ALERTIFY

MINI PROJECT REPORT

Submitted by

Roshan S(AJC22IT064)

in partial fulfillment for the award of

the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

APRIL 2025

DEPARTMENT OF INFORMATION TECHNOLOGY
AMAL JYOTHI COLLEGE OF ENGINEERING, KANJIRAPALLY

DECLARATION

We undersigned hereby declare that the project report "**Alertify**", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Ms Mini Joswin**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Name of student
Roshan S

DEPARTMENT OF INFORMATION TECHNOLOGY

AMAL JYOTHI COLLEGE OF ENGINEERING, KANJIRAPALLY

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**ALERTIFY**" is a bonafide record of the mini project work carried out by **Roshan S (AJC22IT064),** during the academic year 2022 -2023 in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology by APJ Abdul Kalam Technological University.

Ms. Mini Joswin
Assistant Professor
(Guide)

Dr. Manoj T Joy
Head of the Department

PROJECT COORDINATOR

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all, we submit all glory and honour to **GOD ALMIGHTY** for his eternal love and protection throughout the project. The tremendous amount of help that we received from a number of people gave us the incentive to complete this project to the best of our capability.

I express my sincere thanks to **Rev.Fr.Dr. Roy Abraham**, Manager of our college for giving an opportunity to undertake this project. I am very much grateful to **Dr. Lillykutty Jacob**, Principal of our college for supporting all the way long. I express my special gratitude to the HOD of Information Technology **Dr. Manoj T Joy**, for providing me constant guidance and encouragement throughout the project.

I also express my gratitude to the Project Coordinators, **Mr Jaison Mathew John** Assistant Professor and **Ms Mini Joswin**, Assistant Professor for their inspiration and suggestions.

I feel deeply honored in expressing sincere thanks to my guide **Ms Mini Joswin** for his guidance and support in completing the project. I have to appreciate the guidance
given by the panel members during the project presentation, thanks to their comments and advice.

Last but not the least, I place a deep sense of gratitude to my family members and my friends who have been constant source of inspiration during the preparation of this project work.

ABSTRACT

Drowsiness detection is a critical application in enhancing road safety by preventing accidents caused by fatigued drivers. This project helps to introduces a novel approach to integrate a drowsiness detection model within a React Native-based mobile application designed for vehicles. The system employs advanced computer vision techniques and then machine learning algorithms to monitor driver behaviour in real time, identifying signs of drowsiness through facial landmarks and eye movement patterns. Upon detecting drowsiness, the application have triggers an immediate alert to warn the driver. Simultaneously, the app sends an alert message to a pre-selected list of emergency on their contacts, ensuring timely assistance. Leveraging the capabilities of React Native, the application offers a seamless, cross-platform that interface with features such as destination tracking and personalized notification settings. The proposed system is designed to be lightweight and efficient, ensuring smooth performance on mobile devices. This integration of drowsiness detection with message sharing through SMS represents a significant step forward in utilizing mobile technologies for road safety.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL BACKGROUND

Drowsiness-related accidents pose a major threat to road safety, workplace efficiency, and overall well-being. Fatigue impairs cognitive functions, slows reaction times, and increases the likelihood of critical errors, making it essential to develop proactive solutions. ‘Alertify’ is an intelligent drowsiness detection application designed to mitigate such risks by continuously monitoring facial features for signs of fatigue. Developed using React Native for a seamless mobile experience and Flask for efficient backend processing, the application employs real-time image analysis techniques to detect signs of drowsiness, such as drooping eyelids and prolonged blinking.

When drowsiness is detected, 'Alertify' instantly notifies the user's emergency contact list, ensuring that necessary action can be taken to prevent potential hazards. This real-time alert system provides an extra layer of safety, particularly for drivers, machine operators, and individuals in professions requiring high alertness. By integrating technology with personal safety, 'Alertify' contributes to reducing accidents caused by fatigue, offering a practical and reliable solution for users.

