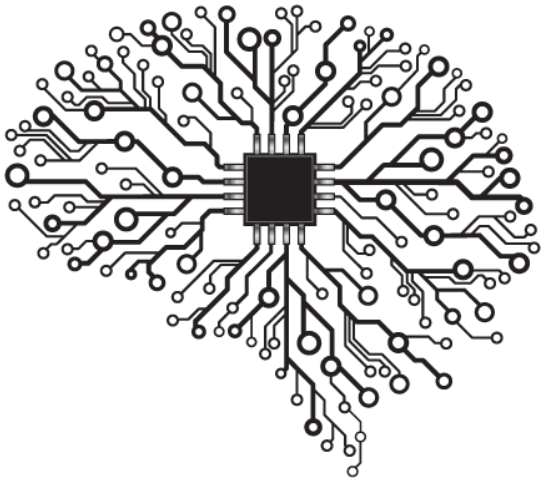
SafeDrive AI

Design Document

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



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

Tables of Contents

Introduction

The aim of this project is to develop a cross platform mobile to increase the safety of users while they are driving on the road. Drivers can install this application on their mobile devices and once activated the application will use machine learning technology to monitor the driver and their location details and issue a warning if the application detects that the driver may be fatigued, distracted or exceeding the speed limit. These conditions are often contributing factors to accidents so by making drivers aware that they may be at risk, this application will increase the safety for both the driver and other road users as well.

This design document details a high level model of the operation of the mobile application, its underlying technologies, how users will interact with the application and the research that informed the approach to its design. The intended purpose is to give the reader insight into the central design decision for the proposed application.

Introduction

This document follows on from the functional specification for the application which set out the requirements that the application should fulfill. This design document will focus on how those requirements will be best achieved and the high level design decisions that will steer the specifics of their implementation going forward.

The detection of driver fatigue through the use of machine learning is the key functionality of this application and as such the document will first look at the best approach to implement this requirement. This section will look at both the best methodologies for detecting driver fatigue and their advantages and disadvantages. Once the primary functionality of the application is complete, the secondary monitoring focus will be the design of the systems for detecting driver distraction and excessive speed, therefore the design document will next discuss the design methodologies for the detection of these two driver safety concerns.

The following section will delve into the technological stack that will be used to implement the chosen methodologies, the reasons they were chosen and how they will be used to build this application. This section will elucidate the design of the machine learning architecture, the front and backend technologies which will power the application and the tools which will be used in the application’s creation.

The final two sections will illustrate the interactions of the components of the application through the use of sequence diagrams and a high level overview of the application's user interface with a flowchart of its navigation menus and prototype screenshots of the application.

Drowsiness Detection

Driver drowsiness or fatigue is a leading contributing factor in road traffic accidents around the world with the Road Safety Authority of Ireland (RSA) estimating it to be a factor in one in five road traffic collisions [1]. 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 its user. As such, the methodology that the machine learning algorithm will use for detecting driver fatigue is a key design decision for this project.

There has been much research on the use of machine learning in recent years that this project draws on in the design of the drowsiness detection system. The primary metrics that such a system may use to determine if a driver is fatigued are blink duration, yawning and head movement. These indicators can be obtained and analyzed through the use of machine learning to track key facial features [2]. However, yawning and head movement while they can be indicative of drowsiness are often less specific and can vary by interpersonal factors [3]. The proposed design will emphasize the use of blink duration as the central parameter for the first iteration due to its advantages in terms of accuracy, simplicity and real-time responsiveness. The second iteration will utilize analysis of both blink duration as a primary indicator and yawning as a secondary indicator as this multifactorial approach is found to be most successful in terms of accuracy with studies showing that this combination for fatigue detection can achieve accuracy level of up to 97% [4].

Blink Detection

Facial Landmark Detection

The detection of blink duration is the primary methodology chosen to detect driver drowsiness. A facial landmark detection library is first used to detect key points on the face corresponding to a number of points around the eye. The Google Mobile Vision package offers a number of facial detection libraries with different capabilities for this application [5]. The Expo Face Detector is part of this suite and offers easy integration with React Native, face tracking functionality and eye blink detection built in [6]. The disadvantage of this library however, is that it offers only a limited number of key detection points. Another

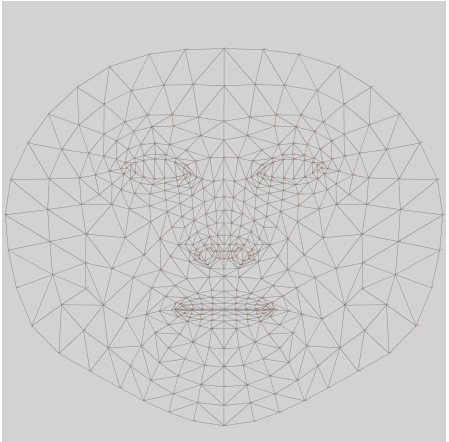


Figure 1. Face contour map of 3D Face mesh from Face-Detection-Landmark library [7].

library powered by Google ML kit which offers more comprehensive facial landmark detection capabilities is the FaceMesh library and its updated version Face-detection-landmarks [7]. This library developed by Google utilizes a CNN powered by MobileNetV2 as the backend model architecture to detect 486 points around any face detected in an image. The model returns an object with a facial bounding box defined as x,y and z coordinates and an array of the key points in form x,y,z which conform to a 3d model face mesh [8]. Once a face has been detected the key points corresponding to the left and right eye can be extracted from this set of points. As part of the development process, this project plans to use a combination of these libraries and as development proceeds discover which library offers the best performance for each of this project's specific requirements.

Processing Face Data

There are two primary methodologies that can be utilized to process the returned face landmark's data to perform blink detection. These two methods are the calculation of the Eye Aspect Ratio to detect the eye state as explained in the next section or the use of the eye location data to first define a bounding box around the eye. This box can then be used to create a cropped image of the eye that can be passed onto a pre-trained machine learning model which can then classify the image based on its training to perform an eye state classification. The former method has been shown to produce highly accurate results but due to its higher overhead, the latter technique will be the primary method used for the first iteration of the machine learning implementation [9].

Eye Aspect Ratio

The Eye aspect ratio is a scalar value which can be calculated from a number of points around an eye. This value 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, the duration that the eye remains in a closed or semi-closed state and the frequency of blinks.
The image here shows the eye points that the Facial Landmark Detection library returns. The EAR can be calculated from the points with the following formula [10].

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

The formula utilizes the Euclidean distance between the points above and below the eyes divided by twice the distance from the points to the left and right of the eye to calculate the scalar value EAR. This value will drop to near zero when a subject blinks so a threshold value near zero will be chosen to indicate an eye in a closed state.

