DriveSafely

By:

Reece Gavin– 17197589

Supervised By:

Prof. Hussain Mahdi

Submitted to the University of Limerick in partial fulfilment of the requirements for the Bachelor of Electronic and Computer Engineering

Department of Electronic & Computer Engineering

University of Limerick

March 2021

Abstract

In Ireland and throughout the world, fatigued driving is an ever-growing problem. A survey conducted by the RSA in 2014 found that over 1 in 10 motorists have fallen asleep at the wheel. With technological advances in recent years, it is now common in newer cars to see crash avoidance systems and driver monitoring systems. However, these can often be expensive optional extras on cars and commercial vehicles. Therefore, there is potential to introduce a low-cost, high-accuracy mobile application that will help the prevention of car accidents caused by driver fatigue and tiredness. In this report, the systems currently available today are discussed, and the design, implementation, and testing of the DriveSafely application are explored in detail. The app continuously monitors a driver's face and eyes to detect different driver states and identify clues of tiredness, fatigue, and/or distraction. The app operates an alarm system to alert a detected tired driver and also has other features such as sending text messages to an emergency contact. The layout design of the application had to be clear, user-friendly, and adaptable to many screen resolutions. The main tools used in the creation of the DriveSafely app were Android Studio SDK and Google Vision.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hussain Mahdi, for his advice and support throughout the entire project.

I would also like to acknowledge the support of my friends Jack Nolan, Seán Browne, and Jane Fitzgerald both in technical aspects of the project and in moral support. Whenever I needed to talk through an issue or get advice on a particular aspect of the project, they were always there to lend an ear.

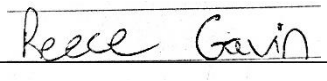
Finally, I would like to take this opportunity to thank my family, who were there to support me throughout the project.

Declaration of Authorship

I, Reece Gavin, declare that this report, titled “DriveSafely” and the project described within are my own.

I confirm that:

- This work was done as part fulfilment of the requirements of the Bachelor of Engineering course.
- Where I have used or consulted the published work of others, it is clearly stated. Apart from the consultations of those sources, the work is entirely my own.
- I have acknowledged all sources used for this work.
- This report has not previously been submitted or published before submission.

Signed:  Date: 19/03/2021

Reece Gavin

Table of Contents

Table of Figures

Table of Code Snippets

Table of Tables

Table of Acronyms

API: Application Programming Interface

APK: Android Package Kit

CPU: Central Processing Unit

CSO: Central Statistics Office

DEX: Dalvik Executable

GPS: Global Positioning System

GUI: Graphical User Interface

IDE: Integrated Development Environment

IR: Infrared

OS: Operating System

PIP: Picture-in-picture

RSA: Road Safety Authority

SDK: Software Development Kit

SMS: Short Message Service

UI: User Interface

UX: User Experience

XML: Extensible Markup Language

Chapter 1: Introduction

1.1 Motivation

When the time came to choose a topic for my final year project, I was interested in this project due to it being a software-based project and related to cars and driving. I have a strong interest in software, and I am also a car enthusiast. By completing this project, I have been able to gain further experience with Java and become proficient in XML (Extensible Markup Language). I have also gained experience with tools such as Android Studio, and Google Vision, all of which are discussed in detail in this report.

1.2 Rationale

The Road Safety Authority of Ireland (RSA) defines fatigue as the physical and mental impairment brought about by inadequate rest over a period of time [1].

Studies show that the general recommended amount of sleep required to promote and maintain optimal health for adults, aged between 18 and 60 years, is 7 or more hours on a continual basis [2]. Drivers that are suffering from a lack of sleep are at a high risk of falling asleep while driving, thus increasing their chances of being involved in a road crash.

A survey conducted by the RSA in 2014 found that over 1 in 10 motorists have fallen asleep at the wheel [1]. The Central Statistics Office (CSO) had recorded that a total of 2,710,664 Irish driving licenses were held at the end of 2014 [3]. This survey infers that approximately 271,000 drivers have fallen asleep at the wheel in Ireland. The RSA survey also found that motorists who drive as part of their work, and motorists who admit to driving after taking any amount of alcohol, had a higher-than-average incidence rate of falling asleep at the wheel (almost 1 in 5 fell asleep at the wheel) [1].

In 2020 the RSA presented the results of two studies at its annual lecture. One study found that 24% of drivers in Ireland admitted they had driven at least once over the previous month when they were so tired, they had trouble keeping their eyes open. In another study, 16% of drivers admitted they had fallen asleep or nodded off at the wheel. The Road Safety Authority has said driver fatigue is estimated to be a factor in 1 in 5 of the deaths of drivers on Irish roads. Fatigue-related incidents are 3 times more likely to be fatal or result in serious injury due to high impact speed due to an absence of action by the driver.

The RSA says that the groups most at risk are young men, people working night shifts, those who drive for a living (such as lorry and taxi drivers), and people with sleep disorders (such as sleep apnoea) [4].

In Ireland, the rate at which drivers have admitted to falling asleep at the wheel has increased from 10% in 2014 to 16% in 2020, which calls for a need to reduce this rate and stop the growing trend. The promoted solution for driver fatigue is to stop, have 2 cups of strong coffee and have a nap of no more than 15 minutes before proceeding back onto the road [5]. However, even with this promotion in place, there is a need to take further action in reducing the amount of fatigue-related driver incidents.

Newer vehicles are being equipped with driver monitoring systems. However, these systems can be very expensive optional extras that a large proportion of the population may not be able to afford to purchase. With rapid technological increases in the smartphone market and the improvement of camera technology in the last ten years, it is now possible to use a smartphone to accurately detect and recognise faces and their features.

Currently, there are Android applications capable of tracking a user's eyes to detect when they are closed. However, some of these applications are outdated, unintuitive, and lacking features. Some apps have obtrusive ads which require the user to pay to remove them.

This project attempts to create an application that will provide a modern, responsive, free, and accurate alternative to existing applications. The project focuses on the development of an Android mobile application that will be more effective than other systems currently deployed.

1.3 Aims of the Project

The project aims to design and develop a user-friendly mobile application for Android devices, which allows them to unobtrusively monitor themselves while driving, with useful features such as being able to use Google maps at the same time. The system detects when a user is beginning to get tired and alerts them if they have fallen asleep. The application is designed to be easily used by all age groups. The project uses Java, XML, and the Android Studio integrated development environment (IDE) to create a user interface.

1.4 Project Outline

This project involves creating a user interface through which the user can start face and eye tracking. The project also involves adding a number of added features to entice the general public to use this application. These features include being able to use Google Maps at the same

time as using the application and being able to see information such as average speed, distance and, time for their previous three journeys. This project involved very thorough testing to ensure that the application works on multiple device resolutions and to ensure that all functionality worked as intended. The following subsections go into detail regarding these elements of the application's development.

1.4.1 Main Application

The primary function of the application is to detect and alert the driver of a vehicle to drowsiness/fatigue.

On opening the application, the user is presented with the application dashboard. Here, there are four options. They can start the face tracking, view recent trips, adjust settings, or get help. On entering the main option, "Start Face Tracking", the user is presented with a preview of the front-facing camera. There are four other information cards present on the screen. These show the user the current journey time, their current speed, their distance travelled, and the number of drowsiness detections. There is an option for the user to disable the camera preview if they feel they may be distracted by seeing themselves on the screen. There is also the option for the user to open Google Maps while continuing to track their face and eyes. If a user's face is initially detected and then goes missing for two seconds or more, a green alert dialog box will be shown to the user. If the user does not blink five times or more in a single minute, a yellow alert will be sounded. This will warn the user that they may be tired and should pull over to get a coffee and to consider stopping. The value of five was selected after reviewing work by *Islam et al* [6]. The average number of blinks per minute, for a user between the hours of 11 pm to 1 am was determined to be 3.33. A value of five was selected to allow for any errors that may be present in the system. A yellow alert will also be triggered if a user's head is calculated to have dipped five times. This is done by comparing the angle of the user's face to the angle it was at three seconds previously. If a user's face is calculated to have been at an angle of 15 degrees less than what it was three seconds ago, for more than five hundred milliseconds, a variable will be incremented by one. Once this variable reaches a value of five, the yellow alert will be triggered. Through application testing, this was determined to be the best way to detect if a user's head was dipping. Finally, if a user's eyes are closed for a user-selectable amount of time, a red alert will be sounded, and an SMS (Short Message Service) message will be sent to a chosen emergency contact if the user has chosen one.

1.4.2 Additional Developments

Recent trips can be viewed, which will show the user the length of time, distance, average speed and the number of detections for their last three journeys. There are additional settings such as the being able to change the length of time before the red alert is sounded, three different alert tones that can be chosen by the user and a user can choose any contact from their contacts list to be selected as their emergency contact. The application will retain the user's alarm delay time, alarm tone and emergency contact for each time the user opens the application until one of the settings is changed. The user has the added ability of being able to open Google Maps and make use of the turn-by-turn navigation that Google Maps offers. The incentive behind doing this is so that anyone who would have been using Google Maps to get to their destination, can now firstly open DriveSafely and be able to monitor themselves while being able to see directions to their destination.

1.4.3 Testing

The testing of the application is a particularly important aspect of the project. The application has been assessed on many different Android mobile phones and emulators to ensure that the application functions correctly on a wide range of phone screen sizes and resolutions.

The user interface was also evaluated in terms of functionality. The user interface has many features such as buttons, seek bars, text views and dialog boxes. These all must be tested to ensure they are functioning correctly.

Another major part of the project that had to be evaluated thoroughly is the face and eye tracker. Testing was necessary to ensure that the tracker works in all environments and at a range of angles and distances, as well as at night-time.

1.5 Report Layout

This report is divided into the following six sections:

- Similar systems:

This section discusses other projects and systems which are similar to this project. It gives a detailed overview of both the positive and negative aspects of their design and functionality.

- System Design:

The design and planning undertaken for this project are discussed in this section. This section details the necessary hardware and software requirements as well as detailing a class diagram, a sequence diagram and the overall application flowchart.

- Technology:

In this section, the technology used throughout this project is discussed such as Android Studio and XML.

- System Implementation:

In this section, the system implementation is described in detail.

- Testing and Evaluation:

This section details the testing of the system. It describes the tests done to ensure all aspects of the application work such as the user interface (UI) and the face tracker.

- Conclusion & Recommendations:

This section explores the results of the project. It explains what was achieved by completing the development of this project. This section also addresses aspects of the project that could be improved upon and providing recommendations for further development.

Chapter 2: Similar Systems & Technology

Over the last number of years, several driver monitoring systems have been created to try and prevent road traffic accidents due to fatigue and tiredness. The following subsections give an overview of some of the main systems currently available. The systems shown were downloaded and tested thoroughly for this report.

2.1 Drowsy Driving Alert: Sleepy Driver Warning

2.1.1 Overview

Drowsy Driving Alert (known as DDA) is a mobile application used to detect a driver's face and eyes and wake them up if their eyes are closed [7]. This application is similar to this project in that it tracks a user's face and eyes, as well as being an Android application.

2.1.2 Functionality

On the first opening of the application, a dialog box is shown. The user is asked do they want to share the app (See *Figure 1a*). Then, a dialog box, similar to the previous one, is shown asking do they want to review the app. This is disgruntling to the user especially as it happens each time the app is opened, and it takes away from the overall user experience (UX). Once these dialog boxes are dismissed, the user is presented with a simple layout (See *Figure 1b*). There is a drop-down options menu where a user can choose from seven different alert sounds (See *Figure 1c*). There is a slider that allows the user to adjust the phone's volume. There is a button to test the alert sound, so a user can hear it before they start the face tracking. There is a drop-down menu in the top right corner of the screen which gives users the option of updating the app, rating the app, viewing more apps created by the developer, sharing the app and leaving feedback (See *Figure 1d*). On pressing the start button, the user is presented with a full-screen image of themselves, with a yellow bounding box being drawn around their face (See *Figure 1e*). There is one button visible on the screen which is a button to activate Picture-in-picture (PIP). On activating PIP, the preview image of the user is shrunk and now the user can use other applications on their phone (See *Figure 1f*). Overall, the app has an easy-to-use layout with simple and effective buttons and options.

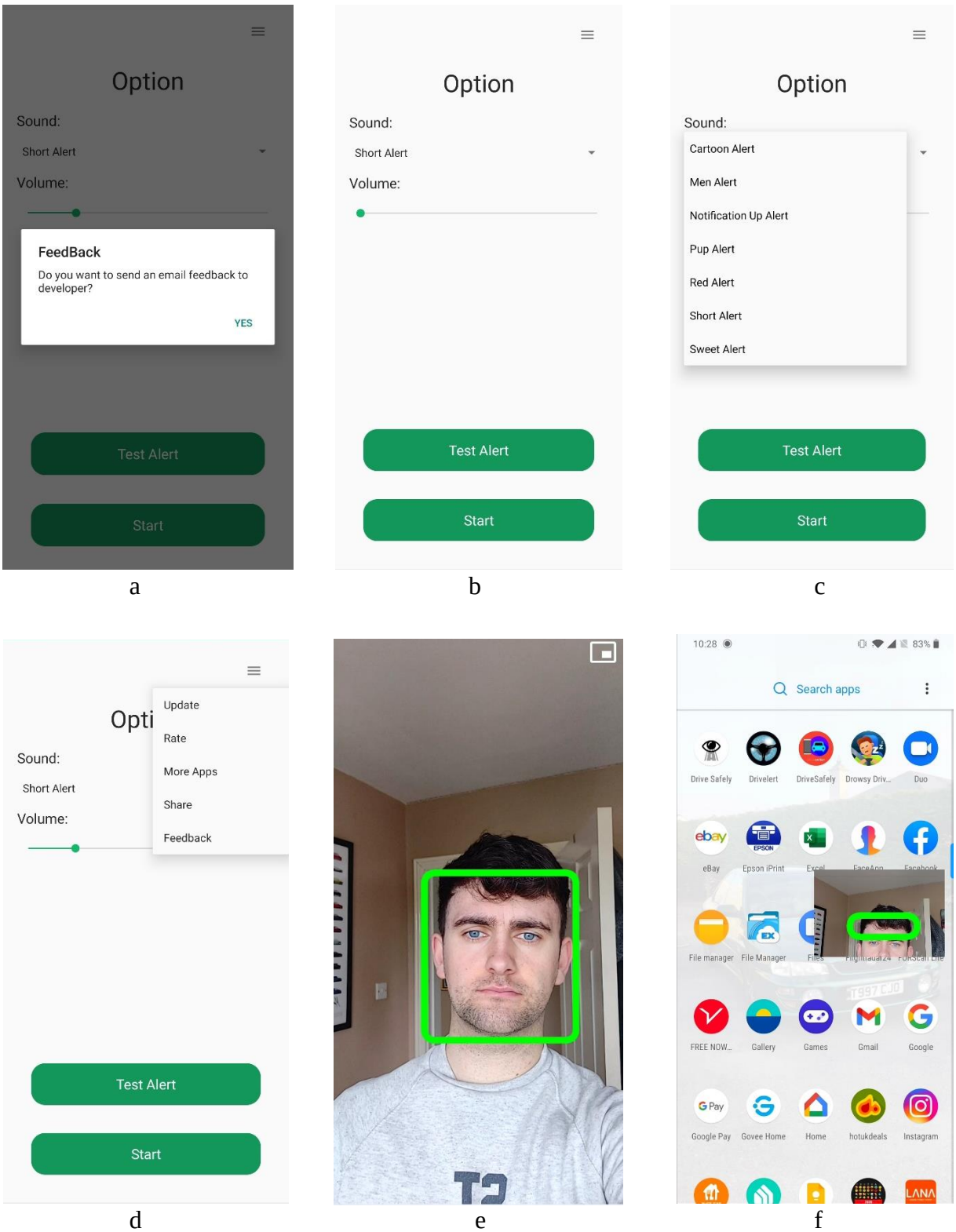


Figure 1: Screenshots of Drowsy Driving Alert *[7]* – a, b, c, d, e, f

2.1.3 Discussion

There are a number of aspects of this application that are similar to the application created throughout this project. The ability to select multiple sounds, use PIP and track a user's face and eyes are the major similarities. The ideology of a simple layout was similar to this project's application also. However, this application is lacking functionality such as being able to adjust the amount of time before an alert is sounded. The default is approximately 1 second. There is no option to remove the camera preview if a user were to find it distracting and there is no option to send a message to an emergency contact. It was observed that the app routinely froze especially after an alert had been signalled. It was also observed that when in PIP mode, the aspect ratio of the camera preview was not retained, so it may be visually difficult for a user to see if their face is being detected when the phone is mounted at a distance.

2.2 Drivelert

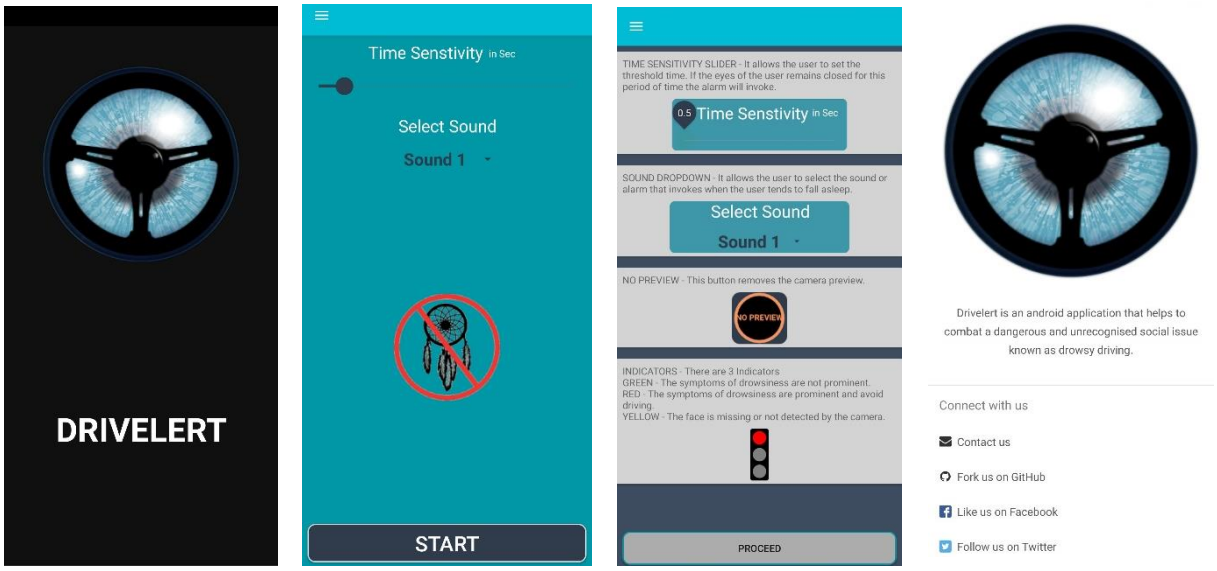
2.2.1 Overview

Drivelert is an Android application that helps to combat an unrecognised social issue known as drowsy driving [8]. It tracks the driver's eyes and initiates an alert when a driver tends to sleep while driving. Drivelert is a personal drowsiness detector helping a user to stay alert and drive safe. This application has the same purpose as this project, to detect and alert drowsy drivers using the front-facing camera on a smartphone.

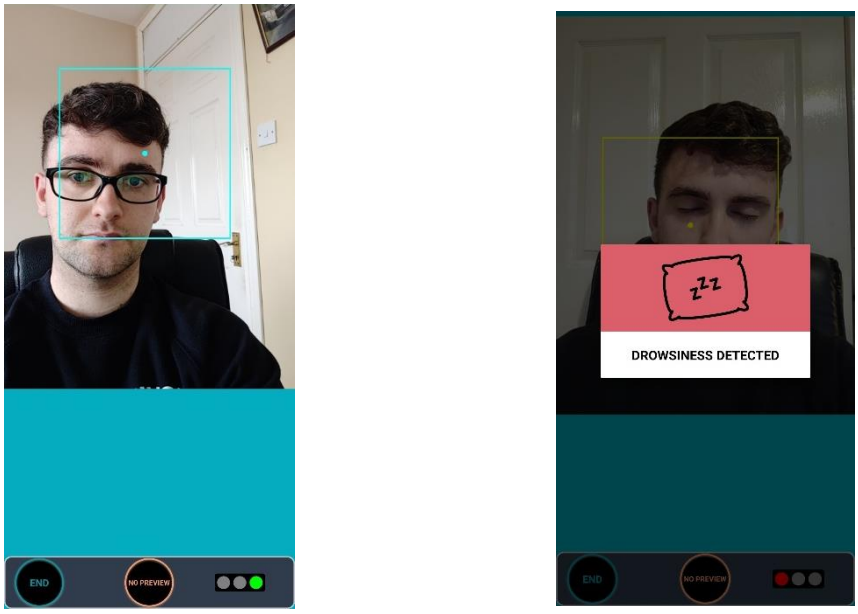
2.2.2 Functionality

On opening the application, the user is presented with a splash screen with the Drivelert logo displayed for approximately two seconds (See *Figure 2a*). The user is then brought to the main dashboard screen of the app (See *Figure 2b*). Here a user can change the time-sensitivity of how long their eyes have to be closed before the alarm will sound. The user can also select one of three different sounds on this page. There is a main start button that is used to start the face and eye tracking. In the top left corner, there is a navigation bar where a user can navigate to a help screen or a contact screen. On the help screen (See *Figure 2c*) the different aspects of the settings and application functions are explained to the user. On navigating to the contact screen (See *Figure 2d*) the user is given a description of the application and some buttons to email the developer and to follow them on Facebook/GitHub/Twitter. On selecting the start button from the main dashboard, the user is presented with a camera preview of themselves with a bounding

box drawn around their face (See *Figure 2e*). On this screen, there is a button to hide the camera preview if a user finds it distracting. There is also an end button to finish the face and eye tracking. In the bottom right-hand corner of the screen, there is a traffic light system displayed. As described in the help section, a green light means no symptoms of drowsiness are detected, a yellow light means the face is missing or not detected by the camera and a red light means that symptoms of drowsiness are detected, and the driver should avoid continuing to drive. When the user's eyes are closed for the user set amount of time, they are presented with a visual and audible alert (See *Figure 2f*).



a b c d



e f

Figure 2: Screenshots of Drivelert [8] – a, b, c, d, e, f

2.2.3 Discussion

Overall, the application UI is simple to understand and navigate. However, the UI is dated. The bounding box that is drawn around the users' face is slow to respond and does not centre on the user's face sometimes (See *Figure 2e*). The face and eye tracking work as intended. The traffic light system is not entirely accurate and can give false readings. The application has the same intended purpose as this project; however, this project aims to improve on the faults of this application and add more features to improve the overall user experience.

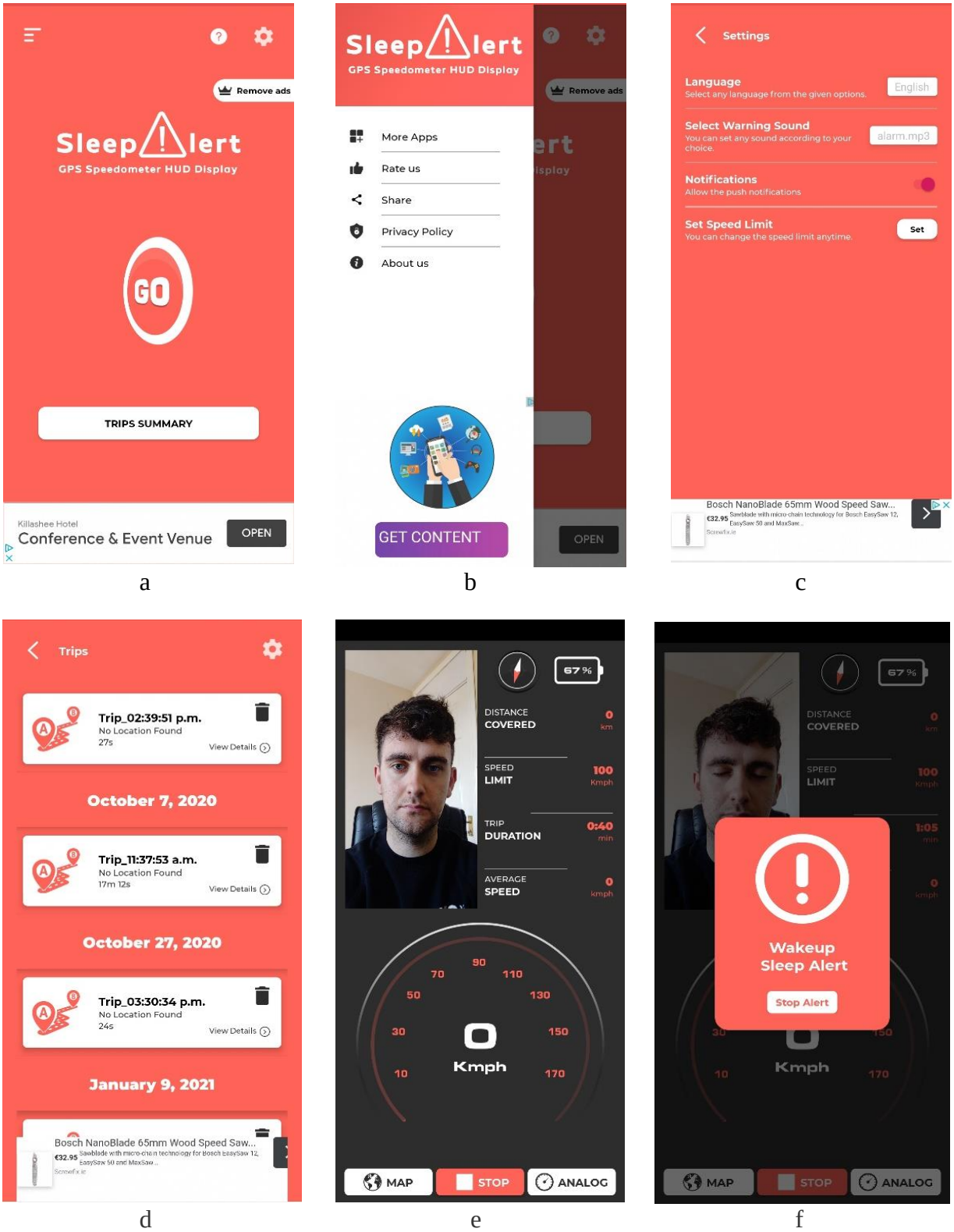
2.3 Sleep Alert & GPS Speedometer Car Heads up Display

2.3.1 Overview

Sleep Alert & GPS Speedometer Car Heads up Display is an Android application that is used to keep a user alert on long-distance journeys [9]. The application has the same intended purpose as this project.

2.3.2 Functionality

On opening the application, a user is presented with a splash screen that leads to the main application dashboard (See *Figure 3a*). On this screen, there is a number of options. In the top left corner, there is a navigation pane (See *Figure 3b*) which when opened, allows a user to see more of the developer's apps, rate the application, share the app, view the privacy policy and learn more about the developer. On the main screen there is also a settings icon, which when pressed gives users the option to change the app language, select a warning sound, turn on/off push notifications and set a speed limit (See *Figure 3c*). If a user presses the trips summary button, they are presented with a list of all their previous trips (See *Figure 3d*) and when a trip is clicked it will show the user their destination, trip duration, average speed and maximum speed. When the main "GO" button is clicked the user has the option to select between miles per hour or kilometres per hour and to view an analogue speedometer, a digital speedometer or a view of a map. Face monitoring then begins (See *Figure 3e*). As seen, there is a compass and battery indicator at the top of the screen. Next to this is a camera preview where the user can see themselves. There is information shown such as distance covered, the user's set speed limit, the trip duration and average speed. At the bottom, the user has the option to switch to the map or an analogue speedometer. There is a stop button to finish the face tracking. When a user's eyes are closed, a wake-up alert is shown, and the selected sound will play (See *Figure 3f).*



a b c

d e f

Figure 3: Screenshots of Sleep Alert app [9] – a, b, c, d, e, f

2.3.3 Discussion

Overall, this application is believed to be the best of the three applications tested. The application UI is modern and also simple to understand and navigate. The face and eye tracking work as intended. All other functionality works as described. A major pitfall of this application is that it is full of intrusive ads. These ads routinely pop up and get in the way of using the application. For this reason alone, this would be a major deterrent for the general public to use this app. The ads can be removed for €5, but for an application that has lifesaving potential, the application should be free to use with minimal to no ads. Overall, the application has the same intended purpose as this project. This application is missing features such as being able to turn off the camera preview or use Google Maps turn-by-turn navigation.

Chapter 3: System Design and Specifications

In this section, the system design and specifications are discussed. The main components of the design process were user requirements, system specifications, and application page flow. Class diagrams, a sequence diagram, as well as an application flow chart, were created to describe the function of the system.

3.1 User Requirements

User requirements, often referred to as user needs, describe what the user does with the system, such as what activities users must be able to perform. The following user requirements have been curated for this system.

- Start face/eye tracking
- View recent trips
- Change relevant settings
- View the help page

3.2 System Specification

The system specifications describe the functional requirements that must be met to run the system. The user must have a mobile device using Android as its operating system (OS). The earliest compatible version is Android 8.0 (Application Programming Interface (API) level 26), with a recommended version of Android 11.0. The user also requires internet access and GPS (Global Positioning System) to be able to use Google Maps in conjunction with this app.

- OS: Android
- Version: 8+ (11.0 Recommended)
- Internet and GPS access (Only needed if the user wishes to use Google Maps)

3.3 Application Page Flow

The application page flow was designed to be user-friendly. There is a clear layout to each page of the application and each page has a clear purpose. The application flow diagram shows what pages are in the app and how they interact (See *Figure 4*).

- Splash Screen: Initial screen shown to the user on opening the app.
- Dashboard: From the dashboard, users can enter any of the other pages.
- Eye Tracking: The main aspect of the application where face/eye tracking is performed.

- Recent Trips: Users can view their three most recent trips with statistics for each trip.
- Settings: Users can change settings relating to the application.
- Help: On this screen, the functions of each setting are explained to the user.

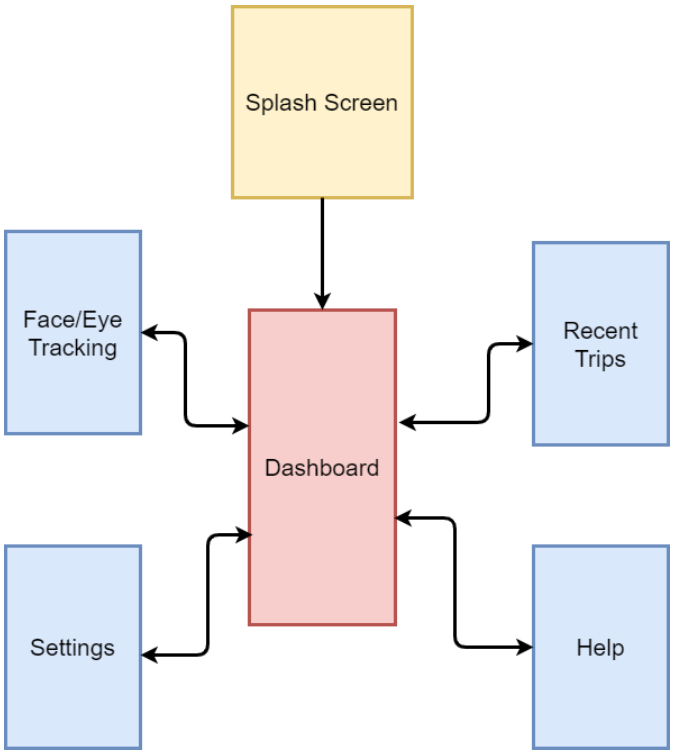


Figure 4: Application page flow diagram

3.4 Necessary Hardware

- The system utilises the front-facing camera to detect fatigued or distracted drivers.
- The phone needs to be sufficiently charged or be charging due to the screen being on constantly.
- GPS is used to incorporate Google Maps into the application.
- A cellular connection is used for sending SMS message as well as utilising a 3G/4G connection for Google Maps.
- The phone's speaker is used to sound alerts to the driver.
- The phone's display is used to show visual alerts as well as to allow overall interaction with the app.

The system is coded using Android Studio as the IDE, Java as the coding language, XML to create the GUI (Graphical user interface) and Google Mobile Vision is used to detect faces and track eyes.

3.5 Class Diagrams

This section gives an example of just one of many class diagrams that were used to aid the production of the application. A class diagram is a static diagram. It describes the attributes and operations of a class and also the constraints imposed on the system. A class diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In *Figure 5*, the different strings and textviews etc., are defined. A "+" symbol indicates that a variable is public and a "–" symbol indicates that a variable is private. The methods used throughout the class are detailed such as *loadContact(), saveContact()* and *requestPermissions().*

A number of class diagrams were created and can be seen in *Appendix D: Class Diagrams*.

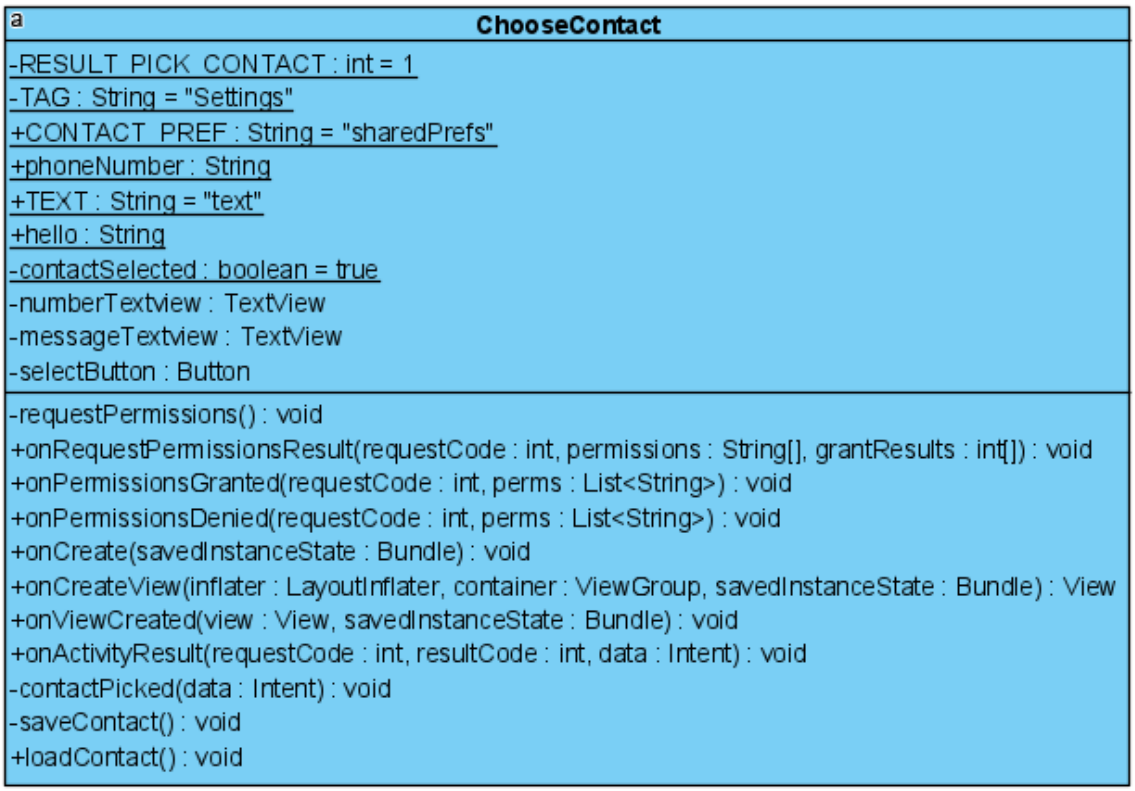


Figure 5: Class diagram for ChooseContact

3.6 Sequence Diagram

A sequence diagram simply depicts the interaction between objects in sequential order i.e., the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function. The sequence diagram shown in *Figure 6* shows the different sequences that occur to trigger a yellow alert and a red alert.

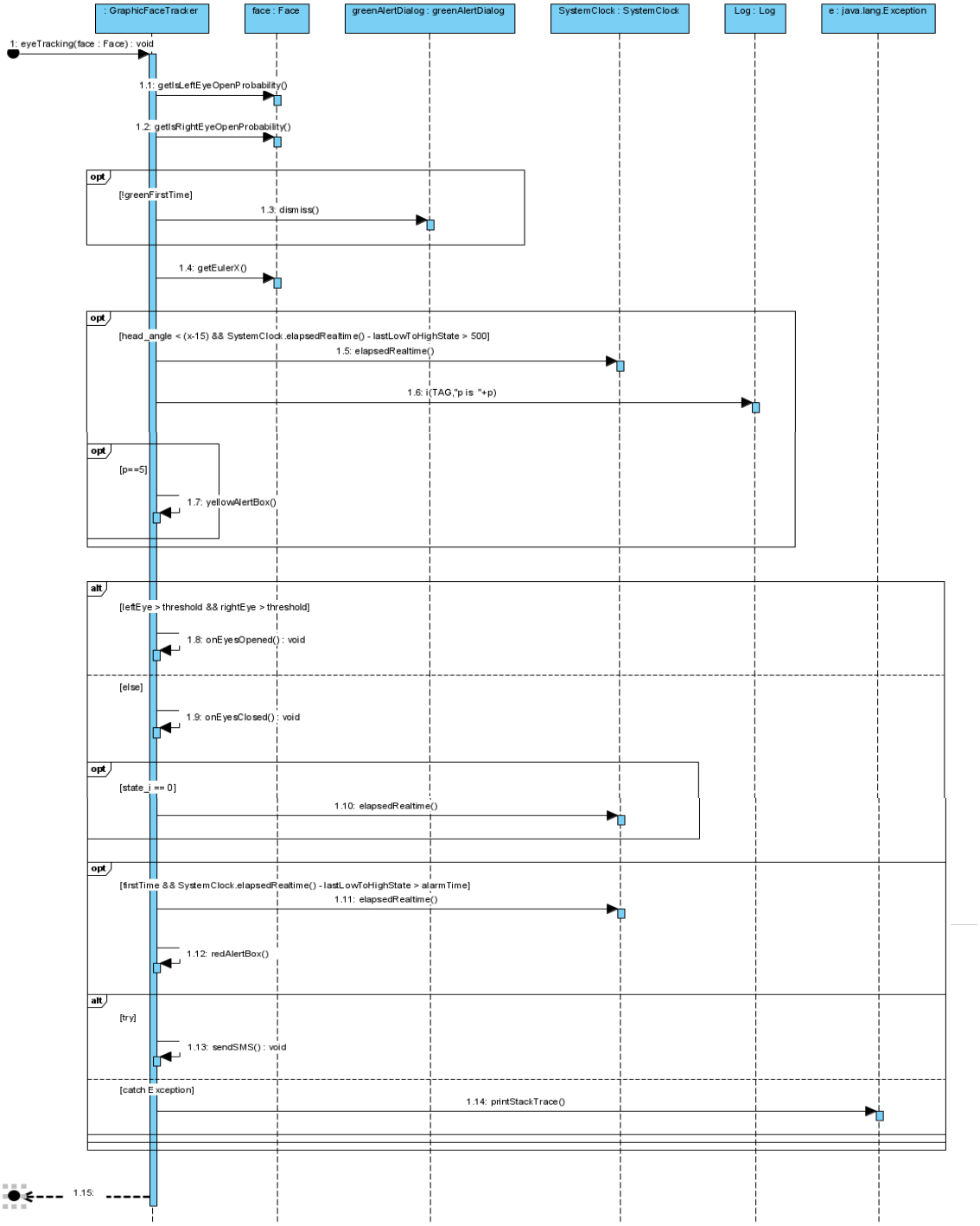


Figure 6: Sequence diagram for various alerts

3.7 Application Flowchart

Seen in *Figure 7* below is the overall application flow chart. This explains the operation of the application. There is a large difference between this and the preliminary flow chart seen on *page 101*.

When the application is opened, the splash screen is displayed. This is displayed for three seconds and then the dashboard is presented to the user. From the dashboard, a user can view their recent trips, enter the settings menu, view the help page or start the eye and face tracking. In the settings menu, a user can adjust the alarm delay time, choose an alarm tone and select an emergency contact.

If a user selects to start tracking, firstly a permission check is done. If a permission has not been accepted, the application checks if any permission has been permanently denied. If a permission has been permanently denied, the user will be brought to the OS settings to manually accept the permission(s). If a permission has not been permanently denied, the application will request the permission(s) from the user again. If all necessary permissions have been granted, the camera preview will be displayed, the journey timer will start, the application will begin to receive location updates and the eye/face tracking will start.

