IBM Project Report

On

Real-Time Monitoring and Evaluation System for Fire Department Applications

Developed By: -

Munish Patwa (21162121017)

Aniket Panjwani(22162102002)

Astik Saxena (21162171002)

Guided By:-

Prof. Sulabh Bhatt(Internal)

Mr. Nirav Raj(External)

Submitted to

Faculty of Engineering and Technology

Institute of Computer Technology

Ganpat University

Year - 2025

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

DIAMOND QS I-GAUGE INDIAN UNIVERSITY RATINGS | NAAC A Grade Accredited University | INSTITUTION'S INNOVATION COUNCIL | GSIRF

CERTIFICATE

This is to certify that the **IBM** Project work entitled **"Real-Time Monitoring and Evaluation System for Fire Department Applications"** by Munish Patwa (Enrolment No. 21162121017) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for the award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

Place: ICT - GUNI

Date: 22 February 2025

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

DIAMOND QS I-GAUGE INDIAN UNIVERSITY RATINGS

National Assessment and Accreditation Council NAAC A Grade Accredited University

INSTITUTION'S INNOVATION COUNCIL (Ministry of Education Initiative)

GSIRF

CERTIFICATE

This is to certify that the **IBM** Project work entitled **"Real-Time Monitoring and Evaluation System for Fire Department Applications"** by Astik Saxena (Enrolment No. 21162171002) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for the award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

Place: ICT - GUNI

Date: 22 February 2025

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

DIAMOND QS I-GAUGE INDIAN UNIVERSITY RATINGS | National Assessment and Accreditation Council NAAC A Grade Accredited University | INSTITUTION'S INNOVATION COUNCIL (Ministry of Education Initiative) | GSIRF

CERTIFICATE

This is to certify that the **IBM** Project work entitled **"Real-Time Monitoring and Evaluation System for Fire Department Applications"** by Aniket Panjwani (Enrolment No. 22162102002) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for the award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

Place: ICT - GUNI

Date: 22 February 2025

ACKNOWLEDGEMENT

The IBM/Industry Internship project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank Dr. Rohit Patel, Principal, ICT, and Prof. Dharmesh Darji, Head, ICT who gave us an opportunity to undertake this project. My thanks to Prof. Sulabh Bhatt, and Mr. Nirav Raj (Internal & External Guides) for their guidance in project work on Real-Time Monitoring and Evaluation System for Fire Department Applications, despite being extraordinarily busy with academics, taking time out to hear, guide, and keep us on the correct path. We do not know where we would have been without his/her help.

MUNISH PATWA (Enrollment No: 21162121017)
ASTIK SAXENA (Enrollment No:21162171002)
ANIKET PANJWANI (Enrollment No:22162102002)

ABSTRACT

The Real-time Fire Detection and Notification System is designed to enhance safety measures by providing immediate alerts upon detecting fire incidents in various environments. The primary goal of this project is to develop an intelligent system capable of detecting fire in real time using computer vision techniques and notifying relevant authorities or individuals through SMS and email alerts.

The system leverages YOLOv8 (You Only Look Once), a state-of-the-art deep learning model for real-time object detection, integrated with OpenCV for image processing. The fire detection model continuously monitors the video feed to identify potential fire events, providing a proactive approach to preventing damage. Upon detecting fire, the system triggers notifications through Twilio for SMS and SMTP protocol for email, ensuring timely alerts.

The frontend of the application is developed using React.js, offering a user-friendly interface for visualization, control, and status monitoring. On the backend, FastAPI is employed for handling API requests efficiently and enabling seamless communication between the frontend, backend, and the notification services. For persistent data storage, Firebase is utilized, ensuring secure, real-time data management and access.

This system aims to improve emergency response times, mitigate damage caused by fire incidents, and promote greater safety awareness in both industrial and residential settings. The project is nearing completion, with all major components developed and integrated successfully. Future work includes optimization of detection accuracy and system performance.

INDEX

CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

Fire incidents are a significant threat to life, property, and the environment, making it crucial to have effective fire detection and prevention systems in place. Traditional fire detection systems, often relying on manual monitoring or basic alarm systems, can be slow to react and may fail to detect fires in their early stages. This delay in response can lead to catastrophic damage and loss. With the rapid advancements in computer vision, machine learning, and IoT technologies, there is a tremendous opportunity to improve fire detection and response systems, making them more efficient, real-time, and automated.

This project focuses on developing a Real-time Fire Detection and Notification System designed to address these challenges. By using YOLOv8, a powerful object detection model, combined with OpenCV for real-time image processing, the system automatically detects the presence of fire in video feeds. When a fire is detected, the system triggers notifications via Twilio (for SMS) and SMTP (for email) to alert relevant personnel or emergency responders, enabling faster action and potentially preventing further damage.

The system architecture is divided into several key components:

- Frontend: Built using React.js, providing an interactive and intuitive user interface for system monitoring and status updates.
- Backend: Developed with FastAPI, ensuring efficient and scalable API handling for real-time operations.
- Storage: Firebase is used for storing data, including logs and alerts, ensuring secure and real-time access across multiple devices.

The primary objective of this project is to provide a fully functional, automated fire detection and alert system that works in real-time to minimize the potential harm caused by fires. The system is designed to be scalable, efficient, and easy to deploy across various environments, from industrial facilities to residential properties. By integrating cutting-edge technologies, the project aims to significantly enhance safety protocols and emergency response times.

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to develop a real-time fire detection and notification system that automatically detects fire incidents and alerts the relevant authorities or individuals. This system aims to enhance safety measures in various settings, including residential, commercial, and industrial environments, by using advanced computer vision and machine learning techniques.

Key Features and Functionalities:

1. Real-time Fire Detection:
    - Utilize YOLOv8, a state-of-the-art deep learning model, for detecting fire in video streams.
    - Integrate OpenCV for processing and analyzing video frames in real time to identify fire-related features (e.g., flames, smoke).
2. Notification System:
    - SMS Alerts: Integrate Twilio API to send immediate SMS notifications to specified contacts (e.g., emergency services, building management, or owners) when fire is detected.
    - Email Alerts: Utilize the SMTP protocol to send email notifications, including detailed information about the detected fire and its location.
3. User Interface:
    - Develop a React.js-based frontend for system monitoring, where users can track the status of fire detection, manage notifications, and visualize the results.
    - Provide a dashboard for live monitoring of the video feed, fire detection alerts, and overall system status.
4. Backend & API Communication:
    - Use FastAPI to handle communication between the frontend and backend, ensuring fast and efficient real-time data processing.
    - The backend processes the fire detection output and triggers notifications accordingly.
5. Data Storage & Management:
    - Integrate Firebase for real-time data storage, such as storing detection logs, alert history, and system performance data.
    - Ensure secure access to the system data and provide analytics capabilities (e.g., historical data for fire detection).
6. Scalability & Deployment:
    - Design the system to be scalable and adaptable for deployment in various environments, including large-scale industrial complexes, office buildings, and residential properties.
    - Ensure the system can be easily deployed with minimal setup and configuration.

Limitations and Exclusions:

- This project will not include physical hardware components such as cameras or sensors. The fire detection will solely rely on video feeds from existing security cameras.
- The system will focus on fire detection and notifications only; additional emergency response features (e.g., automated sprinkler systems) are out of scope.
- The project will be limited to fire detection using visual data and will not extend to other types of environmental hazards like gas leaks or earthquakes.

Future Enhancements:

- Integration of additional sensors such as smoke detectors and temperature sensors to improve detection accuracy.
- Expansion to include automated response mechanisms, such as triggering alarms or activating emergency protocols in buildings.

The Real-time Fire Detection and Notification System will not only address immediate fire detection needs but will also contribute to the broader goal of enhancing safety through automation, enabling quicker and more reliable emergency responses.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i5 / AMD Ryzen 3 (or equivalent)
RAM	8GB or more
HDD/SSD	50GB of free space (SSD preferred for faster access)
GPU	NVIDIA GTX 1060 or better (for YOLOv8 model inference)
Network	Stable internet connection for real-time monitoring & notifications
Web Camera/Video Feed	Standard security camera or webcam (for video input)

