VELLORE INSTITUTE OF TECHNOLOGY, BHOPAL

ArmorBelle- A key to women's safety

Project By: DIVIJA SRIVASTAVA

Registration No.: 25BCY10104

Institute: VIT Bhopal

Date: 24 November 2025

📑 Table of Contents

ABSTRACT:-

An innovative application that protects and helps out the women in distress developed using pythons nltk and Tkinter framework. This project not only ensures the safety of women but also creates a safe and secure environment for the women to work providing them the sense of freedom and independence in this world. The application features initiating SOS for the women in distress while it locates the victim and the nearby help surrounding.
Working on DATA SCIENCE and NLP domain of AI, this app is an one stop solution for all the women out there.

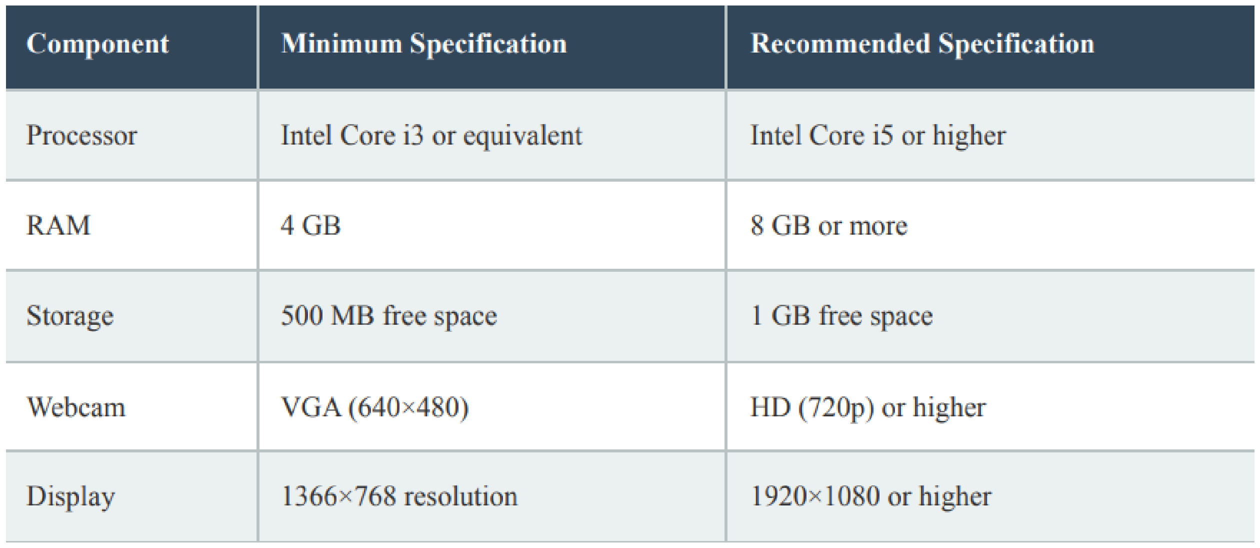
INTRODUCTION

Back in the days, women were not allowed to leave their homes and they were never provided with equal rights however as the time evolved we got to witness about how women fought for their rights and responsibilities and how they stand as competitive as men today. However we could sense that only time progressed, not the mindset of the people. We can witness that women still don’t feel safe to come out of their homes, they don’t get that sense of safety that they should. They are the victims of the most heinous crimes one would ever attest. From rape to acid attacks, they have to go through everything one would ever fear of and they have to bear this all silently. Have we ever given a thought on how do the women bear all of this silently? And all we do is give them advice on how to live in discipline and blame them even when they are not at fault? Did we ever ask them how they feel, ask them if they feel safe to travel alone or maybe just stand up for them when they need us? Where the world talks about how women are progressing in today's time, they don’t see how women's lives are in threat in various parts of the world. We use the word feminism very often, but it's very few of us who use it with a meaning.
Well as they say modern era brings modern solutions, Introducing ArmorBelle- an AI guard which is going to help out women in distress. This app is a one stop solution for all the women out there. This app will work on the Data Science and Natural Language Processing features of artificial intelligence. This app will consist of a SOS initiator which will let women feel safe wherever they go and if it feels anything suspicious in the chat, it will immediately inform the cyber cell. Additionally this app will keep a track of your location and if women feels unsafe in the area all they would have to do is press the SOS button which will give signals to the nearby police stations and keep them informed about their whereabouts until they press the “I’m feeling safe now” button on their screen (this button will be visible after the victim presses the SOS button). This action will also be saved in the safety history so that they have proof that they alerted them if no action is taken against the culprits.

