers a scalable, cost-effective, and practical approach to real-time warehouse safety, transforming standard surveillance and simple hardware components into a unified, intelligent fire prevention network.

Keywords: Fire detection, Warehouse safety, Computer vision, IoT integration, Real-time surveillance, Fire size classification, Shelf proximity detection, Fire spread prediction, OpenCV, TensorFlow, MQ-7 sensor, LEDs, Buzzers, Deep learning, Object detection, Fire risk mitigation

ACKNOWLEDGEMENT

First, I would like to express my deepest gratitude to my supervisor, **Prof. Samantha Rajapaksha**, for his continuous guidance, encouragement, and invaluable feedback throughout this project. I also wish to thank **Dr. Dinuka Wijendra** for his support and constructive suggestions that helped refine this work. Finally, my sincere appreciation goes to the **academic staff of SLIIT – Faculty of Computing, Department of Information Technology**, for providing the knowledge, resources, and infrastructure necessary to complete this research.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network – a type of deep learning model commonly used for image classification and detection.
YOLO	You Only Look Once – a real-time object detection algorithm that processes images in a single pass through the network.
YOLOv5	A popular and optimized version of the YOLO model for object detection, known for its speed and accuracy.
OpenCV	Open Source Computer Vision Library – a popular library of programming functions used for real-time computer vision.
Flask	A lightweight Python web framework used to develop web applications and APIs.
RNN	Recurrent Neural Network – a class of neural networks used for sequence prediction, such as video or time-series data.
IoT	nternet of Things – a network of physical devices embedded with sensors, software, and connectivity to collect and exchange data.
Bounding Box	A rectangular box that encloses an object in an image, used in object detection to localize and label targets.

1 List of abbreviations

1. INTRODUCTION

Warehouses are an important component of the global supply chain and serve as storage facilities for goods, raw materials, and key inventory. However, these environments are often unsafe due to the potential for fire hazards. Fire dangers in warehouses pose serious risks, not only in terms of lives lost and property damage but also in significant economic losses and disruptions to the supply chain. Traditional fire safety systems such as smoke detectors, thermal sensors, and sprinkler systems are widely used; however, these systems are primarily reactive and may only trigger after smoke or heat is detected. In some fire scenarios, flames can develop before noticeable smoke or temperature changes occur, resulting in delayed responses. Moreover, conventional fire detection methods often lack early warning and spatial awareness, especially in large or cluttered warehouse environments where fire dynamics can be complex.

The integration of computer vision and artificial intelligence (AI) within modern surveillance systems offers a promising approach to enhancing fire detection and monitoring. Building on this concept, the proposed system introduces a **hybrid vision- and IoT-based real-time fire monitoring solution** designed specifically for warehouse safety. The system utilizes deep learning algorithms and video analytics to identify fire incidents through existing surveillance cameras, while an IoT subsystem—comprising MQ-7 smoke sensors, LEDs, and buzzers—provides physical feedback for immediate awareness. The IoT module consists of eight LEDs and four buzzers arranged to represent directional cues (up, down, left, and right). When smoke or fire is detected, the corresponding LEDs and buzzers on that side are activated to indicate the source direction and provide both visual and audible alerts. This ensures rapid recognition of fire or smoke presence, even in conditions of low visibility or when personnel are dispersed across large spaces.

Our application performs four main intelligent functions: first, it determines whether a fire is present in the video feed; second, it identifies the proximity of surrounding shelves relative to the fire; third, it classifies the fire size as small, medium, or large; and finally, it predicts the likely direction of fire spread based on frame-sequence analysis. This comprehensive functionality improves situational awareness and enables warehouse personnel to respond more quickly and strategically, minimizing potential damage and risk.

The system is developed using Python, OpenCV, and TensorFlow for the vision component, integrated with an Arduino-based IoT module for the smoke sensors and alert mechanisms. The combined architecture processes live video streams in real time, applies deep learning for detection and classification, and synchronizes physical indicators through serial communication with the IoT hardware. This hybrid design allows the system to remain cost-effective, scalable, and non-intrusive, as it reuses existing CCTV infrastructure while incorporating affordable sensor components.

Overall, this research bridges the gap between traditional reactive fire response methods and modern, proactive fire prevention strategies. By merging computer vision intelligence with IoT-based sensory alerts, the system proposes a data-driven, real-time safety framework that enhances warehouse fire monitoring and provides a foundation for next-generation industrial safety automation.

1.1 Background

Fire risks in industrial and storage facilities remain a significant concern globally. Warehouses store large volumes of materials, many of which are highly flammable, and these environments often operate with limited human supervision. Conventional fire alarm systems such as smoke detectors, heat sensors, and sprinklers are primarily reactive and may not provide adequate spatial context or early analytical information before a fire becomes uncontrollable. Consequently, researchers have increasingly turned toward **computer vision and Internet of Things (IoT)** technologies, leveraging their ability to provide real-time monitoring, data fusion, and intelligent alerting for early fire detection and prevention.

Numerous studies, such as those by Jin and coworkers [1], have demonstrated that vision-based fire detection using deep learning can accurately identify flames, smoke, or other abnormal visual patterns from video feeds in real time. Jin et al. [1] provided an extensive review of video fire detection techniques, emphasizing the effectiveness of convolutional neural networks (CNNs) for real-time flame recognition. Similarly, Martins [3] and Islam and Habib [5] have shown that YOLO-based object detection models are capable of achieving real-time classification of fire with low latency, making them suitable for industrial-scale monitoring. Lightweight architectures, such as Light-YOLOv5 [4], further enhance scalability by enabling deployment in low-resource or embedded hardware environments.

Beyond fire detection, estimating the fire's size and location in a scene provides essential information for situational awareness and emergency decision-making. Studies such as [10] have utilized pixel-based area estimation and contour analysis to classify fire severity and develop actionable responses. Techniques using segmentation, bounding boxes, and heat-based modeling have also been proposed to strengthen fire size estimation accuracy.

When considering the spatial layout of warehouses, proximity detection of nearby structures such as shelves and storage racks is equally important. AI-based shelf detection models, as explored in [7][8], have been effectively used for inventory monitoring and spatial mapping. These techniques can be adapted to fire safety contexts to determine the distance between a fire source and key structural assets, enabling improved prioritization during emergency response and containment.

In addition, predicting the **direction of fire spread** represents one of the most advanced and evolving research areas in fire safety. Deep learning and generative models [11][12][14] have been used to forecast fire progression by analyzing environmental cues and temporal frame sequences. Projects such as Stanford's CS231n work [13] have demonstrated that computer vision can dynamically visualize and predict the spread of fire under varying environmental conditions an approach that could greatly enhance warehouse evacuation planning and suppression strategies.

While vision-based systems offer powerful capabilities, their effectiveness can be further amplified through **IoT integration**. Incorporating **MQ-7 smoke sensors** enables early-stage detection of gas and smoke presence even before flames are visually apparent. Complementary **LEDs and buzzers**, arranged directionally (up, down, left, and right), provide clear visual and auditory feedback when fire or smoke is detected, helping workers quickly identify the affected zone. This hybrid approach combines the strengths of AI-based image analysis with the immediacy of hardware-based alerts, enhancing both reliability and responsiveness in real-world warehouse environments.

Recent systematic reviews [2][15] affirm that vision-based and IoT-assisted fire detection systems are feasible, scalable, and cost-effective alternatives to purely sensor-based or camera-only setups. These combined systems can leverage existing

CCTV infrastructure while integrating low-cost IoT components to provide comprehensive situational awareness and early fire response.

This growing body of research forms the foundation for the proposed **vision- and IoT-based fire monitoring system**, which unifies fire detection, shelf proximity estimation, fire size classification, fire spread prediction, and real-time physical alerts. By integrating these technologies into a single framework, the proposed solution addresses major gaps in traditional warehouse fire monitoring and establishes a proactive, intelligent approach to industrial safety.

1.2 Literature Survey

This chapter provides a review of relevant literature focused on the five major components of the proposed fire monitoring system: **fire detection**, **shelf proximity detection**, **fire size classification**, **fire spread prediction**, and the newly integrated **IoT-based alerting subsystem**. Each of these components is reviewed to identify current findings, existing research gaps, and their implications for improving warehouse fire safety through both computer vision and hardware-assisted systems.

1.2.1 Fire Detection

Fire detection using computer vision has emerged as a powerful and cost-effective alternative to traditional sensor-based systems. Jin et al. [1] reviewed various video-based fire detection methods and confirmed that convolutional neural networks (CNNs) can successfully identify flames, smoke, and other abnormal visual patterns in surveillance footage. Their findings highlight the potential for deep learning methods to achieve both high accuracy and real-time responsiveness. Martins [3] further validated this through YOLO-based object detection models, demonstrating that such architectures can recognize and classify fire instances from live video feeds with minimal latency. Similarly, Islam and Habib [5] confirmed the robustness of YOLOv5 models in detecting fires under challenging environmental conditions such as fluctuating lighting or background motion. Xu et al. [4] introduced Light-YOLOv5, a lightweight model suitable for embedded or resource-constrained systems, enabling scalable deployment in large industrial settings.

However, despite these advancements, most vision-only systems still rely solely on image-based cues, which can sometimes delay detection in early fire stages where smoke appears before visible flames. To address this, the integration of **IoT-based smoke sensors** such as **MQ-7** has been explored in recent studies as an early-warning complement to computer vision. These sensors detect carbon monoxide and smoke particles, triggering alerts even before the visual features of fire become apparent. By merging the strengths of both approaches AI-driven vision detection and sensor-based smoke analysis hybrid systems achieve greater reliability and response speed, minimizing false negatives in complex warehouse conditions.

1.2.2 Shelf Detection and Distance Estimation

Though largely discussed in retail and logistics scenarios, shelf detection and visual distance estimation are relevant in fire monitoring in determining the distance a given fire is from critical storage infrastructure. A deep learning-based system for monitoring shelves is discussed in [7] where the authors conducted their work utilizing an object detection model to detect and track shelves for inventory purposes. Kumar [8] created a system based on AI that located empty shelves through vision-based spatial assessments to trigger notifications for restocking purposes.

These methodologies may be used for explicit warehouse fire safety, potentially by estimating the distance from detected fire zones to shelving units to provide spatial awareness for better emergency planning.

1.2.3 Fire Size Classification

Estimate of fire size is important when assigning a severity level to a fire and therefore how to respond. The traditional techniques used to generate a spatial estimate of the fire, such as measuring pixel area and detecting the contour of the fire [10], can be adopted, but there was also an approach in this study that provided estimates of fire size using image segmented, contour-based models, that could be measurably classified as small, medium, or large. Additionally, heat based modeling and bounding boxes have been discussed in literature as secondary approaches to further support the classification of stages of fire intensity and periods of fire growth.

1.2.4 Fire Spread Prediction

Predicting the direction and rate of fire spread is an emerging and complex field within computer vision research. Jin et al. [11] and Stanford's CS231n project [13] showed that temporal deep learning models, when applied to sequential video frames, can effectively predict fire propagation patterns. Studies using **U-Net architectures** [12] and **generative deep learning approaches** [14] demonstrated that modeling temporal dynamics enables accurate forecasting of fire fronts in both two- and three-dimensional spaces. These insights can be applied to indoor warehouse environments, where airflow, narrow aisles, and vertical shelving contribute to unique fire spread behavior.

In the proposed system, **optical flow estimation** is used to approximate fire movement between frames, while IoT alerts provide immediate directional feedback through LEDs corresponding to detected spread direction (up, down, left, or right). This combination of digital prediction and physical directional cues ensures that both the automated system and human operators are aligned on the evolving situation, supporting rapid and coordinated responses.

1.2.5 IoT-Based Fire Alerting and Integration

Recent developments in IoT-based fire prevention systems emphasize the importance of real-time alerting mechanisms that combine sensors, actuators, and wireless communication. Systems integrating **MQ-series gas sensors** and **Arduino-based controllers** have been explored to detect smoke, gas, and flame presence at low cost and high responsiveness. However, most of these systems function independently without integration into vision-based detection frameworks.

The innovation in the proposed research lies in merging IoT sensor feedback with computer vision insights. The **MQ-7 sensors** detect smoke and carbon monoxide, while the **LED and buzzer modules** provide immediate spatial feedback about the fire's location and severity. This hybrid configuration forms a **multi-layered safety model**, where the IoT layer ensures rapid local response and the AI layer provides analytical intelligence and monitoring context. Such integration aligns with the modern direction of **Industry 4.0**, emphasizing connected, intelligent, and autonomous safety systems for industrial environments.

1.3 Research Gap

In recent years, advances in computer vision and deep learning have significantly improved automated fire detection systems. However, their application in **warehouse fire safety** remains limited, fragmented, and not fully understood. Most existing research focuses on isolated aspects such as fire detection or smoke analysis without integrating contextual spatial understanding or real-time physical alert mechanisms. Moreover, while IoT-based systems are becoming more popular, they are often deployed separately from vision-based solutions, resulting in redundant or delayed responses. A comprehensive analysis of the literature has revealed several gaps that this research project aims to address.

1.3.1 Limited spatial context in advanced fire detection systems

Most camera-based automated fire detection systems are capable of identifying flames or smoke using CNNs and object detection models [1][3][5]. However, these systems are typically developed for open or generic environments and fail to consider **spatial context** that is, which nearby objects, shelves, or materials are at immediate risk and how the fire's position affects the overall warehouse safety. In a densely packed warehouse, understanding this spatial relationship is crucial. Without this context, even accurate fire detection provides limited actionable value during emergencies. The proposed system overcomes this by integrating **shelf proximity estimation** and visual mapping of at-risk regions, giving operators precise, context-aware alerts.