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

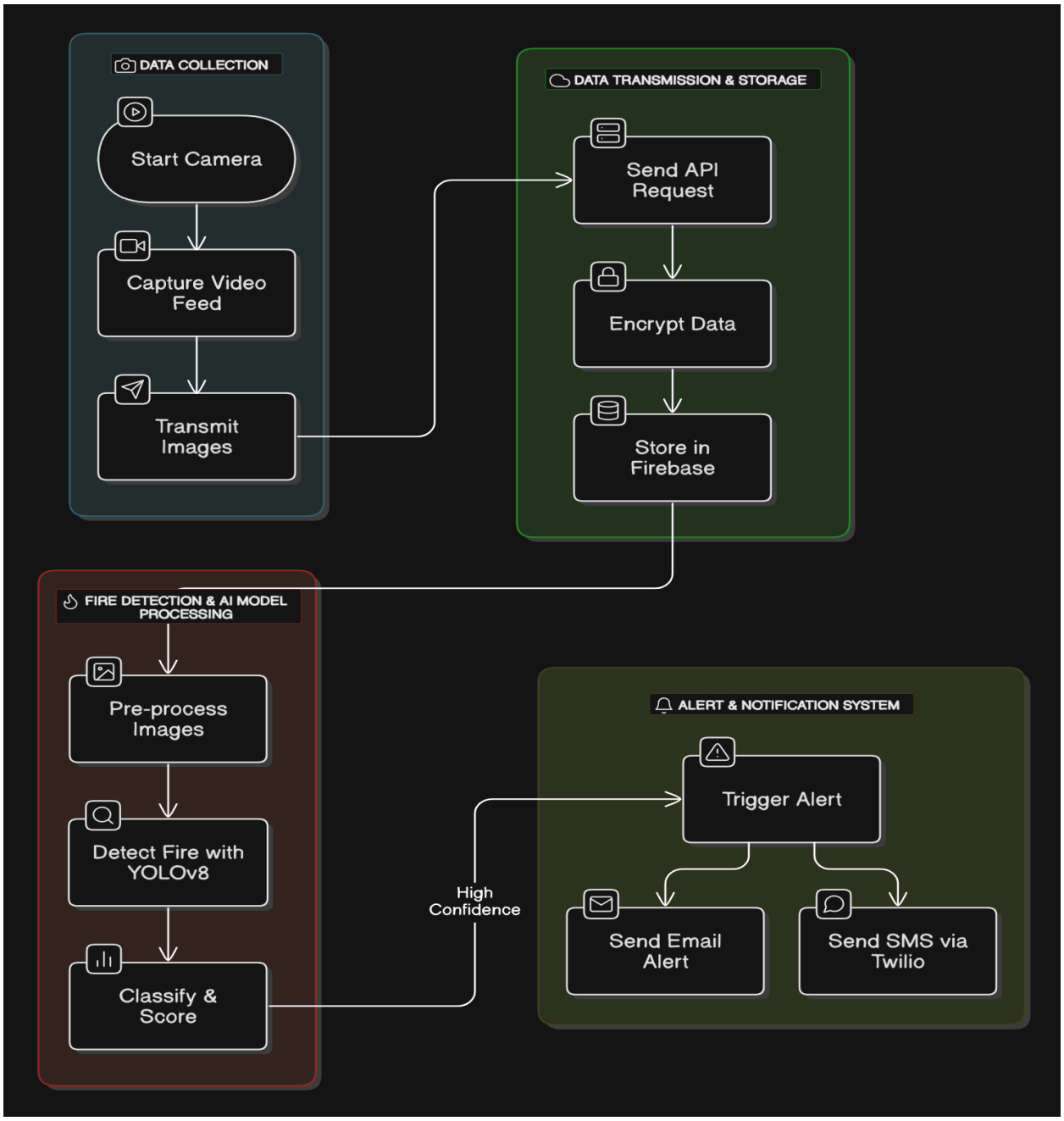
Component	Specification
Operating System	Windows, macOS, or Linux (modern versions supported)
Programming Language	Python 3.8+ (for backend and AI-based fire detection)
Frontend Framework	React.js (for user interface)
Backend Framework	FastAPI (for backend and API communication)
Libraries/Tools	OpenCV, YOLOv8, Twilio, SMTP, Firebase, SQLAlchemy, Pandas
Database	Firebase Realtime Database (for storage)
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Internet Connection	Stable connection for cloud-based services, notifications
Version Control	Git (for version control)

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

Our AI-Driven Fire Detection System follows a structured model integrating AI, real-time image processing, and cloud computing. The system consists of multiple interconnected components that work together to efficiently monitor and secure the environment against fire hazards.



CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List Of Major Activities

1. Requirement Analysis & Research
    - Identify hardware and software requirements for real-time fire detection.
    - Research and select appropriate AI models (YOLOv8 for fire detection) and notification methods (Twilio, SMTP).
2. Dataset Collection & Model Training
    - Collect and preprocess datasets for fire detection.
    - Train the YOLOv8 model using images or videos containing fire instances to optimize detection accuracy.
3. Fire Detection Algorithm Implementation
    - Integrate YOLOv8 with OpenCV for processing video feeds in real-time.
    - Fine-tune the fire detection algorithm for various lighting and environmental conditions.
4. Backend Development
    - Develop the backend using FastAPI to handle real-time data processing.
    - Set up API endpoints for communication between the backend and frontend.
    - Implement logic for generating fire alerts and sending notifications.
5. Frontend Development
    - Design and implement a user-friendly interface using React.js.
    - Create features to display fire alerts, camera feeds, and notification statuses.
6. Notification System Integration
    - Integrate Twilio API for SMS notifications and SMTP for email alerts.
    - Implement dynamic alerts for real-time fire detection with message content based on detection results.
7. Firebase Integration
    - Set up Firebase for storing historical data, logs, and alert information.
    - Implement real-time synchronization with Firebase for instant updates in the UI.
8. Testing & Debugging
    - Perform unit and integration testing to ensure fire detection, notification systems, and user interfaces work as expected.
    - Test the system with different video feeds and evaluate real-time performance.
9. Optimization & Performance Tuning
    - Optimize the real-time fire detection process for low-latency and high accuracy.
    - Enhance the system to ensure smooth operation in diverse real-world environments.

10. Deployment & Hosting

- Set up cloud deployment (e.g., AWS EC2) for hosting the backend and model.
- Ensure smooth integration with the frontend (React.js) and Firebase for cloud storage.

5.2 Estimated Time Duration In Days

Activity	Estimated Duration (Days)
Requirement Analysis & Research	8 Days
Dataset Collection & Preprocessing	10 Days
YOLOv8 Model Training & Fine-Tuning	12 Days
Fire Detection Algorithm Implementation	10 Days
Backend Development & API Setup	14 Days
Frontend Development & UI Design	12 Days
Notification System Integration	7 Days
Firebase Integration & Data Synchronization	8 Days
System Testing & Debugging	10 Days
Optimization & Performance Tuning	8 Days
Deployment & Cloud Hosting	7 Days
Final System Evaluation & Demonstration	5 Days

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the Real-Time Fire Detection and Notification System. It covers the system's architecture, workflow, integration of technologies, and the step-by-step breakdown of each module involved.

6.1 System Architecture

The system follows a client-server architecture designed to provide real-time fire detection and notification capabilities. It integrates modern technologies to ensure seamless fire detection, alerting, and data storage. The main components include:

- **Frontend (React.js)**: The user interface of the application allows users (e.g., security personnel, building managers) to monitor fire detection alerts, view notifications, and access historical fire-related data in a visually appealing dashboard.
- **Backend (FastAPI)**: The backend is responsible for managing API requests, processing sensor and image data, and interacting with the AI models to detect fire. It also handles user authentication, storage, and notification services.
- **Fire Detection Model (YOLOv8 + OpenCV)**: YOLOv8, a state-of-the-art object detection model, is integrated with OpenCV to detect fire in real-time from video feeds or images. The AI model processes incoming frames, and if fire is detected, it triggers an alert notification.
- **Notification System (SMTP / Twilio)**: The system utilizes SMTP for email notifications and Twilio for SMS alerts to notify users instantly when a fire is detected. Notifications are sent in real-time to ensure immediate response.
- **Data Storage (Firebase)**: Firebase is used for storing key data related to the fire detection events, including timestamps, notification logs, and sensor data. It ensures scalable and real-time storage for easy access and analysis.
- **Security (Clerk.js, SSL/TLS, OAuth 2.0)**: User authentication and access control are managed using **Clerk.js**, providing a seamless and secure login experience. Clerk.js enables easy integration of user sign-up, sign-in, and session management features. Additionally, **SSL/TLS encryption** ensures secure communication between the client and server, while **OAuth 2.0** is used for secure third-party authentication. User data is also encrypted and securely stored to prevent unauthorized access.

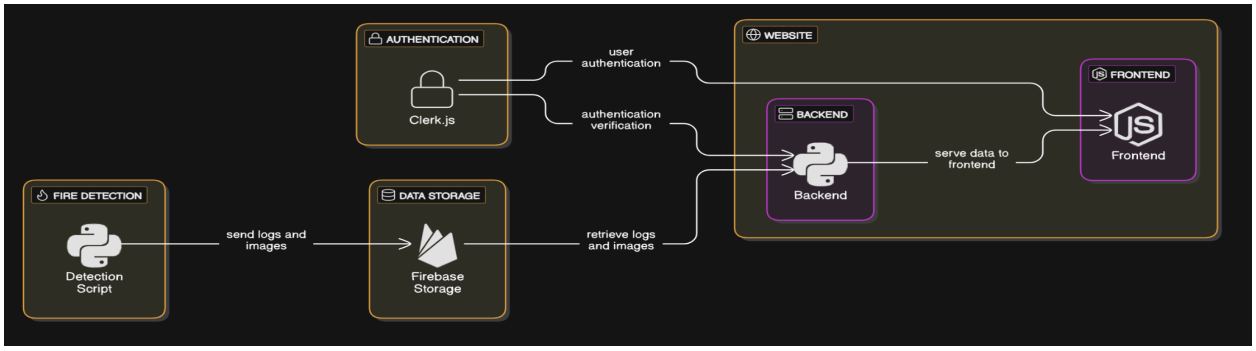


Figure 6.1 System Architecture Diagram

6.2 Workflow Implementation

This section explains the step-by-step execution of the **Real-Time Fire Detection and Notification System**.

6.2.1 Sensor Setup & Data Collection

A **laptop camera** is used to capture real-time video feeds of the environment. The camera continuously monitors the area for any visual signs of fire. These images are then transmitted to the backend for further processing and analysis.

