SafeDrive AI

Project Report

Department of Computing
Bachelor of Science(Hons) in Software Development

Supervisor: Chris Meudec
Student Name: Shane Kennedy
Student Number: C00263504
Date: 2023/2024

Tables of Contents

Introduction

The goal of this report is to offer an overview of the development process used in the creation of the SafeDrive mobile application. The document will illustrate the application developed by demonstrating some of the functionalities with screenshots and give a synopsis of the technologies used to create those functionalities. The document will then review some of the challenges encountered and how they were solved as the application evolved to meet its requirements. In the next section, the report will reflect on the learning outcomes gained and review which of the original requirements were fulfilled and which were not. The final section will examine the hypothetical question of what changes would be made in the development process if starting the project from the start.

Project Description

The aim of this project was to develop a cross platform mobile application to increase the safety of users while they are driving on the road. To this end, much of the start of the development process was spent conducting research to pinpoint what contributing factors to road accidents are and subsequently to consider which of these factors could be targeted with a technological solution that would reduce the risk of that factor contributing to a road traffic accident [1,2]. This question was extremely important as it was this research that would define the key functionalities that the application must contain. Once these factors were defined the application could work to implement solutions to reduce the risks presented by these factors and thereby increase driver safety and package that functionality in a user-friendly mobile application. In the end, the three contributory factors that were targeted were driver fatigue, driver distraction and excessive speed.

Screenshots

Home Screen

Main screen

HomeScreen.js defines the main menu for the SafeDrive application, serving as the home screen. This component also serves as the main navigation hub for the application and is a component of Stacker.js which is the container which handles all of the navigation routes within the application. A stack navigation scheme is used within the application to make navigation simple to understand and utilise by drivers.

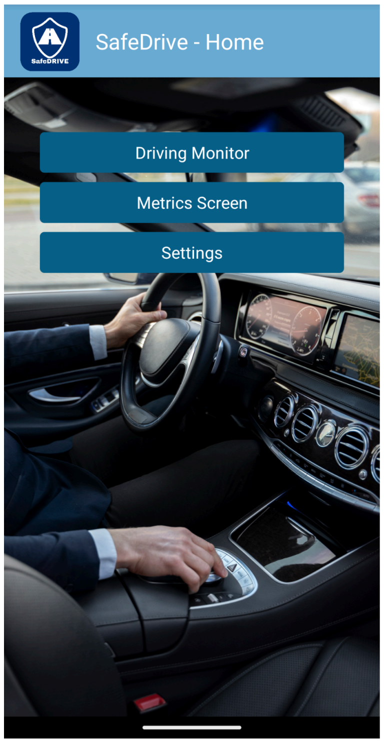


Figure 1. Application Main Menu

Permissions functionality

This screen also includes functionality to verify any necessary permissions needed. When a user attempts to navigate to the Driver Monitoring screen, the component checks which monitoring options are enabled in the application settings and ensures the correct permissions have been granted before allowing access to the screen. This is handled by the handleCameraPress function when the "Driving Monitor" button is pressed.

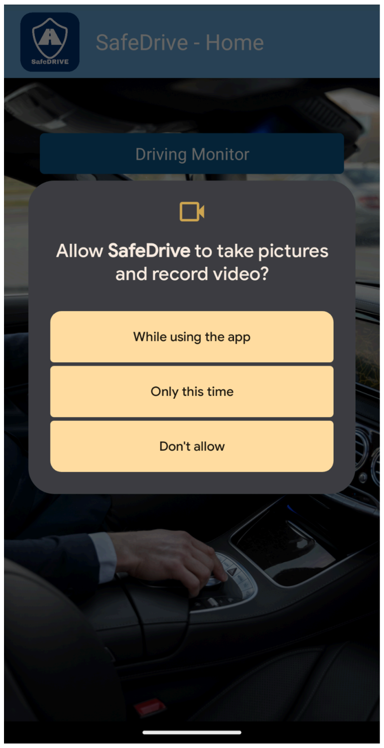


Figure 2. Permissions handling when opening screens.

Stacker.js also initialises the connection to the firebase database for anonymous error logging if the driver has that option enabled in settings and passes it to the various screens for logging any errors to the database. This functionality was included to streamline debugging of errors to increase application maintainability.

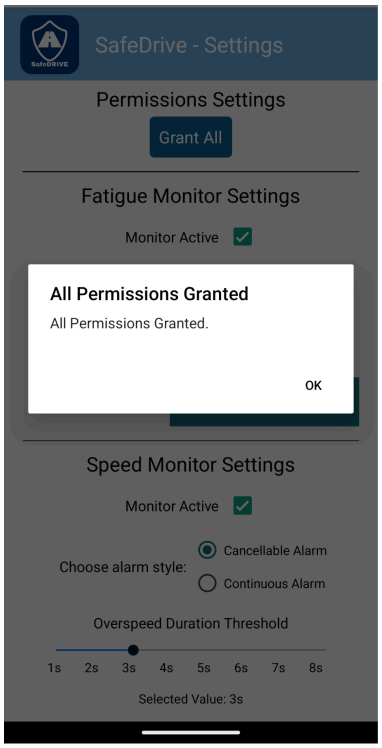
The return statement renders the UI components for the home screen, including a header with the SafeDrive logo and text, and buttons for navigating to the "Driving Monitor" and "Settings" screens.

Overall, HomeScreen.js serves as the entry point for the SafeDrive application, providing users with access to the main features and functionalities.

Setting Screen

Settings.js serves as the configuration screen for the SafeDrive application, allowing users to control permissions and customise various application settings. This component functionality was designed to allow the driver to tailor the application's function to their individual preferences and give drivers complete control over which monitoring systems they wish to utilise.

The screen is divided into sections for different settings categories, such as fatigue monitor settings, speed monitor settings, distraction monitor settings, permissions settings, and logging preferences. Each section contains relevant UI elements like checkboxes, sliders, radio buttons, and pickers for user interaction.


SafeDrive - Settings

Permissions Settings

Grant All

Fatigue Monitor Settings

Monitor Active ✓

All Permissions Granted

All Permissions Granted.

OK

Speed Monitor Settings

Monitor Active ✓

Choose alarm style: [✓ Cancellable Alarm] Continuous Alarm

Overspeed Duration Threshold

1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 3s

Figure 3. Permissions granting functionality in Settings.

Grant Permissions

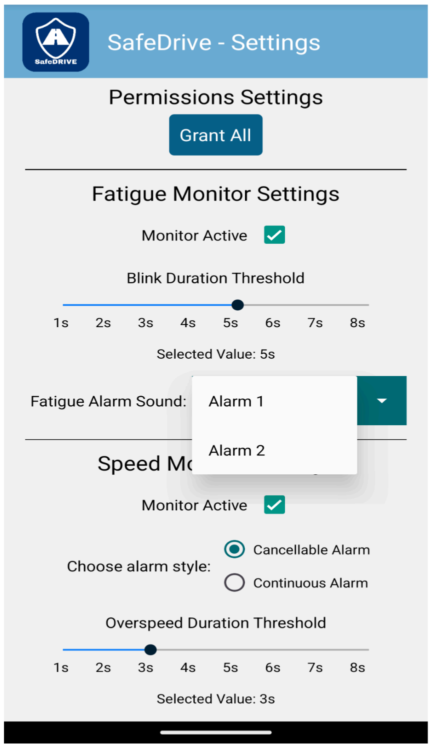
Permission handling is a crucial aspect of the application's functionality with access to the device's camera and/or location needed depending on the monitors being used. Settings allows users to grant all permissions needed with a single button press to enhance usability. Alerts are displayed to inform users about the status of permission grants.

Fatigue Monitoring Settings

The next section is the Fatigue Monitoring settings. Within this section users can control whether the driver fatigue monitor is active, the duration of driver fatigue detection before an alarm is triggered and the alarm sound that should be played during a fatigue alert.