1.2 MOTIVATION

The motivation behind developing ‘Alertify’ stems from the alarming rise in accidents caused by drowsiness, particularly among drivers and machine operators. Fatigue-related incidents account for a significant portion of road accidents and workplace injuries, often resulting in severe consequences. Traditional measures, such as relying on personal awareness or external interventions, are not always effective in preventing these accidents. Recognizing the need for a proactive and technology-driven approach, ‘Alertify’ was created to provide real-time monitoring and immediate alerts to ensure safety.

By leveraging facial analysis techniques, the application detects early signs of drowsiness and notifies emergency contacts before a critical situation arises. The goal is to minimize accidents, enhance personal safety, and promote responsible behavior in situations where alertness is crucial. With the increasing availability of mobile technology, ‘Alertify’ offers a practical and accessible solution for individuals who frequently travel or work in high-risk environments. This project is driven by the belief that technology can play a crucial role in saving lives and preventing avoidable tragedies.

1.3 OVERVIEW OF THE REPORT

This report provides a comprehensive overview of ‘Alertify,’ a drowsiness detection application developed using React Native and Flask. It begins with the background and motivation behind the project, highlighting the need for a real-time fatigue detection system. The report then details the system architecture, implementation process, and key technologies used for facial analysis and alert mechanisms. Additionally, it discusses the functionality, features, and practical applications of the app. Finally, the report concludes with an evaluation of the system’s effectiveness, potential improvements, and future scope in enhancing safety and accident prevention.

1.4 PROPOSED SYSTEM

‘Alertify,’ is a drowsiness detection application that utilizes React Native for the front end and Flask for backend processing to ensure seamless performance and real-time analysis. The system captures live video input from the user’s device, processes facial features such as eye movement and blinking patterns, and detects signs of drowsiness using image analysis techniques. React Native provides a cross-platform mobile interface, ensuring accessibility across different devices, while Flask handles the backend operations, including data processing and alert generation. When drowsiness is detected, the system automatically sends alert messages to the user's emergency contact list, enabling timely intervention. By integrating real-time facial analysis with an automated alert mechanism, 'Alertify' offers a reliable, efficient, and scalable solution to mitigate fatigue-related risks and enhance personal safety.

A key feature of 'Alertify' is its emergency alert mechanism. Upon detecting drowsiness, the system automatically sends warning messages to the user's pre-configured emergency contact list. This ensures timely intervention, preventing potential accidents caused by fatigue. The system is designed to be lightweight, efficient, and scalable, making it suitable for individuals in high-risk professions, frequent travelers, and anyone who needs continuous monitoring of their alertness levels.

1.5 OBJECTIVES

The primary objectives of the **Alertify** are:

1. **To develop a real-time drowsiness detection system** using facial analysis techniques for early identification of fatigue.

2. **To implement a cross-platform mobile application** using React Native for seamless accessibility on both Android and iOS devices.

3. **To integrate Flask as the backend framework** for efficient data processing and communication between the mobile app and server.

4. **To provide an automated alert system** that sends warning messages to the user's emergency contact list upon detecting drowsiness.

5. **To enhance road safety and workplace security** by reducing accidents caused by fatigue-related impairments.

1.6 SIGNIFICANCE OF THE SYSTEM

Its ability to proactively detect drowsiness and prevent accidents caused by fatigue. Drowsiness is a major factor in road accidents and workplace hazards, often leading to severe consequences due to delayed reaction times and impaired decision-making. By utilizing real-time facial analysis, ‘Alertify’ provides an intelligent monitoring system that enhances personal safety and reduces the risk of fatigue-related incidents.

With its cross-platform functionality using React Native, the system ensures accessibility on both Android and iOS devices, making it widely usable. The integration of Flask for backend processing enables efficient data handling and fast response times, ensuring real-time detection and alert generation. The emergency alert mechanism adds an extra layer of security by notifying designated contacts when drowsiness is detected, allowing for timely intervention.