6.2.2 Data Transmission & Cloud Storage

The captured images and video feeds are sent to the backend through API requests. All data is securely transmitted using encryption protocols and stored in **Firebase** cloud storage. Firebase ensures that the data is efficiently managed, scalable, and easily retrievable for analysis or historical purposes.

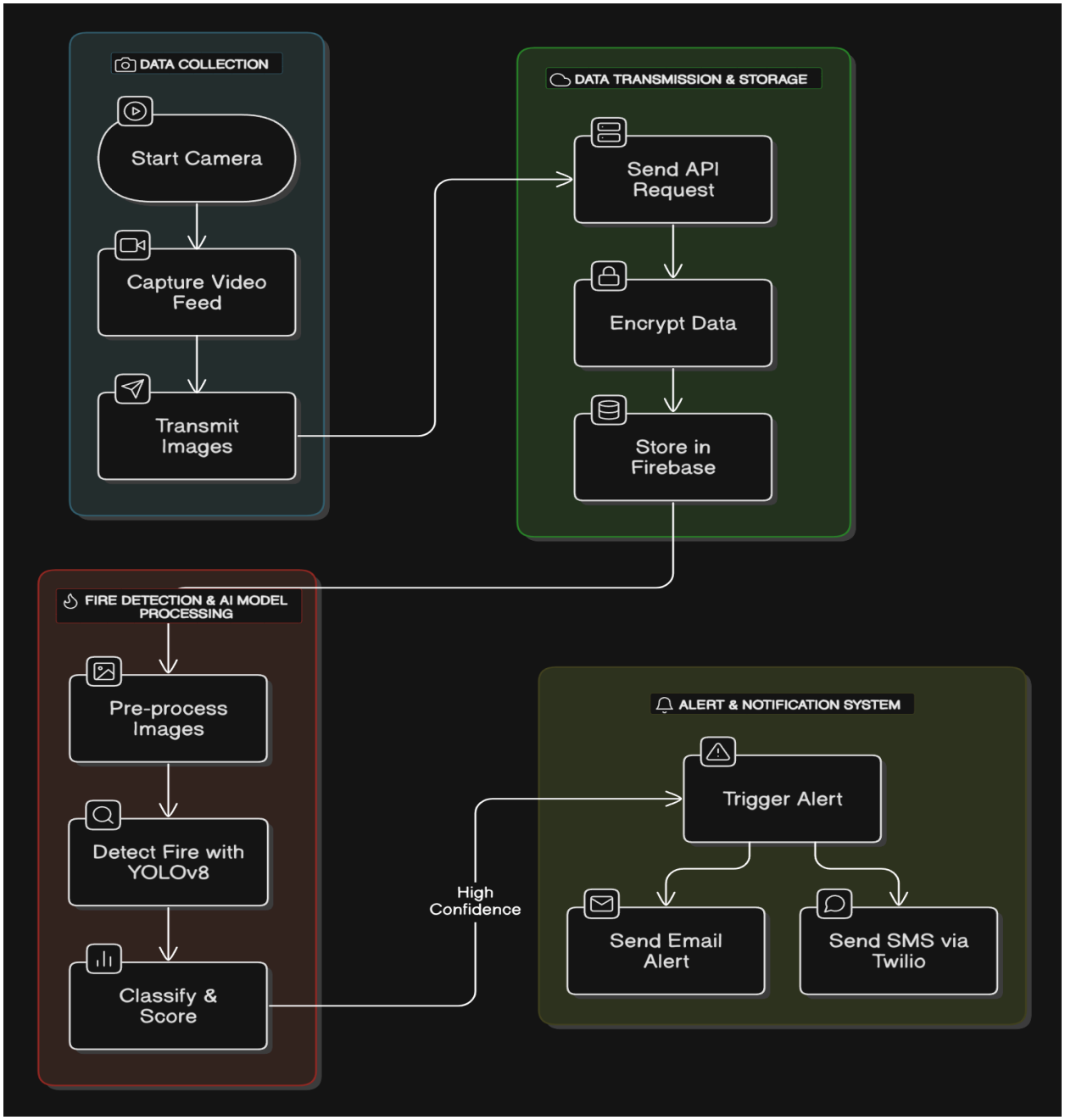
6.2.3 Fire Detection & AI Model Processing

The images captured by the camera are processed using **YOLOv8**, a state-of-the-art object detection model, to detect fire. **OpenCV** is used for real-time image pre-processing, enhancing the accuracy of the fire detection process. YOLOv8 classifies the images, analyzing patterns that indicate the presence of fire, and assigns a confidence score to each detection to ensure high accuracy before triggering any alerts.

6.2.4 Alert & Notification System

Once fire is detected with high confidence, the system triggers an immediate alert. Notifications are sent to the users through multiple channels, including the **web dashboard** for real-time monitoring, **email** for detailed alerts, and **SMS notifications** via **Twilio** to ensure timely communication. Additionally, **audible alerts** such as a **buzzer alarm** may be triggered to notify

users directly, ensuring prompt action is taken.



6.3 Object Detection Model Implementation

The AI models play a crucial role in detecting fire hazards by processing captured images. The system employs a **YOLOv8** model specifically trained for fire detection.

6.3.1 Fire & Smoke Detection Model

The system uses a **YOLOv8 model**, which was trained on a fire & smoke detection dataset to identify hazards in real-time images. The model processes frames captured by the laptop camera, detecting signs of fire, such as flames or smoke, and triggering alerts. Once fire is detected, notifications are sent to the user's dashboard.

```
import ultralytics

from roboflow import Roboflow

from ultralytics import YOLO

from IPython.display import Image

rf = Roboflow(api_key=MY_SECRET_KEY)

project = rf.workspace("detection-e83li").project("smokeandfire")

version = project.version(2)

dataset = version.download("yolov8")
```

```
!yolo task=detect mode=train model='/content/40plus15.pt'
data=/content/smokeandfire-2/data.yaml epochs=20 imgsz=640 plots=True
```

6.3.2 Model Training and Results

The training was conducted using a combined dataset, and performance metrics such as precision, recall, and mean Average Precision (mAP) were analyzed to assess model accuracy

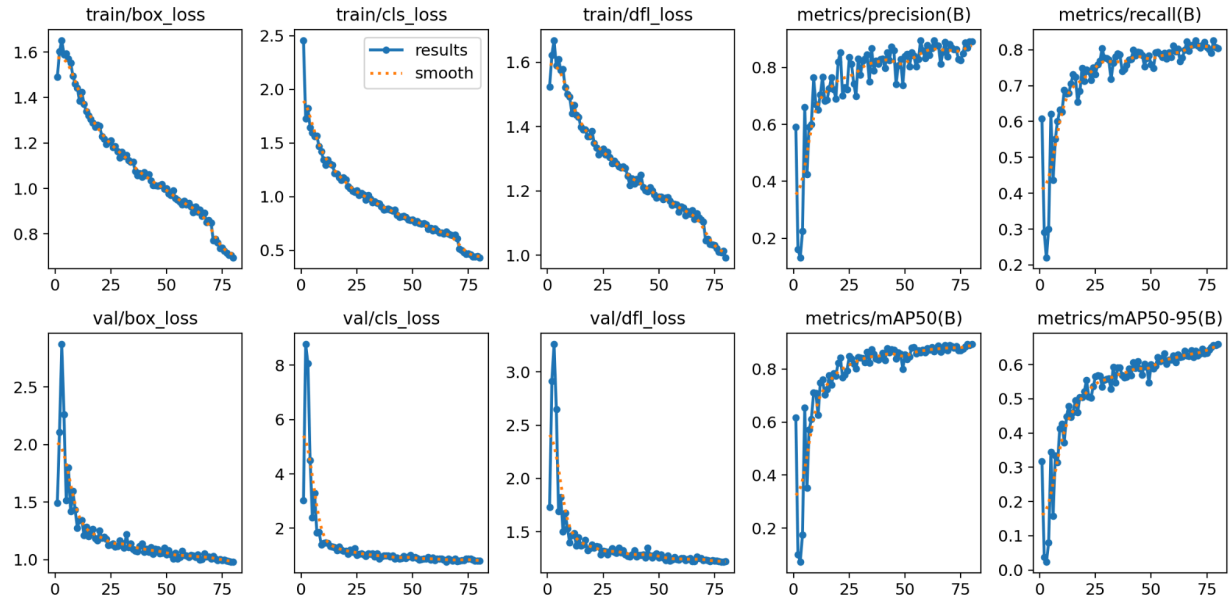


Figure 6.4.1 Results of the Fire Detection Model

6.3.3 Model Predictions and Outputs

This section presents the prediction results of the trained YOLO models. Sample images from real-time detections will be included, demonstrating the accuracy and efficiency of the fire detection model in identifying threats.

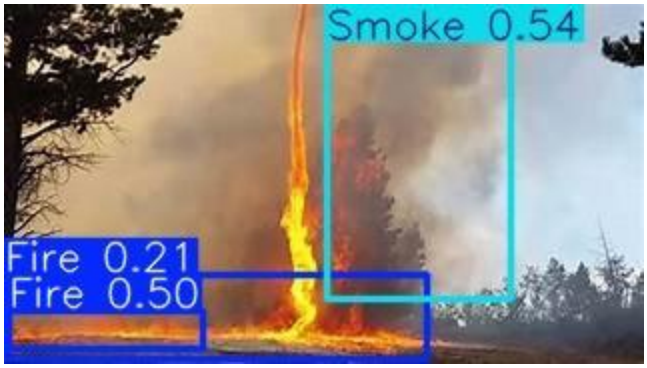


Figure 6.4.2 Fire Detection Model Prediction

The frontend will serve as the main interface for users to monitor real-time fire detection alerts and system status.

