AI WOMEN SAFETY COMPANION

Submitted by

SANJAVI S 810422205142

SIVASAKTHI V 810422205158

SUBIKSHA M 810422205160

Of

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE(AUTONOMOUS)

PERAMBALUR – 621 212

A MINI PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION TECHNOLOGY

In partial fulfillment of the requirements

for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

ANNA UNIVERSITY CHENNAI - 600 025

NOV-DEC 2025

BONAFIDE CERTIFICATE

Certified that this mini project report title **"AI WOMEN SAFETY COMPANION"** is the bonafied work of **"SANJAVI S (810422205142), SIVASAKTHI V (810422205158), SUBIKSHA M(8104222160)"** who carried out the research under my supervision,certified further,that to the best of my knowledge the work reported here in does not form part of any project report or dissertion on the basis of which a degree or award was conferred on an earlier occasion on this or any other candiadate.

SUPERVISOR

Ms.K.LALITHAVANI,M.Tech.,

Assistant professor,

Department of Information Technology

Dhanalakshmi Srinivasan Engineering

College(Autonomous),Perambalur -621212.

HEAD OF THE DEPARTMENT

Mr.S.SARAVANAN,M.Tech.,M.B.A.,(Ph.D).,

Professor & Head,

Department of Information Technology

Dhanalakshmi Srinivasan Engineering

College(Autonomous),Perambalur - 621212.

Submitted for Mini Project Viva-Voce Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

It is with immense pleasure that I present my first venture in the field of real application of computing in the form of project work. First, I am indebted to the Almighty for his choicest blessing showered on me in completing this endeavor.

I express my sincere thanks to **Shri. A. SRINIVASAN,** Chancellor, **Dhanalakshmi Srinivasan University,** for having given me an opportunity to study in this Institution.

I would also like to acknowledge **Prof.Dr.D.SHANMUGASUNDARAM, M.E.,Ph.D.,F.I.E.,C.Eng.,** Principal, **Dr. K. ANBARASAN, M.E., Ph.D.,** Vice Principal, **Dr.M.CHELLAPPAN,M.E.,Ph.D.,** Controller of Examination **Dhanalakshmi Srinivasan Engineering College (Autonomous), Perambalur** for their moral support and encouragement they have rendered throughout the course. I express my sincere thanks to the Head of the Department **Mr. S. SARAVANAN, M.Tech., M.B.A., (Ph.D).,**for having provided us with all the necessary specifications.

I owe my deepest gratitude to my Internal **Guide Ms.K.LALITHAVANI M.Tech.,** for her guidance and suggestions during this project work.

We render our thanks to all the staff members and programmers of department of **INFORMATION TECHNOLOGY** for their timely assistance.

ABSTRACT

The AI Women Safety Companion is an intelligent voice-enabled wearable device designed to provide real-time protection for women in emergency situations. With growing safety concerns and the frequent unavailability of stable mobile networks in many regions of India, there is a critical need for a system that can operate independently of mobile signals. This project proposes a smart wearable solution powered by Artificial Intelligence, capable of detecting distress through voice commands, emotional tone analysis, and abnormal movement patterns. Once distress is identified, the device instantly triggers an alert to nearby trusted contacts or local networks using short-range communication technologies such as Bluetooth, Wi-Fi Direct, or mesh networks—ensuring rapid help even in remote or signal-poor areas.

The system integrates multiple components, including speech recognition, machine learning–based distress classification, sensor-based activity monitoring, and local communication modules, all implemented primarily using Python frameworks. The wearable continuously listens for predefined voice cues or behavioral changes and processes them through AI models to determine whether the user is under threat. If an emergency is confirmed, the system activates an SOS alert that shares essential details such as user identity, timestamp, and approximate location derived from offline sensors. This makes the device reliable in both urban and rural environments.

The project also emphasizes low-latency operation, high accuracy in distress detection, and energy-efficient processing, making it suitable for real-world use. The wearable is designed to be lightweight, discreet, and user-friendly, ensuring that women can access help effortlessly during high-stress situations. By combining AI-driven safety monitoring with offline alerting capability, this system aims to bridge the technological gap in existing safety applications, which heavily rely on mobile networks and manual actions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
ML	-	Machine Learning
CNN	-	Convolutional Neural Network
IoT	-	Internet of Things
SOS	-	Save Our Souls / Emergency Alert Signal
GPS	-	Global Positioning System
GSM	-	Global System for Mobile Communication
API	-	Application Programming Interface
UI	-	User Interface
UX	-	User Experience
CPU	-	Central Processing Unit
NLP	-	Natural Language Processing
DL	-	Deep Learning

DB	-	Database
IDE	-	Integrated Development Environment
OS	-	Operating System
JSON	-	JavaScript Object Notation
API	-	Application Programming Interface
SDK	-	Software Development Kit
AIWSC	-	AI Women Safety Companion

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

AI Women Safety Companion is an innovative project developed using Python that aims to enhance women's safety through the integration of artificial intelligence, sensor technology, and local communication networks. The system functions as an AI-powered companion capable of detecting signs of distress or abnormal situations and triggering automatic emergency alerts, even in the absence of a mobile signal or internet connectivity.

This project is designed to overcome the limitations of traditional safety applications that depend entirely on mobile networks. Using Python's powerful libraries for machine learning, signal processing, and IoT communication, the system continuously analyzes various data inputs—such as voice tone, body movements, and environmental sounds to identify potential threats. Once distress is detected, the device activates a local mesh communication network or Bluetooth-based alert system to immediately notify nearby contacts and security authorities with the user's approximate location and situation details.

The proposed system demonstrates how AI and Python-based automation can create a reliable, intelligent, and self-sustaining safety solution for women, especially in remote or low-network areas. By combining real-time monitoring, AI-driven decision-making, and offline communication mechanisms, this project aims to provide a trustworthy digital safety companion that empowers women with confidence and protection anytime, anywhere.

1.2 PURPOSE

The main purpose of the **AI Women Safety Companion** project is to develop an intelligent, voice-enabled wearable device that ensures women's safety in all situations, even in areas with poor or no mobile network connectivity. This project aims to use Python-based artificial intelligence algorithms to automatically detect signs of distress through voice, motion, or environmental patterns. Once a potential threat or emergency is identified, the system triggers an instant alert to a local network using Bluetooth or mesh communication, ensuring that nearby contacts or authorities are notified immediately. The goal is to create a reliable, offline safety solution that provides continuous protection, quick response, and peace of mind for women. By integrating AI technology with real-time detection and offline communication, this project promotes women's empowerment and personal security through innovation and smart technology.

1.3 PROBLEM STATEMENT

Women's safety remains a serious social concern due to rising incidents of harassment and assault. Most existing safety systems rely on mobile networks or internet connectivity, which may fail in remote or low-signal areas. In critical moments, victims may be unable to use their phones to seek help manually. There is a strong need for an that can detect distress automatically and send alerts without user intervention. The lack of such an intelligent, offline-capable system delays emergency response. Hence, this project aims to develop an **AI Women Safety Companion** that ensures protection even without a mobile signal.

1.4 MOTIVATION

The motivation behind developing the AI Women Safety Companion arises from the increasing safety challenges faced by women in today's society. Despite the availability of several safety apps and emergency response systems, most of them rely heavily on internet connectivity and manual operation, which makes them ineffective in remote areas or critical situations where immediate help is needed. This gap inspired the idea of creating an intelligent, offline-capable wearable device that can detect distress automatically and trigger alerts without user intervention.

By integrating **Python-based artificial intelligence**, the system can analyze voice patterns, body movements, and environmental changes to recognize potential danger. The ability to send alerts through local networks such as mesh communication ensures that help can reach the user even without mobile signal. This project is motivated by the vision of using technology for social good, empowering women with a reliable and intelligent safety companion that provides protection, confidence, and peace of mind at all times.

1.5 OBJECTIVE

- To design a system that can trigger alerts even without mobile signal using local communication like **mesh network**.
- To minimize human effort by enabling hands-free and automatic SOS activation during emergencies.
- To ensure fast and reliable alert transmission to nearby contacts or authorities in real time.

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW

Recent research and product efforts combine on-device machine learning, and ad-hoc/mesh networking to provide emergency communication and automatic distress detection. These approaches aim to reduce reliance on cellular networks and human intervention by enabling local, fast, and private alerting mechanisms.

Their CNN achieved an impressive accuracy of approximately 98%. This work underscores the potential of task-specific deep learning models, illustrating that a welldesigned and properly regularized CNN can outperform more complex models when applied to a focused medical imaging problem.

2.2 LOCAL NETWORK FOR ALERTS

To operate without cellular coverage, many works explore local networking options such as Bluetooth Low Energy (BLE) mesh, mesh networking specifications, ad-hoc BLE forwarding, and other mesh/peer-to-peer approaches. Mesh provides a standardized, decentralized publish/subscribe messaging approach and built-in security, making it attractive for multi-hop alert propagation across nearby devices when a central network is unavailable. Studies and projects (e.g., MeshSOS, mobile BLE mesh evaluations) show BLE mesh can propagate emergency alerts between devices but also note implementation and scalability challenges in dynamic environments.

The **local network alert system** plays a crucial role in ensuring emergency communication even when mobile signals or internet connectivity are unavailable. Instead of

depending on cellular networks, the system uses technologies such as **mesh networking** to transmit distress alerts to nearby trusted contacts or devices within range. When the AI detects distress or the user triggers the SOS command, the wearable immediately forms a local communication link with nearby devices, allowing the alert message—containing location, time, and type of emergency—to be delivered instantly. This decentralized communication method ensures that help can reach the user even in remote areas, underground spaces, crowded public locations, or situations where networks are jammed. By creating a self-sustaining, peer-to-peer safety network, the local alert mechanism significantly enhances reliability and ensures that emergency notifications are never interrupted due to connectivity failures.