This system is particularly beneficial for drivers, machine operators, night shift workers, and individuals in high-risk professions where sustained alertness is crucial. By providing a reliable, scalable, and user-friendly solution, ‘Alertify’ contributes to improving road safety, workplace efficiency, and overall well-being, making it a valuable tool for accident prevention and personal security.

CHAPTER 2

EXISTING SYSTEM

Drawbacks of Existing Systems

Existing drowsiness detection systems often face several limitations that impact their reliability and effectiveness. Many systems struggle with limited detection accuracy, especially in low-light conditions or for users wearing glasses, making them less reliable. Additionally, a lack of real-time alerts in some solutions prevents immediate intervention, reducing their ability to prevent accidents effectively. Several existing technologies require external hardware, such as wearables or specialized sensors, increasing costs and making them less accessible to users. Moreover, most systems do not integrate an emergency contact feature, failing to notify caregivers or authorities in critical situations. High power consumption is another concern, as continuous video processing drains battery life quickly, making long-term use impractical. Lastly, environmental factors such as poor lighting, extreme weather, and different face angles can affect detection reliability, limiting their real-world effectiveness. Alertify addresses these challenges by providing an accurate, real-time, and user-friendly solution with emergency contact integration, ensuring a safer and more efficient drowsiness detection system.With its advanced facial analysis and automated alert system, Alertify enhances road safety and significantly reduces the risk of fatigue-related accidents. The application is designed to be lightweight and efficient, ensuring smooth operation without excessive battery drain or performance lag.

CHAPTER 3

SYSTEM REQUIREMENTS AND DESIGN OVERVIEW

3.1 FUNCTIONAL REQUIREMENTS

The **Alertify** must fulfill the following functional requirements:

1. **Real-time Drowsiness Detection:** The system should continuously monitor the user's facial features to detect signs of drowsiness.

2. **Facial Analysis Processing:** The application should analyze eye movement, blinking patterns, and head position using the smartphone camera.

3. **Cross-Platform Compatibility:** The system should function smoothly on both Android and iOS devices using React Native.

4. **Backend Data Processing:** The Flask-based backend should handle facial analysis, data processing, and alert mechanisms efficiently.

5. **Emergency Alert System:** Upon detecting drowsiness, the application should automatically send alert messages to the user’s emergency contact list.

3.2 SYSTEM DESIGN

- User Interface (Frontend) – A cross-platform mobile application built using React Native, providing a seamless and user-friendly experience for real-time drowsiness detection, emergency contact management, and alert notifications.