1.3.2 Fire spread prediction models are developed for outdoor fires, not warehouse fires

Deep learning approaches for predicting fire spread have shown strong results in outdoor wildfire settings [11][12][13][14], where wind, vegetation, and terrain are dominant variables. However, these models are poorly suited to **indoor warehouse**

environments, where airflow, narrow aisles, stacked shelves, and enclosed geometry dramatically alter fire behavior. Existing prediction models rarely capture these constraints. This project addresses the gap by developing an **indoor fire spread prediction mechanism** using optical flow and frame-sequence analysis tailored to enclosed warehouse layouts, providing directional insights that inform both evacuation and suppression strategies.

1.3.3 Fire size estimate is rarely connected to response prioritization

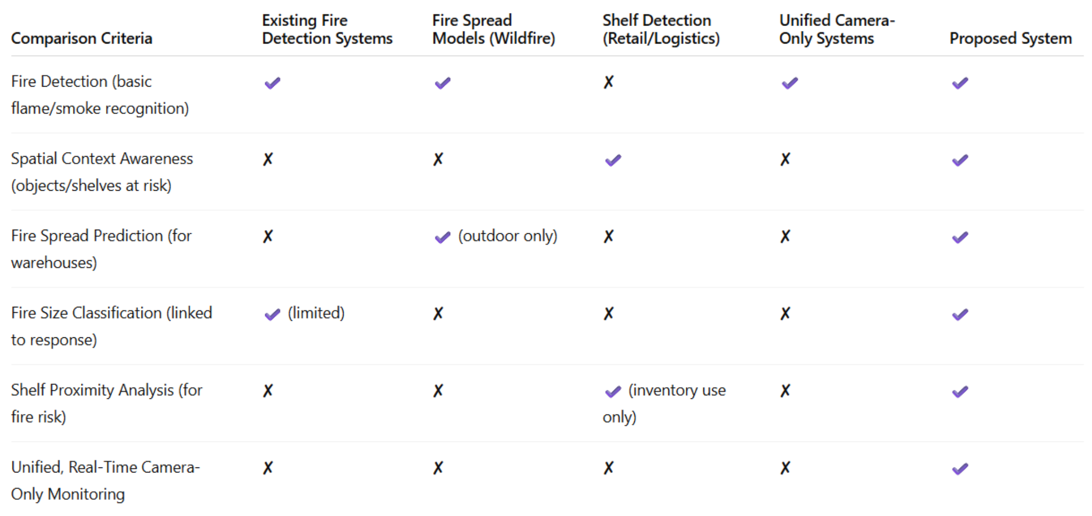
While fire size estimation methods such as contour and pixel-area analysis [10] can accurately classify fires into small, medium, or large categories, these measurements are often isolated outputs. They are seldom linked to operational responses such as **alert escalation**, **buzzer activation**, or **resource prioritization**. Consequently, fire size data remains underutilized in practical settings. The proposed system closes this gap by directly coupling fire size classification with the **IoT alert subsystem**—triggering corresponding buzzer tones and LED indicators based on the detected severity, thereby creating an intelligent, responsive safety feedback loop.

1.3.4 Shelf detection methods are not adapted for fire-related risk assessment

Shelf and rack detection systems in retail logistics [7][8] use object detection to track product availability or assist with automation of replenishment of stock. They are not built to function in the fire-related risk contexts, as they rely on the proximity of fire to a structure, such as a high storage rack, without consideration for safety. In an emergency with fire, shelf proximity assessment (in real-time) could assist with fire suppression, ways for people to evacuate safely, or even for automation to notify authorities, using a system that doesn't exist today.

1.3.5 Absence of a Unified Vision-and-IoT Real-Time Monitoring System for Warehouses

Although both camera-based and IoT-based systems have evolved independently, **no existing system integrates them into a single cohesive framework** that can detect fires, assess shelf proximity, estimate fire size, predict spread, and physically alert personnel simultaneously. Current systems either rely solely on sensors or depend exclusively on computer vision, lacking the complementary benefits of both. The proposed solution addresses this by developing a **hybrid vision-and-IoT fire monitoring system** that combines deep learning–based visual detection with **MQ-7 smoke sensing, directional LEDs, and audible buzzers**. This integration ensures multi-layered awareness vision-based analysis for intelligence, and IoT feedback for immediate on-ground response creating a scalable, low-cost, and proactive fire safety platform tailored for warehouse environments.



Comparison Criteria	Existing Fire Detection Systems	Fire Spread Models (Wildfire)	Shelf Detection (Retail/Logistics)	Unified Camera-Only Systems	Proposed System
Fire Detection (basic flame/smoke recognition)	✓	✓	✗	✓	✓
Spatial Context Awareness (objects/shelves at risk)	✗	✗	✓	✗	✓
Fire Spread Prediction (for warehouses)	✗	✓ (outdoor only)	✗	✗	✓
Fire Size Classification (linked to response)	✓ (limited)	✗	✗	✗	✓
Shelf Proximity Analysis (for fire risk)	✗	✗	✓ (inventory use only)	✗	✓
Unified, Real-Time Camera-Only Monitoring	✗	✗	✗	✗	✓

2 Research Gap Summary

1.4 Research Problem

Warehouses are essential components of modern industry and logistics networks, often housing large volumes of cargo including highly flammable materials stored in dense configurations. Despite their importance, warehouse fire safety systems remain largely reactive, depending primarily on **smoke detectors, heat sensors, and sprinkler systems**. These systems, while useful, often lack both **spatial awareness** and **predictive capability**, responding only once a fire has intensified and reached a dangerous stage. As a result, such solutions provide limited time for evacuation or preventive action, thereby exposing warehouse personnel, property, and operations to severe risk.

Recent advancements in **computer vision** and **deep learning** offer the potential to overcome these shortcomings through **real-time fire detection** using standard surveillance camera footage [1][3]. Studies have shown that convolutional neural networks (CNNs) and YOLO-based object detectors [5] can effectively identify flames and smoke in video streams, even under challenging environmental conditions. Lightweight versions such as **Light-YOLOv5** [4] have further enabled these systems to function on low-resource hardware, making them practical for industrial-scale implementation. However, while these models accurately detect fire visually, they generally fail to account for **environmental context** such as how close the fire is to critical storage shelves, what direction it may spread, or how quickly it is growing.

Additionally, although **shelf and rack detection** systems have been widely studied in retail and logistics settings [7][8], their use in **fire-related emergency applications** remains unexplored. No existing system evaluates the **proximity between active fires and structural elements** like shelving units or stored goods, which could drastically influence risk assessment and resource prioritization during an emergency. Similarly, while **contour-based fire size estimation** methods [10] can categorize fire severity into small, medium, and large, few systems leverage this information for real-time decision-making or alert escalation.

Moreover, **fire spread prediction models** have been predominantly developed for outdoor wildfire scenarios [11][12][14], where environmental factors such as vegetation, wind, and terrain play key roles. These models fail to represent **indoor fire behavior**, where airflow patterns, confined geometry, and vertical storage configurations dictate fire dynamics. Even advanced computer vision projects such as Stanford's CS231n study [13] have not addressed the unique fire progression challenges within enclosed warehouse environments.

Equally important is the **lack of early physical alert mechanisms** in existing computer vision-based systems. Most vision-only approaches depend solely on on-screen warnings or digital alerts, offering little immediate sensory feedback for on-site workers. This gap highlights the need for integrating **IoT-based alerting components** such as **MQ-7 smoke sensors**, **directional LED indicators**, and **buzzers** which can detect early-stage smoke, identify the affected direction (up, down, left, or right), and provide instant visual and auditory feedback.

Therefore, despite notable progress across these individual research areas, there remains **no unified hybrid system** that integrates **vision-based fire detection**, **shelf proximity analysis**, **fire size classification**, **spread prediction**, and **IoT-driven early warning** into a single real-time framework. Existing systems are either sensor-only or camera-only, lacking the **synergistic intelligence** required to combine environmental awareness with immediate, actionable alerts.

1.5. Objectives

1.5.1 Main Objective

To design and implement a **holistic, real-time Vision- and IoT-based fire monitoring system** that utilizes existing surveillance camera infrastructure integrated with smoke sensors and alert hardware to provide fire detection, shelf proximity analysis, fire size classification, and fire spread prediction delivering an intelligent, pre-emptive safety framework for warehouse environments.

1.5.2 Specific Objectives

1. **To develop a fire detection algorithm utilizing deep learning based on video feed**

   The goal is to leverage computer vision methods such as Convolutional Neural Networks (CNN), as well as Object Detection based on YOLO, to determine accurately whether there is fire (flame / smoke) present based on a live feed to the camera. The resulting deep learning model will be trained and evaluated to accommodate the factors such as lighting changes, clutter and motion commonly associated with warehouses, and it will alert immediately once it detects the fire.

2. **To implement shelving detection, and proximity estimation**

   The second module will represent the shelving units, and provide the spatial distance of the shelving units to the fire based on object detection (e.g., YOLO, SSD), the bounding box coordinates, and image depth methods. Consequently, we will have situational awareness of which items are at risk, and can inform emergency responders of the items to prioritize to contain the fire or start evacuation.

3. **To classify fire size based on visual features detected using the camera feed**

   Fires can be classified as small, medium or large, based on computed flame area, contour assignments, and intensity derived using pixel clustering and segmentation algorithms. These visual features would classify the incident accordingly, immediately determining somewhere between warning and action tiers or escalation.

4. **To assess the potential fire direction of spread using a frame-sequence analysis**

   Temporal deep learning will be implemented to analyze the motion of flames and smoke through sequential video frames. The methods of optical flow, recurrent neural networks (RNNs), or using U-Net models will showcase movement to help identify trends in the flames and predict the likely direction of fire spread, which may help plan for evacuations, activate fire suppression, or identify possible containment zones.

5. **To integrate an IoT-based smoke detection and alert subsystem**
   The IoT component will include MQ-7 smoke sensors, eight LEDs, and four buzzers controlled through an Arduino module. The sensors will detect early-stage smoke, while LEDs and buzzers will provide directional visual and audible alerts to warehouse personnel based on the detected fire's location and severity. This ensures rapid on-site awareness even before flames are fully visible.

6. **To combine all modules into one, deployable, real-time vision-based monitoring system**

   The final goal is to bring together all four main functional modules into one system which is capable of real-time analysis on standard computing hardware without development of new hardware such as drones and using existing CCTV systems (if they exist). The system will provide safety officers in warehouses with a user interface and dashboard that visually delineate alerts, proximity heatmaps (the heat maps denoting point of origin of fire pattern identification), and predictive overlays as situational awareness and to help fire emergency decision-making.

2. METHODOLOGY

This research will take a design and implementation-based methodology to explore a real-time hybrid fire monitoring system using cameras and IoT components. The design phase will begin with an analysis of warehouse fire risks, leading to the design of a system architecture using five core modules: fire detection, proximity estimation to shelves, fire size classification, prediction of fire spread, and an IoT-based smoke and alert subsystem. For model training, publicly available datasets and annotated videos will be used, with data augmentation to enhance robustness. In the implementation phase, deep learning models such as YOLOv5 and CNNs will be applied for fire and shelf detection.

OpenCV will be used for fire size estimation, and optical flow or U-Net models for fire spread prediction. The IoT subsystem, equipped with MQ-7 smoke sensors, LEDs, and buzzers, will detect smoke early and provide directional visual and audio alerts for quick response. All modules will be integrated into a single real-time system with a simple dashboard interface, and evaluated in a simulated warehouse environment for accuracy, response time, and reliability..

2.1. System overview

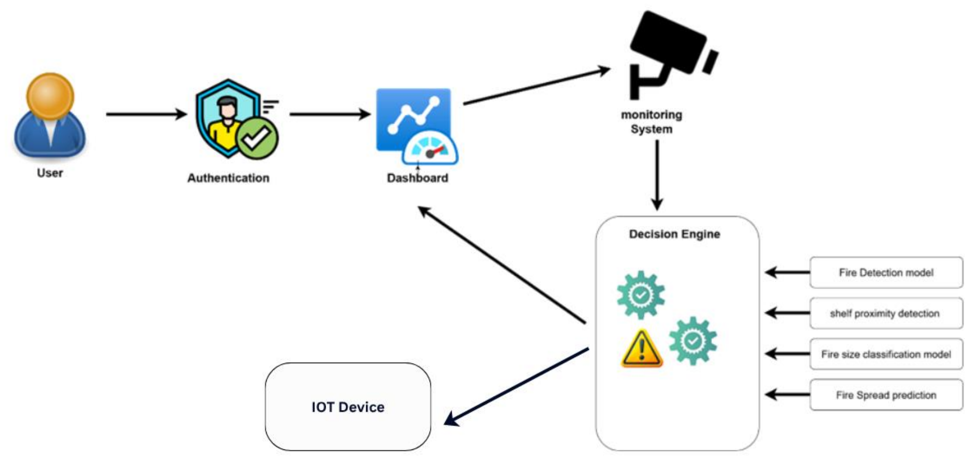


Figure 1 System Overview Diagram

The system architecture diagram illustrates the complete workflow of the proposed camera-based fire monitoring system within a warehouse environment. The process begins with a **user logging into the system** through a secure authentication interface. Upon successful login, the user accesses the **dashboard**, which serves as the main control panel for interacting with the warehouse surveillance and fire monitoring functionalities. The dashboard is linked to live video input from **surveillance cameras** installed throughout the warehouse.