2.3 AUDIO BASED DISTRESS DETECTION

Audio analysis is an active research area for emergency detection. Techniques include voice activity detection (VAD), keyword spotting (wake-word and distress-word detection), acoustic event classification (e.g., scream, shouting, glass break), and emotion / stress classification from speech features (spectrograms, MFCCs). Deep learning models (CNNs, RNNs, and transformer variants) trained on labeled audio corpora have been shown to detect distress sounds and emergency events in real time, supporting automatic alerting. Recent papers specifically address real-time distress/scream detection and voice-based emergency classifiers.

Audio distress detection uses AI to automatically identify signs of fear, panic, or emergency in a woman's voice. The system continuously listens for specific acoustic patterns such as sudden pitch elevation, trembling tones, screaming, crying, or keywords like "help" or "save me." Using machine learning and deep learning—especially models trained on real human emotional speech—the system analyzes voice features such as frequency, intensity, and speech rate to detect abnormal stress signals. When the algorithm recognizes distress, it

immediately triggers an alert to the local safety network, emergency contacts, or the connected wearable device. This automated audio-based detection works even when the user cannot reach or unlock the phone, providing hands-free, real-time protection in dangerous situations.

2.4 SOS AND EMERGENCY BUTTON FEATURES

An SOS (Save Our Souls) button is a core feature in many women safety systems. It allows the user to trigger an immediate alert to nearby people or authorities with a single press. Some advanced systems integrate voice-activated SOS triggers, gesture recognition, or automatic activation based on abnormal activity patterns. In wearable devices, this button can be placed on smart bands, pendants.

The AI Women Safety Companion enhances this feature by combining manual SOS buttons with AI-driven automatic activation. Even when the user cannot press the button or has no mobile signal, the system detects distress and sends alerts through a local communication network (Bluetooth or mesh). The SOS and emergency button features provide an instant and reliable way for users to request help during dangerous or uncomfortable situations. By pressing a dedicated physical button on a wearable device or a virtual SOS button in the mobile app, the user can trigger an immediate distress alert without navigating through menus. This feature is designed for quick activation—often requiring just a long press or double tap—to ensure it works even under panic or limited mobility.

Once activated, the system automatically shares the user's real-time location, audio recording, and status with trusted contacts or the local alert network. Some implementations also activate loud alarms, vibration feedback, or automated calls to emergency numbers. These

features ensure rapid response and increase the chance of preventing harm by providing a simple, fast, and discreet method of seeking help.

2.5 SUMMARY

There is still a research gap in creating offline, **AI-powered devices** that can function without human intervention. The AI Women Safety Companion addresses these limitations by integrating Python-based AI models, voice and motion detection, SOS emergency features, and local network communication to ensure women's safety in all situations.

The SOS and emergency button features provide a fast and reliable way for users to request help during emergencies. With just a single press or long hold, the system instantly activates an alert and sends the user's live location, audio evidence, and status to trusted contacts or nearby responders. This quick-access mechanism is designed to work even in high-stress situations, offering discreet activation and immediate feedback through vibration or sound. By ensuring rapid communication and automatic data sharing, the SOS feature greatly improves user safety and enables faster response during critical moments.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the existing women safety systems, most solutions are mobile-based applications that depend heavily on internet connectivity and GPS services to send alerts during emergencies. When a woman feels unsafe, she must manually open the app and press an SOS button to share her location with friends, family, or police. These applications often use SMS or online servers information.

Some systems also include wearable panic buttons or GPS trackers, but they are limited in functionality and often require an active mobile network to operate. The detection of distress or emergency situations in these systems is mostly manual rather than automatic, relying entirely on user interaction. As a result, these existing methods are not always reliable in low-signal or offline environments, and in some situations, the victim may not even get a chance to trigger the alert manually.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- **Requires Strong Mobile Signal** – Most existing women safety apps depend heavily on mobile networks or internet connectivity, making them unreliable in rural or remote areas.
- **Manual Activation Needed** – Users must physically open the app or press an SOS button, which may not be possible during panic situations or when the attacker restricts movement.

- **Slow Emergency Response** – Due to network delays and server dependency, alerts often reach contacts late, reducing the effectiveness of the system.
- **No Continuous Monitoring** – Traditional safety apps only respond when activated, offering no real-time tracking or behavioral analysis to detect early signs of danger.
- **Limited Offline Functionality** – Without internet or GPS, most current safety apps become completely unusable, leaving the user without support.
- **Lack of Automation** – Existing systems rely on the user's actions, which is risky during high-stress or life-threatening scenarios.
- **Low Accuracy in Alerts** – Alerts may not always reach the intended contacts due to dependency on cloud servers, unstable networks, or app crashes.
- **Not Suitable for Instant Situations** – Many apps take time to open and send alerts, making them ineffective during sudden emergencies.

3.2 PROPOSED SYSTEM

The proposed AI Women Safety Companion introduces an AI-driven system designed to detect distress situations automatically and trigger alerts to a local communication network.This system uses Python-based Artificial Intelligence algorithms to analyze voice tone, motion patterns, and environmental cues for identifying abnormal or threatening situations. Once the system detects distress, it activates an SOS alert through Bluetooth or mesh network, sending the user's location and distress signal to nearby devices or authorities.

The device can also be equipped with a manual SOS button and voice activation feature to ensure redundancy. By integrating machine learning models, sensor data processing, and offline communication, this system ensures faster response, real-time protection, and high reliability. The wearable nature of the device provides continuous monitoring and hands-free operation, making it suitable for real-world use even in remote areas.

3.2.1. ADVANTAGES OF PROPOSED SYSTEM

- **Works Without Mobile Signal** – The system can trigger alerts using offline communication methods like Bluetooth and mesh networks, ensuring safety even in remote or low-network areas.
- **AI-Based Distress Detection** – Automatically identifies distress using voice tone analysis, emotion detection, and abnormal movement patterns, reducing dependency on manual SOS triggers.
- **Hands-Free Voice Activation** – Users can activate emergency mode with simple voice commands, making it highly effective during situations where physical access is restricted.
- **Faster Emergency Response** – The system sends real-time alerts to nearby trusted contacts, allowing quicker assistance compared to traditional apps that depend on internet connectivity.
- **Secure and Private** – Alerts are shared only with authorized local network members, ensuring privacy and preventing misuse of personal data.
- **Portable and Convenient** – As a wearable device, it is discreet, easy to carry, and practical for daily use.
- **Real-Time Local Communication** – Nearby devices form a safety network that helps in faster transmission of distress signals without relying on centralized servers.

CHAPTER 4
SYSTEM SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS

- Processor : Intel Core i5
- RAM : 8-16 GB
- Storage : 256 GB SSD
- Keyboard : Standard keyboard
- Monitor : 15-inch color monitor

4.2 SOFTWARE REQUIREMENTS

- Operating systems : Windows 10/11,Ubuntu 20.04+
- Programming language : Python 3.13
- Python libraries : Numpy,Pandas,JSON,SQLAlchemy, Tensorflow
- Development tools : VS Code/Jupyter Notebook
- Web based interface : Flask, HTML,CSS,JavaScript

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 LIST OF MODULES

1. User Input and Sensor Data Module
2. AI-Based Distress Detection Module
3. SOS Alert Activation Module
4. Local Communication and Alert Transmission Module
5. Notification and Feedback Module
6. Data Logging and Monitoring Module
7. User Registration and Setup Module
8. Emergency Contact Management Module

5.2 MODULE DESCRIPTION

5.2.1 User Input and Sensor Data Module

The User Input and Sensor Data Module is responsible for continuously collecting information from the sensors attached to the wearable device. It captures voice, body movements, and environmental sounds through components like microphones, accelerometers, and gyroscopes. The raw data is filtered to remove background noise and irrelevant signals to ensure accurate detection. This module serves as the foundation of the system, providing essential real-time inputs to the AI model. Using Python-based libraries, it manages sensor

interfacing and data transmission efficiently. The continuous monitoring ensures that any unusual activity or distress signal is captured immediately. This reliable input mechanism allows the system to respond quickly in emergency situations.

5.2.2 AI-Based Distress Detection Module

The AI-Based Distress Detection Module forms the core intelligence of the system. It uses Python-based machine learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to identify distress situations. The module analyzes the user's speech tone, screaming patterns, and sudden motion changes to detect abnormalities. The data collected from the sensors is classified as either normal or distress using pre-trained models. This enables the system to make automatic, real-time decisions without human intervention. The use of artificial intelligence ensures high accuracy and faster detection than manual methods. Over time, the model can be improved by retraining it with new datasets for better results. This module ensures that every possible sign of danger is recognized instantly and reliably.

5.2.3 SOS Alert Activation Module

The SOS Alert Activation Module is responsible for initiating the alert mechanism once distress is detected by the AI model. It can also be triggered manually through a panic button on the wearable device. The module functions even in the absence of internet or mobile signal by using alternative communication channels. Once activated, it generates an SOS message that contains essential details such as the user's identity and location information if available. The alert is then sent to nearby devices or emergency contacts through local communication modules. The design ensures minimal delay between detection and alert activation, providing instant response during emergencies. This automation reduces the need for manual input,

especially in critical or panic situations. The module guarantees a fast and dependable safety alert process.

5.2.4 Local Communication and Alert Transmission Module

The Local Communication and Alert Transmission Module ensures that the emergency alerts reach nearby users or authorities without depending on mobile networks. It uses Bluetooth or mesh networking technologies to build a short-range local network. This allows the system to communicate distress messages even in remote areas where cellular connectivity is weak. The module identifies nearby trusted devices and transmits the SOS signal securely. It can also connect with local servers for storing and managing alert information. Python libraries such as bleak or socket are used to handle the communication protocols efficiently. This module ensures that no alert goes unnoticed and that nearby people can respond immediately. It provides an essential offline communication layer that enhances the system's reliability and robustness.

5.2.5 Notification and Feedback Module

The Notification and Feedback Module plays a crucial role in confirming to the user that the SOS alert has been successfully activated. Once the system detects distress or receives manual input, this module provides feedback using vibration, LED lights, or small buzzers. The purpose is to reassure the user that the signal has been sent and help is on the way. It also receives acknowledgment messages from nearby devices when they receive the alert. Python code controls the output devices through GPIO pins connected to LEDs or vibration motors. This module ensures that the user stays informed about the alert status in real time. It improves the usability and trustworthiness of the system, especially during stressful moments.

