Project Report

On

FIRE DETECTION WITH ALARM

Submitted in partial fulfillment of the requirement of the award of the degree of

Bachelor of Computer Application

Session 2021 - 2024

Submitted By

Name: Prem Kumar

Roll No.: 220219021306

Department of Bachelor of Computer Application

ANNADA COLLEGE HAZARIBAGH

Certificate for Project

This is to certify that this is a bona fide record of the project work done satisfactory at "**SAI CODING SOLUTION"** by **PREM KUMAR** Roll No**. 220219021306,** in partial fulfillment of BCA Examination.

This report or similar report on the topic has not been submitted for any other examination and doesn't form part of any other course undergone by the candidate.

Signature Internal **Signature External**

Certificate for Bonafide

This is to certify that this is a bona fide record of the project work done satisfactory at "**SAI CODING SOLUTION"** by **PREM KUMAR Roll No.220219021306,** in partial fulfillment of BCA Examination.

This report or similar report on the topic has not been submitted for any other examination and doesn't form part of any other course undergone by the candidate.

Date: 18-07-24

Place: RANCHI

MR. KANCHAN RAJU

GUIDE, SAI CODING SOLUTION

RANCHI JHARKHAND

ACKNOWLEDGEMENT

This thesis work has been an intellectually invigorating experience for me. I am sure that the knowledge and experience gathered during the course of this work will make me stand in good stead in future.

With immense pleasure and due respect, I express my sincere gratitude to **(principal),** In charge, Annada College Hazaribagh, for all his support and co-operation in successfully completing this thesis work by providing excellent facilities.

I would also like to extend my sincere gratitude to all faculties' members and staff for helping me in my college during my BCA course.

I would like to take this opportunity to extend my sincere gratitude and thanks to my Pioneer **Prof. Sanjeev Sir,** firstly for coming up with such an innovative thesis idea. He has not only made us to work but guided us to orient toward research. It has been real pleasure working under her guidance and it is chiefly her encouragement and motivation that has made this thesis a reality.

Last, but not the last I am heartily thankful to almighty God for showering his blessing forever during my entire life and also to my family members for providing me a great support.

BCA Annada College Hazaribagh **(PREM KUMAR)**

Declaration by the Candidate

By **PREM KUMAR** Roll No**. 220219021306** hereby declare that the work, which is being presented in the dissertation, entitled **"FIRE DETECTION WITH ALARM",** in partial fulfillment of requirement for the award of the degree of "Bachelor Of computer application**"** submitted in **Vinoba Bhave University, Hazaribagh** is an authentic record of our work carried out under the guidance of **Mr. Kanchan Raju**.

We have not submitted the matter embodied in this dissertation for the award of any other degree.

Date:18-07-24 **PREM KUMAR**

Place: Hazaribagh

Table of Contents:

ABSTRACTIONS

The Fire Detection with Alarm System is an integrated solution designed to detect fire incidents in real-time and alert users through an audible alarm and email notifications. The project leverages computer vision techniques, specifically using OpenCV, to identify fire within a video feed from a webcam or external camera. The system's architecture comprises several key components:

1. **Video Capture and Processing**: The system continuously captures video frames from a camera. These frames are processed using the Haar cascade classifier to detect fire. If fire is detected in any frame, an alarm is triggered, and an email notification is sent to predefined recipients.
2. **User Interface**: The graphical user interface (GUI) is built using Tkinter, providing users with controls to start and stop the fire detection process. The interface also displays the live video feed, making it easy to monitor the camera in real-time.
3. **Alarm System**: When fire is detected, an alarm sound is played using the playsound library. This immediate auditory alert helps to quickly notify anyone in the vicinity of the potential danger.
4. **Email Notification**: To ensure that remote monitoring is possible, the system sends an email alert to a list of recipients. The email includes details such as the location, date, time of detection, and suggested immediate actions.

5. **System Integration and Control**: The integration of video processing, alarm activation, and email notification is managed through multi-threading. This ensures that each component operates efficiently without hindering the performance of others.

The project aims to provide a reliable and easy-to-use fire detection solution that can be deployed in various settings, such as homes, offices, and schools. By combining real-time monitoring with immediate alerts, the system enhances safety and enables prompt responses to potential fire hazards.

INTRODUCTION

Fire safety is a critical concern across residential buildings, commercial establishments, and industrial facilities. Traditional fire detection systems like smoke detectors and heat sensors, while widely used, suffer from drawbacks such as delayed response times, false alarms, and lack of visual verification. These limitations can lead to significant damage and loss of life before appropriate measures are taken.