Speed Monitoring Settings

The Speed Monitoring setting follows with all the configuration options for the application's speed monitoring functionality. The first setting allows the drivers to


SafeDrive - Settings

Permissions Settings

Grant All

Fatigue Monitor Settings

Monitor Active ✓

Blink Duration Threshold

1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 5s

Fatigue Alarm Sound: Alarm 1 / Alarm 2

Speed Mc

Monitor Active ✓

Choose alarm style: [✓ Cancellable Alarm] Continuous Alarm

Overspeed Duration Threshold

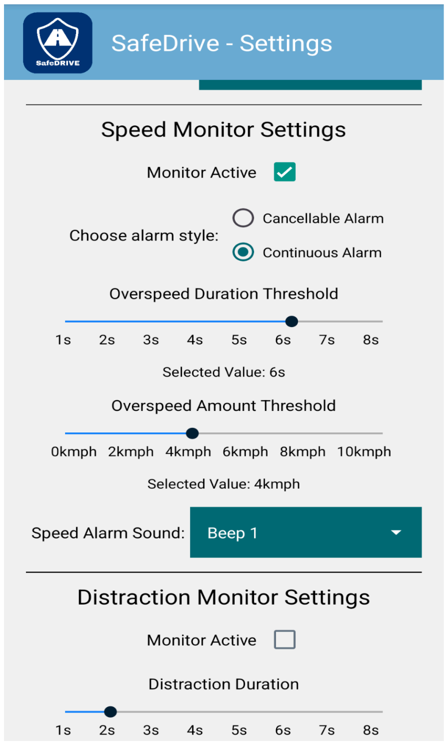
1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 3s

Figure 4. Fatigue Monitor Settings.

enable or disable speed monitoring and the next being the choice of alarm style. For excessive speed alerts there are two types of alarm each with different types of alarm sounds:

1. **Cancellable alarm:** This alarm is simply a popup alarm that displays if the driver exceeds the speed limit past the threshold values. This alarm stops sounding only when the driver acknowledges the alarm by pressing the cancel button.
2. **Continuous alarm:** This alarm is a continuous alarm which will continue sounding until the driver drops back under the speed limit thresholds whereupon it cancels itself automatically.


SafeDrive - Settings

Speed Monitor Settings

Monitor Active ✓

Choose alarm style: Cancellable Alarm [✓ Continuous Alarm]

Overspeed Duration Threshold

1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 6s

Overspeed Amount Threshold

0kmph 2kmph 4kmph 6kmph 8kmph 10kmph

Selected Value: 4kmph

Speed Alarm Sound: Beep 1

Distraction Monitor Settings

Monitor Active ☐

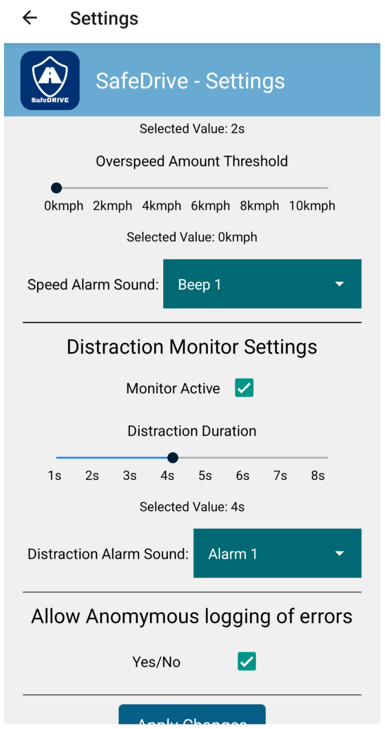
Distraction Duration

1s 2s 3s 4s 5s 6s 7s 8s

Figure 5 Speed Monitor Settings.

The next setting is Overspeed duration which controls how long a driver must be over the speed limit threshold before an alarm is activated. The driver can also set the Overspeed amount threshold which controls how far over the speed limit they must be before a speed alert is triggered. These two settings were designed to prevent alarms from being activated if a driver is briefly overtaking another vehicle for example. This design decision was made to prevent annoying drivers with unnecessary alerts which would prompt them to stop using the functionality.

The final setting is the alarm tone that is activated during speed alerts, the choice of alarms available updates automatically when a new Alarm style is selected.


Settings

SafeDrive - Settings

Selected Value: 2s

Overspeed Amount Threshold

0kmph 2kmph 4kmph 6kmph 8kmph 10kmph

Selected Value: 0kmph

Speed Alarm Sound: Beep 1

Distraction Monitor Settings

Monitor Active ✓

Distraction Duration

1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 4s

Distraction Alarm Sound: Alarm 1

Allow Anomymous logging of errors

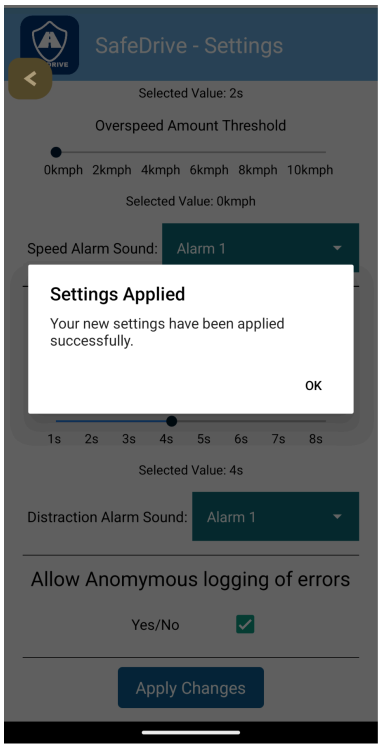
Yes/No ✓

Apply Changes

Figure 6. Distraction Monitor Settings.

Distraction Monitor Settings

The next monitoring control section is for the Distraction Monitoring functionality. Within this section users can control whether the driver distraction monitor is active, the duration of driver distraction detection before an alarm is triggered and the alarm sound that should be played during a distraction alert.


SafeDrive - Settings

Selected Value: 2s

Overspeed Amount Threshold

0kmph 2kmph 4kmph 6kmph 8kmph 10kmph

Selected Value: 0kmph

Speed Alarm Sound: Alarm 1

Settings Applied

Your new settings have been applied successfully.

OK

1s 2s 3s 4s 5s 6s 7s 8s

Selected Value: 4s

Distraction Alarm Sound: Alarm 1

Allow Anomymous logging of errors

Yes/No ✓

Apply Changes

Figure 7. Example of Settings being applied.

Error Logging Settings

The final section is a checkbox which controls whether anonymous error logging is active. When this control is set to true, if an exception is raised within the application, an error log will be automatically uploaded to firebase. This error log contains no personal data, just hardware device information and the stack trace of the exception raised.

In brief, the Settings.js screen allows users to customise monitoring preferences to suit their needs allowing them to customise and select only the monitoring functions that they wish to utilise. It contributes to the application's functionality and user experience by providing a comprehensive configuration interface.

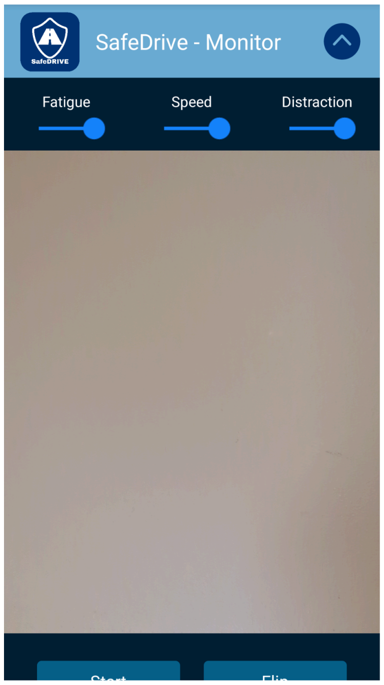


Figure 8. Dropdown monitor controls.

Monitoring Screen

Monitoring Screen

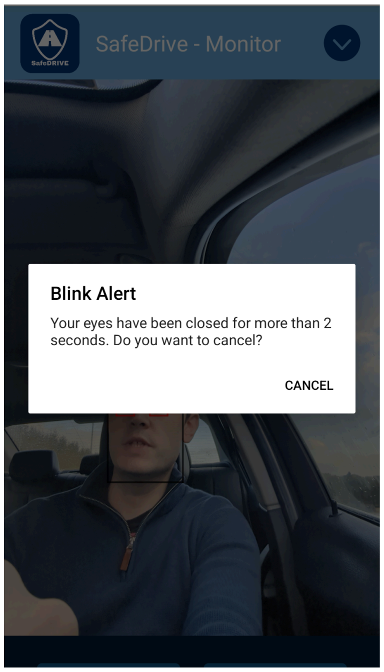
The MonitorScreen.js component is the screen which delivers the main functionality of the mobile application. Through this screen the application allows drivers access to three different monitors, each monitor targeting a contributory factor in road traffic accidents; Driver fatigue, Excessive speed and Driver distraction.