5.2.6 Data Logging and Monitoring Module

The Data Logging and Monitoring Module keeps a record of all system events and sensor readings for future analysis. It logs data such as timestamps, detected motions, voice analysis results, and alert activations. This information is vital for testing, improving accuracy, and detecting false positives or missed alerts. The data is securely stored in local databases or simple files such as CSV or SQLite. Python's logging and file-handling capabilities are used for efficient data management. This module supports long-term improvement and ensures reliability through consistent monitoring.

5.2.7 User Registration and Setup Module

The User Registration and Setup Module manages the initial setup process of the device. It allows the user to register their personal details and link their wearable to their profile. This includes entering name, ID, and configuration settings needed for device operation. The module ensures that each user has a unique identification within the system. It also allows updates to user information when required. Python scripts handle the user interface and validation of inputs during registration.

5.2.8 Emergency Contact Management Module

The Emergency Contact Management Module is a vital component of the women's safety system that handles the storage, retrieval, and management of the user's trusted emergency contacts. It ensures that during distress or emergency situations, the system can instantly notify these contacts with accurate and real-time information such as the user's location, alert message, and timestamp.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

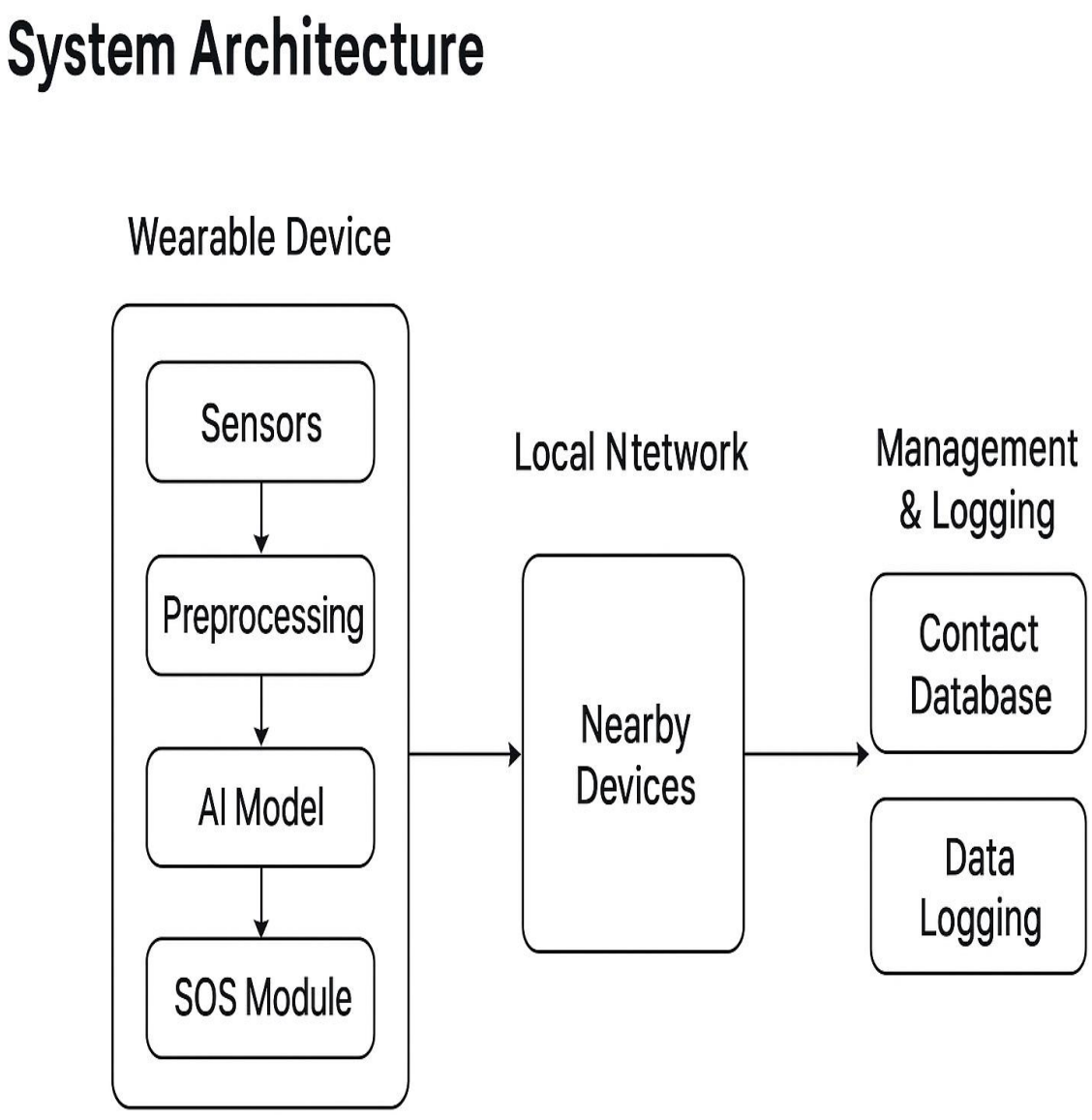

Figure 6.1 System Architecture

6.2 USE CASE DIAGRAM

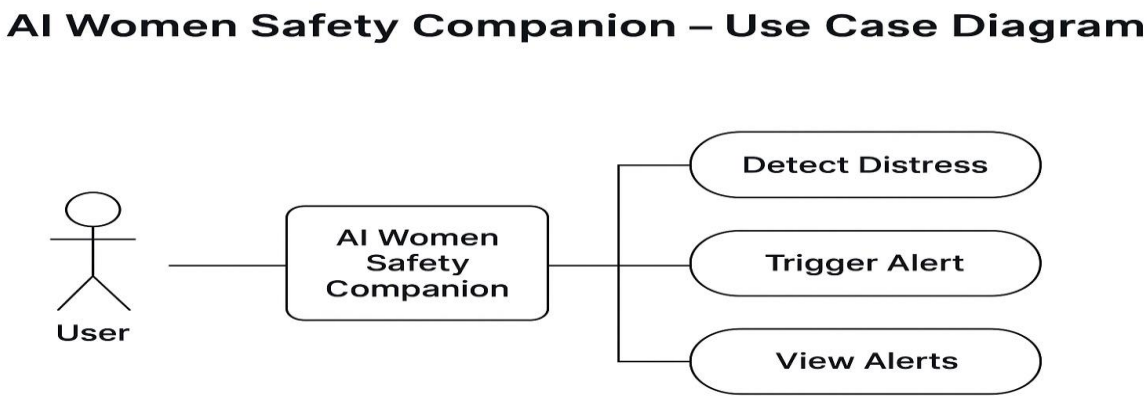

Figure 6.2 Use Case Diagram

6.3 CLASS DIAGRAM

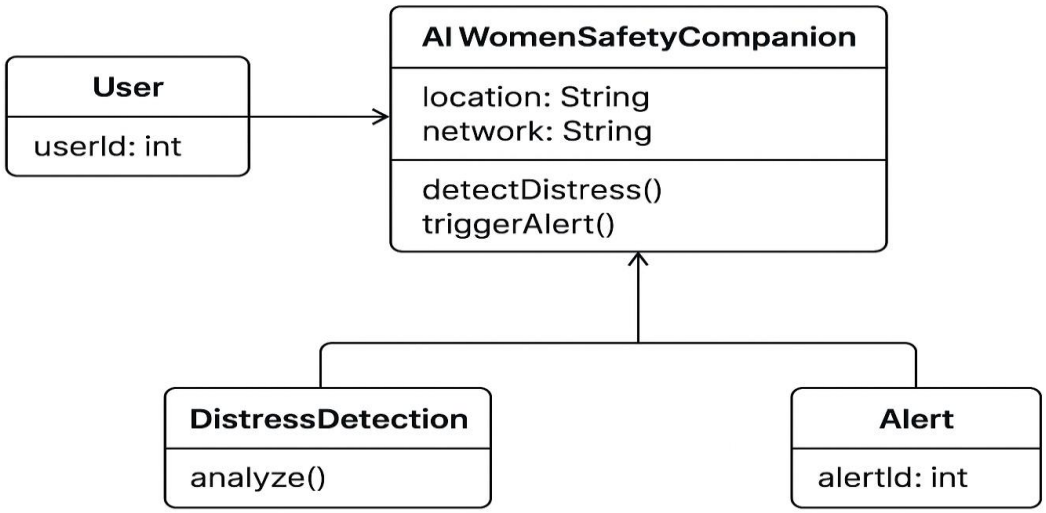

Figure 6.3 Class Diagram

6.4 SEQUENCE DIAGRAM

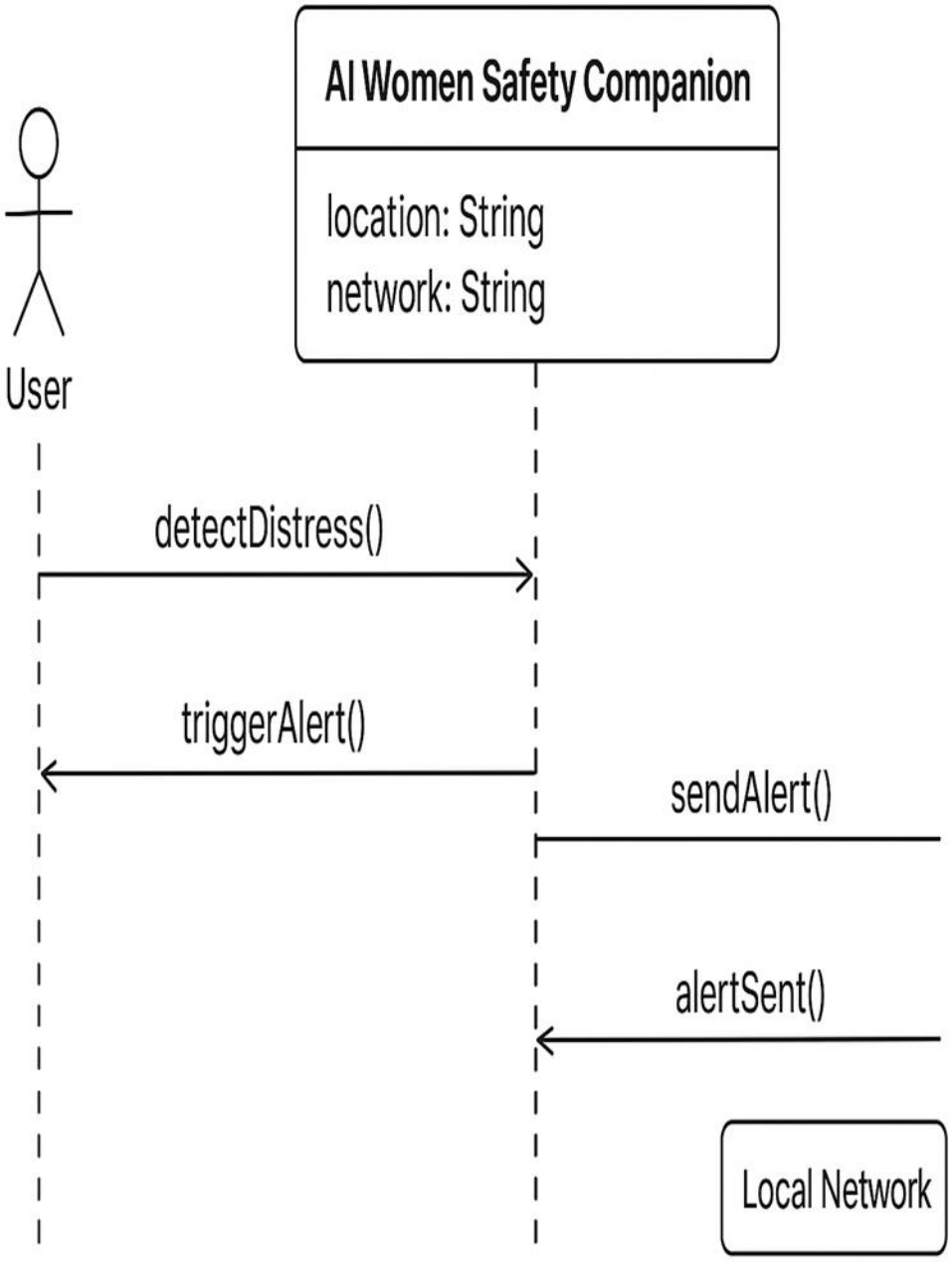

Figure 6.4 Sequence Diagram

6.5 ACTIVITY DIAGRAM

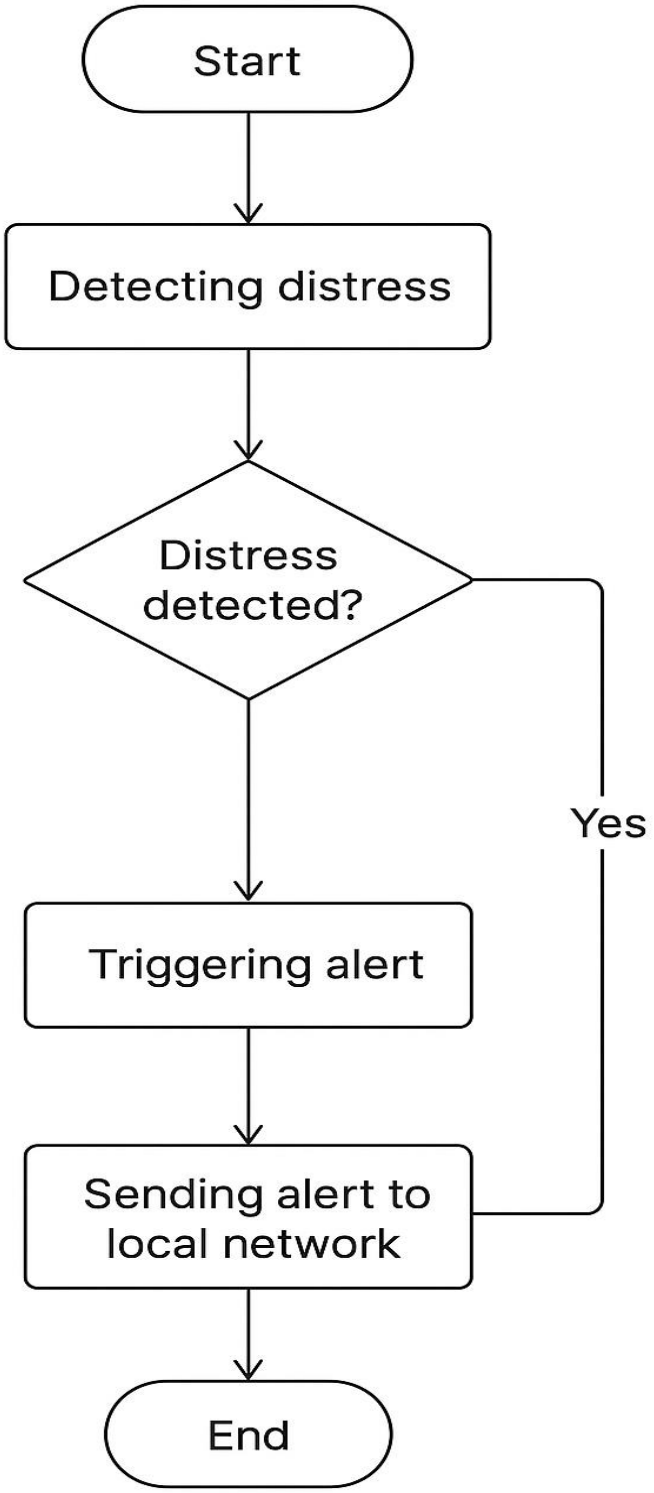

Figure 6.5 Activity Diagram

CHAPTER 7

SOFTWARE DESCRIPTION

7.1 OBJECTIVES