6.4.1 Dashboard Design

A user-friendly dashboard will be built using **React.js**, displaying real-time fire alerts, notifications, and camera feed images. The dashboard will provide users with a clear view of fire detection events, timestamps, and related data to help them respond quickly and effectively.

6.4.2 User Interaction & Alerts

The frontend will support immediate **alert notifications** when fire is detected, ensuring users can respond promptly. Alerts will be displayed on the dashboard, and users will receive **email** and **SMS notifications** for fast action. The frontend will allow users to view the status of their system and access the historical data of fire events and alerts.

6.5 Backend Implementation

The backend serves as the communication hub, processing data from the camera and the AI model, while handling the system's core functionality.

6.5.1 API Development

A **FastAPI-based** API will handle incoming data from the camera, process it using the **YOLOv8 fire detection model**, and return the results to the frontend. The backend will process images in

real time and send the fire detection results back, allowing the frontend to display alerts and update the status in real time.

6.5.2 Security and Data Management

For user authentication, **Clerk.js** will be used to manage secure sign-ups, sign-ins, and session management. Additionally, secure data transmission is ensured with **SSL/TLS encryption**. User data and fire event logs will be stored in **Firebase**, ensuring secure access and reliable storage with proper access control mechanisms. **OAuth 2.0** will be implemented for secure, third-party authentication when necessary. The website has been scanned for security using netsparker and the link the the report will be provided in references

6.6 Cloud Integration

Cloud services will be used for model execution, data storage, and scalability of the application.

6.6.1 Storage and Hosting

Firebase will be used for storing the fire detection event logs, sensor data, and any related metadata. Firebase provides real-time data management and scalable storage.

6.7 Deployment & Hosting (Live on Vercel)

The deployment phase has been successfully completed, with the system now live and fully deployed.

- **Model Deployment**: The trained **YOLOv8 fire detection model** is ran locally.
- **Frontend Deployment**: The **React.js-based frontend** is live and deployed on **Vercel**, providing a fast, scalable, and reliable platform for real-time alerts and monitoring. The deployment on Vercel ensures efficient performance and a smooth user experience.

The cloud-based infrastructure ensures efficient data flow, secure communication, and real-time fire detection and notifications for users.

6.8 Conclusion

This chapter provided an overview of the key implementation aspects of the **Real-Time Fire Detection and Notification System**, including the system architecture, workflow, AI model integration, backend and frontend development, and deployment strategies. By combining AI-based fire detection with cloud integration, the system aims to offer real-time monitoring and alerts to improve safety and response times. Once fully deployed, this solution will provide an efficient, reliable, and scalable system to ensure fire safety in real-time.

CHAPTER: 7 CONCLUSION AND FUTURE WORK

CHAPTER 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

The Real-time Fire Detection and Notification System project aims to develop an intelligent, scalable, and efficient solution for detecting fires in real-time and notifying the relevant parties immediately. Using advanced technologies like YOLOv8 for fire detection, OpenCV for video processing, and Twilio and SMTP for notifications, this system is designed to operate efficiently under real-world conditions, providing timely alerts and improving safety.

To date, the project has made substantial progress in several key areas including dataset collection, model training, backend development, and initial integration of core components. Although there are still a few areas requiring additional work, such as further optimization of the fire detection algorithm and frontend integration, the groundwork has been successfully laid.

In the coming weeks, the focus will be on finalizing the integration between the backend and frontend, performing extensive testing, and ensuring that the system functions reliably in different environments. Once these steps are completed, the system will be ready for final deployment and demonstration, providing real-time fire detection and alert notifications.

The project has the potential to be a vital tool for enhancing fire safety and can be expanded to various use cases, such as in smart homes, factories, and forests, where timely fire detection and rapid notification are crucial. Moving forward, enhancements and refinements will ensure the system's robustness and scalability, allowing it to serve as an effective and reliable fire detection solution in various scenarios.

7.2 Future Work

While the Real-time Fire Detection and Notification System is progressing well, there are several enhancements and additional features that could be integrated to further improve the system's functionality and reliability. The future work focuses on incorporating IoT capabilities, expanding detection mechanisms, and enhancing the overall system's scalability and robustness.

Here are some potential future enhancements:

1. Integration of IoT Sensors for Smoke and Temperature Detection
    - Currently, the system relies solely on visual fire detection using YOLOv8. To improve accuracy and provide an additional layer of safety, integrating IoT

sensors such as smoke detectors and temperature sensors could help detect early signs of a fire before it becomes visible.
    - These sensors can send real-time data to the system, allowing for quicker alerts and reducing false positives.
2. Expanded Detection Capabilities (Gas, Smoke, and Flame Detection)
    - In addition to fire detection through visual recognition, integrating sensors for detecting gas leaks, smoke, and flames would enhance the system's ability to identify fire hazards early.
    - Combining thermal imaging or infrared cameras with AI-based detection would help in spotting fire hazards in dark or obscured environments.
3. Improved Notification System
    - Future work will involve improving the notification system by adding multi-channel alerts, including integration with mobile apps, push notifications, or automated phone calls for critical situations.
    - Adding AI-powered predictive alerts based on historical data could help predict potential fire outbreaks before they occur, improving preparedness.
4. Data Analytics for Fire Prevention Insights
    - Analyzing the historical data collected by the system can provide valuable insights into fire hazards, allowing users to identify patterns, high-risk areas, and potential fire triggers. This data could be used to prevent future incidents by providing actionable insights.
    - Implementing predictive analytics and integrating with other smart home or industrial systems could further enhance the overall fire safety management strategy.

CHAPTER: 8 REFERENCES

CHAPTER 8 REFERENCES

YOLOv8: You Only Look Once (v8). - https://github.com/ultralytics/yolov8
OpenCV for Computer Vision - https://opencv.org/
Twilio: Programmable Messaging API - https://www.twilio.com/docs/sms/send-messages
SMTP (Simple Mail Transfer Protocol) - https://docs.python.org/3/library/smtplib.html
ReactJS Documentation - https://reactjs.org/
FastAPI Documentation - https://fastapi.tiangolo.com/
Firebase Documentation - https://firebase.google.com/docs
Twilio SMS Service - Python SDK - https://www.twilio.com/docs/usage/api
OpenCV for Fire Detection - https://medium.com/
Firebase Cloud Storage - https://firebase.google.com/docs/storage
Sensors for Smoke and Fire Detection - https://link.springer.com/
Website Scan Report - fire-watch-kohl.vercel.app - Detailed Scan Report.pdf
www.coursehero.com

Munish Patwa

IBM PROJECT_W_O_CERTIFICATES.pdf

Assignment 10

Research_1

Ganpat University

Document Details

Submission ID

trn:oid:::1:3235890459

Submission Date

May 1, 2025, 9:00 PM GMT+5:30

Download Date

May 1, 2025, 9:04 PM GMT+5:30

File Name

IBM_PROJECT_W_O_CERTIFICATES.pdf

File Size

1.2 MB

31 Pages

3,780 Words

22,182 Characters

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

18 Not Cited or Quoted 5%
Matches with neither in-text citation nor quotation marks

3 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
1% Publications
0% Submitted works (Student Papers)

Match Groups

- **18** Not Cited or Quoted 5%
  Matches with neither in-text citation nor quotation marks
- **3** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
1% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
www.irjet.net		<1%
2	Internet	
eprints.lancs.ac.uk		<1%
3	Internet	
www.framboise314.fr		<1%
4	Publication	
Shashi Kant Dargar, Shilpi Birla, Abha Dargar, Avtar Singh, D. Ganeshaperumal. "...		<1%
5	Internet	
dspace.bracu.ac.bd:8080		<1%
6	Internet	
www.mdpi.com		<1%
7	Publication	
Howard Anderson, Sharon Yull, Bruce Hellingsworth. "Higher National Computin...		<1%
8	Internet	
gg.deals		<1%
9	Internet	
irojournals.com		<1%
10	Internet	
keylabs.ai		<1%

11 Internet

ntnuopen.ntnu.no <1%

12 Internet

libstore.ugent.be <1%

13 Internet

www.coursehero.com <1%

14 Internet

www.gymkosb.sk <1%

15 Internet

www.ijisae.org <1%

16 Internet

www.nerc.com <1%

ABSTRACT

The Real-time Fire Detection and Notification System is designed to enhance safety measures by providing immediate alerts upon detecting fire incidents in various environments. The primary goal of this project is to develop an intelligent system capable of detecting fire in real time using computer vision techniques and notifying relevant authorities or individuals through SMS and email alerts.

The system leverages YOLOv8 (You Only Look Once), a state-of-the-art deep learning model for real-time object detection, integrated with OpenCV for image processing. The fire detection model continuously monitors the video feed to identify potential fire events, providing a proactive approach to preventing damage. Upon detecting fire, the system triggers notifications through Twilio for SMS and SMTP protocol for email, ensuring timely alerts.

The frontend of the application is developed using React.js, offering a user-friendly interface for visualization, control, and status monitoring. On the backend, FastAPI is employed for handling API requests efficiently and enabling seamless communication between the frontend, backend, and the notification services. For persistent data storage, Firebase is utilized, ensuring secure, real-time data management and access.