Dropdown Monitor Controls

The header contains a dropdown menu which allows drivers to activate and deactivate the monitors dynamically during use without having to navigate back to the Settings screen.
The Monitoring Screen is designed to be simple to use with only two further buttons beyond the dropdown controls. The first button is the Start button which allows drivers to activate and deactivate the monitoring functions. The second button is the Flip button which enables drivers to choose between the front and back camera for the video input.

Fatigue Monitor

The illustration in Figure 9 shows an example where the driver has started the fatigue monitoring system. Once active, the system will detect any faces in the image and overlay a box around the face to indicate to the driver that the system is working. The monitor will also overlay boxes around the driver's eyes that will be green if



Blink Alert

Your eyes have been closed for more than 2 seconds. Do you want to cancel?

Figure 10. Driver Fatigue Alarm activated.

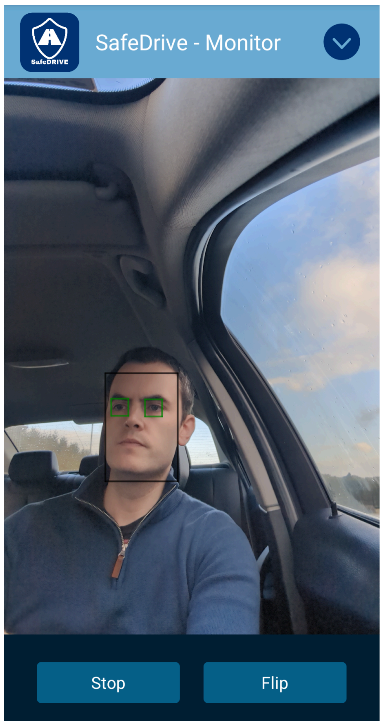


Figure 9.Fatigue Monitor Active.

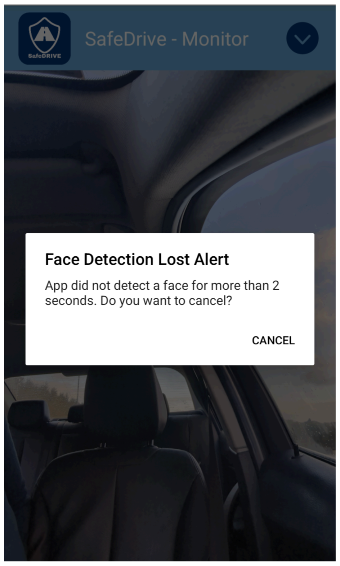
the eyes are detected as open and red if they are detected as closed.

Driver Fatigue Detection

Figure 10 is a sample screenshot of a driver fatigue event detection. The application has detected that the driver's eyes have been closed for more than the specified threshold and activated an alarm.

Face Detection lost

Figure 13 is illustrating another function of the driver fatigue system. If the application cannot detect a face within the video frame, it will activate a face detection loss alarm. This was implemented as a safety function to alert drivers that the monitoring system is not functional due to loss of face detection.



Face Detection Lost Alert

App did not detect a face for more than 2 seconds. Do you want to cancel?

Figure 11. Face detection lost alarm.

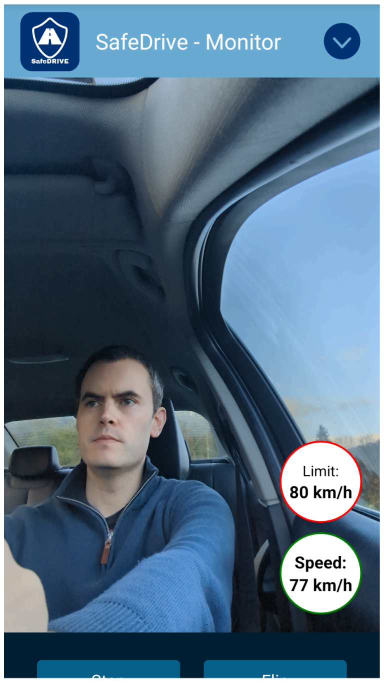


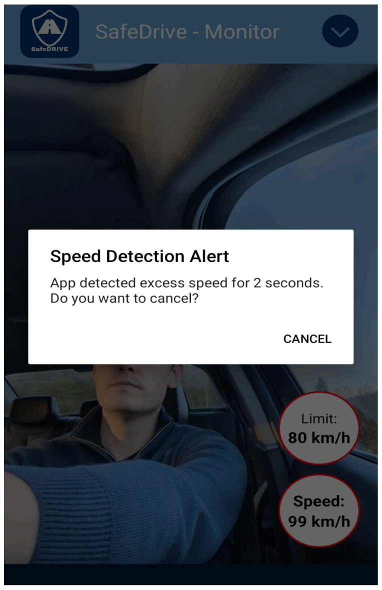
Figure 12. Speed Monitor activated.

Speed Monitor Active

The image in Figure 12 illustrates the functionality of the Speed monitoring system Once activated, the application will generate two speed indicators, one for the current speed of the driver and another showing the speed limit detected for the current road section that the driver is traversing. These indicators will be green when below the speed limit and turn red if the driver exceeds the speed limit to give a visual warning that the speed limit has been exceeded.

Cancellable Alarm activation

If the driver has selected the cancellable alarm option and the speed limit is exceeded past the threshold values in settings, the application will activate an alarm that will sound until the driver acknowledges the alarm by pressing cancel on the alert. The driver can set the threshold value which controls how many seconds they may be over the speed limit before the alarm activates. The driver may also choose the sound that will play when the alarm is activated. This functionality was included to enable the application to have different alarm sounds depending on the type of alarm activated.



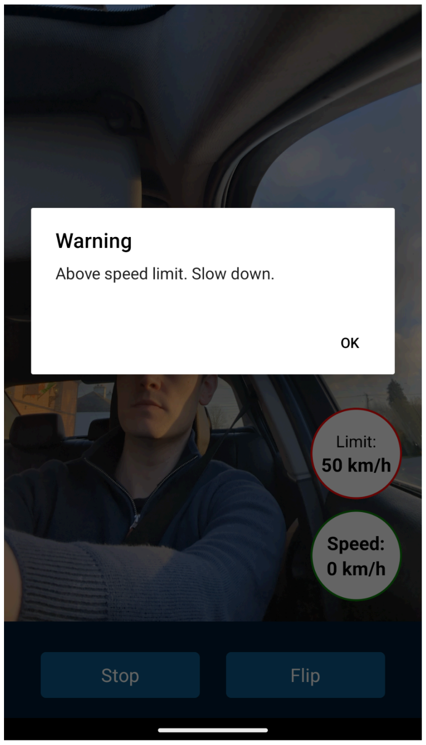
Speed Detection Alert

App detected excess speed for 2 seconds. Do you want to cancel?

Figure 13. Cancellable speed alarm activation.

Continuous Alarm activation

If the driver has selected the continuous alarm option and the speed limit is exceeded past the threshold values in settings, the application will activate an alarm that will sound until the driver drops below the speed limit, once which it will cancel itself automatically.

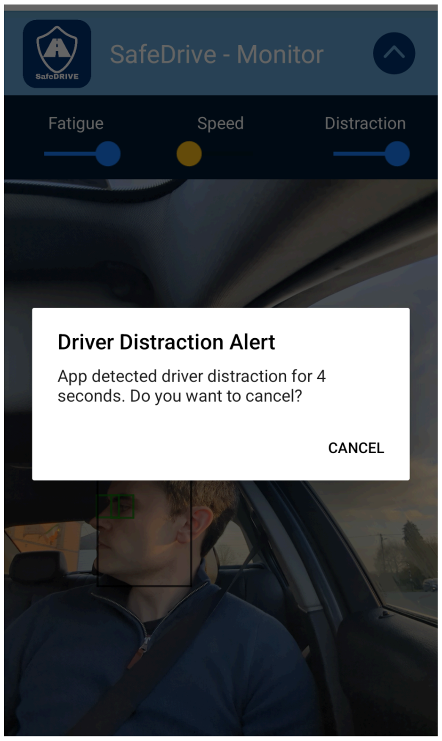


Warning

Above speed limit. Slow down.