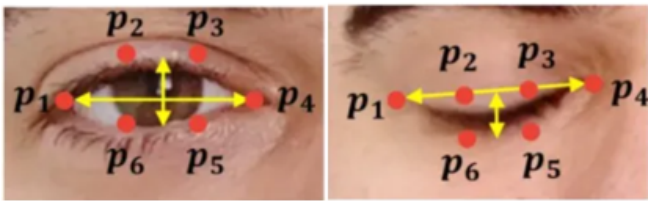


Figure 2. Horizontal and Vertical examples of how the calculation of Eye Aspect Ratio relates to eye open/closed state [10].

Research in this area has found an EAR threshold of 0.3 to be an effective first value for registering an instance of a blink so this is the initial value which will be used [11]. Further fine tuning of this value will then be performed to suit the non-functional requirements of this application.

Secondary Focus Detections

Driver Distraction

In the computer vision space, various methodologies have been explored in the use of machine learning for the detection of driver distraction. An effective approach to this problem has been the use of CNNs and given this project's use of a CNN for the detection of driver fatigue, the same approach can be used for the detection of Driver distraction. Effective approaches have been demonstrated which utilized CNN which tracked driver head position, posture and gaze estimation [12]. Given the limited resources of running these algorithms on a mobile platform, this project will utilize the facial landmarks data which will be collected already for the driver fatigue detection to track driver head position and gaze to make an estimation of driver distraction events defined by a duration in which the driver head position and gaze deviates from the average head and gaze position while driving.

Excessive Speed

At its core, the design of the Excessive Speed monitoring process revolved around continuously monitoring a Driver's location data and comparing it against the speed limit of the corresponding road section. This functionality will be achieved through the use of the mobile devices GPS data location and an external provider to return the current road's speed limit.

GPS Data Collection

The collection of the device's GPS location data will be performed by utilizing the Geolation library provided by react-native community. This library provides the functionality to read a device's location data and return the current location's latitude and longitude data [13]. This data can be used to calculate an estimation of the distance between two separate points across a time range and calculate the Driver's current speed. For an accurate estimation of the distance between these points, the curvature of the Earth must be considered. Here the application will use the Haversine formula which is a mathematical formula which calculates the distance between two points on a sphere and can therefore return a reasonable approximation of the distance between two sets of latitudes and longitudes across the Earth [14].

Estimation of Road Segment Speed Limit

For the retrieval of the current speed limit of the Driver's current location, the collation of a group of points into a path is required. The GoogleMaps API can take as a parameter this path and from this path can estimate the most likely road segment that the Driver is traversing [15]. This road segment can then be used to query the API for the road segment speed limit. In the application, the Driver can specify a time limit over which excessive speed can be detected

before a warning is triggered. If the Driver's average speed is greater than the current road segment being traveled for over this specified period, a warning for excessive speed can then be issued to alert the Driver to the potentially unsafe driving condition.

Technologies

Machine Learning Architecture

Model Selection

The selection of the machine learning model for this project is a key design decision for its implementation going forward. The basis for this decision was the functionality that the model would need to support which is essentially an image classification task and as such the model selected was the Convolutional Neural Network (CNN) due to its high suitability for such an application. CNNs great strength is their ability to recognize patterns in visual data and this makes them ideal for identifying signs of driver fatigue or the more complex task of identifying driver distraction [16].

The primary advantage of CNNs for this task are they offer high accuracy in image classification tasks, reduced calculation compared to other neural network types which is ideal for a mobile platform and they offer reduced complexity to implement compared to other models which is an advantage for this project due to the lack of developer experience with machine learning technologies [17]. The disadvantage of CNNs for this project is that they require large training datasets to gain the necessary accuracy, however this downside is mitigated by the large publicly available datasets which are available which can be utilized for model training.

Model Training

The model training approach that was chosen for this project centers on the use of transfer learning using a pretrained CNN. This method was selected due to its ability to reduce the need for intense computational power for model development by leveraging models that have already been trained for image classification tasks. These models then only need to be fine tuned to this project's specific requirements, saving computational resources and giving the project a strong foundation to build upon while also saving development time [18].

The machine learning framework that was selected to implement model training is the TensorFlow library. TensorFlow is highly suitable for this project and offers many advantages in both the training and deployment of this project's machine learning model. It has achieved wide scale adoption across the machine learning community and as such offers a comprehensive suite of both tool sets for training models as well as a strong selection of pretrained models with different capabilities that can be leveraged to implement this project's functionality [19]. This wide adoption also means that is capable with many languages and mobile platforms making it a good choice for development for a mobile application.The disadvantage of TensorFlow is its complexity to utilize and need for increased computational power, however the wide availability

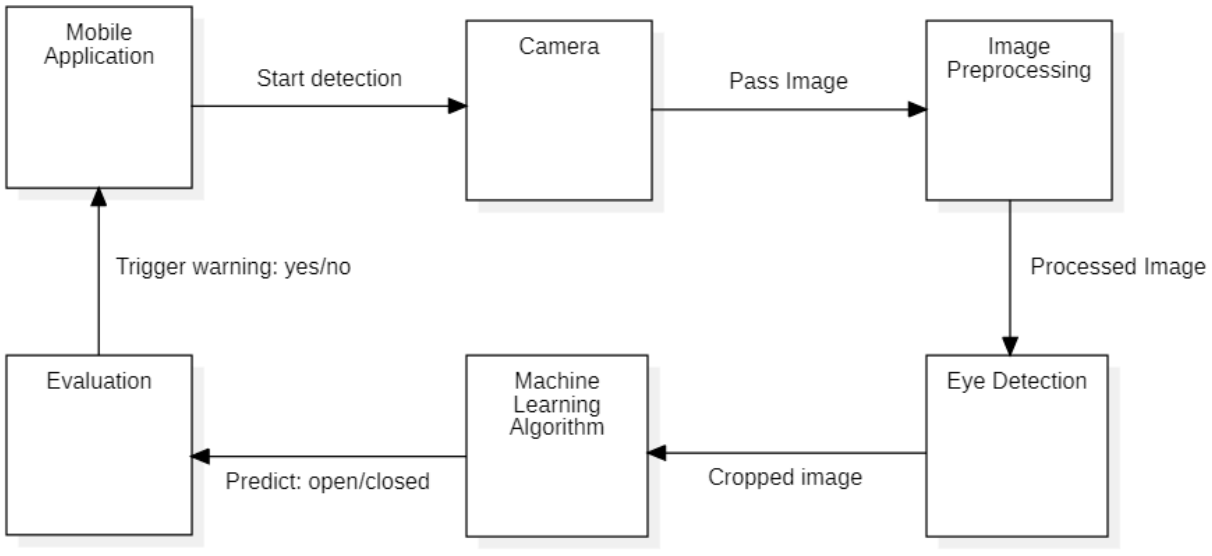
of training resources and use of transfer learning make this framework align with this projects requirements [20].