To address these challenges, we introduce a sophisticated Fire Detection System utilizing advancements in computer vision and machine learning technologies. Developed as a final year project for a Bachelor of Computer Applications (BCA) student, this system aims to deliver a robust solution for real-time fire detection through video surveillance. It is designed to promptly trigger audio alarms and send email notifications to alert authorities or property owners, ensuring swift response and minimizing potential damage.

At the core of this Fire Detection System is a pre-trained fire detection cascade classifier powered by OpenCV, a powerful open-source computer vision library. By analyzing video frames in real-time, the system accurately identifies fire incidents, offering a crucial advantage over traditional methods. Integration of an audio alarm system ensures immediate alerts

to individuals nearby, facilitating rapid evacuation and initial firefighting efforts.

Additionally, the system features email notifications that inform predefined recipients, particularly valuable when premises are unattended or occupants cannot respond promptly. By notifying remote stakeholders, emergency services can be mobilized promptly, potentially preventing catastrophic outcomes.

A user-friendly graphical interface, developed using Tkinter, enables seamless interaction with the system. Users can initiate and halt fire detection, as well as monitor live video feeds, enhancing accessibility and usability. Multithreading ensures concurrent operation of alarm sound and email notification processes with video processing, maintaining system efficiency and responsiveness.

This project not only demonstrates the practical application of theoretical knowledge acquired during the BCA program but also showcases the integration of diverse technologies to address real-world challenges. The Fire Detection System exemplifies modern technology's potential in enhancing safety and security measures, offering a comprehensive solution surpassing traditional fire detection systems.

By providing reliable, real-time detection and alert capabilities, this project significantly contributes to fire safety efforts, potentially saving lives and reducing property damage. It embodies the innovative spirit of BCA education, equipping students to tackle complex issues with creative and effective solutions.

ABOUT THE PROJECT

The Fire Detection with Alarm System project represents a significant advancement in fire safety technology, specifically developed as the final year project for a Bachelor of Computer Applications (BCA) student. This project aims to address critical gaps in traditional fire detection systems, such as smoke detectors and heat sensors, which often suffer from delayed responses, false alarms, and lack of visual verification capabilities.

At its core, the system leverages sophisticated computer vision techniques implemented through Python programming language and OpenCV (Open Source Computer Vision Library). These technologies enable the system to analyze live video feeds in real-time, allowing for the early detection of fire incidents with high accuracy and efficiency.

Key features of the Fire Detection with Alarm System project include:

- **Real-Time Fire Detection**: Utilizing a pre-trained fire detection cascade classifier, the system continuously monitors video streams to identify the presence of fire. This proactive approach ensures prompt detection and immediate response, crucial for minimizing the spread of fire and protecting lives and property.

- **Audible Alarm System**: Upon detecting a fire, the system triggers audible alarms to alert individuals in the vicinity. This immediate notification facilitates swift evacuation and initiates emergency protocols, enhancing overall safety measures.

- **Email Notifications**: In addition to audible alerts, the system sends email notifications to designated recipients, providing detailed information about the fire detection event. This feature ensures that relevant stakeholders are promptly informed, even if they are not physically present at the monitored location.

- **User-Friendly Interface**: Developed using Tkinter, the system boasts an intuitive graphical user interface (GUI) that allows users to interact seamlessly with the application. Users can start and stop the fire detection process, view live video feeds, and manage system settings with ease.

- **Multithreading for Efficiency**: Multithreading is employed to handle concurrent tasks such as video processing, alarm activation, and email sending. This ensures that the system operates efficiently without delays or interruptions, maintaining responsiveness during critical situations.

By integrating these technologies and functionalities, the Fire Detection with Alarm System project aims to significantly improve fire safety measures across various environments, including residential buildings, commercial establishments, and industrial facilities. It exemplifies the practical application of computer vision and automation in addressing real-world safety challenges, preparing BCA students to innovate and contribute effectively in their field of study.

This project not only underscores the importance of leveraging modern technology for enhancing safety and security but also highlights the educational value of applying theoretical knowledge to develop practical solutions. Through comprehensive detection and alert mechanisms, the Fire Detection with Alarm System project aims to mitigate risks, protect lives, and reduce property damage associated with fire incidents.