Figure 14. Continuous alarm that sounds until driver is back below speed limit..

Driver Distraction Alarm activation



Driver Distraction Alert

App detected driver distraction for 4 seconds. Do you want to cancel?

Figure 15. Driver Distraction Alarm.

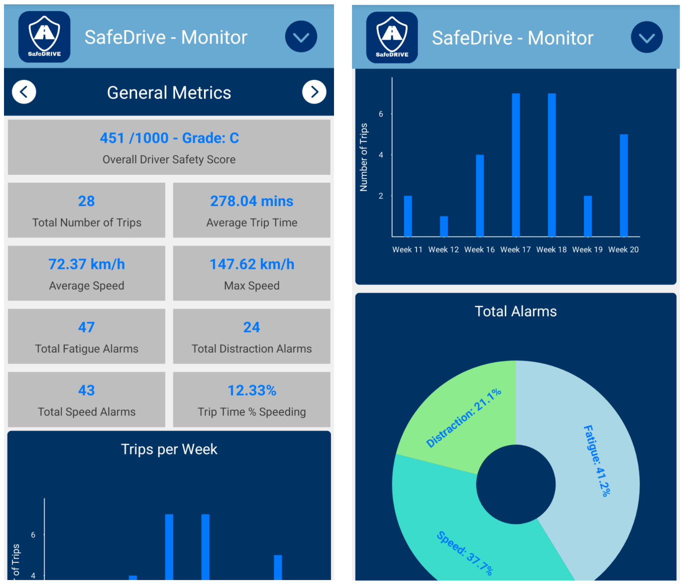
Figure 15 gives an illustration of the driver distraction alarm activation. This function operates by taking the average head position in both roll (up and down movements) and yaw (side to side movements) over the first thirty seconds of monitor activation. These averages are then taken as the probable correct head driving position that the driver utilises. Once calibrated, the monitor will then sound an alarm if the driver's head moves in either roll or yaw beyond the set threshold. This monitor is designed to alert the driver if they look away from the road, for example, looking down at their lap or to either side.

Metrics Screen

The metrics screen is designed to allow the driver to visualise and reflect on their overall driving performance and safety. The application collects the data for these calculations and visualisations any time the application monitor is activated and then collates and displays these key performance indicators when the metrics screen is opened . The metrics screen is divided into four sub-screens that the driver can switch between each focusing on a separate set of metrics, General Metrics, Fatigue Metrics, Speed Metrics and Distraction Metrics. The application also calculates a driver score between 0 and 1000 for Fatigue performance, Speed Performance and Distraction performance to give the driver an idea of their driving safety related to these factors. These three scores are combined on the General Metrics screen to give the driver an overall safety score and a corresponded grade between A and F.

General Metrics

The General Metrics screen displays the overall driver score and a number of other calculated values relating to overall journey statistics such as number of trips, average trips time, average speed etc. The screen also contains visualisations relating to trip and alarm history.



SafeDrive - Monitor

General Metrics

451 /1000 - Grade: C
Overall Driver Safety Score

28
Total Number of Trips

278.04 mins
Average Trip Time

72.37 km/h
Average Speed

147.62 km/h
Max Speed

47
Total Fatigue Alarms

24
Total Distraction Alarms

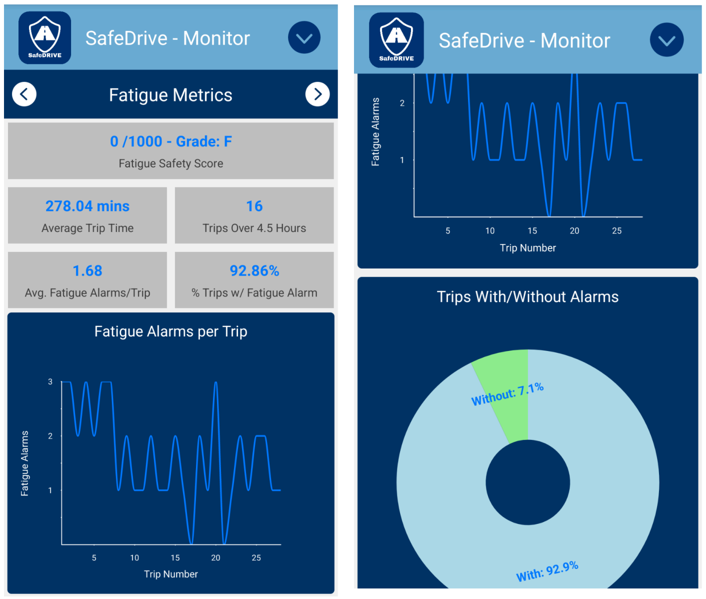
43
Total Speed Alarms

12.33%
Trip Time % Speeding

Figure 16. General Metrics Screen.

Fatigue Metrics

Fatigue Metrics is the screen that gives the driver an overview of all trips where the Driver fatigue monitor was active . The screen displays the driver score for fatigue safety which is a value between 0 and 1000. Points are deducted from the score based on the average number of fatigue alarms per trip and the percentage of trips where a fatigue alarm occurred. The screen displays a number of calculated metrics on fatigue performance and also contains visualisations to illustrate some of the key performance indicators.



SafeDrive - Monitor

Fatigue Metrics

0 /1000 - Grade: F
Fatigue Safety Score

278.04 mins
Average Trip Time

16
Trips Over 4.5 Hours

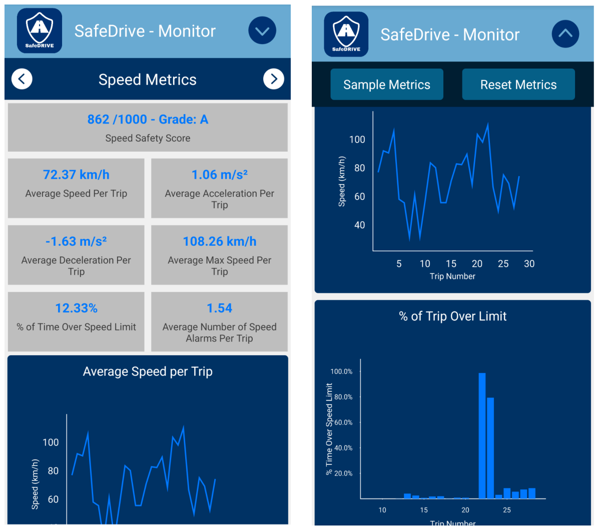
1.68
Avg. Fatigue Alarms/Trip

92.86%
% Trips w/ Fatigue Alarm

Figure 17. Fatigue Metrics Screen.

Speed Metrics

The Speed Metrics screen presents the driver with an overview of all trips where the Speed monitor was active. The screen displays the driver score for speed safety which is a value between 0 and 1000. Points are deducted from the score based on the average number of speed alarms per trip and the percentage of time of all trips where the driver was exceeding the speed limit. The screen displays a number of calculated metrics on speed performance and also contains visualisations to illustrate some of the key performance indicators.



862 /1000 - Grade: A
Speed Safety Score

72.37 km/h
Average Speed Per Trip

1.06 m/s²
Average Acceleration Per Trip

-1.63 m/s²
Average Deceleration Per Trip

108.26 km/h
Average Max Speed Per Trip

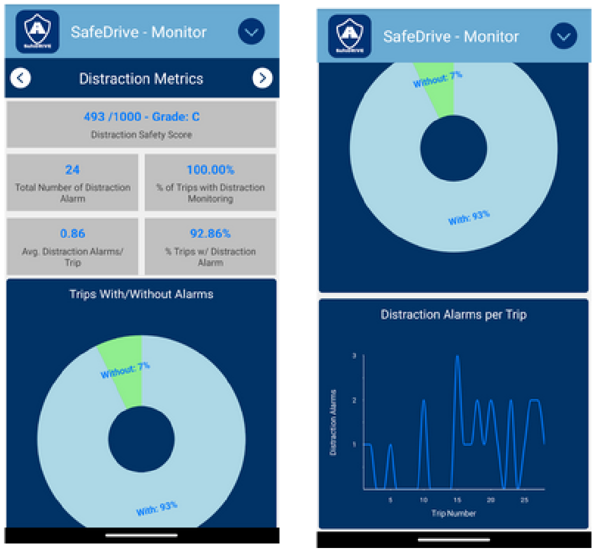
12.33%
% of Time Over Speed Limit

1.54
Average Number of Speed Alarms Per Trip

Figure 18. Speed Metrics Screen.