The application will then check to see if a face has been detected. If a face has been detected and then goes missing, the application will wait two seconds and check to see if the face is still missing. If it is, a green alert will be triggered. If not, the application proceeds to do a number of tasks simultaneously.

The application first checks if the user's eyes are closed. If they are, the application begins to count to the user-selected amount of alarm delay time i.e 2, 3 or 4 seconds. If the user's eyes are still closed a red alert will be triggered and an SMS message will be sent to the emergency contact. If the eyes are not closed, the value of a variable that holds the user's number of blinks will increase by one.

In addition to checking if the user's eyes are closed, the application will continually count up to one minute. If after one minute, a user has not blinked more than five times, a yellow alert will be triggered and a list of nearby petrol stations will be displayed to the user so they can pull over to get a coffee. If a user has blinked more than five times in a minute, the counter will reset and will begin to check again for the next minute. Finally, the application is also continually checking the angle of the user's head. If the application detects that the user's head is dipping, a variable *dips* will increase by 1. If *dips* is equal to five, a yellow alert will be triggered and a list of nearby petrol stations will be displayed.

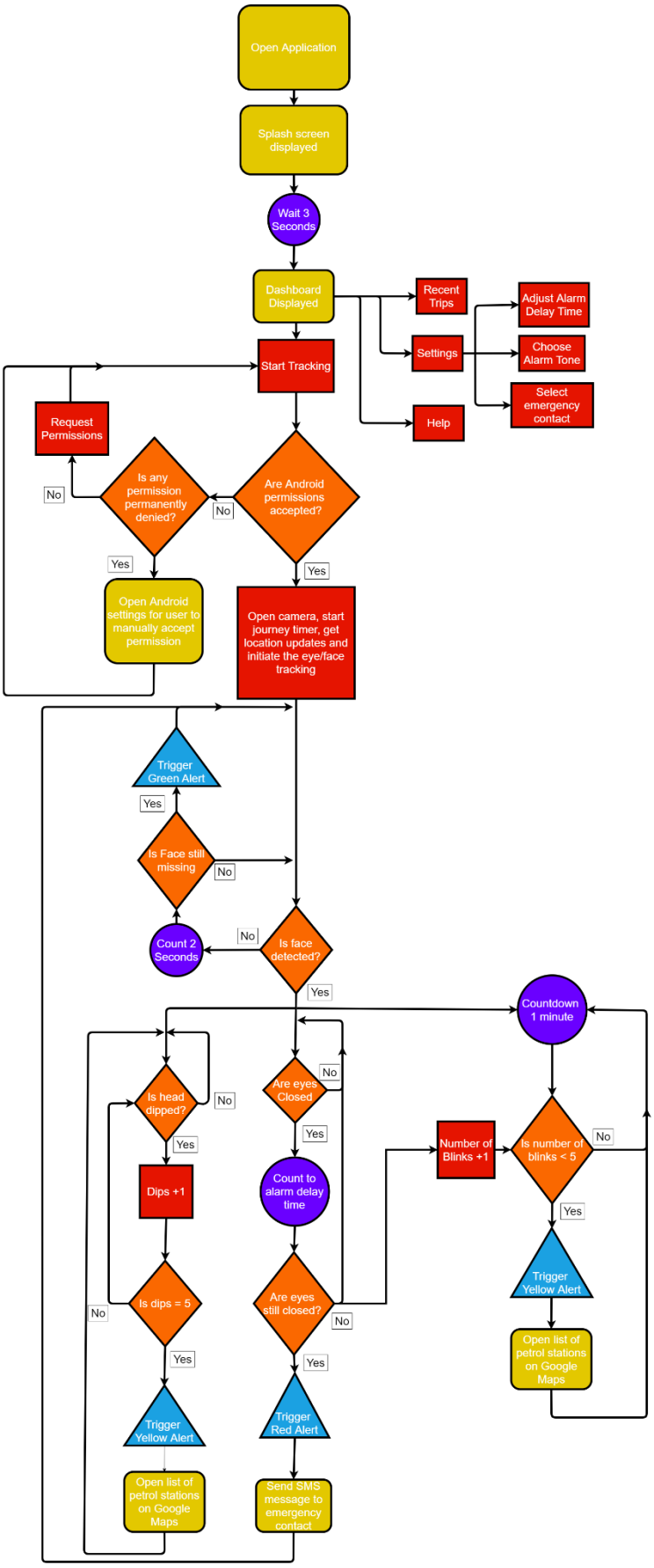


Figure 7: Overall application flow chart

Chapter 4: Technology and Programming Languages

4.1 Android Studio:

Android Studio is the official IDE for Android app development, based on IntelliJ IDEA [10]. Android Studio comes with the Android Studio software development kit (SDK), which provides tools for developers such as libraries, a phone emulator, debugger and documentation.

Android Studio uses a Gradle-based build system, has a built-in emulator, code templates and has GitHub integration for large scale app development. Gradle is an open-source build automation tool that is designed to be flexible enough to build almost any type of software [11]. Gradle is useful for tasks such as downloading dependencies, compiling source code into executable code, packing that executable code and generating an Android Package Kit (APK) [12].

Android Studio includes features such as Android app modules, library modules, and Google app engine modules. Android Studio uses an instant push feature to push code and resource changes to a running application. There is a built-in code editor that assists a developer, offering code completion, suggested changes, and refraction.

4.1.1 Emulator

The Android Emulator simulates Android devices on a computer so that a developer can test their application on a variety of devices and API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. A developer can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate the rotation of the screen, access the Google Play Store, and more [13].

The emulator comes with predefined configurations for various Android phones, tablets, Wear OS, and Android TV devices. The emulator is especially useful for being able to test layout configurations on different sizes of screens to ensure functionality across a range of screen sizes and resolutions.

4.1.2 APK Analyzer

Android Studio includes an APK Analyzer that provides immediate insight into the composition of an APK after the build process completes. Using the APK Analyzer can reduce the time a developer spends debugging issues with Dalvik Executable (DEX) files and resources within their app. It can also help to reduce the APK size [14].

With the APK Analyzer, a developer can accomplish the following:

- APK size optimisation: It can show the raw file size (the unzipped size on disk) and the download size (the estimated compressed size). The list of files and folders is sorted by total size in descending order.
- View the AndroidManifest.xml file in XML form instead of binary: The manifest file in the APK is usually a binary file. Furthermore, if the project includes more than one manifest file in the APK they will be merged together in a single manifest file. With the APK Analyzer it's possible to reconstruct the XML form of the manifest file.
- Understand the composition of the DEX files*:* When a developer clicks on any DEX file, they will see a summary of how many classes and methods it defines, and how many total references it contains.
- Compare APK files: It can be used to show the difference between two different APK files for example between debug and release builds.

4.1.3 Code Editor

Android Studio contains a code editor based on IntelliJ's code editor. The editor window is the section of the IDE in which a developer can create and modify code. It provides three types of code completion, namely basic completion, smart completion and statement completion. The code editor can provide code completion for many languages, including Kotlin, Java, XML, and C/C++. Android Studio also provides sample code. The Code Sample Browser in Android Studio helps a developer to find high-quality, Google-provided Android code samples based on the currently highlighted symbol in the project. As a developer edits, Android Studio automatically applies formatting and styles as specified in the developer's code style settings. One can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines [10].

4.1.4 Visual Layout Editor

The Android Studio Layout Editor enables a developer to quickly build layouts by dragging UI elements such as buttons and ImageViews into a visual design editor instead of writing layout XML manually. The design editor can preview an apps layout on different Android devices and versions, and it allows a developer to dynamically resize the layout to be sure it works well on different screen sizes [15].

4.1.5 Profilers

The Android Profiler is available in Android Studio 3.0 and higher. The Android Profiler tools provide real-time data to help a developer to understand how their app uses CPU (Central processing unit), memory, network, and battery resources. The Android Profiler is compatible with Android 5.0 (API level 21) and higher. Profiler data can be saved as sessions, which are retained until Android Studio is exited. By recording profiling information in multiple sessions and switching between them, a developer can compare resource usage in various scenarios [16].

4.1.6 Build System

The Android build system compiles app resources and source code and packages them into APKs that a developer can test, deploy, sign, and distribute. Android Studio uses Gradle to automate and manage the build process while allowing a developer to define flexible custom build configurations

The build process for a typical Android app module (See *Figure 8*) follows these general steps:

The compilers convert the source code into DEX files, which include the bytecode that runs on Android devices.

The APK Packager combines the DEX files and compiled resources into a single APK. Before an app can be installed and deployed onto an Android device the APK must be signed.

The APK Packager signs an APK using either the debug or release keystore. If a developer is building a debug version of an app, that is an app they intend only for testing and profiling, the packager signs the app with the debug keystore. Android Studio automatically configures new projects with a debug keystore.
If a developer is building a release version of an app that they intend to release externally, the packager signs the app with the release keystore. Before generating the final APK, the packager uses the zipalign tool to optimize the app to use less memory when running on a device. At the

end of the build process, a developer either has a debug APK or a release APK of their app that they can use to deploy, test, or release to external users [17].

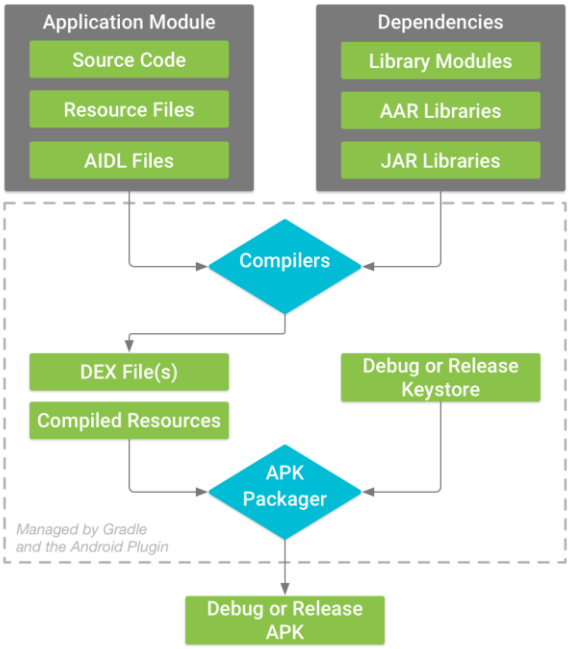


Figure 8: Android Studio build process [17]

4.2 Google Vision API

Google Vision is an API that allows developers to analyse the content of an image through extracted data. For this purpose, Google utilises machine learning models trained on a large dataset of images. All of that is available with a single API request. The engine behind the API classifies images, detects objects, people's faces, and recognises printed words within images.

Some of the main features of Google Vision are:

- Label Detection: Detect a set of categories within an image.
- Explicit Content Detection: Detect if there is explicit content (adult/violent) within an image.
- Landmark Detection: Detect natural and man-made structures within an image.
- Optical Character Recognition: Detect and extract text within an image. It can detect the language of the text.
- Face Detection: Detect multiple faces within an image, along with other attributes like emotional state or wearing headwear.

Google Vision provides the ability to perform face tracking which extends face detection to video sequences. Any face appearing in a video for any length of time can be tracked. Included in face detection is the ability to find landmarks detected on a face. [18]

A landmark is a point of interest within a face. The left eye, right eye, and nose base are all examples of landmarks (See *Figure 9*). The Face API provides the ability to find 12 total landmarks on the face (See *Table 1*).

Landmark
BOTTOM_MOUTH
LEFT_CHEEK
LEFT_EAR
LEFT_EAR_TIP
LEFT_EYE
LEFT_MOUTH
NOSE_BASE
RIGHT_CHEEK
RIGHT_EAR
RIGHT_EAR_TIP
RIGHT_EYE
RIGHT_MOUTH

Table 1: Table of available landmarks [19]

Figure 9: Example of detected landmarks [18]

In addition to landmark detection, Google Vision provides classification detections. Classification is determining whether a certain facial characteristic is present. For example, a face can be classified with regards to whether its eyes are open or closed. Another example is whether the face is smiling or not. The Android Face API currently supports two classifications: eyes open and smiling**.** The "eyes open" and "smiling" classification only work for frontal faces, that is, faces with a small Euler Y angle (at most about +/- 18 degrees) [18].

The Android Face API also provides the ability to determine a faces Orientation.
The face API detects faces at a range of different angles, as illustrated below (See *Figure 10*)

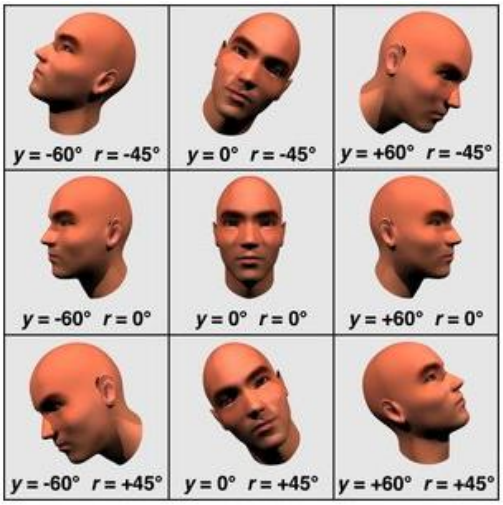


Figure 10: Pose angle examples where yEuler Y, rEuler Z [18]

The Face API provides a measurement of Euler Y, Euler Z and Euler X for detected faces. The Euler X and Z angles of the face are always reported. The Euler Y angle is available only when using the "accurate" mode setting of the face detector (as opposed to the "fast" mode setting, which takes some shortcuts to make detection faster).

The following table summarizes all of the landmarks that can be detected, for an associated face Euler Y angle (See *Table 2*).

Euler Y angle	Detectable Landmarks
< -36 degrees	Left eye, left mouth, left ear, nose base, left cheek
-36 degrees to -12 degrees	Left mouth, nose base, bottom mouth, Right eye, left eye, left cheek, left ear tip
-12 degrees to 12 degrees	Right eye, left eye, nose base, left cheek, right cheek, left mouth, right mouth, bottom mouth
12 degrees to 36 degrees	Right mouth, nose base, bottom mouth, left eye, right eye, right cheek, right ear tip
>36 degrees	Right eye, right mouth, right ear, nose base, right cheek

Table 2: Landmarks that can be detected, for an associated face Euler Y angle

4.3 Java

There are numerous ways to implement an Android app. Java is the official language for Android app development and as a result, is the most used language for producing Android apps. Most Google Play Store apps are created using Java. Java is a widely used language in modern-day programming and as such has a lot of online resources, tutorial videos and forums based around Java and app development. Java can be a difficult language to learn and use due to some complex topics such as constructors, null pointer exceptions, concurrency, inheritance etc.

4.4 XML

XML stands for Extensible Markup Language. XML is readable both by human and machine. It is also scalable and simple to develop. XML is the primary language used in Android Studio to create layout files. XML tags identify data and are used to store and organise data, rather than specifying how to display it. XML carries data, it does not present it. XML allows a developer to create their own self-descriptive tags, or language, that suits your application.

Chapter 5: System Implementation

In the following section, the implementation of the system design is discussed.

5.1 Application Files

5.1.1 SplashScreen

The first aspect of the application that was implemented was the *SplashScreen* activity (See *Figure 11*). On first opening the application, the user is presented with a splash screen, which shows them the logo of the application. The logo floats in from the top of the screen using an animation. The logo was imported to Android Studio using a tool called Batch Drawable Import. This tool allows for an image to be imported in multiple different sizes so that Android Studio can automatically scale the image to suit different screen sizes. The splash screen has a total time of 3 seconds from start to finish, then the Dashboard activity is shown. The splash screen was implemented using the following code (See *Code 1*)

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_splash_screen);

        //Animation
        topAnim = AnimationUtils.loadAnimation(this,
R.anim.top_animation);

        // The animationImage is found in the layout using findViewById.
        // This allows the value to be altered programmatically.*/
        image = findViewById(R.id.animationImage);
        image.setAnimation(topAnim);

        //Splash screen will be displayed for 3 seconds
        int SPLASH_SCREEN = 3000;

        //After the Splash Screen, the Dashboard will be displayed after 3
seconds
        new Handler().postDelayed(() -> {
            Intent intent = new Intent(SplashScreen.this,
Dashboard.class);
            startActivity(intent);
            finish();
        }, SPLASH_SCREEN);
    }
}
```

Code 1: Splash screen

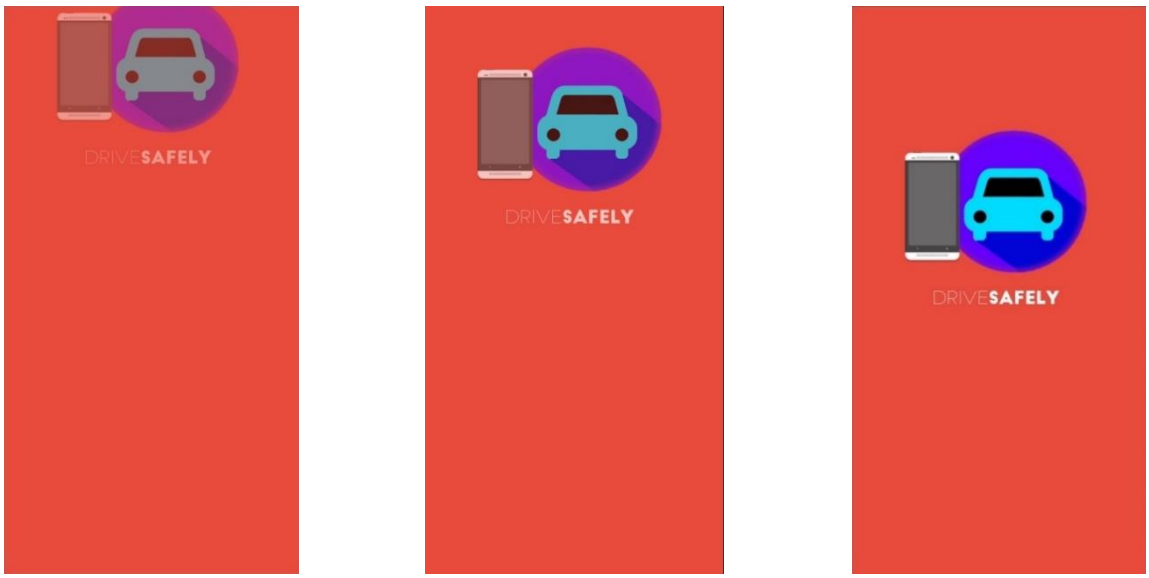


Figure 11: Splash screen animation

5.1.2 Dashboard

The dashboard is the main screen for interaction with the application. It contains four cards which when clicked brings the user into another activity. On the dashboard, there is the option to start the face tracking, view recent trips, view and change settings, and view the help screen. This is done via the following code (See *Code 2*).

```
public void onClick(View view) {
    Intent i;
    switch (view.getId()) {
        case R.id.start_tracking:
            i = new Intent(this, EyeTracking.class);
            startActivity(i);
            break;
        case R.id.recent_trips:
            i = new Intent(this, RecentTrips.class);
            startActivity(i);
            break;
        case R.id.settings:
            i = new Intent(this, Settings.class);
            startActivity(i);
            break;
        case R.id.help:
            i = new Intent(this, Help.class);
            startActivity(i);
            break;
        default:
            break;
    }
}
```

Code 2: Dashboard code

The Dashboard activity makes use of a Constraint Layout, like all other files in the application. It is especially important that the dashboard is shown correctly to the user so that they can navigate the app effectively. There is a number of icons, text and layouts within the dashboard layout. It is an intentionally simple layout to ensure that the application is usable by a wide range of users with varying mobile competency. The layout of the dashboard can be seen below (See *Figure 12*).

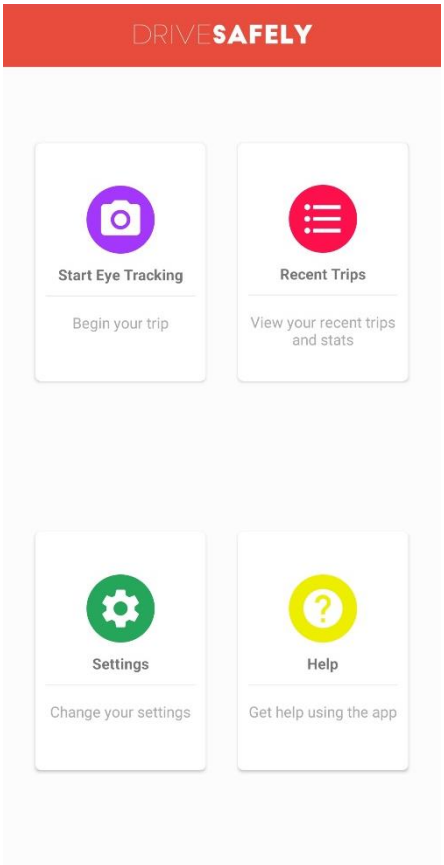


Figure 12: Dashboard layout

5.1.3 Help

The help screen gives the user an explanation of the different settings available in the *Settings* activity. It simply inflates the XML layout for the file (See *Code 3*).

```
public class Help extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_help);
    }
}
```

Code 3: Help Screen

On the help screen, there is a number of images and cards displayed. The user is given an explanation of what the alarm delay slider, alarm tone selector and contact selector do (See *Figure 13*).

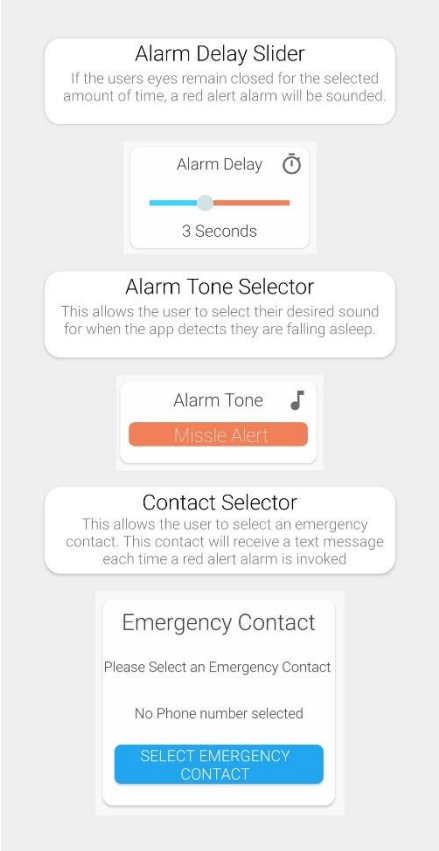


Figure 13: Help screen layout

5.1.4 Settings

The settings screen (See *Figure 14*) is where the user is able to adjust the amount of delay before the red alert alarm is sounded (i.e if 3 seconds is chosen, the user's eye's will have to be closed for 3 consecutive seconds before the alarm is sounded). The user is also able to choose from one of three different alarm tones. The user can also choose an emergency contact that will receive an SMS message if a red alert is sounded.

This activity makes use of SharedPreferences to save the values of each setting, for when the activity/app is destroyed and restored. On opening the settings activity, the app loads the values stored in the SharedPreferences (See *Code 4*). To select their desired alarm delay time, the user slides their finger on a SeekBar. The SeekBar has three options, being 2, 3 or 4 seconds. Once the value of the previously saved time is loaded, the progress of the SeekBar is set appropriately (See *Code 5*).

```
loadTimeData();
loadToneData();
timeTextView.setText(time + " Seconds");
```

Code 4: Load time and tone data

```
switch (time) {
    case 2:
        Progress = 0;
        break;
    case 3:
        Progress = 1;
        break;
    case 4:
        Progress = 2;
        break;
}
sBar.setProgress(Progress);
```

Code 5: Set SeekBar progress

The ability to choose an emergency contact is done using a fragment. A fragment is an independent Android component that can be used by an activity. A fragment encapsulates functionality so that it is easier to reuse within activities and layouts. The *ChooseContact* fragment is loaded into the settings page with the following code (See *Code 6*). The *ChooseContact* fragment code is detailed in *5.1.5 ChooseContact*.

```
Fragment fragment;
fragment = new ChooseContact();
FragmentManager fm = getSupportFragmentManager();
FragmentTransaction ft = fm.beginTransaction();
ft.add(R.id.contactlist, fragment).commit();
```

Code 6: ChooseContact fragment

To enable the user to choose their desired alarm tone, the use of a spinner is employed. There are three options available: Missile Alert, Railroad and Police Siren. The value of `Position` was previously loaded (See *Code 4*). The ArrayList and ArrayAdapter needed to implement the ability to change the tone is done via the code below (See *Code 7*). The previously selected tone is displayed on the spinner using the loaded value of `Position.` If no value has been previously set, the default tone will be set to “Missile Alert” which corresponds to a position of 0.

```
toneSpinner = findViewById(R.id.spinner);
String[] value = {"Missile Alert", "Railroad", "Police Siren"};
ArrayList<String> arrayList = new ArrayList<>(Arrays.asList(value));
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(this,
R.layout.style_spinner, arrayList);
toneSpinner.setAdapter(arrayAdapter);
toneSpinner.setSelection(Position);
```

Code 7: Creating tone spinner and setting previously selected tone.

An *OnItemSelectedListener* was used to allow the user to select different tones. The following code was used to implement this (See *Code 8*). The tone changes only when the user clicks on a corresponding selection from the ArrayList.

```
toneSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
        if (position == 0) {
            tone = 0;
        }
        if (position == 1) {
            tone = 1;
        }
        if (position == 2) {
            tone = 2;
        }
        Position = position;
    }
```

Code 8: Set desired alarm tone for selected spinner item

An *OnSeekBarChangeListener* was used to allow the user to select the different values for alarm delay. The following code was used to implement this (See *Code 9*).

```
sBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {

        time = progress + 2;
        Progress = progress;
        timeTextView.setText(time + " Seconds");
    }
```

Code 9: Set desired alarm delay time for certain seek bar progress

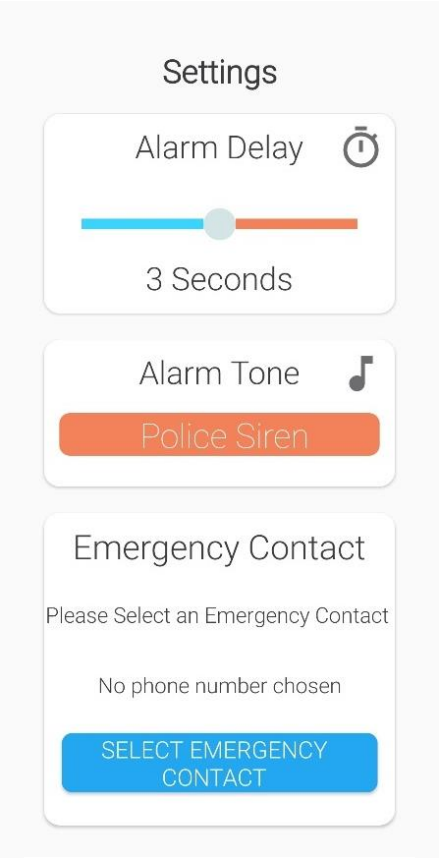


Figure 14: Settings screen

5.1.5 ChooseContact

The ChooseContact fragment gives the user the ability to select a contact number that will be used by the app as the emergency contact number which will receive an SMS message in the event that a red alert alarm is triggered.

On opening the settings activity, the ChooseContact fragment is added to the activity (See *Code 6*).

Firstly, when a user selects the button to choose an emergency contact, a permission check is done (See *Code 10*) (See *Figure 15a*). If the user has not yet accepted the necessary permission (READ_CONTACTS), then the user will have to accept the permission to proceed. The permission check is implemented using the EasyPermissions library [20]. EasyPermissions is a wrapper library to simplify basic system permissions logic when targeting Android Marshmallow (Android 6.0) or higher. If the required permission has previously been denied a dialog box is shown telling the user that permissions are needed to use the app (See *Figure 15b*). If the user presses “OK”, they are returned to the dashboard.

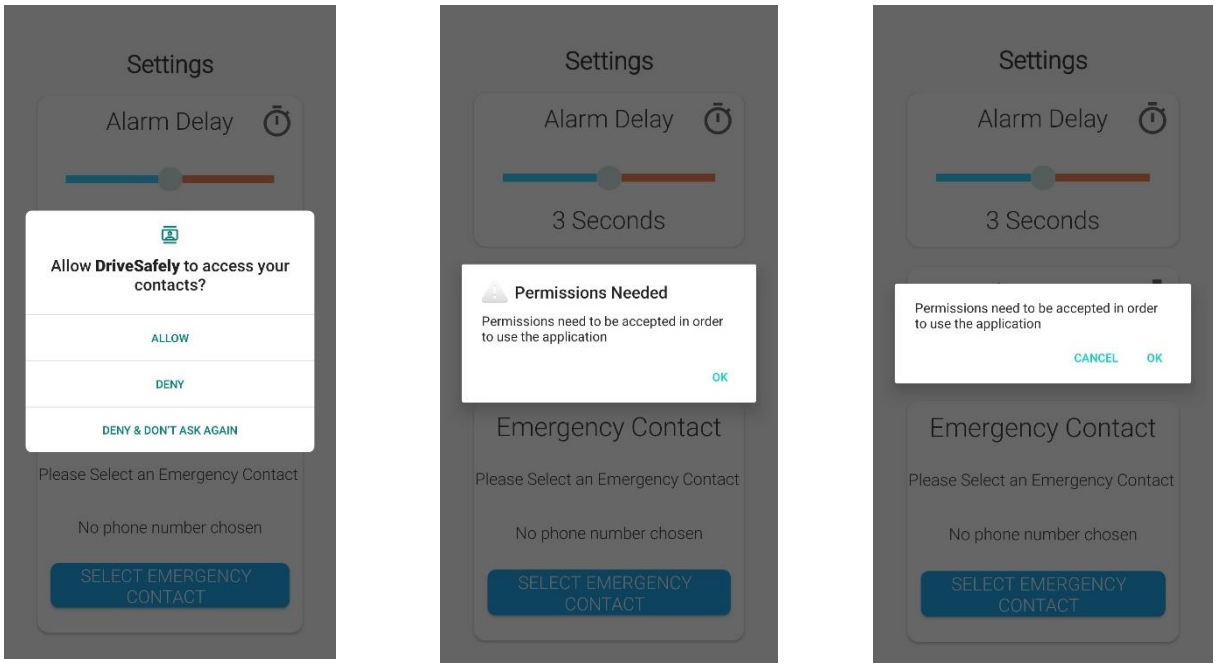
```
@AfterPermissionGranted(123)
private void requestPermissions() {
    String[] perms = {Manifest.permission.READ_CONTACTS};
    if (EasyPermissions.hasPermissions(getContext(), perms)) {
        Intent in = new Intent(Intent.ACTION_PICK,
ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
        startActivityForResult(in, RESULT_PICK_CONTACT);
    } else {
        EasyPermissions.requestPermissions(this, "Permissions need to be "
+
                        "accepted in order to use the application",
                123, perms);
    }
}
```

Code 10: Check for READ_CONTACTS permission

If the required permission is not granted, and the user tries to pick a contact again, a dialog box is shown to the user, explaining to them that the permission is needed in order to use the application (See *Code 11*) (See *Figure 15c*). If the user clicks OK, the permission is requested again (See *Figure 15a*). If the user click cancel, they remain on the settings screen.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    EasyPermissions.onRequestPermissionsResult(requestCode, permissions,
grantResults, this);

    if (EasyPermissions.somePermissionDenied(this, permissions)) {

        new AlertDialog.Builder(getActivity())
                .setTitle("Permissions Needed")
                .setMessage("Permissions need to be accepted in order to
use the application")
                .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which)
{
  //The user is returned to the dashboard screen on clicking OK
                        getActivity().finish();
                    }
                }).setIcon(android.R.drawable.ic_dialog_alert)
                .show();
        };
    }
```

Code 11: Determine what should be done if the permission is denied



a b c

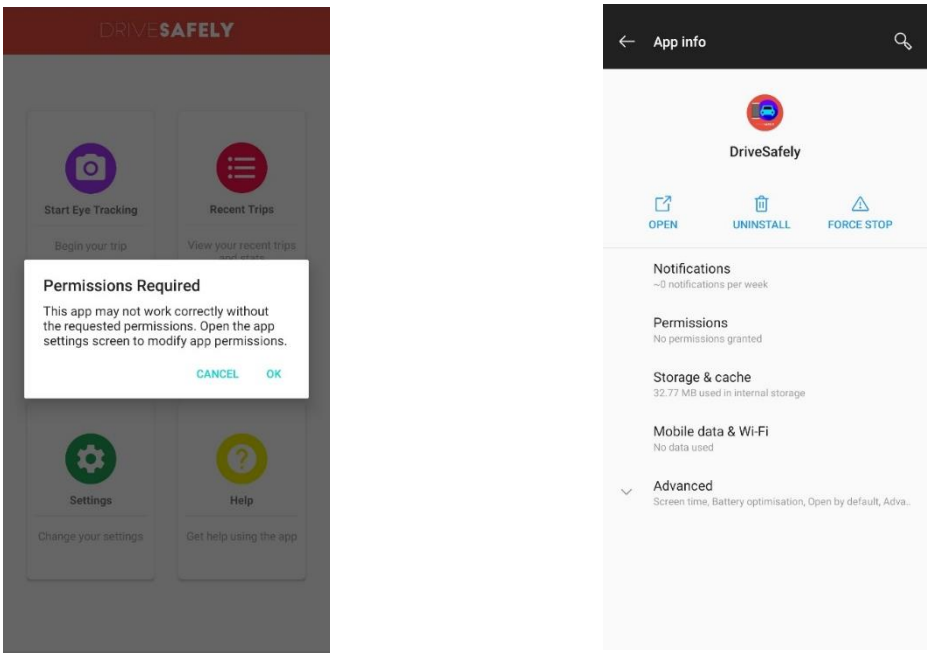
Figure 15: Permission process for READ_CONTACTS permission – a, b, c

If a user permanently denies the permission, they cannot use the application. This is to ensure full application functionality. When a permission is permanantly denied, the user is given the option to enter their phones OS settings to manually allow the required permission (See *Code 12*) (See *Figure 16a*).

On clicking OK, the user is brought to the application settings, which is in the general OS settings. From here, the user can manually accept the permissions (See *Figure 16b*). On returning to the application after accepting, the application will work as intended.

```
//Method used if some permission is permanently denied.
@Override
public void onPermissionsDenied(int requestCode, @NonNull List<String>
perms) {
     /* If some permission is permanently denied,
       the user is brought to their phone settings, where they can
        manually accept the apps permissions*/
    if (EasyPermissions.somePermissionPermanentlyDenied(this, perms)) {
        getActivity().finish();
        new AppSettingsDialog.Builder(this).build().show();
    }
}
```

Code 12: Code for permanently denied permission



a b

Figure 16: Manual acceptance of permission check (a) Device OS application settings (b)

Provided the permission is accepted to select a contact, a new intent is made (See *Code 13*). A separate window opens, overlaying the entire screen (See *Figure 17*). From here, a user can scroll through their contacts list, or use the search function to search for a specific contact. On selecting a contact, the number is saved to the contact shared preference.

```
Intent in = new Intent(Intent.ACTION_PICK,
ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
startActivityForResult(in, RESULT_PICK_CONTACT);
```

Code 13: Intent to pick contact

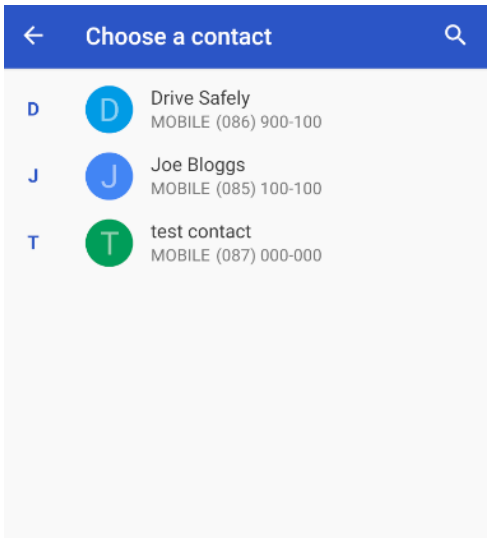


Figure 17: Choose a contact window

5.1.6 RecentTrips

The recent trips section of the application allows a user to see their past three trips' statistics. These statistics include the length of time their journey took, the distance they travelled in km, their average speed for the journey, and the number of drowsy driving detections counted on the journey.

The recent trips page consists of CardViews, TextViews and ImageViews (See *Figure 18*). The CardViews and TextViews are assigned to different variables so the values in the XML file can be changed programmatically.

The values for each statistic are assigned to the relative variable by using shared preferences. For a certain trip, the value of each statistic is saved to a corresponding shared preference. For trip one, the value of time is loaded as shown (See *Code 14*). The value of time is saved in milliseconds. To properly display it in hours: minutes: seconds, the time value is passed to the *stringFormatter* method (See *Code 15*). For trips two and three, the same code is used, replacing necessary variables with the respective trip's variables.

```
//SharedPreference to load value of time for trip one
SharedPreferences spTimeT1 =
getSharedPreferences(EyeTracking.TIME_TRIP_ONE_PREF, MODE_PRIVATE);
timeTrip = spTimeT1.getInt((EyeTracking.TIME_TRIP_ONE_PREF), 0);
String timeTripOne = stringFormatter(timeTrip);
timeTextOne.setText(timeTripOne);
```

Code 14: Value of time for trip one loaded to recent trips page

```
//Method used to properly format the time data
private String stringFormatter(int trip) {
    @SuppressLint("DefaultLocale")
    String hms = String.format("%02d:%02d:%02d",
            TimeUnit.MILLISECONDS.toHours(trip),
            TimeUnit.MILLISECONDS.toMinutes(trip) -
            TimeUnit.HOURS.toMinutes(TimeUnit.MILLISECONDS.toHours(trip)),
            TimeUnit.MILLISECONDS.toSeconds(trip) -
      TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(trip)));
return hms;
}
```

Code 15: String formatter method used to display time in hours:minutes:seconds

The value of detections for trip one is loaded to the recent trips page (See *Code 16*). For trips two and three, the same code is used, replacing necessary variables with the respective trip's variables.

```
SharedPreferences spDetect1 =
getSharedPreferences(EyeTracking.DETECT_TRIP_ONE_PREF, MODE_PRIVATE);
detectTrip = spDetect1.getInt((EyeTracking.DETECT_TRIP_ONE_PREF), 0);
detectTextOne.setText(String.valueOf(detectTrip));
```

Code 16: Value of detections for trip one loaded to recent trips page

The value of speed for trip one is loaded to the recent trips page (See *Code 17*). The average speed value is formatted to not include decimal places and display km/h after the value. For trips two and three, the same code is used, replacing necessary variables with the respective trip's variables.

```
SharedPreferences spSpeed1 =
getSharedPreferences(EyeTracking.SPEED_TRIP_ONE_PREF, MODE_PRIVATE);
speedTrip = spSpeed1.getFloat((EyeTracking.SPEED_TRIP_ONE_PREF), 0);
speedTextOne.setText(String.format("%.0f", speedTrip) + " km/h");
```

Code 17: Value of speed for trip one loaded to recent trips page

The value of distance for trip one is loaded to the recent trips page (See *Code 18*). The distance value is formatted to one decimal place. Again, for trips two and three, the same code is used, replacing necessary variables with the respective trip's variables.

```
SharedPreferences spDistance1 =
getSharedPreferences(EyeTracking.DISTANCE_TRIP_ONE_PREF, MODE_PRIVATE);
distanceTrip = spDistance1.getFloat((EyeTracking.DISTANCE_TRIP_ONE_PREF),
0);
distanceTextOne.setText(new DecimalFormat("#.#").format(distanceTrip) +
" km");
```

Code 18: Value of distance for trip one loaded to recent trips page

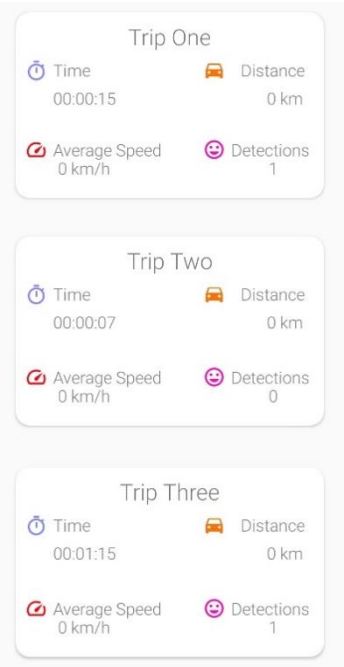


Figure 18: Recent trips screen

5.1.7 CameraSourcePreview

CameraSourcePreview is part of the sample code that Google provides as part of Google Vision API [21]. An adapted version of the file is used in this application. Its main purpose is to provide a surface holder with the correct dimensions for the camera preview to be housed in. It ensures that the correct camera aspect ratio is maintained. There is a number of different calculations carried out throughout the file to get the width and height of the current phone screen and scale the camera preview accordingly (See *Code 19*). In the *activity_eye_tracking* file, the *CameraSourcePreview* is added to the layout (Grey rounded rectangle) (See *Figure 19*).

```
if (widthRatio > heightRatio) {
    childWidth = viewWidth;
    childHeight = (int) ((float) previewHeight * widthRatio);
    childYOffset = (childHeight - viewHeight) / 2;
} else {
    childWidth = (int) ((float) previewWidth * heightRatio);
    childHeight = viewHeight;
    childXOffset = (childWidth - viewWidth) / 2;
}

for (int i = 0; i < getChildCount(); ++i) {

    getChildAt(i).layout(

 -1 * childXOffset, -1 * childYOffset,
            childWidth - childXOffset, childHeight - childYOffset);
}
```

Code 19: Fill the view with the camera preview, while preserving the correct aspect ratio

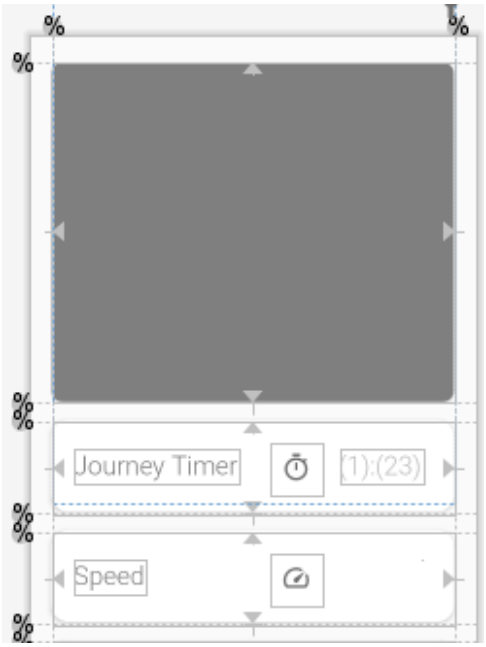


Figure 19: CameraSourcePreview surface holder in EyeTracking

5.1.8 GraphicOverlay

The *GraphicOverlay* class is also part of the sample code that Google provides as part of Google Vision API [22]. It is used in conjunction with *CameraSourcePreview* and *FaceGraphic*. Its purpose is to render a series of custom graphics to be overlayed on top of an associated preview (i.e., the camera preview). A developer can add graphic objects, update the objects, and remove them from the screen.