OBJECTIVES

1. Instant emergency response
2. Automatic location detector
3. Clear Visual communication
4. Responsive user interface
5. Reliable Emergency workflow

System Requirements



Component	Minimum Specification	Recommended Specification
Processor	Intel Core i3 or equivalent	Intel Core i5 or higher
RAM	4 GB	8 GB or more
Storage	500 MB free space	1 GB free space
Webcam	VGA (640×480)	HD (720p) or higher
Display	1366×768 resolution	1920×1080 or higher

Software Requirements

Operating System: Windows 10/11, macOS 10.14+, or Linux (Ubuntu 20.04+)

Python Version: Python 3.8 or higher

Required Libraries: tkinter (GUI framework), Requests,Requests (HTTP requests), Threading, Math

Internet Connection: Required for loading weather data from Google Sheets

KEY FEATURES

1. One click Emergency activation
2. Automatic geolocation
3. Colour coded status indicators
4. Animated alert systems
5. Multithreaded Architecture
6. Emergency sequence workflow
7. Cancellation capabilities
8. Real time status logging
9. Offline Resilience

SYSTEM ARCHITECTURE

1. Overall Architecture Style

The application follows a **monolithic, event-driven desktop architecture** built on:

- **Tkinter GUI framework** (UI layer)
- **Threading module** for background execution (logic layer)
- **requests library** for external HTTP API calls (data input)
- A **single-class MVC-like structure** (SafetyDashboardApp) that acts as:
    - View Controller (UI management)
    - State handler
    - Business logic engine

2. Layered Architectural Breakdown

A. Presentation Layer (UI Layer)

Technologies: Tkinter (Frames, Buttons, Canvas, Label, LabelFrame, messagebox)

Main UI components:

1. Root Window

- Initializes the main application container

- Handles Tk event loop

2. Main Frame

- Wraps entire content inside a styled panel

3. Sub-Panels

a. Status Panel

- Shows system/SOS state
- Contains:
  - Status text label
  - Colored indicator circle (Canvas)
  - Pulsing effect for alert state

b. Location Data Panel

- Displays:
  - Public IP
  - City
  - Country & region
  - Log messages

c. SOS Button Panel

- Button toggles between:
  - *SOS Start*
  - *Cancel SOS*

B. Application/Logic Layer

1. State Machine System

State is stored in self.current_sos_state:

State	Meaning	UI Color
SAFE	Default operation	Teal
ALERT	SOS initiated	Red/Orange pulse
TIMEOUT	Awaiting response	Yellow

UI updates are always handled in the **main thread**.

2. SOS Sequence Engine (Background Thread)

Executed inside self._sos_sequence_logic():

Flow:

1. **Start SOS → 5 sec delay**
2. Check cancellation
3. Enter **TIMEOUT** state
4. **Wait 5 seconds**
5. Randomly simulate success or failure
6. Show messagebox response
7. Auto-reset system

This sequence runs in a **separate thread** to keep UI responsive.

Tkinter rules:

- Background thread **must NOT** update widgets directly
- Uses root.after(0, ...) to push updates to the main loop

3. Logging Engine

self.log_label displays real-time system messages.

C. Data Layer

1. External API Dependency

Your system calls:

http://ipinfo.io/json

This retrieves:

- IP address
- City
- Region
- Country

If API fails → fallback to "N/A - Offline" values.

These variables are stored at **module level** and pulled into the class.

3. Concurrency Model

A. Main Thread

- Runs Tk event loop
- Renders UI
- Handles button clicks
- Executes after()-scheduled UI updates

B. Worker Thread (SOS sequence)

- Sleeps (blocking operations)
- Simulates network communication
- Does *not* touch UI directly

Communication:
Worker thread → Tk thread via root.after()
This keeps the structure thread-safe.

4. Data Flow Summary

On App Startup:

1. requests.get(ip.info) → receives geolocation JSON
2. Stores results in global vars
3. Dashboard loads and displays them

On SOS Button Press:

1. UI changes to ALERT
2. Thread begins background SOS workflow
3. After timeouts, thread sends results → main thread updates via after()

IMPLEMENTATION DETAILS

The ArmorBelle Safety Dashboard is designed as a lightweight, event-driven desktop application constructed using Python's Tkinter framework. The architecture integrates three main components: (1) a graphical user interface (GUI), (2) a threaded SOS event engine, and (3) a data retrieval layer for geolocation information. The system follows a monolithic structure but internally separates concerns through class-based encapsulation and state-driven design principles.