This system aims to improve emergency response times, mitigate damage caused by fire incidents, and promote greater safety awareness in both industrial and residential settings. The project is nearing completion, with all major components developed and integrated successfully. Future work includes optimization of detection accuracy and system performance.

INDEX

CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

Fire incidents are a significant threat to life, property, and the environment, making it crucial to have effective fire detection and prevention systems in place. Traditional fire detection systems, often relying on manual monitoring or basic alarm systems, can be slow to react and may fail to detect fires in their early stages. This delay in response can lead to catastrophic damage and loss. With the rapid advancements in computer vision, machine learning, and IoT technologies, there is a tremendous opportunity to improve fire detection and response systems, making them more efficient, real-time, and automated.

This project focuses on developing a Real-time Fire Detection and Notification System designed to address these challenges. By using YOLOv8, a powerful object detection model, combined with OpenCV for real-time image processing, the system automatically detects the presence of fire in video feeds. When a fire is detected, the system triggers notifications via Twilio (for SMS) and SMTP (for email) to alert relevant personnel or emergency responders, enabling faster action and potentially preventing further damage.

The system architecture is divided into several key components:

- Frontend: Built using React.js, providing an interactive and intuitive user interface for system monitoring and status updates.
- Backend: Developed with FastAPI, ensuring efficient and scalable API handling for real-time operations.
- Storage: Firebase is used for storing data, including logs and alerts, ensuring secure and real-time access across multiple devices.

The primary objective of this project is to provide a fully functional, automated fire detection and alert system that works in real-time to minimize the potential harm caused by fires. The system is designed to be scalable, efficient, and easy to deploy across various environments, from industrial facilities to residential properties. By integrating cutting-edge technologies, the project aims to significantly enhance safety protocols and emergency response times.

16

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to develop a real-time fire detection and notification system that automatically detects fire incidents and alerts the relevant authorities or individuals. This system aims to enhance safety measures in various settings, including residential, commercial, and industrial environments, by using advanced computer vision and machine learning techniques.

Key Features and Functionalities:

1. Real-time Fire Detection:
    - Utilize YOLOv8, a state-of-the-art deep learning model, for detecting fire in video streams.
    - Integrate OpenCV for processing and analyzing video frames in real time to identify fire-related features (e.g., flames, smoke).
2. Notification System:
    - SMS Alerts: Integrate Twilio API to send immediate SMS notifications to specified contacts (e.g., emergency services, building management, or owners) when fire is detected.
    - Email Alerts: Utilize the SMTP protocol to send email notifications, including detailed information about the detected fire and its location.
3. User Interface:
    - Develop a React.js-based frontend for system monitoring, where users can track the status of fire detection, manage notifications, and visualize the results.
    - Provide a dashboard for live monitoring of the video feed, fire detection alerts, and overall system status.
4. Backend & API Communication:
    - Use FastAPI to handle communication between the frontend and backend, ensuring fast and efficient real-time data processing.
    - The backend processes the fire detection output and triggers notifications accordingly.
5. Data Storage & Management:
    - Integrate Firebase for real-time data storage, such as storing detection logs, alert history, and system performance data.
    - Ensure secure access to the system data and provide analytics capabilities (e.g., historical data for fire detection).
6. Scalability & Deployment:
    - Design the system to be scalable and adaptable for deployment in various environments, including large-scale industrial complexes, office buildings, and residential properties.
    - Ensure the system can be easily deployed with minimal setup and configuration.

Limitations and Exclusions:

- This project will not include physical hardware components such as cameras or sensors. The fire detection will solely rely on video feeds from existing security cameras.
- The system will focus on fire detection and notifications only; additional emergency response features (e.g., automated sprinkler systems) are out of scope.
- The project will be limited to fire detection using visual data and will not extend to other types of environmental hazards like gas leaks or earthquakes.

Future Enhancements:

- Integration of additional sensors such as smoke detectors and temperature sensors to improve detection accuracy.
- Expansion to include automated response mechanisms, such as triggering alarms or activating emergency protocols in buildings.

The Real-time Fire Detection and Notification System will not only address immediate fire detection needs but will also contribute to the broader goal of enhancing safety through automation, enabling quicker and more reliable emergency responses.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i5 / AMD Ryzen 3 (or equivalent)
RAM	8GB or more
HDD/SSD	50GB of free space (SSD preferred for faster access)
GPU	NVIDIA GTX 1060 or better (for YOLOv8 model inference)
Network	Stable internet connection for real-time monitoring & notifications
Web Camera/Video Feed	Standard security camera or webcam (for video input)

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

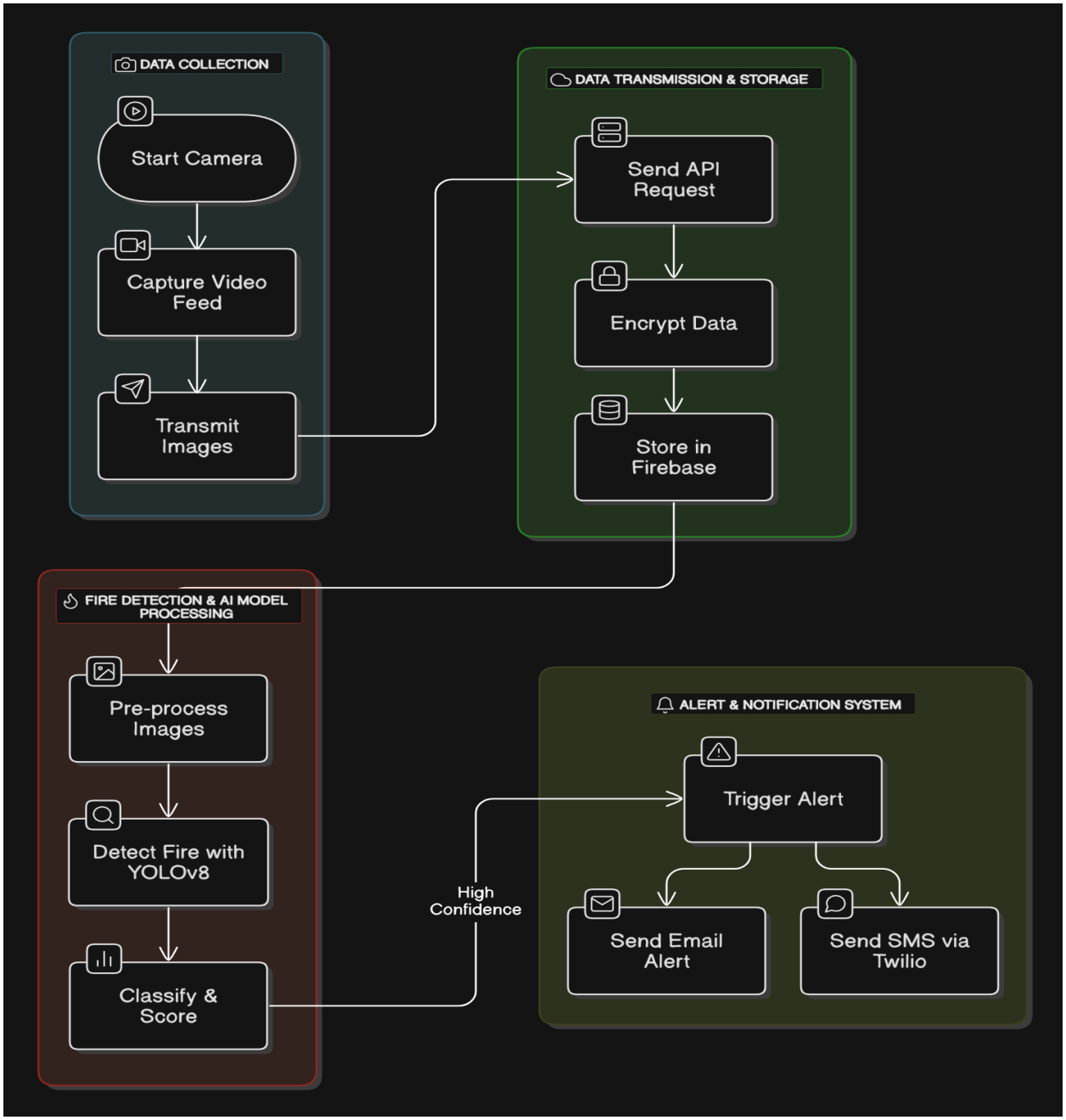
Component	Specification
Operating System	Windows, macOS, or Linux (modern versions supported)
Programming Language	Python 3.8+ (for backend and AI-based fire detection)
Frontend Framework	React.js (for user interface)
Backend Framework	FastAPI (for backend and API communication)
Libraries/Tools	OpenCV, YOLOv8, Twilio, SMTP, Firebase, SQLAlchemy, Pandas
Database	Firebase Realtime Database (for storage)
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Internet Connection	Stable connection for cloud-based services, notifications
Version Control	Git (for version control)

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

Our AI-Driven Fire Detection System follows a structured model integrating AI, real-time image processing, and cloud computing. The system consists of multiple interconnected components that work together to efficiently monitor and secure the environment against fire hazards.