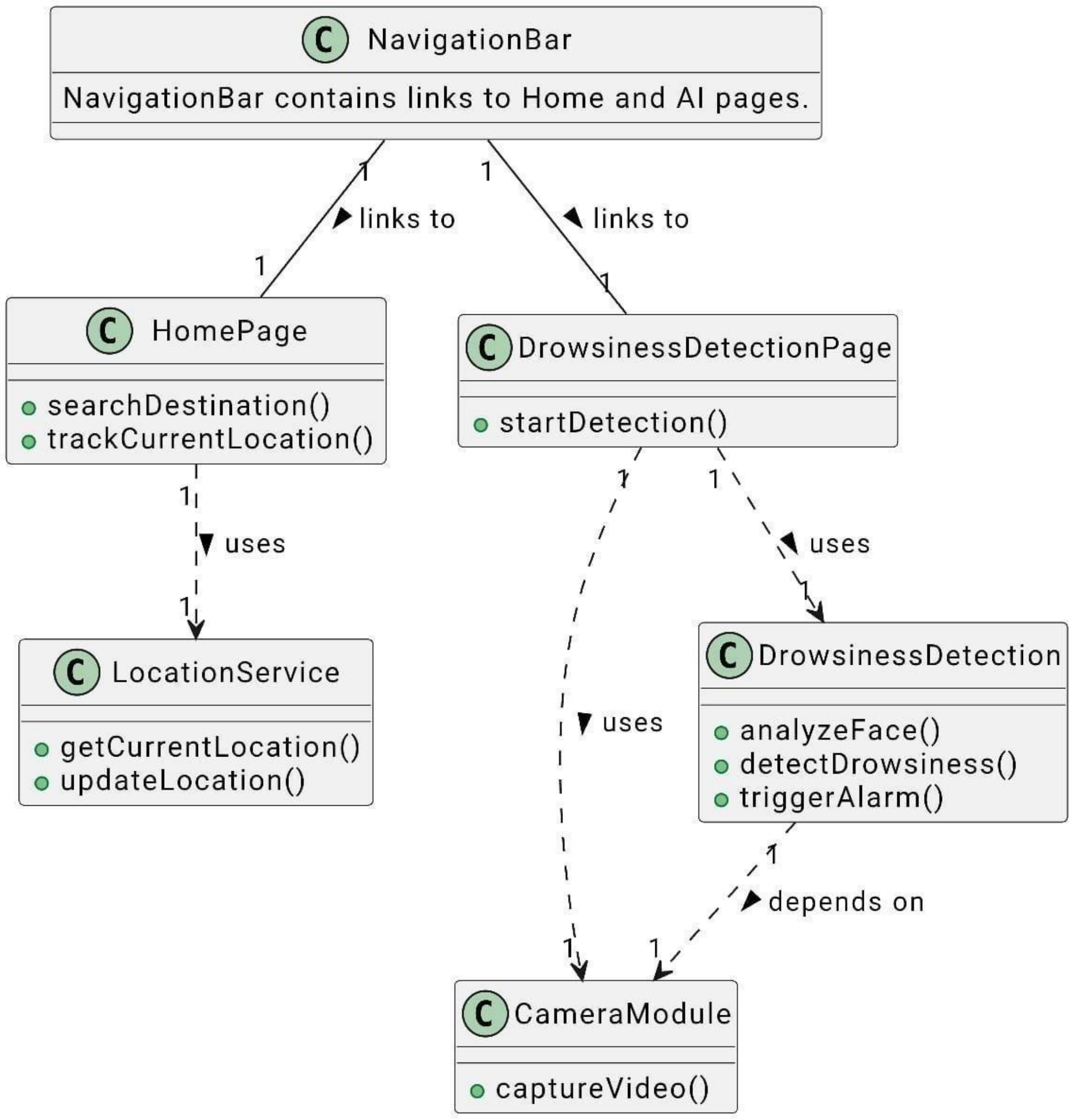
- Backend Processing (Flask Server) – A lightweight and efficient backend developed using Flask, handling facial analysis, drowsiness detection logic, and communication between the mobile app and server.

- Facial Analysis and Drowsiness Detection Module – Utilizes OpenCV and deep learning-based techniques to analyze facial landmarks, including eye movement, blinking rate, and head position, for detecting fatigue.

