CSE 535 Mobile Computing (2023 Fall) Project 5 (Team 25) - TripSafe

Adrija Nag, Anurag Banerjee, Daljit Singh Bhalla, Nicole Diana Fernandes, Saif Masood, Soham Nag
{anag9, abaner40, dbhalla, nferna18, saif.mas94, snag8}@asu.edu
Arizona State University, Tempe, AZ

***Abstract*—In the rapidly evolving digital landscape, smartphones have become indispensable yet often passive companions to our dynamic surroundings. This paper introduces the Guardian Angel, a context-aware mobile application designed to revolutionize the user experience. Leveraging sensors, location data, and time-based information, the Guardian Angel ensures users are in control, informed, and safe. Safety takes precedence with features such as Car Crash Detection, Location-based Speed Limit Alert, and Drowsiness Detection and Alert System, offering a comprehensive solution for user well-being. This abstract outlines the key features and benefits of the Guardian Angel, presenting it as a holistic, user-centric application.**

I. Introduction

In an era where smartphones have seamlessly integrated into the fabric of our daily lives, the untapped potential of actively engaging with our surroundings persists. Addressing this gap, the Guardian Angel emerges as a pioneering solution, introducing a context-aware mobile application poised to redefine the user experience. It actively engages with your environment, leveraging sensors, location data, and time specifics to shape a mobile experience that's tailored and effortlessly integrated into your life.

This paper explores the Guardian Angel in-depth, examining its standout features with a strong emphasis on safety. The innovative Car Crash Detection feature underscores the application's dedication to providing swift assistance during emergencies. Concurrently, the Location-based Speed Limit Alert actively promotes responsible driving practices, contributing to a safer road environment. Additionally, the context-aware Drowsiness Detection and Alert System enhances user safety during driving, showcasing the application's proactive stance toward potential hazards.

The Guardian Angel incorporates technologies such as fuzzy logic controllers, Google Maps API, GPS, and machine learning models. This integration is aimed at elevating the application's capabilities, providing users with a sophisticated toolkit to make informed decisions. By intelligently adapting to the user's context, the Guardian Angel strives to empower individuals, emphasizing safety while enhancing efficiency and peace of mind in the relentless pace of the digital age.

II. Architecture

Figure 1 illustrates the structure of the Guardian Angel TripSafe app, consisting of four key components: Car crash detection, Location-based Speed Limit Alert API, Gas Station Suggestion, and Drowsiness Detection and Alert. The app leverages smartphone sensors and services to stay context-aware. For instance, GPS data supports crash detection, speed limit alerts, and gas station suggestions, while the camera aids in accurately gauging user alertness for drowsiness detection. The crash detection's precision is boosted by accelerometer data. The drowsiness detection employs a lightweight machine learning model, while real-time predictions for other features rely on frequent sensor data polling. Dialog alerts notify the user, and in emergencies, the app uses phone and SMS services to alert emergency responders of a hazardous situation.

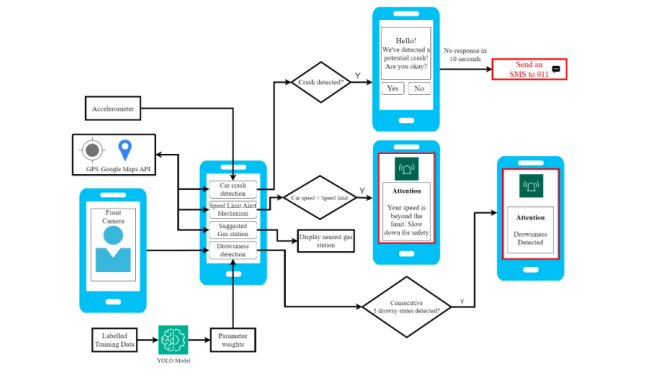

Fig. 1. Architecture diagram

Fig. 2. Main screen

III. The Application Suite

The different features are broadly categorized into driver safety and driver assistance. They are stitched together into a main activity and provided as options to the user. The user can pick the feature that would act as his guardian angel before starting the trip. Figure 2 depicts the main screen of the application. A brief description of each feature is provided below: Figure 2 illustrates the main screen UI.

- **Car Crash Detection:**
  The Car Crash Detection feature in Guardian Angel TripSafe operates through a straightforward process. It begins by regularly collecting data from the phone's GPS and accelerometer. When there's a sudden and significant change in both GPS and accelerometer readings, indicating a potential crash, the system uses fuzzy logic and estimates the likelihood of a crash based on the collected data. If there's a high probability of a crash, the user receives a quick notification prompting them to check-in. In cases where the user doesn't respond within a reasonable timeframe, the system takes proactive measures by sending a text to emergency services for assistance. This systematic and user-centric approach ensures a vigilant response to potential emergencies, prioritizing the safety of users on the road. Figure 3 illustrates the UI when a car crash is detected.