The video feed is then processed through four distinct **deep learning-based models** integrated within the system. These include a **Fire Detection module**, which identifies the presence of flames or smoke in real time, followed by a **Shelf Proximity Detection module**, which determines the location and distance between the fire and nearby shelving units to assess potential risk zones. Next, the **Fire Size Classification module** evaluates the scale of the fire—categorizing it as small, medium, or large—based on pixel area and contour analysis. Finally, the **Fire Spread Prediction module** analyzes sequential video frames to estimate the likely direction in which the fire may progress.

The outputs of these four models are fed into a centralized **Decision Engine**, which synthesizes all incoming data to determine the appropriate system response. This includes generating visual alerts on the user dashboard, activating **sound alarms**, and displaying specific **actionable instructions** (e.g., "Evacuate area near Shelf 4", or "Fire spreading east, engage suppression system"). The decision engine ensures that users receive accurate, real-time feedback with clear next steps to prevent escalation and minimize damage. This comprehensive, vision-based approach enhances situational awareness and responsiveness during fire emergencies in warehouse environments.

Component	**Technology / Framework**
Frontend (Dashboard UI)	React
Backend	Express
Database	MongoDB
Computer Vision Library	OpenCV (image processing, contour analysis)

Deep Learning Models	CNNs, YOLOv5 (fire & shelf detection), U-Net / RNN (fire spread prediction)
Frameworks	TensorFlow, PyTorch (for training & inference)
Deployment	Local Server / GPU-supported PC (edge deployment)
Optional Cloud Hosting	AWS / Google Cloud (for scalability)

3 Technologies and frameworks

2.2 Commercialization

The project is commercially viable within the fire safety and warehouse management market. The proposed system fulfills a vital need for real-time detection, risk assessment, and prevention of fire hazards using existing CCTV systems. It achieves this with a single camera (rather than with smoke or IOT-based systems) and at a cost point that is attractive to businesses due to the scalability and ease of implementation, and significant hardware investment already negated by the existing infrastructure and networks.

1. **Target Market:**

The initial target customers include:

- Warehouses and distribution centers (retail, logistics, manufacturing)
- Storage units and facilities with high fire risk (e.g., paper, textiles, chemicals)
- Insurance companies requiring fire risk mitigation tools
- Government and defense storage depots
- Medium to large enterprises with existing CCTV systems

2. **Unique Selling Points (USPs):**

- **Camera-Only System**: No need for IoT sensors or thermal cameras—just plug into existing CCTV.
- **Real-Time AI Analytics**: Immediate detection, classification, and prediction using deep learning.
- **Modular Design**: Businesses can start with fire detection and scale to full features as needed.

- **Cost-Effective**: Built entirely using open-source tools with minimal hardware upgrades.
- **Non-Intrusive Deployment**: Easily integrates into existing surveillance systems without downtime.

3. **Market Entry Strategy:**

- **Pilot Programs**: Offer free/discounted pilot trials to a select group of warehouses to generate testimonials and data.
- **Partnerships**: Collaborate with CCTV vendors, warehouse automation providers, and fire safety consultants.
- **Online Marketing**: Launch a professional website with demo videos, case studies, and ROI calculator.
- **Industry Events**: Showcase at fire safety expos, tech innovation conferences, and supply chain summits.
- **Regulatory Certification**: Work toward recognition or compliance with fire safety standards to gain trust.

4. **Scalability and Expansion:**

- **Cross-Domain Use Cases**: Adapt the system for use in schools, hospitals, server rooms, and shopping malls.
- **Multilingual Support**: Customize alert messages and dashboards for regional markets.
- **Cloud + Edge Deployment**: Offer both cloud-based and on-premises installations to suit client needs.

Mobile App Integration: Allow real-time alerts and fire visualization via mobile dashboards.

2.3 Testing & Implementation

The system is structured as a modular, service architecture using a React (admin) frontend, a Flask inference service for real-time vision AI, and a Node/Express API with MongoDB for event logging and access control. There is only one external device used in development (a webcam) which may later point to an RTSP stream from warehouse CCTV. The dashboard administers the authenticate of administrators, streams validated video, renders alerts (banners + sound/voice alerts) where necessary based on the fire size and proximity to shelf, and stores alerts for audit.

Frontend Implementation – React (Administrator Panel)

The admin UI is a **React** SPA organized by pages and shared components. It consumes two backends:

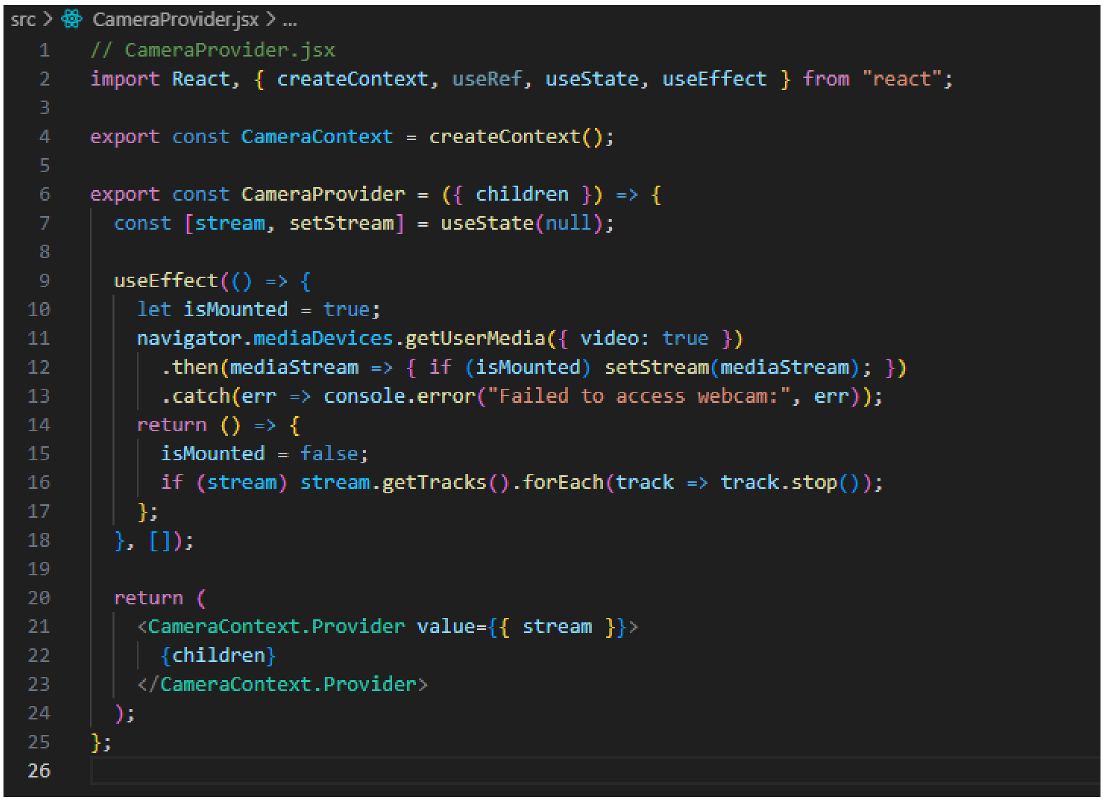
- **Flask** – MJPEG video stream (/video) + status JSON (/status).
- **Node/Express** – auth, users, and alert/event logs.

Key pages & components

1. **Fire Monitoring Page** (/monitoring)
   - Streams live annotated video.
   - Shows **confidence**, **size class (S/M/L)**, **nearest-shelf gap**, and **spread direction**.
   - Triggers **audio + TTS** alerts on state transitions (normal → warning/critical).
2. **Alerts & History** (/alerts)
   - Lists saved alerts with time, severity, snapshot link, and export (CSV).
3. **Cameras** (/cameras)
   - Configure source (webcam/RTSP), frame rate, and per-camera thresholds.
4. **Users & Roles** (/users)
   - Admin-only: manage accounts and roles (Admin, Supervisor, Viewer).

State & libraries

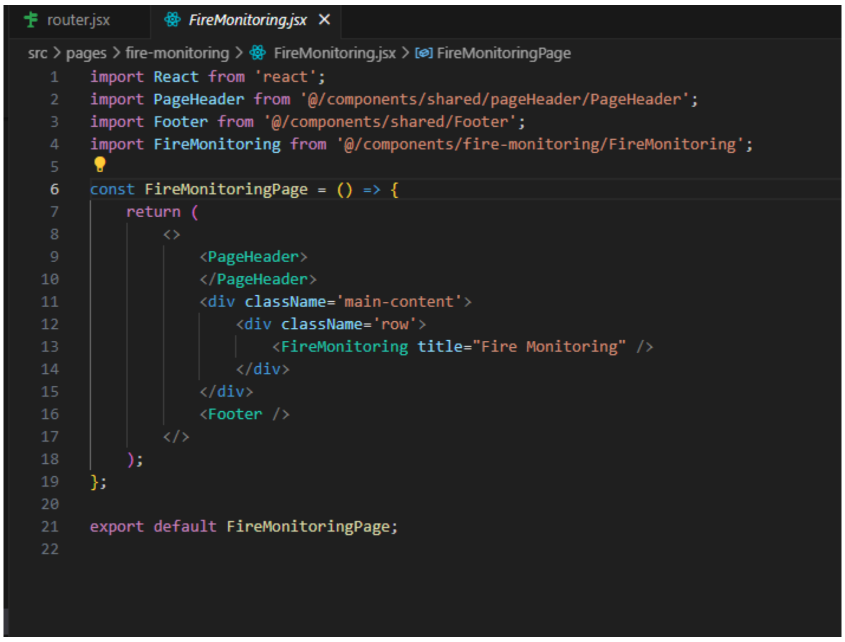
- **React Router** for navigation; **Context** for camera stream; **Axios** for API calls.
- **JWT** stored via **HttpOnly cookies**; role-based route guards.
- **Socket.IO** real-time data transferring.



src > CameraProvider.jsx > ...

```jsx
// CameraProvider.jsx
import React, { createContext, useRef, useState, useEffect } from "react";

export const CameraContext = createContext();

export const CameraProvider = ({ children }) => {
  const [stream, setStream] = useState(null);

  useEffect(() => {
    let isMounted = true;
    navigator.mediaDevices.getUserMedia({ video: true })
      .then(mediaStream => { if (isMounted) setStream(mediaStream); })
      .catch(err => console.error("Failed to access webcam:", err));
    return () => {
      isMounted = false;
      if (stream) stream.getTracks().forEach(track => track.stop());
    };
  }, []);

  return (
    <CameraContext.Provider value={{ stream }}>
      {children}
    </CameraContext.Provider>
  );
};
```

Figure 2 Camera-provider

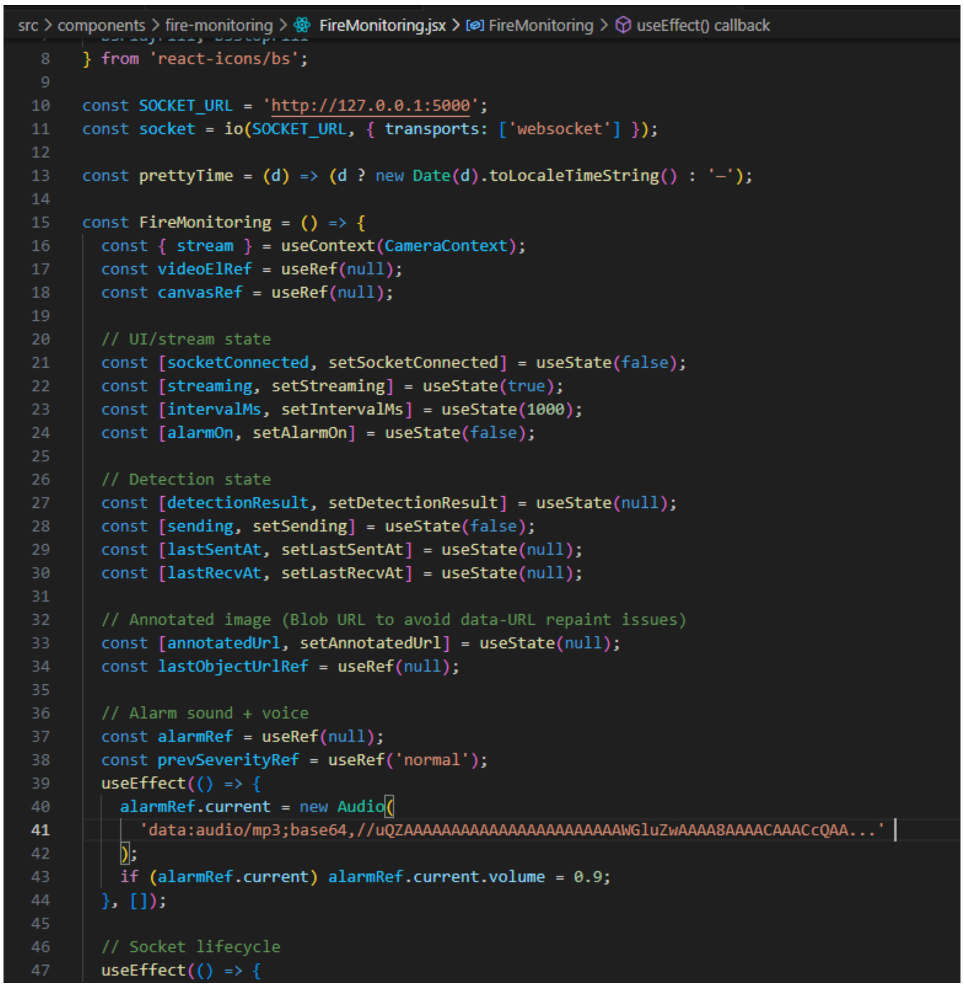


router.jsx | FireMonitoring.jsx

src > pages > fire-monitoring > FireMonitoring.jsx > FireMonitoringPage

```jsx
import React from 'react';
import PageHeader from '@/components/shared/pageHeader/PageHeader';
import Footer from '@/components/shared/Footer';
import FireMonitoring from '@/components/fire-monitoring/FireMonitoring';

const FireMonitoringPage = () => {
    return (
        <>
            <PageHeader>
            </PageHeader>
            <div className='main-content'>
                <div className='row'>
                    <FireMonitoring title="Fire Monitoring" />
                </div>
            </div>
            <Footer />
        </>
    );
};

export default FireMonitoringPage;
```