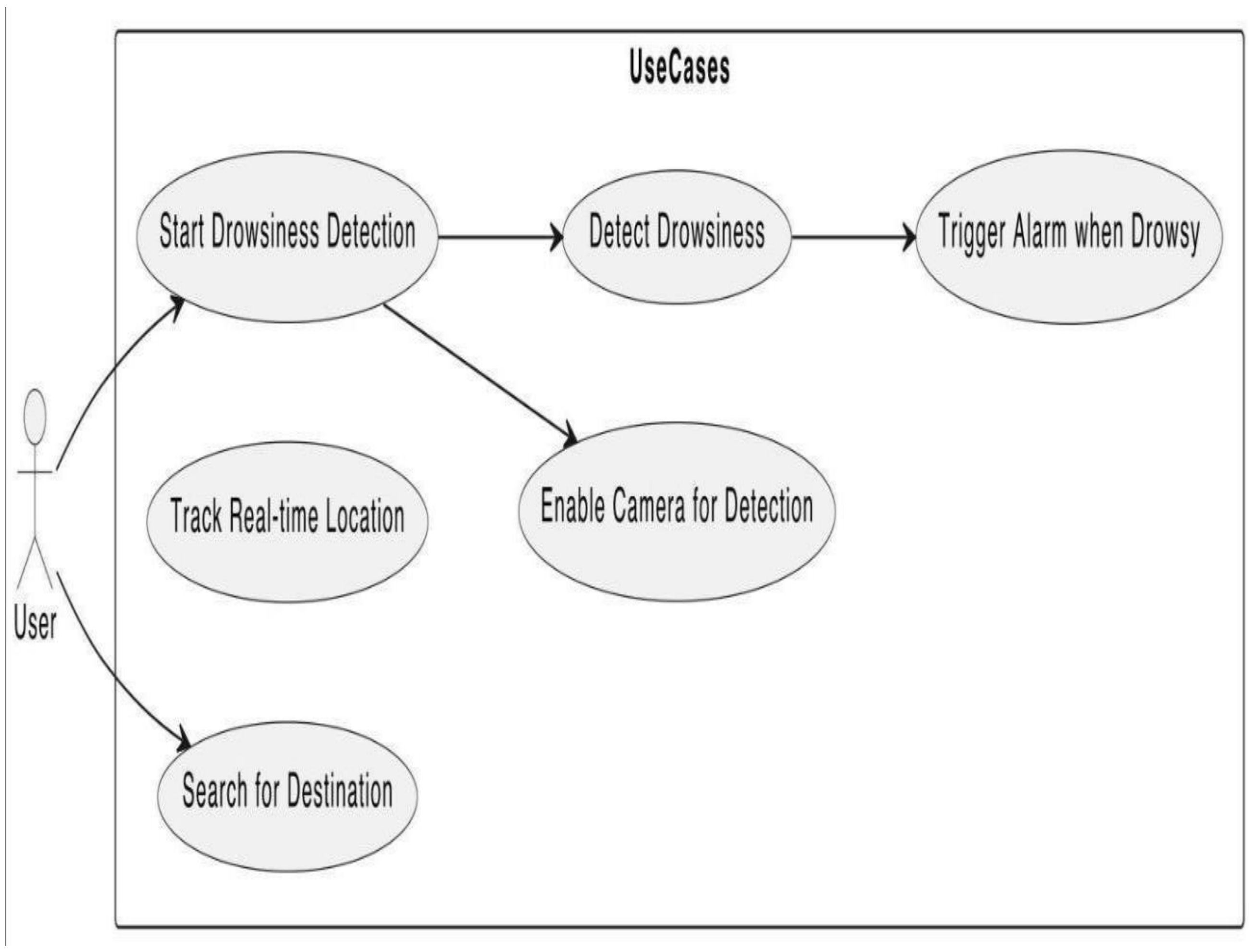
- Emergency Alert System – Automatically sends SMS or in-app notifications to the user's predefined emergency contact list when drowsiness is detected, ensuring timely intervention.

- Analytics System – Keeps track of past drowsiness detection events, allowing users to view trends and patterns in their fatigue levels.

3.2.1 CLASS DIAGRAM



3.2.1 USE-CASE DIAGRAM


UseCases
Start Drowsiness Detection
Detect Drowsiness
Trigger Alarm when Drowsy
Track Real-time Location
Enable Camera for Detection
User
Search for Destination

CHAPTER 4

IMPLEMENTATION

It provides a real-time monitoring feature that allows users to send emergency alerts to pre-selected contacts when needed. The app also tracks the user's live location and shares it with emergency contacts for timely assistance.

- Real-Time Drowsiness Detection: Continuously monitors facial landmarks and eye movements.
- Instant Alert System: Triggers an alarm to wake the driver upon detecting drowsiness.
- Live Location Sharing: Sends the driver's real-time location to emergency contacts.
- Cross-Platform Compatibility: Works efficiently on both Android and iOS devices.

4.1 Project Module

- Purpose: Tracks the user's live location in real time.
- Features:
    - Continuous location monitoring using GPS.Real-time updating and sharing of the location with emergency contacts.