Fig. 3. Car crash detection feature

- **Location-based Speed Limit Alert API:**
  The Speed Limit Alert functionality in Guardian Angel TripSafe involves a systematic process to ensure user awareness and safety on the road. Firstly, the system periodically collects data on the vehicle's location and speed using GPS technology. It then seamlessly integrates with the OpenStreetMap API to fetch real-time speed limit information for the detected location. If the vehicle surpasses the designated speed limit, the system promptly notifies the user through a speech alert. This proactive approach aims to raise immediate awareness of overspeeding, promoting responsible driving habits, and contributing to a safer driving environment.

- **Gas Station Suggestion:**
  The Best Gas Station Suggestion feature in Guardian Angel TripSafe employs a straightforward process for optimal user experience. Firstly, it utilizes data effectively by tapping into the phone's GPS and the Google Maps API to precisely determine the user's location. Then, the app uses Places API to make a list of nearby gas stations. Leveraging fuzzy logic, the system then factors in considerations like traffic, rating, and distance to intelligently recommend the best gas station available. Finally, the recommendation is seamlessly presented to the user on the map present on their screen. This user-friendly approach ensures that drivers are efficiently directed to the most suitable gas station based on real-time factors, enhancing convenience and overall road travel experience. Figure 4 illustrates the UI when overspeeding is detected and the best gas station is suggested.

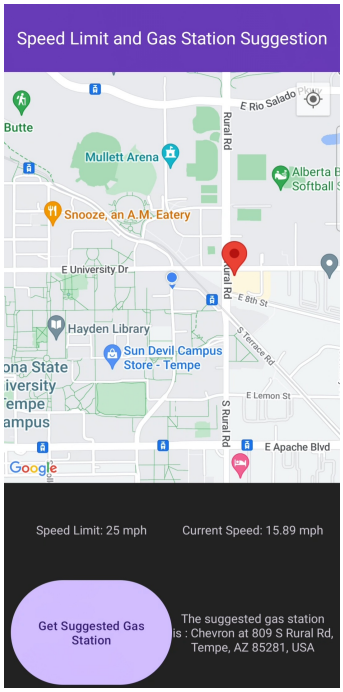

Fig. 4. Overspeeding detection and Gas Station suggestion

- **Drowsiness Detection and Alert:**
  The Drowsiness Detection and Alert feature in Guardian Angel TripSafe is designed with simplicity and effectiveness in mind. To initiate the system, the user positions their phone on the dashboard, allowing the front camera

to engage in real-time monitoring. The detection model at the core of this feature is powered by a YOLO (You Only Look Once) model, trained on custom-labeled data to accurately identify signs of drowsiness. When drowsiness is detected, the system promptly activates a 10-second alarm, serving as a swift and effective alert mechanism to notify the user. The integration process is seamless, with the YOLO model effortlessly embedded into the Android application through the use of the Chaquopy library. This user-friendly and tech-savvy approach ensures that users remain vigilant and alert while driving, promoting a safer and more secure journey on the road. Figure 5 illustrates the UI when drowsiness is detected.

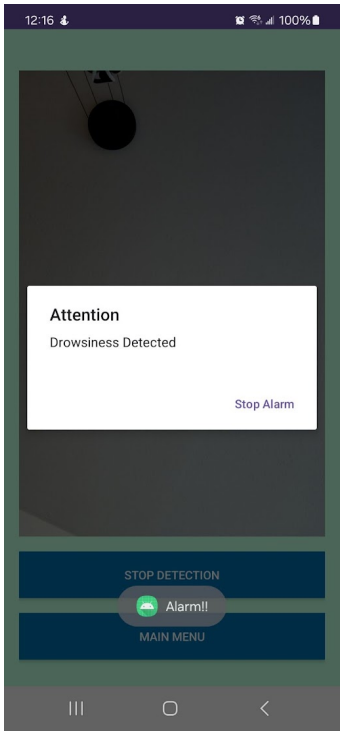

Fig. 5. Drowsiness detection

IV. Implementation

The Guardian Angel is developed using Kotlin as an Android application designed for Android-based smartphones. The application interacts with the sensors and services of the smartphone to receive updates concerning contextual changes. In particular, features relying on location services—such as Crash Detection, Speed Limit Alert, and Gas Station Suggestion—utilize the GPS-based location manager to access the car's location and speed. Moreover, the car crash detection feature incorporates the *SensorEventListener* to retrieve accelerometer data, enhancing its functionality.

$$Acceleration = \sqrt{x^2 + y^2 + z^2}$$

Every 5 seconds, the app checks the car's speed. If the speed plummets from a previously recorded high value above a set threshold (let's say 45 km/hr) to 0, accompanied by a substantial acceleration indicated by the accelerometer, both these readings become inputs for a fuzzy inference system.

Should this system forecast a collision probability surpassing 0.85, it triggers a dialog alert to notify the driver. In case the driver doesn't respond within 30 seconds, the app initiates an *ACTION_CALL* intent, prompting an automatic call to 911. This sequence aims to promptly alert the driver and, if necessary, engage emergency services in the event of a potential collision.