The main objective of the AI Women Safety Companion project is to design and implement a voice-enabled, AI-powered system that can automatically detect distress situations and send emergency alerts to nearby networks, even in the absence of a mobile signal. In today's world, especially in a country like India, the safety of women is a pressing concern, and the project aims to provide a technological solution that is both reliable and accessible.

7.2 KEY FEATURES

7.2.1 AI-Based Distress Detection

- To implement advanced **Artificial Intelligence (AI)** algorithms that can detect distress situations based on voice tone, speech emotion, and motion sensor data.
- The system continuously monitors user patterns and identifies panic or fear through abnormal movements or voice fluctuations.

7.2.2 Offline Alert Mechanism

- To ensure communication and alert triggering even without mobile signal or internet connectivity.
- The system uses **mesh network technology** to send distress alerts to nearby trusted devices or contacts.

7.3 SOFTWARE COMPONENTS

7.3.1. INPUT LAYER

The input layer receives raw data from the wearable sensors such as accelerometers, gyroscopes, microphones, and heart-rate sensors. These inputs represent continuous streams of real-world signals like movement speed, acceleration patterns, voice pitch, and environmental sound. The input data is then preprocessed using Python libraries like **Numpy** for normalization, noise filtering, and feature scaling. This step ensures that the data fed into the AI model is clean, consistent, and ready for accurate analysis.

7.3.2.SYSTEM ARCHITECTURE

Input Layer

- The layer also receives manual inputs, such as user-triggered SOS button presses or emergency gestures.
- Audio signals are preprocessed to remove background noise, enhance clarity, and normalize volume levels.

Preprocessing Layer

- Filters out unnecessary background noise or irregular sensor readings.

AI Distress Detection Layer

- Analyzes preprocessed data to detect distress patterns in motion and voice signals.
- Trained using labeled datasets of normal vs. emergency behaviors.