The file supports scaling and mirroring of the graphics, relative to the camera's preview properties. The idea is that detection items are expressed in terms of a preview size but need to be scaled up to the full-size view, and also mirrored in the case of the front-facing camera. Associated graphic items should use the following methods to convert to view coordinates for the graphics that are drawn: scaleX and scaleY adjust the size of the supplied value from the preview scale to the view scale. translateX and translateY adjust the coordinates from the preview's coordinate system to the views coordinate system.

The code below (See *Code 20*) is used to draw the graphic overlay and its associated objects onto the canvas. In the case of this application, the file *FaceGraphic* extends *GraphicOverlay* and it is used to draw the bounding box and landmarks on a user's face.

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    synchronized (mLock) {
        if ((mPreviewWidth != 0) && (mPreviewHeight != 0)) {
            mWidthScaleFactor = (float) canvas.getWidth() / (float)
mPreviewWidth;
            mHeightScaleFactor = (float) canvas.getHeight() / (float)
mPreviewHeight;
        }

        for (Graphic graphic : mGraphics) {
            graphic.draw(canvas);
        }
    }
}
```

Code 20: Draw graphic overlay and associated objects onto canvas

5.1.9 FaceGraphic

The *FaceGraphic* file is the file that draws the bounding box around the user's face (See *Figure 20*). It is also a part of the Google Vision sample code [23]. It has been adapted for this application. Firstly, the colour of the box and the size of the stroke etc is set (See *Code 21*).

```
//Define parameters for colour and style etc. of bounding box
mFacePositionPaint = new Paint();
mFacePositionPaint.setColor(selectedColor);
mBoxPaint = new Paint();
mBoxPaint.setColor(selectedColor);
mBoxPaint.setStyle(Paint.Style.STROKE);
mBoxPaint.setStrokeWidth(BOX_STROKE_WIDTH);
```

Code 21: Parameters set for bounding box styling

The draw method is used to calculate the position of the user's face within the camera frame. A circle is drawn at the detected centre of the face. A number of operations are carried out to calculate the area in which the face resides. With these calculated values, the bounding box is drawn. There is a list of seven different colours used. The colour changes each time a face is detected. The drawing of the circle and bounding box are implemented below (See *Code 22*).

```
public void draw(Canvas canvas) {
    Face face = mFace;

    if (face == null) {
        return;
    }
// Draws a circle at the position of the detected face, with the
face's track id below
    float x = translateX(face.getPosition().x + face.getWidth() / 2);
    float y = translateY(face.getPosition().y + face.getHeight() / 2);
    canvas.drawCircle(x, y, FACE_POSITION_RADIUS, mFacePositionPaint);

    // Draws a bounding box around the face.
    float xOffset = scaleX(face.getWidth() / 2.0f);
    float yOffset = scaleY(face.getHeight() / 2.0f);
    float left = x - xOffset;
    float top = y - yOffset;
    float right = x + xOffset;
    float bottom = y + yOffset;
    canvas.drawRect(left, top, right, bottom, mBoxPaint);

    Paint paint = new Paint();
    paint.setColor(Color.GREEN);
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(5);
```

Code 22: Calculations used to determine position of face

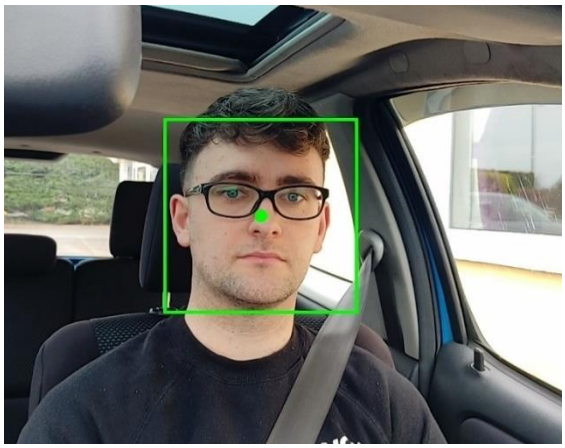


Figure 20: Bounding box drawn around the user's face

For debugging and report writing purposes, the detected facial landmarks were drawn on the camera preview. This was primarily done to debug issues when trying to detect yawning (Expanded in *5.2 Issues Faced During Implementation*). This was implemented by getting the X and Y coordinates of each landmark from *Table 1*. At the location of each landmark, a circle was drawn to illustrate each landmark on the face (See *Code 23*). This was done for each of the 12 landmarks. A screenshot of a face with the detected landmarks drawn can be seen below (See *Figure 21*).

```
int cLeftEyeX;
int cLeftEyeY;

cLeftEyeX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
4)).getPosition().x);
cLeftEyeY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
4)).getPosition().y);
canvas.drawCircle(cLeftEyeX, cLeftEyeY, 5, paint);
```

Code 23: Code used to draw landmarks on user's face

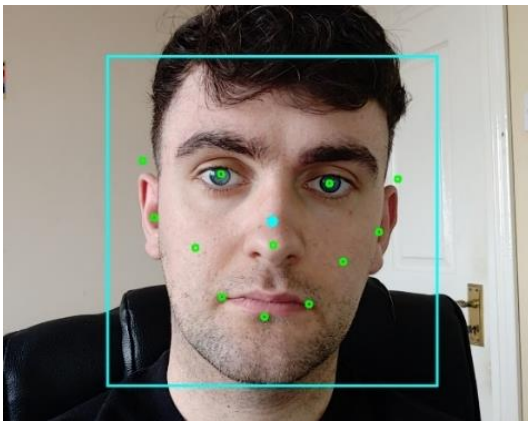


Figure 21: Landmarks drawn on face

5.1.10 EyeTracking

The *EyeTracking* Java file is the main application file. In this file, the user's eyes and face are tracked to detect tiredness/drowsiness. Depending on the severity of tiredness, a certain alarm will be triggered. Also, in this file, the location tracking will commence, and a user can also open Google Maps and overlay the application on top of it.

When the activity is first opened, the user is requested to accept any necessary permissions. The permissions needed to use the application are: Camera, Send_SMS, Access_Fine_Location, Internet, Vibrate and Read_Contacts. The same process as seen in *5.1.5 ChooseContact* is used to request the listed permissions. Provided all permissions are accepted, a method is called to show a dialog box to the user, to show them how to position the phone (See *Figure 22*). The current trip value is loaded also. This is used to save the correct trip statistics into the correct shared preferences for the recent trips section of the application (See *5.1.6 RecentTrips*). The two methods are called via the code below (See *Code 24*).

```
//Run methods shown
showPositionDialog();
loadTripValue();
```

Code 24: Methods called on activity starting

As seen below, the user is instructed to place the phone relatively close to eye level and in a position that is close to directly in front of the user. For the application to accurately classify a user's eyes as being open or closed, the users face must have a small Euler Y angle, at most about ± 18 degrees [18].

Figure 22: Phone positioning instructions

The *SpeedometerFragment* is added to the layout next. It is added to the layout using a fragment manager and fragment transaction. This is used to show the real-time speed on the screen. It is added to the layout as seen below (See *Code 25*)

```
//Start the SpeedometerFragment and add its view to fragment_speedo in
activity_eye_tracking
fragment = new SpeedometerFragment();
fm = getSupportFragmentManager();
ft = fm.beginTransaction();
ft.replace(R.id.fragmentSpeedo, fragment).commit();
```

Code 25: SpeedometerFragment added to layout

Next, the journey timer is started. This is done by creating a chronometer, initialising it and starting it. The initialisation and starting is done via the below code (See *Code 26*)

```
//Initialise chronometer and start
journeyTimer.setBase(SystemClock.elapsedRealtime());
journeyTimer.start();
```

Code 26: Journey timer initialised and started

The values for alarm delay time and the desired users alarm tone are loaded from the shared preferences for each, stored in the *Settings* file (See *5.1.4 Settings*). If a user has started the app without first altering the settings, a default value of 3 seconds for alarm tone delay is chosen. Equally, the default value for alarm tone is 0 which corresponds to the "Missile Alert" alarm tone. The values are loaded as seen below (See *Code 27*).

```
// Get the value stored in Settings.TIME_PREF . If no value found, default
is 3 seconds
SharedPreferences sharedPreferencesTime =
getSharedPreferences(Settings.TIME_PREF, MODE_PRIVATE);
alarmTime = sharedPreferencesTime.getInt(Settings.TIME_PREF, 3) * 1000;
Log.i(TAG, "Alarm time is: " + alarmTime);

// Get the value stored in Settings.TONE_PREF . If no value found, default
is tone 0 - Missile Alert
SharedPreferences sharedPreferencesTone =
getSharedPreferences(Settings.TONE_PREF, MODE_PRIVATE);
alarmTone = sharedPreferencesTone.getInt(Settings.TONE_PREF, 0);
Log.i(TAG, "Alarm tone is: " + alarmTone);
```

Code 27: Alarm time and alarm tone values loaded from shared preferences

Likewise, the emergency contact number is attempted to be loaded from the *ChooseContact* file (See *Code 28*). It is encased within a try and catch statement to ensure that if no contact number has been chosen, the application does not crash.

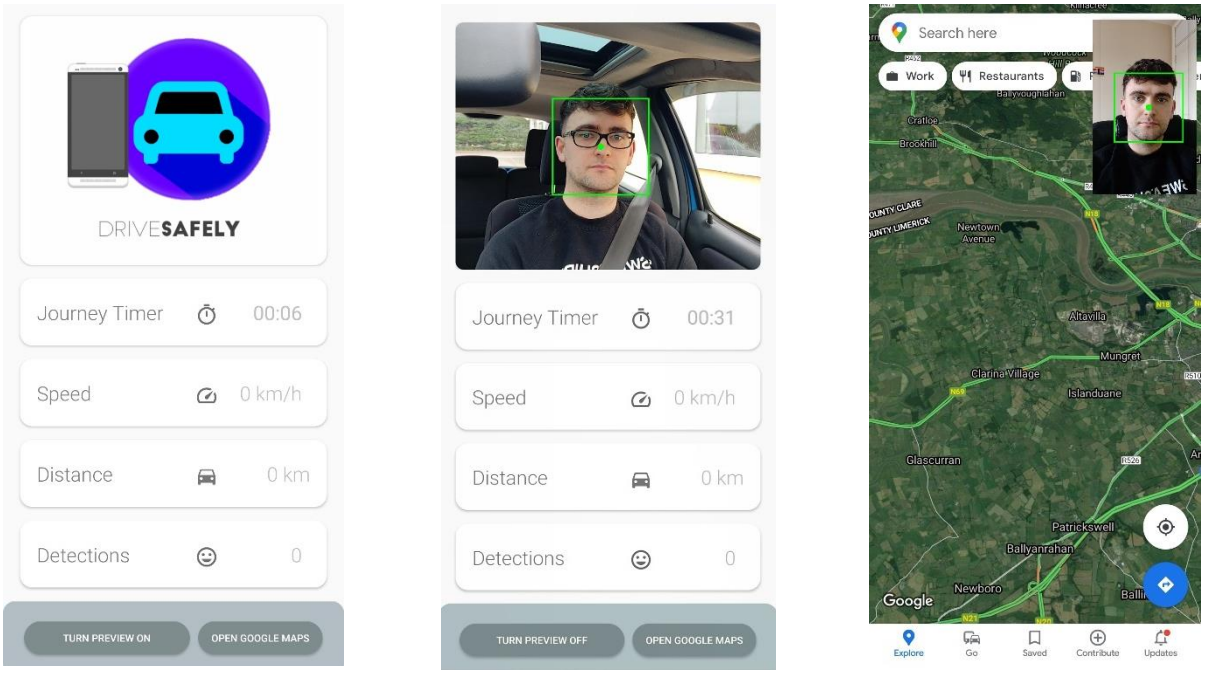
```
//Attempt to get the chosen emergency contact. If no contact chosen.
exception is thrown
try {
    SharedPreferences sharedPreferencesContact =
getSharedPreferences(ChooseContact.CONTACT_PREF, MODE_PRIVATE);
    emergencyContact =
sharedPreferencesContact.getString(ChooseContact.CONTACT_PREF, "");

    // For logging and testing purposes
    Log.i(TAG, "Number is  " + emergencyContact);
} catch (Exception e) {
    e.printStackTrace();
}
```

Code 28: Code to load emergency contact number

At the bottom of the screen, there is a button to turn off the camera preview. This hides the preview from the user in the event they find it distracting (See *Figure 23a*). There is a button at the bottom of the screen to open Google maps (See *Figure 23b*). When this button is clicked, a method called *startPIP* is called. This method enters the application into Picture-in-picture mode. PIP is a special type of multi-window mode mostly used for video playback. It lets the user watch a video in a small window pinned to a corner of the screen while navigating between apps or browsing content on the main screen [24]. In the case of this application, PIP is used to overlay the camera preview in a small window on top of the main Google Maps screen (See *Figure 23c*). When the *openMapsBtn* is clicked, *startPIP()* is called and Google Maps is opened at the user's current location (See *Code 29*). Once Google Maps is opened, the application works as normal, tracking the user's face and eyes. The user can use Google Maps as normal, being able to search destinations and use turn-by-turn navigation.

```
//Start Google Maps in picture-in-picture mode when openMapsBtn is clicked
openMapsBtn.setOnClickListener(v -> {
    startPIP();
    Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse("http://maps.google.com/maps?"));
    intent.setPackage("com.google.android.apps.maps");
    startActivity(intent);
});
```

Code 29: openMapsBtn OnClickListener code



a b c

Figure 23: Camera preview hidden (a) EyeTracking activity (b) Activity in PIP mode (c)

The next method is used to check the number of times a user blinks in one minute (See *Code 30*). A timer and timer task are used to check for this. If the user's face is present, the timer is started. If the user blinks less than five times in a minute, the *yellowAlertBox()* method is called. Every minute, the number of counted blinks is reset to zero and the timer task will reset and check how many blinks are counted in the subsequent minute.

```
//Method to start checking if a user is blinking enough
private void checkForBlinks() {

    /*This method is used to trigger a yellow alert is the number of
blinks in a minute is less than 5*/
    yellowAlertTimeTask = new TimerTask() {
        @Override
        public void run() {
            if (facePresent) {
                yellowAlertTimer = new Timer();
                if (numbOfBlinks < 5 && yellowAlert && !isOnPause) {
                    yellowAlert = false;
                    yellowAlertBox();
                }
            }
            numbOfBlinks = 0;
        }
    };
    yellowAlertTimer.schedule(yellowAlertTimeTask, 60* 1000, 60 * 1000);
}
```

Code 30: Method used to check the number of blinks in a given minute

A method is created to update the Euler X value stored in a variable *updateHeadAngle*. On *page 62*, the value of *updateHeadAngle* is compared against the value of *head_angle* to determine if a user's head has dipped multiple times. A timer and timer task are used again to carry out the updating process, every three seconds. It is implemented with the following code (See *Code 31*).

```
//Method to assign a new value to updateHeadAngle every 3 seconds
private void calcHeadAngle() {
    Timer headAngleTimer= new Timer();

   TimerTask angleTimerTask = new TimerTask() {
        @Override
        public void run() {
            updateHeadAngle = head_angle;
        }
    };
    //The timer is scheduled to run every 3 seconds with an initial delay
of 3 seconds

    headAngleTimer.schedule(angleTimerTask, 3* 1000, 3 * 1000);
}
```

Code 31: Method used to update variable updateHeadAngle every three seconds

The face detector is initialised next. A new FaceDetector object called detector is created (See *Code 32*). The detector is set to detect all facial landmarks, use accurate mode, determine all classifications and only work on the most prominent face.

```
FaceDetector detector = new FaceDetector.Builder(context)
            .setTrackingEnabled(true)
            .setLandmarkType(FaceDetector.ALL_LANDMARKS)
            .setClassificationType(FaceDetector.ALL_CLASSIFICATIONS)
            .setMode(FaceDetector.ACCURATE_MODE)
            .setProminentFaceOnly(true)
            .build();
```

Code 32: Face detector builder

A new camera source *mCameraSource* is created with a resolution of 1600x1200 pixels. The front facing camera is selected. Frames are requested at 60fps and autofocus is enabled (See *Code 33*)

```
mCameraSource = new CameraSource.Builder(context, detector)
        .setRequestedPreviewSize(1600, 1200)
        .setFacing(CameraSource.CAMERA_FACING_FRONT)
        .setRequestedFps(60.0f)
        .setAutoFocusEnabled(true)
        .build();
```

Code 33: Camera source parameters set

In order to send SMS messages to the users chosen emergency contact, the use of the built-in Android *SmsManager* is used. The emergency contact is passed to the *SmsManager* along with a message that will be sent to the emergency contact. This is implemented via the below code (See *Code 34*).

```
// Method used to send a message to the user's chosen emergency contact
private void sendSMS() {
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(emergencyContact, null,
getString(R.string.Emergency_message), null, null);
}
```

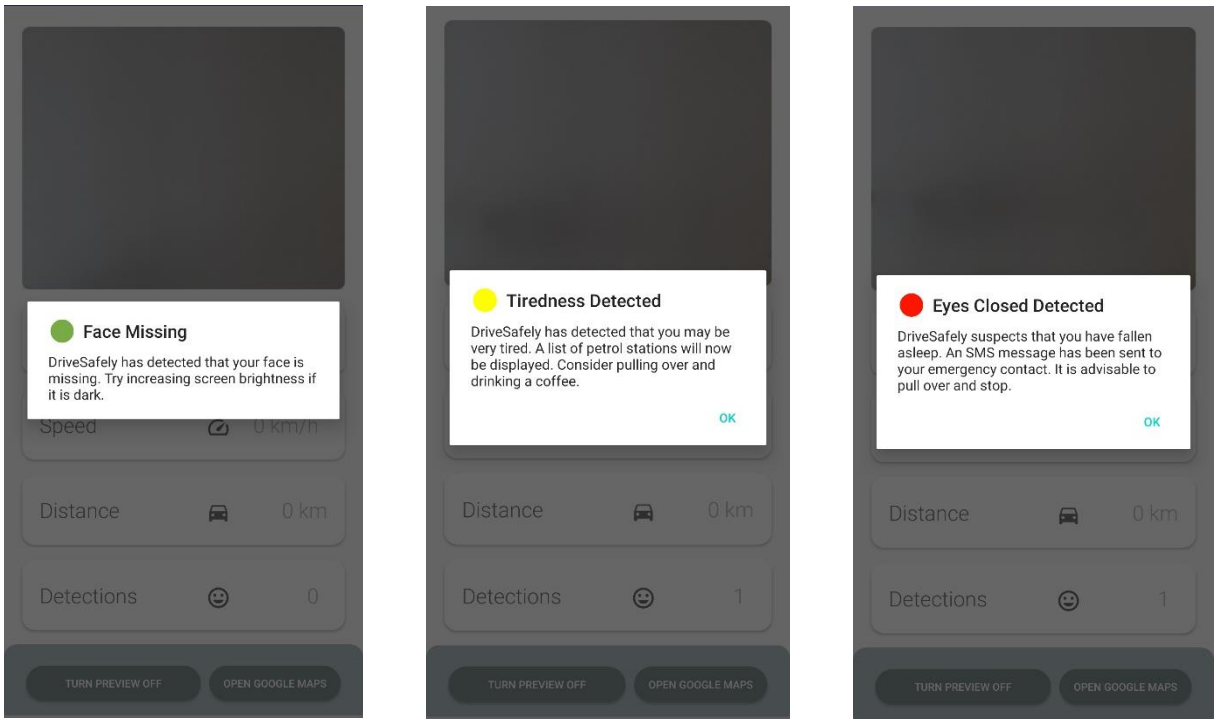
Code 34: Method used to send message to user's chosen emergency contact

There are three different types of alerts used within the application. They are as follows:

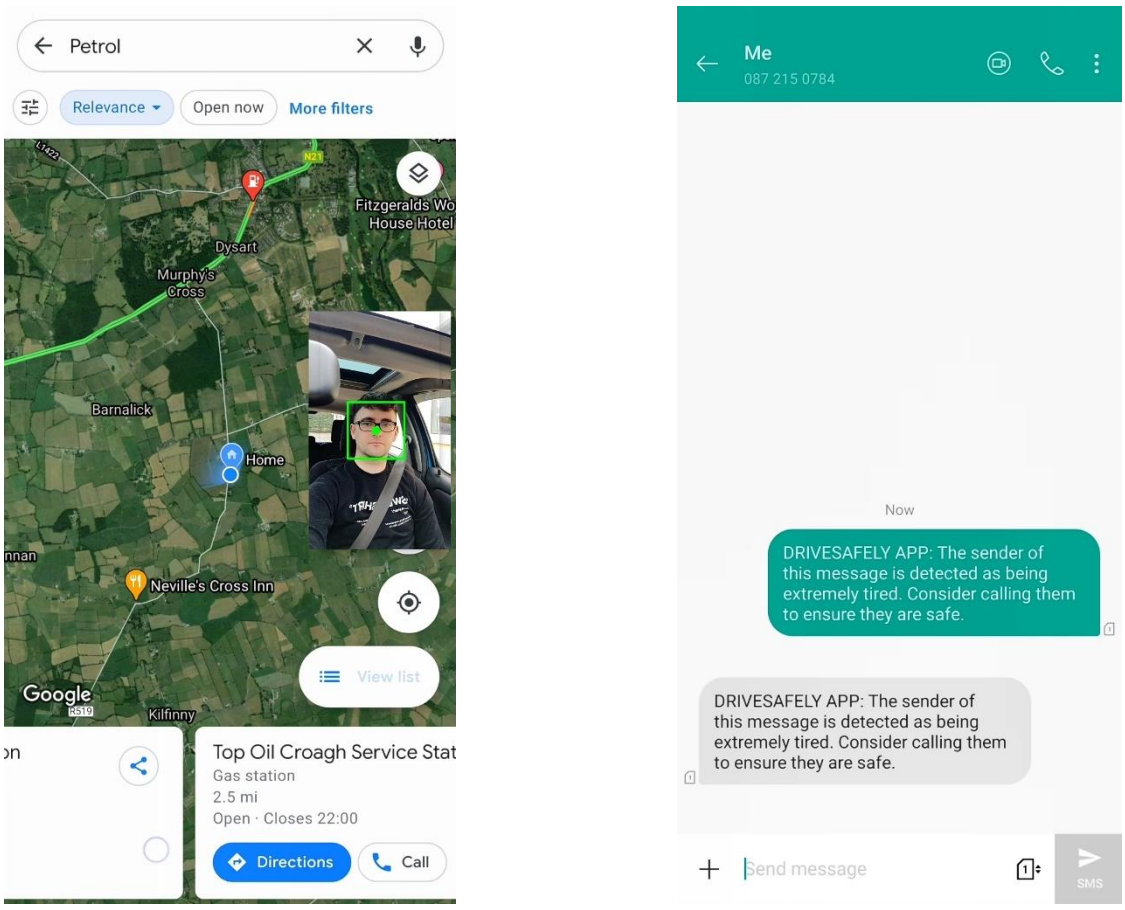
- Green alert – Triggered when the user's face has initially been detected but goes missing for more than two seconds. No audio or phone vibration is used. A dialog box is shown, indicating that the user's face is missing and to increase their phone brightness if necessary (See *Figure 24a*). The dialog box is automatically removed once a user's face re-enters the camera frame.

- Yellow alert – This alert is triggered when a user does not blink five times or more in any given minute OR if the user's head dips a total of five times. A police siren sound is played, and phone vibration is used. A dialog box is shown, stating to the user that the application has detected that they are drowsy and should consider pulling over and getting a coffee (See *Figure 24b*). They are informed that the application will open Google Maps and show a list of nearby petrol stations. On dismissing the alert, the application enters Picture-in-picture mode, and Google Maps opens, showing a list of nearby petrol station to the user (See *Figure 24d*). The user can then choose to get directions to any given petrol station. The application will continue to monitor the user as normal in PIP mode.

- Red alert – The final and most severe alert is triggered when the application detects that the user has closed their eyes for their chosen amount of alarm delay time or longer (2, 3 or 4 seconds, chosen in Settings (See *5.1.4 Settings*)). A dialog box is shown indicating to the user that they have been detected as having fallen asleep (See *Figure 24c*). They are informed that an SMS message has been sent to their emergency contact if they chose one. The SMS message is shown below (See *Figure 24e*).



a b c



d e

Figure 24: Green alert (a) Yellow alert (b) Red alert (c) List of petrol stations (d) Emergency text message (e)

The final and main method of the application is *eyeTracking*. This method is used to calculate the probability that the user's eyes are closed, get the angle of their head and call any method to trigger alerts and send SMS messages etc.

The below code (See *Code 35*) is used to get the probability of each eye being open. A value between 0.0 and 1.0 is returned giving a probability of each eye being open. 0.0 being an eye is certainly closed and 1.0 being that an eye is certainly open. The green dialog box is removed when a face is detected, and the Euler X angle is stored in the variable *head_angle*.

```
// Method used to carry out operations based on the user's eyes
    private void eyeTracking(Face face) {

        float leftEye = face.getIsLeftEyeOpenProbability();
        float rightEye = face.getIsRightEyeOpenProbability();

        // Removes the greenAlertDialog is condition is false
        if (!greenFirstTime) {
            greenAlertDialog.dismiss();
            greenFirstTime = true;
        }

        head_angle = face.getEulerX();
```

Code 35: Get probability that eyes are open, remove dialog box and get Euler X angle

The if statement below (See *Code 36*) is used to compare the value of *head_angle* to *updateHeadAngle* (See *Code 31*) minus 15 degrees. If *head_angle* is less than that value for more than 500ms, a variable *dips* is incremented by one. If dips is equal to 5, a yellow alert is triggered. The value of dips is then reset back to 0.

```
if (head_angle < (updateHeadAngle - 15) && SystemClock.elapsedRealtime() -
lastLowToHighState > 500) {
    dips++;
    // Used to prevent the function from continuously firing
    lastLowToHighState = SystemClock.elapsedRealtime();
    if (dips == 5) {
        yellowAlertBox();
        dips = 0;
    }
}
```

Code 36: If statement used to determine if a user's head is dipping/nodding

A threshold value of 0.2 is set. An if statement is used to check if both the left and right eyes are above the threshold i.e. The user's eyes are considered to be open if the probability that their eyes are open is above 0.2. An else statement is used to carry out operations if the probability that a user's eyes are open, is less than 0.2. i.e., a user's eyes are less than or equal to 20% open. The inverse of this is the probability that the user's eyes are closed is greater than or equal to 80%. This is defined by:

$$P(EyeClosed) = 1 - Probability\ eyes\ are\ open$$

Figure 25: Equation to calculate probability that a user's eyes are closed

If a user's eyes are determined to be closed for a period of time longer than *alarmTime*, then the red alert method is called, and a try catch statement is used to try send an emergency SMS if an emergency contact has been picked. *onEyesOpened()* and *onEyesClosed()* methods are used to count the number of times a user blinks (See *Code 37*).

```
        float threshold = 0.2f;
        // Checks if left or right eyes is considered open
        if (leftEye > threshold && rightEye > threshold) {
            state_i = 0;
            firstTime = true;
            facePresent = true;

           onEyesOpened();
         }
         else {
            onEyesClosed();
            if (state_i == 0) {
                lastLowToHighState = SystemClock.elapsedRealtime();
            }
            if (firstTime && SystemClock.elapsedRealtime() -
lastLowToHighState > alarmTime) {

                // Used to prevent the function from continuously firing
                lastLowToHighState = SystemClock.elapsedRealtime();
                firstTime = false;
                redAlertBox();
                try {
                    sendSMS();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                flag = 1;
            }
            state_i = 1;
        }
    }
}
```

Code 37: Determine if eyes are open or closed and carry out operations

5.1.11 SpeedometerFragment

This fragment is used to display and calculate the speed that the user is travelling and also the distance that they have travelled. Location requests are setup with high accuracy and fast interval refresh times. A location manager *lm* is setup and initialised. Location updates are requested (See *Code 38*).

```
//Method used to get the setup the location requests and initialise the
location manager
private void SetupLocation() {

    //Location request parameters are set and high accuracy is requested
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(INTERVAL);
    mLocationRequest.setFastestInterval(FASTEST_INTERVAL);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    //Location updates requested
    lm = (LocationManager)
getActivity().getSystemService(Context.LOCATION_SERVICE);
    lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
    }
```

Code 38: Location requests setup and location manager initialised

The *onLocationChanged* method is used to assign variables to current and most recent locations. The method *updateUI()* is called every time the user's location changes (See *Code 39*).

```
public void onLocationChanged(Location location) {

    //Used to calculate distance
    mCurrentLocation = location;
    if (lStart == null) {
        lStart = mCurrentLocation;
    }
    lEnd = mCurrentLocation;
    updateUI();
}
```

Code 39: Method to calculate distance travelled

The values of distance and speed are updated in the *updateUI()* method. Below is an extract from the method (See *Code 40*). Distance is calculated in metres, so it is divided by 1000 to convert to km and speed is measured in m/s, so it is multiplied by 3.6 to convert to km/h.