Distraction Metrics

The Distraction Metrics screen follows the format of the previous metrics screen but displays the metrics for trips where the Distraction monitor was active . The screen displays the driver score for distraction safety which is a value between 0 and 1000. Points are deducted from the score based on the average number of distraction alarms per trip and the percentage of time of all trips where a distraction alarm occurred. The screen displays the calculated metrics on



493 /1000 - Grade: C
Distraction Safety Score

24
Total Number of Distraction Alarm

100.00%
% of Trips with Distraction Monitoring

0.86
Avg. Distraction Alarms/ Trip

92.86%
% Trips w/ Distraction Alarm

Figure 19. Distraction Metrics Screen.

distraction performance and also visualisations to illustrate some of the key performance indicators.

Technologies Used

The front end of the application was developed using React Native, a popular framework for building mobile applications using JavaScript and React. React Native was chosen based on its suitability for fulfilling this project's particular requirements. React Native allows developers to build cross-platform mobile applications that can run on both iOS and Android using a single codebase [3]. For this project, this capability significantly reduced the development time and effort of creating a cross-platform application. React Native also provides a rich eco-system of third party libraries which proved essential in implementing some of the more advanced functionality such as the real-time processing of imagery for driver fatigue and distraction detection on a mobile device.

React Native follows a component-based architecture where UI and functional elements are built using reusable components. These components can accept parameters (props) which can be used to customise their behaviour and appearance. This approach again reduced development time by allowing for code reuse across the application while also offering simplified maintenance and enhancing scalability as these modular and self-contained components can be created and then reused in any future expansions of the application. Furthermore, React Native renders these components dynamically based on the props passed and application state which enables the creation of adaptable UIs that can change during runtime as the application state evolves.

The Expo framework was used for the building of the React Native application. This framework is a library and platform designed to abstract some of the complexities of building React Native applications [4]. The Expo CLI offers a number of tools and services that simplify the development process. This framework enables a managed workflow which can handle the configuration of some dependencies automatically and its development servers allow hot refreshes of code during development without having to stop and restart the application when changes are made. This technology brought significant benefits to the project including streamlined building and deploying of the application.

Firebase was utilised as an Infrastructure-as-a-Service (IaaS) database for the anonymous logging of errors within the application. By integrating Firebase as a backend database, the project was able to leverage its real-time database capabilities to efficiently store and manage error logs for any application errors generated by users [5]. This implementation was developed to enhance future maintainability through the creation of a monitoring and debugging process enabling developers to quickly identify and address any issues or bugs encountered by the application's users.

The machine learning technology employed for the detection of driver fatigue and distraction was the Google Machine Learning(ML) kit integrated into the Expo Face-Detector library [6]. This library was chosen due to its capability of providing fast real-time performance on mobile devices, ensuring efficient monitoring of drivers for signs of fatigue and distraction. Python and TensorFlow were utilised for developing a custom driver fatigue detection model which was used to compare and contrast its performance versus the Google ML Kits performance. Additionally, MediaPipe's Facial Landmark detection model was used to experiment with an alternate approach to driver fatigue detection for a comparison of detection methodologies [7]. These technologies allowed experimentation with various machine learning techniques to refine the methodology used to achieve optimal performance in the detection of fatigue and distraction given the limited resources available on mobile devices.

Other Technologies

- ***GitHub -*** The Git version control system was used for tracking and merging project changes. GitHub serves as the online datastore for backup of project files.
- ***VSCode -*** VSCode was utilised as the primary IDE for the development of the source code of the main application.
- ***Anaconda -*** Anaconda and Jupyter notebooks were used for creation of the machine learning model which were created as part of the development process.

Challenges

There were many challenges encountered over the development process of this application which necessitated continuous learning and persistence to overcome. The task was made more difficult due to my personal lack of experience with the majority of the technologies needed to complete the application's functionalities. The complexities of the project combined with this lack of experience meant many deadends were encountered where work completed could not be integrated into the project and had to be discarded necessitating new pathways and solutions to be found to overcome these roadblocks.

React Native

As the frontend architecture of the project React Native was one of the key technologies of the project. While it is a powerful framework for developing dynamic mobile applications, many challenges and issues arose due to the decision to use this framework, especially given my lack of experience with the framework and its architecture.

State Management

Transitioning to using React Native took several months in learning the basics especially around the state management system the framework uses. The lack of the ability to simply use global variables for dynamic components means that state must be utilised for the passing of any shared variable and while the use of state allows components to be modular and dynamically rendered, initially I found that many errors were caused simply by misunderstanding how the application's state was being tracked. Additionally, these errors were often hard to debug given the lack of simple debugging tools that allow visibility over the evolution of state during runtime.

This challenge was overcome by slowly learning how state management works and finding better methods of debugging problems such as using development builds and integrating react devtools to better track the evolution of states [8].

Package Selection

Navigating the vast array of third-party packages and libraries available for React Native proved to be another challenge for this project. Many packages are poorly maintained with little documentation and are very prone to causing integration issues with other packages within the application. Many deadends were caused by attempting to integrate packages which would at first seem to work in one context such as on emulator or on web but would then cause build errors when migrated to Android. This challenge was overcome by experimentation with different combinations of packages until a variation of packages was found which produced the desired functionality while maintaining application stability.

Build Errors

Understanding how the build process worked when deploying a React Native codebase to a mobile build was another roadblock in the development process. Many packages have special requirements and configuration of gradle build settings before building which will cause build errors if not correctly completed. These build errors proved difficult to debug due to often confusing error messages and little online documentation as errors are often specific to the project's unique configuration. This challenge was overcome through better understanding of how building configuration is handled within the React Native Expo framework, in particular the use of plugins to customise the build configuration and the different approaches needed for managed workflow deployment when utilising the Expo CLI.

On-device Machine Learning

This project's requirements for real-time on-device machine learning to process and monitor for driver fatigue and distraction presented a significant challenge. Initially, the plan was to develop a custom machine learning model capable of monitoring camera input to detect these events. However, a major roadblock was encountered due to the performance limitations inherent in the way React Native operates. React Native, while offering advantages in cross-platform development, operates by translating JavaScript code into native code. While this approach facilitates rapid development, it can lead to performance issues, especially when dealing with computationally intensive tasks such as image processing and machine learning [9].

In the first development attempts, the process of translating camera inputs into images suitable to be processed by a machine learning model introduced significant performance overhead meaning real-time processing of multiple frames per second was not possible. Many solutions to this problem were attempted such as writing a native code plugin that would perform the processing in native code and send the results across the React Native bridge, but these efforts proved to be a failure causing build errors which could not be resolved due to developer inexperience with the framework. Ultimately, this problem was solved by utilising a pre-trained machine learning model from the Google ML kit which performs its processing in native-code and thereby can achieve the required functionality without compromising performance or responsiveness.

Speed Limit Detection

The real-time detection of speed limits on the section of road that a driver is currently travelling offered another challenge to the project. The initial approach relied on the Google Speed Limits API, however, after some development work in this direction, it was discovered that access to this functionality is restricted to Enterprise customers. To overcome this limitation, an alternate solution was found by leveraging the OpenMaps API [10].

This change in direction introduced an alternate challenge to overcome as, unlike Google Speed Limits, the OpenMaps API lacks inbuilt functionality for snapping to the closest road section. Instead, its functionality is limited to only providing a number of road nodes in a certain radius of the input geographical position. This limitation was subsequently overcome through the creation of a function that used the Haversine formula to calculate the distances from the driver to every node returned and ultimately return the node which is closest to the driver's current location [11]. Despite this limitation, this meant that the speed limit at that node location could then be found. This solution overcame these issues and meant that in the end, the application could effectively determine the current speed limit and handle road section detection.

Learning Outcomes

Findings

The detection of driver drowsiness or fatigue was one of the keystone requirements in the development of this mobile application from the inception of the project. This was due to the fact that driver fatigue is one of the leading contributing factors in road traffic accidents around the world according to the Road Safety Authority of Ireland (RSA) [12]. As the project's aim is to increase driver safety, this ability of this application to detect and prevent such fatigue driven accidents is one of its primary value propositions for the application's potential user base.