DATA COLLECTION
Start Camera
Capture Video Feed
Transmit Images
DATA TRANSMISSION & STORAGE
Send API Request
Encrypt Data
Store in Firebase
FIRE DETECTION & AI MODEL PROCESSING
Pre-process Images
Detect Fire with YOLOv8
Classify & Score
High Confidence
ALERT & NOTIFICATION SYSTEM
Trigger Alert
Send Email Alert
Send SMS via Twilio

CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List Of Major Activities

1. Requirement Analysis & Research
    - Identify hardware and software requirements for real-time fire detection.

   - Research and select appropriate AI models (YOLOv8 for fire detection) and notification methods (Twilio, SMTP).
2. Dataset Collection & Model Training
   - Collect and preprocess datasets for fire detection.
   - Train the YOLOv8 model using images or videos containing fire instances to optimize detection accuracy.
3. Fire Detection Algorithm Implementation
   - Integrate YOLOv8 with OpenCV for processing video feeds in real-time.
   - Fine-tune the fire detection algorithm for various lighting and environmental conditions.
4. Backend Development
   - Develop the backend using FastAPI to handle real-time data processing.
   - Set up API endpoints for communication between the backend and frontend.
   - Implement logic for generating fire alerts and sending notifications.
5. Frontend Development
   - Design and implement a user-friendly interface using React.js.
   - Create features to display fire alerts, camera feeds, and notification statuses.
6. Notification System Integration
   - Integrate Twilio API for SMS notifications and SMTP for email alerts.
   - Implement dynamic alerts for real-time fire detection with message content based on detection results.
7. Firebase Integration
   - Set up Firebase for storing historical data, logs, and alert information.
   - Implement real-time synchronization with Firebase for instant updates in the UI.
8. Testing & Debugging
   - Perform unit and integration testing to ensure fire detection, notification systems, and user interfaces work as expected.
   - Test the system with different video feeds and evaluate real-time performance.
9. Optimization & Performance Tuning
   - Optimize the real-time fire detection process for low-latency and high accuracy.
   - Enhance the system to ensure smooth operation in diverse real-world environments.

10. Deployment & Hosting

- Set up cloud deployment (e.g., AWS EC2) for hosting the backend and model.
- Ensure smooth integration with the frontend (React.js) and Firebase for cloud storage.

5.2 Estimated Time Duration In Days

Activity	Estimated Duration (Days)
Requirement Analysis & Research	8 Days
Dataset Collection & Preprocessing	10 Days
YOLOv8 Model Training & Fine-Tuning	12 Days
Fire Detection Algorithm Implementation	10 Days
Backend Development & API Setup	14 Days
Frontend Development & UI Design	12 Days
Notification System Integration	7 Days
Firebase Integration & Data Synchronization	8 Days
System Testing & Debugging	10 Days
Optimization & Performance Tuning	8 Days
Deployment & Cloud Hosting	7 Days
Final System Evaluation & Demonstration	5 Days

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the Real-Time Fire Detection and Notification System. It covers the system's architecture, workflow, integration of technologies, and the step-by-step breakdown of each module involved.

6.1 System Architecture

The system follows a client-server architecture designed to provide real-time fire detection and notification capabilities. It integrates modern technologies to ensure seamless fire detection, alerting, and data storage. The main components include:

- **Frontend (React.js)**: The user interface of the application allows users (e.g., security personnel, building managers) to monitor fire detection alerts, view notifications, and access historical fire-related data in a visually appealing dashboard.
- **Backend (FastAPI)**: The backend is responsible for managing API requests, processing sensor and image data, and interacting with the AI models to detect fire. It also handles user authentication, storage, and notification services.
- **Fire Detection Model (YOLOv8 + OpenCV)**: YOLOv8, a state-of-the-art object detection model, is integrated with OpenCV to detect fire in real-time from video feeds or images. The AI model processes incoming frames, and if fire is detected, it triggers an alert notification.
- **Notification System (SMTP / Twilio)**: The system utilizes SMTP for email notifications and Twilio for SMS alerts to notify users instantly when a fire is detected. Notifications are sent in real-time to ensure immediate response.
- **Data Storage (Firebase)**: Firebase is used for storing key data related to the fire detection events, including timestamps, notification logs, and sensor data. It ensures scalable and real-time storage for easy access and analysis.
- **Security (Clerk.js, SSL/TLS, OAuth 2.0)**: User authentication and access control are managed using **Clerk.js**, providing a seamless and secure login experience. Clerk.js enables easy integration of user sign-up, sign-in, and session management features. Additionally, **SSL/TLS encryption** ensures secure communication between the client and server, while **OAuth 2.0** is used for secure third-party authentication. User data is also encrypted and securely stored to prevent unauthorized access.

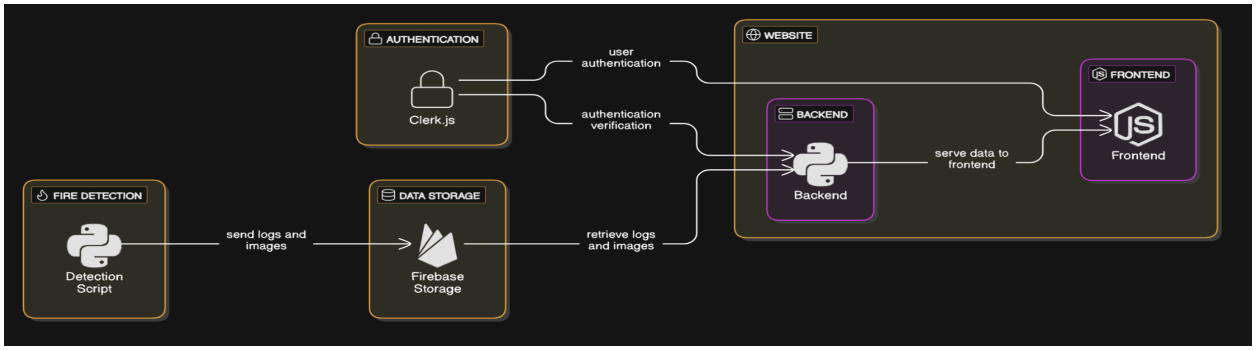


Figure 6.1 System Architecture Diagram

6.2 Workflow Implementation

This section explains the step-by-step execution of the **Real-Time Fire Detection and Notification System**.

6.2.1 Sensor Setup & Data Collection

A **laptop camera** is used to capture real-time video feeds of the environment. The camera continuously monitors the area for any visual signs of fire. These images are then transmitted to the backend for further processing and analysis.

6.2.2 Data Transmission & Cloud Storage

The captured images and video feeds are sent to the backend through API requests. All data is securely transmitted using encryption protocols and stored in **Firebase** cloud storage. Firebase ensures that the data is efficiently managed, scalable, and easily retrievable for analysis or historical purposes.

6.2.3 Fire Detection & AI Model Processing

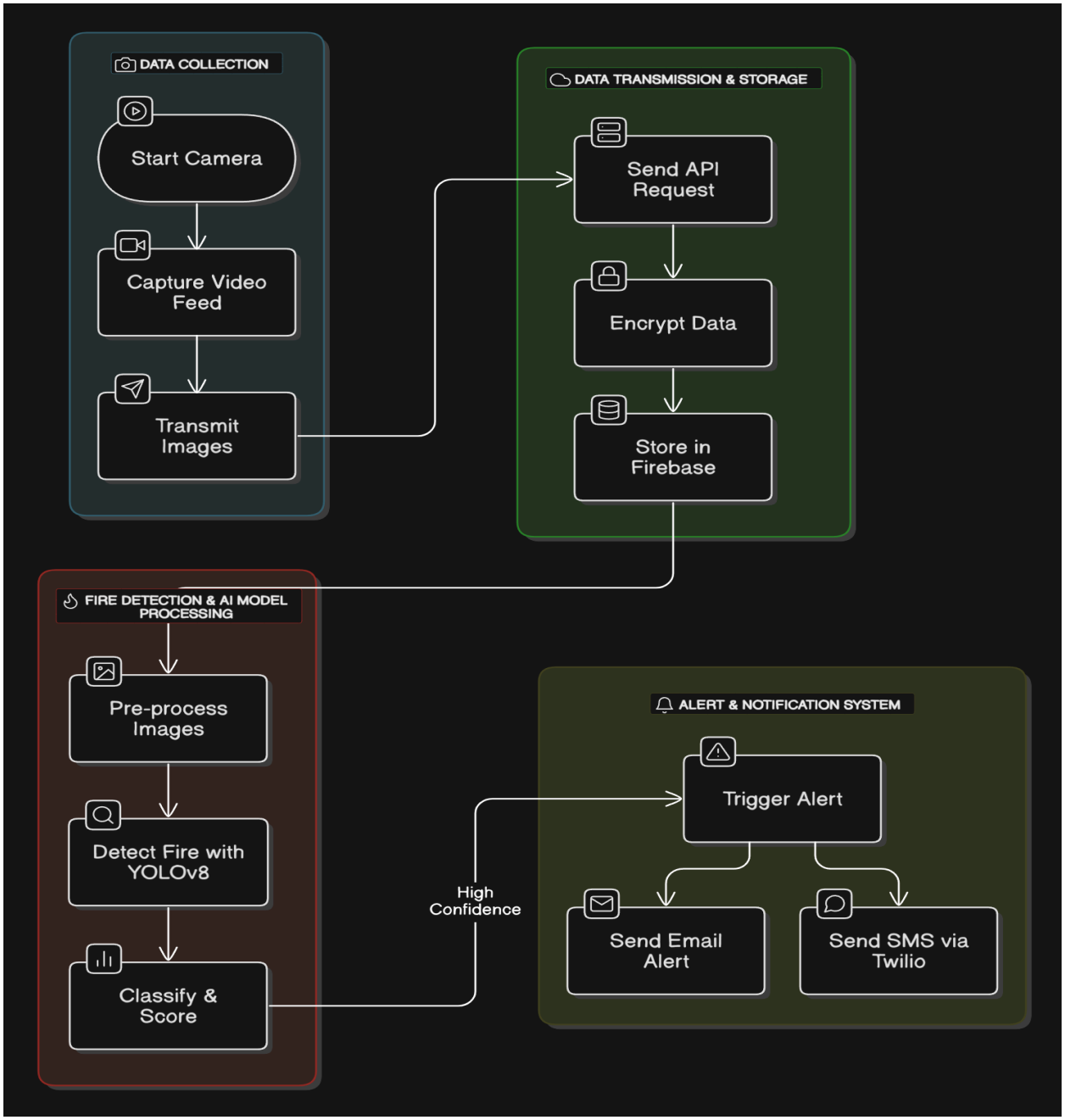
9

The images captured by the camera are processed using **YOLOv8**, a state-of-the-art object detection model, to detect fire. **OpenCV** is used for real-time image pre-processing, enhancing the accuracy of the fire detection process. YOLOv8 classifies the images, analyzing patterns that indicate the presence of fire, and assigns a confidence score to each detection to ensure high accuracy before triggering any alerts.