Architecture

The MobileNetV2 machine learning architecture was selected for this project due to its high suitability for real-time computer vision tasks on mobile platforms. This architecture offers an exceptional balance between computational efficiency while retaining high accuracy levels making it a strong candidate to fulfill the requirements of this project [21]. The architecture offers a pretrained model that has 53 layers and contains a model pre-trained on more than one million images. This project will leverage this model and utilize TensorFlow combined with transfer learning to train its parameter weights to the specific needs of this task [22]. The model will be trained using TensorFlow for the binary classification task of predicting whether an image is of an open or closed eye over many epochs until the accuracy required is reached.

Once model training is complete the design will use the OpenCV library for image capture and cropping. This library was chosen due to its wide support and ability to process real time images efficiently [23]. The NumPy library will be used to reshape the images to the format needed for the machine learning input layer and the pre-processed images will then be passed to the model which will return a prediction of whether an eye is open or closed. This prediction can then be used to determine if a warning should be issued. Based on previous research and to maintain a balance of conservation of processing power while maintaining accuracy a target capture rate of 20 frames per second will be used [24].

Data Flow Model



Model Specification

The specification for MobileNetV2's pretrained model is tuned for efficiency targeting applications running on platforms with limited resources such as mobile devices. With a total of 3,229,889 parameters, MobileNetV2 offers an extremely lightweight design compared to other popular machine learning architectures such as InceptionV3 which has a much larger 23,851,784 parameter set [25]. Of these MobileNetV2 offers 3,208,001 trainable parameters giving it the flexibility to be tuned for this project's needs. Overall, MobileNetV2 provides adequate accuracy while performing well in terms of saving on mobile device processing power and battery life which is the reason it was chosen for this design.

Dataset

Name: MRL Eye Dataset
Description:
This dataset is a publicly available dataset of images of human eyes originally intended to be used for a gaze estimation project. It consists of 84,898 images of eyes collected from 37 subjects, 33 men and 4 women [26].
Preprocessing:
Data preprocessing will be performed where the dataset will be broken up into two binary classes, images with open eyes and images with closed eyes. The images will then be reformatted to the input shape of the machine learning architecture which is 224 pixels in height, 224 pixels in width and with 3 color channels (RGB) [27].
Link: http://mrl.cs.vsb.cz/eyedataset

Technology Stack

Frontend Technologies

The technology chosen to create the frontend of the application is React Native due to its robust ecosystem, versatility and cross platform capabilities. React emphasis on component based architecture facilitates rapid development through the creation of modular and reusable code blocks. React Native is powered by Javascript and a Javascript XML (JSX) and incorporates CSS for styling which allows the creation of attractive mobile applications.The strong ecosystem around React includes many available libraries that support integration of frameworks such as TensorFlow which is ideal for this project's design. React Native also allows the creation of a single codebase that can be deployed to both iOS and Android which is another big strength for this project's requirements [28].

Backend Technologies

This project's requirements include the ability for the application to send anonymous debug reports for quick debugging of any issues, an important feature for an application which must offer a high level of reliability as it is dealing with driver safety. For this requirement, Firebase was chosen as the backend server infrastructure due to its suitability for reliable, real-time data logging. Firebase's real-time database ensures that the debug reports are instantly accessible allowing fast response to any application's issues. Firebase also offers inbuilt security and authentication features which are easy to implement and deploy making it suitable for future development of more advanced Driver profile systems [29].

Machine Learning

The Project will use TensorFlow JS for model selection and model training and on device image preprocessing.

The Project will use the MobileNetV2 model as the model architecture upon which the machine learning algorithm will be built. This model architecture comes included in the Keras library of TensorFlow.

The OpenCV library will be used for the image preprocessing during model training before the images are passed to the machine learning algorithm.

Version Control

For application data backup and version control, this project leverages the industry standard of Git for local version control, paired with GitHub for remote backup. Git is renowned for its highly effective version control abilities and allowed seamless tracking of changes and enabled major changes to the project code to be stored in separate branches in case a reversion to the previous design proved necessary. This ability was important in this particular use case as the overall design changed and evolved over the course of the initial development period. GitHub as the chosen hosting service, provided a central repository for storing backups of the project code and was chosen for its ease of use and its ability to integrate well with the Git tool.

IDEs

The development tools used for creating the actual code were chosen to leverage their unique advantages for different applications. Visual Studio Code (VSCode) was used as the primary tool for coding of the React

Native frontend application due to its wide range of extensions for React coding. The VSCode integrated development environment provides a seamless environment for JavaScript and React development, offering features like integrated React code suggestions with IntelliSense and a rich environment of helpful extensions such as debugging support from the extension React Native Tools [30]. On the machine learning side, Anaconda paired with Jupyter notebooks were the IDE of choice with their particular strengths in this area. Anaconda's comprehensive data science toolkit, coupled with Jupyter notebooks ability to visualize the coding process allowing the easy exploration and iteration of the machine learning models. This combination of tools suits the particular use case, optimized the project's workflow and offered a more efficient development and design process.

Technical Difficulties

React-Native

Dependencies

The choice of using React Native for this project has caused many delays and technical difficulties due to the complex dependency tree required for this project and the many libraries which must be integrated together. An advantage of React Native is the wide array of community support and libraries available for use but this does lead to many instances of libraries which have conflicting peer dependencies. These conflicts can cause bugs and errors which can be difficult to track and resolve. These conflicts are responsible for much wasted time and effort as certain libraries have led to whole solutions which are attempted only to find out later that they are incompatible with other libraries which are required to implement the needed functionality.

Expo Camera

The Expo Camera library is being used to capture video images from the front or back camera of the mobile platform. The use of this library was one technical obstacle for development in this project.

1. First difficulty is that it is not compatible with Android emulators so development and testing of code must be performed either on web platforms, through the Expo Go app or by building a development build and running directly on a mobile device
2. Expo Camera library does not support frame processing or outputting images as tensors. The first attempt to overcome this issue was to pass the image to the machine learning algorithm encoding in Base64 format. However, Expo Camera only allowed images to be captured in a number of defined aspect ratios so for an application with dynamic window resizing this was not an ideal solution. The next solution was to utilize the react-native tensor flow library which has an extension for Expo Camera called Tensor Camera which

is capable of capturing whole frames as Tensor arrays. This package however caused many dependency errors to get working with the rest of the application.

3. Tensor Camera caused project delays to implement not just with dependency errors but also with errors caused by not having the Tensor Flow model ready before requesting a frame. This was solved by creating a function to ensure the model was ready before allowing the Tensor Camera component to be rendered.