Due to the importance of this functionality, much time and effort was spent during this project in experimentation of different methodologies for the detection of driver fatigue. Drawing on previous research on the use of machine learning in recent years in the design of the drowsiness detection system, the primary metric to determine if a driver is fatigued chosen for this project is through blink detection [13]. The other possible metric of yawning that previous research papers on this topic considered was discounted due to the personal variations in different individuals in yawning behaviour although the additional detection of yawning is a functionality which the project would have included if time permitted [14].

With the metric chosen, the project then developed and evaluated three separate approaches to enable blink detection functionality. All three approaches utilised convolutional neural networks to determine if the driver's eyes were open or closed which the application could then time to determine a blink event and its duration. The choice to use a convolutional neural network was made based on previous research undertaken which showed that it has become possible in recent years to achieve high accuracy in blink detection using this method even on devices with limited resources [15]. The three approaches evaluated for this project were a custom Convolutional Neural Network based on the MobileNetV2 architecture [16], a prebuilt library Expo Face-Detector based on the GoogleMLKit mobile machine learning architecture [6] and finally an approach which used another variation of Google's machine learning technologies, the MediaPipe Facial Landmarks Detection architecture [7].

MobileNetV2

The methodology used in this particular approach was to firstly use transfer learning in the development of a Convolutional Neural Network utilising cropped images of eyes as the training data for the model. The model was designed as a binary classifier with the two classes being open eyes or closed eyes. As such, the training data was split into two labelled folders, one for each class. As the model was trained on cropped images of eyes, to use it with live video the next step was to leverage the Expo Face-Detector library to process the live video, detect the presence of a face and the corresponding boundary coordinates of the eyes of that face. These boundaries could then be used to crop the image around the eyes and return two cropped images, one from each eye. These images could then be fed into the trained model which would return a prediction for each eye to determine if they were open or closed.

The MobileNetV2 machine learning architecture was selected as the basis of the model which would be trained via transfer learning due to its high suitability for real-time computer vision tasks on mobile platforms. The architecture offers a pretrained model that has 53 layers and contains a model pre-trained on more than one million images [16]. The model was adjusted to this project's requirements by creating a layer with one node and setting it as the output layer of the mode. This layer was created with sigmoid as its activation function to suit the binary classification requirement with the model returning a probability between 0 and 1 with prediction less than 0.5 indicating one class and above 0.5 the second class. The model was trained for 20 epochs on a dataset which combined two open-source datasets of eye images, the MRL dataset containing 84,898 images and the Closed Eyes in the Wild (CEW) dataset containing 2,425 images [17,18].

Table 1. MobileNetV2 Model Performance			
Accuracy	Precision	Recall	F1-Score
.96	.96	.96	.96

Figure 20. Table showing the MobileNetV2 model's performance with classifying eyes open/closed.

The size of this dataset created challenges in the amount of resources required to train this model with the first attempts on local hardware failing due to lack of system memory. This challenge was overcome by utilising Google Collab which allowed the model to be trained by using a high-ram runtime [19].

The eventual model performance achieved shows how effective CNNs can be at performing blink detection with an overall accuracy of 96% with a training time of twenty epochs. False Positives and False Negatives were low with both Precision and Recall scores set at 96% also.

Facial Landmarks Detection

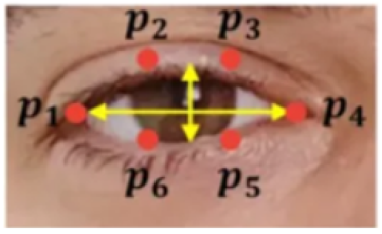


Figure 21. The keypoints used for the calculation of EAR.

To fully explore the optimal approach to blink detection, another methodology for blink detection was tested which uses a different mechanism to determine if an image of a face has eyes open or closed. For this approach, the Google ML library Facial Landmarks Detection is utilised to detect a face from the video input. This library developed by Google utilises a CNN powered by MobileNetV2 as the backend model architecture to detect any faces in the image and return 486 points around each face detected.

The model returns an object with a facial bounding box defined as x,y and z coordinates and the array of the key points in form x,y,z from which the key points corresponding to the left and right eye can be extracted. These points are then used to calculate the Eye Aspect Ratio (EAR) which is used to predict if an eye is in an open or closed state.

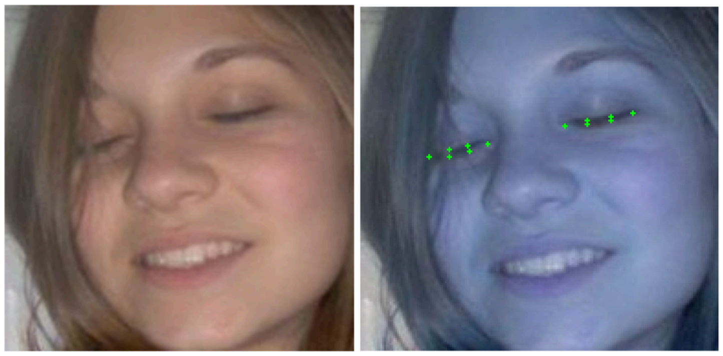


Figure 22. An example of the model detecting the necessary keypoints to calculate the EAR

The EAR ratio is a scalar which will change in real time as the aspect ratio of a subject's eye fluctuates allowing the detection of whether an eye is in an open or closed state. This value can be calculated from the points with the following formula as discussed in the design document [20].

$$EAR = (\| p2 - p6 \| + \| p3 - p5 \|) / (2 * (\| p1 - p4 \|))$$

Once this approach was implemented, the model was evaluated using the Closed Eyes in the Wild (CEW) dataset to determine performance. One of the key hyperparameters associated with this detection mechanism is the threshold EAR value which distinguishes between open and closed classes. Previous research conducted prior to development suggested 0.3 as a possible starting point so this was the initial value tested [21].

Once the model was developed however, testing found that this value was not the optimal figure for best classification performance. The results obtained shown in Table 2 instead found the optimal value for this implementation to be between 0.1 and 0.2.

Table 2. EAR Model Performance				
EAR Threshold	Accuracy	Precision	Recall	F1-Score
0.05	.78	.85	.77	.77
0.1	.93	.93	.93	.93
0.15	.95	.95	.95	.95
0.2	.91	.92	.91	.91
0.3	.66	.80	.67	.63

Figure 23. Table showing the EAR model's performance with classifying eyes open/closed with various EAR threshold values.

An EAR threshold of 0.15 returned the best accuracy of any of the values tested with an accuracy of 95% over the images dataset. The next closest threshold was 0.1 with an accuracy of 93%, however further analysis of these results is needed to also minimise false negatives.

As this application is dealing with human safety, reducing the occurrence of false negatives is arguably more important than small differences in accuracy as if the model predicts open while a driver's eyes are actually closed, that is a much greater problem than the model generating a false alarm. While this may be annoying to a driver, it is not causing a potential safety issue. Given this requirement, the value of 0.15 was found to be the optimal value returning the highest accuracy and lowest overall occurrence of false negatives.

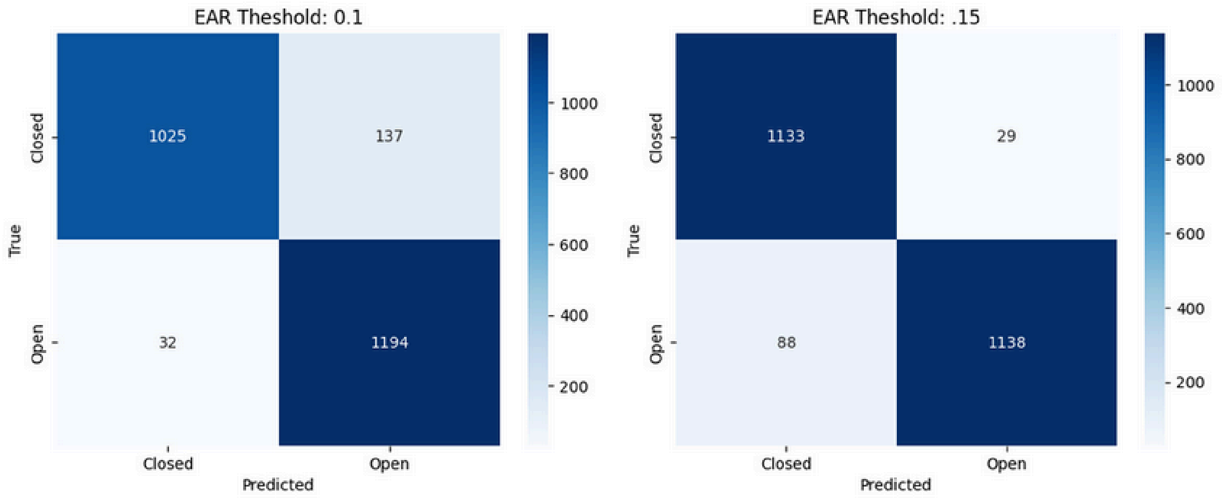


Figure 24. A comparison of the confusion matrices produces with the two best performing EAR models.