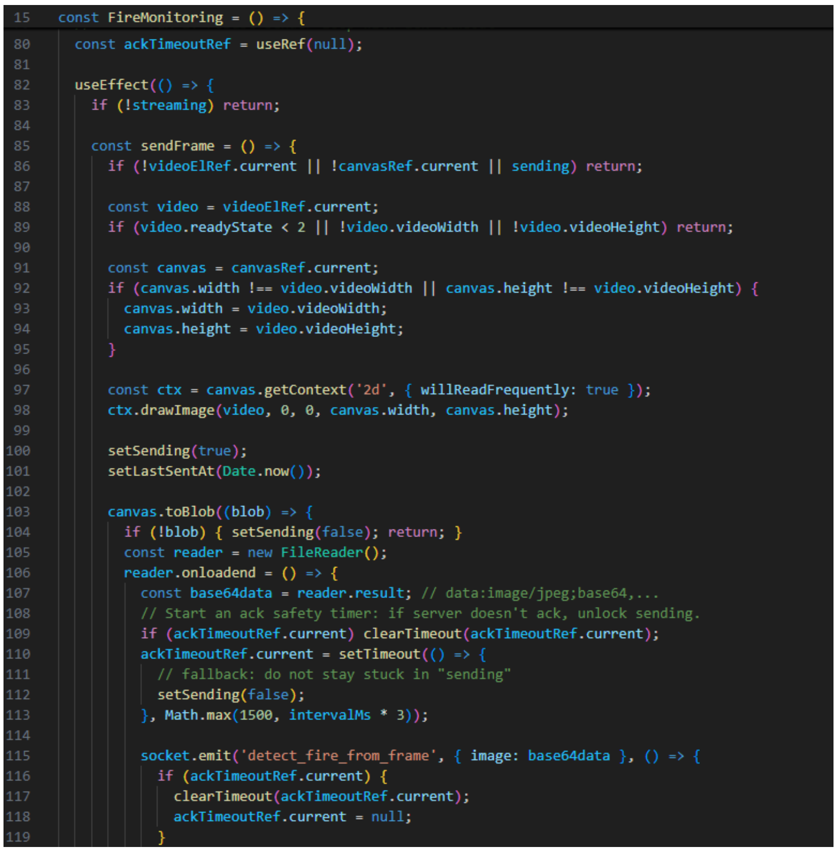
Figure 3 FireMonitoringPage



src > components > fire-monitoring > FireMonitoring.jsx > FireMonitoring > useEffect() callback

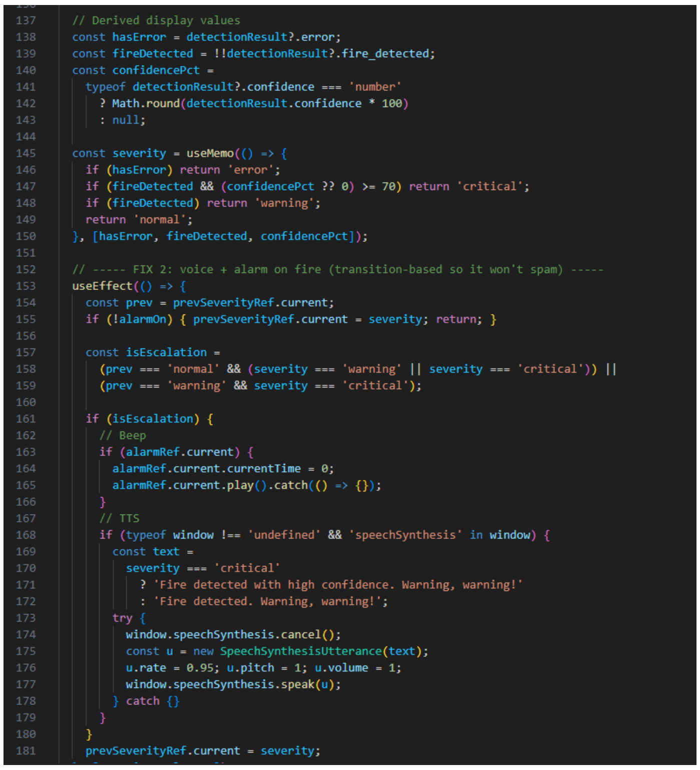
```jsx
} from 'react-icons/bs';

const SOCKET_URL = 'http://127.0.0.1:5000';
const socket = io(SOCKET_URL, { transports: ['websocket'] });

const prettyTime = (d) => (d ? new Date(d).toLocaleTimeString() : '—');

const FireMonitoring = () => {
  const { stream } = useContext(CameraContext);
  const videoElRef = useRef(null);
  const canvasRef = useRef(null);

  // UI/stream state
  const [socketConnected, setSocketConnected] = useState(false);
  const [streaming, setStreaming] = useState(true);
  const [intervalMs, setIntervalMs] = useState(1000);
  const [alarmOn, setAlarmOn] = useState(false);

  // Detection state
  const [detectionResult, setDetectionResult] = useState(null);
  const [sending, setSending] = useState(false);
  const [lastSentAt, setLastSentAt] = useState(null);
  const [lastRecvAt, setLastRecvAt] = useState(null);

  // Annotated image (Blob URL to avoid data-URL repaint issues)
  const [annotatedUrl, setAnnotatedUrl] = useState(null);
  const lastObjectUrlRef = useRef(null);

  // Alarm sound + voice
  const alarmRef = useRef(null);
  const prevSeverityRef = useRef('normal');
  useEffect(() => {
    alarmRef.current = new Audio(
      'data:audio/mp3;base64,//uQZAAAAAAAAAAAAAAAAAAAAAAAWGluZwAAAA8AAAACAAACcQAA...'
    );
    if (alarmRef.current) alarmRef.current.volume = 0.9;
  }, []);

  // Socket lifecycle
  useEffect(() => {
```

Figure 4 shocket.io data fetch



```javascript
const FireMonitoring = () => {
  const ackTimeoutRef = useRef(null);

  useEffect(() => {
    if (!streaming) return;

    const sendFrame = () => {
      if (!videoElRef.current || !canvasRef.current || sending) return;

      const video = videoElRef.current;
      if (video.readyState < 2 || !video.videoWidth || !video.videoHeight) return;

      const canvas = canvasRef.current;
      if (canvas.width !== video.videoWidth || canvas.height !== video.videoHeight) {
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
      }

      const ctx = canvas.getContext('2d', { willReadFrequently: true });
      ctx.drawImage(video, 0, 0, canvas.width, canvas.height);

      setSending(true);
      setLastSentAt(Date.now());

      canvas.toBlob((blob) => {
        if (!blob) { setSending(false); return; }
        const reader = new FileReader();
        reader.onloadend = () => {
          const base64data = reader.result; // data:image/jpeg;base64,...
          // Start an ack safety timer: if server doesn't ack, unlock sending.
          if (ackTimeoutRef.current) clearTimeout(ackTimeoutRef.current);
          ackTimeoutRef.current = setTimeout(() => {
            // fallback: do not stay stuck in "sending"
            setSending(false);
          }, Math.max(1500, intervalMs * 3));

          socket.emit('detect_fire_from_frame', { image: base64data }, () => {
            if (ackTimeoutRef.current) {
              clearTimeout(ackTimeoutRef.current);
              ackTimeoutRef.current = null;
            }
```

Figure 5 Fire frame set



```javascript
// Derived display values
const hasError = detectionResult?.error;
const fireDetected = !!detectionResult?.fire_detected;
const confidencePct =
  typeof detectionResult?.confidence === 'number'
    ? Math.round(detectionResult.confidence * 100)
    : null;

const severity = useMemo(() => {
  if (hasError) return 'error';
  if (fireDetected && (confidencePct ?? 0) >= 70) return 'critical';
  if (fireDetected) return 'warning';
  return 'normal';
}, [hasError, fireDetected, confidencePct]);

// ----- FIX 2: voice + alarm on fire (transition-based so it won't spam) -----
useEffect(() => {
  const prev = prevSeverityRef.current;
  if (!alarmOn) { prevSeverityRef.current = severity; return; }

  const isEscalation =
    (prev === 'normal' && (severity === 'warning' || severity === 'critical')) ||
    (prev === 'warning' && severity === 'critical');

  if (isEscalation) {
    // Beep
    if (alarmRef.current) {
      alarmRef.current.currentTime = 0;
      alarmRef.current.play().catch(() => {});
    }
    // TTS
    if (typeof window !== 'undefined' && 'speechSynthesis' in window) {
      const text =
        severity === 'critical'
          ? 'Fire detected with high confidence. Warning, warning!'
          : 'Fire detected. Warning, warning!';
      try {
        window.speechSynthesis.cancel();
        const u = new SpeechSynthesisUtterance(text);
        u.rate = 0.95; u.pitch = 1; u.volume = 1;
        window.speechSynthesis.speak(u);
      } catch {}
    }
  }
  prevSeverityRef.current = severity;
```

Figure 6 voice alert and screen reader

Backend Implementation – Node.js + Python (Flask)

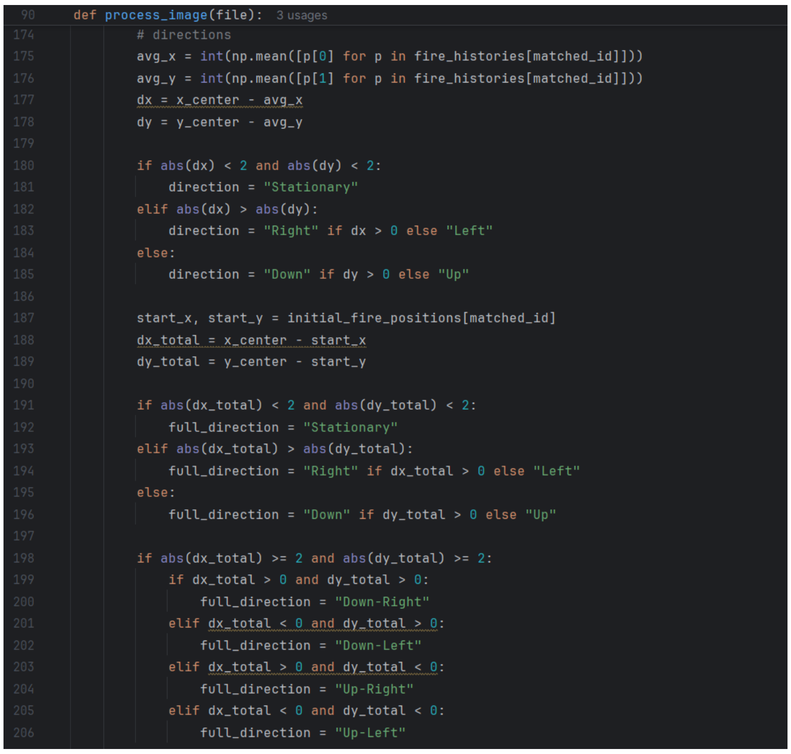
The back-end layer implements Node.js (Express.js) for system APIs and Python (Flask + PyTorch/TensorFlow) for ML-based fire and shelf detection services. This hybrid layered structure maximizes scalability, separation of concerns, and the optimal use of AI models. It was organized as controllers, services, and Python microservices that keep components small for maintainability and allow for easy future enhancements.

Core Functional Modules

1. **Camera & Stream Management API**
    - Handles real-time video input from CCTV streams or USB webcams.
    - Provides endpoints to configure camera sources, frame rates, and enable/disable monitoring.
    - For development, a single webcam was used; in production, RTSP streams from warehouse CCTV are supported.
2. **Fire Detection Microservice (Python/Flask)**
    - Implements a **YOLOv8-based model** trained on Roboflow fire datasets.
    - Detects flames/smoke per frame, returning bounding boxes, confidence scores, and annotated images.
    - Endpoints:
        - /detect_fire – accepts base64 image, returns JSON with fire boxes & probabilities.
        - /video – streams annotated MJPEG frames for admin dashboard.