```
distance = distance + (lStart.distanceTo(lEnd) / 1000.00);
float nCurrentSpeed = mCurrentLocation.getSpeed() * 3.6f;
```

Code 40: Distance and speed updated

5.2 Issues Faced During Implementation

Throughout the development of the DriveSafely app, numerous issues were encountered during the implementation. The main problem that was encountered early on in the project was implementing turn-by-turn navigation within the application. The original goal of the application was for it to be fully self-contained i.e., a user could perform face/eye tracking while using turn-by-turn navigation from within the app itself. The use of the Mapbox API [25] was explored in detail. The turn-by-turn navigation was successfully implemented, however it would not work from within a fragment, after many days of trying to debug. Only a map could be displayed in a fragment. Also, to select a destination, a user would have to click the exact location they want to go to as opposed to being able to search for the destination. The addition of a search function within the Mapbox API was explored, but due to lack of coding experience and understanding meant that too much time would have been spent on this aspect of the project as opposed to focusing on the face and eye tracking. Also, even if the search function had been implemented, there were little destinations available on it. i.e., a user could not search for Eircode's or non-well-known locations. For this reason, it was determined to just revert to using Google Maps for the navigation. It was decided to use Picture-in-picture mode to allow the user to use Google Maps with the eye/face tracking overlaid, thus being unobtrusive to the user, while getting the benefits of Google Maps.

As seen below in *Figure 26*, the original design of the application is shown. Here the camera preview is displayed over the map using a fragment. No Picture-in-picture is used.

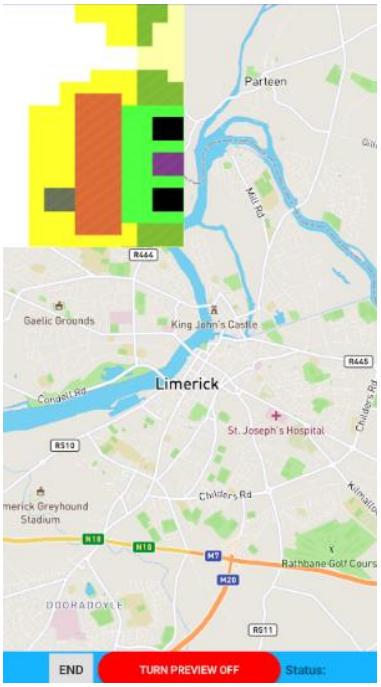


Figure 26: Original app layout with Mapbox map

Another issue faced during the implementation of the project was trying to detect yawning. Yawning was initially planned to be used to detect if a driver were about to fall asleep. This was going to be done by using the mouth landmarks detected on a user's face. Either the angle or the co-ordinate distance between two points was going to be used in order to calculate if a user's mouth was fully open or closed. Unfortunately, it was found through testing that when a user fully extended their mouth, the BOTTOM_LIP landmark behaved erratically. It would move into the middle of the user's mouth (See *Figure 27*) and then latch onto the bottom of the lip again. This meant that this method could not be used, and it was decided to not implement this into the application.

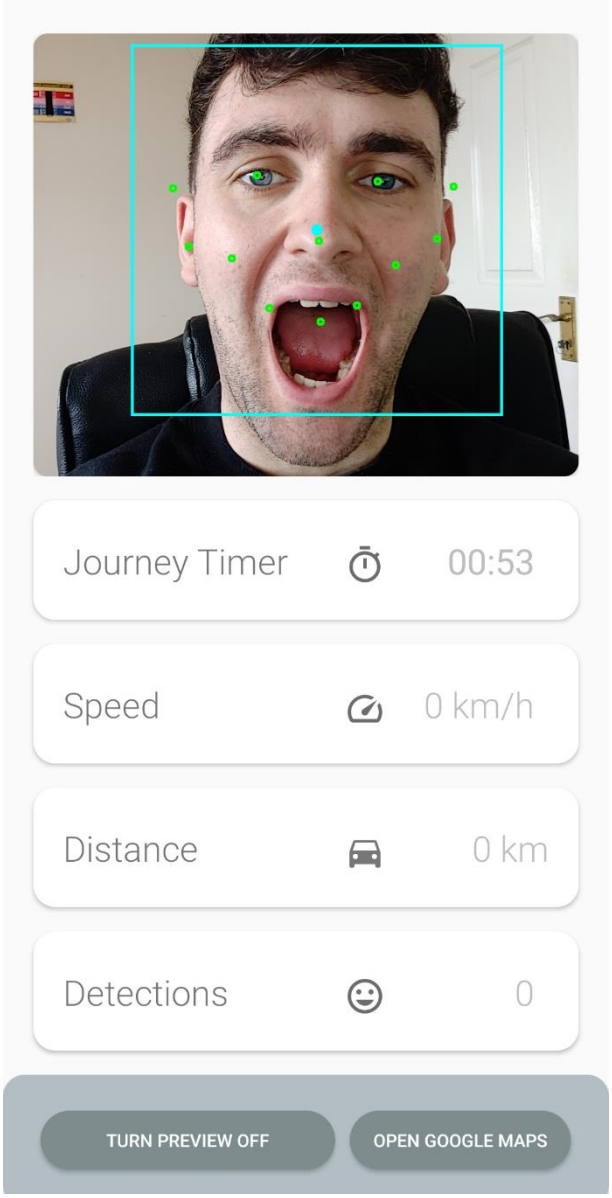


Figure 27: Inaccurate landmark detection

Chapter 6: Testing and Evaluation

In this section, the testing of the system is discussed in detail. Many aspects of the application have been assessed and evaluated. In the following subsections, the application layout is evaluated on different devices and emulators. Also, the functionality of different aspects of the application is thoroughly tested through a number of test cases.

6.1 Display

It was essential to test the display of the application on multiple display proportions to ensure that the application is correctly displayed on different devices. In this section, the effect of different screen resolutions on the application is evaluated. In *Figure 28a-e*, the face tracking page of the application is visible (See *Table 3* for device details).

Device	Resolution	Figure
OnePlus 6T (Physical device)	1080 x 2340	*Figure* 28*a*
Pixel XL (Emulator)	1440 x 2560	*Figure* 28*b*
OnePlus 5 (Physical device)	1080x 1920	*Figure* 28*c*
Pixel 3 XL (Emulator)	1440x2960	*Figure* 28*d*
Nexus S (Emulator)	480x800	*Figure* 28*e*

Table 3: Device resolutions

In *Figure 28a-d* the layout fits perfectly in each screen. However, the layout of the Eye Tracking screen is condensed in *Figure 28e* due to less screen resolution but the application is still usable on this device. The resolution does influence the user experience visually, but it does not have a big effect on the functionality of the application. It would be very uncommon for a phone to have such low resolution that can use Android 8.0+ as its operating system.

Another page of the application that is affected by the resolution of the device is the dashboard screen. However, this screen is still functional. Some text is missing from the CardViews to describe what each one does. The user can still access all four CardViews as normal. The dashboard is very unappealing on a low-resolution device (See *Figure 28f*) in comparison to the design viewed on a higher resolution (See *Figure 12*).

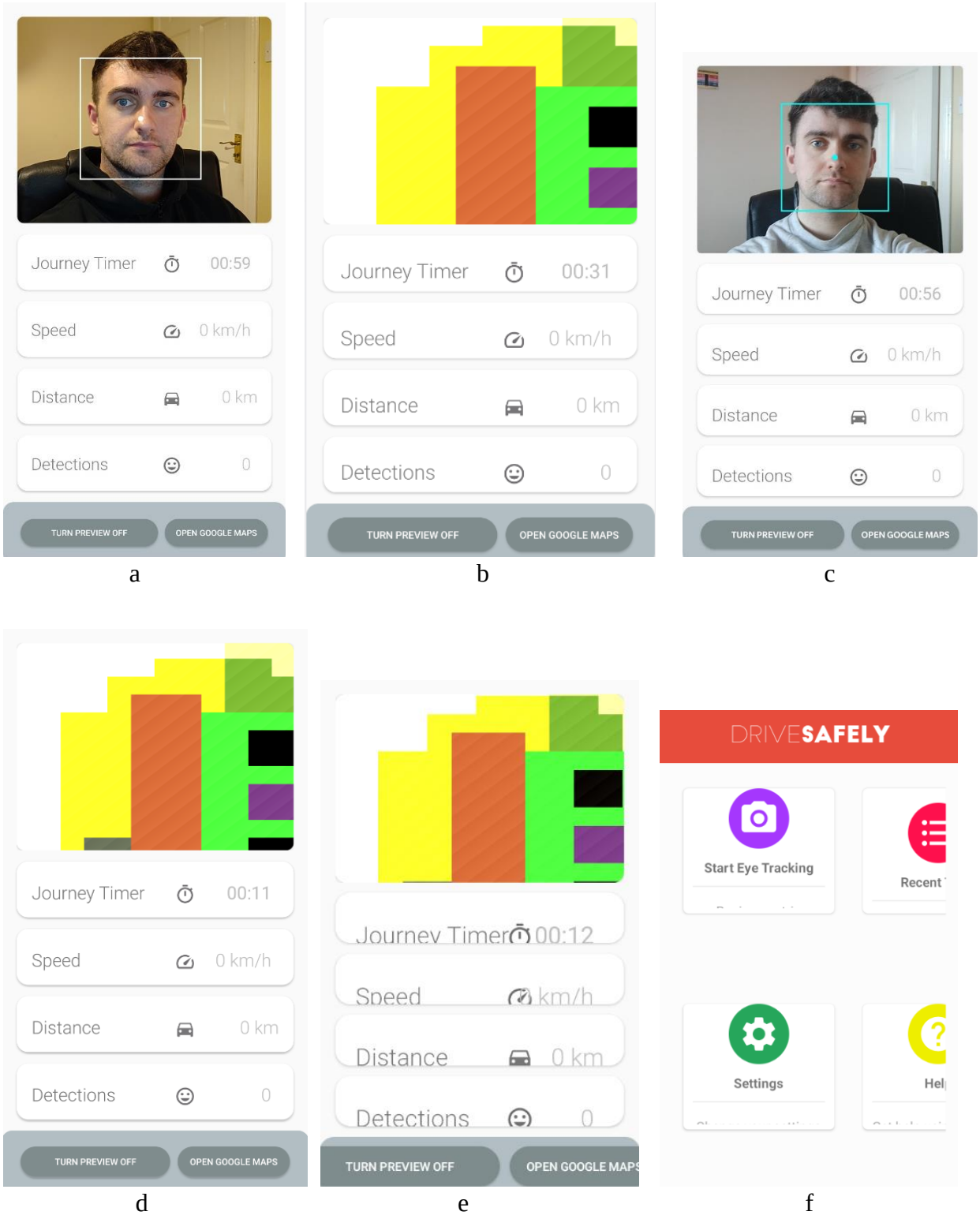


Figure 28: EyeTracking for different screen resolutions (a-e) Low resolution device (f)

6.2 Functionality

The application had to be tested thoroughly to ensure that it had the required functionality. Each activity was first evaluated separately. This involved assessing that basic UI elements worked as designed and that the settings and eye/face tracking work as intended.

6.2.1 SplashScreen

The splash screen is the first interaction that a user has with the application. A single test was implemented as described in *Table 4*. The test was successful, providing the expected results.

	Description	Expected Result	Actual Result	Pass/Fail
1	Open the application.	Logo descends from the top of the screen. The dashboard will be displayed after 3 seconds.	The logo descended correctly, and the dashboard was displayed after 3 seconds.	Pass.

Table 4: Splash screen test case

To test the *Dashboard* activity of the system, five tests were implemented. These tests are described in *Table 5*. All tests were successful, providing the expected results.

6.2.2 Dashboard

	Description	Expected Result	Actual Result	Pass/Fail
1	Press “Start Eye Tracking” button.	EyeTracking activity will start.	EyeTracking activity successfully started.	Pass.
2	Press “Recent Trips” button.	RecentTrips activity will start.	RecentTrips activity successfully started.	Pass.
3	Press “Settings” button.	Settings activity will start.	Settings activity successfully started.	Pass.
4	Press “Help” button.	Help activity will start.	Help activity successfully started.	Pass.
5	Press back button.	Application closes.	The application closed.	Pass.

Table 5: Dashboard test cases

To test the *Help* activity, two tests were implemented. These tests are described in *Table 6*. All tests were successful, providing the expected results.

6.2.3 Help

	Description	Expected Result	Actual Result	Pass/Fail
1	Enter help page.	Text and images should be displayed, giving information about what each setting does on the settings page.	The help screen and associated images/text were displayed correctly.	Pass.
2	Press the back button.	The dashboard screen will be shown.	The dashboard was shown.	Pass.

Table 6: Help screen test cases

To test the *Settings* activity, ten tests were implemented. These tests are described in *Table 7*. All tests were successful, providing the expected results.

6.2.4 Settings

	Description	Expected Result	Actual Result	Pass/ Fail
1	Change alarm delay slider to 2 seconds.	The slider will move to position zero and “2 Seconds” will be displayed.	The slider moved to position zero and “ 2 Seconds” was displayed.	Pass.
2	Change alarm delay slider to 3 seconds.	The slider will move to position one and “3 Seconds” will be displayed.	The slider moved to position one and “3 Seconds” was displayed.	Pass.
3	Change alarm delay slider to 4 seconds.	The slider will move to position two and “ 4 Seconds” will be displayed.	The slider moved to position two and “4 Seconds” was displayed.	Pass.
4	Select “Missile Alert” for alarm tone.	The array element selected will be zero.	The array element selected was zero. “Missile alert” was	Pass.

		"Missile alert" will be displayed as chosen tone.	displayed as the chosen tone.	
5	Select "Railroad" for alarm tone.	The array element selected will be one. "Railroad" will be displayed as chosen tone.	The array element selected was one. "Railroad" was displayed as the chosen tone.	Pass.
6	Select "Police Siren" for alarm tone.	The array element selected will be two. "Police Siren" will be displayed.	The array element selected was two. "Police Siren" was displayed as the chosen tone.	Pass.
7	Click "Select Emergency Contact" for the first time without previously accepting permissions.	A dialog box should appear asking the user to accept the "READ_CONTACTS" permission.	The dialog box appeared, and the permission could be accepted or denied.	Pass.
8	Click "Select Emergency Contact" having previously accepted permissions.	A list of the user's contacts should be displayed. Clicking a contact will select it is the emergency contact.	The list of contacts was displayed. Clicking a certain contact selected them as the emergency contact.	Pass.
9	Press back button with contacts list displayed.	A message saying "Failed to pick contact" will be displayed and the settings screen will be shown.	The message was displayed, and settings was shown again.	Pass.
10	Press back button on Settings Screen.	The currently selected settings should be saved to their relevant shared preferences. The application should return to the dashboard.	All settings were saved to their respective shared preferences. The application returned to the dashboard.	Pass.

Table 7: Settings test cases

To test the *RecentTrips* activity, five tests were implemented. These tests are described in *Table 8.* All tests were successful, providing the expected results.

6.2.5 RecentTrips

	Description	Expected Result	Actual Result	Pass/Fail
1	Start face and eye tracking, stop and then return to recent trips.	Trip one should show details of journey time, average speed, distance travelled and the number of detections for the trip just finished.	Trip one displayed all the correct information for the first trip.	Pass.
2	Start face and eye tracking a second time, stop and then return to recent trips.	Trip two should show details of journey time, average speed, distance travelled and the number of detections for the trip just finished.	Trip two displayed all the correct information for the second trip.	Pass.
3	Start face and eye tracking a third time, stop and then return to recent trips.	Trip three should show details of journey time, average speed, distance travelled and the number of detections for the trip just finished.	Trip three displayed all the correct information for the third trip.	Pass.
4	Start face and eye tracking a fourth time, stop and then return to recent trips.	Trip one details should be overwritten with the details of journey time, average speed, distance travelled and the number of detections for the trip just finished.	Trip one was overwritten successfully. The correct information was displayed.	Pass.
5	Press the back button.	The application should return to the dashboard.	The application returned to the dashboard.	Pass.

Table 8: Recent trips test cases

To test the *EyeTracking* activity, twenty-seven tests were implemented. This activity was rigorously tested throughout the development of the application to ensure that all aspects of the activity work correctly. These tests are described in *Table 9.* All tests were successful, providing the expected results.

6.2.6 EyeTracking

	Description	Expected Result	Actual Result	Pass/ Fail
1	Open Eye Tracking for the first time after installing the app.	The user should be prompted to accept permissions for contact access, camera access, location access, send and view SMS messages.	The user is prompted to accept permissions for contact access, camera access, location access, send and view SMS messages.	Pass.
2	User denies a single permission or more .	A dialog box will be displayed indicating informing the user that permissions are needed to use the application. The application will revert back to the dashboard.	A dialog box is displayed indicating that permissions are needed to use the application. The application reverted back to the dashboard.	Pass.
3	After denying permission, open Eye Tracking again.	A dialog box will be displayed telling the user than permissions are needed to use the app. Pressing ok will show another dialog box asking the user to press cancel or ok.	The dialog box is displayed telling the user permissions are needed to use the app. The 2nd dialog box was shown.	Pass.
4	Press cancel on the previous dialog.	The user will remain on the settings screen.	The user remained on the settings screen.	Pass.
5	Press OK on previous dialog	The application will request permissions again.	The application requested permissions again.	Pass.

6	User permanently denies permission (Clicks “Deny & Don’t ask again”).	A dialog box will be shown, informing the user that permissions are necessary to use the app. If they press OK, they will be brought to OS settings to manually accept permissions. Pressing cancel, they will remain on the dashboard.	The dialog box was shown, and the user is given the option to press cancel or accept.	Pass.
7	Manually deny permission in OS Settings and start Eye Tracking.	A dialog box will be shown, informing the user that permissions are necessary to use the app. If they press OK, they will be brought to OS settings to manually accept permissions. Pressing cancel, they will remain on the dashboard.	The dialog box was shown, and the user is given the option to press cancel or accept.	Pass.
8	Open Eye Tracking with permissions previously granted.	A dialog box showing the user how to position the phone is shown. On clicking ok, the user will see the camera preview and associated information.	The dialog box showing how to position the phone was shown. The main application was then displayed correctly.	Pass.
9	Close eyes for 2 seconds with 2 seconds set as the alarm delay time in Settings.	After 2 seconds, a red alert should be sounded, and an SMS message sent to the emergency contact if one is chosen.	The red alert was sounded after eyes were closed for 2 seconds. An SMS message was received by the emergency contact.	Pass.
10	Close eyes for 3 seconds with 3 seconds set as	After 3 seconds, a red alert should be sounded, and an SMS message sent to the	The red alert was sounded after eyes were closed for 3 seconds. An SMS	Pass.

	the alarm delay time in Settings.	emergency contact if one is chosen.	message was received by the emergency contact.	
11	Close eyes for 4 seconds with 4 seconds set as the alarm delay time in Settings.	After 4 seconds, a red alert should be sounded, and an SMS message sent to the emergency contact if one is chosen.	The red alert was sounded after eyes were closed for 3 seconds. An SMS message was received by the emergency contact.	Pass.
12	Remove face from screen after initially being detected.	After 2 seconds, a green alert will be triggered and a dialog box will be shown, indicating that a face is not detected.	The green alert was triggered.	Pass.
13	Introduce face into the frame after the green alert.	The green alert dialog box should be automatically removed.	The green alert dialog box was removed as soon as a face was detected.	Pass.
14	Blink more than 5 times in a given minute.	The application should continue tracking the face and eyes normally.	No alerts were triggered. The application continued driver monitoring as normal.	Pass.
15	Blink less than 5 times in a given minute.	A yellow alert should be triggered. Audio will play and the phone will vibrate. On dismissing the alert, Google Maps will open, displaying a list of petrol stations.	A yellow alert was triggered. An audio and visual alert was shown. The phone also vibrated. The application entered PIP mode and Google Maps opened at the current location.	Pass.
16	Trigger red alert with no emergency contact chosen.	The red alert will be triggered. No SMS message will be sent. The app will continue to monitor the driver as normal.	The red alert was triggered. On dismissing the alert, the app continued monitoring the driver.	Pass.

17	Click the "Turn preview off" button.	The camera preview will disappear and a DriveSafely logo will be shown.	The DriveSafely logo was shown. The camera preview was no longer visible.	Pass.
18	With the preview off click the "Turn preview on" button.	The DriveSafely logo will disappear, and the camera preview will be shown.	The camera preview was shown. The DriveSafely logo was no longer visible.	Pass.
19	Turn the camera preview off and close eyes for 3 seconds with 3 seconds set as alarm delay time in Settings.	After 3 seconds, a red alert should be sounded, and an SMS message sent to the emergency contact if one is chosen.	The red alert was triggered, and an SMS message was sent to the chosen emergency contact.	Pass.
20	Press "Open Google Maps " Button.	The application should enter PIP mode. Google Maps should open at the user's current location.	The app entered PIP mode and Google Maps opened at the user's current location.	Pass.
21	Press X to close PIP Window.	The camera preview will be removed from the screen and eye/face tracking will be stopped. Google Maps will remain open.	The camera preview was removed from the screen. Eye/Face tracking ceased. Google Maps remained open.	Pass.
22	Press [-] to enlarge PIP window.	The application will enlarge and fill the phone screen, as normal.	The main application screen reopened and filled the screen.	Pass.
23	Turn brightness to half of the max in low light.	The application should still detect a user's face.	The application detected the user's face.	Pass.

24	Turn brightness to a minimum in low light.	The user's face will be undetectable.	The user's face was not detected.	Pass.
25	Travel at a constant speed of 60km/h.	The speedometer should read 60km/h +-5 %.	The speedometer read 62km/h .	Pass.
26	Travel a pre-determined distance of 5km.	The trip distance should be equal to 5km +-0.1km.	The trip distance was equal to 5.1km.	Pass.
27	Turn phone volume to a minimum then open Eye Tracking.	The phone's volume should be set to half the max volume.	The volume was set to half volume as expected.	Pass.

Table 9: EyeTracking test cases

6.3 Further Testing

A test was carried out whereby for each value of alarm time delay, a video was recorded to ensure that after the specified time of the eyes being closed, the alarm went off. This was to ensure that Google Vision was correctly determining when a user's eyes are closed and that the system functioned as intended. A test is shown below with an alarm time of 3 seconds (See *Figure 29*). As seen, the time from the eyes being closed to the moment the alarm is triggered is 3.3 seconds. This is acceptable as certain operations take time, such as displaying the dialog box and sending an SMS may slow the system. There may also be error introduced in the editing software used to check and display the time stamp shown.

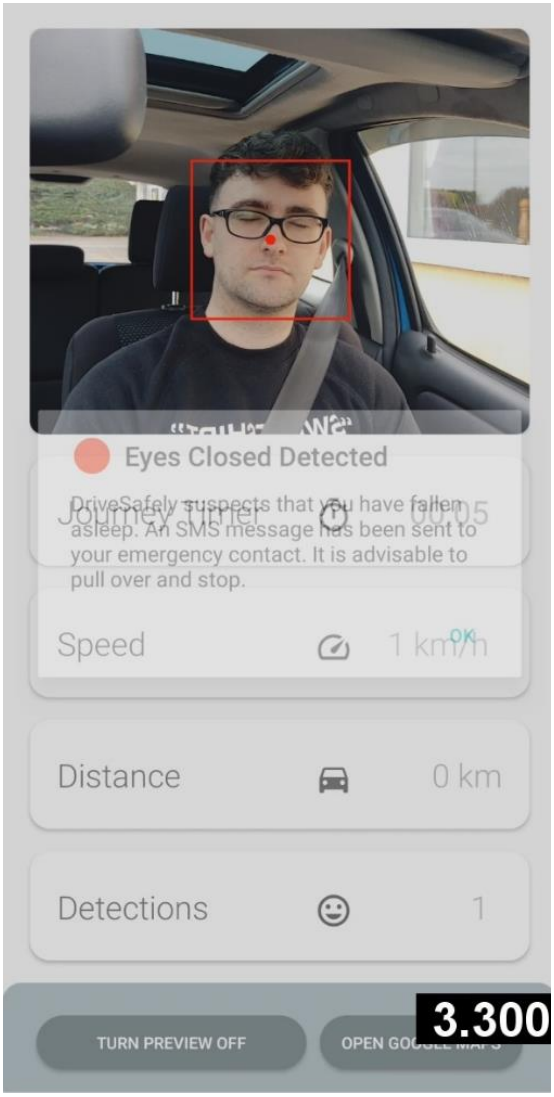
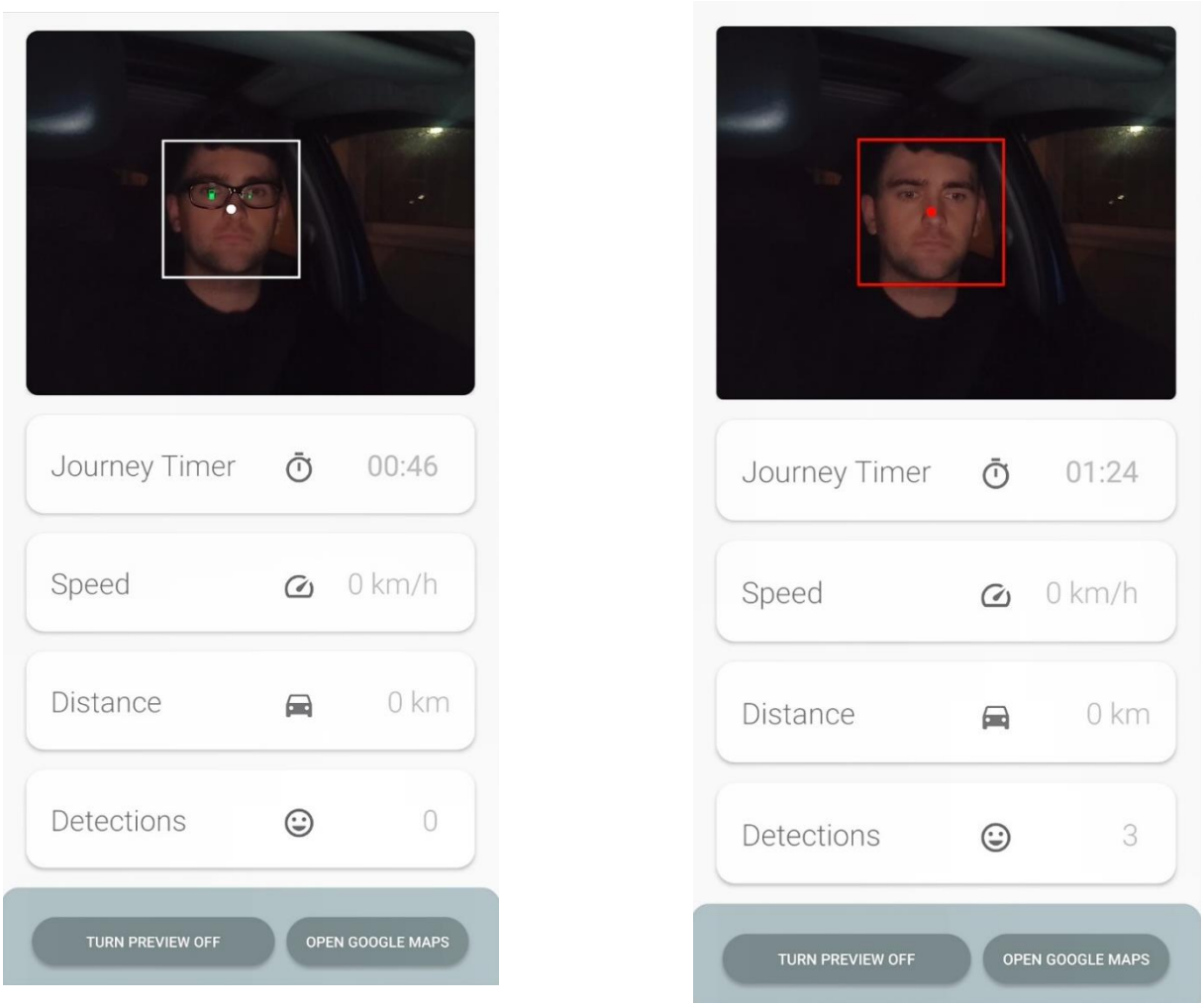


Figure 29: Test to ensure alert is triggered at the correct time

Another test was done late at night with the phone in a phone holder mounted on the car windscreen. The test was performed with the phone's brightness set to 50%. It was found that any brightness less than 50%, the user's face would not be detected when sitting in a normal driving position. With this brightness setting and phone placement, it was found that when wearing glasses, the user's right eye was difficult to detect due to the reflection of the phone screen. The application wasn't able to reliably detect when the user closed their eyes when wearing glasses. The application performed correctly for a user wearing glasses in natural daylight. When the glasses were removed, the application worked as intended. It was able to detect when the user's eyes were closed, detect the number of blinks in a minute and was able to determine if a user's head was dipping. This is an inherent downfall of using visible light to illuminate the user's face. If an infrared (IR) camera and infrared illumination could be used, it is believed that the application would be highly accurate for users with glasses and without. The phone's brightness could also be set to a minimum to avoid driver distraction.



a b

Figure 30: User wearing glasses at night (a) User without glasses at night (b)

Chapter 7: Conclusion and Recommendations

7.1. Conclusions

An Android application has been created that aims to combat drowsy driving and reduce road fatalities. It does this by tracking the driver's face and eyes. The user is alerted if their face is missing, if their head has dipped/nodded five times, if they do not blink five times or more in a given minute, or if they close their eyes for a certain period of time.
The application has other features including the ability to record journey statistics and use Google Maps while tracking their face and eyes. The application is also able to send SMS messages to an emergency contact.

From using various tools over the course of this project, it can be concluded that the Android Studio IDE is invaluable in the creation of Android applications. It provided a highly intelligent code editor and a very useful device emulator.

It can also be concluded that Java was an efficient language for application development. There is a lot of support and documentation available to aid in the creation of a mobile application using Java. Likewise, the same applies to XML.

Google Vision was paramount to the success of the project. It was the necessary tool that enabled the application to track a user's face and eyes.

7.2. Recommendations

With further development, this project could be improved in many ways by introducing additional capabilities. The first improvement that could be implemented would be to allow connectivity to a smart wearable device such as a smartwatch. This could be used to measure data such as the user's heart rate and blood oxygen levels. This would greatly improve the application as it would not rely on the camera, which may be unreliable at times when the driving environment is especially dark.
As mentioned previously, the idea of exploring the use of IR lighting and an IR camera could be useful to improve the reliability of the system in low lit environments.
Finally, an attempt could be made to port the application to the iOS platform to enable it to work on Apple smartphones. This would allow the app to reach much more of the global population and hopefully complete the goal of this project, which is to reduce the number of road fatalities caused by drowsiness and tiredness.

References

[1] Road Safety Authority, “Driver Fatigue,” 2014. [Online]. Available: https://www.rsa.ie/RSA/Road-Safety/Campaigns/Current-road-safety-campaigns/Drunk-With-Tiredness/. [Accessed 15 03 2021].

[2] N. F. Watson, “Recommended Amount of Sleep for a Healthy Adult: A Joint Consensus Statement of the American Academy of Sleep Medicine and Sleep Research Society,” *Journal of Clinical Sleep Medicine,* vol. 11, no. 06, pp. 591-592, 2015.

[3] Central Statistics Office, “CSO statistical publication,” 2015.

[4] RTE, “Studies show extent of driver fatigue at wheel,” 05 10 2020. [Online]. Available: https://www.rte.ie/news/ireland/2020/1005/1169424-rsa-driving-asleep/. [Accessed 15 03 2020].

[5] Road Safety Authority, “Driver Tiredness: The Facts,” 07 2008. [Online]. Available: https://www.rsa.ie/Documents/Campaigns/Drunk%20With%20Tiredness/Driver_Tiredness_The_facts%5b1%5d.pdf. [Accessed 15 03 2021].

[6] N. R. &. M. A. R. A. Arafat Islam, “A Study on Tiredness Assessment by Using Eye Blink Detection,” *Jurnal Kejuruteraan,* vol. 31(2), pp. 209-214, 2019.

[7] Galaxy Lab, “Drowsy Driving Alert: Sleepy driver warning,” 31 August 2020. [Online]. Available: https://play.google.com/store/apps/details?id=com.galaxylab.drowsydriver&hl=en_IE&gl=US. [Accessed 07 02 2021].

[8] A. Gandotra, “Drivelert,” 30 September 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.drivelert.akarshan.drivelert&hl=en_IE&gl=US. [Accessed 07 02 2021].

[9] Auto Xtreme, “Sleep Alert & GPS Speedometer Car Heads up Display,” 09 July 2019. [Online]. Available: https://play.google.com/store/apps/details?id=com.ax.speedometer.sleep.alert&hl=en_IE&gl=US.

[10] Google Developers, “Meet Android Studio,” 28 10 2020. [Online]. Available: https://developer.android.com/studio/intro#build_variants. [Accessed 10 02 2021].

[11] Gradle, “What is Gradle?,” [Online]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Accessed 16 03 2021].

[12] J. Tillu, “What is Gradle? Why Google choose it as an official build tool for Android.,” 05 12 2018. [Online]. Available: https://medium.com/jay-tillu/what-is-gradle-why-google-choose-it-as-official-build-tool-for-android-adafbff4034. [Accessed 10 02 2021].

[13] Google Developers, “Run apps on the Android Emulator,” 12 10 2020. [Online]. Available: https://developer.android.com/studio/run/emulator?gclid=Cj0KCQiApY6BBhCsARIs

AOI_Gjb-OGvyaa6GnxLNBF4-uBqj6FVwly0fvljl1rqCQBbgBHevdTIjF3oaAuQNEALw_wcB&gclsrc=aw.ds. [Accessed 10 02 2021].

[14] Google Developers, “Analyze your build with APK Analyzer,” 25 08 2020. [Online]. Available: https://developer.android.com/studio/build/apk-analyzer. [Accessed 13 02 2021].

[15] Google Developers, “Build a UI with Layout Editor,” 25 08 2020. [Online]. Available: https://developer.android.com/studio/write/layout-editor. [Accessed 16 02 2021].

[16] Google Developers, “Measure app performance with Android Profiler,” 12 10 2020. [Online]. Available: https://developer.android.com/studio/profile/android-profiler. [Accessed 16 02 2021].

[17] Google Developers, “Configure your build,” 13 03 2021. [Online]. Available: https://developer.android.com/studio/build. [Accessed 16 03 2021].

[18] Google Developers, “Face Detection Concepts Overview,” 12 09 2020. [Online]. Available: https://developers.google.com/vision/face-detection-concepts. [Accessed 23 02 2021].

[19] Google Developers, “Landmark,” 08 12 2020. [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/vision/face/Landmark. [Accessed 23 02 2021].

[20] “EasyPermissions,” 28 12 2020. [Online]. Available: https://github.com/googlesamples/easypermissions. [Accessed 2021 02 27].

[21] Google Vision, “CameraSourcePreview.java,” 26 01 2021. [Online]. Available: https://github.com/googlesamples/mlkit/blob/master/android/vision-quickstart/app/src/main/java/com/google/mlkit/vision/demo/CameraSourcePreview.java. [Accessed 06 03 2021].

[22] Google Vision, “GraphicOverlay,” 6 08 2015. [Online]. Available: https://github.com/googlesamples/android-vision/blob/master/visionSamples/FaceTracker/app/src/main/java/com/google/android/gms/samples/vision/face/facetracker/ui/camera/GraphicOverlay.java. [Accessed 07 03 2021].

[23] Google Vision, “FaceGraphic.java,” 29 12 2020. [Online]. Available: https://github.com/googlesamples/mlkit/blob/master/android/vision-quickstart/app/src/main/java/com/google/mlkit/vision/demo/java/facedetector/FaceGraphic.java. [Accessed 06 03 2021].

[24] Google Developers, 18 11 2020. [Online]. Available: https://developer.android.com/guide/topics/ui/picture-in-picture. [Accessed 06 03 2021].

[25] Mapbox, [Online]. Available: https://docs.mapbox.com/android/maps/guides/. [Accessed 13 03 2021].

Appendices

Appendix A: Interim Report

DriveSafely

By:

Reece Gavin – 17197589

Supervised By:

Prof. Hussain Mahdi

Final Year Project Interim Report 2020/21

Department of Electronic & Computer Engineering

University of Limerick

November 2020

1. Introduction and Project Outline

1.1 Project Rationale

In Ireland and throughout the world, fatigued driving is an ever-growing problem. A survey conducted by the RSA in 2014 found that over 1 in 10 motorists have fallen asleep at the wheel [1]. With technological advances in recent years, it is now common in newer cars to see crash avoidance systems and driver monitoring systems. These, however, can often be expensive optional extras on cars and commercial vehicles. Therefore, there is potential to introduce a low cost, high accuracy mobile application that will help the prevention of car accidents caused by driver fatigue and tiredness. The app will utilise camera image capture and digital image processing. With the help of the mobile phone's camera, the app will continuously monitor the driver's face and process each frame to detect different driver states and identify clues of tiredness, fatigue and/or distraction. The app will operate an alarm system to wake up a detected tired driver and will also have other features such as sending text messages in cases of emergency.

1.2 Project Aims and Objectives

The overall aim of the project is to reduce road traffic accidents that are attributed to driver fatigue and tiredness.

The objective of the project is to create an Android application that can directly compete with existing apps and vehicle built in driver monitoring systems.

1.3 System Outline

As mentioned in Section 4.1, Figure 9, the preliminary system and main features are as follows:

- The system will utilise the front facing camera to detect fatigued or distracted drivers.
- The phone will need to be sufficiently charged or be charging due to the screen potentially being on constantly (depending on driver selected settings).
- GPS will be used to incorporate Google maps into the application.
- A cellular connection will be used for sending SMS messages as well as utilising a 3G/4G connection for Google Maps and possibly other features.
- The phone's speaker will be used to sound alerts to the driver.
- The phone's display will be used to show visual alerts as well as to allow overall interaction with the app.

2. Literature Review

2.1 Fatigue: Symptoms and Causes

Fatigue is defined as: That state, following a period of mental or bodily activity, characterized by a lessened capacity for work and reduced efficiency of accomplishment, usually accompanied by a feeling of weariness, sleepiness, or irritability; may also supervene when, from any cause, energy expenditure outstrips restorative processes and may be confined to a single organ [2].

Symptoms of fatigue include:

- Chronic tiredness or sleepiness
- Headache
- Dizziness
- Slowed reflexes and responses
- Impaired hand-to-eye coordination

Causes of fatigue:

- Medical causes – unrelenting exhaustion may be a sign of an underlying illness, such as a thyroid disorder, heart disease or diabetes.
- Lifestyle-related causes – alcohol or drugs or lack of regular exercise can lead to feelings of fatigue.
- Workplace-related causes – workplace stress can lead to feelings of fatigue.
- Emotional concerns and stress – fatigue is a common symptom of mental health problems, such as depression and grief, and may be accompanied by other signs and symptoms, including irritability and lack of motivation [3].

2.2 Irish Driver Fatigue Statistics

The Road Safety Authority of Ireland (RSA) defines fatigue as the physical and mental impairment brought about by inadequate rest over a period of time.

Studies show that the general recommended amount of sleep required to promote and maintain optimal health for adults, aged between 18 and 60 years, is 7 or more hours on a continual basis [4]. Drivers that are suffering from a lack of sleep are at a high risk of falling asleep while driving, thus increasing their chances of being involved in a road crash.

According to the RSA it is estimated that driver fatigue is a contributory factor in as many as 1 in 5 driver deaths in Ireland every year. In addition to this, fatigue related incidents are 3 times more likely to be fatal or result in serious injury due to high impact speed due to an absence of action by the driver.

A survey conducted by the RSA in 2014 found that over 1 in 10 motorists have fallen asleep at the wheel [1]. The Central Statistics Office (CSO) had recorded a total of 2,710,664 Irish driving licenses were held at the end of 2014 [5]. This survey infers that approximately 271,000 drivers have fallen asleep at the wheel in Ireland. The RSA survey also found that motorists who drive as part of their work, and motorists who admit to driving after taking any amount of alcohol, had a higher than average incidence of falling asleep at the wheel (almost 1 in 5 fell asleep at the wheel) [1].

In 2020 the RSA presented the results of two studies at its annual lecture. One study found that 24% of drivers in Ireland admitted they had driven at least once over the previous month when they were so tired, they had trouble keeping their eyes open. In another study, 16% of drivers admitted they had fallen asleep or nodded off at the wheel. The Road Safety Authority has said driver fatigue is estimated to be a factor in 1 in 5 of the deaths of drivers on Irish roads. The RSA say that the groups most at risk are young men, people working night shifts, those who drive for a living (such as lorry and taxi drivers), and people with sleep disorders (such as sleep apnoea) [6].

In conclusion, in Ireland, the rate at which drivers have admitted to falling asleep at the wheel has increased from 10% in 2014 to 16% in 2020, which calls for a need to reduce this rate and stop the growing trend. The promoted solution to driver fatigue is to stop, have 2 cups of strong coffee and have a nap of no more than 15 minutes before proceeding back onto the road [7]. However, even with this promotion in place there is a need to take further action in reducing the amount of fatigue related driver incidents.

2.3 Worldwide Driver Statistics

The ESRA (E-Survey of Road Users' Attitudes) is a joint initiative of road safety institutes, research centres, public services, and private sponsors from all over the world. Its goal is to collect and analyse data on road safety, particularly road safety culture and behaviour of road users. These surveys provide scientific evidence for policy making at national and international levels. The most recent report was published in 2019, based on statistics from 2018.

The survey covers topics such as driving under the influence of alcohol, drugs and medicines, speeding and fatigue. The survey collects data from over 35,000 road users across 32 countries.

The ESRA publishes a thematic report solely based on driver fatigue which describes the rate of self-declared fatigued driving, the personal acceptability of fatigued driving, and the perception of driving fatigue as an accident cause amongst road users in the 32 countries.

The key results in relation to driver fatigue were listed as:

- In most countries one fifth to one quarter of car drivers report to have driven while having trouble keeping their eyes open, in the past 30 days.
- In all participating countries worldwide, less than 3% of road users find fatigued driving personally acceptable.
- 74% of European road users most frequently perceive tired driving as a frequent crash cause, with lower rates reported amongst road users in North America at 69%, Africa 64% and Asia-Oceania 53%. This is illustrated below in Figure 2.
- In Europe, Austrian and Finnish drivers report the highest rates of fatigued driving at 32% and 29% respectively, whereas drivers in UK, Italy, and Serbia report the lowest rates at 15% in the UK and 14% in Italy and Serbia. In Asia-Oceania, Japanese drivers most frequently report fatigued driving at 33% and Australian drivers least frequently at 17%. In Africa, Egyptian drivers more frequently report fatigued driving at 31%, than drivers in Kenya and Nigeria, both at 18%.

Figure 1 presents the results of car drivers for self-declared fatigued driving in the past 30 days per global region and country. The self-declared fatigued driving in the past 30 days varies from 20% in Europe to 25% in Africa. The rates of self-declared fatigued driving are 22% for North America and 23% for Asia-Oceania. The self-declared fatigued driving rates is approximately 17% in Europe.

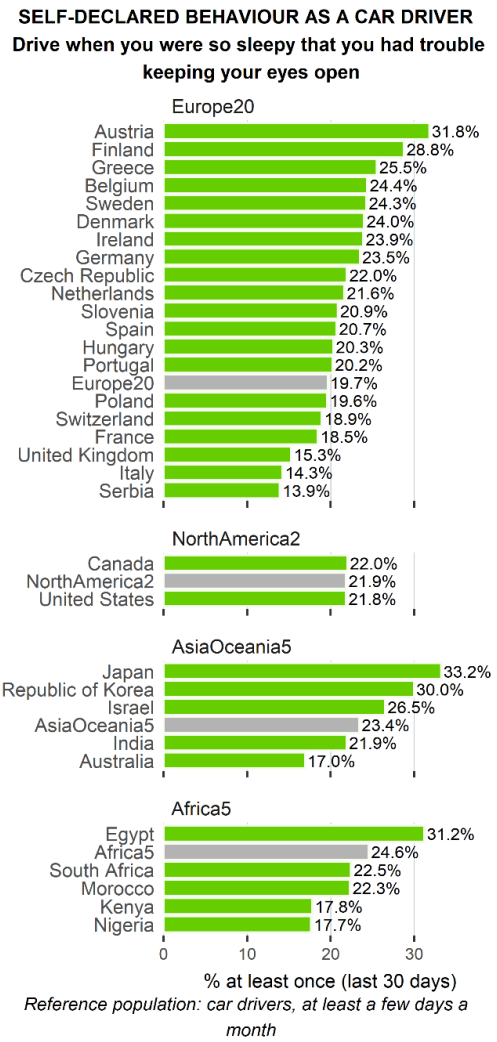


Figure 1: Self-declared fatigued driving by car drivers per region at least once in the past 30 days [8].

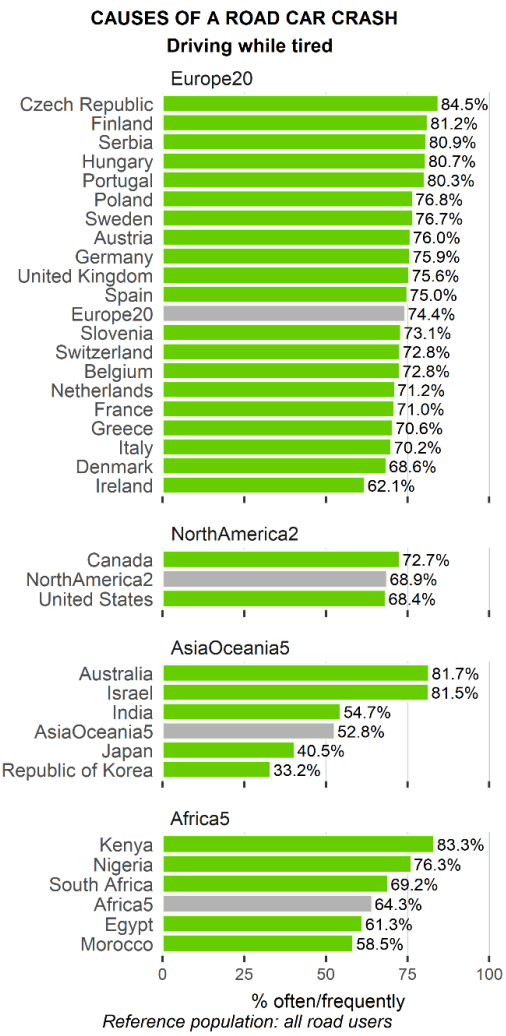


Figure 2: Results of the question on how often fatigued or tired driving is perceived to be a frequent road crash cause [8].

The report concluded that fatigued driving is one of the major problems for road safety worldwide. The high prevalence of self-declared fatigue driving warrants serious attention of road safety policy makers.

To prevent driving with fatigue, more attention should be paid to create further measures in the fields of legislation, road infrastructure, education and campaigns, the implementation of safety culture and fatigue management programmes in companies [8].

2.4 Existing Applications to Prevent Fatigue Related Accidents

There are several pre-existing Android apps that carry out some form of eye tracking and/or facial recognition to prevent road traffic accidents due to tiredness and fatigue. The details of the three most effective apps are outlined below.

2.4.1 DriveSafely

This application recognises driver's drowsiness and distraction, and provides audible and visual alerts. The application uses a front-facing camera and sensors built into smartphone, including GPS, accelerometer, gyroscope, magnetometer, and microphone.

The application works with GPS systems such as Google Maps and OpenStreetMap. The image of the driver can be hidden from view or overlayed onto the map in a small window so the driver can see themselves. An example of what a driver may see is shown in Figure 3. As seen, the image is overlaid on top of Google Maps, while also showing the points on the face which are used for tracking. The application does not interfere with incoming texts or phone calls.

The application overlays alert icons to the screen and plays warning sounds that correspond to dangerous driving events. It helps drivers to be notified of a dangerous event occurrence and reduces the probability of a traffic accident [9].

The application works as intended; however, the user interface is not user friendly and has too many options with regards to tuning the application to detect hazardous states.

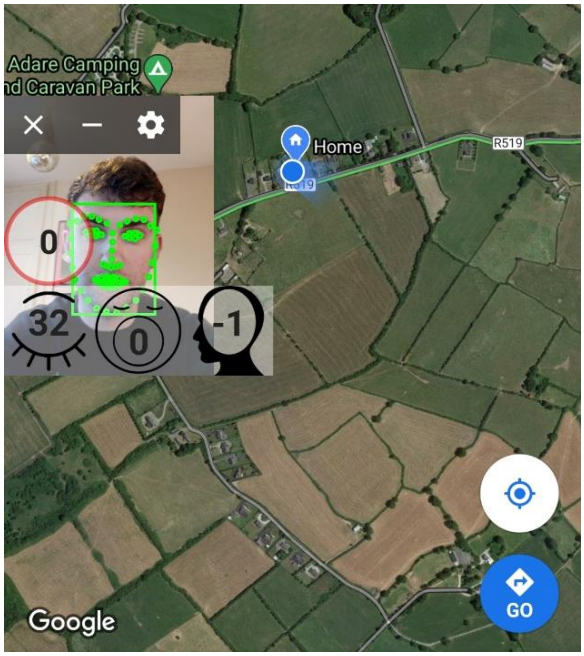


Figure 3: DriveSafely app overlayed on Google Maps

2.4.2 Drivelert

Drivelert is an Android application that helps to combat drowsy driving. It tracks drivers' eyes and sounds an alert when it detects a driver's eyes have been closed after a user defined time length.

Drivelert is highly accurate, provided that the driver's eyes are in view of the front facing camera. It has a user customisable time sensitivity. It works for users who wear glasses and has three alarm tones that can be chosen by the user.

It also introduces a traffic light system that can be seen in Figures 4 and 5 below. A green light signifies that no symptoms of drowsiness are prominent. A yellow light signifies that no face is present. A red light indicates that symptoms of drowsiness are present, and a driver should not continue to drive.

As seen in Figure 4, a box is drawn around a user's face and follows their face. This confirms to the user that the face and eyes are being tracked. Seen at the bottom of Figure 4, the preview can be turned off in case a user is being distracted by their own preview.

Seen in Figure 5, a visual alert is displayed when a driver is detected as showing symptoms of drowsiness. The traffic light has turned to red also. The user defined audio alarm is also played during the time that it detects a user's eyes are closed and stops when they reopen.

The app is very effective and accurate; however, the UI is lacking as the application was last updated in 2017.

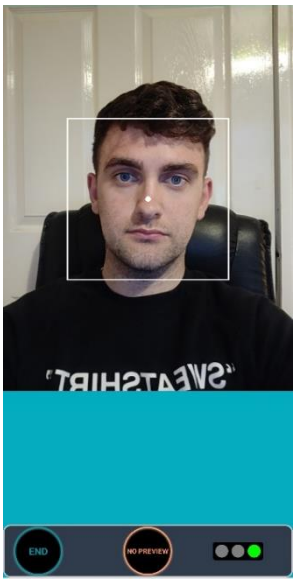


Figure 4: Application view when eyes are open

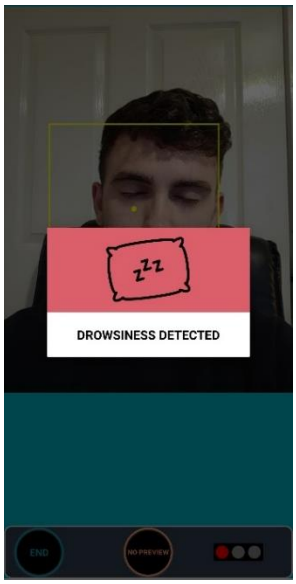


Figure 5: Visual and audio alerts are used when drowsiness is detected

2.4.3 Sleep Alert & GPS Speedometer Car Heads up Display

"Sleep Alert GPS Speedometer Car Heads up Display" is an application based on the facial recognition of a driver, which is obtained with a preview of the camera on their mobile phone. It is designed to keep a user awake on long distance journeys. It tracks driver's eyes and initiates an alert if a driver falls asleep while driving [10].

The app is feature rich with the ability to show a user's speed in kilometres per hour or in miles per hour. It also calculates distance travelled and average speed as seen in Figure 6. It has an integrated compass, journey timer and has an integrated map, seen in Figure 7, if a user does not wish to view the speedometer.

The app is advertised as having a very low percentage of false alarms and having an extremely reliable interactive UI. Figure 8 displays the visual and audible warning that is shown when the user's eyes are detected as being closed.

Overall, this app has the best functionality and useability. The eye tracking is very accurate, and a warning is displayed when no eyes are detected. It also saves each journey so that a user can review it later. However, there is some features that are missing such as being able to turn off the camera preview or being able to input a destination into the built in Google Maps.

Figure 6: Distance and speed data shown to the user.

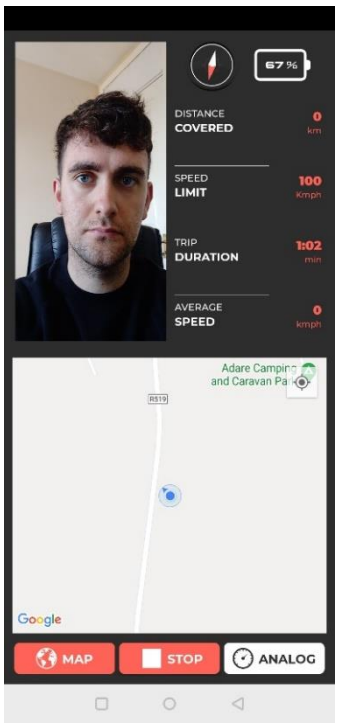


Figure 7: Map data can be shown instead of speed data

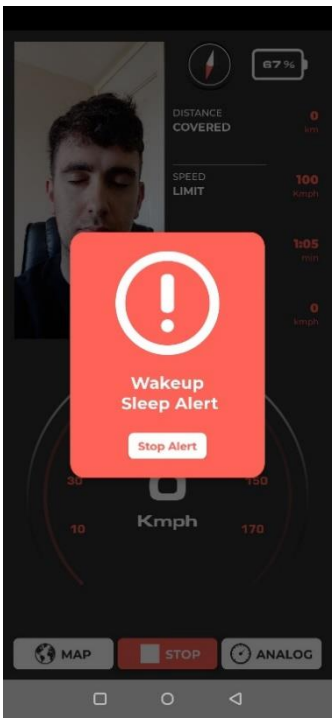


Figure 8: Audible and visual alarm displayed

3. Theoretical Background

3.1 Eye Tracking Methods

Eye tracking dates back as far as the 1920's with the introduction of Electro-Oculography (EOG). It was discovered that by placing electrodes on the skin close to the eyes, electrical activity could be recorded which changed synchronously with the movements of an eye.

At first it was believed that these potentials reflected the action potentials in the muscles that are responsible for moving the eyes. However, it is now generally agreed that these electrical potentials are generated by the permanent potential difference which exists between the cornea and the ocular fundus. The cornea-retinal potential has been determined to be a positive charge of between 10-30mV.

This potential difference sets up an electrical field in the tissues surrounding the eye. As the eye moves, the field vector moves correspondingly. Therefore, eye movements can be detected by placing electrodes on the skin around the eyes. Vertical movements of the eyes are best measured by placing the electrodes on the lids, while horizontal eye movements can be best measured by placing the electrodes on the bone at the side of the eye, called the canthi [11].

EOG is primarily used to detect retinal abnormalities in medicinal patients and has been succeeded by other methods in terms of modern-day eye tracking techniques.

There are several modern different techniques to detect and track the movements of the eyes. However, the most used technique in commercial and medical eye tracking is pupil centre corneal reflection (PCCR). In PCCR, a light source, sometimes infrared, is used to illuminate the eye causing a visible (or invisible) reflection in conjunction with a camera to capture an image of the eye showing these reflections. The image captured by the camera is then used to identify the reflection of the light source on the cornea (glint) and in the pupil. A vector is then calculated by using the angle between the pupil and corneal reflections. This vector can be used with other geometrical features of the reflections to calculate a gaze direction [12].

Eye tracking for use in mobile and PC based applications can be implemented in numerous different ways, using different algorithms, some with more success than others. The algorithms listed are possible implementations that can be used, as they are easily accessible through an open source software called OpenCV.

One method of tracking eyes is template matching. Template matching is a technique for finding areas of an image that are similar to a template image. In this technique there are two necessities, being a source image and a template image. The source image is the original image in which the template is applied, in order to find a specific area. The template image is the image that is used for the comparison with the source image in order to detect the best matching area. To identify the matching area, the template image has to move on top of the source image one pixel at a time. Template matching calculations take place with every pixel of the image, to show if the point from the template image is a good match to the respective point of the source image at that specific time. The value of this comparison is stored in a matrix. There are several methods for template matching in OpenCV. One possible implementation is called correlation coefficient.

Another possible method used for detecting the centre of the pupil is the one curated by Timm and Barth [13] . Below, the following outline of Timm and Barths method is outlined by Iosef Karkanis's dissertation "Exploring Eye Tracking Techniques on Smartphones".

To find the centre of a circular object, the pupil of the eyes in this case, image gradient vectors are used. A possible centre depends on the orientation of these vectors.

Assuming c is a possible centre and g_i (equation 1.1) is the gradient vector at position x_i, then d_i (equation 1.2) which is the normalized vector needs to have the same orientation with the gradient vector g_i.

$$g_i = \left(\frac{\partial(x_i, y_i)}{\partial x_i}, \frac{\partial I(x_i, y_i)}{\partial y_i} \right)^T \tag{1.1}$$

$$d_i = \frac{x_i - c}{\| x_i - c \|_2} \quad , \quad \forall i : \| g_i \|_2 = 1 \tag{1.2}$$

where:

$$\frac{\partial(x_i, y_i)}{\partial x_i} = \frac{I(x+1, y) - I(x-1, y)}{2} \tag{1.3}$$

$$\frac{\partial I(x_i, y_i)}{\partial y_i} = \frac{I(x, y+1) - I(x, y-1)}{2} \tag{1.4}$$

The possible centre c^* is calculated by finding the dot products between the normalised vector d_i and the gradient g_i at the pixel x_i , $i \in \{1, \dots, N\}$

$$c^* = argmax_c \left\{ \frac{1}{N} \sum_{i=1}^{N} (d_i^T g_i)^2 \right\} \tag{1.5}$$

To obtain equal weight for every pixel position, both d_i vectors and the corresponding g_i should be scaled to unit length vectors to improve lighting and contrast in the image.

Occasionally glasses or eyelids may cause a wrong centre estimation. To counteract this, a weight w_c is applied so the dark possible centres are chosen instead of the bright ones. w_c is the grey value of the inverted input image I^*, shown is equation 1.6

$$w_c = I^*(c_x, c_y) \tag{1.6}$$

A gaussian filter is applied to avoid reflections of glasses and over bright spots, such as in direct sunlight.

The possible centre of the eye is calculated by using the weight as shown in equation 1.7.

$$c^* = argmax_c \left\{ \frac{1}{N} \sum_{i=1}^{N} w_c (d_i^T g_i)^2 \right\} \tag{1.7}$$

This is all the necessary information needed when the picture obtains the image of the eye. Sometimes hair, glasses, and eyebrows show image gradients with different direction in comparison with these from the pupil. A post processing step would help correct this. A threshold is applied in order to remove all the gradient values from the borders of the image. The maximum of these values is used to determine the centre of the pupil [14].

3.2 Eye Tracking and Facial Recognition Technologies

3.2.1 OpenCV

A possible software library that can be used to monitor a driver's facial features and eyes is OpenCV. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products [15].

The OpenCV library has more than 2,500 algorithms which include computer vision and machine learning algorithms. These algorithms can be used for purposes such as to detect and recognise faces, identify objects, classify human actions in videos, track camera movements, track moving objects, follow eye movements and much more.

OpenCV is employed by companies such as Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, and Toyota. It is used for purposes such as stitching Google Street View images together, detecting intrusions on CCTV cameras, helping robots navigate and pick up objects, and detection of swimming pool drowning accidents in Europe.

OpenCV has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. There are many pre-written eye tracking and facial recognition libraries available online which allows for quick integration into an app.

3.2.2 Google Vision

Google vision is an API that allows developers to analyse the content of an image through extracted data. For this purpose, Google utilises machine learning models trained on a large dataset of images. All of that is available with a single API request. The engine behind the API classifies images, detects objects, people's faces, and recognises printed words within images [16].

Google vision has many different uses, one example of this is if it is shown a photo of a dog, it can classify firstly that an animal is in the photo. It can then classify that the animal is a dog and finally be able to determine what breed of dog is seen in the photo.

Some of the main features of Google vision are:

- Label Detection: Detect a set of categories within an image (the example above).

- Explicit Content Detection: Detect if there are explicit content (adult/violent) within an image.
- Logo Detection: Detect popular logos within an image.
- Landmark Detection: Detect natural and man-made structures within an image.
- Optical Character Recognition: Detect and extract text within an image. It can even detect the language of the text.
- Face Detection: Detect multiple faces within an image, along with other attributes like emotional state or wearing headwear [17].

A negative aspect of Google Vision is that it is free for the first 1000 units/per month and after that threshold has been reached, the price is $1.50 per 1000 units [18].

3.2.3 PyGaze

The PyGaze toolbox is an open-source software package for Python. It is designed for creating eye-tracking applications in Python with the least possible effort and offers programming ease and script readability without affecting functionality and flexibility. PyGaze is advantageous as it provides an easy-to-use layer on top of the many different software libraries that are required for implementing eye-tracking experiments. Essentially, PyGaze is a software-bridge for eye-tracking research [19].

PyGaze is currently compatible with SR Research's Eyelink systems, all SMI products that run via iViewX, and Tobii devices, the likes of which have various uses such as laboratory experiments and in use of quadriplegic non vocal wheelchair users to communicate via their eyes. This software is usually provided along with the eye-trackers. The classes for Eyelink, SMI, and Tobii use the same methods, meaning that a PyGaze script can be used for all three types of systems, without having to adjust the code [19].

3.3 Possible Programming Language Implementations

There are numerous ways to implement an Android app. Java is the official language for Android app development and as a result, is the most used language for producing Android apps. Most Google Play Store apps are created using Java. Java is a widely used language in modern day programming and as such has a lot of online resources, tutorial videos and forums

based around Java and app development. Java can be a difficult language to learn and use due to some complex topics such as constructors, null pointer exceptions, concurrency, inheritance etc. [20].

Kotlin is a statically typed programming language for Java Virtual Machine (JVM) and JavaScript. Described as a general-purpose language, Kotlin introduces functional features to support Java interoperability. A goal of the Kotlin language is to simplify and reduce the amount of code that developers must write. Kotlin uses improved syntax, concise expression, and abstractions. It reduces excessive boilerplate code. In computer programming, boilerplate code or boilerplate refers to sections of code that have to be included in many places with little or no alteration. It is often used when referring to languages that are considered verbose, i.e. the programmer must write a lot of code to do minimal jobs [21]. Many large corporations such as Pinterest, Twitter, Netflix, Uber, AirBnB have switched to Kotlin for the development of their Android applications [22].

It is possible to use C++ for Android app development using the Android Native Development Kit (NDK). An app cannot solely be created using C++. It is much more difficult to implement an app in C++ compared to Kotlin or Java. It may also be a cause for bugs in the code due to complexity. As a result, it is ill advised to use C++ in app development.

C# is a very similar language to Java, so it is ideal for Android app development. C# has a simpler syntax than Java and the code is comparatively cleaner than Java. C# can be used to write native Android apps.

Android does not natively support Python; however, Python can be used for app development. This can be done using various tools that convert the Python apps into Android packages that can run on Android devices. An example of this is Kivy. Kivy is an open source, cross-platform Python framework for the development of applications that make use of innovative, multi-touch user interfaces. The aim is to allow for quick and easy interaction design and rapid prototyping whilst making your code reusable and deployable [23]. A downside to Kivy is that it is not updated frequently, the latest version being released on June 21, 2019. There also will not be native benefits for Kivy as it is not natively supported.

Corona is a software development kit that can be used for developing Android apps using Lua. It has two operational modes, namely Corona Simulator and Corona Native. The Corona

Simulator is used to build apps directly while the Corona Native is used to integrate the Lua code with an Android Studio project to build an app using native features [20].

Corona has built in monetization features as well as various plugins which contribute to the overall app development. It is primarily used to create graphical apps and games.

Android apps can be created using HTML, CSS, and JavaScript. However, apps are usually primitive and there is a substantial amount of work to create a sophisticated app.

In conclusion, there are several different ways to implement an Android app. The decision of what method and language to use is primarily down to the coder themselves based on their coding background, experience, and what specifications the overall app needs to achieve.

3.4 Development of Android Applications

Android Studio is the most widely used software used in the development of Android applications due to ease of use as well as a wide range of support and online information. XML is often used in conjunction with Android Studio to create the overall layout of the application, such as laying out buttons, text boxes, etc.

3.4.1 Android Studio

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools [24].

Android Studio was first announced in 2013 and the first stable build released in late 2014. It is available for Windows, Mac, and Linux.

Android Studio uses a Gradle-based build system, has a built-in emulator, code templates and has GitHub integration for large scale app development. Gradle is an open-source build automation tool that is designed to be flexible enough to build almost any type of software [25]. Gradle is useful for tasks such as downloading dependencies, compiling source code into executable code, packing that executable code and generate APK or executable form, running different tests etc. [26].

Android Studio includes features such as Android app modules, library modules, and Google app engine modules. Android Studio uses an instant push feature to push code and resource changes to a running application. There is a built-in code editor that assists a developer, offering

code completion, suggested changes, and refraction. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

3.4.2 XML

XML stands for Extensible Markup Language. It is a text based Markup language derived from the Standard Generalized Markup Language (SGML). XML is well readable both by human and machine. It is also scalable and simple to develop.

XML tags identify data and are used to store and organise data, rather than specifying how to display it. Often HTML is used to present XML data. XML as a language does not do anything.

There are three important characteristics of XML that make it useful in a variety of systems and solutions:

- XML is extensible − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- XML carries the data, does not present it.
- XML is a public open standard.

In Android app development there are several XML files used for several different purposes:

1. Layout XML files are used to define the actual UI (User Interface) of the application. It holds elements such as TextView's, Button's and other UI elements.
2. The Manifest XML file is used to define all the components of our application. It includes the names of the application packages, activities, receivers, services, and the permissions that the application needs.
3. The Strings XML file is used to replace the hard-coded strings. This file enhances the overall reusability of the code.
4. The Styles file (styles.xml): This file is used to define different styles and looks for the UI (User Interface) of application. Custom themes and styles are defined in this file.
5. **Drawable XML files** are files that are used to provide various graphics to the elements or views of application. It can be used to create custom UI's.
6. **The Color XML file is** used to define the colour codes that are intended to be used in an app.
7. The Dimension XML file is used to define the dimensions of aspects of an App such as pixel densities of a button.

4. Preliminary System Design and Specification

4.1 System Outline and Main Features

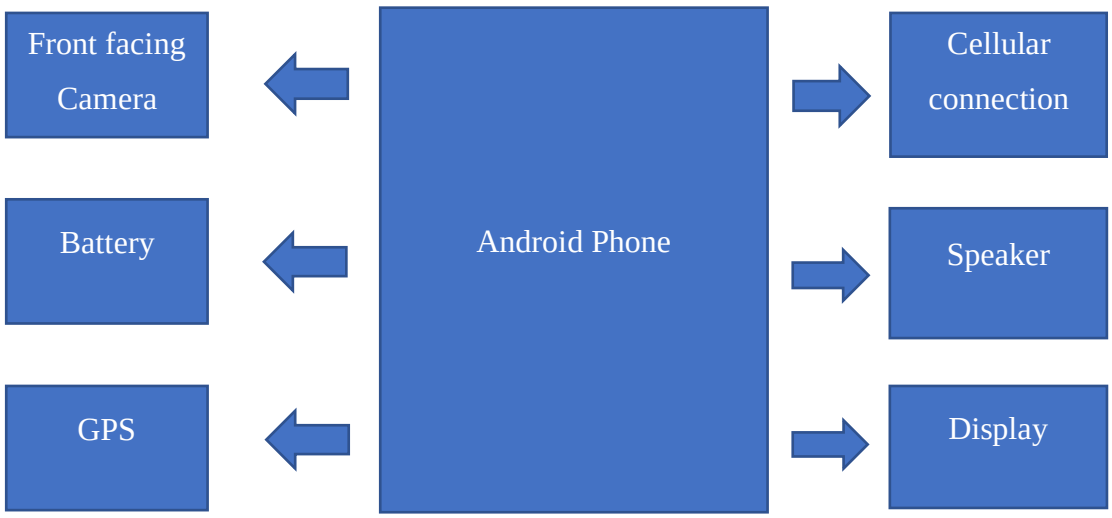


Figure 9: System outline and main components

As previously mentioned in Section 1 of this report. The below points set out the main components the application will rely on to operate with full functionality.

- The system will utilise the front facing camera to detect fatigued or distracted drivers.
- The phone will need to be sufficiently charged or be charging due to the screen potentially being on constantly (depending on driver selected settings).
- GPS will be used to incorporate Google maps to into the application.
- A cellular connection will be used for sending SMS message as well as utilising a 3G/4G connection for Google Maps and possibly other features.
- The phones speaker will be used to sound alerts to the driver.
- The phones display will be used to show visual alerts as well as to allow overall interaction with the app.

The system will be coded using Android Studio as the IDE, Java as the coding language, XML to layout the GUI and OpenCV will be used to detect faces and track eyes.
Possible other languages and services may be used to implement the described features and additional features, during the application development.

4.2 Preliminary Application Flow Chart

Seen in Figure 10 below is the preliminary application flow chart. This explains the intended operation of the application. When the app is opened, a user will select their desired settings such as being able to turn off the camera preview, select mph or km/h etc. Once the user starts the app, the application will first detect if a face is detected. If no face is detected a warning will be displayed on screen. If a face is detected, the app will check if the driver's eyes are open or closed. If they are open, the app will continue to monitor the eyes. If the eyes are detected as being closed, a warning sound will be played and displayed on screen. If the driver's eyes are reopened the app will return to checking if the eyes are open or closed. If the driver does not reopen their eyes, the warning sound will continue to play and after twenty seconds an emergency SMS will be sent to a preferred contact.

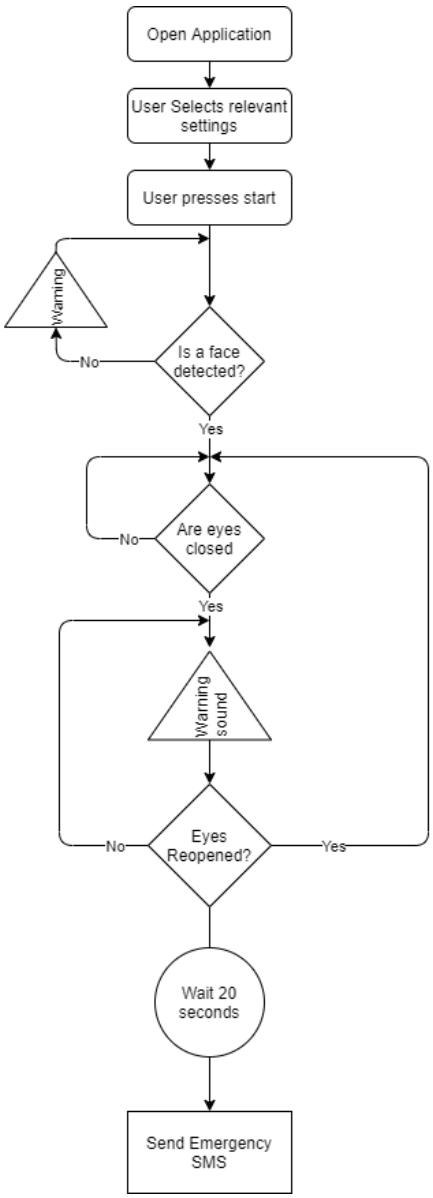


Figure 10: Preliminary application flowchart

5. Detailed Action Plan

For this project, time management and planning are crucial to meeting each of the project's deadlines. Below is the main list of tasks that need to be carried out at each stage of the project and a short description of what the task will entail. A Gantt chart can be found on the next page which gives a breakdown of time allocated to each task.

1. **Initial Contact with Supervisor** – I emailed my academic supervisor, formally introducing myself and giving any necessary information to him. I also asked questions relating to the project. After the initial contact, we agreed to hold a fortnightly Skype meeting to review my progress and for me to ask any questions.
2. **Literature Review** – The literature review involves surveying scholarly sources such as books, journal articles, reports, online documents, and web sites relating to my project. It gives an overview of the necessity for my project and identifies similar projects.
3. **Theoretical Background** – This is where I define, discuss, and evaluate theories, technologies, and tools relevant to my project. I also identified available technologies, coding languages and other developmental tools.
4. **Interim Report** – The interim report is created to gather all documented information in one location, to detail preliminary designs and to give a brief overview of the project and what it will entail.
5. **Presentation to Panel** – A brief presentation will be given to a panel of academics, to outline the findings of the interim report and explain the details of the project.
6. **Application Creation** – The application creation will involve coding the functionality of the project. This will entail creating a Graphical User Interface (GUI), implementing eye and facial tracking using OpenCV, as well as implementing functionality to send emergency text messages and incorporate GPS/ Google maps into the app.
7. **Application Testing** – Once the application is made, it will be thoroughly tested to ensure that all aspects of it work without bugs before being made available on the Google Play Store.
8. **Final FYP Report** – The final report will clearly state the purpose and background of the project, what has been achieved during the project.
9. **Final Presentation** – A final presentation will be presented to a panel based on a poster that will be created, which will give an overview of the completed project and demonstrate its functionality.

Task name	Start date	End date	Progress	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16	WEEK 17	WEEK 18	WEEK 19	WEEK 20	WEEK 21	WEEK 22	WEEK 23	WEEK 24	WEEK 25
Final Year Project	**28/09/20**	**24/03/21**	**25%**																									
1. Initial contact with supervisor	04/10/20	04/10/20	**100%**																									
2. Literature review	28/09/20	20/11/20	**100%**																									
3.Theoretical background	05/10/20	20/11/20	**100%**																									
4. Interim report	28/09/20	20/11/20	**100%**																									
5. Presentation to panel	25/11/20	25/11/20	**0%**																									
6. Application creation	30/11/20	14/02/20	**0%**																									
6.1 Code basic app functionality and UI	30/11/20	03/01/21	**0%**																									
6.2 Implement OpenCV eye tracking and facial recognition	04/01/21	24/01/21	**0%**																									
6.3 Implement alarm-based systems and SMS and GPS	25/01/21	07/02/21	**0%**																									
6.4 Application testing	08/02/21	21/02/21	**0%**																									
7. Final report	21/12/20	14/03/21	**0%**																									
8. Final presentation	24/03/21	24/03/21	**0%**																									

6. Requirements of Facilities and Materials

For the project, the following facilities and materials are required:

- An Android phone
- A computer to code the application
- Android Studio
- Android SDK (Software Developer Kit)
- Microsoft Word
- Endnote X9

For this project I will use my own personal Android phone, which is a OnePlus 6T, which uses Android version 10 as its operating system.

I will use my own personal PC to research, create and design the application and accompanying report. Windows 10 is the operating system that will be used.

The latest version of Android Studio, version 4.1, will be used to code and create the Android application. Android Studio contains the most up to date version of Android SDK.

Microsoft word will be used to create the necessary Interim and Final reports.

Endnote X9 will be used as a referencing tool. It has been obtained from the University of Limerick's Library.

7. References and Sources of Information

[1] R. S. A. (RSA). (2014). "Driver Fatigue." [Online]. Available: https://www.rsa.ie/RSA/Road-Safety/Campaigns/Current-road-safety-campaigns/Drunk-With-Tiredness/

[2] G. W. W. E. S. Williams, *Basic Anesthesiology Examination Review* 2015.

[3] BetterHealth. (2015). "Fatigue." [Online]. Available: https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/fatigue

[4] N. F. Watson *et al.*, "Recommended Amount of Sleep for a Healthy Adult: A Joint Consensus Statement of the American Academy of Sleep Medicine and Sleep Research Society," (in eng), *Sleep*, vol. 38, no. 6, pp. 843-844, 2015, doi: 10.5665/sleep.4716.

[5] C. S. Office. (2014). "Transport Omnibus 2014." [Online]. Available: https://www.cso.ie/en/releasesandpublications/ep/p-tranom/to2014/vs/dt/

[6] RTE. (2020). "Studies show extent of driver fatigue at wheel." [Online]. Available: https://www.rte.ie/news/ireland/2020/1005/1169424-rsa-driving-asleep/

[7] R. S. Authority. *Driver Tiredness: The Facts*. (2008). Pamphlet. [Online]. Available: https://www.rsa.ie/Documents/Campaigns/Drunk%20With%20Tiredness/Driver_Tiredness_The_facts%5b1%5d.pdf

[8] ESRA, "Driver Fatigue," in "ESRA Thermatic report Nr. 4 ", www.esranet.eu, 2019. [Online]. Available: https://www.esranet.eu/storage/minisites/esra2018thematicreportno4fatigue.pdf

[9] Fruct. (2020). "DriveSafely." [Online]. Available: https://play.google.com/store/apps/details?id=ru.igla.drivesafely

[10] A. Xtreme. (2019). "Sleep Alert & GPS Speedometer Car Heads up Display." [Online]. Available: https://play.google.com/store/apps/details?id=com.ax.speedometer.sleep.alert

[11] M. University. (2016). "Biological Signals Acquisition." [Online]. Available: https://www.medicine.mcgill.ca/physio/vlab/Other_exps/EOG/eogintro_n.htm

[12] T. AB. (2020). "How do Tobii Eye Trackers work?" [Online]. Available: https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/

[13] F. T. a. E. Barth, "ACCURATE EYE CENTRE LOCALISATION BY MEANS OF GRADIENTS," *In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP-2011)*, pp. 125-130, 2011, doi: 10.5220/0003326101250130.

[14] I. Karkanis, "Exploring Eye Tracking Techniques On Smartphones," 2015.

[15] OpenCV. (2020). "About openCV." [Online]. Available: https://opencv.org/about/

[16] B. Biskupski. (2019). "Powerful Image Analysis With Google Cloud Vision And Python." [Online]. Available: https://www.smashingmagazine.com/2019/01/powerful-image-analysis-google-cloud-vision-python/

[17] L. Pirro. (2018). "Quick Look at Google Cloud Vision API on Android." [Online]. Available: https://medium.com/iquii/quick-look-at-google-cloud-vision-api-on-android-66d87d08943

[18] Google. (2020). "Pricing." [Online]. Available: https://cloud.google.com/vision/pricing

[19] E. S. Dalmaijer, S. Mathôt, and S. Van der Stigchel, "PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments," *Behavior Research Methods,* vol. 46, no. 4, pp. 913-921, 2014/12/01 2014, doi: 10.3758/s13428-013-0422-2.

[20] H. Kaur. (2019). "Top Programming Languages for Android App Development." [Online]. Available: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

[21] M. Zaveri. (2018). "What is boilerplate and why do we use it?" [Online]. Available: https://www.freecodecamp.org/news/whats-boilerplate-and-why-do-we-use-it-let-s-check-out-the-coding-style-guide-ac2b6c814ee7/

[22] ClearBridgeMobile. (2020). "Kotlin vs. Java: Which is the Better Option for Android App Development?" [Online]. Available: https://clearbridgemobile.com/java-vs-kotlin-which-is-the-better-option-for-android-app-development/

[23] PyPi. (2020). "Kivy." [Online]. Available: https://pypi.org/project/Kivy/

[24] M. Rouse. (2018). "Android Studio." [Online]. Available: https://searchmobilecomputing.techtarget.com/definition/Android-Studio

[25] Gradle. "What is Gradle?" [Online]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html

[26] J. Tillu. (2018). "What is Gradle." [Online]. Available: https://medium.com/jay-tillu/what-is-gradle-why-google-choose-it-as-official-build-tool-for-android-adafbff4034

Appendix B: Java Source Code

a. SplashScreen.java