TensorFlowJS

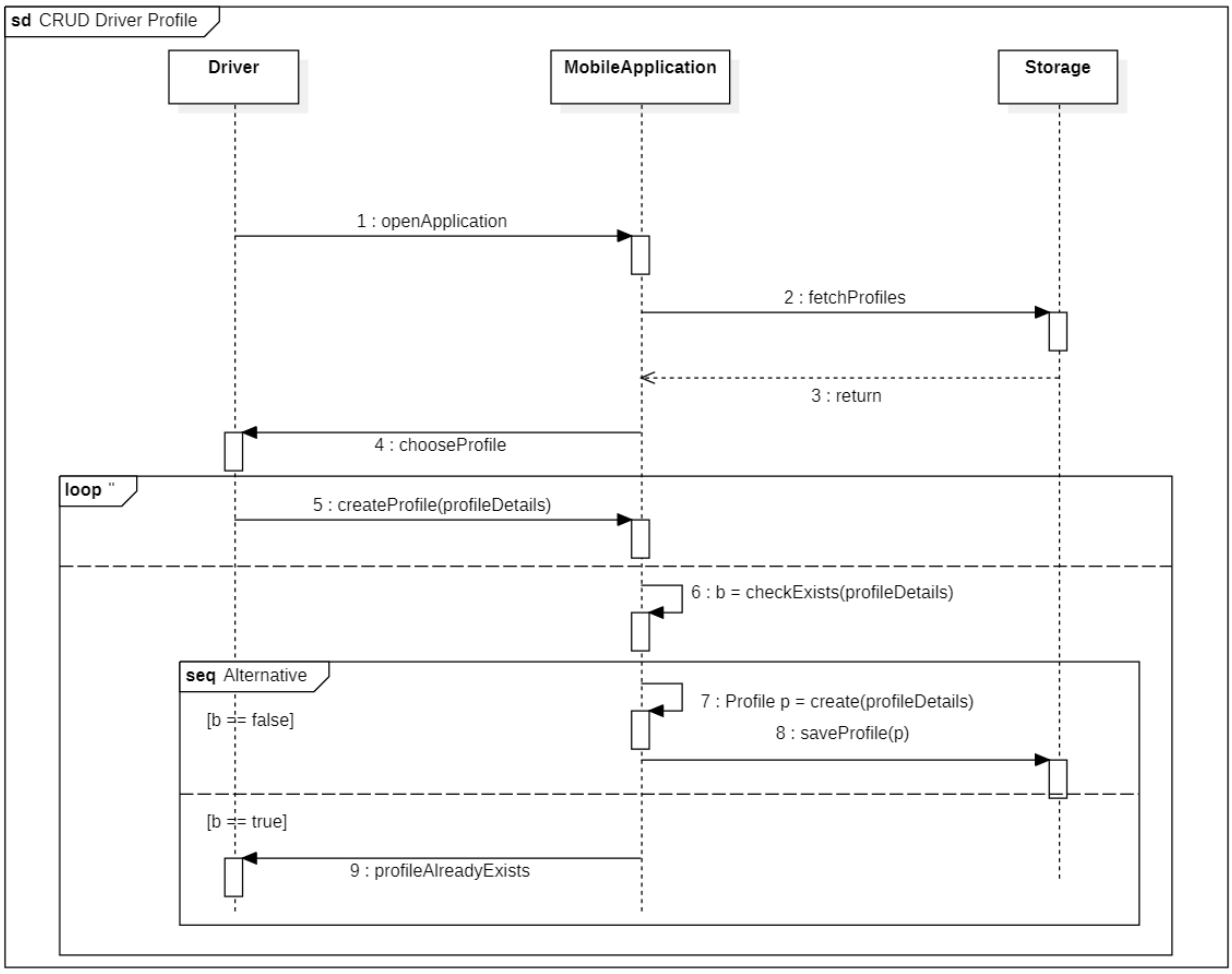
The development of machine learning functionality with TensorFlow for React Native required the use of TensorFlowJS. Any models which are developed in Python must be translated into Javascript and loaded into the application using the TensorFlowJS library. This use of this library leads to many dependency issues to get working, especially around implementing the correct backend engine to run on a mobile device.

Face-Landmarks-Detection

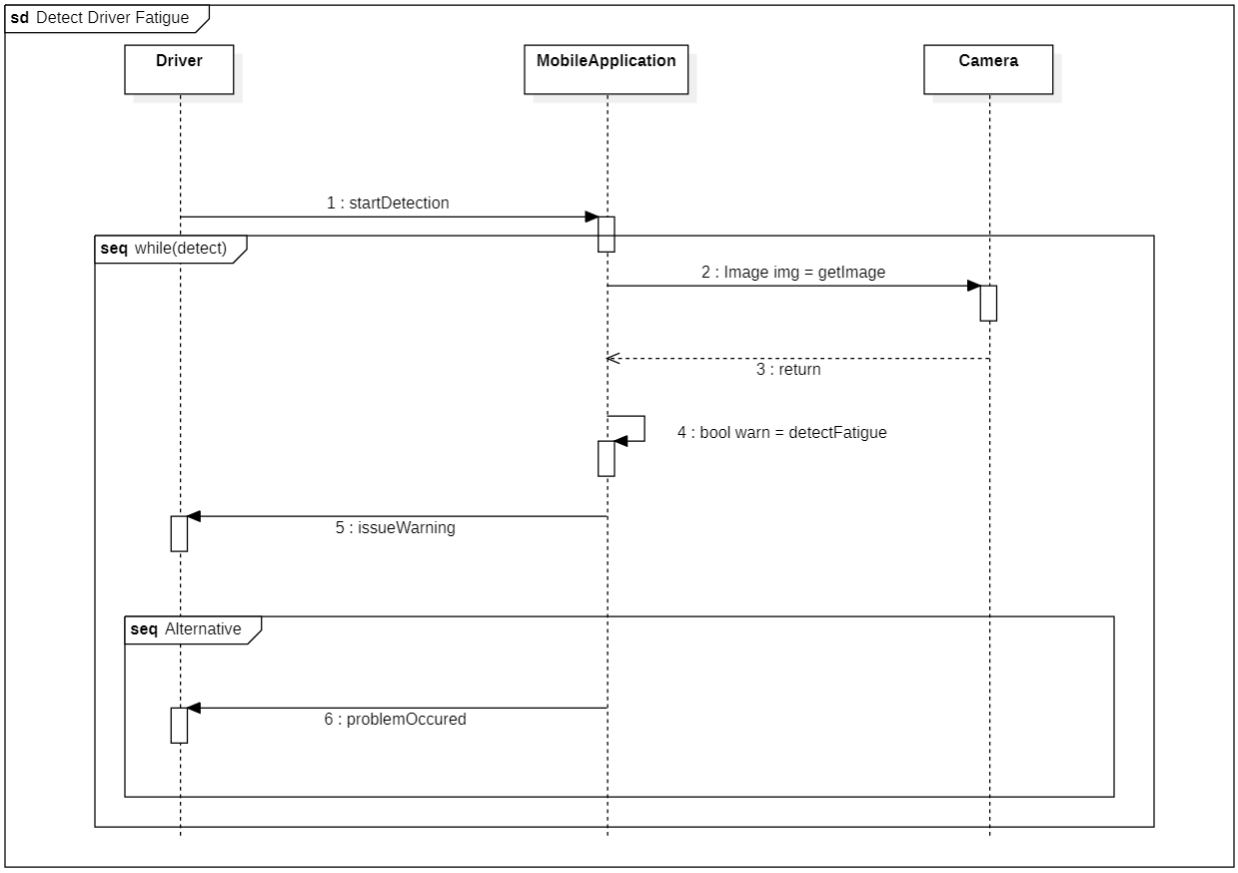
The Face-Landmarks-Detection library is part of the Google Machine Learning kit and is the newest version of their facial detection capabilities. The library is built on TensorFlow.Js and returns an array of 486 annotated key points from faces detected from an input parameter which can be in a number of different video,image or matrix array formats. This was the library that was first envisioned as providing the facial landmarks detection but this library caused major delays due to technical difficulties in implementation. The library works when the application is run on a Web browser but when ported to Android, it causes module import errors which have not yet been resolved. In the end, a switch to an older version called FaceMesh was required to get the mobile application working correctly on Android.