6.2.4 Alert & Notification System

Once fire is detected with high confidence, the system triggers an immediate alert. Notifications are sent to the users through multiple channels, including the **web dashboard** for real-time monitoring, **email** for detailed alerts, and **SMS notifications** via **Twilio** to ensure timely communication. Additionally, **audible alerts** such as a **buzzer alarm** may be triggered to notify

users directly, ensuring prompt action is taken.



6.3 Object Detection Model Implementation

The AI models play a crucial role in detecting fire hazards by processing captured images. The system employs a **YOLOv8** model specifically trained for fire detection.

6.3.1 Fire & Smoke Detection Model

The system uses a **YOLOv8 model**, which was trained on a fire & smoke detection dataset to identify hazards in real-time images. The model processes frames captured by the laptop camera, detecting signs of fire, such as flames or smoke, and triggering alerts. Once fire is detected, notifications are sent to the user's dashboard.

```
import ultralytics

from roboflow import Roboflow

from ultralytics import YOLO

from IPython.display import Image

rf = Roboflow(api_key=MY_SECRET_KEY)

project = rf.workspace("detection-e83li").project("smokeandfire")

version = project.version(2)

dataset = version.download("yolov8")
```

```
!yolo task=detect mode=train model='/content/40plus15.pt'
data=/content/smokeandfire-2/data.yaml epochs=20 imgsz=640 plots=True
```

6.3.2 Model Training and Results

The training was conducted using a combined dataset, and performance metrics such as precision, recall, and mean Average Precision (mAP) were analyzed to assess model accuracy

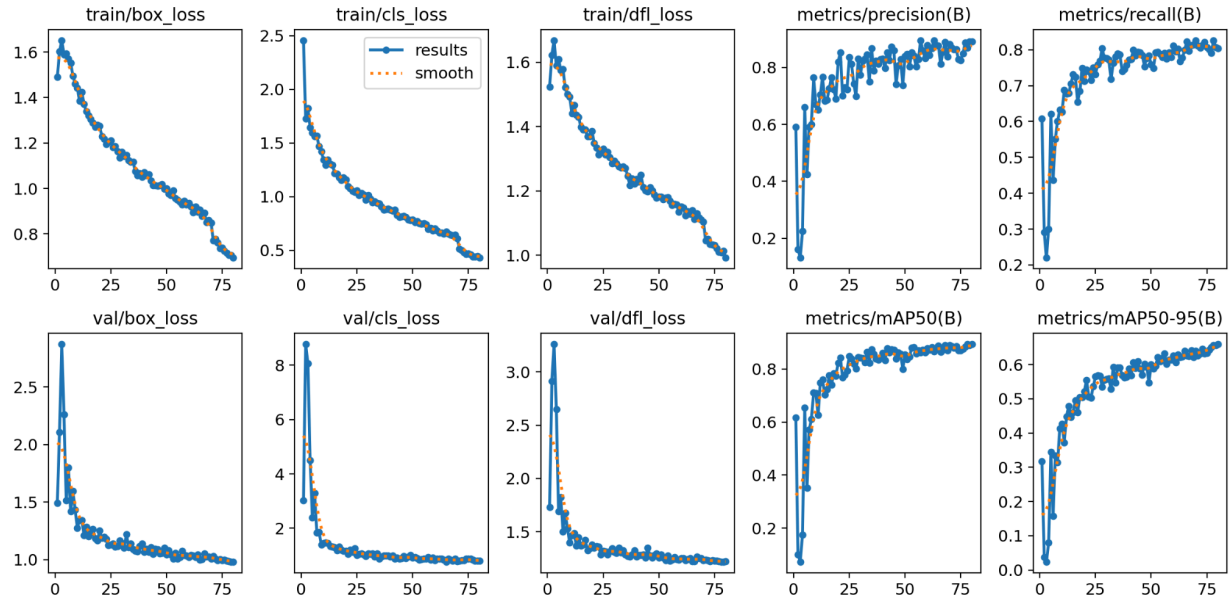


Figure 6.4.1 Results of the Fire Detection Model

6.3.3 Model Predictions and Outputs

This section presents the prediction results of the trained YOLO models. Sample images from real-time detections will be included, demonstrating the accuracy and efficiency of the fire detection model in identifying threats.

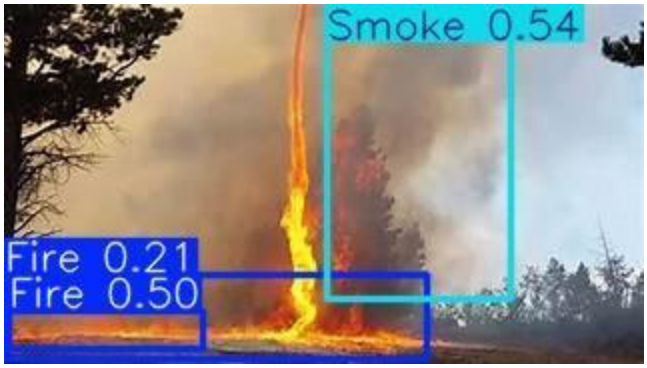


Figure 6.4.2 Fire Detection Model Prediction

The frontend will serve as the main interface for users to monitor real-time fire detection alerts and system status.

6.4.1 Dashboard Design

A user-friendly dashboard will be built using **React.js**, displaying real-time fire alerts, notifications, and camera feed images. The dashboard will provide users with a clear view of fire detection events, timestamps, and related data to help them respond quickly and effectively.

6.4.2 User Interaction & Alerts

The frontend will support immediate **alert notifications** when fire is detected, ensuring users can respond promptly. Alerts will be displayed on the dashboard, and users will receive **email** and **SMS notifications** for fast action. The frontend will allow users to view the status of their system and access the historical data of fire events and alerts.

6.5 Backend Implementation

The backend serves as the communication hub, processing data from the camera and the AI model, while handling the system's core functionality.

6.5.1 API Development

A **FastAPI-based** API will handle incoming data from the camera, process it using the **YOLOv8 fire detection model**, and return the results to the frontend. The backend will process images in

real time and send the fire detection results back, allowing the frontend to display alerts and update the status in real time.

6.5.2 Security and Data Management

For user authentication, **Clerk.js** will be used to manage secure sign-ups, sign-ins, and session management. Additionally, secure data transmission is ensured with **SSL/TLS encryption**. User data and fire event logs will be stored in **Firebase**, ensuring secure access and reliable storage with proper access control mechanisms. **OAuth 2.0** will be implemented for secure, third-party authentication when necessary. The website has been scanned for security using netsparker and the link the the report will be provided in references

6.6 Cloud Integration

Cloud services will be used for model execution, data storage, and scalability of the application.

6.6.1 Storage and Hosting

Firebase will be used for storing the fire detection event logs, sensor data, and any related metadata. Firebase provides real-time data management and scalable storage.

6.7 Deployment & Hosting (Live on Vercel)

The deployment phase has been successfully completed, with the system now live and fully deployed.

- **Model Deployment**: The trained **YOLOv8 fire detection model** is ran locally.
- **Frontend Deployment**: The **React.js-based frontend** is live and deployed on **Vercel**, providing a fast, scalable, and reliable platform for real-time alerts and monitoring. The deployment on Vercel ensures efficient performance and a smooth user experience.

The cloud-based infrastructure ensures efficient data flow, secure communication, and real-time fire detection and notifications for users.

6.8 Conclusion

This chapter provided an overview of the key implementation aspects of the **Real-Time Fire Detection and Notification System**, including the system architecture, workflow, AI model integration, backend and frontend development, and deployment strategies. By combining AI-based fire detection with cloud integration, the system aims to offer real-time monitoring and alerts to improve safety and response times. Once fully deployed, this solution will provide an efficient, reliable, and scalable system to ensure fire safety in real-time.

turnitin Page 30 of 35 - Integrity Submission Submission ID trn:oid:::1:3235890459

CHAPTER: 7 CONCLUSION AND FUTURE WORK

CHAPTER 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

The Real-time Fire Detection and Notification System project aims to develop an intelligent, scalable, and efficient solution for detecting fires in real-time and notifying the relevant parties immediately. Using advanced technologies like YOLOv8 for fire detection, OpenCV for video processing, and Twilio and SMTP for notifications, this system is designed to operate efficiently under real-world conditions, providing timely alerts and improving safety.