```
/*
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 23/02/2021
 */

package com.example.drivesafely;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;

public class SplashScreen extends AppCompatActivity {

    // Create variables for the animation
    private Animation topAnim;
    private ImageView image;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_splash_screen);

        //Animation
        topAnim = AnimationUtils.loadAnimation(this,
R.anim.top_animation);

        // The animationImage is found in the layout using findViewById.
        // This allows the value to be altered programmatically.*/
        image = findViewById(R.id.animationImage);
        image.setAnimation(topAnim);

        //Splash screen will be displayed for 3 seconds
        int SPLASH_SCREEN = 3000;

        //After the Splash Screen, the Dashboard will be displayed after 3
seconds
        new Handler().postDelayed(() -> {
            Intent intent = new Intent(SplashScreen.this,
Dashboard.class);
```

```
            startActivity(intent);
            finish();
        }, SPLASH_SCREEN);
    }
}
```

b. Dashboard.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import all necessary imports
import android.content.Intent;
import android.graphics.Typeface;
import android.os.Bundle;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

//Class used to create the main dashboard of the application
public class Dashboard extends AppCompatActivity implements
View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Set the layout to activity_dashboard.XML
        setContentView(R.layout.activity_dashboard);
        TextView txt = (TextView) findViewById(R.id.custom_fonts);
        txt.setTextSize(24);
        Typeface font = Typeface.createFromAsset(getAssets(),
"fonts/slim_joe.otf");
        Typeface font2 = Typeface.createFromAsset(getAssets(),
"fonts/big_john.otf");
        SpannableStringBuilder SS = new
SpannableStringBuilder("DriveSafely");
        SS.setSpan (new com.example.drivesafely.CustomTypefaceSpan("",
font), 0, 5, Spanned.SPAN_EXCLUSIVE_INCLUSIVE);
        SS.setSpan (new com.example.drivesafely.CustomTypefaceSpan("",
font2), 5, 11,Spanned.SPAN_EXCLUSIVE_INCLUSIVE);
        txt.setText(SS);
        //Create 4 CardView Variables to be used to display the different
cards
        CardView startCard = findViewById(R.id.start_tracking);
        CardView recentCard = findViewById(R.id.recent_trips);
        CardView settingsCard = findViewById(R.id.settings);
        CardView helpCard = findViewById(R.id.help);
```

```
        //Add Click listener to the cards
        startCard.setOnClickListener(this);
        recentCard.setOnClickListener(this);
        settingsCard.setOnClickListener(this);
        helpCard.setOnClickListener(this);

    }

    /*Create a method for when a card is clicked a new intent is created
and the relevant activity
    is started*/
    @Override
    public void onClick(View view) {

        //Create new intent i
        Intent i;

        //Switch statement used to start the selected cards activity
        switch (view.getId()) {
            case R.id.start_tracking:
                i = new Intent(this, EyeTracking.class);
                startActivity(i);
                break;
            case R.id.recent_trips:
                i = new Intent(this, RecentTrips.class);
                startActivity(i);
                break;
            case R.id.settings:
                i = new Intent(this, Settings.class);
                startActivity(i);
                break;
            case R.id.help:
                i = new Intent(this, Help.class);
                startActivity(i);
                break;
            default:
                break;
        }
    }
}
```

c. Help.java

```java
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

//Class used to create the help activity
public class Help extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Set the view to the activity_help XML files
        setContentView(R.layout.activity_help);
    }
}
```

d. Settings.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 13/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies

import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.SeekBar;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import java.util.ArrayList;
import java.util.Arrays;

//Settings class enables the user to change application settings
public class Settings extends AppCompatActivity {

    //Declare public variables
    public static final String TONE_PREF =null ;
    public static final String TIME_PREF ="Hello" ;
    private static final String TAG = "EyeTracker";
    public static int time;
    public static int tone;

    //Declare private variables
    private static SeekBar sBar;
    private static TextView timeTextView;
    private static int progress;
    private static int position;
    private Spinner toneSpinner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        //Set the layout to the settings XML file
        setContentView(R.layout.activity_settings);

       /* SeekBar, TextView and Spinner are found in the layout using
findViewById.
        This allows the values of each to be changed programmatically.*/
        sBar = findViewById(R.id.seekBarSettings);
        timeTextView = findViewById(R.id.timeText);
        toneSpinner = findViewById(R.id.spinner);

        //Load previously saved time and tone data
        loadTimeData();
        loadToneData();
        Log.i(TAG, "After Loadata Time is  " + time);
        Log.i(TAG, "After Loadata progress is  " + progress);

        //Set the time TextView to display the correct amount of seconds
        timeTextView.setText(time + " Seconds");

        //Switch statement used to set the progress of the seekbar for a
corresponding time
        switch (time) {
            //If time is equal to 2 seconds, set progress to 0
            case 2:
                progress = 0;
                break;
            //If time is equal to 3 seconds, set progress to 1
            case 3:
                progress = 1;
                break;
            //If time is equal to 4 seconds, set progress to 2
            case 4:
                progress = 2;
                break;
        }

        //Set the seekbar progress after the switch statement has executed
        sBar.setProgress(progress);
        Log.i(TAG, "After switch statement progress is  " + progress);


        //Insert the ChooseContact fragment into the settings page
        Fragment fragment;
        fragment = new ChooseContact();
        FragmentManager fm = getSupportFragmentManager();
        FragmentTransaction ft = fm.beginTransaction();
        ft.add(R.id.contactlist, fragment).commit();

        //Create an ArrayList and ArrayAdapter to allow selection of
different tones
        String[] value = {"Missile Alert", "Railroad", "Police Siren"};
        ArrayList<String> arrayList = new
ArrayList<>(Arrays.asList(value));
```

```
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(this,
R.layout.style_spinner, arrayList);
        toneSpinner.setAdapter(arrayAdapter);
        toneSpinner.setSelection(position);
        Log.i(TAG, "spinner  " + position);

        //listener used to set the alarm tone for a certain ArrayList
position
        toneSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
                if (position == 0) {
                    tone = 0;
                }
                if (position == 1) {
                    tone = 1;
                }
                if (position == 2) {
                    tone = 2;
                }
                Settings.position = position;
            }

            //Do nothing if nothing is selected
            @Override
            public void onNothingSelected(AdapterView<?> parentView) {
            }
        });

        //SeekBar listener used to change the value of time and Seekbar
position
        sBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {

                //Set value of time and progress
                time = progress + 2;
                Settings.progress = progress;
                timeTextView.setText(time + " Seconds");
                Log.i(TAG, "Time is saved as " + time);
            }

            //Do nothing
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            //Do nothing
            @Override
```

```
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
    }

    //When the back button is pressed, save the time and tone data to
their shared preferences
    @Override
    public void onBackPressed() {
        saveTimeData();
        saveToneData();
        finish();
    }

    //Method used to save the value of time to its relative shared
preference
    private void saveTimeData() {
        SharedPreferences sharedPreferences =
getSharedPreferences(TIME_PREF, MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putInt(TIME_PREF, time);
        editor.commit();
        Toast.makeText(this, "Data saved", Toast.LENGTH_SHORT).show();
        Log.i(TAG, "(SavedData) Time is " + time);
    }

    //Method used to load the value of time from its relative shared
preference
    public void loadTimeData() {
        SharedPreferences sharedPreferences =
getSharedPreferences(TIME_PREF, MODE_PRIVATE);
        time = sharedPreferences.getInt(TIME_PREF, 3);
        Log.i(TAG, "(LoadData) Time is " + time);

    }
    //Method used to save the tone value to its relative shared preference
    private void saveToneData() {
        SharedPreferences sharedPreferences3 =
getSharedPreferences(TONE_PREF, MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences3.edit();
        editor.putInt(TONE_PREF, position);
        editor.apply();
        Log.i(TAG, "(SavedData) Position is " + position);
    }
    //Method used to load the tone value to its relative shared preference
    private void loadToneData() {
        SharedPreferences sharedPreferences =
getSharedPreferences(TONE_PREF, MODE_PRIVATE);
        position = sharedPreferences.getInt(TONE_PREF, 0);
        Log.i(TAG, "(LoadData) Position " + position);
    }