Sequence Diagrams

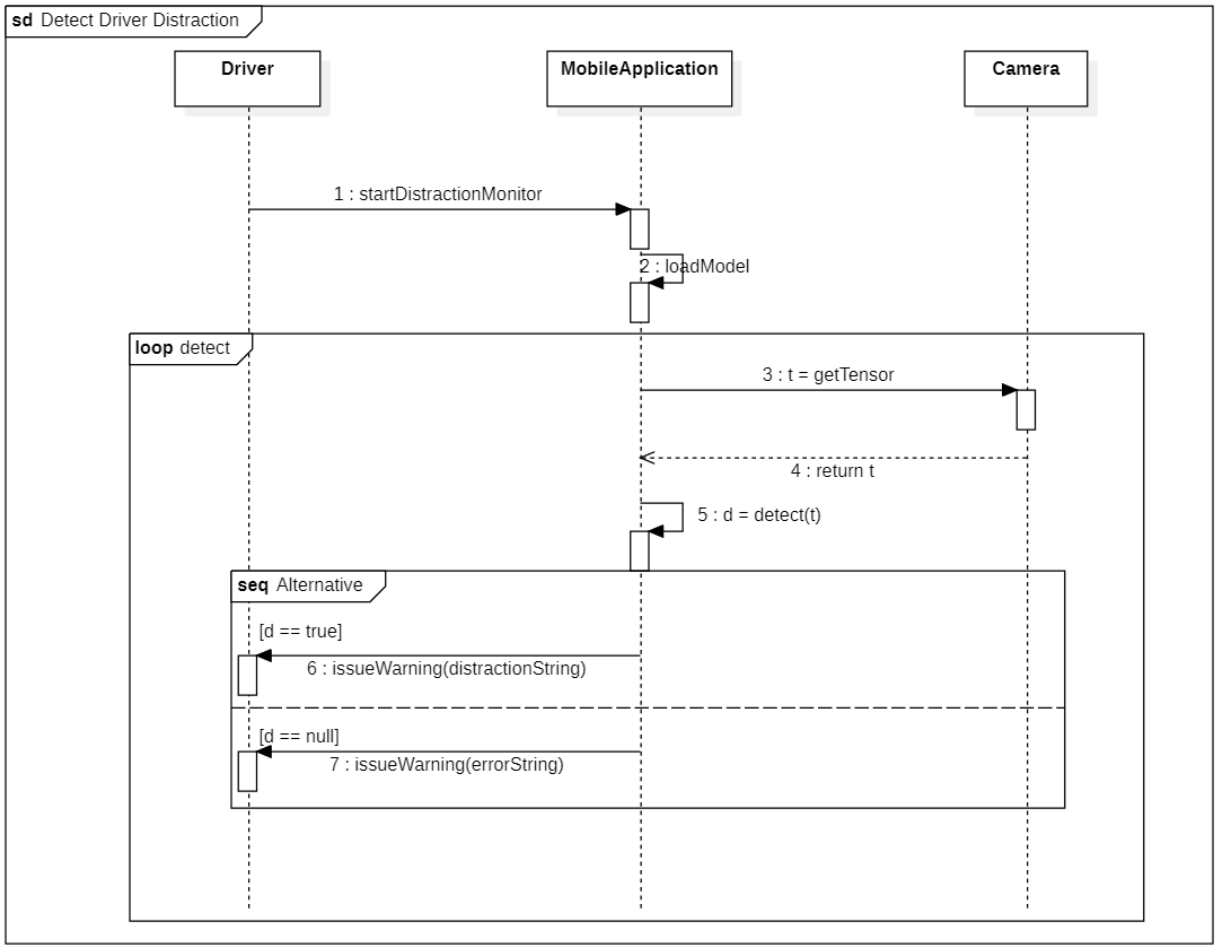
CRUD Profile



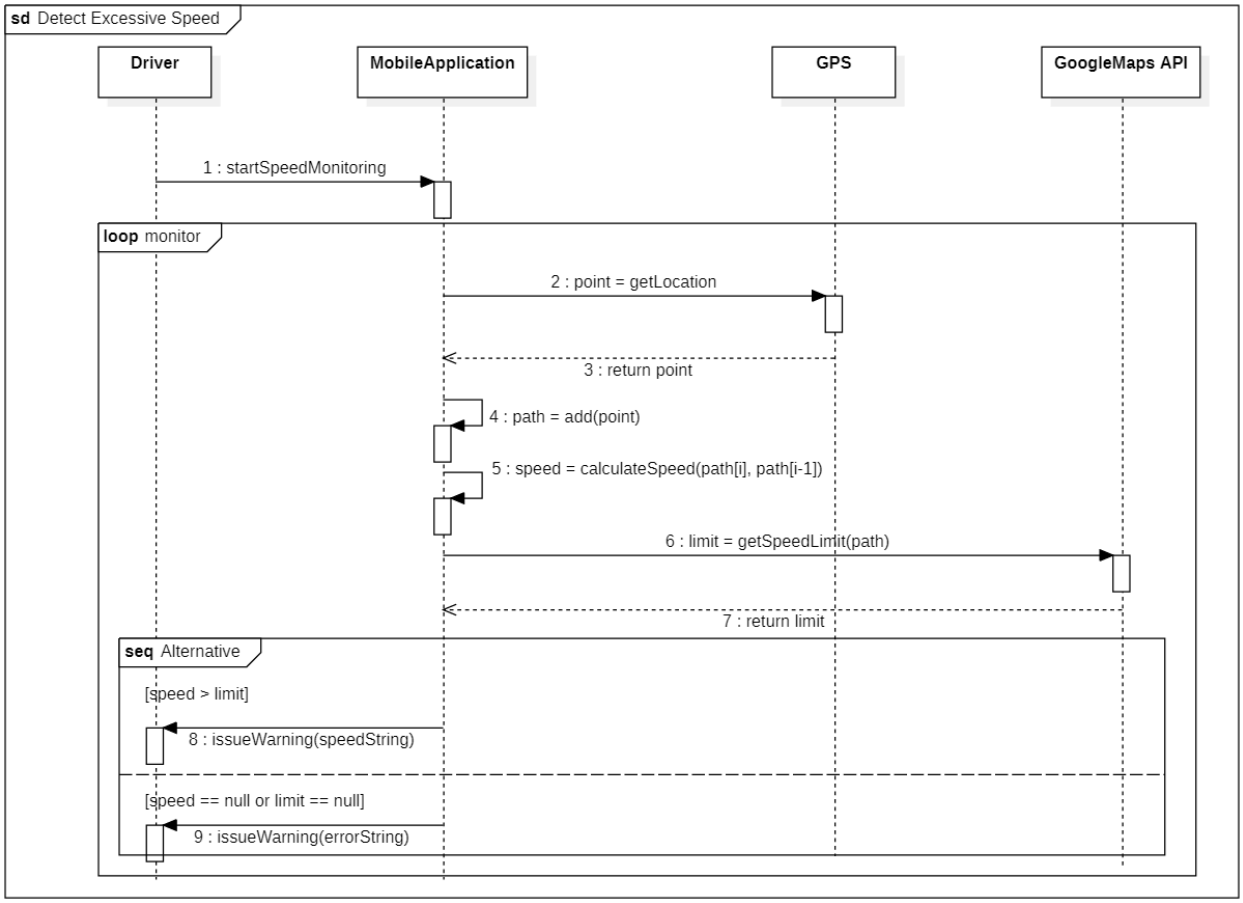
Detect Driver Fatigue



Detect Driver Distraction

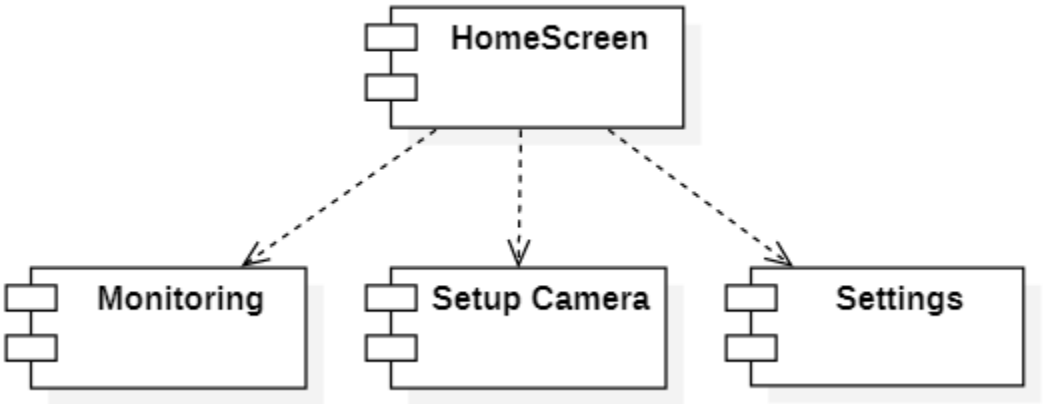


Detect Excessive Speed



UI Design

Navigation

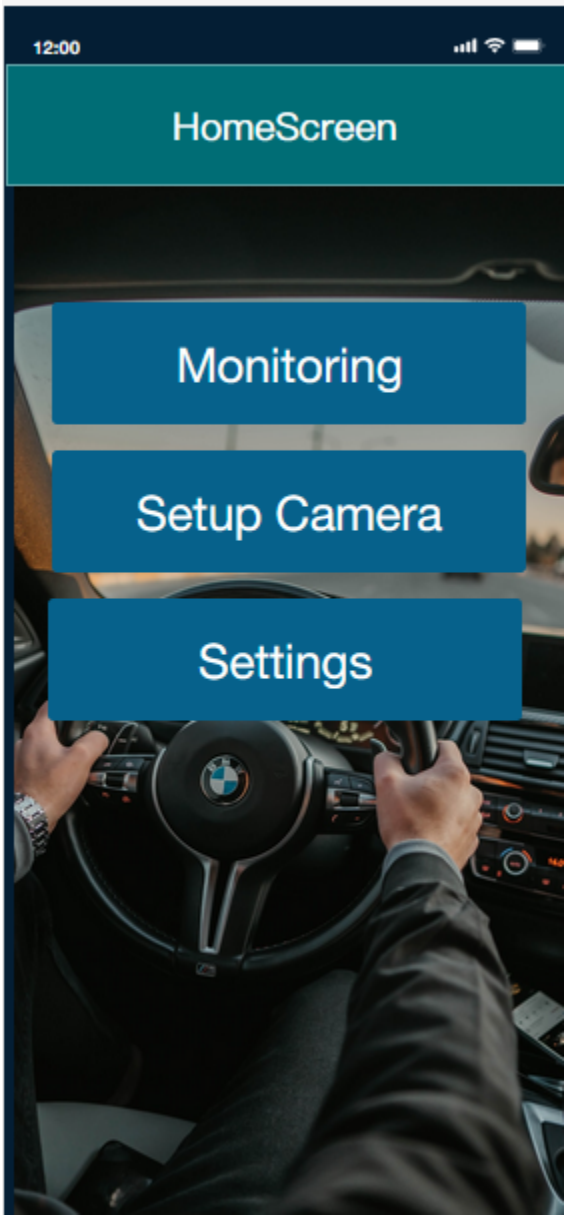


Navigation Design

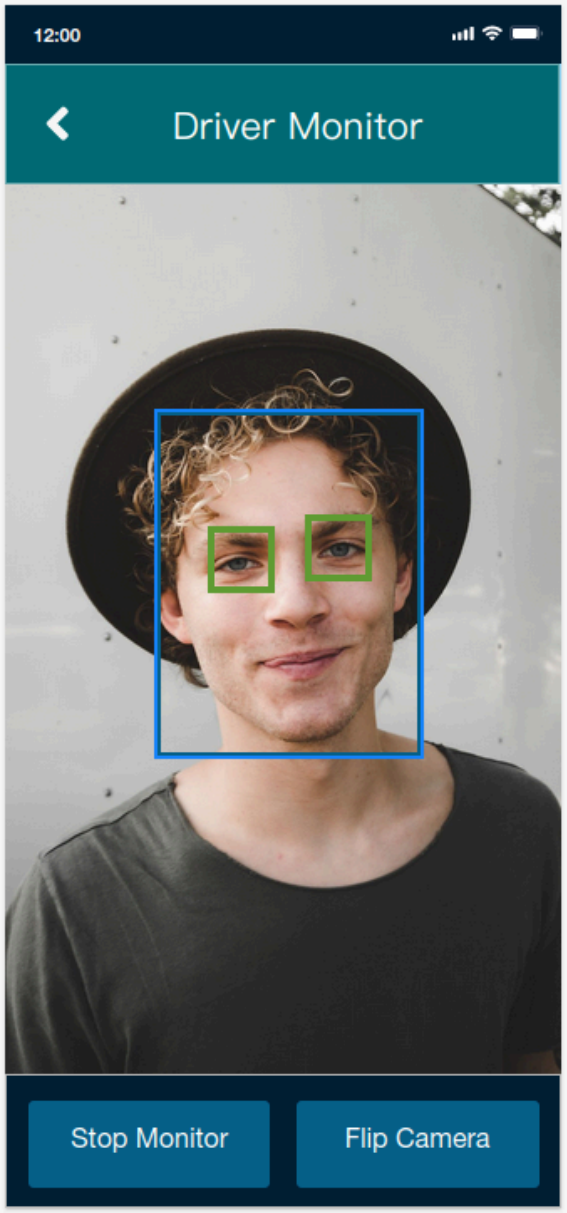
The chosen design scheme for the User Interface navigation centers around a simple and intuitive Stack design. The Homepage will serve as a central hub from which all other pages will seamlessly branch. This design philosophy seeks to prioritize simplicity to enhance usability, aligning with the overarching goal of making the application easy to use and understand. By adopting the Stack structure, the navigation is organized in a clear hierarchy, minimizing user cognitive load especially for new users of the app. This streamlined design seeks to enhance user engagement by making a navigation scheme which is as intuitive as possible. This design decision was made with the hope that this scheme will maximize the usability and encourage users to actually utilize the functionality which is of particular importance in this case due to the overall enhanced driver safety that the application offers to Drivers.