SYSTEM ANALYSIS

System analysis for the Fire Detection with Alarm System project involves evaluating its components, functionality, and effectiveness in meeting specified requirements. This analysis typically includes:

1. **Requirements Analysis**: Identifying and documenting functional and non-functional requirements such as real-time fire detection, accuracy, responsiveness, user interface (UI) requirements, and integration with alarm systems and email notifications.
2. **System Design**: Detailing the architecture and design choices, including the use of Python for scripting, OpenCV for computer vision tasks such as object detection (fire detection), and Tkinter for creating the graphical user interface. Design considerations also encompass multithreading for concurrent processing of video feeds, alarms, and notifications.
3. **Component Analysis**: Assessing individual components such as the fire detection cascade classifier (pre-trained model), video capture and processing module using OpenCV, UI components developed with Tkinter, and integration of audio alarms and email notification functionalities.
4. **Performance Evaluation**: Testing the system's performance under various conditions, including different

lighting environments, fire sizes, and distances from the camera. Performance metrics may include detection accuracy, response time from detection to alert triggering, and system stability.

5. **Usability and User Experience**: Evaluating the ease of use and effectiveness of the graphical user interface (GUI) developed with Tkinter. This includes user interaction with controls for starting/stopping detection, viewing video feeds, and receiving alerts.
6. **Security and Reliability**: Assessing the system's reliability in continuous operation, robustness against false alarms, and security measures implemented for email notifications (e.g., authentication and secure transmission protocols).
7. **Integration and Deployment**: Analyzing the integration of different system components (Python scripts, OpenCV modules, Tkinter UI) and deployment considerations on various platforms (Windows, Linux). This includes packaging the application for ease of installation and deployment.
8. **Maintenance and Scalability**: Considering the ease of maintenance, future updates, and scalability of the system to accommodate additional features or support for larger environments (e.g., multiple cameras or networked systems).

System analysis ensures that the Fire Detection with Alarm System project meets its intended objectives effectively,

providing a comprehensive evaluation of its design, functionality, performance, and usability in enhancing fire safety measures through advanced technology integration.

OBJECTIVES

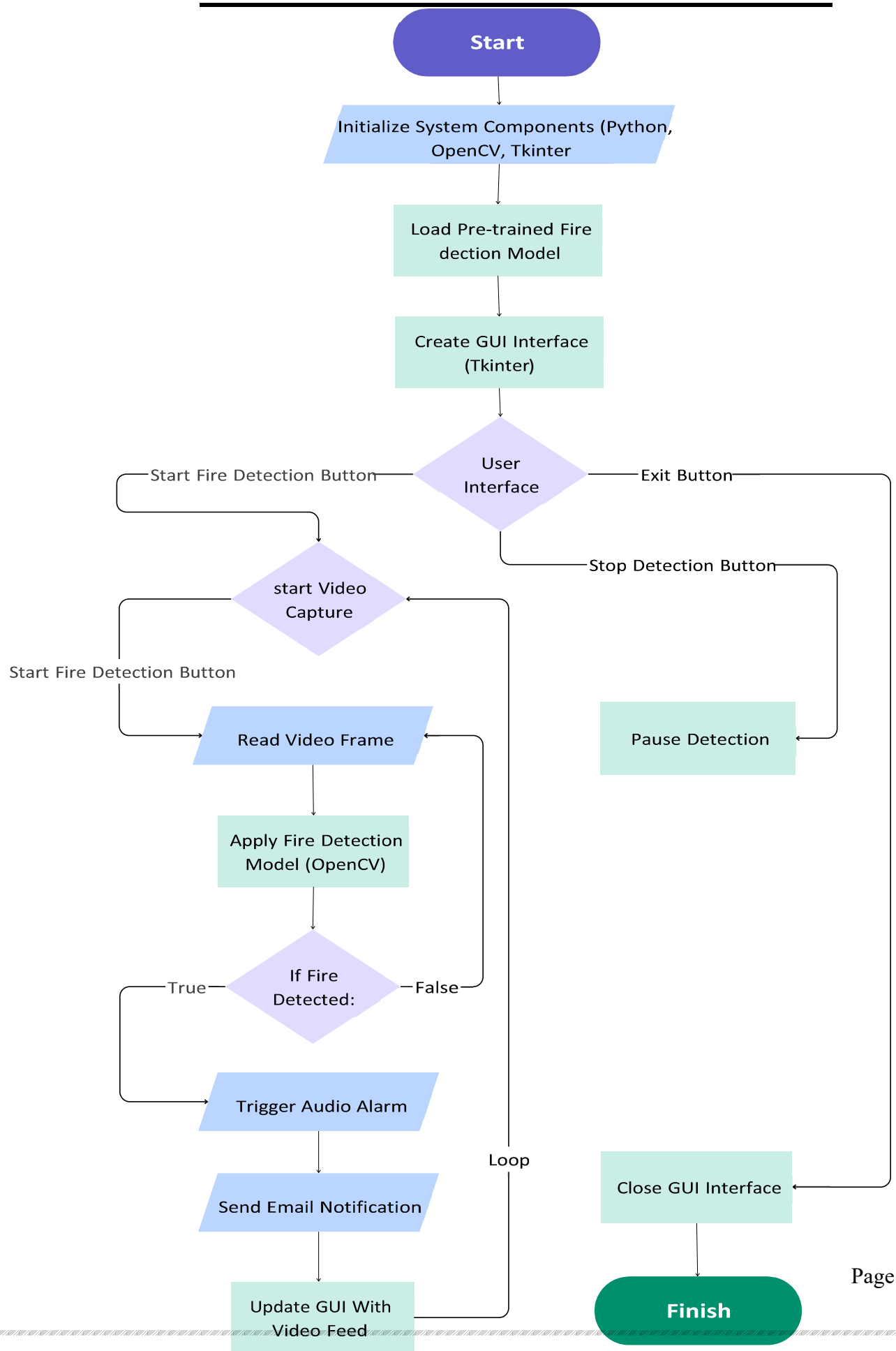
The objective of the Fire Detection System project is to develop a robust and efficient system that utilizes computer vision and machine learning technologies to detect fire incidents in real-time through video surveillance. The primary goals include:

1. **Real-time Fire Detection**: Implementing a system capable of promptly identifying fire in video streams using a pre-trained Haar Cascade classifier.
2. **Alarm Triggering**: Automatically triggering an audio alarm upon detecting fire to alert individuals in the vicinity.
3. **Email Notification**: Sending email notifications to predefined recipients, including authorities or property owners, to facilitate immediate response and action.
4. **User Interface**: Developing a user-friendly graphical interface (GUI) using Tkinter to allow easy interaction and monitoring of the detection process.
5. **Integration of Technologies**: Demonstrating the integration of Python, OpenCV for computer vision tasks, threading for concurrent processes, and Tkinter for GUI development to create a comprehensive fire detection solution.
6. **Enhancing Safety Measures**: Contributing to fire safety by providing a reliable and efficient detection mechanism that can potentially minimize damage and save lives in residential, commercial, and industrial settings.

7. **Application of Academic Knowledge**: Applying theoretical concepts learned during the BCA program to solve practical, real-world challenges in the domain of safety and security.

Overall, the project aims to showcase the application of modern technologies in enhancing safety measures against fire incidents, emphasizing real-time detection and prompt notification capabilities.

Flow Chart


Start
Initialize System Components (Python, OpenCV, Tkinter
Load Pre-trained Fire dection Model
Create GUI Interface (Tkinter)
User Interface
Start Fire Detection Button
Exit Button
Stop Detection Button
start Video Capture
Start Fire Detection Button
Read Video Frame
Pause Detection
Apply Fire Detection Model (OpenCV)
If Fire Detected:
True
False
Trigger Audio Alarm
Send Email Notification
Update GUI With Video Feed
Loop
Close GUI Interface
Finish