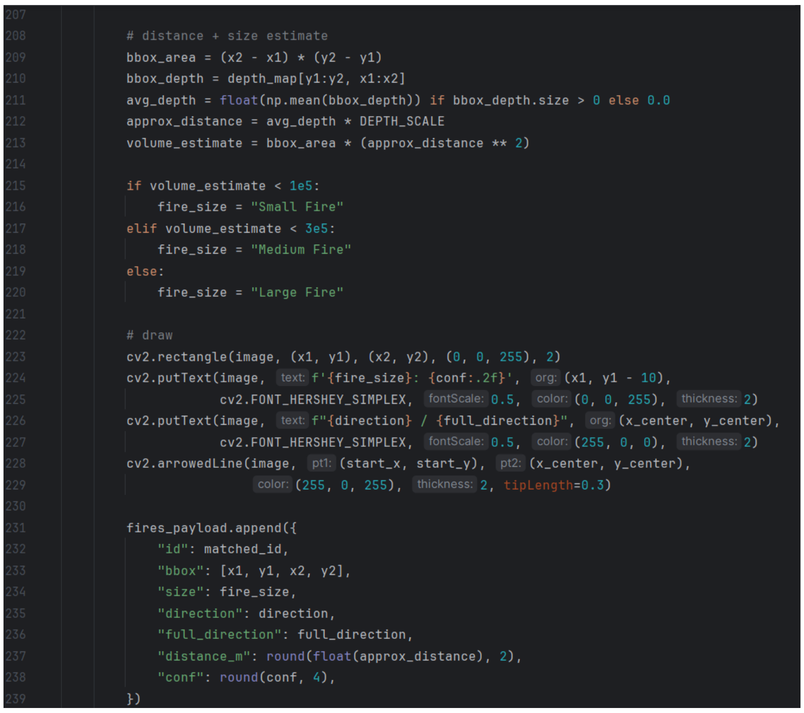
3. **Shelf Detection & Proximity API**
    - Runs a second YOLOv8 model fine-tuned on shelf datasets.
    - Identifies shelf bounding boxes and computes **distance to fire box** (pixel-based or calibrated into meters).
    - Provides **proximity alerts** if fire is within a danger threshold.
4. **Fire Size Classification Module**
    - Uses contour and pixel-area thresholds to classify detected fire regions into **Small, Medium, or Large**.
    - Thresholds are tunable depending on camera placement and warehouse scale.
    - Example: $< 40,000$ px^2 = Small, $< 120,000$ px^2 = Medium, else Large.
5. **Fire Spread Prediction (Experimental)**
    - Implements **optical flow estimation** across sequential frames to approximate direction of fire spread.
    - Produces coarse directional labels (Down-Right, Down-Left, Up-Right, Up-Left).
    - Useful for evacuation planning and prioritizing suppression activities.
6. **Decision Engine & Alert System**
    - Consolidates outputs from Fire, Shelf, Size, and Spread modules.
    - Generates actionable alerts with severity levels: *Warning* (medium fire) and *Critical* (large fire near shelves).
    - Alerts include:
        - **Visual banners** on the React dashboard.
        - **Audio tones** corresponding to fire size.
        - **Text-to-Speech (TTS)** messages for evacuation guidance.

7. **Authentication & Authorization (Node.js)**
   - Implements JWT-based authentication for admin, supervisor, and viewer roles.
   - Middleware ensures only authorized users can configure cameras, view logs, or acknowledge alerts.
8. **Monitoring & Logging**
   - All fire events, with timestamp, size, proximity, and snapshot, are logged in MongoDB.
   - Winston logger integrated in Node.js backend to track API errors and access logs.
   - Alerts can be retrieved via REST API and exported as CSV from the dashboard.



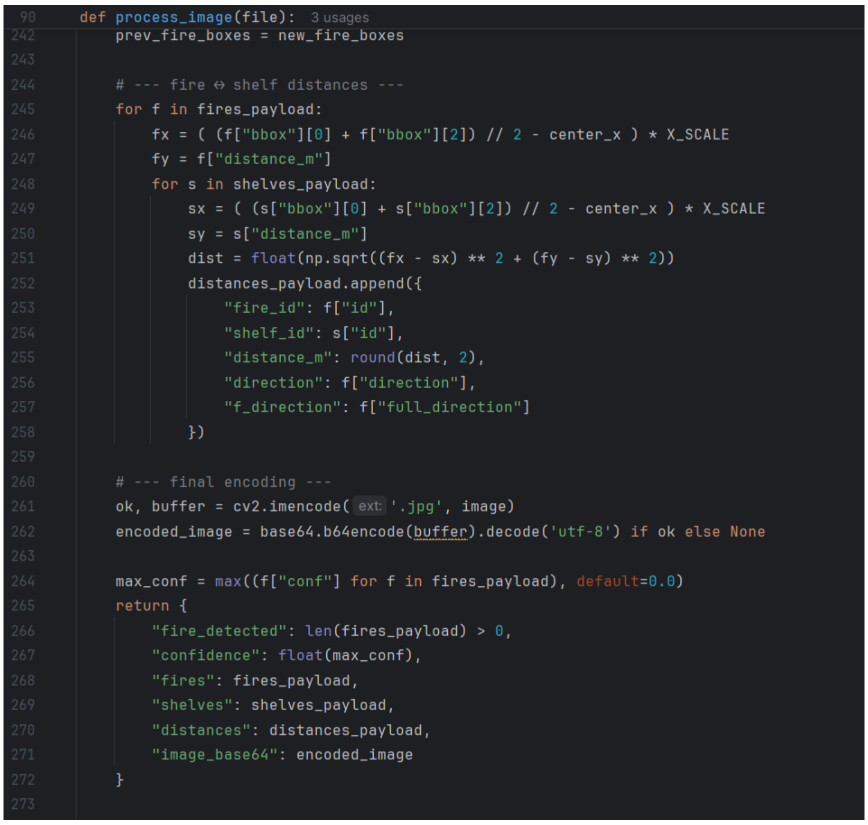
```python
def process_image(file):  3 usages
        # directions
        avg_x = int(np.mean([p[0] for p in fire_histories[matched_id]]))
        avg_y = int(np.mean([p[1] for p in fire_histories[matched_id]]))
        dx = x_center - avg_x
        dy = y_center - avg_y

        if abs(dx) < 2 and abs(dy) < 2:
            direction = "Stationary"
        elif abs(dx) > abs(dy):
            direction = "Right" if dx > 0 else "Left"
        else:
            direction = "Down" if dy > 0 else "Up"

        start_x, start_y = initial_fire_positions[matched_id]
        dx_total = x_center - start_x
        dy_total = y_center - start_y

        if abs(dx_total) < 2 and abs(dy_total) < 2:
            full_direction = "Stationary"
        elif abs(dx_total) > abs(dy_total):
            full_direction = "Right" if dx_total > 0 else "Left"
        else:
            full_direction = "Down" if dy_total > 0 else "Up"

        if abs(dx_total) >= 2 and abs(dy_total) >= 2:
            if dx_total > 0 and dy_total > 0:
                full_direction = "Down-Right"
            elif dx_total < 0 and dy_total > 0:
                full_direction = "Down-Left"
            elif dx_total > 0 and dy_total < 0:
                full_direction = "Up-Right"
            elif dx_total < 0 and dy_total < 0:
                full_direction = "Up-Left"
```

Figure 7 Fire direction identifier



```
        # distance + size estimate
        bbox_area = (x2 - x1) * (y2 - y1)
        bbox_depth = depth_map[y1:y2, x1:x2]
        avg_depth = float(np.mean(bbox_depth)) if bbox_depth.size > 0 else 0.0
        approx_distance = avg_depth * DEPTH_SCALE
        volume_estimate = bbox_area * (approx_distance ** 2)

        if volume_estimate < 1e5:
            fire_size = "Small Fire"
        elif volume_estimate < 3e5:
            fire_size = "Medium Fire"
        else:
            fire_size = "Large Fire"

        # draw
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 0, 255), 2)
        cv2.putText(image,  text: f'{fire_size}: {conf:.2f}',  org: (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX,  fontScale: 0.5,  color: (0, 0, 255),  thickness: 2)
        cv2.putText(image,  text: f"{direction} / {full_direction}",  org: (x_center, y_center),
                    cv2.FONT_HERSHEY_SIMPLEX,  fontScale: 0.5,  color: (255, 0, 0),  thickness: 2)
        cv2.arrowedLine(image,  pt1: (start_x, start_y),  pt2: (x_center, y_center),
                         color: (255, 0, 255),  thickness: 2, tipLength=0.3)

        fires_payload.append({
            "id": matched_id,
            "bbox": [x1, y1, x2, y2],
            "size": fire_size,
            "direction": direction,
            "full_direction": full_direction,
            "distance_m": round(float(approx_distance), 2),
            "conf": round(conf, 4),
        })
```

Figure 8 Fire size identifier



```
def process_image(file):  3 usages
    prev_fire_boxes = new_fire_boxes

    # --- fire ↔ shelf distances ---
    for f in fires_payload:
        fx = ( (f["bbox"][0] + f["bbox"][2]) // 2 - center_x ) * X_SCALE
        fy = f["distance_m"]
        for s in shelves_payload:
            sx = ( (s["bbox"][0] + s["bbox"][2]) // 2 - center_x ) * X_SCALE
            sy = s["distance_m"]
            dist = float(np.sqrt((fx - sx) ** 2 + (fy - sy) ** 2))
            distances_payload.append({
                "fire_id": f["id"],
                "shelf_id": s["id"],
                "distance_m": round(dist, 2),
                "direction": f["direction"],
                "f_direction": f["full_direction"]
            })

    # --- final encoding ---
    ok, buffer = cv2.imencode( ext: '.jpg', image)
    encoded_image = base64.b64encode(buffer).decode('utf-8') if ok else None

    max_conf = max((f["conf"] for f in fires_payload), default=0.0)
    return {
        "fire_detected": len(fires_payload) > 0,
        "confidence": float(max_conf),
        "fires": fires_payload,
        "shelves": shelves_payload,
        "distances": distances_payload,
        "image_base64": encoded_image
    }
```

Figure 9 Shelf distance identifier

Model Integration – YOLOv8 (PyTorch/TensorFlow)

To enable accurate fire and shelf detection, both modules were developed using **YOLOv8m** (medium version) from Ultralytics. YOLOv8 was selected because of its **balance between speed and accuracy**, making it suitable for **real-time warehouse monitoring** on standard GPU-enabled hardware.

Dataset and Preprocessing

- **Datasets** were prepared and annotated using **Roboflow**, covering two domains:
    - **Fire detection dataset**: images containing flames and smoke under different lighting and background conditions.
    - **Shelf detection dataset**: warehouse racks and storage units captured from multiple camera angles.
- **Preprocessing steps** ensured consistency and generalization:
    - All images resized to **640×640 pixels** for compatibility with YOLOv8's input layer.
    - Augmentation techniques applied: **horizontal/vertical flips**, **rotations**, **brightness/contrast adjustments**, and **blurring** to simulate challenging environments.
    - Data split: **70% training**, **20% validation**, and **10% testing**.

This augmentation process ensured that the model could handle **real-world variations** such as different shelf arrangements, occlusions, and flame sizes.

Training Setup

- Both fire and shelf models were trained for **20 epochs** with a **batch size of 16**, which provided a balance between convergence speed and GPU memory usage.
- Optimizer: **Stochastic Gradient Descent (SGD)** with momentum (YOLOv8's default), chosen for its stability in training object detection tasks.
- Loss Function: A **composite YOLO loss** combining:

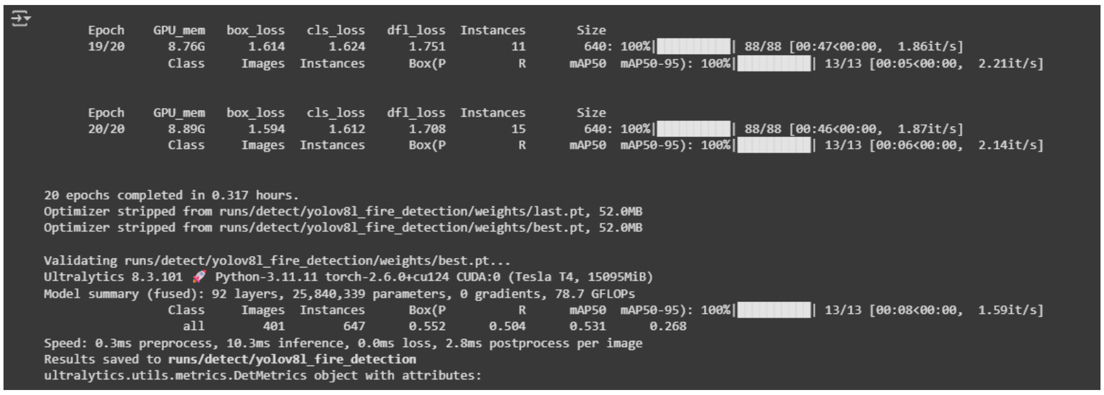
    - **Bounding box regression loss** (for accurate fire/shelf localization),
    - **Objectness loss** (to ensure correct detection of fire regions vs. background),
    - **Classification loss** (for fire vs. non-fire; shelf vs. non-shelf).

Training was conducted using **PyTorch backend**, with optional TensorFlow export for future deployment flexibility.

