W-Secure: Women Safety App

Mayank Sharma (E22CSEU0839) Shivansh Chauhan (E22CSEU0814)
Rakesh Sharma (E22CSEU0823)

Abstract

In today's world, women's safety is a critical issue that requires innovative technological solutions to ensure protection. This project presents a Women Safety App, W-Secure, that utilizes video recording and mobile application-based SOS features to monitor and analyze a woman's surroundings in real-time. Mobile applications for women's safety often focus on utilizing features like location tracking and SMS alerts [13], and some newer apps incorporate features like shake or scream detection to trigger alerts [16].

The project employs AI-based video analytics (computer vision) to detect potential threats, such as harassment, violence, or distress signals, by analyzing body language, facial expressions, and unusual activities like men surrounding women suspiciously. Advances in deep convolutional neural networks have significantly improved tasks like image classification and detection [8, 6], which form the basis for such analysis.

Upon detecting a threat, the app will automatically:

- Alert emergency contacts added by the user.
- Notify nearby mobile application users with an ALERT.
- Dial the Women Helpline along with location details.

Additionally, the system integrates AI algorithms (computer vision and deep learning) to improve threat detection accuracy over time. By utilizing real-time surveillance, AI-based threat detection, mobile SOS features, and automated emergency response, this app aims to create a safer environment for women.

1 Methodology

1.1 Android App Development

Objective

Build a security-focused Android app with discreet emergency features. Development can leverage existing guides and practices for Android programming [4]. Frameworks like Flutter and Dart are also used for building cross-platform applications, often integrated with Firebase for backend services [3, 16].

Key Features & Implementation

- **Location Sharing**: Use GeoLocator for real-time GPS tracking. Sharing location details is a common feature in safety apps [13].
- **Helpline Calling**: Pre-set emergency numbers (e.g., 100, 1091). One-tap calling without confirmation pop-ups.
- **Stealth Video Recording & Storage**: Use Android's `MediaRecorder` API to record video without opening the camera UI. Upload recordings directly to cloud storage (Firebase Storage) to prevent tampering. Firebase is a common backend for mobile apps [3].
- **Safe Spaces Marking**: Utilize Stadia Maps API (or similar) to mark verified safe places (e.g., police stations, hospitals, malls).

- **SOS Button with Live Location**: Implement a persistent floating SOS button. This allows users to quickly send alerts with their location to contacts [16].
- **Phone Shutdown Prevention**: Use `DeviceAdminReceiver` and `DevicePolicyManager` to restrict power-off options. Implement a background service to restart the app in case of forced closure.

1.2 AI Models for CCTV-Based Threat Detection

Objective

Detect potential threats by analyzing movement patterns and gestures. AI analysis of posture and movement is also explored in areas like sports injury prediction [7].

AI Models & Techniques

1. **Human Detection & Tracking (Multi-Object Tracking)**: Use YOLOv8 [2] or Mask R-CNN for real-time person detection. YOLO models (like YOLOv5 [15] and YOLOv8 [2]) are known for object detection, with research focusing on adapting them for specific challenges like small object detection [2]. Implement DeepSORT for tracking multiple individuals, building on techniques for tracking objects [14].
2. **Pattern Recognition (Women in Distress Detection)**: Identify anomalies like a single woman being followed by multiple men. Use a threshold system (e.g., 3+ men following for X seconds triggers an alert). Anomaly detection is a broad research area [7].
3. **Gesture & Behavioral Analysis**: Use OpenPose or MediaPipe [10] to analyze body posture and movements. MediaPipe provides capabilities for real-time hand tracking, useful for gesture recognition [10, 12]. Train an LSTM-based CNN model to detect: timid or defensive gestures (e.g., arms crossed, looking back frequently), sudden movements like running or erratic walking. Compare detected behavior with predefined distress patterns. Deep learning models, including CNNs [1] and combinations like ResNet-BiGRU [7] or ResNet-1D-CNN [11], are often employed for classification and analysis tasks based on visual or sequential data.

1.3 Backend & Cloud Infrastructure

- **Cloud Storage**: Firebase for video storage.
- **Database**: Firebase Firestore for user & emergency data. Firebase is frequently used with Flutter/Dart apps [3].

1.4 Development Phases

1.4.1 Phase 1: Research & Planning

- Understanding User Requirements: Define key features like SOS alerts, live location tracking, Safe Spaces, and video recording. Surveys of existing apps can inform requirements [13].
- Identifying AI Use Cases: Crowd Analysis: Tracking the number of men following a woman. Gesture Recognition: Recognizing distress gestures (timid, running, etc.).
- Finding & Collecting Datasets: Crowd Analysis: Datasets like CrowdHuman, MOT17 for tracking individuals. Gesture Recognition: Datasets like Jester, NTU RGB+D for identifying distress-related gestures. Large datasets like ImageNet are crucial for training deep learning models [8].

1.4.2 Phase 2: AI Model Development

- Choosing the Best AI Model: Mask R-CNN for detecting and segmenting individuals. YOLOv8 [2] or EfficientDet for fast person detection and tracking, leveraging advancements in object detection [15]. LSTM/RNN-based gesture recognition for distress detection. Models like ResNet [11] are also used in related areas.

- Training & Fine-Tuning the Model: Train AI models on datasets using PyTorch/TensorFlow. Libraries like NumPy are essential for numerical operations [9]. Fine-tune the models on real-world surveillance videos.
- Testing AI Models: Validate models with test datasets. Optimize model accuracy and reduce false positives.