CODEING

```
from tkinter import *   # GUI toolKit
import cv2              # Library for openCV
import threading        # Library for threading -- which allows
code to run in backend
import playsound        # Library for alarm sound
import smtplib          # Library for email sending
from PIL import Image, ImageTk
from email.mime.text import MIMEText
from datetime import datetime

class Resize(Frame):
    def __init__(self, master):
        Frame.__init__(self, master)

        # Create a Canvas widget
        self.canvas = Canvas(self, width=900, height=600)
        self.canvas.pack(fill=BOTH, expand=YES)

        # Load the background image
        self.image_bg =
Image.open("C:/Fire_detection_with_alarm/fire1.png")
        self.img_copy_bg = self.image_bg.copy()  # Copy for
resizing

        # Load the overlay image
        self.image_overlay = Image.open("fire_text.png")
        self.img_copy_overlay = self.image_overlay.copy()  # Copy
for resizing

        # Define images using PhotoImage function
        self.background_image = ImageTk.PhotoImage(self.image_bg)
        self.overlay_image =
ImageTk.PhotoImage(self.image_overlay)

        # Create and display the background image on the Canvas
```

```
        self.bg_image_obj = self.canvas.create_image(0, 0,
image=self.background_image, anchor=NW)

        # Create and display the overlay image on the Canvas
        self.overlay_image_obj = self.canvas.create_image(100,
100, image=self.overlay_image, anchor=NW)  # Adjust coordinates
as needed

        # Create the first button with custom styles
        self.button1 = Button(
            self.canvas, text="Start Fire Detection",
command=self.button_click, font=("Helvetica", 15, "bold"),
bg="green", fg="white", activebackground="cyan",
activeforeground="red", padx=10, pady=10, relief=RAISED, bd=10
        )
        self.button_window1 = self.canvas.create_window(10, 10,
anchor=NW, window=self.button1)  # Initial coordinates

        # Create the second button with custom styles
        self.button2 = Button(
            self.canvas, text="Exit", command=self.button_click2,
font=("Helvetica", 15, "bold"), bg="red", fg="white",
activebackground="purple", activeforeground="red", padx=10,
pady=10, relief=RAISED, bd=10
        )
        self.button_window2 = self.canvas.create_window(200, 10,
anchor=NW, window=self.button2)  # Initial coordinates

        # Create the third button with custom styles
        self.button3 = Button(
            self.canvas, text="Stop Detection",
command=self.stop_detection, font=("Helvetica", 15, "bold"),
bg="blue", fg="white", activebackground="orange",
activeforeground="red", padx=10, pady=10, relief=RAISED, bd=10
        )
        self.button_window3 = self.canvas.create_window(400, 10,
anchor=NW, window=self.button3)  # Initial coordinates

        # Create a frame to hold the video feed
```

```
        self.video_frame = Frame(self.canvas, width=640,
height=480, bg="black")
        self.video_frame.place(x=130, y=420)  # Adjust
coordinates to place it at the bottom

        # Create a label to display the video feed within the
frame
        self.video_label = Label(self.video_frame)
        self.video_label.pack(fill=BOTH, expand=YES)

        # Bind function resize_background to screen resize
        self.canvas.bind('<Configure>', self.resize_all)

        self.stop_flag = False  # Flag to control the video feed
loop
        self.runOnce = False  # Flag to track whether actions
have been initiated

    def resize_all(self, event):
        # Get the new width and height for background image
        new_width_bg = event.width
        new_height_bg = event.height

        # Resize the background image according to new dimensions
        self.image_bg = self.img_copy_bg.resize((new_width_bg,
new_height_bg))

        # Define new background image using PhotoImage function
        self.background_image = ImageTk.PhotoImage(self.image_bg)

        # Update background image on the Canvas
        self.canvas.itemconfig(self.bg_image_obj,
image=self.background_image)

        # Calculate new size for the overlay image
        overlay_width = new_width_bg // 1  # Example: Resize to
quarter the width of the background
        overlay_height = new_height_bg // 2  # Example: Resize to
quarter the height of the background
```

```
        # Resize the overlay image according to new dimensions
        self.image_overlay =
self.img_copy_overlay.resize((overlay_width, overlay_height))

        # Define new overlay image using PhotoImage function
        self.overlay_image =
ImageTk.PhotoImage(self.image_overlay)

        # Update overlay image on the Canvas
        self.canvas.itemconfig(self.overlay_image_obj,
image=self.overlay_image)

        # Reposition the overlay image on the Canvas
        overlay_x = new_width_bg // 2 - overlay_width // 2  #
Centered horizontally
        overlay_y = new_height_bg // 2.8 - overlay_height // 1  #
Centered vertically
        self.canvas.coords(self.overlay_image_obj, overlay_x,
overlay_y)

        # Calculate new coordinates for the first button
        button_x1 = new_width_bg // 2 - overlay_width // 2.5 #
Adjust for button width
        button_y1 = new_height_bg // 2 - overlay_height // 2  #
Below the overlay image

        # Reposition the first button on the Canvas
        self.canvas.coords(self.button_window1, button_x1,
button_y1)

        # Calculate new coordinates for the second button
        button_x2 = new_width_bg // 1 - overlay_width // 3   #