To date, the project has made substantial progress in several key areas including dataset collection, model training, backend development, and initial integration of core components. Although there are still a few areas requiring additional work, such as further optimization of the fire detection algorithm and frontend integration, the groundwork has been successfully laid.

In the coming weeks, the focus will be on finalizing the integration between the backend and frontend, performing extensive testing, and ensuring that the system functions reliably in different environments. Once these steps are completed, the system will be ready for final deployment and demonstration, providing real-time fire detection and alert notifications.

The project has the potential to be a vital tool for enhancing fire safety and can be expanded to various use cases, such as in smart homes, factories, and forests, where timely fire detection and rapid notification are crucial. Moving forward, enhancements and refinements will ensure the system's robustness and scalability, allowing it to serve as an effective and reliable fire detection solution in various scenarios.

7.2 Future Work

While the Real-time Fire Detection and Notification System is progressing well, there are several enhancements and additional features that could be integrated to further improve the system's functionality and reliability. The future work focuses on incorporating IoT capabilities, expanding detection mechanisms, and enhancing the overall system's scalability and robustness.

Here are some potential future enhancements:

1. Integration of IoT Sensors for Smoke and Temperature Detection
    - Currently, the system relies solely on visual fire detection using YOLOv8. To improve accuracy and provide an additional layer of safety, integrating IoT

sensors such as smoke detectors and temperature sensors could help detect early signs of a fire before it becomes visible.
    - These sensors can send real-time data to the system, allowing for quicker alerts and reducing false positives.
2. Expanded Detection Capabilities (Gas, Smoke, and Flame Detection)
    - In addition to fire detection through visual recognition, integrating sensors for detecting gas leaks, smoke, and flames would enhance the system's ability to identify fire hazards early.
    - Combining thermal imaging or infrared cameras with AI-based detection would help in spotting fire hazards in dark or obscured environments.
3. Improved Notification System
    - Future work will involve improving the notification system by adding multi-channel alerts, including integration with mobile apps, push notifications, or automated phone calls for critical situations.
    - Adding AI-powered predictive alerts based on historical data could help predict potential fire outbreaks before they occur, improving preparedness.
4. Data Analytics for Fire Prevention Insights
    - Analyzing the historical data collected by the system can provide valuable insights into fire hazards, allowing users to identify patterns, high-risk areas, and potential fire triggers. This data could be used to prevent future incidents by providing actionable insights.
    - Implementing predictive analytics and integrating with other smart home or industrial systems could further enhance the overall fire safety management strategy.

CHAPTER: 8 REFERENCES

CHAPTER 8 REFERENCES

YOLOv8: You Only Look Once (v8). - https://github.com/ultralytics/yolov8
OpenCV for Computer Vision - https://opencv.org/
Twilio: Programmable Messaging API - https://www.twilio.com/docs/sms/send-messages
SMTP (Simple Mail Transfer Protocol) - https://docs.python.org/3/library/smtplib.html
ReactJS Documentation - https://reactjs.org/
FastAPI Documentation - https://fastapi.tiangolo.com/
Firebase Documentation - https://firebase.google.com/docs
Twilio SMS Service - Python SDK - https://www.twilio.com/docs/usage/api
OpenCV for Fire Detection - https://medium.com/
Firebase Cloud Storage - https://firebase.google.com/docs/storage
Sensors for Smoke and Fire Detection - https://link.springer.com/
Website Scan Report - fire-watch-kohl.vercel.app - Detailed Scan Report.pdf
www.coursehero.com

PaperRater

IBM Project Report On Real-Time Monitoring and Evaluation System for Fire Department Applications Developed By: - Guided By:- Munish Patwa (21162121017) Prof. Sulabh Bhatt(Internal) Aniket Panjwani(22162102002) Mr. Nirav Raj(External) Astik Saxena (21162171002) Submitted to Faculty of Engineering and Technology Institute of Computer Technology Ganpat University Year - 2025 1 CERTIFICATE This is to certify that the IBM Project work entitled ?Real-Time Monitoring and Evaluation System for Fire Department Applications? by Munish Patwa (Enrolment No. 21162121017) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology ? Computer Science and... (only first 800 chars shown)

Analysis complete. Our feedback is listed below in printable form. Some of the items have been truncated or removed to provide better print compatibility.

Plagiarism Detection

Original Work

Originality: 95%

No sign of plagiarism was found. That's what we like to see!

The following web pages may contain content matching this document:
https://www.studymode.com/essays/a-Report-On-Se...
https://www.studymode.com/essays/Investors-Pref...
https://arxiv.org/abs/1805.00471
https://arxiv.org/abs/1909.13391
https://www.mdpi.com/2073-8994/12/9/1566
https://www.studymode.com/essays/Chronicles-Of-...
https://www.studymode.com/essays/Standard-Mba-P...
http://sisaljournal.org/archives/mar13/editorial/ (http://sisaljournal.org/archives/mar13/editorial/)
https://www.bartleby.com/essay/Wireless-Securit...
https://www.bartleby.com/essay/What-Is-The-Futu...
https://en.wikipedia.org/wiki/Gross%20farm%20in...
https://en.wikipedia.org/wiki/Raw%20data
https://www.frontiersin.org/article/10.3389/fme...
https://arxiv.org/abs/1905.11922
https://arxiv.org/abs/2102.04460
https://www.termpaperwarehouse.com/essay-on/Pro...
https://www.termpaperwarehouse.com/essay-on/Dsf...
https://iq.opengenus.org/future-of-artificial-i...
View Matching Text (http://www.PaperRater.com/proofreader/plag_results/4509eb80442ea28e0fdba6c23?from_sub=true)
Click the button above to see which portions of your text match which sources. This is a premium-only feature.

More info on our originality scoring process (http://www.PaperRater.com/page/plagiarism-detection) .

Word Choice

Usage of Bad Phrases

Bad Phrase Score: 0.44 (lower is better)
The Bad Phrase Score is based on the quality and quantity of trite or inappropriate words, phrases, egregious misspellings, and cliches found in your paper. You did equal or better than **100%** of the people in your grade.

Congratulations - your writing exhibits some high quality phrases. Keep it up!

Style

Usage of Transitional Phrases

Transitional Words Score: 53
This score is based on quality of transitional phrases used within your paper. You did equal or better than **30%** of the people in your grade.

Your usage of transitional phrases is below average. Please review the writing tips below.

One sign of an excellent writer is the use of transitional phrases (e.g. therefore, consequently, furthermore). Transitional words and phrases contribute to the cohesiveness (http://www.PaperRater.com/vocab_builder/show/cohesiveness) of a text and allow the sentences to flow smoothly. Without transitional phrases, a text will often seem disorganized and will most likely be difficult to understand. When these special words are used, they provide organization within a text and lead to greater understanding and enjoyment on the part of the reader.

The following transitional phrases were found in your document:
specifically, and, in addition, although, accordingly

Consider using additional transitions where appropriate:
- consequently (http://paperrater.com/vocab_builder/show/consequently)
- moreover (http://paperrater.com/vocab_builder/show/moreover)
- nevertheless (http://paperrater.com/vocab_builder/show/nevertheless)
- notwithstanding (http://paperrater.com/vocab_builder/show/notwithstanding)
- conversely (http://paperrater.com/vocab_builder/show/conversely)
- ordinarily (http://paperrater.com/vocab_builder/show/ordinarily)

Transitional phrases may be used in various places in a text:
- between paragraphs
- between sentences
- between sentence parts
- within sentence parts

Consider this example:

She is one of the most kind persons I have ever known. **Moreover**, she is generous, patient and possesses a magnificent sense of humor.

The word **'moreover'** contributes to greater unity or cohesion between sentences and allows the text to flow more smoothly.

Style

Sentence Length Info

Total Sentences: 315
Avg. Length: 12.9 words
Short Sentences (< 17 words): 209 (66%)
Long Sentences (> 35 words): 6 (2%)
Sentence Variation: 11.8 words (std deviation)

Your paper seems to be lacking in complex sentences (http://blog.paperrater.com/2015/05/effective-use-of-sentence-length.html) , due to overuse of shorter sentences.

Line chart of the length of each sentence (first 50 sentences). A jagged chart indicates variation.

Helpful Resources:

- Effective Use of Sentence Length (http://blog.paperrater.com/2015/05/effective-use-of-sentence-length.html)

Vocabulary Words

Usage of Academic Vocabulary

Vocabulary Score: 254.56
This score is based on the quantity and quality of scholarly vocab words found in the text. You did equal or better than **78%** of the people in your grade.

Vocabulary Word Count: 219
Percentage of Vocab Words: 6.57%
Vocab Words in this Paper (top 20):
leverages, mitigate, implementation, integrating, mechanisms, activating, framework, integrates, employs, epochs, groundwork, refinements, predictive, monitoring, evaluation, external, findings, undertake, enhance, measures

Great work! Your paper scored well above average for the vocabulary category. Why don't you check out our Vocab Builder (http://www.PaperRater.com/vocab_builder/index) to keep your skills sharp.

Tips

Whether you are writing for a school assignment or professionally, it is imperative that you have a vocabulary that will provide for clear communication of your ideas and thoughts. You need to know the type and level of your audience and adjust your vocabulary accordingly. It is worthwhile to constantly work at improving your knowledge of words. To help with this task, please consider using our Vocabulary Builder (http://www.PaperRater.com/vocab_builder/index) to improve your comprehension and usage of words.