- Purpose: Triggers alerts when the user presses the "Start" button or in case of an emergency.
- Features:
  - Push notifications to users and emergency contacts.
  - Alert system that sends text, call, or location alerts based on the situation.

- Purpose: Detects signs of drowsiness in the user.
- Features:
  - Facial recognition using camera input to analyze eye movement and facial landmarks.

- Purpose: Allows users to add, edit, and manage emergency contacts.
- Features:
  - Adding and removing emergency contacts.
  - Customizable notification settings for alerts to specific contacts

- Purpose: Provides a smooth and intuitive interface for users.
- Features:
  - Easy-to-navigate design with minimal clutter.
  - Interactive features such as the "Start" button for emergency situations.

CHAPTER 5

TESTING AND RESULT

5.1 Validation

- **Integration Testing:** Ensures that different modules work together seamlessly, validating data flow and communication between components.
- **System Testing:** Validates the complete application's functionality and performance as a whole, ensuring all features work in harmony.
- **Usability Testing:** Assesses the user interface and experience to ensure the application is easy to use and intuitive.
- **Performance Testing:** Evaluates the app's speed, scalability, and resource usage under various conditions to ensure it operates efficiently.

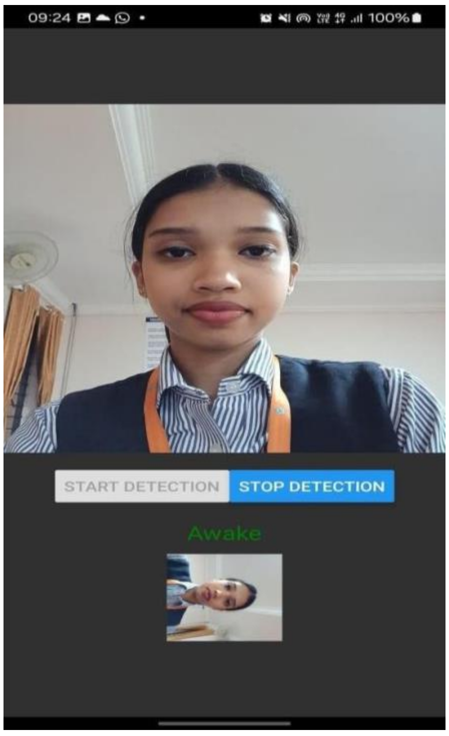
5.2 Output

- **Facial Detection Success:** The system successfully detects drowsiness through facial landmarks and eye movement analysis.
- **Emergency Contact Message:** The app successfully sends a location-based emergency alert to pre-selected contacts.
- **App Integration :** All modules (authentication, location tracking, alerting) work together seamlessly without issues.