}
```

e. ChooseContact.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import all necessary dependencies
import android.Manifest;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import java.util.List;

import pub.devrel.easypermissions.AfterPermissionGranted;
import pub.devrel.easypermissions.AppSettingsDialog;
import pub.devrel.easypermissions.EasyPermissions;

import static android.app.Activity.RESULT_OK;
import static android.content.Context.MODE_PRIVATE;


public class ChooseContact extends Fragment implements
EasyPermissions.PermissionCallbacks {

    //Declare private variables
    private static final int RESULT_PICK_CONTACT = 1;
    private static final String TAG = "Settings";

    //Declare public variables
    public static String CONTACT_PREF = "sharedPrefs";
    public static String phoneNumber;
    public static String TEXT = "text";
```

```
    public static String hello;
    private static boolean contactSelected = true;
    private TextView numberTextview, messageTextview;
    private Button selectButton;

    /* Adds permission check when the user clicks to choose a contact. If
the permission has already
    * been accepted, when this method is called, a new intent is started
to allow the user to select
    * theirr desired contact*/
    @AfterPermissionGranted(123)
    private void requestPermissions() {
        String[] perms = {Manifest.permission.READ_CONTACTS};
        if (EasyPermissions.hasPermissions(getContext(), perms)) {
            Intent in = new Intent(Intent.ACTION_PICK,
ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
            startActivityForResult(in, RESULT_PICK_CONTACT);
        } else {
            EasyPermissions.requestPermissions(this, "Permissions need to
be " +
                            "accepted in order to use the application",
                    123, perms);
        }
    }

    //Method used to determine what is done if a permission is denied
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        EasyPermissions.onRequestPermissionsResult(requestCode,
permissions, grantResults, this);

        // If the permission READ_CONTACTS is denied, the below dialog box
is shown
        if (EasyPermissions.somePermissionDenied(this, permissions)) {

            new AlertDialog.Builder(getActivity())
                    .setTitle("Permissions Needed")
                    .setMessage("Permissions need to be accepted in order
to use the application")
                    .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
which) {
                            //The user is returned to the dashboard screen
on clicking OK
                            getActivity().finish();
                        }
                    }).setIcon(android.R.drawable.ic_dialog_alert)
                    .show();
            };
```

```
        }

    //Unused required method
    @Override
    public void onPermissionsGranted(int requestCode, @NonNull
List<String> perms) {

    }

    //Method used if some permission is permanently denied.
    @Override
    public void onPermissionsDenied(int requestCode, @NonNull List<String>
perms) {
         /* If some permission is permanently denied,
           the user is brought to their phone settings, where they can
manually
           accept the apps permissions*/
        if (EasyPermissions.somePermissionPermanentlyDenied(this, perms))
{
            getActivity().finish();
            new AppSettingsDialog.Builder(this).build().show();
        }
    }

    //Unused required method
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    //Method used to inflate the layout of this fragment file
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        {
            // Inflate the layout for this fragment
            return inflater.inflate(R.layout.fragment_contact, container,
false);
        }
    }

    /* Method used to set different button and textview values and call
    required methods to enable user to choose an emergency contact*/
    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {

        //Buttons, TextViews and CardViews are found in the layout using
findViewById
        numberTextview = view.findViewById(R.id.phone);
        selectButton = view.findViewById(R.id.select);
        messageTextview = view.findViewById(R.id.contact_message);

        try {
```

```
            //Load previously saved contact
            loadContact();

            //Set message and button text
            if (phoneNumber.startsWith("No")) {
                contactSelected = false;
                messageTextview.setText("Please Select an Emergency
Contact ");
                selectButton.setText("Select Emergency Contact");

            } else {
                selectButton.setText("Deselect  Contact");
                messageTextview.setText("Your chosen emergency contact
number is ");
                }
        }
        // Catch exception if there is any error loading previously saved
contact
        catch (Exception e) {
            e.printStackTrace();
        }

        // Button listener to allow user to pick their contact
        selectButton.setOnClickListener(v -> {
            if (contactSelected) {
                contactSelected = false;
                TEXT = "";
                phoneNumber = null;
                numberTextview.setText("No Phone number selected");
                selectButton.setText("Select Emergency Contact");
                messageTextview.setText("Please Select an Emergency
Contact ");
                saveContact();
            } else {
                requestPermissions();
            }
        });
    }

    //Method used to check if a contact was picked or not
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if (resultCode == RESULT_OK) {
            switch (requestCode) {
                case RESULT_PICK_CONTACT:
                    contactPicked(data);
                    break;
            }
        } else {
            Toast.makeText(getActivity(), "Failed To pick contact",
Toast.LENGTH_SHORT).show();
        }
```

```
    }

    //Method used to assign the phone number chosen to the phoneNumber
variable and save it.
    private void contactPicked(Intent data) {
        Cursor cursor = null;

        try {
            String phoneNo = null;
            Uri uri = data.getData();
            cursor = getActivity().getContentResolver().query(uri, null,
null, null, null);
            cursor.moveToFirst();
            int phoneIndex =
cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
            phoneNo = cursor.getString(phoneIndex);

            // Define strings to be excluded
            String toRemove = "-";
            String space = " ";
            String plus = "\\+";

            //If statement to remove hyphens, plus symbols and spaces from
the displayed phone number
            if (phoneNo.contains(toRemove) || phoneNo.contains(space) ||
phoneNo.contains(plus)) {
                phoneNo = phoneNo.replaceAll(toRemove, "");
                phoneNo = phoneNo.replaceAll(plus, "");
                phoneNo = phoneNo.replaceAll(space, "");

            }

            //Display chosen contact number
            numberTextview.setText(phoneNo);
            messageTextview.setText("Your chosen emergency contact number
is ");
            phoneNumber = phoneNo;
            contactSelected = true;
            Log.i(TAG, "Phone number is " + phoneNumber);

            //Set text of button and save contact number to its assigned
sharedPreference
            selectButton.setText("Deselect Contact");
            saveContact();
        }

        // Catch any exceptions that may occur when choosing the contact
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    //Method to save chosen contact number into its assigned
```

```
sharePreference
    private void saveContact() {
        SharedPreferences sharedPreferences =
this.getActivity().getSharedPreferences(CONTACT_PREF, MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString(CONTACT_PREF, phoneNumber);
        editor.apply();
        Log.i(TAG, "(SavedData) Phone Number is " + CONTACT_PREF);
    }

    //Method to load chosen contact number from its assigned
sharePreference
    public void loadContact() {
        SharedPreferences sharedPreferences2 =
getActivity().getSharedPreferences(CONTACT_PREF, MODE_PRIVATE);
        phoneNumber= sharedPreferences2.getString(CONTACT_PREF,"No phone
number chosen");
        numberTextview.setText(phoneNumber);
        Log.i(TAG, "(LoadData) Phone is " + phoneNumber);

    }
}
```

f. RecentTrips.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies

import android.annotation.SuppressLint;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import java.text.DecimalFormat;
import java.util.concurrent.TimeUnit;


public class RecentTrips extends AppCompatActivity {

    //Declare private variables
    private static final String TAG = "";
    private TextView timeTextOne;
    private TextView timeTextTwo;
    private TextView timeTextThree;
    private TextView detectTextOne;
    private TextView detectTextTwo;
    private TextView detectTextThree;
    private TextView speedTextOne;
    private TextView speedTextTwo;
    private TextView speedTextThree;
    private TextView distanceTextOne;
    private TextView distanceTextTwo;
    private TextView distanceTextThree;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recent__trips);

        /*CardViews and TextViews are found in the layout using
findViewById.
        This allows the values of each to be changed programmatically.*/
        CardView tripOne = findViewById(R.id.tripOne);
        CardView tripTwo = findViewById(R.id.tripTwo);
```

```
        CardView tripThree = findViewById(R.id.tripThree);
        timeTextOne = findViewById(R.id.timeTextOne);
        timeTextTwo = findViewById(R.id.timeTextTwo);
        timeTextThree = findViewById(R.id.timeTextThree);
        detectTextOne = findViewById(R.id.detectTextOne);
        detectTextTwo = findViewById(R.id.detectTextTwo);
        detectTextThree = findViewById(R.id.detectTextThree);
        speedTextOne = findViewById(R.id.speedTextOne);
        speedTextTwo = findViewById(R.id.speedTextTwo);
        speedTextThree = findViewById(R.id.speedTextThree);
        distanceTextOne = findViewById(R.id.distanceTextOne);
        distanceTextTwo = findViewById(R.id.distanceTextTwo);
        distanceTextThree = findViewById(R.id.distanceTextThree);

        //The last three trips data are loaded when the activity is
started
        loadTripData();
    }

    //Method used to load the data for each trip
    public void loadTripData() {

        //Declare variables
        int timeTrip;
        int detectTrip;
        float speedTrip;
        float distanceTrip;
        float distanceTrip2;
        float distanceTrip3;

        //SharedPreference to load value of time for trip one
        SharedPreferences spTimeT1 =
getSharedPreferences(EyeTracking.TIME_TRIP_ONE_PREF, MODE_PRIVATE);
        timeTrip = spTimeT1.getInt((EyeTracking.TIME_TRIP_ONE_PREF), 0);
        String timeTripOne = stringFormatter(timeTrip);
        timeTextOne.setText(timeTripOne);

        //SharedPreference to load value of time for trip two
        SharedPreferences spTimeT2 =
getSharedPreferences(EyeTracking.TIME_TRIP_TWO_PREF, MODE_PRIVATE);
        timeTrip = spTimeT2.getInt((EyeTracking.TIME_TRIP_TWO_PREF), 0);
        String timeTripTwo = stringFormatter(timeTrip);
        timeTextTwo.setText(timeTripTwo);

        //SharedPreference to load value of time for trip three
        SharedPreferences spTimeT3 =
getSharedPreferences(EyeTracking.TIME_TRIP_THREE_PREF, MODE_PRIVATE);
        timeTrip = spTimeT3.getInt((EyeTracking.TIME_TRIP_THREE_PREF), 0);
        String timeTripThree = stringFormatter(timeTrip);
        timeTextThree.setText(timeTripThree);

        //SharedPreference to load value of detections for trip one
        SharedPreferences spDetect1 =
```

```
getSharedPreferences(EyeTracking.DETECT_TRIP_ONE_PREF, MODE_PRIVATE);
        detectTrip = spDetect1.getInt((EyeTracking.DETECT_TRIP_ONE_PREF),
0);
        detectTextOne.setText(String.valueOf(detectTrip));

        //SharedPreference to load value of detections for trip two
        SharedPreferences spDetect2 =
getSharedPreferences(EyeTracking.DETECT_TRIP_TWO_PREF, MODE_PRIVATE);
        detectTrip = spDetect2.getInt((EyeTracking.DETECT_TRIP_TWO_PREF),
0);
        detectTextTwo.setText(String.valueOf(detectTrip));

        //SharedPreference to load value of detections for trip three
        SharedPreferences spDetect3 =
getSharedPreferences(EyeTracking.DETECT_TRIP_THREE_PREF, MODE_PRIVATE);
        detectTrip =
spDetect3.getInt((EyeTracking.DETECT_TRIP_THREE_PREF), 0);
        detectTextThree.setText(String.valueOf(detectTrip));

        //SharedPreference to load value of speed for trip one
        SharedPreferences spSpeed1 =
getSharedPreferences(EyeTracking.SPEED_TRIP_ONE_PREF, MODE_PRIVATE);
        speedTrip = spSpeed1.getFloat((EyeTracking.SPEED_TRIP_ONE_PREF),
0);
        speedTextOne.setText(String.format("%.0f", speedTrip) + " km/h");
        Log.i(TAG, "speed one is " + EyeTracking.SPEED_TRIP_ONE_PREF);

        //SharedPreference to load value of speed for trip two
        SharedPreferences spSpeed2 =
getSharedPreferences(EyeTracking.SPEED_TRIP_TWO_PREF, MODE_PRIVATE);
        speedTrip = spSpeed2.getFloat((EyeTracking.SPEED_TRIP_TWO_PREF),
0);
        speedTextTwo.setText(String.format("%.0f", speedTrip) + " km/h");

        //SharedPreference to load value of speed for trip three
        SharedPreferences spSpeed3 =
getSharedPreferences(EyeTracking.SPEED_TRIP_THREE_PREF, MODE_PRIVATE);
        speedTrip = spSpeed3.getFloat((EyeTracking.SPEED_TRIP_THREE_PREF),
0);
        speedTextThree.setText(String.format("%.0f", speedTrip) + "
km/h");

        //SharedPreference to load value of distance for trip one
        SharedPreferences spDistance1 =
getSharedPreferences(EyeTracking.DISTANCE_TRIP_ONE_PREF, MODE_PRIVATE);
        distanceTrip =
spDistance1.getFloat((EyeTracking.DISTANCE_TRIP_ONE_PREF), 0);
        Log.i(TAG, "Dist1" + distanceTrip);
        distanceTextOne.setText(new
DecimalFormat("#.#").format(distanceTrip) + " km");

        //SharedPreference to load value of distance for trip two
        SharedPreferences spDistance2 =
```

```
getSharedPreferences(EyeTracking.DISTANCE_TRIP_TWO_PREF, MODE_PRIVATE);
        distanceTrip2 =
spDistance2.getFloat((EyeTracking.DISTANCE_TRIP_TWO_PREF), 0);
        distanceTextTwo.setText(new
DecimalFormat("#.#").format(distanceTrip2) + " km");

        //SharedPreference to load value of distance for trip three
        SharedPreferences spDistance3 =
getSharedPreferences(EyeTracking.DISTANCE_TRIP_THREE_PREF, MODE_PRIVATE);
        distanceTrip3 =
spDistance3.getFloat((EyeTracking.DISTANCE_TRIP_THREE_PREF), 0);
        distanceTextThree.setText(new
DecimalFormat("#.#").format(distanceTrip3) + " km");

    }

    //Method used to properly format the time data
    private String stringFormatter(int trip) {
        @SuppressLint("DefaultLocale")
        String hms = String.format("%02d:%02d:%02d",
                TimeUnit.MILLISECONDS.toHours(trip),
                TimeUnit.MILLISECONDS.toMinutes(trip) -

TimeUnit.HOURS.toMinutes(TimeUnit.MILLISECONDS.toHours(trip)), // The
change is in this line
                TimeUnit.MILLISECONDS.toSeconds(trip) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(trip)));
        return hms;
    }


}
```

g. EyeTracking.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 16/03/2021
 */

package com.example.drivesafely;

//Import necessary dependencies

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.PictureInPictureParams;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.graphics.Typeface;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Handler;
import android.os.Looper;
import android.os.SystemClock;
import android.os.VibrationEffect;
import android.os.Vibrator;
import android.telephony.SmsManager;
import android.util.Log;
import android.util.Rational;
import android.view.View;
import android.widget.Button;
import android.widget.Chronometer;
import android.widget.FrameLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
```

```
import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.MultiProcessor;
import com.google.android.gms.vision.Tracker;
import com.google.android.gms.vision.face.Face;
import com.google.android.gms.vision.face.FaceDetector;

import java.io.IOException;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

import pub.devrel.easypermissions.AfterPermissionGranted;
import pub.devrel.easypermissions.AppSettingsDialog;
import pub.devrel.easypermissions.EasyPermissions;

//Class used to implement the eye/face tracking and other associated
methods.
public final class EyeTracking extends AppCompatActivity implements
EasyPermissions.PermissionCallbacks {

    //Declare public variables
    public static final String TIME_TRIP_ONE_PREF = "tripOne";
    public static final String TIME_TRIP_TWO_PREF = "tripTWo";
    public static final String TIME_TRIP_THREE_PREF = "tripThree";
    public static final String DETECT_TRIP_ONE_PREF = "detectTripOne";
    public static final String DETECT_TRIP_TWO_PREF = "detectTripTWo";
    public static final String DETECT_TRIP_THREE_PREF = "detectTripThree";
    public static final String SPEED_TRIP_ONE_PREF = "speedTripOne";
    public static final String SPEED_TRIP_TWO_PREF = "speedTripTWo";
    public static final String SPEED_TRIP_THREE_PREF = "speedTripThree";
    public static final String DISTANCE_TRIP_ONE_PREF = "distanceTripOne";
    public static final String DISTANCE_TRIP_TWO_PREF = "distanceTripTWo";
    public static final String DISTANCE_TRIP_THREE_PREF =
"distanceTripThree";
    public static final String TRIP_VALUE_PREF = "trip3";
    //Declare private variables
    private static final String TAG = "Log";
    public static TextView distance;
    public static int tripValue;
    public static boolean inPipFragment;
    public int detections = 0;
    public Chronometer journeyTimer;
    public int numbOfBlinks;
    public int flag = 0;
    public boolean yellowAlert = true;
    public boolean facePresent = false;
    public FragmentManager fm;
    public FragmentTransaction ft;
    public Fragment fragment;
    public AlertDialog greenAlertDialog;
    public boolean isOnPause = false;
    private CameraSource mCameraSource = null;
    private MediaPlayer mp;
```

```
    private Vibrator vibrator;
    private CameraSourcePreview mPreview;
    private GraphicOverlay mGraphicOverlay;
    private FrameLayout frameLayout;
    private int alarmTime;
    private int alarmTone;
    private String emergencyContact;
    private ConstraintLayout.LayoutParams params;
    private CardView previewCard, speedCard, journeyCard, distanceCard,
detectionCard, logoCard;
    private TextView detectionsTextview;
    private Timer yellowAlertTimer;
    private TimerTask yellowAlertTimeTask;
    private boolean greenFirstTime = true;
    private int headAngle = 0;
    private float head_angle;
    private float updateHeadAngle;

    @SuppressLint({"ClickableViewAccessibility", "CutPasteId"})
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        //Layout is set to activity_eye_tracking
        setContentView(R.layout.activity_eye_tracking);

       /* Buttons, TextViews and CardViews are found in the layout using
findViewById.
        This allows the values of each to be changed programmatically.*/
        Button openMapsBtn = findViewById(R.id.mapButton);
        ToggleButton previewBtn = findViewById(R.id.previewButton);
        mPreview = findViewById(R.id.preview);
        mGraphicOverlay = findViewById(R.id.faceOverlay);
        frameLayout = findViewById(R.id.previewFrame);
        speedCard = findViewById(R.id.speedCard);
        journeyCard = findViewById(R.id.journeyCard);
        distanceCard = findViewById(R.id.DistanceCard);
        detectionCard = findViewById(R.id.DetectionsCard);
        params = (ConstraintLayout.LayoutParams)
frameLayout.getLayoutParams();
        previewCard = findViewById(R.id.previewCard);
        distance = findViewById(R.id.distancemain);
        detectionsTextview = findViewById(R.id.detections_text);
        logoCard = findViewById(R.id.logoCard);
        previewBtn.setTextOff("Turn Preview Off");
        previewBtn.setTextOn("Turn Preview On");
        journeyTimer = findViewById(R.id.journeyTimer);

        //Run methods shown
        showPositionDialog();
        loadTripValue();
```

```
//Start the speedometerFragment and add its view to fragment_speedo in
activity_eye_tracking
        fragment = new SpeedometerFragment();
        fm = getSupportFragmentManager();
        ft = fm.beginTransaction();
        ft.replace(R.id.fragmentSpeedo, fragment).commit();

        //Initialise chronometer and start
        journeyTimer.setBase(SystemClock.elapsedRealtime());
        journeyTimer.start();

        // For logging and testing purposes
        Log.i(TAG, "Before calling shared preferences " + alarmTime);

        // Get the value stored in Settings.TIME_PREF . If no value found,
default is 3 seconds
        SharedPreferences sharedPreferencesTime =
getSharedPreferences(Settings.TIME_PREF, MODE_PRIVATE);
        alarmTime = sharedPreferencesTime.getInt(Settings.TIME_PREF, 3) *
1000;
        Log.i(TAG, "Alarm time is: " + alarmTime);

        // Get the value stored in Settings.TONE_PREF . If no value found,
default is tone 0 - Missile Alert
        SharedPreferences sharedPreferencesTone =
getSharedPreferences(Settings.TONE_PREF, MODE_PRIVATE);
        alarmTone = sharedPreferencesTone.getInt(Settings.TONE_PREF, 0);

        //Attempt to get the chosen emergency contact. If no contact
chosen. exception is thrown
        try {
            // For logging and testing purposes
            Log.i(TAG, "Before calling shared number " +
emergencyContact);

            SharedPreferences sharedPreferencesContact =
getSharedPreferences(ChooseContact.CONTACT_PREF, MODE_PRIVATE);
            emergencyContact =
sharedPreferencesContact.getString(ChooseContact.CONTACT_PREF, "");

            // For logging and testing purposes
            Log.i(TAG, "Number is  " + emergencyContact);
        } catch (Exception e) {
            e.printStackTrace();
        }

        //Start Google Maps in picture-in-picture mode when openMapsBtn is
clicked
        openMapsBtn.setOnClickListener(v -> {
            startPIP();
            Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse("http://maps.google.com/maps?"));
```

```
            intent.setPackage("com.google.android.apps.maps");
            startActivity(intent);
        });


        previewBtn.setOnCheckedChangeListener((buttonView, isChecked) -> {
            /*If previewBtn is checked, hide the preview and its
associated card view
             and display toast message. Show the drivesafely logo*/
            if (isChecked) {
                mPreview.setVisibility(View.INVISIBLE);
                Toast.makeText(getApplicationContext(), "Increase
Brightness to maximum for higher accuracy", Toast.LENGTH_SHORT).show();
                previewCard.setVisibility(View.INVISIBLE);
                logoCard.setVisibility(View.VISIBLE);

                /*If previewBtn is not checked, hide the DriveSafely logo.
The preview and its
                 * associated card view are visible.*/
            } else {
                mPreview.setVisibility(View.VISIBLE);
                previewCard.setVisibility(View.VISIBLE);
                logoCard.setVisibility(View.INVISIBLE);
            }
        });

        //Declare an audio manager and set the volume to half the max
volume
        final AudioManager audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
        int volume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC) / 2;
        audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, volume,
0);

        // If volume is set to 0 by user, display Toast message
        if (volume == 0) {
            Toast.makeText(getApplicationContext(), "Volume is MUTE",
Toast.LENGTH_LONG).show();
        }

        //Call method requestPermissions
        requestPermissions();

    }

    // Method to start checking if a user is blinking enough
    private void checkForBlinks() {
        yellowAlertTimer = new Timer();

        /*This method is used to trigger a yellow alert is the number of
blinksin a minute is less than 4*/
```

```
        yellowAlertTimeTask = new TimerTask() {
            @Override
            public void run() {
                if (facePresent) {
                    yellowAlertTimer = new Timer();
                    if (numbOfBlinks < 5 && yellowAlert && !isOnPause) {
                        yellowAlert = false;
                        yellowAlertBox();
                    }
                }
                numbOfBlinks = 0;
            }

        };
        //The timer is scheduled to run every 60 seconds with an initial
delay of 60 seconds
        yellowAlertTimer.schedule(yellowAlertTimeTask, 60 * 1000, 60 *
1000);
    }

    // Method to stop checking if a user is blinking enough
    private void stopCheckForBlinks() {

        yellowAlertTimeTask.cancel();
        yellowAlertTimer.cancel();
        yellowAlertTimer.purge();
    }

    // Method to assign a new value to updateHeadAngle every 3 seconds
    private void calcHeadAngle() {
        Timer headAngleTimer = new Timer();

        /*This class is used to trigger a yellow alert is the number of
blinks
        in a minute is less than 4*/
        TimerTask angleTimerTask = new TimerTask() {
            @Override
            public void run() {
                updateHeadAngle = head_angle;
            }
        };
        //The timer is scheduled to run every 3 seconds with an initial
delay of 3 seconds
        headAngleTimer.schedule(angleTimerTask, 3 * 1000, 3 * 1000);
    }


    /* Method used to call the dialog box which shows users how they
should position their phone
     for optimal eye and face tracking*/
    private void showPositionDialog() {
        //Create new dialog and initialise views
```

```
        final Dialog dialog = new Dialog(EyeTracking.this);
        dialog.setCanceledOnTouchOutside(false);
        ConstraintLayout xp = findViewById(R.id.backgroundDialog);
        dialog.setContentView(R.layout.placement_dialog);
        final Button button = dialog.findViewById(R.id.okBtn);

        //Make the dialog box visible
        dialog.show();


        //Dismiss dialog when button is clicked
        button.setOnClickListener((v) -> {
            dialog.dismiss();
        });

    }

    //Method used to change alter the layout when picture-in-picture (PIP)
mode is activated
    @Override
    public void onPictureInPictureModeChanged(boolean
isInPictureInPictureMode, Configuration newConfig) {

        super.onPictureInPictureModeChanged(isInPictureInPictureMode);

        /*If the application is changed to PIP mode, different buttons and
cards are set to invisible
        The camera preview is set to occupy the full width and height of
the PIP window*/
        if (isInPictureInPictureMode) {
            mPreview.setVisibility(View.VISIBLE);
            previewCard.setVisibility(View.VISIBLE);
            logoCard.setVisibility(View.INVISIBLE);
            previewCard.setRadius(0);
            ConstraintLayout.LayoutParams layPra = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.MATCH_PARENT,
ConstraintLayout.LayoutParams.MATCH_PARENT);
            frameLayout.setLayoutParams(layPra);
            distanceCard.setVisibility(View.INVISIBLE);
            detectionCard.setVisibility(View.INVISIBLE);
            journeyCard.setVisibility(View.INVISIBLE);
            speedCard.setVisibility(View.INVISIBLE);

        }
        //If PIP mode is exited, the application is reset to it's normal
view
        else {
            distanceCard.setVisibility(View.VISIBLE);
            detectionCard.setVisibility(View.VISIBLE);
            journeyCard.setVisibility(View.VISIBLE);
            speedCard.setVisibility(View.VISIBLE);
            previewCard.setRadius(15);
            frameLayout.setLayoutParams(params);
```

```
        }
    }

    // Method used to start PIP mode
    private void startPIP() {
        // Set inPipFragment to true for use in speedometerFragment.java
        inPipFragment = true;

        // Set the parameter for the PIP mode. A view with a 2:1 aspect
ratio is used
        PictureInPictureParams pip = new PictureInPictureParams.Builder()
                .setAspectRatio(new Rational(1, 2))
                .build();

        // Picture in picture mode is started with the configured
parameters
        enterPictureInPictureMode(pip);


    }

    // Method used to request permissions
    @AfterPermissionGranted(123)
    private void requestPermissions() {
        //Array of Strings holds a list of the necessary permissions
        String[] perms = {Manifest.permission.CAMERA,
Manifest.permission.SEND_SMS,
                Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.INTERNET,
                Manifest.permission.VIBRATE,
Manifest.permission.READ_CONTACTS};

        //If permissions are granted, createCameraSource() is called
        if (EasyPermissions.hasPermissions(this, perms)) {
            createCameraSource();
        }
        //If permissions are not granted the user is displayed a message
        else {
            EasyPermissions.requestPermissions(this, "Permissions need to
be accepted in order to use the application", 123, perms);
        }
    }

    // Method used to determine what is done if a permission is denied
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {

        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        EasyPermissions.onRequestPermissionsResult(requestCode,
permissions, grantResults, this);
```

```
        // If some permission is denied, a dialog box is shown with the
below info
        if (EasyPermissions.somePermissionDenied(this, permissions)) {
            new AlertDialog.Builder(EyeTracking.this)
                    .setTitle("Permissions Needed")
                    .setMessage("Permissions need to be accepted in order
to use the application")
                    .setPositiveButton(android.R.string.yes, (dialog,
which) -> {

                        //The user is returned to the dashboard screen on
clicking OK
                        finish();
                    }).setIcon(android.R.drawable.ic_dialog_alert)
                    .show();
        }
    }

    // Unused required method
    @Override
    public void onPermissionsGranted(int requestCode, @NonNull
List<String> perms) {
    }

    // Method used if some permission is permanently denied
    @Override
    public void onPermissionsDenied(int requestCode, @NonNull List<String>
perms) {
        /* If some permission is permanently denied,
           the user is brought to their phone settings, where they can
manually
           accept the apps permissions*/
        if (EasyPermissions.somePermissionPermanentlyDenied(this, perms))
{
            finish();
            new AppSettingsDialog.Builder(this).build().show();
        }
    }


    // Method used to initialise the camera source and Face Detector
    private void createCameraSource() {

        Context context = getApplicationContext();
        /*Create new FaceDetector called detector with parameters chosen
        Refer to
https://developers.google.com/android/reference/com/google/android/gms/vis
ion/face/FaceDetector.Builder*/
        FaceDetector detector = new FaceDetector.Builder(context)
                .setTrackingEnabled(true)
                .setLandmarkType(FaceDetector.ALL_LANDMARKS)
                .setClassificationType(FaceDetector.ALL_CLASSIFICATIONS)
                .setMode(FaceDetector.ACCURATE_MODE)
```

```
                .setProminentFaceOnly(true)
                .build();


        //The detectors processor is set to GraphicFaceTrackerFactory
        detector.setProcessor(
                new MultiProcessor.Builder<>(new
GraphicFaceTrackerFactory())
                        .build());

        mCameraSource = new CameraSource.Builder(context, detector)

                .setRequestedPreviewSize(1600, 1200)
                .setFacing(CameraSource.CAMERA_FACING_FRONT)
                .setRequestedFps(60.0f)
                .setAutoFocusEnabled(true)
                .build();

    }

    // Called when the activity starts interacting with user or after
resuming from onPause
    @Override
    protected void onResume() {
        super.onResume();
        isOnPause = false;
        checkForBlinks();
        calcHeadAngle();
        //createCameraSource();

        startCameraSource();
    }

    // Method called when the application enters into a paused state
    @Override
    protected void onPause() {
        super.onPause();

        /*onPause() is called when PIP mode is activated. The app
functions should not be altered
        when PIP is active*/
        if (isInPictureInPictureMode()) {

        }
        /*If the app is not in PIP mode, the application shuts down the
camera source and stops
         * checking for user blinks. */
        else {
            stopCheckForBlinks();
            stopPlaying();
            isOnPause = true;
            mPreview.stop();
            inPipFragment = false;
```

```
        }
    }

    // Method called just before the activity is destroyed.
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mCameraSource != null) {
            mCameraSource.release();
        }
        stopPlaying();
        saveTrip();
        saveTripValue();
        SpeedometerFragment.distance = 0;
    }

    // Method used to store the value of the next trip.
    private void saveTripValue() {
        if (tripValue >= 3) {
            tripValue = 1;
        } else {
            tripValue++;
        }

        // Store the tripValue in TRIP_VALUE_PREF
        SharedPreferences sharedPreferencesSaveTripValue =
getSharedPreferences(TRIP_VALUE_PREF, MODE_PRIVATE);
        SharedPreferences.Editor editor =
sharedPreferencesSaveTripValue.edit();
        editor.putInt(TRIP_VALUE_PREF, tripValue);
        editor.commit();
        Log.i(TAG, "(SavedData) TripValue is " + tripValue);

    }

    // Method used to load the previously saved value of tripValue.
    public void loadTripValue() {
        SharedPreferences sharedPreferencesLoadTripValue =
getSharedPreferences(TRIP_VALUE_PREF, MODE_PRIVATE);
        tripValue = sharedPreferencesLoadTripValue.getInt(TRIP_VALUE_PREF,
1);
        Log.i(TAG, "(LoadData) TripValue is  " + tripValue);
    }

    /*Method used to save the current trips information such as time,
average speed,
    distance travelled and number of detections*/
    private void saveTrip() {

        // Initialise Strings for each of the 4 pieces of data
        String timePref = "";
        String detectPref = "";
```

```
        String speedPref = "";
        String distancePref = "";

        /* For a certain tripValue, the 4 data points are stored in the
        respective Shared preferences*/
        if (tripValue == 1 || tripValue == 2 || tripValue == 3) {
            if (tripValue == 1) {
                timePref = TIME_TRIP_ONE_PREF;
                detectPref = DETECT_TRIP_ONE_PREF;
                speedPref = SPEED_TRIP_ONE_PREF;
                distancePref = DISTANCE_TRIP_ONE_PREF;
            }

            if (tripValue == 2) {
                timePref = TIME_TRIP_TWO_PREF;
                detectPref = DETECT_TRIP_TWO_PREF;
                speedPref = SPEED_TRIP_TWO_PREF;
                distancePref = DISTANCE_TRIP_TWO_PREF;
            }


            if (tripValue == 3) {
                timePref = TIME_TRIP_THREE_PREF;
                detectPref = DETECT_TRIP_THREE_PREF;
                speedPref = SPEED_TRIP_THREE_PREF;
                distancePref = DISTANCE_TRIP_THREE_PREF;
            }

        }

        // The overall journey time, elapsedTime is saved in timePref
        SharedPreferences sharedPreferencesTime =
getSharedPreferences(timePref, MODE_PRIVATE);
        SharedPreferences.Editor editorTime =
sharedPreferencesTime.edit();
        int elapsedTime = (int) (SystemClock.elapsedRealtime() -
journeyTimer.getBase());
        editorTime.putInt(timePref, elapsedTime);
        editorTime.apply();

        // The number of detections are stored in detectPref
        SharedPreferences sharedPreferencesDetect =
getSharedPreferences(detectPref, MODE_PRIVATE);
        SharedPreferences.Editor editorDetect =
sharedPreferencesDetect.edit();
        editorDetect.putInt(detectPref, detections);
        editorDetect.apply();

        // The average journey speed is calculated
        float avgSpeed = (float) (SpeedometerFragment.distance /
elapsedTime * 3600 * 1000);
        Log.i(TAG, "avgspeed" + avgSpeed);
```

```
        // The average journey speed is saved in speedPref
        SharedPreferences sharedPreferencesSpeed =
getSharedPreferences(speedPref, MODE_PRIVATE);
        SharedPreferences.Editor editorSpeed =
sharedPreferencesSpeed.edit();
        editorSpeed.putFloat(speedPref, avgSpeed);
        editorSpeed.apply();

        // The journey distance is saved in distancePref
        SharedPreferences sharedPreferencesDistance =
getSharedPreferences(distancePref, MODE_PRIVATE);
        SharedPreferences.Editor editorDistance =
sharedPreferencesDistance.edit();
        editorDistance.putFloat(distancePref, (float)
SpeedometerFragment.distance);
        editorDistance.apply();

    }


    /* Method used to start showing the camera preview.
    Throws exception if unable to start the camera.
    */
    private void startCameraSource() {
        if (mCameraSource != null) {
            try {
                mPreview.start(mCameraSource, mGraphicOverlay);

            } catch (IOException e) {
                Log.e(TAG, "Unable to start camera source.", e);
                mCameraSource.release();
                mCameraSource = null;
            }
        }
    }

    // Method used to play the user selected audio, when an alert is
triggered.
    public void playMedia() {
        stopPlaying();

        if (alarmTone == 0) {
            mp = MediaPlayer.create(this, R.raw.missle_alert);
        }

        if (alarmTone == 1) {
            mp = MediaPlayer.create(this, R.raw.railroad);
        }
        if (alarmTone == 2) {
            mp = MediaPlayer.create(this, R.raw.police);
        }
        mp.setLooping(true);
        mp.start();
```

```
    }

    // Method used to stop playing the user selected audio, normally when
the alert is dismissed
    public void stopPlaying() {
        if (mp != null) {
            mp.stop();
            mp.release();
            mp = null;
        }
    }

    // Method used to vibrate the phone when an alert is triggered.
    public void vibrate() {
        vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
        long[] pattern = {800, 800, 800, 800};
        VibrationEffect vibe = VibrationEffect.createWaveform(pattern, 0);
        vibrator.vibrate(vibe);
    }

    // Method used to stop the phone vibrating
    public void stopVibrating() {
        vibrator.cancel();
    }

    // Method to create alert dialog box for when the user's face is
missing.
    public void greenAlertBox() {
        // New dialog box created with the below parameters
        greenAlertDialog = new AlertDialog.Builder(EyeTracking.this)
                .setTitle("Face Missing")
                .setMessage("DriveSafely has detected that your face is
missing. " +
                        "Try increasing screen brightness if it is dark.")
                .setIcon(R.drawable.ic_baseline_lens__green_24)
                .show();
        TextView textView =
greenAlertDialog.findViewById(android.R.id.message);
        Typeface face = Typeface.createFromAsset(getAssets(),
"fonts/roboto_light.xml");
        textView.setTypeface(face);
    }

    /*Method used to create alert dialog box for when the user does not
blink
    enough times in a given minute or if their head is beginning to nod.
This method will also
    open Google Maps and display a list of nearby petrol stations to the
    user when they dismiss the dialog box
    */
    public void yellowAlertBox() {

        runOnUiThread(() -> {
```

```
            // The media player plays a police siren and the phone
vibrates
            mp = MediaPlayer.create(this, R.raw.police_siren_2);
            mp.setLooping(true);
            mp.start();
            vibrate();

            // The value of detections is increased and updated
            detections++;
            detectionsTextview.setText(String.valueOf(detections));

            // New alert dialog created with the below parameters
            AlertDialog yellowAlertDialog;
            yellowAlertDialog = new AlertDialog.Builder(EyeTracking.this)
                    .setTitle("Tiredness Detected")
                    .setMessage("DriveSafely has detected that you may be
very tired." +
                            " A list of petrol stations will now be
displayed. " +
                            "Consider pulling over and drinking a
coffee.")
                    .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
which) {
                            stopPlaying();
                            flag = 0;
                        }
                    }).setIcon(R.drawable.ic_baseline_lens__yellow_24)
                    .show();
            TextView textView =
yellowAlertDialog.findViewById(android.R.id.message);
            Typeface face = Typeface.createFromAsset(getAssets(),
"fonts/roboto_light.xml");
            textView.setTypeface(face);

            /* On dismissing the dialog, the media player stops,
               the phone stops vibrating and numbOfBlinks is reset to 0*/
            yellowAlertDialog.setOnDismissListener(dialog -> {
                stopPlaying();
                stopVibrating();
                flag = 0;
                yellowAlert = true;
                // numbOfBlinks = 0;

                /*PIP mode is started and the user is presented
                with a list/map of nearby petrol stations
                */
                startPIP();
                Uri gmmIntentUri = Uri.parse("geo:0,0?q=Petrol");
                Intent mapIntent = new Intent(Intent.ACTION_VIEW,
gmmIntentUri);
                mapIntent.setPackage("com.google.android.apps.maps");
```

```
                startActivity(mapIntent);
            });
        });
    }

    /*Method used to create alert dialog box for when the user's eyes are
closed for the user
    selected amount of time. A SMS is sent to the user's emergency contact
if they are deemed to be
    asleep
     */
    public void redAlertBox() {
        runOnUiThread(() -> {
            // The selected tone is played and the phone vibrates
            playMedia();
            vibrate();

            // The value of detections is increased and updated
            detections++;
            detectionsTextview.setText(String.valueOf(detections));

            // New alert dialog created with the below parameters
            AlertDialog redAlertDialog;
            redAlertDialog = new AlertDialog.Builder(EyeTracking.this)
                    .setTitle("Eyes Closed Detected")
                    .setMessage("DriveSafely suspects that you have fallen
asleep." +
                            " An SMS message has been sent to your
emergency contact. " +
                            "It is advisable to pull over and stop.")
                    .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
which) {
                            stopPlaying();
                            flag = 0;
                        }
                    }).setIcon(R.drawable.ic_baseline_lens_24)
                    .show();
            TextView textView =
redAlertDialog.findViewById(android.R.id.message);
            Typeface face = Typeface.createFromAsset(getAssets(),
"fonts/roboto_light.xml");
            textView.setTypeface(face);

            /* On dismissing the dialog, the media player stops
               and the phone stops vibrating */
            redAlertDialog.setOnDismissListener(dialog -> {
                stopPlaying();
                stopVibrating();
                flag = 0;
            });
        });
```

```
    }


    // Graphic Face Tracker created
    private class GraphicFaceTrackerFactory implements
MultiProcessor.Factory<Face> {
        @Override
        public Tracker<Face> create(Face face) {
            return new GraphicFaceTracker(mGraphicOverlay);
        }
    }

    private class GraphicFaceTracker extends Tracker<Face> {

        // Variables initialised
        private final GraphicOverlay mOverlay;
        private final FaceGraphic mFaceGraphic;
        private int state_i = 0;
        private long lastLowToHighState = 0;
        private boolean firstTime = true;
        private boolean eyesClosed;
        private int dips;

        GraphicFaceTracker(GraphicOverlay overlay) {
            mOverlay = overlay;
            mFaceGraphic = new FaceGraphic(overlay);
        }

        /* Method used to add the bounding box and calls the eyeTracking
method.
           Called when the face position is updated .
        */
        @Override
        public void onUpdate(FaceDetector.Detections<Face>
detectionResults, Face face) {

            mOverlay.add(mFaceGraphic);
            mFaceGraphic.updateFace(face);

            if (flag == 0) {
                eyeTracking(face);
            }

        }

        // Called when the face is detected as being missing
        @Override
        public void onMissing(FaceDetector.Detections<Face>
detectionResults) {

            facePresent = false;

            // Bounding box removed from camera preview and numbOfBlinks
```

```
reset to 0
            mOverlay.remove(mFaceGraphic);
            numbOfBlinks = 0;
            Log.i(TAG, "face_present is" + facePresent);

            // Method faceMissing is called
            faceMissing();
        }

        // Method used to show a dialog box if a user's face has not been
detected for 2 seconds
        public void faceMissing() {
            new Handler(Looper.getMainLooper()).post(()
                    -> new CountDownTimer(1000, 1000) {

                // Unused necessary method
                @Override
                public void onTick(long millisUntilFinished) {
                }

                // Called when the CountDownTimer finished.
                @Override
                public void onFinish() {
                    if (!facePresent && greenFirstTime) {
                        greenFirstTime = false;
                        greenAlertBox();
                    } else {
                        // Do nothing
                    }
                }
            }.start());
        }

        // Method used to remove the bounding box from the screen
        @Override
        public void onDone() {
            mOverlay.remove(mFaceGraphic);
        }

        // Method used to set variable eyesClose to true
        private void onEyesClosed() {
            eyesClosed = true;
        }

        // Method used to count the number of blinks
        private void onEyesOpened() {

            if (eyesClosed) {
                numbOfBlinks++;

                // Log.i(TAG, "numb of blinks" + numbOfBlinks);
            }
            eyesClosed = false;
```

```
        }

        // Method used to send a message to the user's chosen emergency
contact
        private void sendSMS() {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(emergencyContact, null,
getString(R.string.Emergency_message), null, null);
        }

        // Method used to carry out operations based on the user's eyes
        private void eyeTracking(Face face) {

            /*
            Gets the probability of each eye being open.
            Returns a value between 0.0 and 1.0 giving a probability that
the face's left eye is open.
             */
            float leftEye = face.getIsLeftEyeOpenProbability();
            float rightEye = face.getIsRightEyeOpenProbability();

            // Removes the greenAlertDialog is condition is false
            if (!greenFirstTime) {
                greenAlertDialog.dismiss();
                greenFirstTime = true;
            }

            /*
            Returns the rotation of the face about the horizontal axis of
the image.
            Positive euler x is when the face turns toward the
            top of the of the image that is being processed.
            */
            head_angle = face.getEulerX();
            // Log.i(TAG,"EulerX "+head_angle);
            // Log.i(TAG,"X is  "+x);

            if (head_angle < (updateHeadAngle - 15) &&
SystemClock.elapsedRealtime() - lastLowToHighState > 500) {
                dips++;
                // Used to prevent the function from continuously firing
                lastLowToHighState = SystemClock.elapsedRealtime();
                Log.i(TAG, "p is  " + dips);
                if (dips == 5) {
                    yellowAlertBox();
                    dips = 0;
                }
            }

            // Sets the threshold value for when the eyes are deemed to be
closed
            float threshold = 0.2f;
```

```
            //Log.i(TAG,"Left eye"+leftEye);

            // Checks if left or right eyes is considered open
            if (leftEye > threshold && rightEye > threshold) {
                state_i = 0;
                firstTime = true;
                facePresent = true;

                onEyesOpened();

            }

            // If both eyes are deemed closed a number of operations are
carried out
            else {

                //onEyesClosed method is called
                onEyesClosed();

                // If state_i is = 0 the elapsedRealTime value is stored
in lasLowToHighState
                if (state_i == 0) {
                    lastLowToHighState = SystemClock.elapsedRealtime();

                }

 /* If firstTime is True and the current elapsedRealTime -
lastLowToHighState is greater than the user selected alarm time, the
redAlertBox method is called and the application attempts to send a SMS to
the chosen emergency contact, if one is selected
                 */
                if (firstTime && SystemClock.elapsedRealtime() -
lastLowToHighState > alarmTime) {

                    // Used to prevent the function from continuously
firing
                    lastLowToHighState = SystemClock.elapsedRealtime();
                    firstTime = false;
                    redAlertBox();
                    try {
                        sendSMS();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    flag = 1;
                }
                state_i = 1;

            }
        }
    }
}
```