Technical

React Native

As the key technology utilised in this project, the largest technical learning was around the use of the React Native framework. Through the course of the project, I slowly developed a deeper understanding of the framework and its fundamental concepts around components, props and particularly the subtleties of state management. This was my first experience with a functional programming approach where components are written as functions and ideally input data is immutable with global effects being handled through the use of state and functions such as useState. I also gained experience with the development of user-friendly User Interfaces and how React Native can be used to develop modular and dynamic User Interface elements.

Gradle/ Build Tools

A significant aspect of the project involved understanding Gradle and other build tools for the compilation and packaging of React Native mobile applications using the Expo framework. This involved how to configure build settings, the management of dependencies and ensuring compatibility with various Android SDK versions.

Firebase

Incorporating Firebase into the application provided my first experience with the technology and was a valuable learning experience in NoSQL databases and cloud technologies.

TensorFlow

Although the attempts of integrating the model into a React Native mobile application proved to be failures, this project still allowed me to explore TensorFlow for the tasks of facial detection. This project was my first experience with the use of Transfer learning to speed model development and I gained valuable insights into how to configure and develop models using Convolutional Neural Networks.

Personal

Problem Solving

Throughout the project, there were many challenges to be overcome that required adopting a problem solving mindset with each obstacle encountered always needing a new approach to be considered. Often, these new approaches would introduce new challenges that had to be resolved. At the outset of the project, I made many mistakes where I would jump on the first solution that I could think of without considering all approaches which ultimately led to me wasting a lot of time on the non-optimal solution to an issue. This project taught me to take a more considered approach to problem solving and consider each solution and the problems it is likely to introduce instead of just jumping to coding immediately.

Persistence

Another personal learning from the project was the importance of persistence when faced with an obstacle. During the development process there were many unforeseen roadblocks which often meant that work that I had spent a lot of time creating had to be thrown away and new approaches taken. This was often very discouraging but this process did teach me that even problems that seem insurmountable can be solved through perseverance and determination in the face of setbacks.

Time Management

Time Management is another skill that was crucial in juggling this project's deadlines with the various work and projects of other modules of the course. I adopted a calendar system where each deadline was marked with an accompanying percentage of work remaining. This approach worked for me allowing me to better manage which project I would next focus on based on the amount of work remaining versus the time remaining until the particular deadline. Once I started using this system, I found this allowed me to better manage my time and ensure that each project was completed in time for each deadline.

Adaptability and Flexibility

While I have some experience with the development of projects using the agile process, this was my first foray into a large individual project developed from the ground up utilising an agile methodology. Given the many unknowns and new technologies required to bring the project from conception to the current state, the agile methodology of embracing change and remaining flexible proved to be an invaluable tool but it would be a lie to say that an agile mindset always came easily to me and I made many mistakes along the journey.

One challenge that I had not anticipated when executing an agile project was how difficult it was to overcome the instinct to look past the central functionalities and begin to focus too much on what the finished completed project should look like. Particularly at the start of the project, I often deviated from the Agile methodology of focusing on the present and allowing the project requirements to evolve from the work in the current iteration. This was probably a factor in some of the deadends I wasted time working on as some of these setbacks were based around advanced functions I was envisioning for the final product instead of focusing my efforts on getting the central functionality needed for the current iteration working first and allowing the future shape of the project to evolve from the experience and feedback gained from the completion and testing of the current iteration.

This experience has underscored a valuable lesson, establishing a robust foundation centred on a project's core features in the early stages enables each subsequent iteration to build upon the insights gained. Focusing too much on future functionalities can lead to wasted time, as these efforts might head in the wrong direction initially. This approach reinforces the importance of grounding each phase in the lessons of the previous ones to ensure continuous, directed progress.

The many deadends encountered instead necessitated adopting a flexible approach where I focus on the current iteration enabling me to remain flexible and not be wed to my first idea of what the final product should look like. This taught me that there is no one optimal solution to any problem and to adopt what works instead of your first idea.

Project Requirement Fulfilment

Achieved

- Face Detection
- Alarm for loss of Face Detection
- Driver Fatigue Detection
- Alarm for Driver Fatigue Detection
- Excessive Speed Detection
- Alarm for Excessive Speed Detection
- Multiple Alarm Types for Speed Alarms
- Permissions Management
- Light Sensor to disable Fatigue Monitoring in low light conditions
- Persistent User Settings
- Driver Distraction Detection

Not Achieved

- Separate Driver Profiles
- Addition of yawn monitoring to Driver Fatigue Detection
- Sunglasses Detection
- Gaze Estimation Model

Project Retrospective

The start of this project was delayed by a personal injury which resulted in the loss of around one month at the start of the development process. Given that the project was thus delayed, one change I would make is to adapt the development process earlier with a revised project plan which accounted for this time loss as I jumped into the project without accounting for the time loss which led to an unrealistic expectation of what could be achieved in Iteration One. This misstep meant a lot of the work for Iteration One being rushed with no grounded project plan guiding the development. In later Iterations, I made sure to have a rough project plan to follow which led to more measured progress in later Iterations.

Another change I would make for this project is that I would not use React Native as the framework. Although React Native is a powerful framework for cross-platform development, its use caused much more problems than I had foreseen in implementing the machine learning aspects of the project. The decision to develop a cross-platform application guided the decision to pick React Native but given the many unknowns at the outset of the project, the complexities this decision introduced proved it to be a mistake. Instead, I would consider simply using a native coding language such as Kotlin and develop the application for Android only initially which would eliminate all the issues that were caused in developing and packaging the application from React Native to Android. Many of the major deadends I encountered involved the building of versions which would work on web during build and testing and then fail during building for mobile applications due to various issues such as package incompatibilities in certain contexts, package conflicts and Android SDK compatibilities. I was unable to test and build iOS versions of the application due to not having access to a Mac computer so the cross-platform version of the application was never achieved anyway [22].

As mentioned earlier in the document, another major mistake that I made during this project was working on versions and functionality which ultimately did not work out and ended up in the Scrap folder on my computer. I wasted weeks after Christmas trying to switch to a different camera library which was capable of outputting the images as Tensor arrays which could be input directly into a custom model. Ultimately, while I managed to get this working as a camera and outputting Tensors as required, this version had a build issue where it would not build when integrating into the application as developed so far due to package version conflicts. Again I wasted the first few weeks of the final iteration again trying another approach using React Native Vision Camera which would work as a camera but would not build when the Face Detection was implemented due to build issues around the CMake translation of the C++ code within the Face Detection library. If React Native is to be used and I started this project from scratch, I would not rely so heavily on third-party libraries but instead create my own plugin with the exact project requirements and dependencies in mind so these build issues would not be a problem and I could be assured it would work as intended. Unfortunately I only gained the knowledge to create my own library late in the process but if starting over this would definitely be the approach I would take. I would also pivot more quickly when a particular solution is introducing more problems than it is solving and not waste so much time on what ultimately turned out to be a complete deadend.

While there is a balance to be maintained while working on Agile projects between how much research and planning is done before starting to code, for this project, I think I often jumped to coding somewhat sooner than I should. Agile recommends allowing the project requirements to evolve from the lessons learned in coding and getting feedback on previous iterations but if you are jumping to coding before taking the time to actually reflect on the successes and failures in that previous iteration, you are missing a vital piece of the Agile process. This is related to the problem of too much time wasted on solutions which ultimately did not work as many of the issues were ones that could have been foreseen if I had more thoroughly considered the solution picked before starting to code. I would adopt this philosophy if restarting the project from scratch and spend much more time using Agile tools like retrospectives and iteration planning to evolve the course of the next iteration rather than rushing to implement the first possible solution that I conceived.