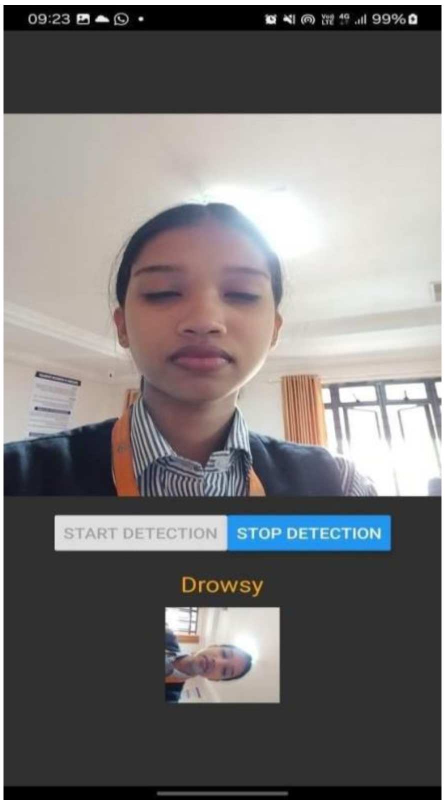
5.3 Result

Alertify

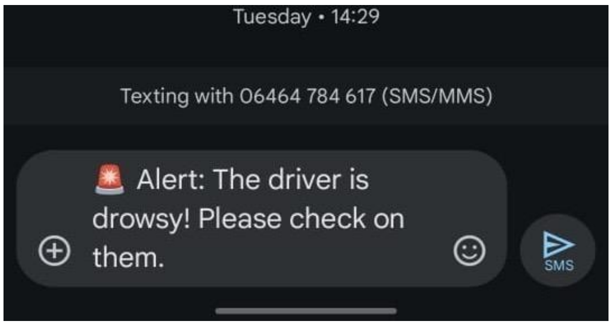
This is an application design for alertify (drowsiness detection). By starting this application camera will automatically displayed as given below inorder to check if the driver feels drowsy



This is an image of a person whose eyes are widely open so that it detects awake.This is the second page that comeup by opening the application.



A close-up of someone with their eyes fully closed, displaying a deep sense of drowsiness. The image captures a moment of fatigue, where the person is on the verge of falling asleep.


Tuesday • 14:29

Texting with 06464 784 617 (SMS/MMS)

🚨 Alert: The driver is drowsy! Please check on them.

This is a type of message that have been received by the emergency contact persons

CHAPTER 6

CONCLUSION

The Alertify Drowsiness Detection System is an innovative and efficient solution designed to enhance personal safety by preventing accidents caused by fatigue. Utilizing React Native for a seamless cross-platform mobile experience and Flask for backend processing, the system performs real-time facial analysis to detect drowsiness based on eye movement, blinking rate, and head position. Upon detecting fatigue, it triggers an automated emergency alert system, notifying predefined contacts to ensure timely intervention. Unlike traditional methods that rely on costly wearables or vehicle-based monitoring, Alertify is a cost-effective, accessible, and hardware-independent solution that operates solely through a smartphone camera. With its lightweight architecture, optimized performance, and secure authentication, the system ensures smooth functionality while prioritizing user privacy. By providing a reliable, scalable, and real-time monitoring solution, Alertify plays a crucial role in reducing fatigue-related risks in various high-alert situations such as driving, night shifts, and industrial operations. Future enhancements like AI-driven predictive analysis and cloud integration could further improve its capabilities, making it an even more powerful tool for drowsiness prevention and safety.

REFERENCES

- Driver Drowsiness Detection System: An Approach By Machine Learning Application *[Submitted on 11 Mar 2023]*

  Jagbeer Singh, Ritika Kanojia, Rishika Singh, Rishita Bansal, Sakshi Bansal

- Detection of Driver Cognitive Distraction: A Comparison Study of Stop-Controlled Intersection and Speed-Limited Highway IEEE Transactions on Intelligent Transportation Systems, Volume 17, Issue 6 Pages 1628 - 1637