h. SpeedometerFragment.java

```
/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies

import android.annotation.SuppressLint;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import com.google.android.gms.location.LocationRequest;

import java.text.DecimalFormat;

//SpeedometerFragment is used to calculate distance travelled and current
speed
public class SpeedometerFragment extends Fragment implements
LocationListener {

    //Declare private variables
    private static final String TAG = "";
    private static final long INTERVAL = 1000 * 2;
    private static final long FASTEST_INTERVAL = 1000 * 1;
    //Declare public variables
    public static double distance = 0;
    public static LocationManager lm;
    private Location mCurrentLocation, lStart, lEnd;
    private LocationRequest mLocationRequest;
    private TextView speedText;
    private boolean pip;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        SetupLocation();

        //Inflate the fragment_speedometer layout
        View view = inflater.inflate(R.layout.fragment_speedometer,
container, false);
        speedText = view.findViewById(R.id.speedTextview);

        return view;

    }

    //Method used to get the setup the location requests and initialise
the location manager
    @SuppressLint("MissingPermission")
    private void SetupLocation() {

        //Location request parameters are set and high accuracy is
requested
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(INTERVAL);
        mLocationRequest.setFastestInterval(FASTEST_INTERVAL);

mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

        //Location updates requested
        lm = (LocationManager)
getActivity().getSystemService(Context.LOCATION_SERVICE);
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
this);
        }

    //Method used to carry out operations when a user's location is
changed
    @Override
    public void onLocationChanged(Location location) {

        //Used to calculate distance
        mCurrentLocation = location;
        if (lStart == null) {
            lStart = mCurrentLocation;
        }
        lEnd = mCurrentLocation;

        //Call method to update the live values of distance and speed
        updateUI();
    }
```

```
 //The live feed of Distance and Speed are being set in the method below .
    private void updateUI() {

        //Calculate the distance value
        distance = distance + (lStart.distanceTo(lEnd) / 1000.00);

        if (distance < 100) {
            //Display distance with 1 decimal place if less than 100km
            EyeTracking.distance.setText(new
DecimalFormat("#.#").format(distance) + " km");
        } else {
            //Display distance with 0 decimal places if greater than 100km
            EyeTracking.distance.setText(new
DecimalFormat("#").format(distance) + " km");
        }
        Log.i(TAG, "Distance  " + distance);

        /*Calculating the speed with getSpeed method returns speed in m/s
so we are converting it
        to km/h*/
        float nCurrentSpeed = mCurrentLocation.getSpeed() * 3.6f;
        //Update the speed textview with 0 decimal place format
        speedText.setText(String.format("%.0f", nCurrentSpeed) + " km/h");
        Log.i(TAG, "Speed  " + nCurrentSpeed);

        lStart = lEnd;
     }

    //Method used to stop getting location updates when the app is closed
    @Override
    public void onStop() {
        super.onStop();
        lm.removeUpdates(this);
        Log.i(TAG, "Location stopped");
    }

    @Override
    public void onPause() {
        pip = EyeTracking.inPipFragment;
        Log.i(TAG, "PIP is" + pip);
        super.onPause();
        if (pip) {

        } else {

            lm.removeUpdates(this);
            Log.i(TAG, "Location paused");
        }
    }

}
```

i. CameraSourcePreview.java

```
/*
This file is an adapted version of the googlesamples/android-vision file
found here:
https://tinyurl.com/2nx2ejum
 */

/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.res.Configuration;
import android.util.AttributeSet;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.ViewGroup;

import com.google.android.gms.common.images.Size;
import com.google.android.gms.vision.CameraSource;

import java.io.IOException;

//This class is used to enable a camera preview to be shown to the user
public class CameraSourcePreview extends ViewGroup {

    //Declare private variables
    private static final String TAG = "CameraSourcePreview";
    private final Context mContext;
    private final SurfaceView mSurfaceView;
    private boolean mStartRequested;
    private boolean mSurfaceAvailable;
    private CameraSource mCameraSource;
    private GraphicOverlay mOverlay;

    //Create new CameraSourcePreview
    public CameraSourcePreview(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        mStartRequested = false;
        mSurfaceAvailable = false;
        mSurfaceView = new SurfaceView(context);
        mSurfaceView.getHolder().addCallback(new SurfaceCallback());
        addView(mSurfaceView);
```

```
    }

    //Start the camera source
    public void start(CameraSource cameraSource) throws IOException {
        if (cameraSource == null) {
            stop();
        }

        mCameraSource = cameraSource;

        if (mCameraSource != null) {
            mStartRequested = true;
            startIfReady();
        }
    }

    public void start(CameraSource cameraSource, GraphicOverlay overlay)
throws IOException {
        mOverlay = overlay;
        start(cameraSource);
    }

    //Stop showing the camera preview
    public void stop() {
        if (mCameraSource != null) {
            mCameraSource.stop();
        }
    }

    //Start the camera if ready with mOverlay parameters
    @SuppressLint("MissingPermission")
    private void startIfReady() throws IOException {
        if (mStartRequested && mSurfaceAvailable) {
            mCameraSource.start(mSurfaceView.getHolder());
            if (mOverlay != null) {
                Size size = mCameraSource.getPreviewSize();
                int min = Math.min(size.getWidth(), size.getHeight());
                int max = Math.max(size.getWidth(), size.getHeight());
                if (isPortraitMode()) {
                    mOverlay.setCameraInfo(min, max,
mCameraSource.getCameraFacing());
                } else {
                    mOverlay.setCameraInfo(max, min,
mCameraSource.getCameraFacing());
                }
                mOverlay.clear();
            }
            mStartRequested = false;
        }
    }

   /* This method find dimensions of the screen and adapts the preview
width and height
```

```
      to ensure the correct aspect ratio is kept at all times*/
    @Override
    protected void onLayout(boolean changed, int left, int top, int right,
int bottom) {

        int previewWidth = 320;
        int previewHeight = 240;
        if (mCameraSource != null) {
            Size size = mCameraSource.getPreviewSize();
            if (size != null) {
                previewWidth = size.getWidth();
                previewHeight = size.getHeight();
            }
        }

        // Swap width and height sizes when in portrait, since it will be
rotated 90 degrees
        if (isPortraitMode()) {
            int tmp = previewWidth;
            previewWidth = previewHeight;
            previewHeight = tmp;
        }

        final int viewWidth = right - left;
        final int viewHeight = bottom - top;

        int childWidth;
        int childHeight;
        int childXOffset = 0;
        int childYOffset = 0;
        float widthRatio = (float) viewWidth / (float) previewWidth;
        float heightRatio = (float) viewHeight / (float) previewHeight;

        /*To fill the view with the camera preview, while also preserving
the correct aspect ratio,
          it is usually necessary to slightly oversize the child and to
crop off portions along one
          of the dimensions.  We scale up based on the dimension requiring
the most correction, and
          compute a crop offset for the other dimension.*/
        if (widthRatio > heightRatio) {
            childWidth = viewWidth;
            childHeight = (int) ((float) previewHeight * widthRatio);
            childYOffset = (childHeight - viewHeight) / 2;
        } else {
            childWidth = (int) ((float) previewWidth * heightRatio);
            childHeight = viewHeight;
            childXOffset = (childWidth - viewWidth) / 2;
        }

        for (int i = 0; i < getChildCount(); ++i) {
            // One dimension will be cropped.  We shift child over or up
by this offset and adjust
```

```
            // the size to maintain the proper aspect ratio.
            getChildAt(i).layout(
                    -1 * childXOffset, -1 * childYOffset,
                    childWidth - childXOffset, childHeight -
childYOffset);
        }

        try {
            startIfReady();
        } catch (IOException e) {
            Log.e(TAG, "Could not start camera source.", e);
        }
    }

    //Checks to see is screen in portait or landscape mode
    private boolean isPortraitMode() {
        int orientation =
mContext.getResources().getConfiguration().orientation;
        if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
            return false;
        }
        if (orientation == Configuration.ORIENTATION_PORTRAIT) {
            return true;
        }

        Log.d(TAG, "isPortraitMode returning false by default");
        return false;
    }

    private class SurfaceCallback implements SurfaceHolder.Callback {
        @Override
        public void surfaceCreated(SurfaceHolder surface) {
            mSurfaceAvailable = true;
            try {
                startIfReady();
            } catch (IOException e) {
                Log.e(TAG, "Could not start camera source.", e);
            }
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder surface) {
            mSurfaceAvailable = false;
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {
        }
    }
}
```

j. FaceGraphic.java

```
/*
This file is an adapted version of the googlesamples/android-vision file
found here:
https://tinyurl.com/22kpsafw
 */

/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */

package com.example.drivesafely;

//Import necessary dependencies

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

import com.google.android.gms.vision.face.Face;
import com.google.android.gms.vision.face.Landmark;

import java.util.List;

//Class used to draw the bounding box around the users face and landmark
positions(for testing only)
class FaceGraphic extends GraphicOverlay.Graphic {

    //Declare private variables
    private static final float FACE_POSITION_RADIUS = 10.0f;
    private static final float BOX_STROKE_WIDTH = 5.0f;
    private static final int[] COLOR_CHOICES = {
            Color.BLUE,
            Color.CYAN,
            Color.GREEN,
            Color.MAGENTA,
            Color.RED,
            Color.WHITE,
            Color.YELLOW
    };
    private static int mCurrentColorIndex = 0;
    private final Paint mFacePositionPaint;
    private final Paint mBoxPaint;
    private volatile Face mFace;

    FaceGraphic(GraphicOverlay overlay) {
        super(overlay);

        mCurrentColorIndex = (mCurrentColorIndex + 1) %
```

```
COLOR_CHOICES.length;
        final int selectedColor = COLOR_CHOICES[mCurrentColorIndex];

        //Define parameters for colour and style etc. of bounding box
        mFacePositionPaint = new Paint();
        mFacePositionPaint.setColor(selectedColor);
        mBoxPaint = new Paint();
        mBoxPaint.setColor(selectedColor);
        mBoxPaint.setStyle(Paint.Style.STROKE);
        mBoxPaint.setStrokeWidth(BOX_STROKE_WIDTH);
    }

    void updateFace(Face face) {
        mFace = face;
        postInvalidate();
    }

    @Override
    public void draw(Canvas canvas) {
        Face face = mFace;

        if (face == null) {
            return;
        }

// Draws a circle at the position of the detected face, with the face's
track id below.
        float x = translateX(face.getPosition().x + face.getWidth() / 2);
        float y = translateY(face.getPosition().y + face.getHeight() / 2);
        canvas.drawCircle(x, y, FACE_POSITION_RADIUS, mFacePositionPaint);


        // Draws a bounding box around the face.
        float xOffset = scaleX(face.getWidth() / 2.0f);
        float yOffset = scaleY(face.getHeight() / 2.0f);
        float left = x - xOffset;
        float top = y - yOffset;
        float right = x + xOffset;
        float bottom = y + yOffset;
        canvas.drawRect(left, top, right, bottom, mBoxPaint);


        Paint paint = new Paint();
        paint.setColor(Color.GREEN);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5);


        if ((contains(face.getLandmarks(), 0) != 99)
                && (contains(face.getLandmarks(), 1) != 99)
                && (contains(face.getLandmarks(), 2) != 99)
                && (contains(face.getLandmarks(), 3) != 99)
                && (contains(face.getLandmarks(), 4) != 99)
```

```
                && (contains(face.getLandmarks(), 5) != 99)
                && (contains(face.getLandmarks(), 6) != 99)
                && (contains(face.getLandmarks(), 7) != 99)
                && (contains(face.getLandmarks(), 8) != 99)
                && (contains(face.getLandmarks(), 9) != 99)
                && (contains(face.getLandmarks(), 10) != 99)
                && (contains(face.getLandmarks(), 11) != 99)) {

//            Get position of each landmark and draw a circle at the
detected x,y coordinate
//            int cBottomMouthX;
//            int cBottomMouthY;
//
//            cBottomMouthX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
0)).getPosition().x);
//            cBottomMouthY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
0)).getPosition().y);
//            canvas.drawCircle(cBottomMouthX, cBottomMouthY, 5, paint);
//
//
//            int cLeftCheekX;
//            int cLeftCheekY;
//
//            cLeftCheekX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
1)).getPosition().x);
//            cLeftCheekY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
1)).getPosition().y);
//            canvas.drawCircle(cLeftCheekX, cLeftCheekY, 5, paint);
//
//
//            int cLeftEarX;
//            int cLeftEarY;
//
//            cLeftEarX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
2)).getPosition().x);
//            cLeftEarY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
2)).getPosition().y);
//            canvas.drawCircle(cLeftEarX, cLeftEarY, 5, paint);
//
//            int cLeftEarTipX;
//            int cLeftEarTipY;
//
//            cLeftEarTipX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
3)).getPosition().x);
//            cLeftEarTipY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
```

```
3)).getPosition().y);
//            canvas.drawCircle(cLeftEarTipX, cLeftEarTipY, 5, paint);
//
//            int cLeftEyeX;
//            int cLeftEyeY;
//
//            cLeftEyeX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
4)).getPosition().x);
//            cLeftEyeY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
4)).getPosition().y);
//
//            canvas.drawCircle(cLeftEyeX, cLeftEyeY, 5, paint);
//
//            int cLeftMouthX;
//            int cLeftMouthY;
//
//            cLeftMouthX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
5)).getPosition().x);
//            cLeftMouthY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
5)).getPosition().y);
//            canvas.drawCircle(cLeftMouthX, cLeftMouthY, 5, paint);
//
//            int cNoseBaseX;
//            int cNoseBaseY;
//
//            cNoseBaseX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
6)).getPosition().x);
//            cNoseBaseY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
6)).getPosition().y);
//            canvas.drawCircle(cNoseBaseX, cNoseBaseY, 5, paint);
//
//            int cRightCheekX;
//            int cRightCheekY;
//
//            cRightCheekX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
7)).getPosition().x);
//            cRightCheekY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
7)).getPosition().y);
//            canvas.drawCircle(cRightCheekX, cRightCheekY, 5, paint);
//
//            int cRightEarTipX;
//            int cRightEatTipY;
//
//            cRightEarTipX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
```

```
8)).getPosition().x);
//            cRightEatTipY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
8)).getPosition().y);
//            canvas.drawCircle(cRightEarTipX, cRightEatTipY, 5, paint);
//
//            int cRightEarX;
//            int cRightEarY;
//
//            cRightEarX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
9)).getPosition().x);
//            cRightEarY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
9)).getPosition().y);
//            canvas.drawCircle(cRightEarX, cRightEarY, 5, paint);
//
//            int cRighteyeX;
//            int cRighteyeY;
//
//            cRighteyeX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
10)).getPosition().x);
//            cRighteyeY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
10)).getPosition().y);
//
//            canvas.drawCircle(cRighteyeX, cRighteyeY, 5, paint);
//
//            int cRightMouthX;
//            int cRightMouthY;
//
//            cRightMouthX = (int)
translateX(face.getLandmarks().get(contains(face.getLandmarks(),
11)).getPosition().x);
//            cRightMouthY = (int)
translateY(face.getLandmarks().get(contains(face.getLandmarks(),
11)).getPosition().y);
//
//            canvas.drawCircle(cRightMouthX, cRightMouthY, 5, paint);
        }
    }
 //Method used to aid in drawing the facial landmarks
    private int contains(List<Landmark> list, int name) {
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).getType() == name) {
                return i;
            }
        }
        return 99;
    }
}
```

k. GraphicOverlay.java

```
/*
This file is an adapted version of the googlesamples/android-vision file
found here:
https://tinyurl.com/5evn5ne7
 */

/**
 * Author : Reece Gavin
 * ID Number : 17197589
 * Date last modified: 17/02/2021
 */


package com.example.drivesafely;

//Import necessary dependencies
import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.View;

import com.google.android.gms.vision.CameraSource;

import java.util.HashSet;
import java.util.Set;

/**
 * A view which renders a series of custom graphics to be overlayed on top
of an associated preview
 * (i.e., the camera preview).  The creator can add graphics objects,
update the objects, and remove
 * them, triggering the appropriate drawing and invalidation within the
view.<p>
 *
/* * Supports scaling and mirroring of the graphics relative the camera's
preview properties.  The
 * idea is that detection items are expressed in terms of a preview size,
but need to be scaled up
 * to the full view size, and also mirrored in the case of the front-
facing camera.<p>
 *
 * Associated {@link Graphic} items should use the following methods to
convert to view coordinates
 * for the graphics that are drawn:
 * <ol>
 * <li>{@link Graphic#scaleX(float)} and {@link Graphic#scaleY(float)}
adjust the size of the
 * supplied value from the preview scale to the view scale.</li>
 * <li>{@link Graphic#translateX(float)} and {@link
```

```
Graphic#translateY(float)} adjust the coordinate
 * from the preview's coordinate system to the view coordinate
system.</li>
 * </ol>
 */

public class GraphicOverlay extends View {

    //Define private variables
    private final Object mLock = new Object();
    private final Set<Graphic> mGraphics = new HashSet<>();
    private int mPreviewWidth;
    private float mWidthScaleFactor = 1.0f;
    private int mPreviewHeight;
    private float mHeightScaleFactor = 1.0f;
    private int mFacing = CameraSource.CAMERA_FACING_FRONT;

    public GraphicOverlay(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    //Removes all graphics from the overlay.
    public void clear() {
        synchronized (mLock) {
            mGraphics.clear();
        }
        postInvalidate();
    }

    //Adds a graphic to the overlay.
    public void add(Graphic graphic) {
        synchronized (mLock) {
            mGraphics.add(graphic);
        }
        postInvalidate();
    }

    //Removes a graphic from the overlay.
    public void remove(Graphic graphic) {
        synchronized (mLock) {
            mGraphics.remove(graphic);
        }
        postInvalidate();
    }

    /* Sets the camera attributes for size and facing direction, which
informs how to transform
     * image coordinates later.
     */
    public void setCameraInfo(int previewWidth, int previewHeight, int
facing) {
        synchronized (mLock) {
            mPreviewWidth = previewWidth;
```

```
            mPreviewHeight = previewHeight;
            mFacing = facing;
        }
        postInvalidate();
    }

    // Draws the overlay with its associated graphic objects.
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        synchronized (mLock) {
            if ((mPreviewWidth != 0) && (mPreviewHeight != 0)) {
                mWidthScaleFactor = (float) canvas.getWidth() / (float)
mPreviewWidth;
                mHeightScaleFactor = (float) canvas.getHeight() / (float)
mPreviewHeight;
            }

            for (Graphic graphic : mGraphics) {
                graphic.draw(canvas);
            }
        }
    }


     /* Base class for a custom graphics object to be rendered within the
graphic overlay.  Subclass
     * this and implement the {@link Graphic#draw(Canvas)} method to
define the
     * graphics element.  Add instances to the overlay using {@link
GraphicOverlay#add(Graphic)}.
     */
    public static abstract class Graphic {
        private final GraphicOverlay mOverlay;

        public Graphic(GraphicOverlay overlay) {
            mOverlay = overlay;
        }


        public abstract void draw(Canvas canvas);


         /* Adjusts a horizontal value of the supplied value from the
preview scale to the view
          * scale*/

        public float scaleX(float horizontal) {
            return horizontal * mOverlay.mWidthScaleFactor;
        }
```

```
        //Adjusts a vertical value of the supplied value from the preview
scale to the view scale.

        public float scaleY(float vertical) {
            return vertical * mOverlay.mHeightScaleFactor;
        }


         /* Adjusts the x coordinate from the preview's coordinate system
to the view coordinate
          * system. */

        public float translateX(float x) {
            if (mOverlay.mFacing == CameraSource.CAMERA_FACING_FRONT) {
                return mOverlay.getWidth() - scaleX(x);
            } else {
                return scaleX(x);
            }
        }


         /* Adjusts the y coordinate from the preview's coordinate system
to the view coordinate
          * system*/

        public float translateY(float y) {
            return scaleY(y);
        }

        public void postInvalidate() {
            mOverlay.postInvalidate();
        }
    }

}
```

Appendix C: XML Source Code

a. activity_splash_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/redBG"
    tools:context=".SplashScreen">


    <!--ImageView used to display DriveSafely logo-->
    <ImageView
        android:id="@+id/animationImage"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="150dp"
        android:layout_marginEnd="10dp"
        android:src="@drawable/logo_trans"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

b. activity_dashboard.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/ll"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#fcfcfc"

    android:orientation="vertical"

    tools:context="com.example.drivesafely.Dashboard">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/redBG"
        android:minHeight="60dp"
        android:theme="?attr/actionBarTheme"
        tools:ignore="MissingConstraints" />

    <TextView
        android:id="@+id/custom_fonts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Test Location"
        android:textColor="#FFFF"
        android:textSize="14dp"
        app:layout_constraintBottom_toBottomOf="@id/toolbar"
        app:layout_constraintEnd_toEndOf="@id/rightGuideline"
        app:layout_constraintStart_toStartOf="@id/leftGuideline"
        app:layout_constraintTop_toTopOf="@id/toolbar" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_gravity="center"
        android:orientation="horizontal"
        app:layout_constraintBottom_toTopOf="@id/horizontalGuide2"
        app:layout_constraintEnd_toStartOf="@+id/rightGuideline"
        app:layout_constraintStart_toStartOf="@+id/leftGuideline"
        app:layout_constraintTop_toTopOf="@id/horizontalGuide1">

        <androidx.cardview.widget.CardView
            android:id="@+id/start_tracking"
            android:layout_width="160dp"
            android:layout_height="match_parent"
```

```
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginBottom="10dp"
            android:clickable="true"
            android:foreground="?android:attr/selectableItemBackground"
            app:cardCornerRadius="5dp">


            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center"
                android:orientation="vertical">

                <ImageView
                    android:layout_width="64dp"
                    android:layout_height="64dp"
                    android:background="@drawable/circlepurple"
                    android:padding="10dp"

android:src="@drawable/ic_baseline_photo_camera_24_white" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="10dp"
                    android:text="Start Eye Tracking"
                    android:textStyle="bold" />

                <View
                    android:layout_width="match_parent"
                    android:layout_height="1dp"
                    android:layout_margin="10dp"
                    android:background="@color/lightgray" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center"
                    android:padding="5dp"
                    android:text="Begin your trip"
                    android:textColor="@android:color/darker_gray" />

            </LinearLayout>
        </androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
            android:id="@+id/recent_trips"
            android:layout_width="160dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
```

```
            android:layout_marginBottom="10dp"
            android:clickable="true"

            android:foreground="?android:attr/selectableItemBackground"
            app:cardCornerRadius="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center"
                android:orientation="vertical">

                <ImageView
                    android:layout_width="64dp"
                    android:layout_height="64dp"
                    android:layout_marginTop="15dp"
                    android:background="@drawable/circlepink"
                    android:padding="10dp"


android:src="@drawable/ic_baseline_format_list_bulleted_24" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="10dp"
                    android:text="Recent Trips"
                    android:textStyle="bold" />

                <View
                    android:layout_width="match_parent"
                    android:layout_height="1dp"
                    android:layout_margin="10dp"
                    android:background="@color/lightgray" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center"
                    android:padding="5dp"
                    android:text="View your recent trips and stats"
                    android:textColor="@android:color/darker_gray" />

            </LinearLayout>
        </androidx.cardview.widget.CardView>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_gravity="center"
```

```
android:orientation="horizontal"
app:layout_constraintBottom_toTopOf="@id/horizontalGuide4"
app:layout_constraintEnd_toEndOf="@id/rightGuideline"
app:layout_constraintStart_toStartOf="@+id/leftGuideline"
app:layout_constraintTop_toTopOf="@+id/horizontalGuide3">

<androidx.cardview.widget.CardView
    android:id="@+id/settings"
    android:layout_width="160dp"
    android:layout_height="match_parent"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:clickable="true"
    android:foreground="?android:attr/selectableItemBackground"
    app:cardCornerRadius="5dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">

        <ImageView
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:background="@drawable/circlegreen"
            android:padding="10dp"
            android:src="@drawable/ic_baseline_settings_24" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="Settings"
            android:textStyle="bold" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_margin="10dp"
            android:background="@color/lightgray" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="5dp"
            android:text="Change your settings"
            android:textColor="@android:color/darker_gray" />

    </LinearLayout>
```

```
        </androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
            android:id="@+id/help"
            android:layout_width="160dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginBottom="10dp"
            android:clickable="true"
            android:foreground="?android:attr/selectableItemBackground"
            app:cardCornerRadius="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center"
                android:orientation="vertical">

                <ImageView
                    android:layout_width="64dp"
                    android:layout_height="64dp"
                    android:background="@drawable/circleyellow"
                    android:padding="10dp"
                    android:src="@drawable/ic_baseline_help_24" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="10dp"
                    android:text="Help"
                    android:textStyle="bold" />

                <View
                    android:layout_width="match_parent"
                    android:layout_height="1dp"
                    android:layout_margin="10dp"
                    android:background="@color/lightgray" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center"
                    android:padding="5dp"
                    android:text="Get help using the app"
                    android:textColor="@android:color/darker_gray" />

            </LinearLayout>
        </androidx.cardview.widget.CardView>

    </LinearLayout>
```

```
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/leftGuideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.05" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/rightGuideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.95" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalGuide1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.15" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalGuide2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.45" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalGuide3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.6" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalGuide4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.9" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/textguide"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.09" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

c. activity_help.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#1E9C9696"
    tools:context=".Help">

    <!--CardView to hold alarm delay slider text-->
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horGuideline2"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline1">

        <!--TextView to display Alarm Delay Slider Heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:fontFamily="@font/roboto_light"
            android:text="Alarm Delay Slider"
            android:textColor="#020000"
            android:textSize="20dp" />

        <!--TextView to show info about alarm delay slider-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:layout_marginTop="5dp"
            android:layout_marginEnd="5dp"
            android:fontFamily="@font/roboto_light"
            android:gravity="center"
            android:text="@string/alarm_slider" />
    </androidx.cardview.widget.CardView>

    <!--ImageView to show picture of alarm delay slider-->
    <ImageView
        android:layout_width="0dp"
        android:layout_height="0dp"

        android:src="@drawable/alarm"
```

```
        app:layout_constraintBottom_toBottomOf="@id/horGuideline4"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline3" />

    <!--CardView to hold alarm tone selector text-->
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horGuideline6"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline5">

        <!--TextView to display Alarm Tone Selector Heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:fontFamily="@font/roboto_light"
            android:text="Alarm Tone Selector"
            android:textColor="#020000"
            android:textSize="20dp" />

        <!--TextView to show info about alarm tone selector-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="5dp"
            android:layout_marginTop="5dp"
            android:layout_marginEnd="5dp"
            android:fontFamily="@font/roboto_light"
            android:gravity="center"
            android:text="@string/alarm_tone" />
    </androidx.cardview.widget.CardView>

    <!--ImageView to show picture of alarm tone selector-->
    <ImageView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@drawable/alarm_tone_"
        app:layout_constraintBottom_toBottomOf="@id/horGuideline8"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline7"/>

    <!--CardView to hold Contact Selector text-->
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="0dp"
```

```
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horGuideline10"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline9">

        <!--TextView to display Contact Selector Heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:fontFamily="@font/roboto_light"
            android:text="Contact Selector"
            android:textColor="#020000"
            android:textSize="20dp" />

        <!--TextView to show info about contact selector-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="5dp"
            android:fontFamily="@font/roboto_light"
            android:gravity="center"
            android:text="@string/contact_help" />
    </androidx.cardview.widget.CardView>


    <!--ImageView to show picture of contact selector-->
    <ImageView
        android:layout_width="0dp"
        android:layout_height="0dp"

        android:src="@drawable/contact"
        app:layout_constraintBottom_toBottomOf="@id/horGuideline12"
        app:layout_constraintEnd_toEndOf="@id/Vertguideline2"
        app:layout_constraintStart_toStartOf="@id/Vertguideline"
        app:layout_constraintTop_toTopOf="@id/horGuideline11" />


    <!--Guidelines used to create the constraint layout-->
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/Vertguideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.1" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/Vertguideline2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.9" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.05" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.15" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.3" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.17" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.32" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.42" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.44" />
```

```
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.55" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.57" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.67" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.69" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horGuideline12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.95" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

d. activity_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Settings">

    <!--TextView to hold Settings Heading-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:fontFamily="@font/roboto_light"
        android:gravity="center"
        android:text="Settings"
        android:textColor="#3C3B3D"
        android:textSize="30dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@id/settingscard1"
        app:layout_constraintEnd_toEndOf="@id/horizontalcard2"
        app:layout_constraintStart_toStartOf="@id/horizontalcard1" />

    <!--CardView for Alarm Delay-->
    <androidx.cardview.widget.CardView
        android:id="@id/settingscard1"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="100dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toTopOf="@id/verticalcard3"
        app:layout_constraintLeft_toLeftOf="@id/horizontalcard1"
        app:layout_constraintRight_toRightOf="@id/horizontalcard2"
        app:layout_constraintTop_toTopOf="@id/verticalcard1">

        <!--Seekbar to change alarm delay time-->
        <SeekBar
            android:id="@+id/seekBarSettings"
            android:layout_width="300dp"
            android:layout_height="50dp"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:max="2"
            android:progress="1"
            android:progressDrawable="@drawable/seek_bar"
            android:thumb="@drawable/seek_thumb" />

        <!--TextView to display Alarm Delay text-->
        <TextView
```

```
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:fontFamily="@font/roboto_light"
            android:text="@string/alarm_delay"
            android:textColor="#3C3B3D"
            android:textSize="30sp" />

        <!--TextView to display Alarm Delay value-->
        <TextView
            android:id="@+id/timeText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="60dp"
            android:fontFamily="@font/roboto_light"
            android:text="3 Seconds"
            android:textColor="#3C3B3D"
            android:textSize="30sp" />

        <!--ImageView to display time icon-->
        <ImageView
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:layout_gravity="right"
            android:padding="10dp"
            android:src="@drawable/ic_baseline_timer_24" />
    </androidx.cardview.widget.CardView>

    <!--CardView for Alarm tone-->
    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="100dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toTopOf="@id/verticalcard5"
        app:layout_constraintLeft_toLeftOf="@id/horizontalcard1"
        app:layout_constraintRight_toRightOf="@id/horizontalcard2"
        app:layout_constraintTop_toTopOf="@id/verticalcard4">

        <!--TextView to display Alarm Tone text-->
        <TextView
            android:id="@+id/textView12"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:fontFamily="@font/roboto_light"
            android:text="Alarm Tone"
            android:textColor="#3C3B3D"
            android:textSize="30sp" />
```

```
        <!--Spinner used to allow user to select alarm tone-->
        <Spinner
            android:id="@+id/spinner"
            android:layout_width="300dp"
            android:layout_height="40dp"
            android:layout_gravity="center"
            android:layout_marginTop="20dp"
            android:background="@drawable/style_spinner"
            android:padding="5dp" />

        <!--ImageView to display tone icon-->
        <ImageView
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:layout_gravity="right"
            android:padding="10dp"
            android:src="@drawable/ic_baseline_music_note_24" />

    </androidx.cardview.widget.CardView>

    <!--CardView for Emergency contact-->
    <androidx.cardview.widget.CardView
        android:id="@id/contactlist"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="100dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/verticalcard7"
        app:layout_constraintEnd_toStartOf="@id/horizontalcard2"
        app:layout_constraintStart_toStartOf="@id/horizontalcard1"
        app:layout_constraintTop_toBottomOf="@id/verticalcard6">

        <!--TextView to display Emergency Contact text-->
        <TextView
            android:id="@+id/textView12"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:layout_marginRight="0dp"
            android:fontFamily="@font/roboto_light"
            android:text="Emergency Contact"
            android:textColor="#3C3B3D"
            android:textSize="32sp" />


    </androidx.cardview.widget.CardView>
```

```
<!--Guidelines used for constraint layout-->
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.37" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.14" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.6" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.96" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.37" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.57" />


<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verticalcard4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.4" />
```

```
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalcard1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.1" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/horizontalcard2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.9" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

e. fragment_contact.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bottomGuide2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".SplashScreen">

    <!--TextView to ask user to select an emergency contact-->
    <TextView
        android:id="@+id/contact_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/roboto_light"
        android:gravity="center"
        android:text=""
        android:textColor="#3C3B3D"
        android:textSize="20sp"
        app:layout_constraintBottom_toTopOf="@+id/bottomGuide6"
        app:layout_constraintEnd_toEndOf="@id/verticalGuide2"
        app:layout_constraintStart_toStartOf="@id/verticalGuide1"
        app:layout_constraintTop_toTopOf="@id/bottomGuide5" />

    <!--TextView to show emergency contact number-->
    <TextView
        android:id="@+id/phone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/roboto_light"
        android:gravity="center"
        android:text=""
        android:textColor="#3C3B3D"
        android:textSize="20sp"
        app:layout_constraintBottom_toTopOf="@+id/bottomGuide3"
        app:layout_constraintEnd_toEndOf="@id/verticalGuide2"
        app:layout_constraintStart_toStartOf="@id/verticalGuide1"
        app:layout_constraintTop_toTopOf="@id/bottomGuide4" />

    <!--Button to select an emergency contact-->
    <Button
        android:id="@+id/select"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@drawable/style_contact_btn"
        android:elevation="5dp"
        android:fontFamily="@font/roboto_light"
```

```
        android:text="Choose Emergency Contact"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="22sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline13"
        app:layout_constraintEnd_toEndOf="@id/guideline2_cbtn"
        app:layout_constraintStart_toStartOf="@id/guideline_cbtn"
        app:layout_constraintTop_toTopOf="@id/bottomGuide1" />

    <!--Guidelines used for constraint layout-->
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.9" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bottomGuide1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.7" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/verticalGuide1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.15" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/verticalGuide2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.85" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bottomGuide3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.6" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bottomGuide4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <androidx.constraintlayout.widget.Guideline
```

```
        android:id="@+id/bottomGuide5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.25" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bottomGuide6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.4" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline_cbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.05109489" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2_cbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.95" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

f. activity_recent_trips.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".RecentTrips">

    <!--CardView used to show information for trip one-->
    <androidx.cardview.widget.CardView
        android:id="@+id/tripOne"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:clickable="true"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horizGuideline2"
        app:layout_constraintEnd_toEndOf="@id/verGuideline2"
        app:layout_constraintStart_toStartOf="@id/verGuideline1"
        app:layout_constraintTop_toTopOf="@id/horizGuideline1">

        <!--TextView used to display trip one heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:fontFamily="@font/roboto_light"
            android:text="Trip One"
            android:textSize="24dp" />

        <!--TextView used to display time heading for trip one -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
            android:layout_marginTop="50dp"
            android:fontFamily="@font/roboto_light"
            android:text="Time"
            android:textSize="18dp" />

        <!--TextView used to display time value for trip one -->
        <TextView
            android:id="@id/timeTextOne"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
```

```
            android:layout_marginTop="80dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display time icon for trip one -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="left"
            android:layout_marginStart="10dp"
            android:layout_marginTop="51dp"
            android:src="@drawable/ic_baseline_timer_blue_24" />

        <!--TextView used to display average speed heading for trip one --
>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="40dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Average Speed"
            android:textSize="18dp" />

        <!--TextView used to display average speed value for trip one -->
        <TextView
            android:id="@+id/speedTextOne"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="45dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display speed icon for trip one -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="bottom"
            android:layout_marginStart="10dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_speed_red_24" />

        <!--TextView used to display distance heading for trip one -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
```

```
            android:layout_marginEnd="20dp"
            android:fontFamily="@font/roboto_light"
            android:text="Distance"
            android:textSize="18dp" />

        <!--TextView used to display distance value for trip one -->
        <TextView
            android:id="@+id/distanceTextOne"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="80dp"
            android:layout_marginEnd="22dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display distance icon for trip one -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
            android:layout_marginEnd="105dp"
            android:src="@drawable/ic_baseline_directions_car_orange_24"
/>

        <!--TextView used to display detections heading for trip one -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="230dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Detections"
            android:textSize="18dp" />

        <!--ImageView used to display detections icon for trip one -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="bottom"
            android:layout_marginStart="200dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_tag_faces_pink_24" />

        <!--TextView used to display detections value for trip one -->
        <TextView
            android:id="@+id/detectTextOne"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
```

```
            android:layout_marginStart="270dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />


    </androidx.cardview.widget.CardView>

    <!--CardView used to show information for trip two-->
    <androidx.cardview.widget.CardView
        android:id="@+id/tripTwo"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:clickable="true"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horizGuideline4"
        app:layout_constraintEnd_toEndOf="@id/verGuideline2"
        app:layout_constraintStart_toStartOf="@id/verGuideline1"
        app:layout_constraintTop_toTopOf="@id/horizGuideline3">

        <!--TextView used to display trip two heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:fontFamily="@font/roboto_light"
            android:text="Trip Two"
            android:textSize="24dp" />

        <!--TextView used to display time heading for trip two -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
            android:layout_marginTop="50dp"
            android:fontFamily="@font/roboto_light"
            android:text="Time"
            android:textSize="18dp" />

        <!--TextView used to display time value for trip two -->
        <TextView
            android:id="@id/timeTextTwo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
            android:layout_marginTop="80dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
```

```
            android:textSize="18dp" />

        <!--ImageView used to display time icon for trip two -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="left"
            android:layout_marginStart="10dp"
            android:layout_marginTop="51dp"
            android:src="@drawable/ic_baseline_timer_blue_24" />

        <!--TextView used to display average speed heading for trip two --
>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="40dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Average Speed"
            android:textSize="18dp" />

        <!--TextView used to display average speed value for trip two -->
        <TextView
            android:id="@+id/speedTextTwo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="45dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display speed icon for trip two -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="bottom"
            android:layout_marginStart="10dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_speed_red_24" />

        <!--TextView used to display distance heading for trip two -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
            android:layout_marginEnd="20dp"
            android:fontFamily="@font/roboto_light"
            android:text="Distance"
```

```
            android:textSize="18dp" />

        <!--TextView used to display distance value for trip two -->
        <TextView
            android:id="@+id/distanceTextTwo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="80dp"
            android:layout_marginEnd="22dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display distance icon for trip two -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
            android:layout_marginEnd="105dp"
            android:src="@drawable/ic_baseline_directions_car_orange_24"
/>