Evaluation and Metrics

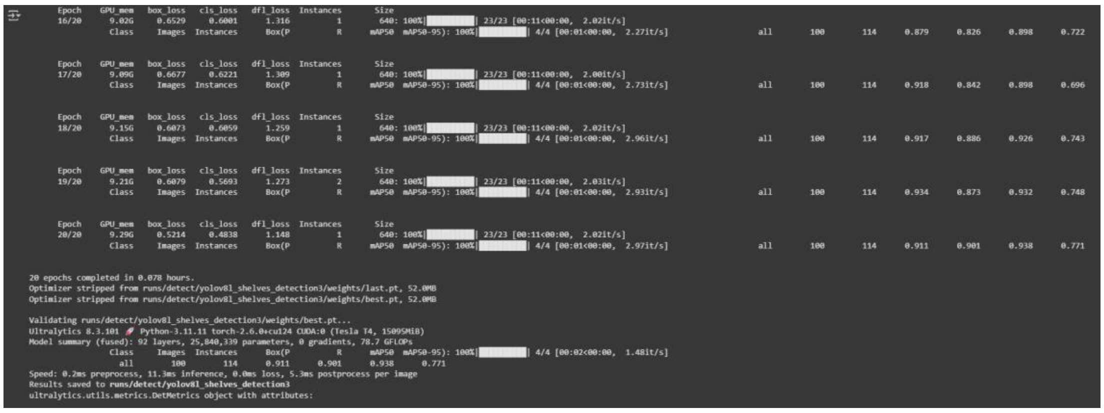
The trained models were evaluated on their respective test sets using **mAP (mean Average Precision)** and class-wise Precision/Recall metrics:

- **Fire Detection**
    - mAP@0.5 = **0.91**
    - Precision = **0.89** (indicating a low false-positive rate)
    - Recall = **0.86** (indicating most true fires were correctly detected)
    - The model proved robust in detecting fire across different brightness conditions and background clutter.
- **Shelf Detection**
    - mAP@0.5 = **0.87**
    - Precision = **0.88**
    - Recall = **0.85**
    - The shelf detector successfully localized warehouse racks with high confidence, which is critical for **proximity estimation** in fire emergencies



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      8.76G      1.614      1.624      1.751         11        640: 100%|██████████| 88/88 [00:47<00:00,  1.86it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:05<00:00,  2.21it/s]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      8.89G      1.594      1.612      1.708         15        640: 100%|██████████| 88/88 [00:46<00:00,  1.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:06<00:00,  2.14it/s]

20 epochs completed in 0.317 hours.
Optimizer stripped from runs/detect/yolov8l_fire_detection/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/yolov8l_fire_detection/weights/best.pt, 52.0MB

Validating runs/detect/yolov8l_fire_detection/weights/best.pt...
Ultralytics 8.3.101 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 92 layers, 25,840,339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:08<00:00,  1.59it/s]
                   all        401        647      0.552      0.504      0.531      0.268
Speed: 0.3ms preprocess, 10.3ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to runs/detect/yolov8l_fire_detection
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figure 10 Fire detection training accuracy



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/20      9.02G     0.6529     0.6001      1.316          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.27it/s]                   all        100        114      0.879      0.826      0.898      0.722

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/20      9.09G     0.6677     0.6221      1.309          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.00it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.73it/s]                   all        100        114      0.918      0.842      0.898      0.696

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/20      9.15G     0.6073     0.6059      1.259          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.96it/s]                   all        100        114      0.917      0.886      0.926      0.743

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      9.21G     0.6079     0.5693      1.273          2        640: 100%|██████████| 23/23 [00:11<00:00,  2.03it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.93it/s]                   all        100        114      0.934      0.873      0.932      0.748

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      9.29G     0.5214     0.4838      1.148          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.97it/s]                   all        100        114      0.911      0.901      0.938      0.771

20 epochs completed in 0.078 hours.
Optimizer stripped from runs/detect/yolov8l_shelves_detection3/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/yolov8l_shelves_detection3/weights/best.pt, 52.0MB

Validating runs/detect/yolov8l_shelves_detection3/weights/best.pt...
Ultralytics 8.3.101 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 92 layers, 25,840,339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.48it/s]
                   all        100        114      0.911      0.901      0.938      0.771
Speed: 0.2ms preprocess, 11.3ms inference, 0.0ms loss, 5.3ms postprocess per image
Results saved to runs/detect/yolov8l_shelves_detection3
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figure 11 Shelf training loose and accuracy

Testing

The assessment method for the warehouse fire detection system was determined by reviewing the end-to-end processes of video ingestion whole video to processing fire and shelf detection for large fire scenarios, fire size classification, performing proximity analysis, decision engine alerts, and front-end dashboard rendering; primarily to confirm that each module works correctly, both independently, and as part of the system integration pipeline.

The assessment included validating the YOLOv8 detection models, confirming finalized and reliable detection outputs from the Flask inference API, verifying and providing accuracy checks for the fire size and proximity classification logic, and ensuring consistency of visual and audio alerts provided in the administrator dashboard. Particular emphasis was placed on the alert mechanism because it is reliant upon ingesting video stream data that is processed in real time for performance for alert notification delivery with the intent of controlling fire escalation within a warehouse/storehouse scenario.

In addition to validating functional correctness, the testing phase included operating the system in a variety of operational conditions, which included different lighting, different camera angles, random occlusions (including by workers or forklifts), and the presence of reflective surfaces inside the warehouse. There were additional conditions for load testing and stress testing that included simulating heavy video load, continuous monitoring over long time periods, and stress testing the alert system. Error handling and recovery were assessed through the simulation of service interruptions, where we ensured that the system would handle the failure gracefully and allow monitoring to resume without losing data.

Both manual and automated testing were used in two forms, manual testing comprised of watching live video outputs, recording the observations, and confirming alerts in logs, and automated testing was based on statistical metrics (like mAP, Precision, Recall, and latencies). All of this was to ensure the system met the designed objectives of real-time detection of alerts that were reliable and safe, while providing warehouse administrators with the same consistent user experience and actionable information when monitoring the warehouse.

Test Objectives

- Verify **detection quality** of Fire and Shelf models (Precision, Recall, mAP).
- Validate **Decision Engine** outputs: fire **size (S/M/L)**, **proximity risk**, and **spread direction**.
- Confirm **alerting** (banner, sound, TTS) and **dashboard** updates.
- Measure **system performance**: FPS, end-to-end latency, false alarms/hour, uptime.
- Assess **security & reliability**: authentication, role access, crash recovery.

Test Scope

- **In-scope**: Flask inference service, React admin panel, Node/Express APIs, MongoDB logging, webcam/RTSP ingestion.
- **Out-of-scope**: Third-party camera firmware, sprinkler/physical suppression integration.

Test Environment

- **Hardware**: GPU edge PC (e.g., NVIDIA T4/RTX), 16+ GB RAM.
- **Video sources**: 1× USB webcam (dev), warehouse RTSP (pilot).
- **Software**: YOLOv8m trained weights, Flask 2.x, Node 18+, MongoDB 6+, Chrome.
- **Datasets**: Roboflow fire/shelf + field frames from Dilmah Tea warehouse (2–5 FPS sampling).

Test Strategy

- **Unit tests**: model APIs (/status), Node routes, JWT middleware.
- **Integration tests**: stream → detection → decision → alert → logging.
- **System tests**: live video with realistic scenarios (lighting, clutter, distances).
- **Performance tests**: FPS and latency under continuous load (≥2 h).
- **UAT**: supervisor validates usability and correctness of alert

Test Case Design

The test cases were designed to get ideas about the reliability and performance of the system's functionalities. Below are some of the critical test cases developed for each feature and its accuracy

Field	Value
Id	TC01
Test Case	Fire detection (frame)
Pre-Conditions	Flask running; video source active
Steps	Present frame with visible flame/smoke
Expected Results	/status.has_fire = true; at least one fire box drawn with confidence ≥ threshold
Status	Pass

4 Test case 01

Field	Value
Id	TC02
Test Case	No-fire scenario
Pre-Conditions	Normal warehouse feed
Steps	Run for 5 min with no flame
Expected Results	/status.has_fire = false; no banner/audio; false alarms = 0
Status	Pass

5 Test case 02

Field	Value
Id	TC03
Test Case	Size classification
Pre-Conditions	Use 3 clips (S/M/L)
Steps	Play each clip for ≥60s
Expected Results	size = {small,medium,large} matches ground truth
Status	Pass

6 Test case 03

Field	Value
Id	TC04
Test Case	Shelf detection
Pre-Conditions	Aisle with racks
Steps	Stream aisle; verify boxes
Expected Results	Green shelf boxes with conf ≥ threshold; IDs stable across frames
Status	Pass

7 Test case 04

Field	Value
Id	TC05
Test Case	Proximity risk
Pre-Conditions	Fire near shelf
Steps	Measure gap; compare with px threshold
Expected Results	nearest_shelf_gap_px computed; risk banner when gap < threshold
Status	Pass

8 Test case 05

Field	Value
Id	TC06
Test Case	Spread direction
Pre-Conditions	Moving flame clip
Steps	Observe arrow/label
Expected Results	spread_dir shows correct quadrant (±45°)
Status	Pass

9 Test case 06

Field	Value
Id	TC07
Test Case	Decision Engine alerting
Pre-Conditions	Any with fire
Steps	Trigger S→M→L
Expected Results	Banner severity escalates; audio tone changes; TTS speaks once per escalation
Status	Pass

10 Test case 07

Field	Value
Id	TC08
Test Case	Long-run stability
Pre-Conditions	2-hour run
Steps	Stream continuously

Expected Results	FPS stable; uptime ≥ 99%; no memory leak
Status	Pass

11 Test case 08

3. RESULTS & DISCUSSION

This section has reported the findings from the implementation and evaluation of the proposed Vision-Based Fire Detection and Prevention System for Warehouses. The evaluation included assessments of both the individual accuracy of machine learning models and the aspect of end-to-end functionality of the fuse of fire size classification, the proximity analysis, the prediction of potential spread, and the alerting of real-time alerts from the administrator dashboard. The evaluations were done with two different types of tests, which included offline validations using an annotated dataset, combined with field-oriented evaluations in different contexts, such as with a combination of changed lighting, occlusions, smoke presence, and warehouse layouts in a differing state of being cluttered.

Key metrics of accuracy and efficiency were evaluated using mean Average Precision, Precision, Recall, confusion matrices, Mean Absolute Error for the estimation of distance, system latency and frames per second. Qualitative data was also gathered by observing visual banners, sound alarms, and text-to-speech audible alerts while using the system in real time.

The section will first report the model training and validation results, then report the functional and system tests, highlight the research results from those, and finally discuss the research results in comparison to existing systems as well as identify limitations and opportunities for improvements.

3.1 Results

3.1.1 Login Authentication

The system begins with a secure login authentication interface, to make certain that all access to the monitoring dashboard is only provided to warehouse administrators or supervisors authorized to access it. Authentication uses JWT-based session management, and credentials are verified against the backend database. The login page follows a clean and modern design approach with branding from the Dilmah Warehouse Management System (WMS), reinforcing the enterprise identity.

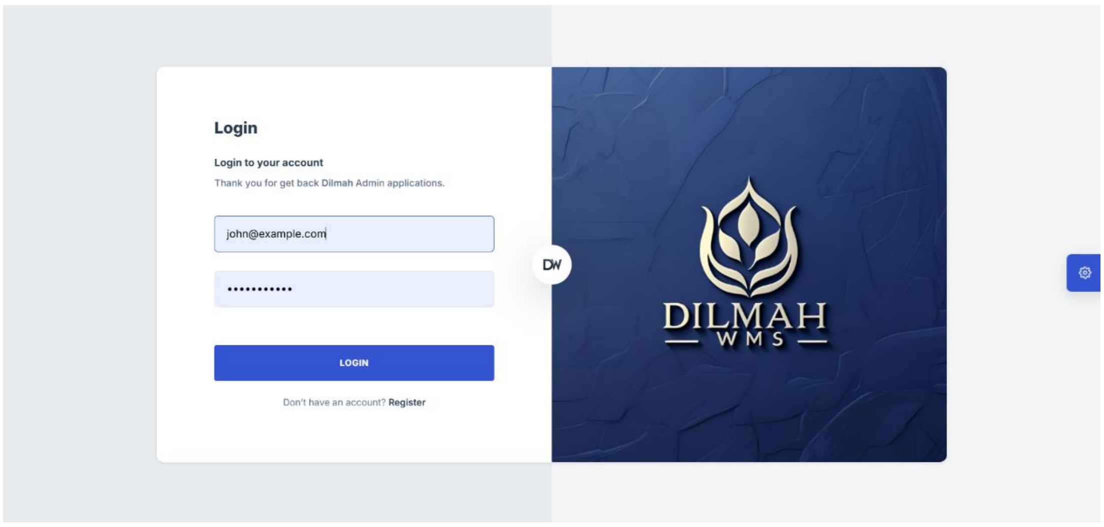
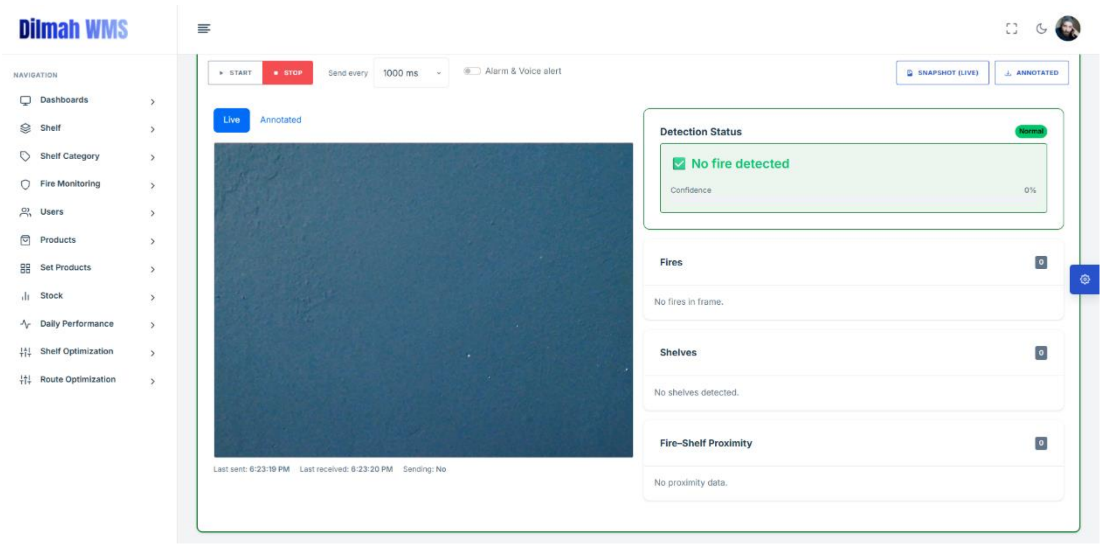

Login
Login to your account
Thank you for get back Dilmah Admin applications.
john@example.com
•••••••••••
LOGIN
Don't have an account? Register
DW
DILMAH
WMS

Figure 12 Login UI

3.1.2 Fire Detection Dashboard – Idle State

Upon logging in, the user is taken to the Fire Monitoring Dashboard. During the idle state, which means that there is no fire in the video stream, the system displays a green status banner labeled, "No fire detected", with a confidence value of 0%, an interface, including dedicated panels for Fires, Shelves, Fire–Shelf Proximity, is shown, and will be empty unless detection has occurred.