The Speed Limit Alert feature employs a comparable approach by periodically polling the GPS location to derive the speed. It computes the speed by dividing the difference between distances in consecutive polls by the time step. Polling is being done every 3 seconds. Subsequently, it queries the OSM API to acquire the current speed limit at that location. If the current speed exceeds the posted limit, the system notifies the user by sending a message through the speakers.

The Gas Station Suggestion takes in the location of the user being calculated during the speed limit alert feature. When the user presses the button to recommend a gas station, the location is taken and sent to the GOOGLE MAPS PLACES API to find the 3 nearest gas stations from the user's location. Once the gas stations are available, the GOOGLE MAPS DISTANCE MATRIX API is called to find out the traffic conditions on the route from the user to each of the 3 gas stations. For each gas station, the distance of the gas station from the user (using the GOOGLE MAPS DISTANCE MATRIX API), the traffic on the route from the current location of the user to the gas station (using the GOOGLE MAPS DISTANCE MATRIX API) and the rating of the gas station (using the GOOGLE MAPS PLACES API) is fed into the fuzzy logic engine (written in Kotlin using the jfuzzylite library [4]) to generate the fuzzy rating for each of the gas station. Finally, the gas station which is rated the highest by the fuzzy logic engine is recommended to the user and shown on the map.

The development of the Drowsiness Detection feature in our application involved a systematic approach. Firstly, a custom dataset was created, featuring images of human faces in both awake and drowsy states, crucial for a robust detection model. Using the LabelImg tool, each image was labeled as 'awake' or 'drowsy.' The core of the functionality lies in the YOLO v5 model [2], specifically Ultralytic's version. Trained on the diverse dataset, the model learned to distinguish between awake and drowsy states through a rigorous training process, resulting in the generation of a 'last.pt' file containing model weights and parameters.

To make this drowsiness detection accessible within our Android app, a 'drowsiness.py' Python script was developed. This script captures frames from the device's video recording and applies the custom-trained model to determine the human face's state. The integration into the Android app is facilitated by the Chaquopy library [1]. Within the app, the TextureView class displays the front camera preview using the Camerax library [3]. The 'detect drowsiness' button initiates

video recording, and frames are sent to the Python code via Chaquopy. If drowsiness is detected consecutively twice, an alert dialog is displayed, accompanied by a 10-second alarm using the AlertDialog class and MediaPlayer class, respectively. The user can stop the alarm by clicking on the alert, ensuring a timely response to drowsiness for enhanced safety.

Code Repository (it is a private repository but ImpactLabASU has been added as a collaborator) can be found at: https://www.github.com/anuragbnrj/CSE_535_MC_Project_4_and_5_Group_25_TripSafe

V. Demonstration

We installed TripSafe app on an Android device that can support all the features. We then conducted real-world simulations to assess the functionality of the TripSafe app. To evaluate the Speed Limit and Gas Station Suggestion features, we activated them while traveling on the Metro Valley bus in Tempe. Aligning with the road's posted speed limit, as verified by the OSM API, no alert was triggered as the bus adhered to the speed limits. Additionally, while on the bus, we prompted the TripSafe app to recommend nearby gas stations. It accurately pinpointed the closest gas station by marking its location on the map. Since testing overspeeding could breach legal limits, we utilized synthetic data to simulate exceeding the speed limit and, consequently, triggering the alarm. Testing the crash detection system posed a significant challenge as it was impractical to replicate a genuine car collision scenario. As an alternative, we utilized synthetic data, setting the car's speed at 49 km/hr, exceeding the predetermined threshold of 45 km/hr—this simulated the assumed speed leading up to the hypothetical collision.

To simulate accelerometer readings akin to those in a crash, we dropped the phone onto a mattress, generating acceleration values surpassing the predefined threshold. The fuzzy system processed both sets of data, predicting a collision likelihood of 0.86, triggering an alert, and initiating a call to emergency services. For the demonstration, the call was directed to one of our project members instead of 911.

In our hands-on demo, we tested the Drowsiness Detection by using the phone's front camera to show our own faces, presenting our faces as we're either awake or drowsy. The outcome was a prompt display of an alert dialog and the activation of a 10-second alarm, effectively illustrating the feature's functionality. This simple test demonstrated that the system can quickly recognize drowsiness and respond promptly, highlighting its practical use in real-life situations for user safety, especially during activities like driving.

VI. Conclusion

In this paper, we demonstrate a suite of features that provide safety, general, and emergency assistance to the user. The app architecture ensures a robust system that intelligently combines data from multiple sources, employs advanced algorithms and utilizes cutting-edge technologies to enhance safety and provide valuable assistance to users during their trips. All these functionalities truly make TripSafe a trusted companion on any road trip, put differently, A Guardian Angel.

References

[1] "The easiest way to use Python in your Android app," *Chaquopy*. https://chaquo.com/chaquopy/

[2] "ultralytics/yolov5," *GitHub*, May 14, 2022. https://github.com/ultralytics/yolov5/

[3] "CameraX overview," *Android Developers*. https://developer.android.com/training/camerax

[4] "FuzzyLite - The FuzzyLite Libraries for Fuzzy Logic Control", *FuzzyLite*. https://fuzzylite.com/