Adjust for button width
        button_y2 = new_height_bg // 2 - overlay_height // 2  #
Below the overlay image

        # Reposition the second button on the Canvas
```

```
        self.canvas.coords(self.button_window2, button_x2,
button_y2)

        # Calculate new coordinates for the third button
        button_x3 = new_width_bg // 2 - overlay_width // 8 #
Adjust for button width
        button_y3 = new_height_bg // 2 - overlay_height // 2  #
Below the overlay image

        # Reposition the third button on the Canvas
        self.canvas.coords(self.button_window3, button_x3,
button_y3)

        # Adjust the video frame position to the center of the
bottom
        self.video_frame.place(x=(new_width_bg - 640) // 2,
y=new_height_bg - 500)

    def button_click(self):
        self.stop_flag = False  # Reset the stop flag
        fire_cascade =
cv2.CascadeClassifier('C:/Fire_detection_with_alarm/fire_detectio
n_cascade_model.xml') # To access xml file which includes
positive and negative images of fire. (Trained images)

        vid = cv2.VideoCapture(0) # To start camera this command
is used "0" for laptop inbuilt camera and "1" for USB attached
camera

        def play_alarm_sound_function(): # defined function to
play alarm post fire detection using threading
            playsound.playsound('C:/Fire_detection_with_alarm/fir
e_alarm.mp3',True) # to play alarm # mp3 audio file is also
provided with the code.
            print("Fire alarm end") # to print in consol

        def send_mail_function(): # defined function to send mail
post fire detection using threading
```

```
            recipientmails = ["codewithprem12345@gmail.com",
"premst12345@gmail.com", "premkrst12345@gmail.com"]  # List of
recipient emails
            location = "Annada College Hzb, Building A, Floor 1,
Room 48"
            detection_time = datetime.now().strftime("%d-%m-%Y
%I:%M:%S:%p")
            severity = "Medium"
            immediate_actions = "Evacuate the building
immediately and call the fire department."

            # Email content
            subject = "Urgent: Fire Detected!"
            body = f"""
Dear Team,

This is to alert you that a fire has been detected at the
following location:

Location: {location}
Date & Time: {detection_time}
Severity: {severity}
Immediate Actions Required: {immediate_actions}

Please take necessary precautions and follow the emergency
protocols immediately.

Stay safe,
[Prem Kumar Team]
                    """

            # Create the MIMEText object
            msg = MIMEText(body)
            msg['Subject'] = subject
            msg['From'] = "premkumar123456b1@gmail.com"  #
Sender's email address
            msg['To'] = ", ".join(recipientmails)  # Join
recipients into a comma-separated string
```

```
            try:
                server = smtplib.SMTP('smtp.gmail.com', 587)
                server.ehlo()
                server.starttls()
                server.login("premkumar123456b1@gmail.com", 'aaaa
bbbb cccc dddd') # Sender's email login id and password
                server.sendmail(msg['From'], recipientmails,
msg.as_string())  # Correct usage of sendmail()
                print("Alert mail sent successfully to
{}".format(", ".join(recipientmails))) # Confirmation message
                server.close() # Close the server
            except Exception as e:
                print(e) # Print any errors encountered

        def show_frame():
            if self.stop_flag:
                vid.release()
                cv2.destroyAllWindows()
                return

            ret, frame = vid.read() # Value in ret is True # To
read video frame
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # To
convert frame into gray color
            fire = fire_cascade.detectMultiScale(frame, 1.2, 5) #
to provide frame resolution

            ## to highlight fire with square
            for (x, y, w, h) in fire:
                cv2.rectangle(frame, (x-20, y-20), (x+w+20,
y+h+20), (255, 0, 0), 2)
                roi_gray = gray[y:y+h, x:x+w]
                roi_color = frame[y:y+h, x:x+w]

                print("Fire alarm initiated")
                threading.Thread(target=play_alarm_sound_function
).start()  # To call alarm thread

                if not self.runOnce:
```

```
                    print("Mail send initiated")
                    threading.Thread(target=send_mail_function).s
tart() # To call alarm thread
                    self.runOnce = True

            cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA) #
Convert the image to RGBA
            img = Image.fromarray(cv2image) # Convert image to
PIL format
            imgtk = ImageTk.PhotoImage(image=img) # Convert image
to ImageTk format

            self.video_label.imgtk = imgtk
            self.video_label.configure(image=imgtk)
            self.video_label.after(10, show_frame)

        show_frame()

    def stop_detection(self):
        self.stop_flag = True

    def button_click2(self):
        print("Exiting the application")
        app.quit()  # This will exit the application

if __name__ == "__main__":
    app = Tk()
    app.title("Fire Detection with Alarm system")
    app.iconbitmap('C:/Fire_detection_with_alarm/fire_icon.ico')
    app.geometry("900x600")

    e = Resize(app)
    e.pack(fill=BOTH, expand=YES)

    app.mainloop()

                server.login("premkumar123456b1@gmail.com", 'aaaa
bbbb cccc dddd') # Sender's email login id and password
```