The application centers around a single controller class, SafetyDashboardApp, which initializes the UI, maintains state, and orchestrates background tasks. All runtime behavior is governed by the dashboard's SOS state machine, composed of three abstract states: **SAFE**, **ALERT**, and **TIMEOUT**. These states act as theoretical representations of system conditions, influencing both behavior (thread logic) and visual output (UI color and text).

Core System Logic

SOS Event Cycle

The SOS event sequence follows a predictable, deterministic workflow implemented within a dedicated background thread to maintain GUI responsiveness. The theoretical model follows a three-stage pipeline:

1. **Initiation Stage (ALERT)**
   User input triggers the transition from *SAFE* → *ALERT*. A 5-second delay simulates communication preparation.

2. **Pending Stage (TIMEOUT)**
   The system transitions from *ALERT → TIMEOUT* to represent waiting for acknowledgment. A second delay simulates remote server response time.

3. **Resolution Stage**
   A randomized success/failure event illustrates external response uncertainty. Depending on outcome, the system produces an informational or error state before returning to *SAFE*.

The key theoretical constraint is thread separation: Tkinter mandates that UI updates must occur exclusively on the main event loop. Therefore, background work communicates through scheduled callbacks (Tk's after() mechanism), exemplifying asynchronous UI-safe operations.

State Abstraction

Internally, UI and logic are synchronized through the abstract state variable self.current_sos_state. Each state symbolically represents a system condition and dictates:

- Status message semantics
- Indicator color selection
- Button behavior
- Animation patterns (e.g., pulsing in ALERT)

This creates a decoupled and predictable theoretical state machine.

Geolocation Data Handling

At startup, the system attempts to acquire geographic information (IP, city, region, country) via an external API. Conceptually, this forms a "best-effort" data layer: failures gracefully degrade to offline placeholders, ensuring functional stability.

UI Design Theory

The interface design follows classical dashboard principles, emphasizing clarity, legibility, and state visibility. The theoretical composition includes:

Visual Layout Model

The UI is divided into three conceptual regions:

1. **Status Panel**
   Offers an immediate visual summary of system state using a text descriptor and colored circular indicator. This dual representation reinforces visual cognition

principles—color and position provide rapid “at-a-glance” comprehension.

2. **Location & Data Panel**
   Presents contextual information such as IP and location. This panel also serves as the log output area, reinforcing the feedback loop between system operations and user awareness.

3. **Control Action Panel**
   The SOS button acts as a binary state controller. Its text and color alter dynamically, visually signaling the active mode of the system.

Color Theory & Information Encoding

Color is used as a semantic indicator:

- **Teal (SAFE)** communicates stability.
- **Red/Orange (ALERT)** signals danger, enhanced with a pulsing animation.
- **Yellow (TIMEOUT)** conveys uncertainty or ongoing processing.

Through consistent usage, colors form a cognitive key that enhances interpretability.

Typography & Aesthetics

Serif fonts (Georgia, Times New Roman) reinforce formality and reliability—attributes desirable in a safety tool. Minimalistic visual structure focuses user attention on status and controls rather than decorative elements.

FUTURE ENHANCEMENTS

UI/UX Improvements

- **Modern Themed Widgets:** Transitioning to CustomTkinter or PyQt could enhance aesthetic uniformity and introduce responsive layouts.
- **Dynamic Animations:** Smooth transitions, toast notifications, and gradient indicators may create a more engaging and intuitive interface.
- **Dark/Light Themes:** User-selectable themes improve accessibility and versatility.

Functional Extensions

- **Real SOS Communications:** Integrating SMS, email, or cloud APIs would transform the dashboard from simulation to operational tool.

- **GPS Integration:** Replacing GeoIP with hardware or mobile GPS enables precise tracking.
- **Event Logging:** Storing entries in SQLite allows long-term history and audit trails.

System Architecture Growth

- **Modularization:** Separating UI, logic, and data into dedicated modules supports maintainability and scalability.
- **State Machine Framework:** Using a formal FSM library would enhance reliability and testability.
- **Improved Concurrency Model:** Thread pooling or async paradigms would reduce overhead and increase resilience to repeated SOS activations.

Security Enhancements

- **Encrypted Communication:** Ensuring confidentiality of SOS transmissions.
- **Privacy Controls:** Allow users to disable geolocation or anonymize data when desired.

CONCLUSION

Overall this solution will give the sense of safety and protection to all the women out there and help the police in taking action against the culprits behind such abominable acts. With this solution in hand, women will live carefree and will get their sense of lost freedom back. This solution will be beneficial in building a society where equality resides and both men and women will have equal & fair competition.