1.4.3 Phase 3: Mobile Application Development

- Setting Up the Flutter Project: Configure Firebase for authentication, real-time database, and storage. Implement Geolocator for live location tracking. Flutter/Dart with Firebase is a common stack [3, 16].
- Building Core Features: Safe Spaces Mapping - Show Safe Spaces using Google Maps. SOS Alert System - Implement an SOS button to send live location to emergency contacts [16]. Stealth Video Recording - Capture and upload video without opening the camera UI. Blocking Phone Shutdown - Prevent forced shutdown in distress situations.

1.4.4 Phase 4: AI Integration in CCTV Systems

- Deploying Models on Edge Devices: Convert AI models to TFLite/ONNX for real-time CCTV processing. Deploy on edge devices like NVIDIA Jetson or Raspberry Pi.
- Testing AI in Real-World Scenarios: Evaluate AI performance on CCTV footage. Improve detection speed and accuracy.

1.4.5 Phase 5: System Testing & Improvement

- Real-World Testing: Validate the app in real-life distress scenarios. Test AI models on live surveillance feeds.
- Performance Optimization: Reduce app latency and AI processing time. Enhance UI/UX for better usability.
- Feedback & Iteration: Gather user feedback and refine security features. Improve AI accuracy and reduce false alerts.

1.4.6 Phase 6: Deployment & Maintenance

- Launch the App on Google Play Store: Ensure compliance with security and privacy policies.
- AI Model Deployment on CCTV Networks: Install models in public places for distress detection.
- Continuous Monitoring & Updates: Improve AI models with new datasets. Update the app with new security features.

References

[1] G. Levi and T. Hassner. (2015). Age and gender classification using convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 34–42). `https://doi.org/10.1109/cvprw.2015.7301352`

[2] M. Chitra, H. Balasubramanian, V. L. Srinivaas, and K. Jayanth. (2024). An Adaptation and Evaluation for Object Detection of Small Objects Using YOLO8. In *International Conference on Evolutionary Computation.* `https://doi.org/10.1109/icec59683.2024.10837540`

[3] K. Marimuthu, A. Panneerselvam, S. Selvaraj, L. P. Venkatesan, and V. Sivaganesan. (2023). Android Based College App Using Flutter Dart. *Green Intelligent Systems and Applications*, 3(2), 69–85. `https://doi.org/10.53623/gisa.v3i2.269`

[4] B. Sills, K. Marsicano, C. Stewart, and B. Gardner. (2022). *Android programming: the Big Nerd Ranch guide* (5th edition). Big Nerd Ranch, LLC. ISBN: 978-0-13-764554-1

[5] Dr. H.M. Naga Raju. (2025). Enhanced Age and Gender Estimation Using Opencv. *INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT.* https://doi.org/10.55041/ijsrem43052

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–9). https://doi.org/10.1109/cvpr.2015.7298594

[7] Q. Wan, Z. Zhang, L. Jiang, Z. Wang, and Y. Zhou. (2024). Image anomaly detection and prediction scheme based on SSA optimized ResNet50-BiGRU model. *arXiv preprint arXiv:2406.13987.* https://doi.org/10.48550/arxiv.2406.13987

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2017). ImageNet classification with deep convolutional neural networks. *Communications of The ACM*, 60(6), 84–90. https://doi.org/10.1145/3065386

[9] I. Idris. (2014). *Learning NumPy Array.* Packt Publishing.

[10] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. (2020). MediaPipe Hands: On-device Real-time Hand Tracking. *arXiv preprint arXiv:2006.10214.* https://arxiv.org/abs/2006.10214

[11] Y. Saheed, O. H. Abdulganiyu, K. U. Majikumna, M. Mustapha, and A. D. Workneh. (2024). ResNet50-1D-CNN: A new lightweight resNet50-One-dimensional convolution neural network transfer learning-based approach for improved intrusion detection in cyber-physical systems. *International Journal of Critical Infrastructure Protection.* https://doi.org/10.1016/j.ijcip.2024.100674

[12] A. Anand, A. Pandey, S. Mehndiratta, H. Goyal, P. Kaushik, and R. Rathore. (2024). Smart AI basedVolume Control System: Gesture Recognition with OpenCV & Mediapipe Integration. In *IEEE International Conference on Solid Dielectrics.* https://doi.org/10.1109/icsd60021.2024.10751362

[13] R. Pavitra and S. Karthikeyan. (2017). Survey on womens safety mobile app development. In *International Conference on Innovations in Information, Embedded and Communication Systems* (pp. 1–5). https://doi.org/10.1109/iciiecs.2017.8276048

[14] X. Zhou, V. Koltun, and P. Krähenbühl. (2020). Tracking Objects as Points. In *European Conference on Computer Vision (ECCV).*

[15] G. Jocher, A. Stoken, J. Borovec, et al. (2021). ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations. *Zenodo.* https://doi.org/10.5281/zenodo.4679653

[16] V. Gawad, D. Mittal, D. Tiwari, N. Tiwari, and V. N. Kawtikwar. (2024). Women's Safety Application using Flutter and Dart. In *2024 4th International Conference on Soft Computing for Security Applications (ICSCSA).* https://doi.org/10.1109/icscsa64454.2024.00117