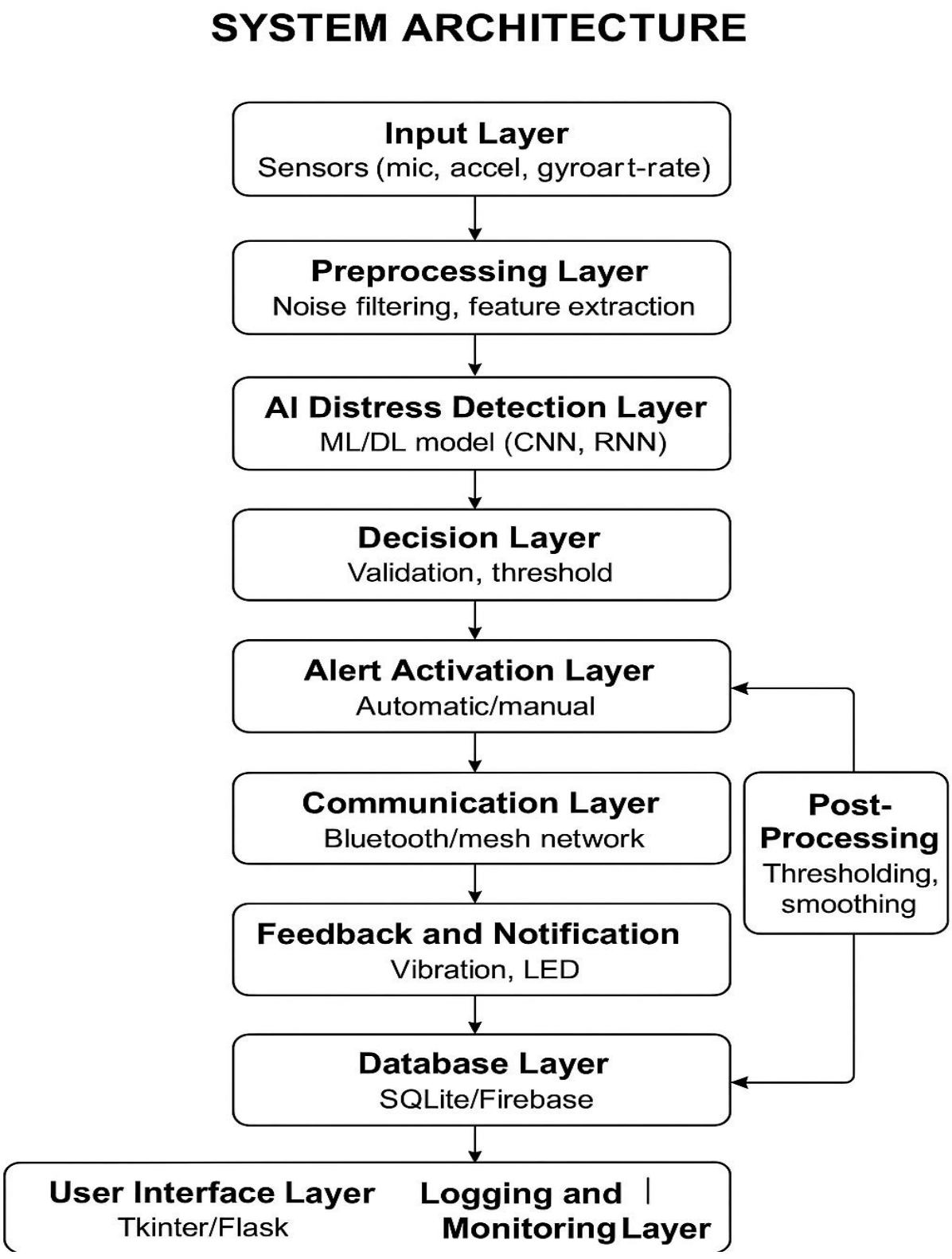

Figure 7.3.2 System Architecture

Communication Layer

- The Communication Layer is responsible for **transmitting alerts and safety signals** once distress is detected by the system.
- It enables communication **without mobile network**, using technologies like Bluetooth, Wi-Fi Direct, or local mesh networks.

Feedback and Notification Layer

- Provides real-time feedback to users after an alert is sent.
- Triggers vibration, LED blink, or audio beep for confirmation.
- Improves user trust by ensuring alerts have been successfully transmitted.

Database Layer

- Stores user profiles, emergency contacts, and alert logs.
- Maintains local database using **SQLite** or **Firebase**.
- Ensures data persistence and retrieval during offline operations.
- Supports tracking and analysis of system performance over time.

User Interface Layer

- Provides an easy-to-use platform for users to register and configure device.
- Displays status of alerts, logs, and device connectivity.

Logging and Monitoring Layer

- The Logging and Monitoring Layer records all **system activities**, including detected events, voice inputs, alerts, and user interactions.
- It maintains a **secure log database** that stores timestamps, system responses, processing results, and communication attempts.
- This layer continuously monitors the performance of the AI model, detecting issues like **false triggers, missed alerts, or system failures**.

7.3.3. TRAINING AND VALIDATION

Training Phase

- The Training Phase involves feeding the AI model with a large set of labeled audio samples, including normal speech, distress keywords, screams, and environmental noises.
- During this phase, the system learns to differentiate between safe and distress scenarios through supervised machine learning techniques.
- Training is performed using frameworks like **TensorFlow or PyTorch**, which help in building deep learning architectures for emotion and distress detection.
- The dataset is split into training, validation, and testing sets to avoid overfitting and ensure that the model generalizes well to new inputs.

7.3.4. SEGMENTATION

Data Stream Division

- The continuous input data from sensors (microphone, accelerometer, gyroscope) is divided into small time frames or segments.
- Each segment may represent a few seconds of data.

Audio Segmentation

- For voice-based distress detection, the system captures continuous speech input.
- Each segment may represent a few seconds of data

7.4 WORKFLOW

Sensor Data Collection:

- Sensors continuously capture voice, motion, and environment data.

Data Preprocessing:

- Noise is filtered and data is normalized.

AI Analysis:

- The neural network analyzes input data and detects distress patterns.

Validation and Decision:

- Model outputs are validated through thresholding to confirm distress.

Alert Activation:

- SOS alert is generated automatically or manually by the user.

Local Communication:

- The alert is broadcasted via Bluetooth/Mesh to nearby trusted contacts.

Feedback and Logging:

- The user receives confirmation through vibration or LED signals, and event data is stored locally.

7.5 TECHNOLOGY USED

Frameworks :

TensorFlow

Used for designing and training deep learning models for distress detection based on sensor and voice data.

Flask

Flask is used for lightweight web-based user interface deployment, while Flask is used for basic desktop interface configuration and alert management.

Libraries:

NumPy

For handling large numerical data arrays, signal computation, and sensor data processing.

SQL/Firebase

Manages data storage for user profiles, contact lists, and distress event logs securely and efficiently.

Dataset :

The dataset consists of audio samples and motion sensor readings collected from various real-life and simulated distress scenarios.

CHAPTER 8

SOFTWARE TESTING

8.1 AIM OF TESTING

The main aim of testing the AI Women Safety Companion system is to ensure that all components of the application function correctly, reliably, and efficiently under various conditions. Testing aims to verify that the system accurately detects distress situations, triggers SOS alerts automatically or manually, and successfully transmits alerts to nearby devices even without mobile or internet connectivity.It also ensures that the AI model performs with high accuracy and minimal false detections, while the communication and feedback modules respond in real time. Through systematic testing, the goal is to identify and correct any functional, logical, or performance issues before deployment, ensuring that the final system is robust, secure, and user-friendly for real-world use.

8.2 TEST CASE

The testing of the AI Women Safety Companion system was carried out through multiple well-defined test cases to ensure the proper functioning of all modules, including AI-based distress detection, SOS alert activation, Bluetooth communication, and the feedback mechanism. Each module was tested individually and then integrated to verify the overall system performance. In the voice detection module, tests were conducted using both panic tones and normal speech inputs. When a distress voice or panic sound was detected, the system successfully triggered the SOS alert automatically, whereas normal speech did not activate any alert, confirming high accuracy and minimal false alarms.

8.3 TYPES OF TESTING

Several types of software testing are applied throughout the development and validation. These include:

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Performance Testing
- Validation Testing
- Regression Testing
- Usability Testing

8.4 DESCRIPTION OF TESTING

8.4.1 UNIT TESTING

Unit testing is the first level of testing performed during the development of the AI Women Safety Companion system. In this phase, each individual module — such as the AI distress detection, SOS activation, communication, and feedback modules — is tested independently to ensure that it performs its intended function correctly. In the AI Women Safety Companion, unit testing ensures that each module — such as voice input processing, distress detection, alert triggering, and network communication — works correctly and independently.

8.4.2 INTEGRATION TESTING

Integration testing is conducted to ensure that all modules work together properly. It verifies the smooth interaction between the AI model, SOS alert system, Bluetooth communication, and feedback mechanisms.The main goal is to identify interface issues, communication mismatches, or logical errors that may occur when modules are interconnected.

8.4.3 FUNCTIONAL TESTING

Functional testing focuses on validating that the system meets all its functional and user requirements. It ensures that the major features — distress detection, SOS button activation, offline alert transmission, and user feedback — operate as expected under real-world conditions.

8.4.4 SYSTEM TESTING

System testing is a comprehensive test performed on the complete, integrated system to verify that all hardware and software components work together harmoniously. It examines the overall behavior of the AI Women Safety Companion, including the input sensors, processing algorithms, and output responses such as alerts or notifications.

8.4.5 PERFORMANCE TESTING:

Performance testing measures how efficiently and quickly the system performs under various conditions. It evaluates parameters such as system response time, processing speed, and stability during long periods of operation. For the AI Women Safety Companion, performance testing ensures that distress signals are detected instantly and that SOS alerts are sent within 1–2 seconds, even when multiple modules are running simultaneously.

8.4.6 VALIDATION TESTING

Validation testing ensures that the final system satisfies all project goals and performs as expected in real-time scenarios. It checks the accuracy of the AI distress detection model, the reliability of the alert mechanism, and the proper functioning of the local communication network. This testing validates that the system meets both the **functional** and **non-functional** requirements defined during the design phase.

8.4.7 REGRESSION TESTING

Regression testing is a type of software testing performed after modifications, updates, or enhancements are made to the system to ensure that the new changes do not negatively affect the existing functionalities. In the AI Women Safety Companion project, regression testing is used to verify that improvements made to the AI distress detection model, SOS alert mechanism, or communication modules do not introduce new bugs or break previously working features.

8.4.8 USABILITY TESTING

Usability testing is a type of software testing that evaluates how easy, efficient, and user-friendly a system is for real users. The main goal is to ensure that users can interact with the application comfortably and effectively without confusion or difficulty.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

The AI Women Safety Companion – Voice-Enabled Wearable project successfully demonstrates how artificial intelligence, sensor integration, and offline communication technologies can be combined to create an intelligent and autonomous safety solution for women. In today's world, personal safety remains a growing concern, particularly in situations where mobile networks are unreliable or when a victim is unable to physically operate a device. Traditional mobile-based safety applications fail in such scenarios because they depend heavily on manual input and stable connectivity. This project addresses these limitations by offering a hands-free, voice-triggered, AI-powered system that continues to function even when external systems fail.

The system incorporates a robust software architecture built using Python, where machine learning models analyze audio patterns, distress keywords, and environmental sounds to identify emergencies. The processing pipeline—from input layer to alert generation—ensures that distress signals are detected in real-time without user intervention. The Communication Layer plays a significant role in bridging the gap between victim and responders by using Bluetooth, Wi-Fi Direct, and mesh networking for sending emergency alerts when mobile networks are unavailable. This ability to communicate locally makes the solution highly dependable in remote, underground, or crowded environments.

The Logging and Monitoring Layer ensures consistent system behavior and enables thorough tracking of all events, improving transparency and reliability. The project also

demonstrates the strength of modular design, allowing different components (input layer, processing layer, communication layer, storage layer) to operate independently while contributing to the overall workflow. This modularity ensures that the project is easily extendable and ready for future upgrades.

In addition to technological capability, the system reflects a strong social purpose. India and many countries worldwide face increasing concerns about women's safety. By enabling a wearable device that listens, learns, and responds instantly, this project aims to empower women and provide a sense of security in everyday life. The project not only serves as a technological solution but also as a safety initiative that can potentially save lives.