Screenshots

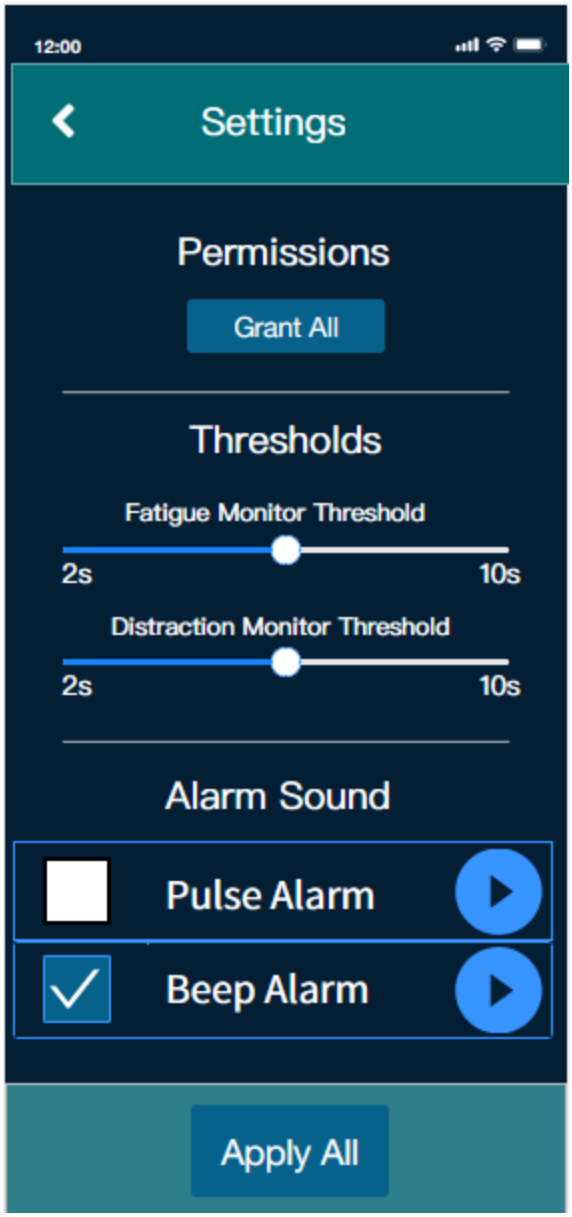
HomeScreen



Monitoring Screen



Settings



References

1. 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).
2. 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).
3. 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).
4. 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).
5. Google ML Kit (2023) *ML Kit | google for developers*, *Google*. Available at: https://developers.google.com/ml-kit (Accessed: 06 December 2023).
6. Expo (2023) *Facedetector*, *Expo Documentation*. Available at: https://docs.expo.dev/versions/latest/sdk/facedetector/ (Accessed: 06 December 2023).
7. Google ML Kit (2023a) *Face mesh detection | ML kit | google for developers*, *Google*. Available at: https://developers.google.com/ml-kit/vision/face-mesh-detection (Accessed: 06 December 2023).
8. Google ML Kit (2023b) *Face mesh detection concepts | ML kit | google for developers*, *Google*. Available at: https://developers.google.com/ml-kit/vision/face-mesh-detection/concepts (Accessed: 06 December 2023).
9. Tramasso, L. (2020) *Scholarship repository @ florida tech | Florida Institute of Technology ...* Available at: https://repository.fit.edu/cgi/viewcontent.cgi?article=1768&context=etd (Accessed: 06 December 2023).
10. Jayarathne, J. (2021) *Eye-blink detection*, *Medium*. Available at: https://jitharijayarathna.medium.com/eye-blink-detection-5d6854520217 (Accessed: 06 December 2023).
11. 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).
12. Hassan, U. and Rahman, A. (2022) *(PDF) automatic driver distraction detection using deep ... - researchgate*, *Intelligent Systems with Applications*. Available at: https://www.researchgate.net/publication/359741824_Automatic_Driver_Distraction_Detection_using_Deep_Convolutional_Neural_Networks (Accessed: 06 December 2023).
13. Chudziak, M. (2023) *@react-native-community/geolocation*, *React Native Community - Geolocation*. Available at: https://www.npmjs.com/package/@react-native-community/geolocation (Accessed: 06 December 2023).
14. 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).