  Yuan Liao, Shengbo Eben Li, Wenjun Wang, Ying Wang, Guofa Li, Bo Cheng

APPENDIX - A: Source Code

```
from flask import Flask, request, jsonify
import numpy as np
import tensorflow.lite as tflite
import cv2
from flask_cors import CORS
import threading
import os
from collections import deque
import time

app = Flask(__name__)
CORS(app, resources={r"/detect-video": {"origins": "*"}})

# Load TFLite model
try:
    interpreter =
tflite.Interpreter(model_path="drowsiness_model.tflite")
    interpreter.allocate_tensors()
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    input_shape = input_details[0]['shape']  # Example: [1, 24, 24, 3]
    MODEL_IMG_SIZE = (input_shape[1], input_shape[2])  # (Height,
Width)
    REQUIRED_CHANNELS = input_shape[3]  # 3 if RGB, 1 if grayscale
    model_lock = threading.Lock()  # Prevent parallel access to
interpreter
    print(f"✅ Model loaded successfully. Expected shape:
{input_shape}")
except Exception as e:
    print(f"❌ Error loading model: {e}")
    interpreter = None
previous_predictions = deque(maxlen=5)  # Last 5 frames for smoothing

def crop_eyes(image):
    """Detects eyes and crops a zoomed-in region."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(face_roi)

        if len(eyes) >= 2:
            ex1, ey1, ew1, eh1 = eyes[0]
            ex2, ey2, ew2, eh2 = eyes[1]
            # Ensure eyes are in correct order (left to right)
            if ex1 > ex2:
                ex1, ex2 = ex2, ex1
                ey1, ey2 = ey2, ey1
                ew1, ew2 = ew2, ew1
                eh1, eh2 = eh2, eh1

            # Merge both eyes into a single zoomed region
            eye_x = ex1
            eye_y = min(ey1, ey2)
            eye_w = ex2 + ew2 - ex1
eye_h = max(eh1, eh2)

            eye_roi = face_roi[eye_y:eye_y + eye_h, eye_x:eye_x +
eye_w]

            # Zoom in by resizing to 2x for better detail
            zoomed_eye_roi = cv2.resize(eye_roi, (MODEL_IMG_SIZE[0] *
2, MODEL_IMG_SIZE[1] * 2))

            return zoomed_eye_roi

    return None  # No eyes detected

def preprocess_image(image):
    """Preprocesses the image: crops eyes, resizes, normalizes."""
    eye_roi = crop_eyes(image)
    if eye_roi is None:
        return None  # No eyes detected

    resized = cv2.resize(eye_roi, MODEL_IMG_SIZE,
interpolation=cv2.INTER_AREA)
    normalized = resized.astype(np.float32) / 255.0

    if REQUIRED_CHANNELS == 3:
        normalized = cv2.cvtColor(normalized, cv2.COLOR_GRAY2RGB)

    reshaped = np.expand_dims(normalized, axis=0)
    return reshaped

def predict_drowsiness(image):
    """Runs inference and determines drowsiness."""
    if interpreter is None:
        return "Model not loaded"

    image = preprocess_image(image)
 # If no eyes are detected, consider it as "Drowsy"
    if image is None:
        return "Drowsy"

    with model_lock:
        interpreter.set_tensor(input_details[0]['index'], image)
        interpreter.invoke()
        output_data =
interpreter.get_tensor(output_details[0]['index'])

    if isinstance(output_data, np.ndarray) and output_data.size > 0:
        probability = float(output_data[0][0])
        print(f"✅ Model output: {probability}")

        previous_predictions.append(probability)
        smoothed_prob = np.mean(previous_predictions)

        return "Drowsy" if smoothed_prob > 0.5 else "Awake"
    else:
        return "Drowsy"  # If model fails, assume drowsy for safety

@app.route("/")
def home():
    return "Flask backend is running!"

@app.route('/detect-video', methods=['POST'])
def detect_video():
    if 'frame' not in request.files:
        return jsonify({"error": "No video frame provided"}), 400

    frame = request.files['frame']
    frame_path = os.path.join("uploads", frame.filename)
    frame.save(frame_path)
    print(f"✅ Received frame: {frame_path}")
  start_time = time.time()
    image = cv2.imread(frame_path)

    if image is None:
        return jsonify({"error": "Failed to read image"}), 400

    prediction = predict_drowsiness(image)
    processing_time = time.time() - start_time

    return jsonify({"status": prediction, "processing_time":
round(processing_time, 4)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5002, debug=True, threaded=True)
```

APPENDIX - B: User Manual

Step 1: Open the Application

- Launch the Alertify app on your smartphone.
- Ensure the front camera is not obstructed and has a clear view of your face.

Step 2: Start Drowsiness Detection

- Tap on Start Detection to begin real-time facial analysis.
- The app will continuously monitor eye movement, blinking rate, and head position to detect signs of drowsiness.

Step 3: Drowsiness Alert Mechanism

- An SMS alert will be automatically sent to the user's emergency contacts with location details.

Step 4: Managing Emergency Contacts

- Navigate to the emergency contacts section.
- Click add contact to select a person from your contact, type the phone number in the given space
- Save the contact to ensure they receive alerts when needed.

Step 6: Exit the Application

- Once done, simply close the app.