        <!--TextView used to display detections heading for trip two -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="230dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Detections"
            android:textSize="18dp" />

        <!--ImageView used to display detections icon for trip two -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="bottom"
            android:layout_marginStart="200dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_tag_faces_pink_24" />

        <!--TextView used to display detections value for trip two -->
        <TextView
            android:id="@+id/detectTextTwo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="270dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
```

```
            android:text=""
            android:textSize="18dp" />

    </androidx.cardview.widget.CardView>

    <!--CardView used to show information for trip three-->
    <androidx.cardview.widget.CardView
        android:id="@+id/tripThree"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:clickable="true"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/horizguideline6"
        app:layout_constraintEnd_toEndOf="@id/verGuideline2"
        app:layout_constraintStart_toStartOf="@id/verGuideline1"
        app:layout_constraintTop_toTopOf="@id/horizGuideline5">

        <!--TextView used to display trip three heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:fontFamily="@font/roboto_light"
            android:text="Trip Three"
            android:textSize="24dp" />

        <!--TextView used to display time heading for trip three -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
            android:layout_marginTop="50dp"
            android:fontFamily="@font/roboto_light"
            android:text="Time"
            android:textSize="18dp" />

        <!--TextView used to display time value for trip three -->
        <TextView
            android:id="@id/timeTextThree"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="40dp"
            android:layout_marginTop="80dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display time icon for trip three -->
        <ImageView
```

```
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="left"
            android:layout_marginStart="10dp"
            android:layout_marginTop="51dp"
            android:src="@drawable/ic_baseline_timer_blue_24" />

        <!--TextView used to display average speed heading for trip three-
->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="40dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Average Speed"
            android:textSize="18dp" />

        <!--TextView used to display average speed value for trip three --
>
        <TextView
            android:id="@+id/speedTextThree"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="45dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display speed icon for trip one -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="bottom"
            android:layout_marginStart="10dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_speed_red_24" />

        <!--TextView used to display distance heading for trip three -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
            android:layout_marginEnd="20dp"
            android:fontFamily="@font/roboto_light"
            android:text="Distance"
            android:textSize="18dp" />

        <!--TextView used to diaply  distance value for trip three -->
```

```
        <TextView
            android:id="@+id/distanceTextThree"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="80dp"
            android:layout_marginEnd="22dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />

        <!--ImageView used to display distance icon for trip three -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="right"
            android:layout_marginTop="50dp"
            android:layout_marginEnd="105dp"
            android:src="@drawable/ic_baseline_directions_car_orange_24"
/>

        <!--TextView used to display detections heading for trip three -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="230dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto_light"
            android:text="Detections"
            android:textSize="18dp" />

        <!--ImageView used to display detections icon for trip three -->
        <ImageView
            android:layout_width="24dp"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="200dp"
            android:layout_marginBottom="40dp"
            android:src="@drawable/ic_baseline_tag_faces_pink_24" />

        <!--TextView used to display detections value for trip three -->
        <TextView
            android:id="@+id/detectTextThree"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="270dp"
            android:layout_marginBottom="20dp"
            android:fontFamily="@font/roboto_light"
            android:text=""
            android:textSize="18dp" />
```

```
</androidx.cardview.widget.CardView>

<!--Guidelines used to create constraint layout-->
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizGuideline1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.075" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizGuideline2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.32" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizGuideline3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.37" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizGuideline4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.62" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizGuideline5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.675" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/horizguideline6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.925" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/verGuideline1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.1" />

<androidx.constraintlayout.widget.Guideline
```

```
        android:id="@+id/verGuideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.9" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

g. activity_eye_tracking.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/background"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true"
    android:orientation="vertical"
    tools:context=".EyeTracking">

    <!--CardView used to hold object relating to speed-->
    <androidx.cardview.widget.CardView
        android:id="@+id/speedCard"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="50dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/guideline11"
        app:layout_constraintEnd_toEndOf="@id/guideline7"
        app:layout_constraintStart_toStartOf="@id/guideline4"
        app:layout_constraintTop_toTopOf="@id/guideline10">

        <!--TextView to show Speed heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="20dp"
            android:layout_marginTop="25dp"
            android:fontFamily="@font/roboto_light"
            android:text="Speed"
            android:textSize="24dp" />

        <!--ImageView to show Speed icon-->
        <ImageView
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="20dp"
            android:padding="10dp"
            android:src="@drawable/ic_baseline_speed_24" />

        <!--FrameLayout used to hold speed fragment-->
        <FrameLayout
            android:id="@id/fragmentSpeedo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_gravity="right"
            android:layout_marginTop="25dp"
            android:layout_marginRight="30dp" />
    </androidx.cardview.widget.CardView>


    <!--Relative layout to hold CardView and Buttons at bottom of screen--
>
    <RelativeLayout
        android:id="@+id/relativeLayout"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_gravity="bottom"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/bottomGuideeye">

        <!--CardView to hold buttons for preview and Google Maps-->
        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginBottom="-8dp"
            android:elevation="100dp"
            app:cardCornerRadius="15dp">

            <!--Table used to arrange buttons-->
            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_alignParentStart="true"
                android:layout_alignParentLeft="false"
                android:layout_alignParentTop="false"
                android:layout_alignParentBottom="true"
                android:layout_weight="0"
                android:background="#B2BEC3"
                android:gravity="center|center_horizontal"
                android:weightSum="2">

                <!--ToggleButton used to turn preview on and off-->
                <ToggleButton
                    android:id="@+id/previewButton"
                    android:layout_width="200dp"
                    android:layout_height="40dp"
                    android:background="@drawable/roundedeyes"
                    android:checked="false"
                    android:text="New ToggleButton"
                    android:textColor="#ffff"
                    android:textOff="@string/turn_preview_off"
                    android:textSize="12dp" />

                <!--Button used to open Google Maps-->
                <Button
```

```
                android:id="@+id/mapButton"
                android:layout_width="150dp"
                android:layout_height="40dp"
                android:layout_marginLeft="10dp"
                android:background="@drawable/roundedeyes"
                android:checked="false"
                android:text="Open Google Maps"
                android:textColor="#ffff"
                android:textSize="12dp" />

        </TableRow>
    </androidx.cardview.widget.CardView>
</RelativeLayout>


<!--CardView used to hold object relating to journey time-->
<androidx.cardview.widget.CardView
    android:id="@id/journeyCard"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:elevation="10dp"
    app:cardCornerRadius="15dp"
    app:layout_constraintBottom_toBottomOf="@id/guideline8"
    app:layout_constraintEnd_toEndOf="@id/guideline7"
    app:layout_constraintStart_toStartOf="@id/guideline4"
    app:layout_constraintTop_toTopOf="@id/guideline6">

    <!--TextView to show Journey Timer heading-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:layout_marginStart="20dp"
        android:layout_marginTop="25dp"
        android:fontFamily="@font/roboto_light"
        android:text="Journey Timer"
        android:textSize="24dp" />

    <!--ImageView to show time icon-->
    <ImageView
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="40dp"
        android:layout_marginTop="20dp"
        android:padding="10dp"
        android:src="@drawable/ic_baseline_timer_24" />

    <!--Chronometer to show journey time-->
    <Chronometer
        android:id="@id/journeyTimer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:layout_gravity="right"
            android:layout_marginTop="25dp"
            android:layout_marginRight="30dp"
            android:fontFamily="@font/roboto_light"

            android:textColor="@android:color/secondary_text_dark"
            android:textSize="24dp" />
    </androidx.cardview.widget.CardView>

    <!--CardView used to hold object relating to distance-->
    <androidx.cardview.widget.CardView
        android:id="@id/DistanceCard"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="10dp"
        app:cardCornerRadius="15dp"
        app:layout_constraintBottom_toBottomOf="@id/guideline3"
        app:layout_constraintEnd_toEndOf="@id/guideline7"
        app:layout_constraintStart_toStartOf="@id/guideline4"
        app:layout_constraintTop_toTopOf="@id/guideline2">

        <!--TextView to show Distance heading-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginStart="20dp"
            android:layout_marginTop="25dp"
            android:fontFamily="@font/roboto_light"
            android:text="Distance"
            android:textSize="24dp" />

        <!--ImageView to show distance icon-->
        <ImageView
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="20dp"
            android:padding="10dp"
            android:src="@drawable/ic_baseline_directions_car_24" />

        <!--TextView used to display distance in km-->
        <TextView
            android:id="@id/distancemain"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_marginTop="25dp"
            android:layout_marginRight="20dp"
            android:fontFamily="@font/roboto_light"
            android:text="0 km"
            android:textColor="@android:color/secondary_text_dark"
```

```
        android:textSize="24dp" />

</androidx.cardview.widget.CardView>

<!--CardView used to hold object relating to detections-->
<androidx.cardview.widget.CardView
    android:id="@id/DetectionsCard"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:elevation="10dp"
    app:cardCornerRadius="15dp"
    app:layout_constraintBottom_toBottomOf="@id/guideline15"
    app:layout_constraintEnd_toEndOf="@id/guideline7"
    app:layout_constraintStart_toStartOf="@id/guideline4"
    app:layout_constraintTop_toTopOf="@id/guideline14">

    <!--TextView to show Detections heading-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:layout_marginStart="20dp"
        android:layout_marginTop="25dp"
        android:fontFamily="@font/roboto_light"
        android:text="Detections"
        android:textSize="24dp" />

    <!--ImageView to show detections icon-->
    <ImageView
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="40dp"
        android:layout_marginTop="20dp"
        android:padding="10dp"
        android:src="@drawable/ic_baseline_tag_faces_24" />

    <!--TextView used to show number of detections-->
    <TextView
        android:id="@id/detections_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:layout_marginTop="25dp"
        android:layout_marginRight="30dp"
        android:fontFamily="@font/roboto_light"
        android:text="0"
        android:textColor="@android:color/secondary_text_dark"
        android:textSize="24dp" />

</androidx.cardview.widget.CardView>

<!--CardView to hold DriveSafely Logo-->
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/logoCard"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:elevation="50dp"
    android:visibility="invisible"
    app:cardCornerRadius="15dp"
    app:layout_constraintBottom_toBottomOf="@id/guideline12"
    app:layout_constraintEnd_toStartOf="@id/guideline7"
    app:layout_constraintStart_toStartOf="@id/guideline4"
    app:layout_constraintTop_toTopOf="@id/guideline5">

    <!--ImageView to hold DriveSafely logo-->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/logo_dark_text" />
</androidx.cardview.widget.CardView>

<!--Framelayout for camera preview-->
<FrameLayout
    android:id="@+id/previewFrame"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:visibility="visible"
    app:layout_constraintBottom_toBottomOf="@id/guideline12"
    app:layout_constraintEnd_toStartOf="@id/guideline7"
    app:layout_constraintStart_toStartOf="@id/guideline4"
    app:layout_constraintTop_toTopOf="@id/guideline5">

    <!--CardView used to hold camera preview-->
    <androidx.cardview.widget.CardView
        android:id="@id/previewCard"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:elevation="100dp"
        app:cardCornerRadius="10dp">

        <!--Add camera source preview-->
        <com.example.drivesafely.CameraSourcePreview
            android:id="@+id/preview"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <!--Add grahpic overlay-->
            <com.example.drivesafely.GraphicOverlay
                android:id="@+id/faceOverlay"
                android:layout_width="match_parent"
                android:layout_height="match_parent" />

        </com.example.drivesafely.CameraSourcePreview>
    </androidx.cardview.widget.CardView>
</FrameLayout>
```

```
<!--Guidelines used for constraint layout-->
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.66" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.76" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline14"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.78" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline15"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.88" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.42" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.52" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.54" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline11"
```

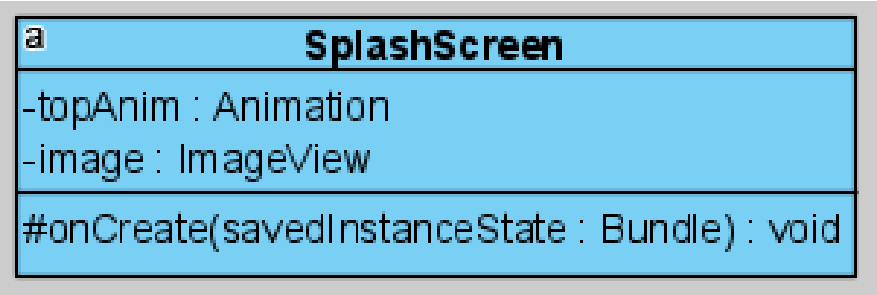
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.64" />


    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/bottomGuideeye"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.9" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.05" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.03" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.95" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.4" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

h. fragment_speedometer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashScreen">

    <!--TextView used to display the users current speed-->
    <TextView
        android:id="@+id/speedTextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/roboto_light"
        android:text="0 km/h"
        android:textColor="@android:color/secondary_text_dark"
        android:textSize="24dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Appendix D: Class Diagrams

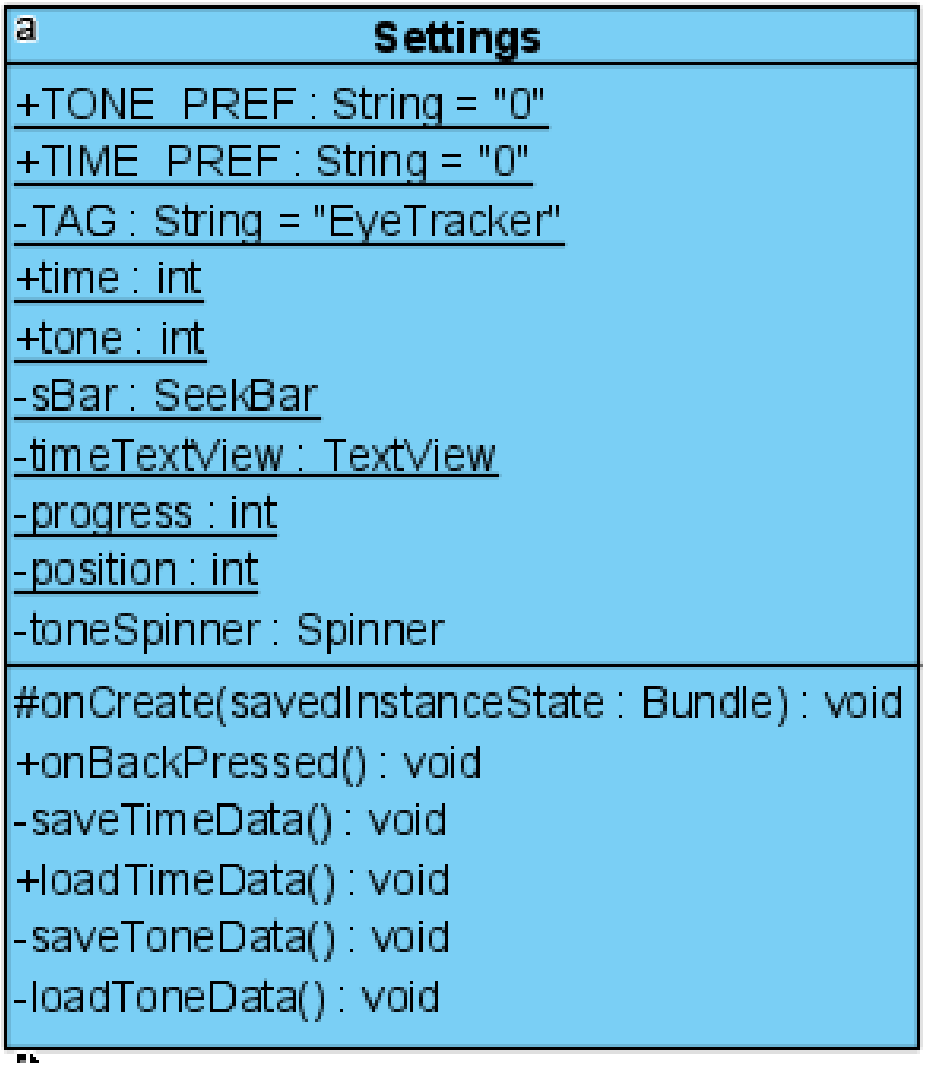
a. SplashScreen



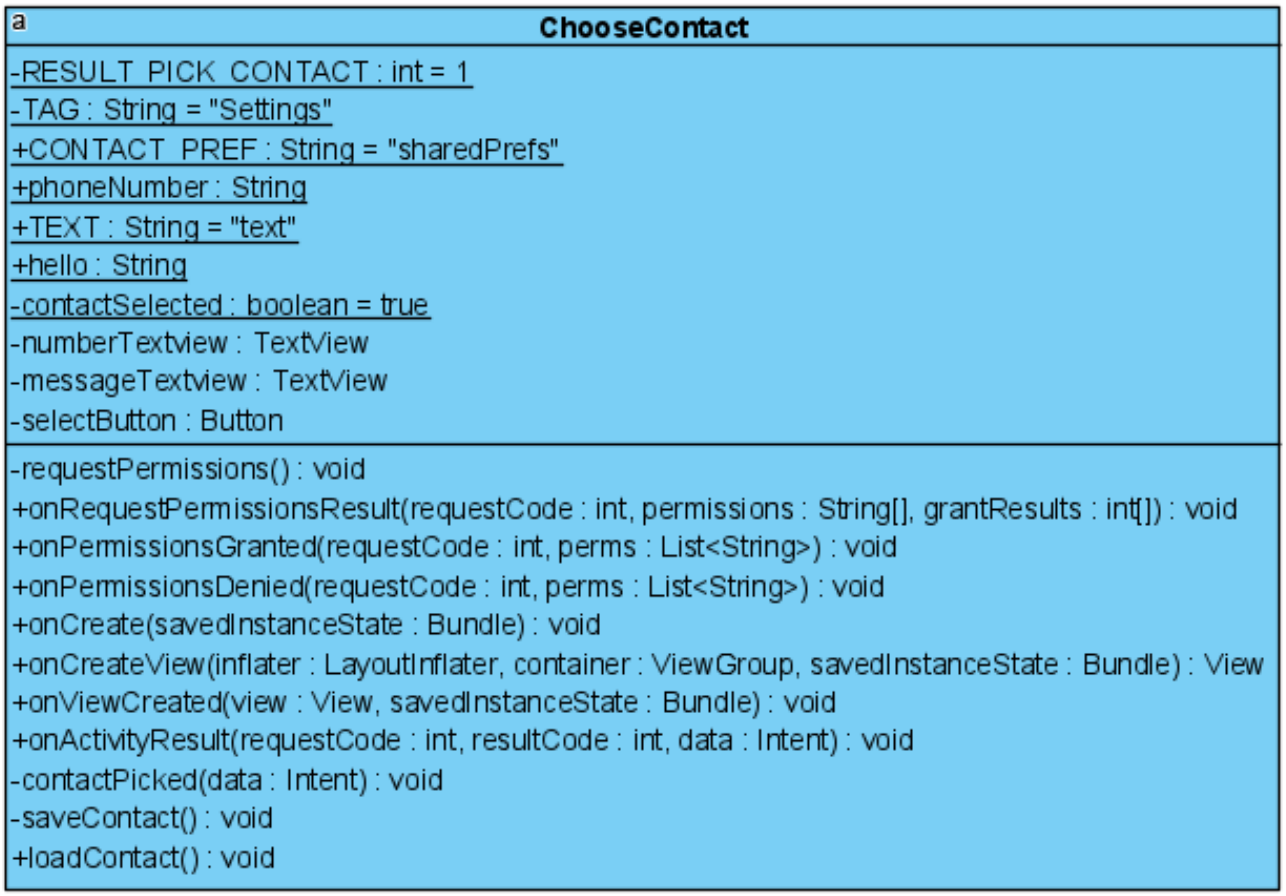
b. Dashboard

c. Help

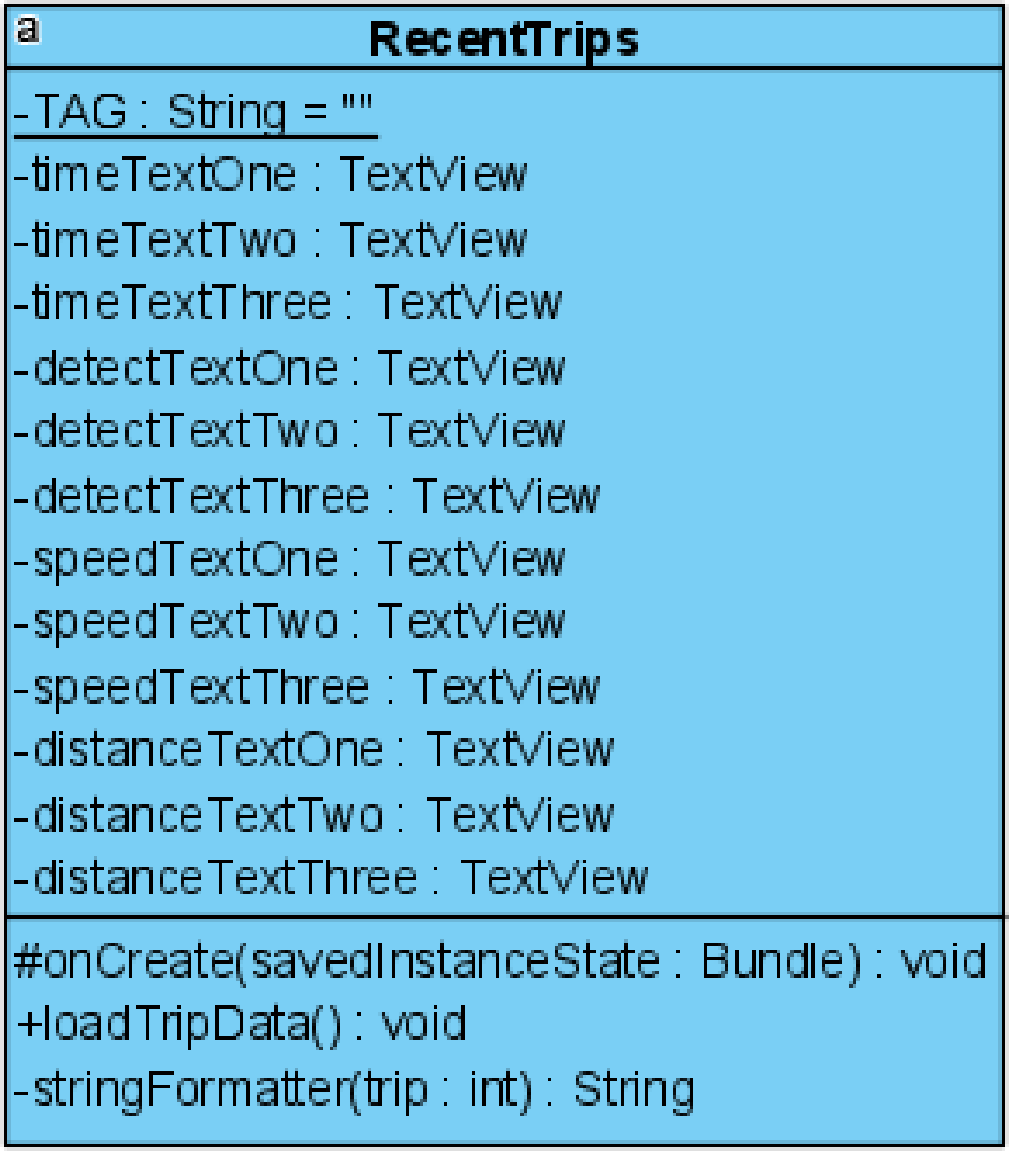
d. Settings


Settings
+TONE_PREF : String = "0"
+TIME_PREF : String = "0"
-TAG : String = "EyeTracker"
+time : int
+tone : int
-sBar : SeekBar
-timeTextView : TextView
-progress : int
-position : int
-toneSpinner : Spinner
#onCreate(savedInstanceState : Bundle) : void
+onBackPressed() : void
-saveTimeData() : void
+loadTimeData() : void
-saveToneData() : void
-loadToneData() : void

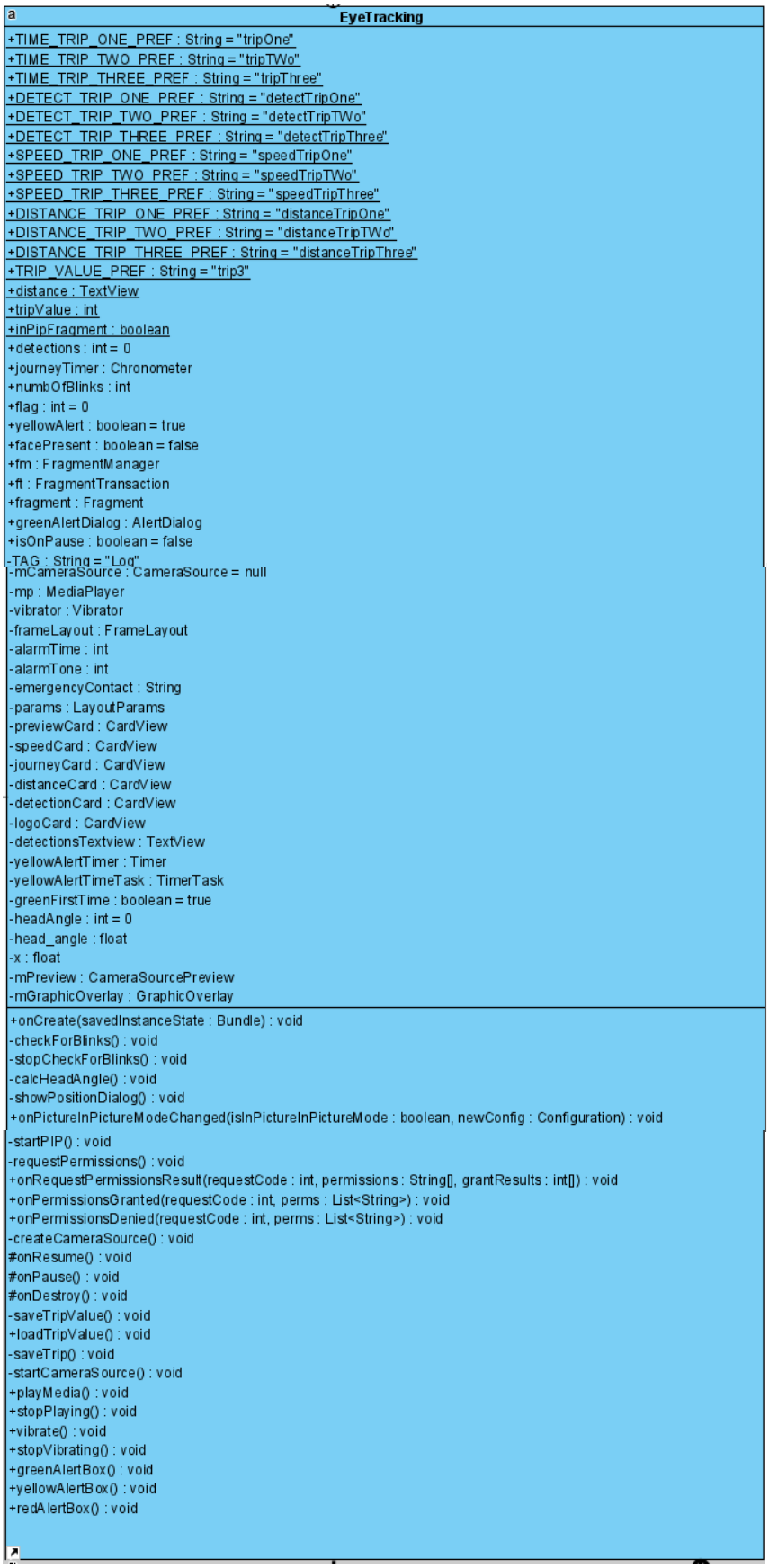
e. ChooseContact



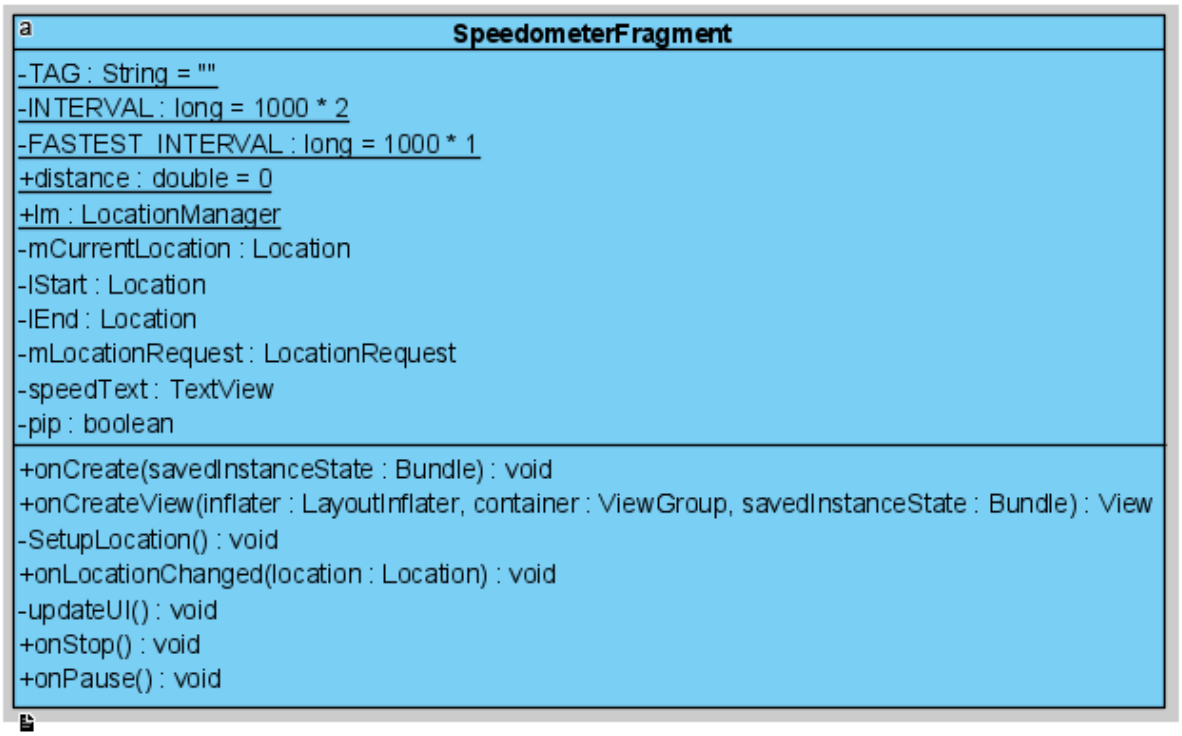
f. RecentTrips



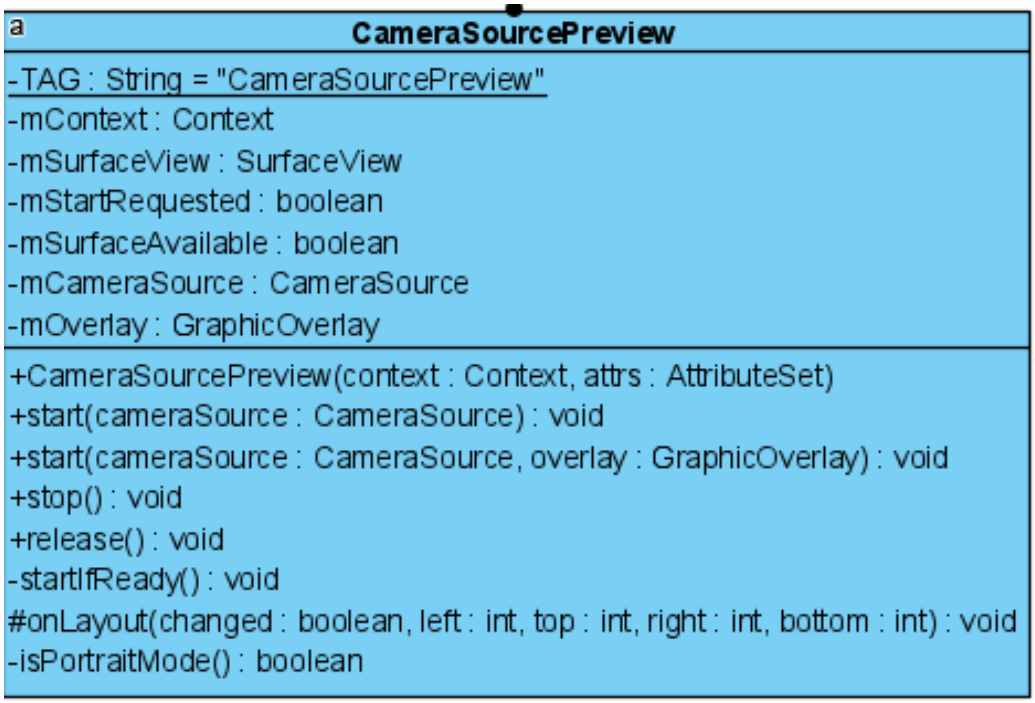
g. EyeTracking



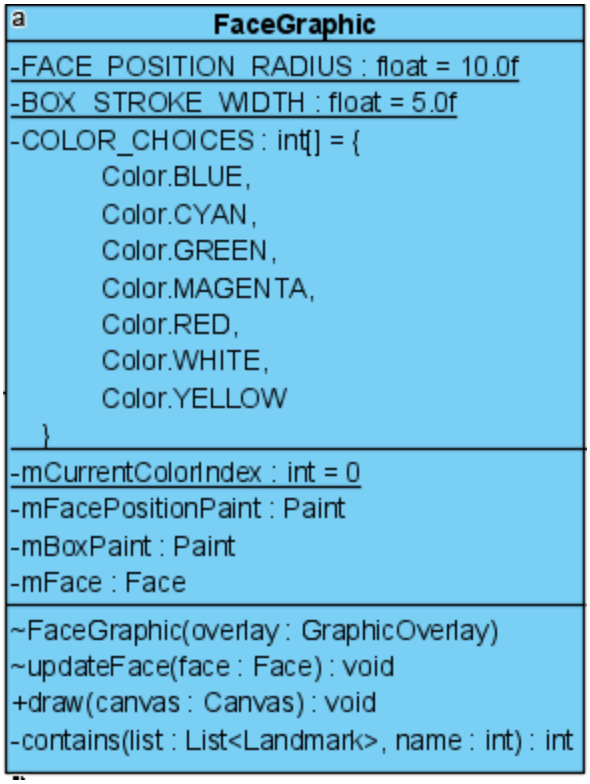
h. SpeedometerFragment



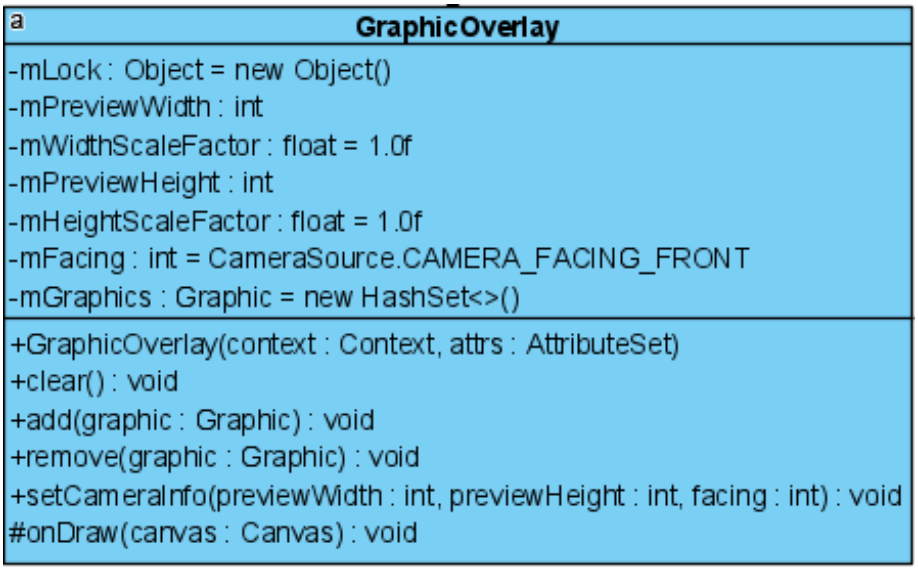
i. CameraSourcePreview



j. FaceGraphic



k. GraphicOverlay



Appendix E: Poster

DriveSafely – Android Application To Detect Drowsy Driving

E&CE | Department of Electronic and Computer Engineering

Reece Gavin

LM118 - Electronic and Computer Engineering

Introduction

This purpose of this project was to create an Android application that continuously monitors a driver's face and eyes to detect different driver states and identify clues of tiredness, fatigue and/or distraction while driving. The app operates an alarm system to wake up a detected tired driver and also has other features such as sending text messages to an emergency contact.

This project was selected for a number of reasons, including a personal interest in cars and the opportunity to gain a greater understanding of software development.

Aim

The aim of the project was to design and develop a user-friendly mobile application for Android devices, which alerts vehicle driver's when they are tired or have fallen asleep at the wheel. The overall goal of the application is to reduce the amount of road traffic accidents and deaths caused by driver fatigue.

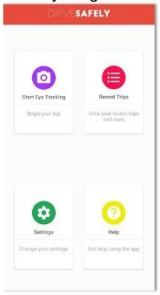
Design Tools

android, Java, Mobile Vision, Android Studio, eXtensible Markup Language

Implementation & Results

The application is designed to be simple and user friendly to cater for both younger and older users.

On opening the application, a user is brought to the dashboard screen. Here a user has four options. They can start the eye and face tracking. They can view their three most recent trips, which will show how long each journey took, what their average speed was, what the distance travelled was and the number of drowsy detections that were counted.

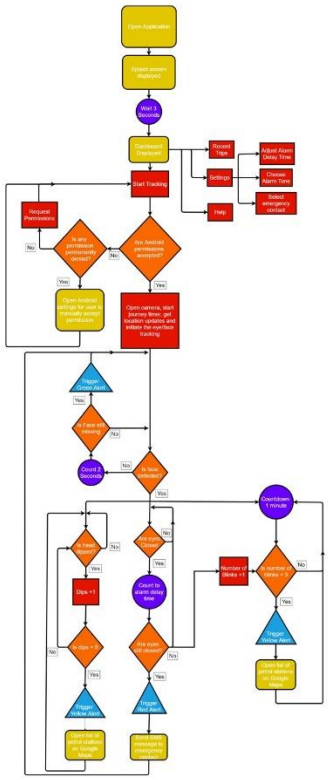


Dashboard Screen

In the Settings screen, users can change the alarm delay time, choose from 3 alarm tones and select an emergency contact.

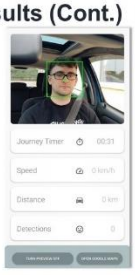
The Help screen gives the user information about the different settings that can be changed and what function they perform.

Application Flowchart



Implementation & Results (Cont.)

The application draws a bounding box around a users face. The user can turn off the camera preview at any time if it is found to be distracting. The current journey time, speed, distance travelled and detections are displayed on CardViews.



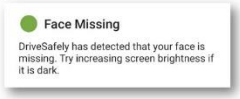
Application running

Picture-in-Picture Mode

The application has the functionality to use a feature of Android 8.0+ called Picture-in-picture. This allows the camera preview to be displayed over other apps. Google Maps can be opened from within the app so that turn-by-turn navigation can be used while tracking the face.

Implementation & Results (Cont.)

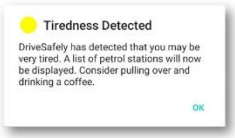
The application operates a three tier alarm system. The first tier is a Green Alert. This is triggered when a face has initially been detected by the application, but then goes missing for 2 seconds or more. A dialog box is shown to the user. When a face re-enters the screen, the dialog box is removed.



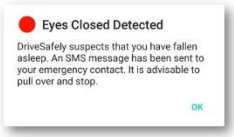
Green Alert

A Yellow alert is triggered when a user does not blink more than 5 times in a given minute or their head dips/nods 5 times. An audio alert is sounded as well as a visual dialog box. When the user dismisses the dialog, a list of nearby petrol stations are displayed on Google Maps.

Yellow Alert



A Red alert is triggered when a user has closed their eyes for an amount of user selected time (2, 3 or 4 seconds). The user's chosen audio is sounded and a dialog box is shown. An SMS message is sent to the user's emergency contact if they chose one in settings.



Red Alert

Conclusion and reflection

In conclusion, a user-friendly Android application has been created to combat drowsy driving using tools such as Android Studio and Google Vision. The creation of this system gave me the opportunity to improve my software skills and use my creativity in an area that I am interested in.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hussain Mahdi, for his advice and support throughout the entire project.

UNIVERSITY OF LIMERICK OLLSCOIL LUIMNIGH