15. Google Developers (2023) *Roads API - Speed Limits*, *Google*. Available at: https://developers.google.com/maps/documentation/roads/speed-limits (Accessed: 06 December 2023).
16. Mishra, A., Jain, V. and Sharma, N. (2018) *An analysis of convolutional neural networks for Image Classification*, *Procedia Computer Science*. Available at: https://www.sciencedirect.com/science/article/pii/S1877050918309335?ref=pdf_download&fr=RR-2&rr=821871e5f85656e4 (Accessed: 05 November 2023).
17. Arafa, M. *et al.* (2021) *(PDF) Image Classification based on CNN: A survey - researchgate*. Available at: https://www.researchgate.net/publication/355800126_Image_Classification_Based_On_CNN_A_Survey (Accessed: 05 November 2023).
18. Dharmaraj (2022) *Image classification and prediction using transfer learning*, *Medium*. Available at: https://medium.com/@draj0718/image-classification-and-prediction-using-transfer-learning-3cf2c736589d (Accessed: 05 November 2023).
19. GeeksforGeeks (2022) *Advantages and disadvantages of tensorflow*, *Advantages and Disadvantages of TensorFlow*. Available at: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-tensorflow/ (Accessed: 05 November 2023).
20. Sharma, M. (2022) *Advantages and disadvantages of tensorflow*, *Medium*. Available at: https://maheshjtp.medium.com/advantages-and-disadvantages-of-tensorflow-16e82af099 (Accessed: 05 November 2023).
21. Alajlan, N.N. and Ibrahim, D.M. (2023) *DDD tinyml: A tinyml-based driver drowsiness detection model using Deep Learning*, *MDPI*. Available at: https://www.mdpi.com/1424-8220/23/12/5696 (Accessed: 05 November 2023).
22. MathWorks (2023) *Deep Network designer*, *MobileNet-v2 convolutional neural network - MATLAB mobilenetv2 - MathWorks United Kingdom*. Available at: https://uk.mathworks.com/help/deeplearning/ref/mobilenetv2.html (Accessed: 05 November 2023).
23. Cherukuri, S. (2021) *Why developers like opencv*, *StackShare*. Available at: https://stackshare.io/opencv (Accessed: 05 November 2023).
24. You, F. *et al.* (2019) *A real-time driving drowsiness detection algorithm with ... - IEEE xplore*, *A Real-time Driving Drowsiness Detection Algorithm With Individual Differences Consideration*. Available at: https://ieeexplore.ieee.org/abstract/document/8930504 (Accessed: 05 November 2023).
25. Qayyum, W., Ahmad, A. and Aljuhni, A. (2022) *Evaluation of googlenet, Mobilenetv2, and Inceptionv3, pre-trained ...*, *Evaluation of GoogLenet, Mobilenetv2, and Inceptionv3, pre-trained convolutional neural networks for detection and classification of concrete crack images*. Available at: https://www.researchgate.net/publication/359615441_Evaluation_of_GoogLenet_Mobilenetv2_and_Inceptionv3_pre-trained_convolutional_neural_networks_for_detection_and_classification_of_concrete_crack_images (Accessed: 05 November 2023).
26. Fusek, R. (2021) *MRL Eye Dataset*, *MRL*. Available at: http://mrl.cs.vsb.cz/eyedataset (Accessed: 05 November 2023).

27. Keras Team (2015) *Keras Documentation: Mobilenet, mobilenetv2, and MobileNetV3*, *Keras*. Available at: https://keras.io/api/applications/mobilenet/ (Accessed: 05 November 2023).
28. 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).
29. Clark, J. (2022) *Firebase Advantages and disadvantages*, *Back4App Blog*. Available at: https://blog.back4app.com/firebase-advantages-and-disadvantages/ (Accessed: 05 November 2023).
30. Microsoft (2023) *React native tools - visual studio marketplace*, *Marketplace*. Available at: https://marketplace.visualstudio.com/items?itemName=msjsdiag.vscode-react-native (Accessed: 08 December 2023).
31. Devtop (2022) *Developing in react native*, *Devtop*. Available at: https://devtop.io/en/development-in-react-native/ (Accessed: 05 November 2023).
32. PngWing (2023) *PNGWING - exclusive PNG Design images*. Available at: https://www.pngwing.com/en/free-png-hjxmr (Accessed: 05 November 2023).
33. PngEgg (2023) *Firebase Database Mobile backend as a service push technology ... - pngegg*. Available at: https://www.pngegg.com/en/png-iulud (Accessed: 05 November 2023).
34. Wikipedia (2023) *Tensorflow*, *Wikipedia*. Available at: https://en.wikipedia.org/wiki/TensorFlow (Accessed: 05 November 2023).
35. AssetSale (2023) *MobileNet v2 - Realtime object classifier: Unity assetstore price down information*, アセットまとめ. Available at: https://assetsale.herokuapp.com/en/contents/83910 (Accessed: 05 November 2023).
36. OpenCV (2020) *Media kit*, *OpenCV - Media Kit*. Available at: https://opencv.org/resources/media-kit/ (Accessed: 05 November 2023).
37. PngFind (2019) *Git's Rebase Command is a common source of fear and - git source control, HD PNG download(1920x660) - pngfind*, *PngFind.com*. Available at: https://www.pngfind.com/mpng/hTToJhT_gits-rebase-command-is-a-common-source-of/ (Accessed: 05 November 2023).
38. Icon Mafia (2022) *Free github logo icon - download in flat style*, *IconScout*. Available at: https://iconscout.com/free-icon/github-169 (Accessed: 05 November 2023).
39. PngWing (2022) *Free download | visual studio code, HD, logo, png | pngwing*. Available at: https://www.pngwing.com/en/free-png-aztoa/download (Accessed: 05 November 2023).
40. Studio, N. (2023) *Millions of PNG images, backgrounds and vectors for free download*, *Pngtree*. Available at: https://www.pngegg.com/en/png-woqvp#google_vignette (Accessed: 08 December 2023).
41. NoHat (2019) *Free your browser bookmarks, free your disk space.*, *Free: Siks/cbs Datacamp Spark Tutorial Notebook - Jupyter Notebook Icon*. Available at: https://nohat.cc/f/siks-cbs-datacamp-spark-tutorial-notebook-jupyter-notebook-icon/m2H7K9d3N4Z5i8G6-201907240851.html (Accessed: 08 December 2023).