Overall, the project accomplishes its objective of creating a smart, practical, and innovative safety companion. It stands as a valuable prototype that can be translated into a real-world product with further improvements. The success of the project showcases how AI and embedded systems can work together to create solutions that make society safer and more supportive.

9.2 FUTURE ENHANCEMENT

The AI Women Safety Companion has strong potential for future development, and several enhancements can be implemented to increase its functionality, accuracy, and scalability. One of the primary improvements involves integrating more advanced deep learning architectures, such as CNN-RNN hybrids, LSTM-based emotion recognizers, or Transformer models (Wav2Vec, Whisper-like architectures). These models can significantly improve detection accuracy and reduce false alarms in noisy environments such as streets, markets, or public transport.

Another major enhancement is incorporating additional biosensors, such as heart rate monitors, pulse oximeters, skin temperature sensors, or stress-level sensors. These inputs can help the system detect unusual physiological changes that often accompany fear or danger. The combination of biometric inputs with audio analysis would make the model far more intelligent and context-aware, enabling it to differentiate between normal and panic situations more accurately.

Future versions of the system may also incorporate edge computing chips or AI accelerators like Google Coral, Raspberry Pi Zero 2 W, or ESP32 AI modules. These specialized hardware components can process AI tasks faster and with lower power consumption, extending battery life and improving real-time monitoring capability.

In terms of communication, the device can be enhanced by adding LoRaWAN or satellite connectivity, enabling long-range communication in regions where even local networks are limited. Satellite-supported distress messaging would make the wearable effective in remote areas such as forests, highways, villages, and terrains with limited infrastructure.

On the software side, a **cloud-based platform** can be introduced to maintain an online backup of logs, provide advanced analytics, store user cases, and generate detailed incident reports. The cloud portal could also be used by family members or police authorities to monitor safety alerts in real time. AI model updates can also be deployed over the cloud as OTA (Over-The-Air) upgrades.

The system can further be expanded by launching a mobile companion application with advanced features such as real-time location tracking, live audio streaming, device health

monitoring, and alert customization. The app can also integrate with government emergency services (112 India), local police networks, or community volunteers, thereby creating an interconnected safety ecosystem.

Future enhancements may also include predictive safety alerts, where the AI monitors the user's daily routes and warns them if they are entering a high-risk area based on crime statistics. Integrating map intelligence and geofencing technology can make the device proactive rather than reactive.

Lastly, improvements in hardware design—such as making the wearable lighter**,** waterproof, shock-resistant, and ergonomically designed—will enhance usability and comfort. Longer battery life can be achieved through solar charging modules or ultra-low-power microcontrollers. Future versions can include voice authentication technology that identifies the user’s unique voice signature. This prevents unauthorized use of the device and ensures that emergency alerts are triggered only by the registered user.

APPENDIX 1

SOURCE CODE

Safety.Py

```
import json

import socket

import time

import queue

import threading

import sys

import os

try:

    from vosk import Model, KaldiRecognizer

    import sounddevice as sd

    import numpy as np

    VOSK_AVAILABLE = True

except Exception as e:

    print(f"⚠ Voice modules not available: {e}")
```

```
    VOSK_AVAILABLE = False

    MODEL_PATH = "model"  # path to vosk model directory

SAMPLE_RATE = 16000

CHANNELS = 1

KEYWORDS = {"help", "emergency", "stop", "save me", "rape", "abuse", "leave me alone",
"get out"}

MULTICAST_GROUP = "239.255.0.1"

MULTICAST_PORT = 5007

ALERT_TTL = 1

ALERT_COOLDOWN = 30

NODE_ID = socket.gethostname()

def detect_keywords(text):

    for kw in KEYWORDS:

        if kw in text:

            return True

    return False

def build_alert(text):
```

```
    return {
        "type": "distress_alert",
        "node_id": NODE_ID,
        "timestamp": int(time.time()),
        "detected_phrase": text,
        "language": "undetermined"
    }
def send_udp_multicast(message_bytes):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
    sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ALERT_TTL)
    try:
        sock.sendto(message_bytes, (MULTICAST_GROUP, MULTICAST_PORT))
        print(f"📡 Sent multicast to {MULTICAST_GROUP}:{MULTICAST_PORT}")
    except Exception as e:
        print("❌ Failed to send multicast:", e)
    finally:
        sock.close()
```

```
def play_local_alarm():
    try:
        print("\a")  # system bell
    except Exception:
        pass
def write_local_alert_file(payload):
    try:
        fn = f"distress_alert_{int(time.time())}.json"
        with open(fn, "w") as f:
            json.dump(payload, f)
        print("💾 Wrote local alert file:", fn)
    except Exception as e:
        print("❌ Failed to write alert file:", e)
def handle_alert(text):
    alert_payload = build_alert(text)
    print("🚨 Triggering local alert:", alert_payload)
```

```
send_udp_multicast(json.dumps(alert_payload).encode('utf-8'))
    play_local_alarm()
    write_local_alert_file(alert_payload)
def text_mode():
    print("\n💬 Voice not working or no speech detected.")
    user_text = input("👉 Please type your distress message: ").strip().lower()
    if user_text:
        print(f"📝 You entered: {user_text}")
        if detect_keywords(user_text):
            handle_alert(user_text)
        else:
            print("⚠️ No emergency keywords detected.")
    else:
        print("⚠️ Empty input, returning to idle mode.")
def start_listening():
    if not VOSK_AVAILABLE:
```

```
        print("⚠ Vosk or sounddevice not installed — switching to text mode.")
        text_mode()
        return
if not os.path.isdir(MODEL_PATH):
        print(f"❌ Model directory not found at {MODEL_PATH}. Switching to text mode.")
        text_mode()
        return
try:
        model = Model(MODEL_PATH)
        rec = KaldiRecognizer(model, SAMPLE_RATE)
        rec.SetWords(True)
        audio_q = queue.Queue()
def audio_callback(indata, frames, time_info, status):
            if status:
                print("⚠ Sounddevice status:", status, file=sys.stderr)
            audio_q.put(bytes(indata))
```

```
with sd.RawInputStream(
        samplerate=SAMPLE_RATE, blocksize=8000, dtype='int16',
        channels=CHANNELS, callback=audio_callback
    ):
        print("🎤 Listening for emergency words (say 'help', 'emergency', etc.)... Press Ctrl+C to stop.")
        last_alert
while True:
            try:
                data = audio_q.get(timeout=10)
            except queue.Empty:
                print("⌛ No voice detected for 10 seconds — switching to text mode.")
                text_mode()
                break
if rec.AcceptWaveform(data):
                res = json.loads(rec.Result())
                text = res.get("text", "").strip().lower()
```

```
                if text:
                    print(f"🗣 Recognized: {text}")
                    if detect_keywords(text) and (time.time() - last_alert) >
ALERT_COOLDOWN:
                        last_alert = time.time()
handle_alert(text)
    except Exception as e:
        print(f"⚠ Voice input failed: {e}")
        print("Switching to text mode...")
        text_mode()
if __name__ == "__main__":
    print("🔐 AI Women Safety Companion — Prototype")
    print("Keywords:", KEYWORDS)
try:
        start_listening()
    except KeyboardInterrupt:
        print("\n🔘 Stopped by user.")
```

```
        sys.exit(0)

    except Exception as e:

        print(f"⚠ Unexpected error: {e}")
```

index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

<meta charset="UTF-8" />

    <meta

      name="viewport"

      content="width=device-width, initial-scale=1.0"

    />

    <meta

      name="description"

      content="AI Women Safety Companion — Emergency Voice and Text Alert System"

    />

    <link rel="icon" type="image/png" href="/favicon.png" />
```

```
    <title>AI Women Safety Companion</title>

 <link

      href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&di
splay=swap"

      rel="stylesheet" />

<!-- Tailwind CSS -->

    <script src="https://cdn.tailwindcss.com"></script>

 <style>

      body {

        background: linear-gradient(

          135deg,

          #1e3a8a 0%,

          #e11d48 50%,

          #fbbf24 100%

        );

        min-height: 100vh;

        margin: 0;
```

```
    padding: 0;
    font-family: "Poppins", sans-serif;
    color: white;
  }
#root {
    animation: fadeIn 1.2s ease-in-out forwards;
    opacity: 0;
  }
@keyframes fadeIn {
    0% {
      opacity: 0;
      transform: translateY(10px);
    }
    100% {
      opacity: 1;
      transform: translateY(0);
    }
```

```
@media (max-width: 768px) {
        body {
          background: linear-gradient(
            180deg,
            #1e3a8a 0%,
            #e11d48 70%,
            #fbbf24 100%
          );
        }
      }
    </style>
  </head>
<body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