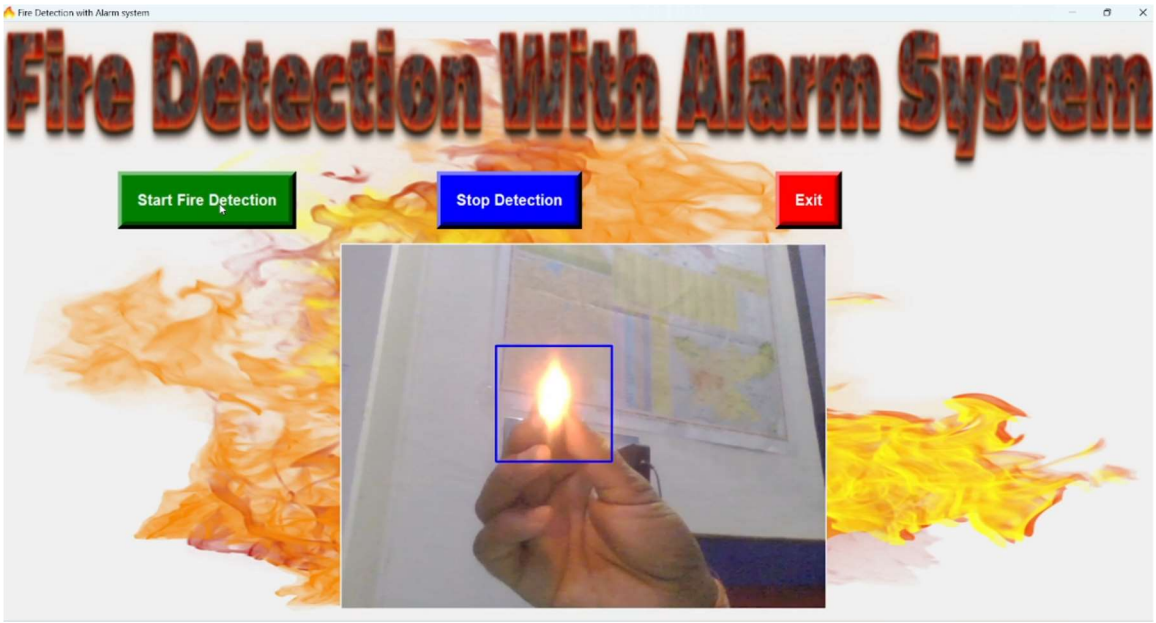
```
                server.sendmail(msg['From'], recipientmails,
msg.as_string())  # Correct usage of sendmail()
                print("Alert mail sent successfully to
{}".format(", ".join(recipientmails))) # Confirmation message
                server.close() # Close the server
            except Exception as e:
                print(e) # Print any errors encountered

        def show_frame():
            if self.stop_flag:
                vid.release()
                cv2.destroyAllWindows()
                return

            ret, frame = vid.read() # Value in ret is True # To
read video frame
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # To
convert frame into gray color
            fire = fire_cascade.detectMultiScale(frame, 1.2, 5) #
to provide frame resolution

            ## to highlight fire with square
            for (x, y, w, h) in fire:
                cv2.rectangle(frame, (x-20, y-20), (x+w+20,
y+h+20), (255, 0, 0), 2)
                roi_gray = gray[y:y+h, x:x+w]
                roi_color = frame[y:y+h, x:x+w]

                print("Fire alarm initiated")
                threading.Thread(target=play_alarm_sound_function
).start()  # To call alarm thread

                if not self.runOnce:
                    print("Mail send initiated")
                    threading.Thread(target=send_mail_function).s
tart() # To call alarm thread
                    self.runOnce = True
```

```
            cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA) #
Convert the image to RGBA
            img = Image.fromarray(cv2image) # Convert image to
PIL format
            imgtk = ImageTk.PhotoImage(image=img) # Convert image
to ImageTk format

            self.video_label.imgtk = imgtk
            self.video_label.configure(image=imgtk)
            self.video_label.after(10, show_frame)

        show_frame()

    def stop_detection(self):
        self.stop_flag = True

    def button_click2(self):
        print("Exiting the application")
        app.quit()  # This will exit the application

if __name__ == "__main__":
    app = Tk()
    app.title("Fire Detection with Alarm system")
    app.iconbitmap('C:/Fire_detection_with_alarm/fire_icon.ico')
    app.geometry("900x600")

    e = Resize(app)
    e.pack(fill=BOTH, expand=YES)

    app.mainloop()
```