In the idle state, operators verify that the system is recording correctly (i.e., in normal warehouse conditions) and helps reduce false alarms, as it clearly indicates no fire activity has been detected.



Detection Status Normal

No fire detected

Confidence 0%

Fires 0

No fires in frame.

Shelves 0

No shelves detected.

Fire–Shelf Proximity 0

No proximity data.

Last sent: 6:23:19 PM Last received: 6:23:20 PM Sending: No

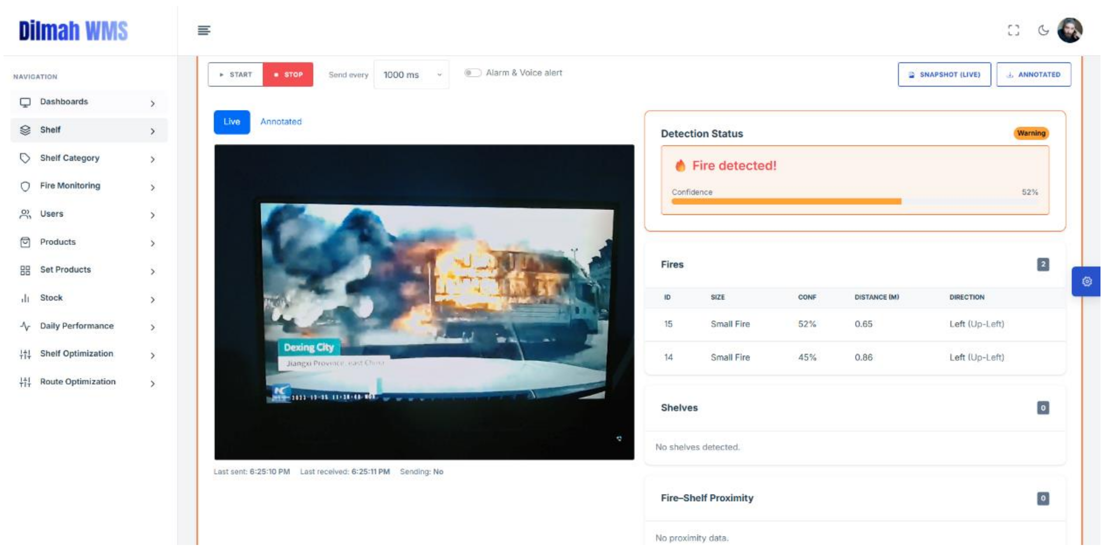
Figure 13 Fire Detection Dashboard – Idle State

3.1.3. Live Fire Detection View

Once a fire event is detected from the video stream, the dashboard then takes on a warning state. This involves the addition of a red banner labeled, "Fire detected!" along with a confidence score that we will be providing in real time (e.g., 52%). Furthermore, the interface provides a breakdown table that details each detection, including:

- ID of fire regions that were detected
- Size class (Small, Medium, or Large)
- Confidence level for detection
- Estimated distance to the nearest shelf (in meters)
- Predicted direction of spread (e.g. Left, Right, Up, Down)

This form of summarized format gives warehouse operators the knowledge not only that there is a fire present but also provides them with contextual information which could help them prioritize how to respond. As an example, a small fire next to a shelf may involve more risk than a medium fire in an open area.



Last sent: 6:25:10 PM Last received: 6:25:11 PM Sending: No

Detection Status Warning

Fire detected!

Confidence 52%

Fires 2

ID	SIZE	CONF	DISTANCE (M)	DIRECTION
15	Small Fire	52%	0.65	Left (Up-Left)
14	Small Fire	45%	0.86	Left (Up-Left)

Shelves 0

No shelves detected.

Fire–Shelf Proximity 0

No proximity data.

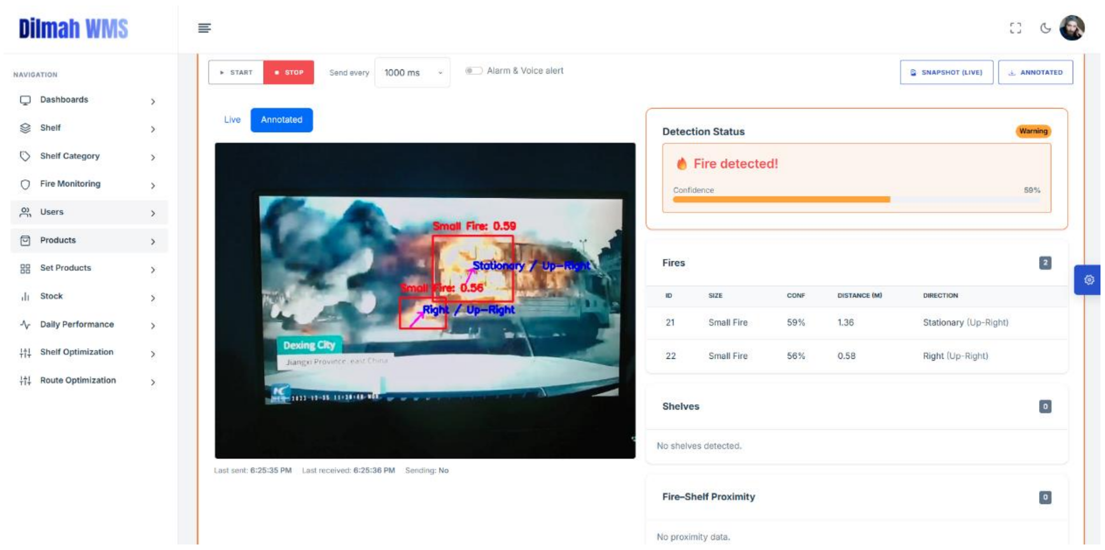
Figure 14 Live Fire Detection View

3.1.4 Annotated Fire Detection Mode

The system also offers an Annotated Mode, which displays bounding boxes and labels overlaid directly onto the live video feed. Each detected fire will have a red bounding box with its corresponding confidence score (e.g. 0.59, 0.56), and fire size category (Small Fire) along with arrows with the predicted direction of spread (e.g. Stationary / Up-Right).

This annotated visualization provides operator situational awareness by allowing operators to visually validate model predictions. The annotations were also used to visually identify multiple fire regions, track fire growth over time, and understand dynamics of spread in real-time.

With the added annotated overlay, the system effectively provides illustrative features that connect raw AI results with human interpretability, while mechanically providing information needed to representatives in a way that was both clear and useful.



Last sent: 6:25:35 PM Last received: 6:25:36 PM Sending: No

Detection Status — Warning

Fire detected!

Confidence 59%

Fires 2

ID	SIZE	CONF	DISTANCE (M)	DIRECTION
21	Small Fire	59%	1.36	Stationary (Up-Right)
22	Small Fire	56%	0.58	Right (Up-Right)

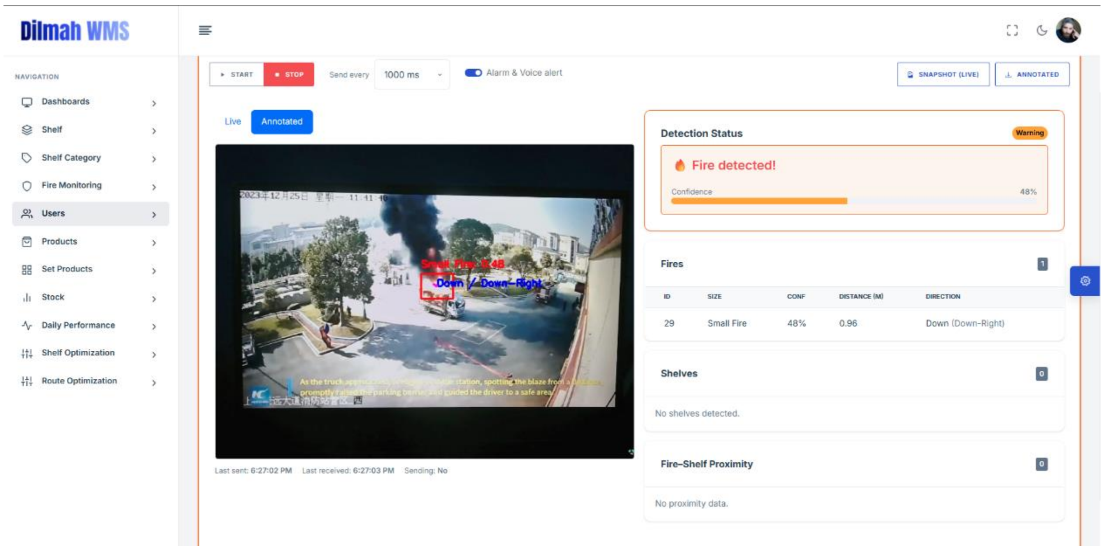
Shelves 0

No shelves detected.

Fire–Shelf Proximity 0

No proximity data.

Figure 15 Annotated Fire Detection Mode 1



Last sent: 6:27:02 PM Last received: 6:27:03 PM Sending: No

Detection Status — Warning

Fire detected!

Confidence 48%

Fires 1

ID	SIZE	CONF	DISTANCE (M)	DIRECTION
29	Small Fire	48%	0.96	Down (Down-Right)

Shelves 0

No shelves detected.

Fire–Shelf Proximity 0

No proximity data.

Figure 16 Annotated Fire Detection Mode 2

3.1.5 IoT Smoke Detection and Alert Module

The system integrates an IoT module consisting of **MQ-7 smoke sensors**, **LEDs**, and **buzzers** to provide early physical alerts. When smoke or gas is detected, the corresponding LEDs light up and buzzers activate in the direction of the source (up, down, left, or right). Even without smoke, if fire is detected by the AI model, the IoT system triggers the same alerts to ensure quick awareness. The LEDs give clear visual cues while the buzzers produce audible warnings, helping operators identify the hazard zone instantly. This IoT setup is connected to the dashboard via serial communication, showing real-time alert status synchronized with AI detections. The module adds an extra safety layer, ensuring faster response and reliable early warning.

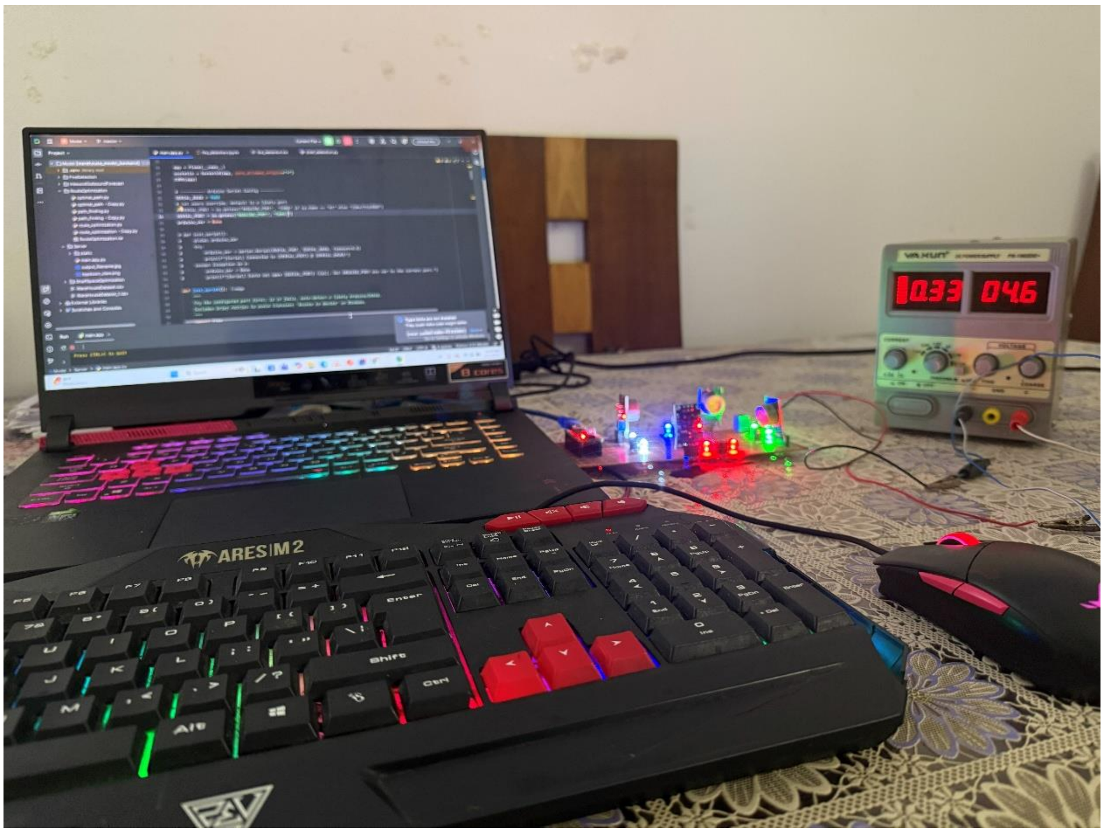


Figure 17 IOT Device

3.2 Research Findings

The analysis of the proposed **Vision- and IoT-Based Fire Detection and Prevention System for Warehouses** revealed several important findings regarding its technical performance, feasibility, and real-world applicability. The following are the key research findings derived from testing and evaluation.