alert_server.py

```
from flask import Flask, request, jsonify

from flask_cors import CORS

import json

import time

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

import threading

NODE_ID = "SAFETY-NODE-001"

EMERGENCY_KEYWORDS = {"help", "emergency", "stop", "save me", "rape", "abuse",
"leave me alone", "get out"}

def build_alert(message):

    return {

        "device_id": NODE_ID,

        "message": message,

        "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")
```

```
def send_udp_multicast(payload):
    print("📡 Broadcasting alert via UDP multicast... (simulated)")
def write_local_alert_file(payload):
    with open("alerts_log.json", "a") as f:
        f.write(json.dumps(payload) + "\n")
    print("💾 Alert logged locally")
def play_local_alarm():
    def alarm():
        print("🚨 Local alarm triggered!")
        for _ in range(3):
            time.sleep(1)
            print("🔊 Beep!")
    threading.Thread(target=alarm).start()
def is_emergency(message):
    """Check if the message contains any emergency keywords"""
    message_lower = message.lower()
```

```
    return any(kw in message_lower for kw in EMERGENCY_KEYWORDS)

app = Flask(__name__)

CORS(app)

POLICE_EMAIL = "publicsafetycenter@gmail.com"   # recipient

SENDER_EMAIL = "youremail@gmail.com"            # your Gmail

SENDER_PASS = "your_app_password"               # Gmail app pass

def send_email_alert(user_name, location, detected_phrase):

    try:

        subject = f" Emergency Alert from {user_name}"

        body = f"""

URGENT EMERGENCY ALERT ⚠

Name: {user_name}

Location: {location}

Message: {detected_phrase}

Timestamp: {time.strftime("%Y-%m-%d %H:%M:%S")}

Device ID: {NODE_ID}
```

```
msg = MIMEMultipart()
        msg["From"] = SENDER_EMAIL
        msg["To"] = POLICE_EMAIL
        msg["Subject"] = subject
        msg.attach(MIMEText(body, "plain"))
with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            server.login(SENDER_EMAIL, SENDER_PASS)
            server.sendmail(SENDER_EMAIL, POLICE_EMAIL, msg.as_string())
print("☑ Emergency email sent successfully!")
        return True
    except Exception as e:
        print("❌ Failed to send email:", e)
        return False
@app.route('/')
def home():
    return " 🚨 Safety Companion Backend is running!"
```

```
@app.route("/api/text_alert", methods=["POST"])

def text_alert():

    try:

        data = request.json

        user_name = data.get("name")

        location = data.get("location")

        message = data.get("message")

if not all([user_name, location, message]):

            return jsonify({"status": "error", "message": "Missing required fields"}), 400

print(f"📩 Text Alert from {user_name} ({location}): {message}")

alert_payload = build_alert(message)

        send_udp_multicast(json.dumps(alert_payload).encode("utf-8"))

        write_local_alert_file(alert_payload)

        play_local_alarm()

# Only send email if message contains emergency keywords

        email_status = False

        if is_emergency(message):
```

```
            email_status = send_email_alert(user_name, location, message)
        else:
            print("⚠ Message does not contain emergency keywords — email not sent.")
    return jsonify({
            "status": "success",
            "message": "Emergency alert processed",
            "email_sent": email_status
        }), 200
    except Exception as e:
        print("Error:", e)
        return jsonify({"status": "error", "message": str(e)}), 500
@app.route("/api/audio_alert", methods=["POST"])
def audio_alert():
    try:
        data = request.json
        user_name = data.get("name")
        location = data.get("location")
```

```
        detected_phrase = data.get("detected_phrase", "Voice distress detected")

print(f"🎤 Audio Alert from {user_name} ({location}): {detected_phrase}")

alert_payload = build_alert(detected_phrase)

        send_udp_multicast(json.dumps(alert_payload).encode("utf-8"))

        write_local_alert_file(alert_payload)

        play_local_alarm()

# Only send email if message contains emergency keywords

        email_status = False

        if is_emergency(detected_phrase):

            email_status = send_email_alert(user_name, location, detected_phrase)

        else:

            print("⚠️ Message does not contain emergency keywords — email not sent.")

return jsonify({

    except Exception as e:

        print("Error:", e)

        return jsonify({"status": "error", "message": str(e)}), 50
```