OUTPUR

Window Screen tkinter GUI:

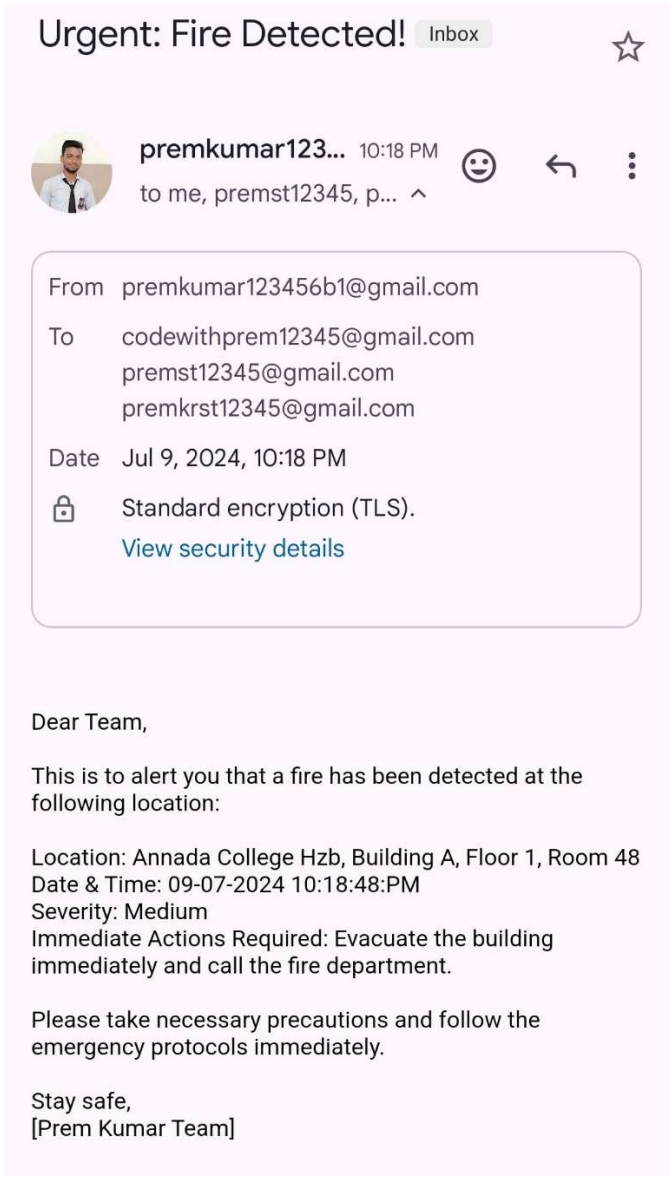
Open Camera Fire Detection:



Terminal Ouput:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        Code
on.py"
Fire alarm initiated
Mail send initiated
Alert mail sent successfully to codewithprem12345@gmail.com, premst12345@gmail.com,
premkrst12345@gmail.com
Fire alarm end
Fire alarm initiated
```

Email Notification:



CONCLUSIONS

The Fire Detection System project demonstrates the integration of advanced technologies in enhancing fire safety through early detection and timely alerts. By combining real-time video processing, computer vision techniques, and automated notification systems, the project addresses critical aspects of fire safety with the following key outcomes:

1. **Effective Real-Time Detection**: The system successfully captures and processes live video feeds to detect the presence of fire with high accuracy. The use of a pre-trained Haar Cascade classifier ensures that the detection mechanism is both reliable and efficient, capable of identifying fire in various conditions and environments.
2. **Immediate Alerts**: The integration of an automated audio alarm and email notification system ensures that alerts are sent promptly when a fire is detected. This dual alert mechanism maximizes the chances of quick response, both locally through the audio alarm and remotely via email notifications.
3. **User-Friendly Interface**: The intuitive graphical user interface (GUI) developed using Tkinter provides users with an easy-to-use platform for interacting with the system. Users can effortlessly start and stop the detection process, view the real-time video feed, and monitor detection

results, making the system accessible to individuals with varying levels of technical expertise.

4. **Enhanced Safety and Reliability**: The system's ability to operate with minimal latency and its robustness in detecting actual fire events while minimizing false alarms contribute significantly to enhancing overall safety. Regular updates and maintenance of the classifier model can further improve the system's accuracy and reliability over time.
5. **Scalability and Modularity**: The modular design of the system allows for easy maintenance and scalability. The system can be adapted to different video resolutions, camera types, and environments, making it a versatile solution for various fire detection applications.
6. **Educational Value**: For final year BCA students, this project provides an excellent opportunity to apply theoretical knowledge in a practical context. It encompasses various aspects of software development, including real-time data processing, machine learning, user interface design, and multithreading, offering a comprehensive learning experience.

In conclusion, the Fire Detection System project not only serves as a valuable tool for enhancing fire safety but also stands as a testament to the capabilities of modern technology in addressing real-world challenges. It underscores the importance of integrating different technological components

to create a cohesive and effective solution. The successful implementation of this project highlights the potential for further innovations in the field of automated safety systems.