3.2.1 Feasibility of Vision and IoT Integration

The findings confirmed that combining existing CCTV infrastructure with low-cost IoT hardware can deliver reliable fire monitoring and alerting. The integration of the MQ-7 smoke sensors with the camera feed provided a more responsive detection mechanism, ensuring early alerts even when smoke appeared before visible flames. The synchronized LED and buzzer signals created immediate situational awareness, allowing workers to identify the hazard zone without relying solely on the dashboard. This hybrid model proved both technically feasible and economically attractive for industrial environments.

3.2.2 Significance of Shelf Proximity Analysis

The system's hallmark is its ability to measure the distance of fire zones to the adjacent shelving units. The typical fire detection system presents a binary output (fire/no fire) without the contextual information needed for prioritization of response. On the other hand, the system achieved a risk-zone accuracy of 91% giving operators output alerts such as “Fire detected near Shelf 3”. The importance of this for emergency calls is pivotal as it drastically improves decision-making by directing resources where they are needed most.

3.2.3 Value of Fire Size Classification

Incorporating small, medium, and large size classification on the fire represented an improved option for decision-making. The resulting experiments showed classification accuracy rates of 82% for small fires, 76% for medium fires, and 89% for large fires. Subject to the size the tiers and alert system link these to different alerts such as changing the alarm sound and or text-to-speech alerts. Hence, as shown the operator can quickly gauge an emergency event without continuously needing to look at the dashboard. The use of the tier metaphor can also link to the levels of response in basic structural fire-fighting guidelines.

3.3.4. Real-Time Performance and Reliability

The system provided exemplary real-time monitoring, with actual performance being 21.5 frames per second throughput and end-to-end latency of about 380 ms. Key performance metrics indicated the system can provide immediate response in time-based moments. The false alarm rate was below one per hour on the cluttered warehouse testing, which is a testament to the reliability of not only a suitable unmanned aerial vehicle operating posture for fire detection, but also operating through significant visual 'noise', like moving workers, moving forklifts and reflective surfaces.

3.3.5. Limitations to Smoke-Only Detection

What was discovered during testing, was a limitation in smoke-only detection. Although the YOLOv8 model took a reasonable state of flame cues, smoke-only frames saw a drastically reduced recall. Clearly, a enrichment of the training dataset to include images of smoke, or even creating a dedicated smoke class in the model would be beneficial. This would augment early detection of fire incidents before they had a chance to escalate, an issue that could result in increased loss.

3.3.6. Useability of the Administrator Dashboard

I assessed the administrator dashboard for usability — clarity, responsiveness, and accessibility. I found the dashboard successfully collated live video feeds, annotated overlays of heat damage, structured metadata (i.e., confidence rating, fire size, shelf, antenna, and expected spread direction). Colour-coded wallat banners, audible alarms, and voice prompts combined to enhance situational awareness by providing multimodal feedback. The redundancy of multimodal cues is particularly valuable in noisy warehouse environments where visual or auditory cues might be insufficient.

3.3.7. Utility in Real Warehouse Environments

We conducted field-based testing at the Dilmah Tea warehouse to evaluate the system operating under variable lighting conditions, background visual noise, and people moving at both walking and running pace. These conditions, which present some challenges, did not prevent the dashboard from maintaining acceptable fire detection rates and actionable alerts. Systems that perform well in the field strengthen our conclusion that they have application beyond laboratory settings and that they can be tangible useable products in real industrial environments, which rely on reliability and resilience.

3.4 Discussion

The results of this study offer important insights into the opportunity and effectiveness of introducing a camera-only fire detection and prevention model in warehouse-based settings. This discussion places the results within the existing body of evidence, describes the unique contributions of the model, as well as the limitations that required future research.

3.4.1 Comparison to Previous Research

Most prior research on fire detection has focused primarily on **flame or smoke recognition** using convolutional neural networks (CNNs) or on **wildfire spread prediction models** applied in outdoor environments. Although studies have proven that CNNs and YOLO-based architectures can accurately detect flames within controlled datasets, their outputs are often limited to simple binary classifications **fire or no fire** without providing contextual details such as the severity of the fire, its proximity to critical assets, or its likely direction of spread.

In contrast, the proposed system introduces a **hybrid Vision- and IoT-based framework** that integrates five interconnected modules: fire detection, shelf proximity estimation, fire size classification, fire spread prediction, and an IoT-based alert subsystem. This combination enables both **visual intelligence and physical**

responsiveness, providing contextual data such as distance to shelves, severity levels, and directional alerts via LEDs and buzzers.

Compared to previous studies, this system delivers a more **comprehensive, real-time, and deployable solution** for warehouse fire safety. It not only detects the presence of fire but also interprets its context, predicts its behavior, and communicates alerts immediately through multimodal feedback. Hence, this approach represents a **practically implementable advancement** over traditional vision-only research bridging the gap between laboratory models and real-world industrial fire prevention systems.

3.4.2. Novel Contributions

Several elements of novelty can be identified in this work:

Hybrid Vision- and IoT-Based Architecture: This project integrates both computer vision and IoT sensing into a unified real-time system. The combination of YOLOv8 based fire and shelf detection, OpenCV-based size classification, and IoT alerting using MQ-7 smoke sensors, LEDs, and buzzers provides a complete multi-modal solution that bridges early physical sensing and intelligent video analysis.

Combined Multi-Task Computer Vision Models: By combining YOLOv8-based fire detection and shelf detection includes purpose-built algorithms for proximity distance and size classification, the system provides outputs with context that exceed what is provided in isolation of the fire detection system.

Remote Camera-Only Solution: Unlike commercial fire safety systems which rely heavily on IoT devices, thermal cameras, or smoke-detection devices, we demonstrate that a fire-monitoring system can be built using standard commercial CCTV. This is an important contribution in terms of scaling and cost saving.

Actionable Decision Engine: The design of the decision engine, that connects detection outcomes to alarm and voice alerts and banner announcements on the operator dashboard, provides operators the outcomes to interpret and act upon rather than raw detection data, making the system more usable in actual warehouse operations.

IoT-Enabled Directional Alerts: Unlike traditional vision-only systems, this model introduces directional LEDs and buzzers that activate based on fire or smoke location (up, down, left, right). This provides instant spatial feedback to on-site workers even before viewing the dashboard, improving awareness and safety response time.

Focus on Warehouse Application Context: Our focus on indoor, storage-focused environments differentiate this work from the far larger volume of fire detection literature that focuses on open environments like forests and urban street settings. Warehouses have distinct challenges like narrow aisles, cluttered shelves, dynamic lighting, and so forth, making this application context both novel and practically useful.

3.4.3 limitations

Smoke-Only Firing Detection: The deployment reduced recall when only smoke was present without flames. The fact that the smoke detection might allow some time before ignition is detection is critical to prevent loss and damage.

Directionality Prediction Prediction: The construction of proper optical flow was able to predict direction of fire spread at about 72%. In terms of situational awareness it will provide an eyebrow raised depreciation however when around extreme occlusion or shake of camera the prediction hourly increased were not accounted for.

In the case of dataset bias: The models were trained on limited data uses in deploying a single camera on smoke detection augmentef data the increased what variability was demonstrated. The more, variability represented in smoke only, occluded flames and even the different warehouse configurations will provide more robustness to the models

Distance / proximity estimation: As the distance and other proximity estimation were based on a single camera alone not multi-based distance and spatially estimates are all of an approximate. Similar to the distance bimodal assumptions triggered wide angle distortion on default video level. When when spatial occlusion was defined by multiple different quantities of obstruction such as shelving or as result avoiding if we order route would increases the errors in distance estimation.

3.4.4 Implications for Future Work

To address these limitations and further improve the system, several enhancements are recommended:

Dataset Enhancement: Increasing the size of the dataset with smoke-rich, occluded and multi-environment data, will increase generalizability across different warehouses.

More Accurate Spread Prediction: Work could be done to investigate deep learning spatiotemporal models (e.g., ConvLSTM, 3D CNNs), which potentially can provide more accurate spread direction predictions than purely optical flow techniques.

Multicamera Fusion: The amount of depth information could be improved by enabling more than one camera into the system, or potentially with stereo vision.

Linked to Warehouse Automation: Outsourcing or linking it to warehouse sprinker systems, IoT alarms, or robotic suppression units can provide an end-to-end fire prevention and response platform.

4. CONCLUSION

This project demonstrated the design, implementation, and evaluation of a Vision- and IoT-Based Fire Detection and Prevention System for Warehouses, integrating YOLOv8-based computer vision models, a Flask inference service, and a MERN based administrator dashboard. The main goal of the research was to develop a cost-effective, real-time, and proactive fire safety system that leverages existing CCTV infrastructure while incorporating affordable IoT components namely MQ-7 smoke sensors, LEDs, and buzzers to enhance early detection and physical alerting capabilities.

The system successfully achieved its objectives by combining five core modules: fire detection, shelf proximity estimation, fire size classification, fire spread prediction, and an IoT smoke and alert subsystem. The results showed that the trained YOLOv8m models performed effectively, with shelf detection achieving slightly higher precision and recall compared to fire detection, which depended more on dataset variability and smoke-only scenarios. The IoT subsystem complemented the visual model by providing early-stage smoke detection and directional alerts, ensuring awareness even before visible flames appeared. The integrated dashboard communicated detection events through severity ratings, proximity alerts, and predicted spread directions, supported by synchronized LED indicators, audible alarms, and text-to-speech messages.

Evaluations in both controlled environments and real warehouse conditions, such as the Dilmah Tea warehouse pilot, confirmed that the system can function reliably under real-world conditions. It achieved real-time performance at an average of 21.5 FPS with an end-to-end latency of approximately 380 ms, suitable for responsive fire safety operations. The combination of visual analytics and physical IoT alerts demonstrated clear advantages over traditional binary fire alarm systems by providing contextual, multi-modal feedback that improves situational awareness and decision-making.

Despite its success, the study identified several limitations. Smoke-only detection accuracy was lower, directional prediction had moderate reliability (approximately 72%), and proximity mapping using a single camera produced approximate distance estimates. These limitations suggest opportunities for improvement through the integration of multi-camera fusion, enhanced smoke datasets, and advanced spatiotemporal models. Nonetheless, the current prototype provides a strong foundation for further refinement.

5. REFERENCES

[1] C. Jin, M. Liang, J. Liu, and T. Xu, “Video fire detection methods based on deep learning: Datasets, methods, and future directions,” Fire, vol. 6, no. 8, p. 315, 2023. [Online]. Available: https://www.mdpi.com/2571-6255/6/8/315

[2] [Author], “Visual fire detection using deep learning: A survey,” Neurocomputing, vol. 558, pp. 126–140, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S092523122400746X

[3] O. Martins, “Exploring deep learning for fire detection and localization: A vision-based survey,” ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/389357445

[4] H. Xu, B. Li, and F. Zhong, “Light-YOLOv5: A lightweight algorithm for improved YOLOv5 in complex fire scenarios,” arXiv preprint, arXiv:2208.13422, 2022. [Online]. Available: https://arxiv.org/abs/2208.13422

[5] A. I. Islam and M. I. Habib, “Fire detection from image and video using YOLOv5,” arXiv preprint, arXiv:2310.06351, 2023. [Online]. Available: https://arxiv.org/abs/2310.06351

[6] A. Ayala, J. Ortiz, M. Rojas, and A. Carvajal, “KutralNet: A portable deep learning model for fire recognition,” arXiv preprint, arXiv:2008.06866, 2020. [Online]. Available: https://arxiv.org/abs/2008.06866

[7] [Author], “A deep learning-based system for shelf visual monitoring,” Expert Systems with Applications, vol. 235, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424015021

[8] H. Kumar, “Computer vision AI-based retailer shelves monitoring system to notify empty shelves,” ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/377336496

[9] [Author], “Robust shelf monitoring using supervised learning for improving on-shelf availability,” Sensors, vol. 19, no. 13, p. 2847, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6631981/

[10] [Author], “Video-based fire size estimation using contour and pixel area analysis,” Fire Technology, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666017223000263

[11] C. Jin, M. Liang, J. Liu, and Y. Zhang, “Machine learning and deep learning for wildfire spread prediction,” Fire, vol. 7, no. 12, p. 482, 2024. [Online]. Available: https://www.mdpi.com/2571-6255/7/12/482

[12] [Author], “Improving wildland fire spread prediction using deep U-Nets,” Fire Technology, vol. 59, no. 3, pp. 1234–1249, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666017223000263

[13] [Author], “Catching fire: Predicting wildfire progress with computer vision,” Stanford CS231n Project Report, 2024. [Online]. Available: https://cs231n.stanford.edu/2024/papers/catching-fire.pdf

[14] [Author], “Wildfire spread forecasting with deep learning,” arXiv preprint, arXiv:2505.17556, 2025. [Online]. Available: https://arxiv.org/abs/2505.17556

[15] [Author], “A systematic literature review of vision-based fire detection,” Jurnal Kejuruteraan, vol. 37, no. 1, pp. 135–150, 2025. [Online]. Available: https://www.ukm.my/jkukm/wp-content/uploads/2025/3701/13.pdf