APPENDIX 2

SCREENSHOTS

WEBPAGE INTERFACES

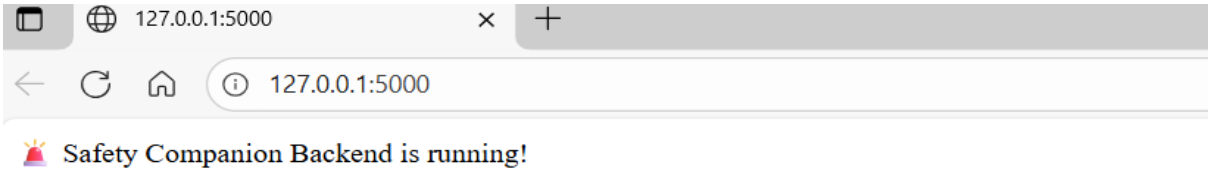

Figure 10.1 WebPage Interfaces

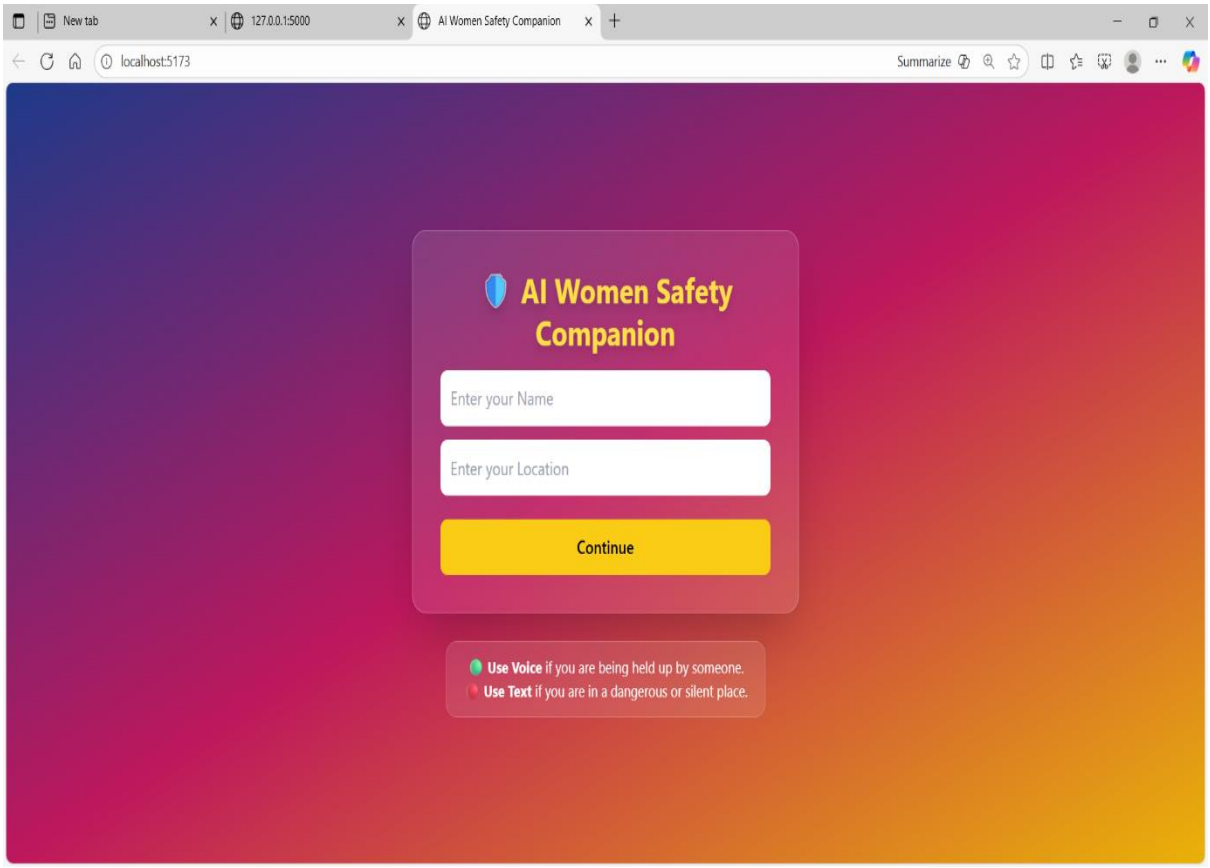

Figure 10.2 Login page

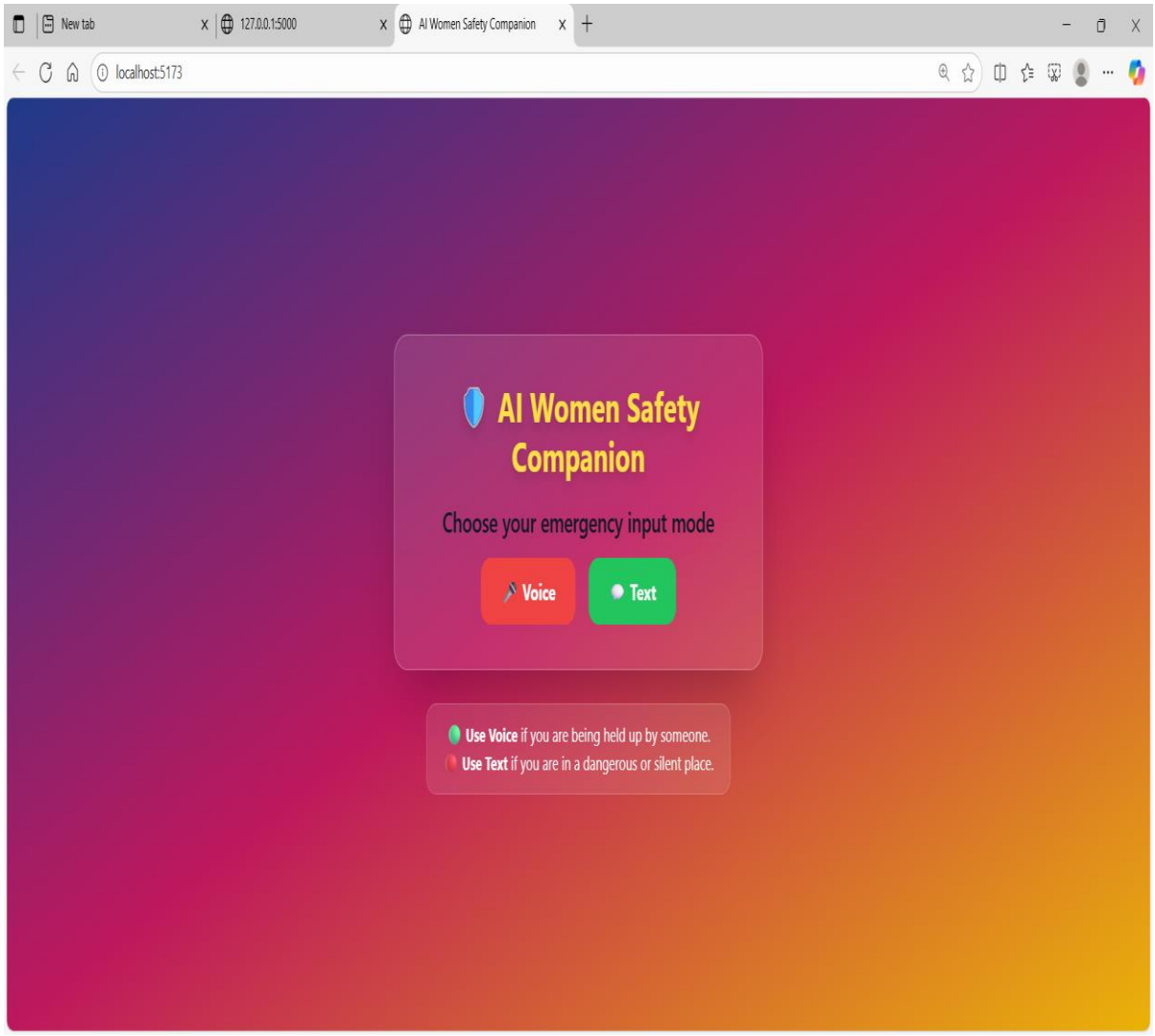

Figure 10.3 Input Mode

EMERGENCY INTERFACE

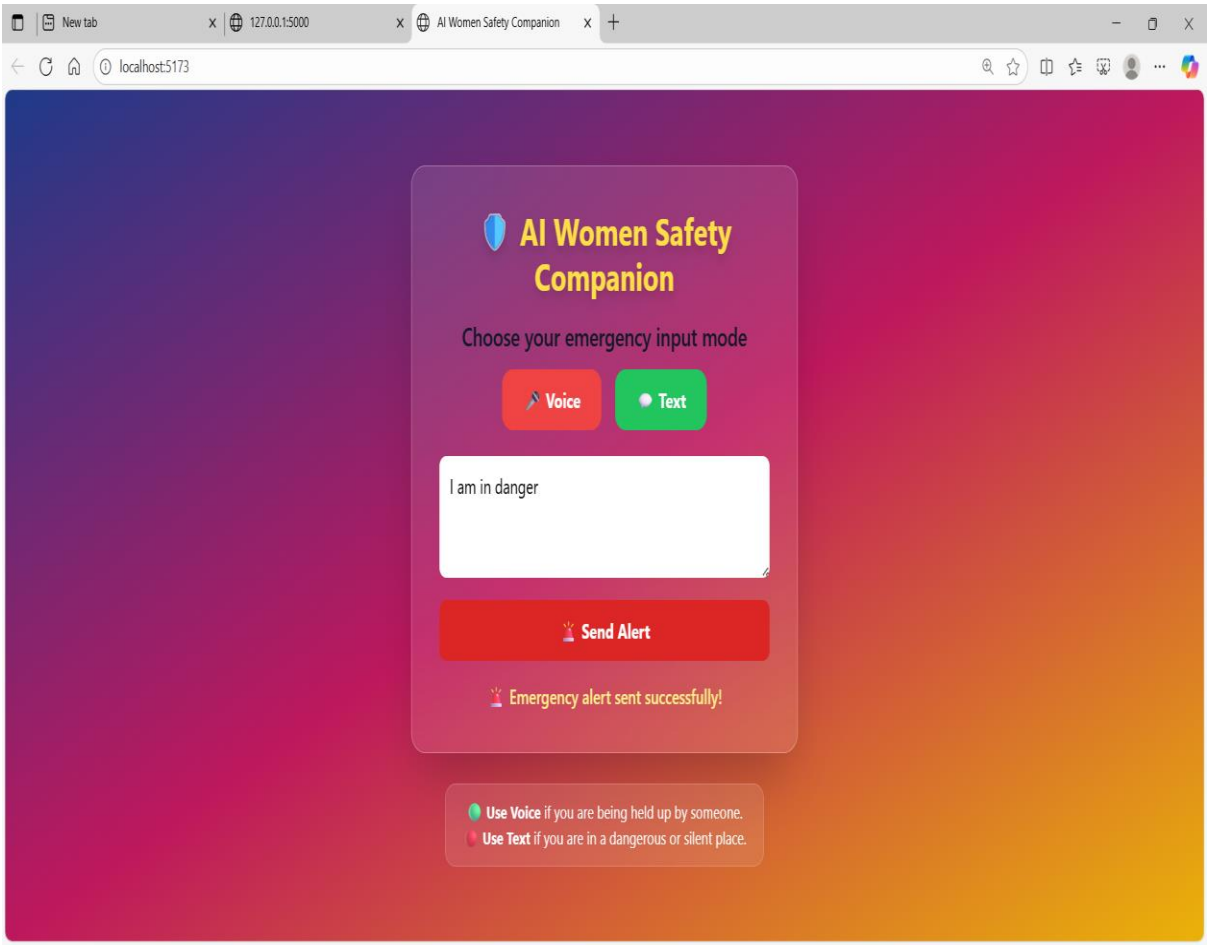

Figure 10.4 Emergency Alert Interface

SAFETY NOTIFICATION LOG

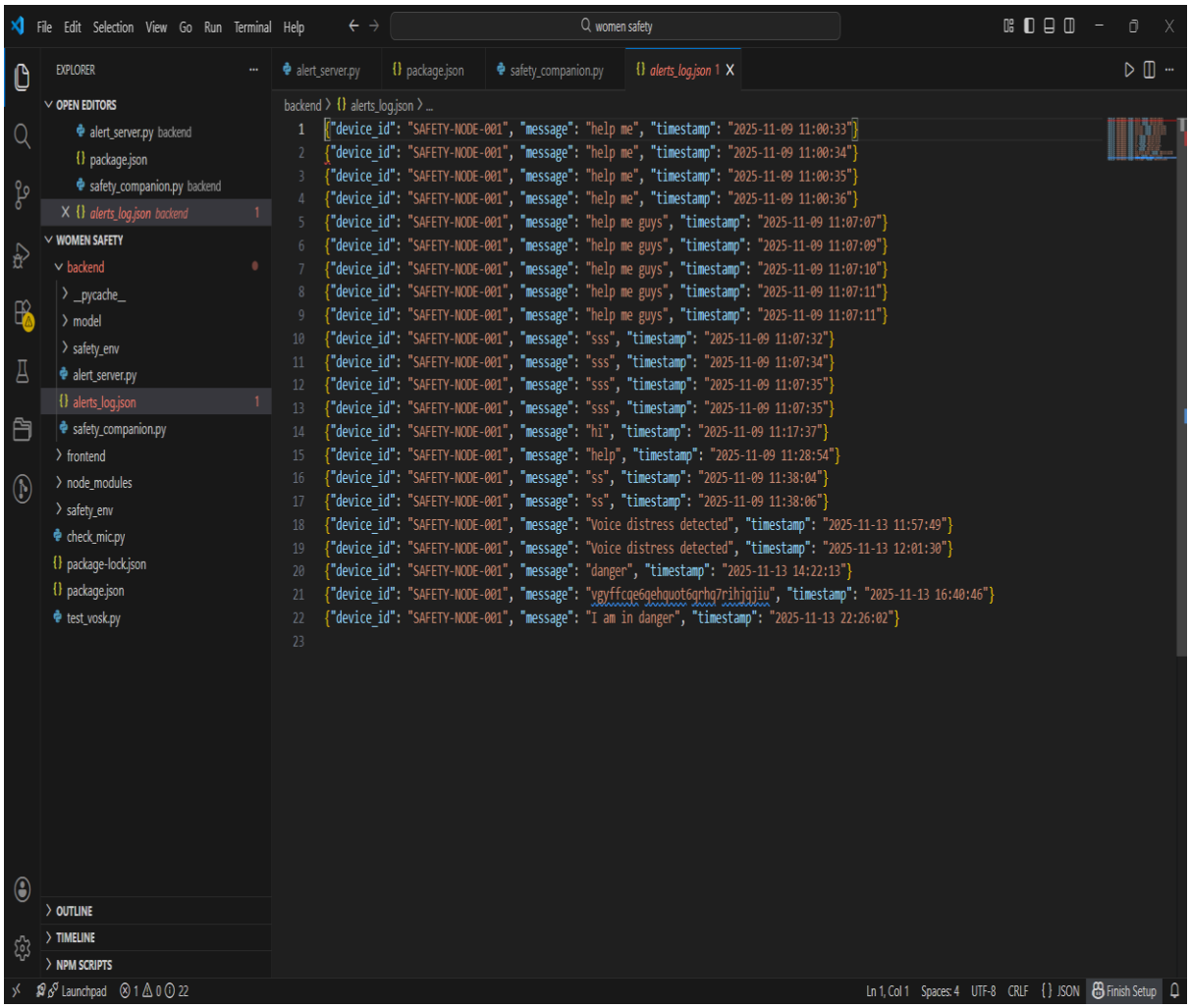

Figure 10.5 Safety Notification Log

REFERENCES

1. R. Kumar and S. Patel, *"*A Smart Women Safety System Using IoT and Machine Learning*,"* International Journal of Computer Applications, vol. 182, no. 45, pp. 15-21, 2023.

2. A. Sharma, P. Gupta, *"*AI-Driven Emergency Assistance Framework for Personal Safety*,"* Journal of Emerging Technologies, vol. 10, no. 2, pp. 112–120, 2022.

3. M. Chen and Y. Hao, "Wearable sensors for safety monitoring: A survey," *IEEE* Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2451–2475, 2021.

4. Google Developers, *"*TensorFlow: Machine Learning Framework*,"* 2024.

5. J. Brown, "Real-Time Audio Classification using Librosa and Deep Learning*,"* Journal of Audio Processing, vol. 14, no. 3, pp. 45-53, 2022.

6. N. Singh and A. Verma, "GPS-Based Tracking and Alert System for Personal Security," International Journal of IoT Applications, vol. 5, no. 1, pp. 29–36, 2021.

7. P. Raj and K. Vinay, *"*Design of SOS Alert System with Location Sharing," Proceedings of the 2022 International Conference on Smart Computing, pp. 210–215, 2022.

8. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

9. Librosa Documentation, Python Library for Audio and Speech Processing, 2024.

10. Flask Documentation, *"*Flask: Web Application Framework for Python," 2024.

11.Twilio Developer Docs, *"*SMS and Voice API for Emergency Alerts*,"* 2024.

12.U. Rustagi et al., "A Machine Learning–Based Safety Alert System Using Real-Time Sensor Data," Sensors, vol. 22, no. 7, pp. 1–14, 2022.