Conclusion

Despite encountering numerous obstacles, making many missteps and facing several deadends over the course of this development journey, the project did manage to deliver the core functionalities of the application despite all the setbacks. The form and implementation of those functionalities were often very different from my initial ideas which illustrated to me the value of remaining flexible when faced with problems and not to be afraid to change approaches when faced with roadblocks.

The process of completing this project from a start point of complete inexperience in the necessary technologies presented endless learning opportunities and the process taught me so much about the difficulties of developing mobile applications and delivering machine learning functions on platforms with limited resources.

I believe that if I was asked to create another mobile application today, I would be capable of planning, documenting and implementing that application in a much more efficient and effective manner. This belief is based on the amount of development in skills that this project has enabled including technical skills, ability to effectively plan projects and personal understanding of what it takes to smoothly manage the process of achieving a successful Agile project.

Future Steps

For any future development efforts going forward, there are still many improvements that could be made to current functionalities and additional functionalities that could be introduced to enhance the overall product.

Test Deployment

The immediate next steps for this project if it was to continue would be the Initiation of a beta test on the Google Play store. This is an essential next step to gather valuable user feedback to ensure the app's functionality meets the expectations and needs of the target audience. This phase will allow real-world testing with a large group of users of the application’s features allowing the final streamlining and finetuning of the application’s functionality before the initiation of a full release.

Functional

From a functionalities viewpoint, the next steps for the application would be around improving the reliability and function of the three main monitoring systems.

- For fatigue detection introducing additional indicators such as yawning or frequency of blinking could be used to improve overall accuracy of detection.
- With the driver distraction detection functionality, the inclusion of a model capable of gaze estimation has the potential to significantly increase its ability to detect other potential driver distraction indicators.
- Another possible feature would be the ability to detect if a driver is wearing sunglasses so a warning could be issued and monitor deactivated due to the difficulty of determining eyes being open/closed in this environment.
- The addition of a capability of the application connecting to external cameras with night vision functionality to enhance monitoring functions in low light conditions.

Technical

On the technical side, one major improvement that would be planned for future iterations is switching the camera input and face detection functionality to utilise the React-Native-Vision-Camera library [23]. This library has the potential to introduce many benefits to the application including better performance, extended support as Expo Camera is being deprecated in future Expo SDK versions and better support for background processing of images. This integration had in fact been mostly complete but was excluded from this final release due to time constraints and instead saved as a future version in the project repository under future technology so another few weeks could see this function easily integrated into a future release.

References

1. International Transport Forum (2021) Ireland: Road Safety Country Profile, 2021 - international transport forum. Available at: https://www.itf-oecd.org/sites/default/files/ireland-road-safety.pdf.
2. Road Safety Authority Ireland (2023) *Driver fatigue*, *RSA.ie*. Available at: https://www.rsa.ie/news-events/events/details/2023/01/30/default-calendar/driver-fatigue (Accessed: 05 November 2023).
3. Technostacks (2023) *Top 10 mobile app development frameworks in 2023: Technostacks*, *Technostacks Infotech*. Available at: https://technostacks.com/blog/mobile-app-development-frameworks/ (Accessed: 05 November 2023).
4. Pregasen, M. and Team, R. (2023) *Putting the expo vs react native debate to rest*, *Retool*. Available at: https://retool.com/blog/expo-cli-vs-react-native-cli (Accessed: 18 April 2024).
5. Clark, J. (2022) *Firebase Advantages and disadvantages*, *Back4App Blog*. Available at: https://blog.back4app.com/firebase-advantages-and-disadvantages/ (Accessed: 05 November 2023).
6. Expo (2023) *Facedetector*, *Expo Documentation*. Available at: https://docs.expo.dev/versions/latest/sdk/facedetector/ (Accessed: 06 December 2023).

7. MediaPipe (2024) *Face landmark detection guide | mediapipe | google for developers*, *Google*. Available at: https://developers.google.com/mediapipe/solutions/vision/face_landmarker (Accessed: 18 April 2024).
8. React.dev (2024) *React developer tools*, *React*. Available at: https://react.dev/learn/react-developer-tools (Accessed: 18 April 2024).
9. Boschi, M. (2021) *React native - when JS is too busy*, *DEV Community*. Available at: https://dev.to/matteoboschi/react-native-when-js-is-too-busy-5fhn (Accessed: 18 April 2024).
10. OpenStreetMaps (2022) *API*, *API - OpenStreetMap Wiki*. Available at: https://wiki.openstreetmap.org/wiki/API (Accessed: 18 April 2024).
11. Abzianidze, G. (2023) *Haversine formula in react native.*, *Medium*. Available at: https://medium.com/@gega.abzianidze.1/haversine-formula-in-react-native-abda04843888 (Accessed: 06 December 2023).
12. Road Safety Authority (2023) *Driver fatigue*, *RSA.ie*. Available at: https://www.rsa.ie/news-events/events/details/2023/01/30/default-calendar/driver-fatigue (Accessed: 06 December 2023).
13. Babu, A. (2022) *Detecting drowsiness in drivers using approaches based on machine ...*, *Detecting Drowsiness in Drivers using Approaches based on Machine Learning Methodologies*. Available at: https://norma.ncirl.ie/6672/1/ashwinsureshbabu.pdf (Accessed: 05 November 2023).
14. Civik, E. and Yuzgec, U. (2023) *Real-time driver fatigue detection system with deep learning on a low-cost embedded system*, *Microprocessors and Microsystems*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0141933123000972 (Accessed: 05 November 2023).
15. Phan, A. and Nguyen, N. (2021) *(PDF) an efficient approach for detecting driver drowsiness based on ...* Available at: https://www.researchgate.net/publication/354550914_An_Efficient_Approach_for_Detecting_Driver_Drowsiness_Based_on_Deep_Learning (Accessed: 05 November 2023).
16. MathWorks (2024) *ImagePretrainedNetwork*, *(Not recommended) MobileNet-v2 convolutional neural network - MATLAB*. Available at: https://www.mathworks.com/help/deeplearning/ref/mobilenetv2.html (Accessed: 18 April 2024).
17. Fusek, R. (2021) *MRL Eye Dataset*, *MRL*. Available at: http://mrl.cs.vsb.cz/eyedataset (Accessed: 05 November 2023).
18. Tan (2014) *Closed eyes in the wild (CEW)*, *The Closed Eyes in the Wild (CEW) dataset*. Available at: http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/ClosedEyeDatabases.html (Accessed: 18 April 2024).
19. Google Collab (2024) *Google colaboratory*, *Google Colab*. Available at: https://colab.research.google.com/notebooks/pro.ipynb#scrollTo=23TOba33L4qf (Accessed: 18 April 2024).

20. Jayarathne, J. (2021) *Eye-blink detection*, *Medium*. Available at: https://jitharijayarathna.medium.com/eye-blink-detection-5d6854520217 (Accessed: 06 December 2023).
21. Dewi, C. *et al.* (2022) *Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks*, *PeerJ. Computer science*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9044337/#ref-36 (Accessed: 06 December 2023).
22. Singh, S. (2023) *Connecting a real IOS device to Xcode for react native development: A step-by-step guide*, *Medium*. Available at: https://sugandsingh5566.medium.com/connecting-a-real-ios-device-to-xcode-for-react-native-development-a-step-by-step-guide-c0e3817a1893 (Accessed: 19 April 2024).
23. Rousavy, M. (2024) *Visioncamera documentation: Visioncamera*, *Documentation*. Available at: https://react-native-vision-camera.com/ (Accessed: 19 April 2024).

Declaration Form

Work submitted for assessment which does not include this declaration will NOT be assessed

DECLARATION

1. **I declare that all material in this submission e.g., thesis/essay/project/assignment is entirely my own work except where fully acknowledged.**
2. **I have cited the sources of all quotations, paraphrases, summaries of information, Tables, diagrams, or other material; including software and other electronic media in which intellectual property rights may reside.**
3. **I have provided a complete bibliography of all works and sources used in the preparation of this submission.**
4. **I understand that failure to comply with SETU's regulations governing Plagiarism constitutes a serious offence.**

Student Name (Printed):	SHANE KENNEDY
Student Number(s):	C00263504
Student Signature(s):	[illegible]
Date:	19/04/24